

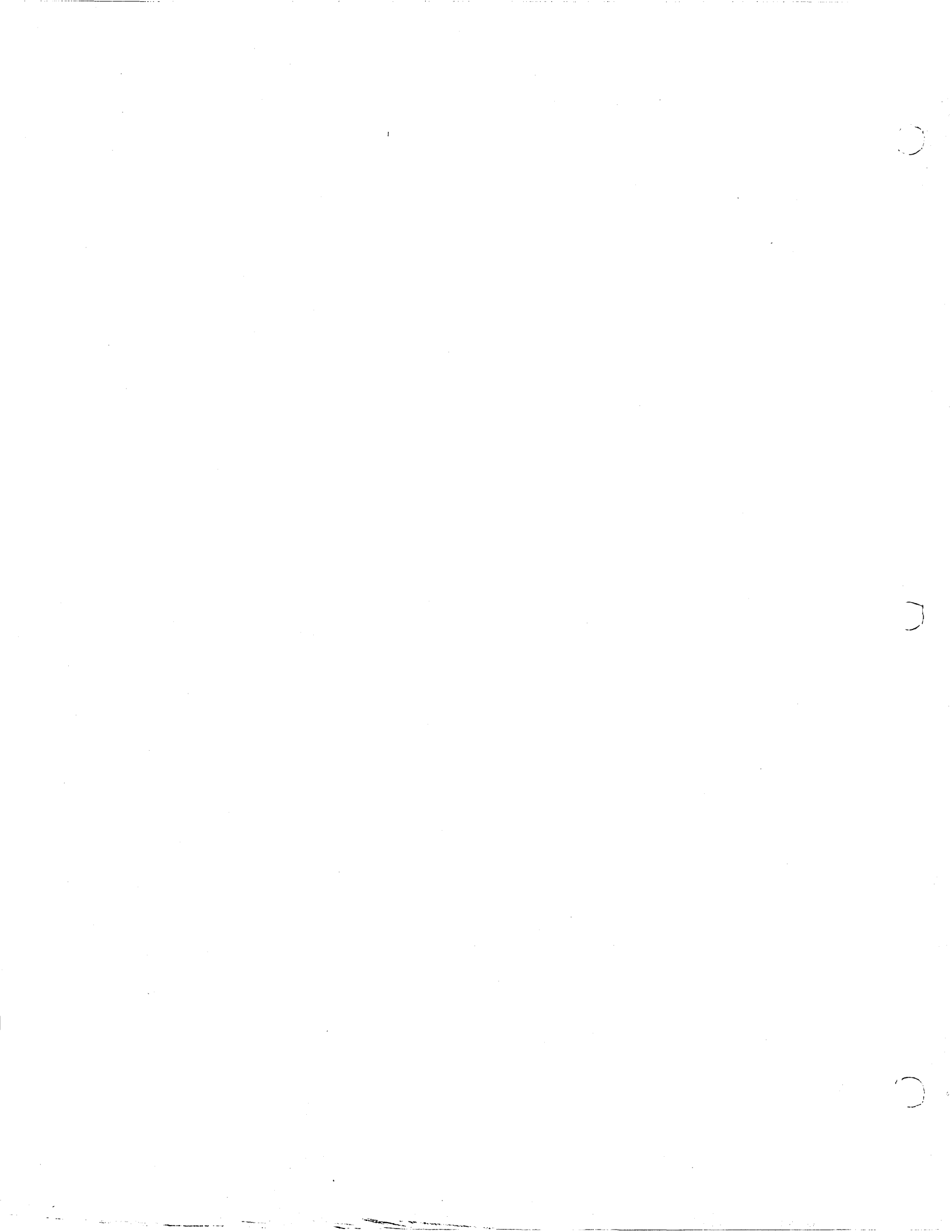
INTERNAL MAINTENANCE SPECIFICATIONS

1700 MASS STORAGE FORTRAN 3.1

under MSOS 4.0

© COPYRIGHT CONTROL DATA CORP. 1972

Contained herein are software products  
copyrighted by Control Data Corporation.  
A reproduction of the copyright notice must  
appear on all complete or partial copies.

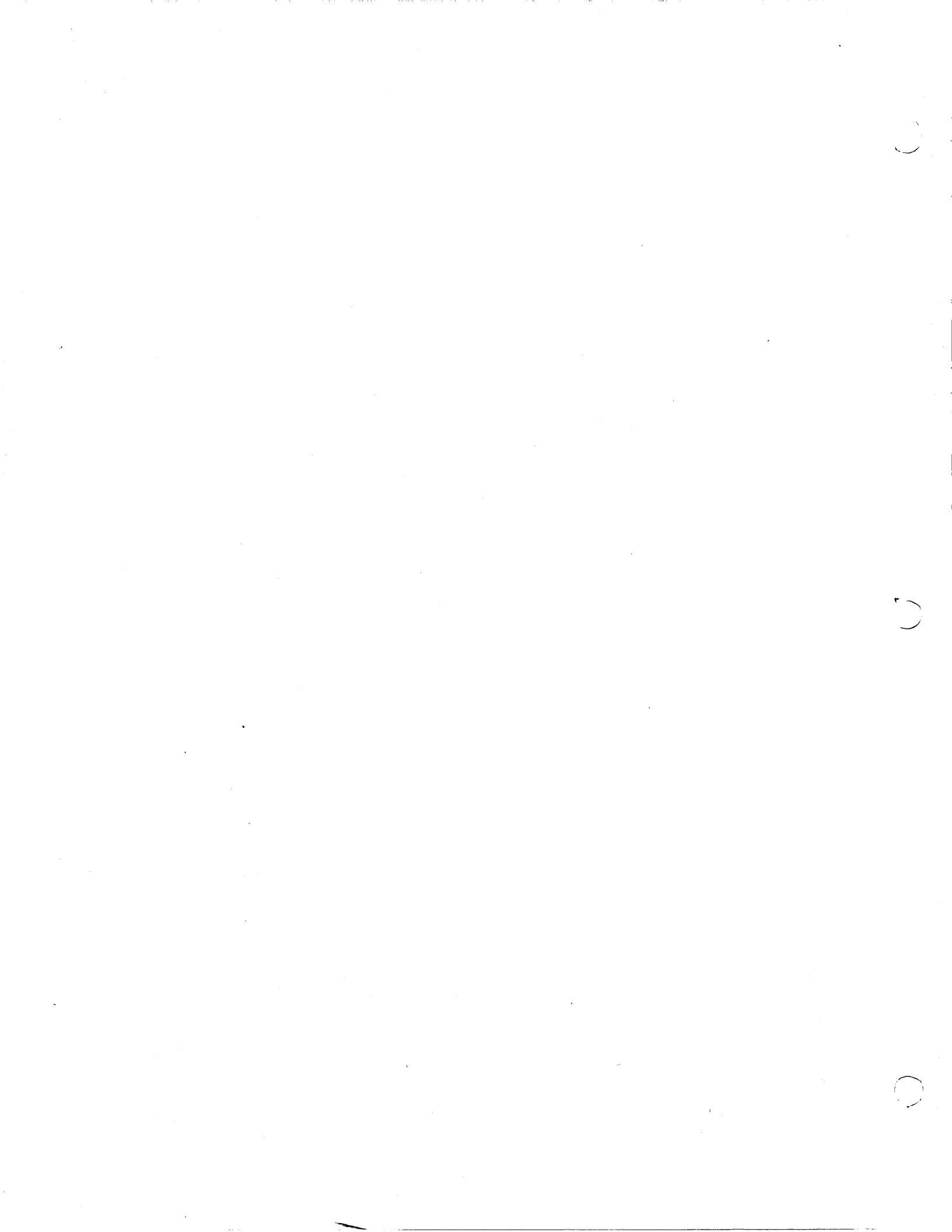




DOCUMENT CLASS IMS PAGE NO. ii  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

T A B L E O F C O N T E N T S

Chapter One	General Description
Chapter Two	Phase A
Chapter Four	Phase B
Chapter Five	Phases C, D and E
Chapter Six	Object Time I/O Routines
Chapter Seven	Tables
Chapter Eight	Object Time Arithmetic Routines {Single Precision}
Chapter Nine	Floating Point Package {Single Precision}
Chapter Ten	Object Time Intrinsic Functions {Single Precision}
Chapter Eleven	Object Time Arithmetic Routines {Double Precision}
Chapter Twelve	Floating Point Package {Double Precision}
Chapter Thirteen	Object Time Intrinsic Functions {Double Precision}



CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS \_\_\_\_\_ IMS \_\_\_\_\_ PAGE NO. \_\_\_\_\_ L-1  
PRODUCT NAME \_\_\_\_\_ 1700 MASS STORAGE FORTRAN \_\_\_\_\_  
PRODUCT MODEL NO. \_\_\_\_\_ C005 V.2.0 \_\_\_\_\_ MACHINE SERIES \_\_\_\_\_ 1700

CHAPTER 1

TABLE OF CONTENTS

1:0	General Description
1.1	Summary
1.2	Phase A Tasks
1.3	Phase B Tasks
1.4	Phase C,D,E Tasks
1.5	Partial Compilation In Special Cases
1.6	Local Optimizations

DOCUMENT CLASSIFICATION \_\_\_\_\_ IMS \_\_\_\_\_ PAGE NO. \_\_\_\_\_ 1-2  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005 V.2.0 \_\_\_\_\_ MACHINE SERIES 1700

## 1.0 GENERAL DESCRIPTION OF COMPILER

### 1.1 Summary

1700 Mass Storage Fortran is oriented toward the users of medium-sized 1700's. Input is from punched cards or paper tape; output is on a printer or paper tape punch and is formatted accordingly. Card-image input and line-image output may be on magnetic tape. Intermediate scratch files are on the disc. 1700 Mass Storage Fortran is released in versions for 16K or 20K machines, and runs under the Mass Storage Operating System.

The compiler consists of four passes over the source code or its equivalent, accomplished in four phases, called A, B, C, and D/E. (The fourth pass is performed by either Phase D or Phase E, according to whether the user wants an assembly-language listing output.) Each phase consists of a "root" which is core-resident throughout the phase, and zero or more "local" subroutine groups which share the same core area and are read from disc as needed.

### 1.2 Phase A Tasks

These tasks read the source input, convert it to statements expressed in an internal code, and assign a statement number to the statement.

The statements are syntax checked and itemized into an output file. The loop structure table (LOOPT), equivalence table (IEQV), specification table (ISTAB), and much of the symbol table (SYMTAB) are built.

### 1.3 Phase B Tasks

PHASE B reads the output of PHASE A and generates pseudo-code from it. Pseudo code is similar to assembler input except that the index to be used in an indexed instruction is not specified and the addressing mode is not specified.

DOCUMENT CLASS IMS PAGE NO. 1-3  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005 V.2.0 MACHINE SERIES 1700

#### 1.4 Phase C, D, E Tasks

Phases C and D/E are a two pass assembler. The output from Phase B is read. Index registers are optimally assigned. One word relative addressing is maximized. Relocatable binary output with futures and an assembly listing are produced.

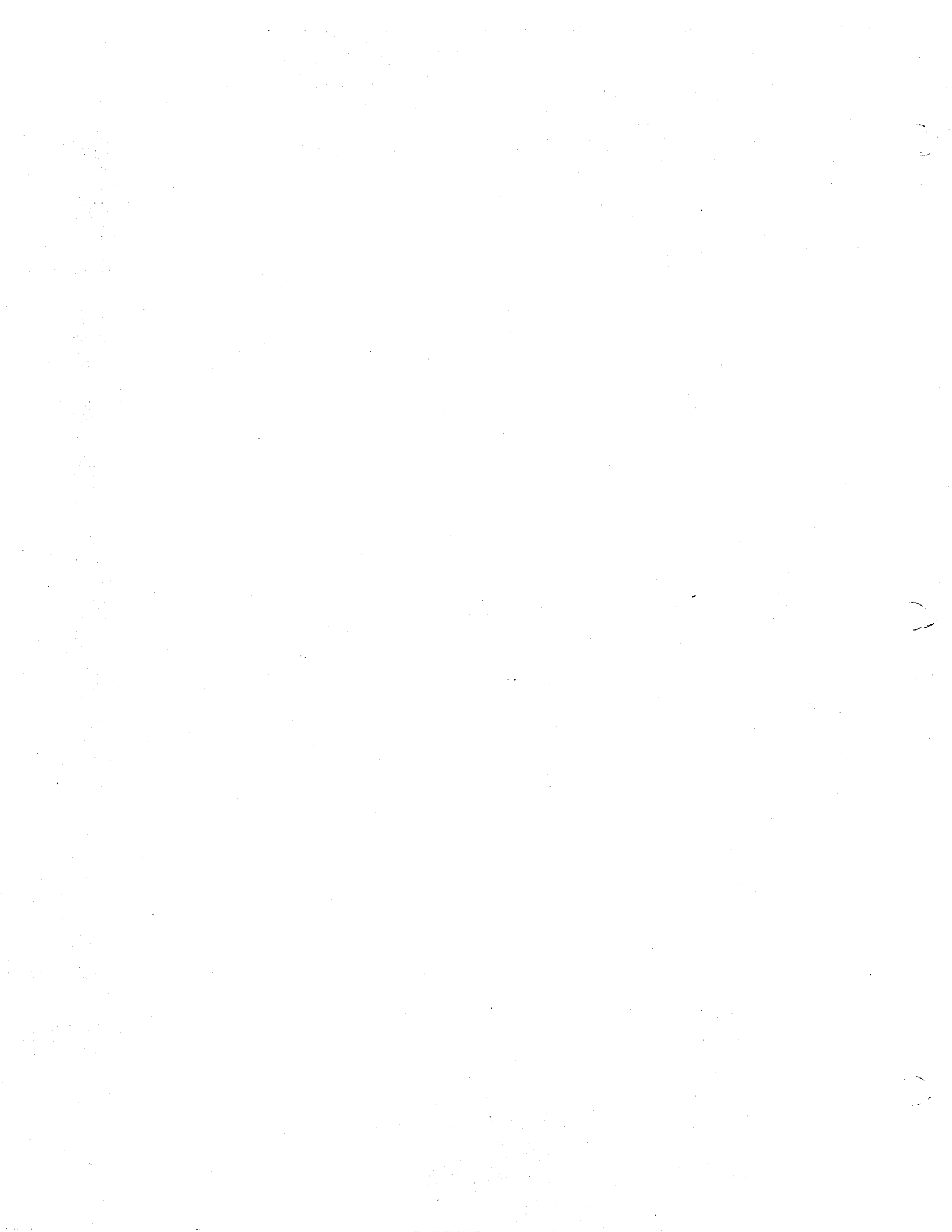
#### 1.5 Partial Compilation In Special Cases

The number of phases actually executed depends on the options selected and success of the compilation. If only a source listing is requested, only phase A is executed; this provides a complete syntax check without producing machine code which is not wanted. Similarly, only phase A will be executed if it detected a fatal error in the program; a fatal error, by definition, precludes the generation of correct machine code, so there is no reason to generate the code. A third abbreviation of the compilation process occurs with stacked compilations when the 'X' option is selected but the 'A', 'M', and 'P' options are not. Detection of a fatal error in a program clears the 'X' option for the stack, and therefore all following programs will be processed only by phase A.

#### 1.6 Local Optimizations

Many local optimizations of the code generated have been implemented.

1. Index registers are optimally assigned.
2. Relative addressing is used where possible.
3. Storage is allocated to maximize relative addressing. For example, some arrays are put into the middle of code.
4. All simple FORTRAN provided functions are inserted in-line (e.g., IABS or AND).
5. A comprehensive analysis of IF statements is made. In the code generated there is cognizance of a transfer from the IF to the label of the next statement and if this statement is a GO TO.



CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 2-1  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005 V.2.0 MACHINE SERIES 1700

- 2.0 Phase A Tasks and Subprograms
  - 2.1 Summary
  - 2.2 Overlay Programs
    - 2.2.1 Flow of Control
      - 2.2.1.1 Versions of GO routine
      - 2.2.1.2 Versions of LOCAL routine

## CONTROL DATA CORPORATION

Arden Hills Development

DIVISION

2-2

DOCUMENT CLASS IMS PAGE NO. \_\_\_\_\_  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005x3.0 A/B-E006x4.0 MACHINE SERIES 1700

## 2.2.2 PHASEA Routine and Related Subroutines

2.2.2.1 Subroutine GNST  
 2.2.2.2 Subroutine TYPE  
 2.2.2.3 Subroutine PEQVS  
 2.2.2.4 Subroutine PLABEL  
 2.2.2.4.1 Subroutine RDLABL  
 2.2.2.5 Subroutine OPTIONS  
 2.2.2.6 Subroutine OUTENT  
 2.2.3.2 Subroutine ENDDO  
 2.2.3.3 Subroutine SAVEID

## 2.3 General Pass Routines

2.3.1.1 Subroutine SYMBOL  
 2.3.1.2 Subroutine STORE  
 2.3.1.3 Subroutine GETSYM  
 2.3.1.4 Subroutine CNVT  
 2.3.1.5 Subroutine SYMSCN

## 2.3.2 Field Processing Routines

2.3.2.1 Subroutine GETF  
 2.3.2.2 Subroutine GETC  
 2.3.2.3 Subroutine GPUT  
 2.3.2.4 Function IGETCF  
 2.3.2.5 Subroutine STCHAR

## 2.3.3 Special Field Analyzing Routines

2.3.3.1 Subroutine CONSUB  
 2.3.3.2 Subroutine CFIVOC  
 2.3.3.3 Subroutine CKIVC  
 2.3.3.4 Subroutine CKNAME

## 2.3.4 Diagnostic Handling Routine

2.3.4.1 Subroutine DIAG  
 2.3.4.2 Subroutine CONV  
 2.3.4.3 Subroutine PACK  
 2.3.4.4 Subroutine PUNT  
 2.3.4.5 Subroutine PRNTNM  
 2.3.4.6 Subroutine IOPR

## 2.4 Statement Specific Phase A Routines

2.4.1 DIMENSION Statements - Subroutine DIMPR  
 2.4.2 COMMON Statements - Subroutine COMNPR  
 2.4.3 TYPE Statements - Subroutine TYPEPR



CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS. IMS PAGE NO. 2-3  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005 V.2.0 MACHINE SERIES 1700

- 2.4.4 BYTE, EQUIVALENCE Statements - Subroutine BYEQPR
- 2.4.5 SUBROUTINE, FUNCTION Statements - Subroutine SUBPPR
- 2.4.6 DATA Statements - Subroutine DATAPR
- 2.4.7 FORMAT Statements
  - 2.4.7.1 Subroutine CHECKF
  - 2.4.7.2 Subroutine FORK
  - 2.4.7.3 Subroutine FGETC
- 2.4.8 ASSEM Statements - Subroutine ASEMPR
- 2.4.9 ASSIGN Statements - Subroutine ASGNPR
- 2.4.10 EXTERNAL & RELATIVE Statements - Subroutine EXRLPR
- 2.4.11 REWIND, ENDFILE, BACKSPACE Statements - Subroutine ERBPR
- 2.4.12 READ, WRITE Statements - Subroutine IOSPR
- 2.4.13 DO Statements - Subroutine BDOPR

3.3 Special Subroutines for Common Allocation

- 3.3.1 Subroutine ARAYSZ
- 3.3.2 Subroutine CPLOOP

3.4 Arithmetic Expression Processors

- 3.4.1 ARITH
- 3.4.2 SUBSCR
- 3.4.3 TREE
- 3.4.4 MODMXR

DOCUMENT CLASS IMS PAGE NO. 2-4  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. CO05\*3.1 A/B MACHINE SERIES 1700

2.0 PHASE A TASKS AND SUBPROGRAMS

2.1 Summary

The driver {FTN} is entered by the Job Processor control statements. The driver indirectly calls the Phase A main program {PHASEA}. The source lines are read in formatted ASCII, converted to statements in an internal code, given a sequential statement number, and stored in the buffer ISORS {routine GNST}. The statement type is computed {TYPE}. The statements are sent to their appropriate processors, syntax checked, itemized into an output buffer {see 7.5}, and output {routine OUTENT}. Upon encountering the first executable statement {or an END statement}, the EQUIVALENCE statements, which are in a tabulated form in core, are processed and the IEQV table is formed {routine PEQVS}. Control is returned to FTN through the routine GOA, which either proceeds to read in and execute Phase B, or restarts Phase A to read the next source program.

## CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS \_\_\_\_\_ IMS \_\_\_\_\_ PAGE NO. \_\_\_\_\_ 2-5  
PRODUCT NAME \_\_\_\_\_ 1700 MASS STORAGE FORTRAN \_\_\_\_\_  
PRODUCT MODEL NO. \_\_\_\_\_ C005 V.2.0 \_\_\_\_\_ MACHINE SERIES \_\_\_\_\_ 1700 \_\_\_\_\_

## 2.2 OVERLAY PROGRAMS

Each phase consists of a main section, or 'root', which is resident in core during the entire execution of its phase, and, if necessary, a number of subroutine groups, or 'local files', which are read in from the disc as needed. The local files overlay one another in core and are so programmed that they are 'read-only'... no program in a local file contains variables which are carried over from one call to the next, so the file may be overlaid by another and recalled later without being saved in the meantime. (Some programs which are in local files do have variables which must be preserved; these variables are placed in blank common, which is not overlaid during a phase.) In effect, the local files are an extension of the phase concept, except that a program in one local file can, with certain restrictions, call one in another local file of the same phase.

## 2.2.1 FLOW OF CONTROL

The compiler is entered by calling the relocatable program FTN from the system library. FTN in turn calls the file FORTAL from the library and transfers to the first instruction of the routine GOA. From this point on control is transferred by normal calling sequences from GOA to the main routine of phase A, PHASEA, and to various subroutines. When it is necessary to use a routine in a different local file, the subroutine LOCAL is called, giving file number and entry number within file. LOCAL reads the overlay portion of the specified file into the overlay area. The overlay portion of each file begins with a list of the addresses of its entry points; LOCAL transfers to the n'th address in the list for the new file. When the routine thus called returns to LOCAL, LOCAL reads the original calling file back in and returns control to the program which originally called it. (In the 20K compiler, most calls are from PHASEA, in the root, to local subroutines. In this case, if the desired subroutine is already in core, LOCAL is bypassed; if not, the routine

DOCUMENT CLASS IMS PAGE NO. 2-6  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. COO5 V.2.0 MACHINE SERIES 1700

is read and called by LOCAL, but when it returns, its file is left in core. Inter-local calls... between IOSPR and ARITH, and between ARITH and MODMXR... are handled as in 16K.) The local-file mechanism is generally transparent to the compiler programs.

At the end of phase A, the main routine (PHASEA) returns to GOA, which in turn returns to FTN. FTN reads in the first file of the next phase (FORTB1) and jumps to GOB. There are similar linkages from phase B to C, and from C to D or E.

In the event that compilation should be aborted after phases A or B (see section 1.5), a call is made to entry SKIPIT in subroutine GOA or GOB. SKIPIT causes an alternate return from GO to FTN, such that FTN re-initializes itself just as though phase D/E had just been completed, and reads FORTAL to start compiling the next program (if any) on the input stack.

The choice between sub-phases D and E is based on the user-selected options; D is used unless an assembly-language listing is requested (A or M option). In the 16K compiler, FTN always reads FORTD1 at the start of phase D/E, and the phase D/E local routine decides whether to use the D or E local routines. (The D and E roots are identical.) In the 20K compiler, GOC tests the options and makes an alternate return to FTN such that FORTD1 or FORTEL is read.

#### 2.2.1.0.1 FLOWCHART OF PROGRAM FTN

#### 2.2.1.1 VERSIONS OF 'GO' ROUTINE

In each phase, the subroutine GO is called by FTN; it calls the initialization entry of the I/O package, then the main routine of the phase; then performs some I/O clean-up and returns to FTN. In addition to this, each GO has special functions which will be detailed below.

## CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 2-7  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. COO5 V.2.0 MACHINE SERIES 1700

GOA checks a flag set by FTN, to determine whether this is the start of a stacked compilation; if so, it STATUS'es the list device and sets a flag telling whether the device can accept standard carriage control; and it calls OPANAL to request options from the typewriter. If this is not the start of a stack, the first source statement is read and checked for being a MON; this saves rebuilding the phase A local files just to read a MON. (The statement, if not a MON, is left in the input work area for later processing by GNST.) Whether or not this is the start of a stack, the phase A local files must then be prepared. (A flag is tested to determine whether the files are still intact in the disc scratch area, as they would be if phase B had not been called; if so, LOCALS is called at entry LOCLZ2, rather than LOCL12, to just rebuild tables in core rather than rebuilding the scratch area.) PHASEA is then called. It will either return normally, or call SKIPIT; GOA's return address is adjusted accordingly. The WRITE entry in IOPR is called to force out the last scratch page, and GOA returns to FTN.

GOB clears the flag which indicates that the phase A local files are intact in the scratch disk area; interchanges the scratch file numbers so that the phase A output file becomes the phase B input file; in the 2.0A version, causes the phase B local files to be generated; calls phase B; and upon return, finishes the phase as GOA did.

GOC, in the 2.0A version, is similar to GOB. In the 2.0B version, after return from PHASEC (There is no SKIPIT entry), it chooses alternate returns to FTN depending on whether or not the A and M options are selected.

GOOD and GOE are identical to one another in function and are similar to GOB. However, after return from the main routine (PHASE6), there is no scratch page to be output; and, if the list device is a magnetic tape, an end of file is written.

DOCUMENT CLASS \_\_\_\_\_ IMS \_\_\_\_\_ PAGE NO. \_\_\_\_\_ 2-8  
PRODUCT NAME \_\_\_\_\_ 1700 MASS STORAGE FORTRAN \_\_\_\_\_  
PRODUCT MODEL NO. \_\_\_\_\_ C005 V.2.0 \_\_\_\_\_ MACHINE SERIES \_\_\_\_\_ 1700 \_\_\_\_\_

The load-and-go file is built at the start of the scratch disk area, if the X option was specified, by the phase D/E routine NPUNCH. The compiler's scratch area "floats" on the growing load-and-go file; at the start of each compilation, GOA saves the address of the top of the file from the communication area in low core, and at the end of each compilation GOOD or GOE adds to that value the length of the current load-and-go output. (Also see discussion of disk allocation in description of subroutine IOPR.)

#### 2.2.1.1.1 FLOWCHARTS OF SUBROUTINE GO

#### 2.2.1.2 VERSIONS OF 'LOCAL' ROUTINE

References to non-resident subroutines (in phases A, B, D and E of the 2.0A compiler and phase A of the 2.0B pass through the subroutine LOCAL. Subroutines to be placed in "local files" are written so that any variables which must be preserved between calls are placed in blank common, so that the routine can be overlaid without first being saved. The non-resident routines are then grouped in files on scratch disk, and references to them are intercepted by dummy routines which call LOCAL. That routine reads the file containing the called routine, executes that routine, and returns control to the dummy's caller.

At the start of a phase containing local subroutines, the entry LOCLIZ is called; it transfers the local files from the library to scratch disk, so that they can be referenced directly rather than via GTFIL. Each file so transferred contains: (1) one word containing the address of ENDLOC, a dummy routine loaded at the end of each file so that the file length can be computed; (2) a list of words containing the address of each "entry point" of that file; and (3) the absolutized subroutines of the file.

A call to a local subroutine not presently in core goes through a dummy routine and becomes a

DOCUMENT CLASS IMS PAGE NO. 2-9  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

call to LOCAL with a file number and entry number. The present state of the overlay system is saved in a number of pushdown stacks. The desired file is read into the overlay area and the desired entry point is called. If that program causes a LOCAL call, the state saving and file reading is repeated; the number of levels of call allowed is limited by the size of the pushdown stacks. Upon return from a local file, the stacks are popped up one level. The calling file is read back in {except, in 2.0B, if the call was from the root}, and the calling program is returned to.

A "flag" parameter is provided in LOCAL and has three quite different uses in different versions of the routine.

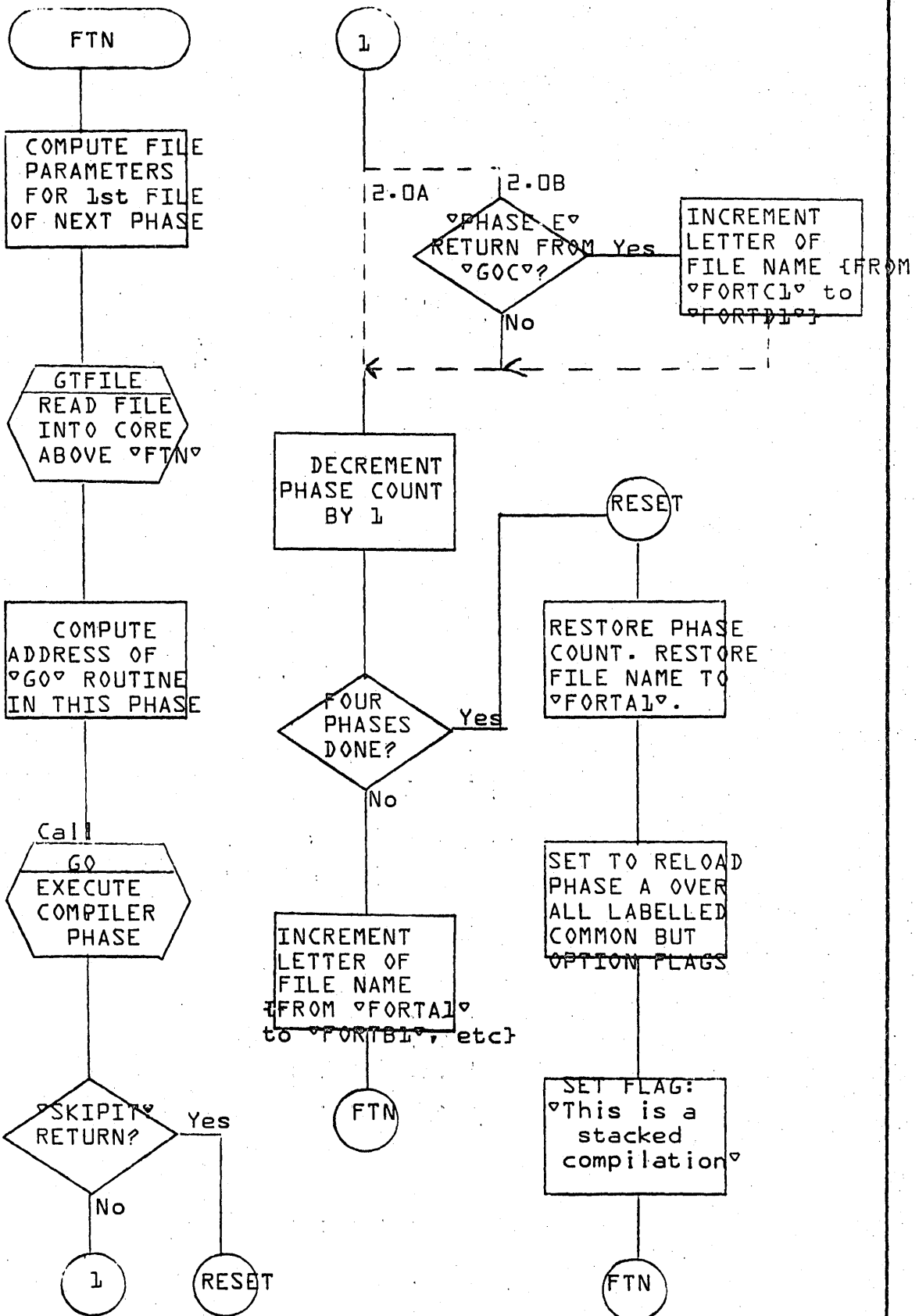
In 2.0A phase A, FLAG=1 means that the calling file has finished execution, and need not be read back in; for example, PHASEA in file A1 calls ARITH in file A2; just before returning, ARITH calls TREE in file A3; and just before returning, TREE may call MODMXR in file A4. There is no reason to reload TREE and ARITH after MODMXR is done, so the stacks are popped up three levels and control goes directly from MODMXR to PHASEA.

In 2.0A phase B, parameters must be passed to local routines; here, FLAG=1 means that the following word is a parameter address to be passed on.

In 2.0B phase A, almost all calls are from PHASEA in the root, to local routines; there FLAG=1 signals the case of a local-to-local call, where a different algorithm is needed to find the proper return address.

In the 3.0A and 3.1A compiler, routines STCHAR and GETC must be loaded at the same point in files FORTA1 and FORTA2 in order to have externals in SAVEID patched properly for both files. This is accomplished by a small BSS in DUMYA2 which is followed immediately by ERBPR to equal the length of DUMYA1.

#### 2.2.1.2.1 Flowcharts of the "Local" Subroutines



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE   
FLCCHART   
DECISION TABLE   
OTHER

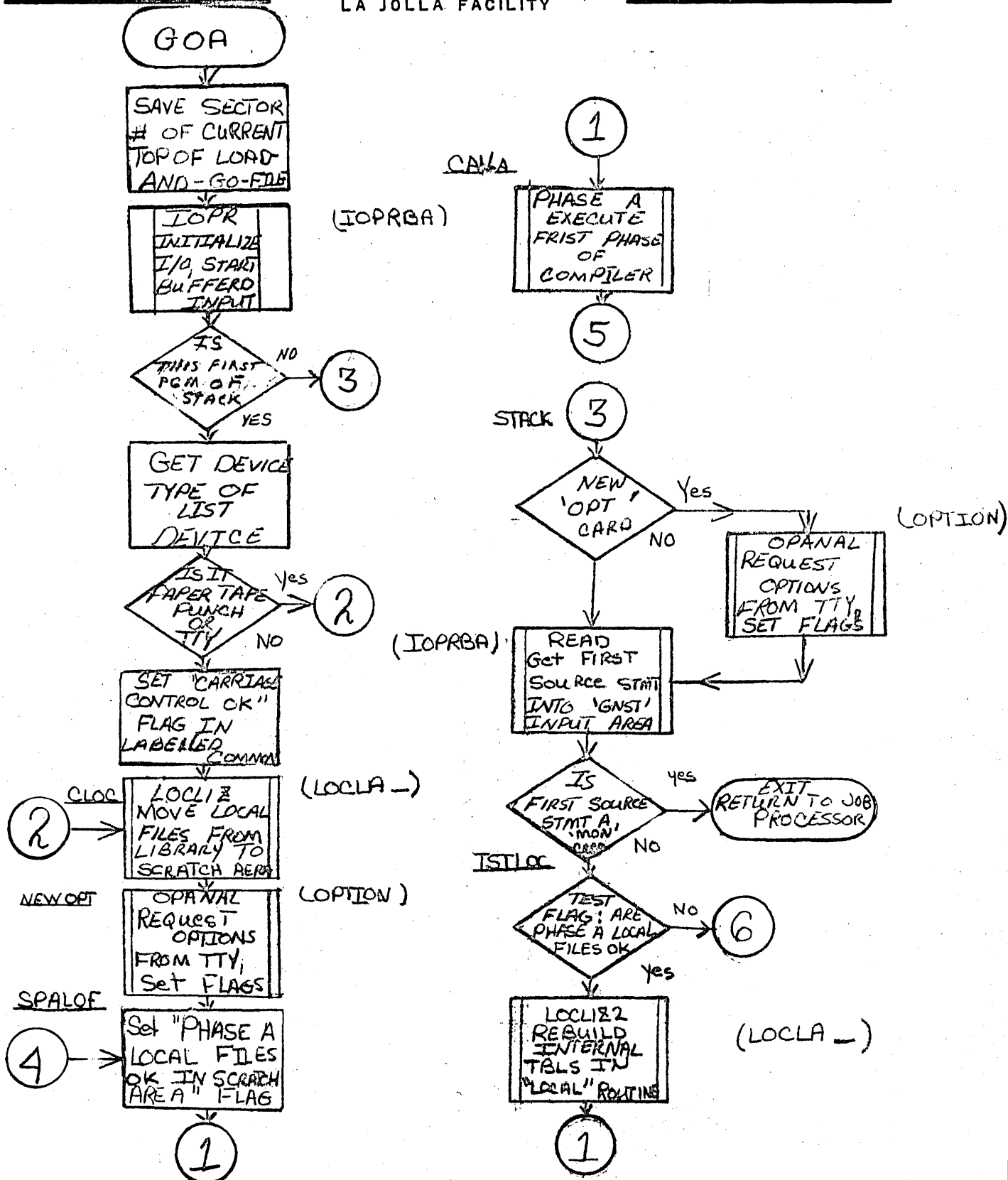
DOCUMENT CLASS	IMS	MACH. TYPE	J700	PROJECT NO.	
DOCUMENT TITLE	MASS STORAGE FORTRAN			PROJECT MGR.	
NUMBER		PAGE / OF	/ /	PROJECT NAME	
ISSUE DATE		TASK NO.		TASK NAME	
DATE					
REV.		APPROVED		DATE	

1  
2  
3  
4  
5



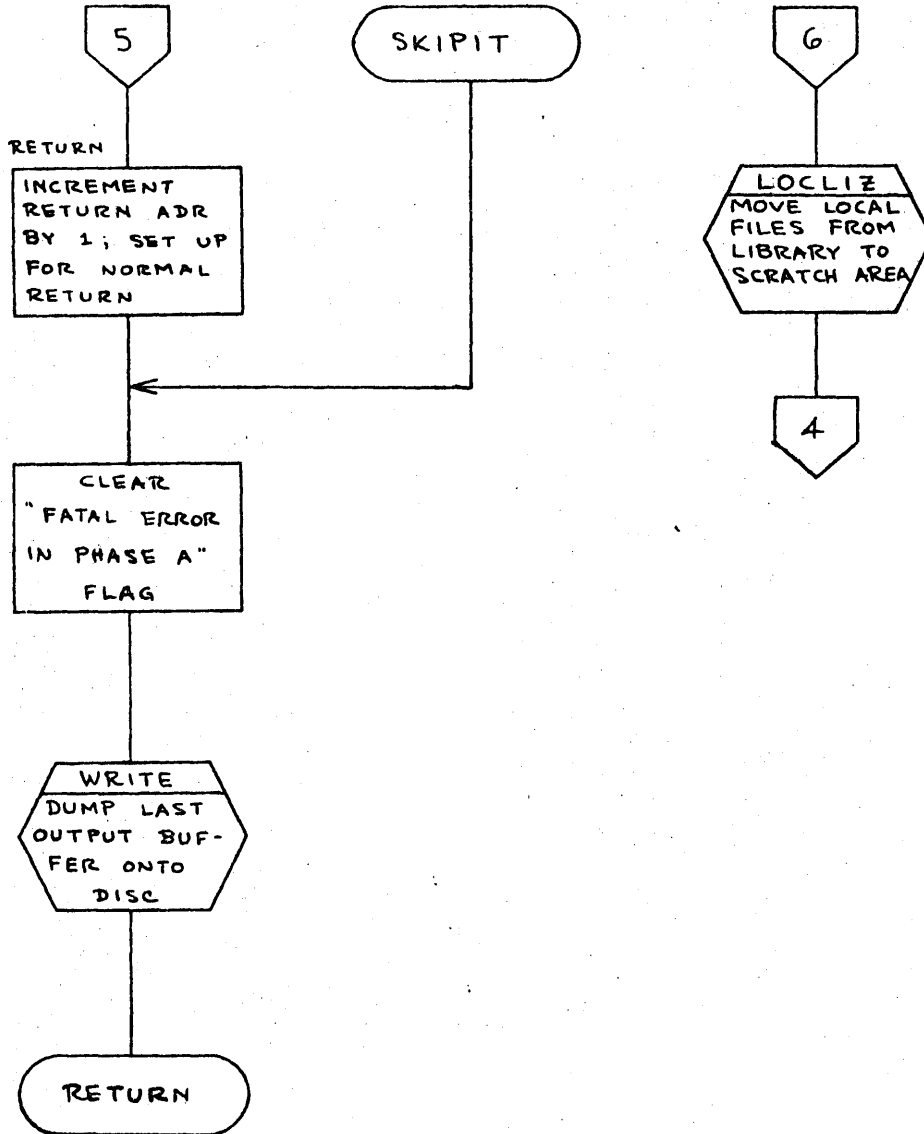


LA JOLLA FACILITY



TITLE		GOA		DRG. NO.
DRAWN BY		PROJ.	DATE	REVISION
CSD 203				SHEET 1 OF 2

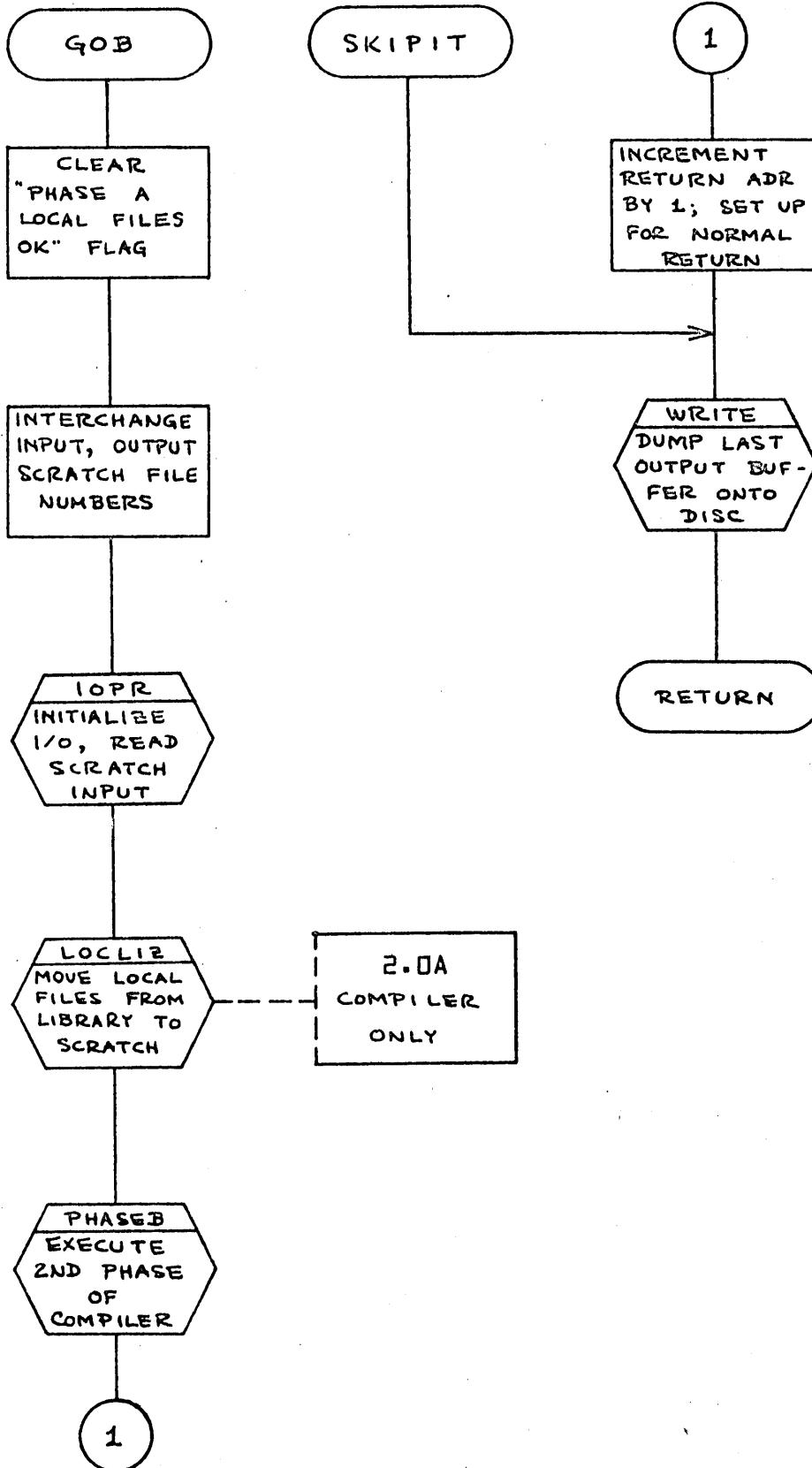
DOCUMENT CLASS IMS PAGE NO. 2-12  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005 V.2.0 MACHINE SERIES 1700

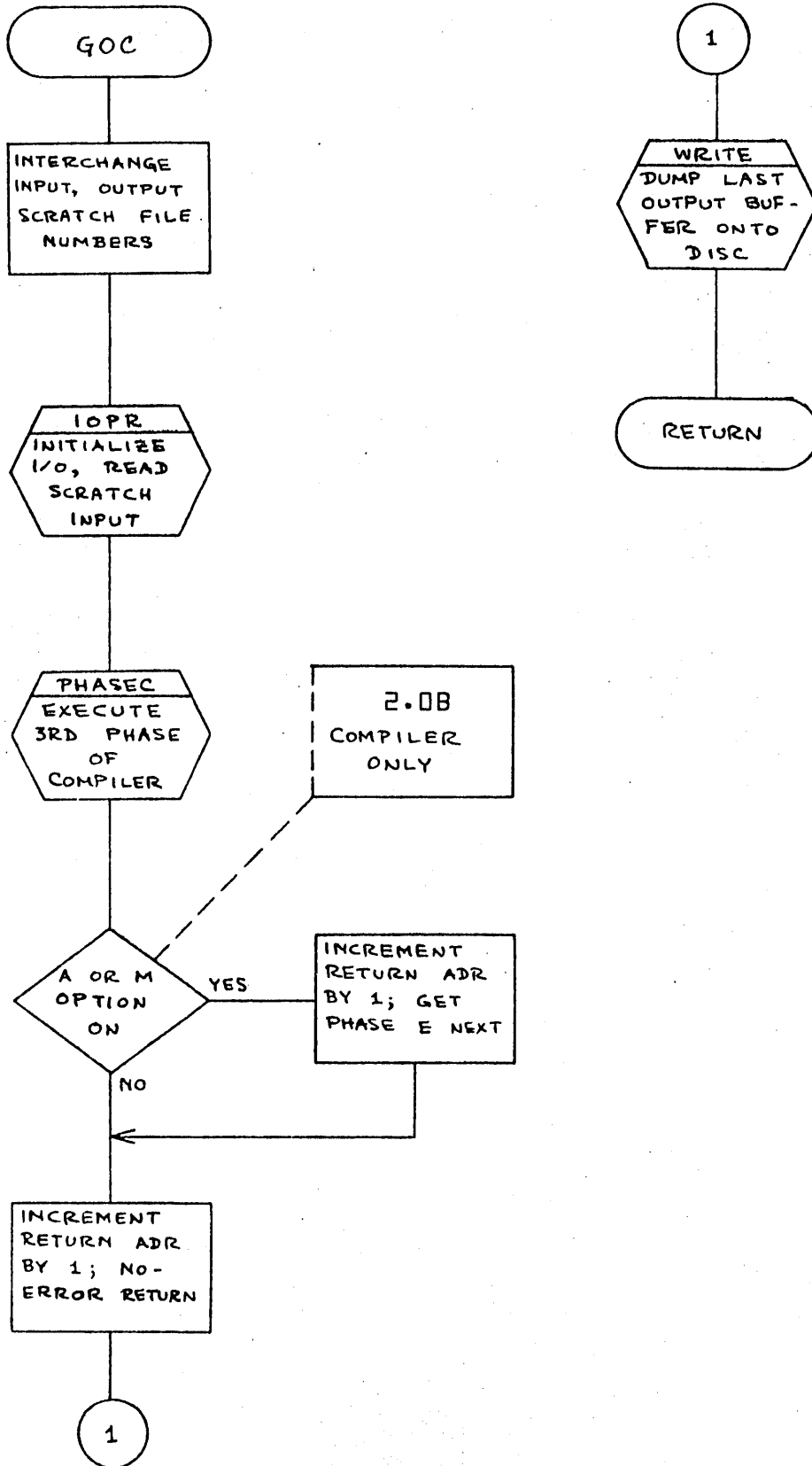


CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 2-13  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005 V.2.0 MACHINE SERIES 1700





DOCUMENT CLASS

IMS

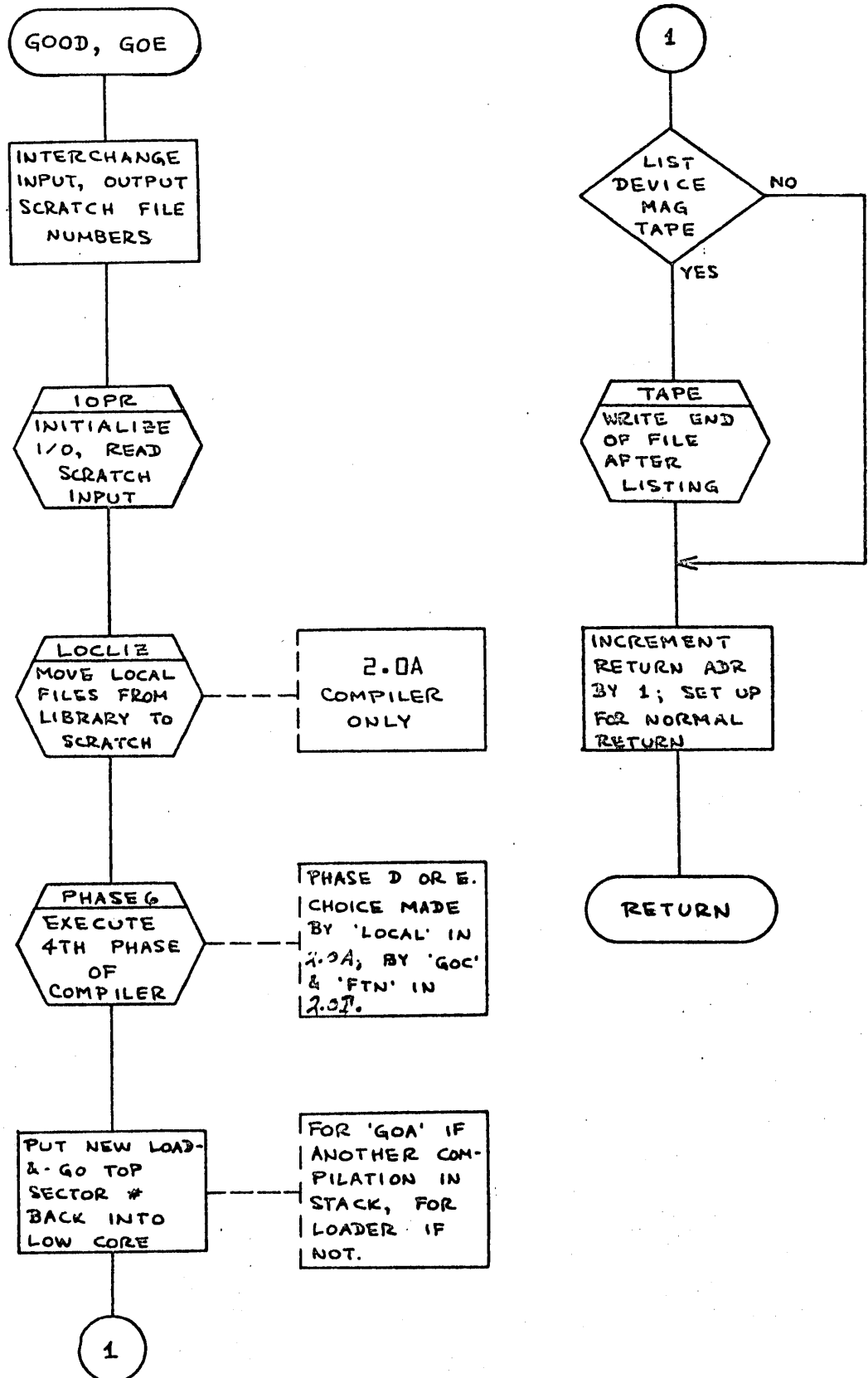
PAGE NO.

2-15

PRODUCT NAME 1700 MASS STORAGE FORTRAN

PRODUCT MODEL NO. C005 V.2.0

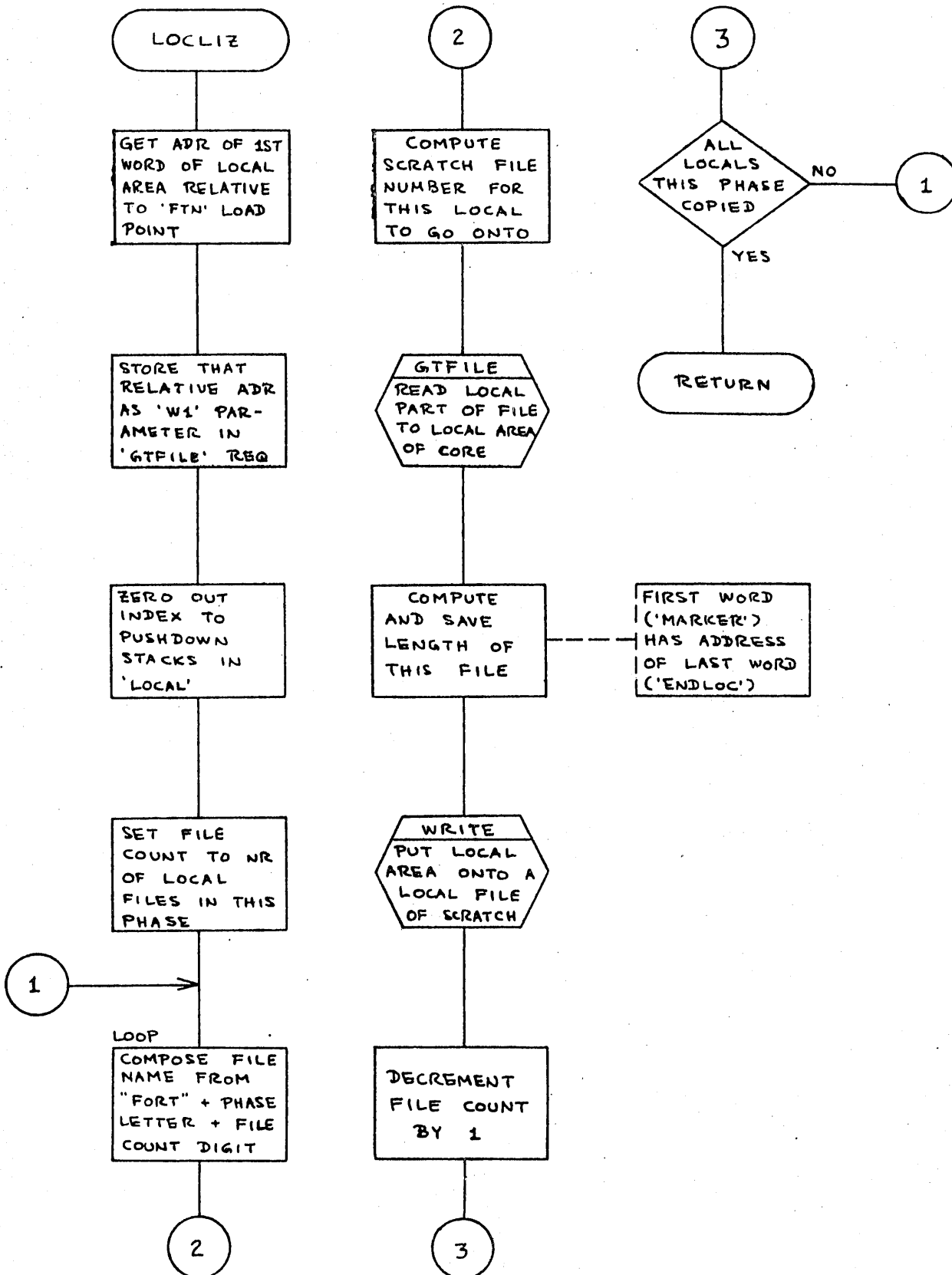
MACHINE SERIES 1700



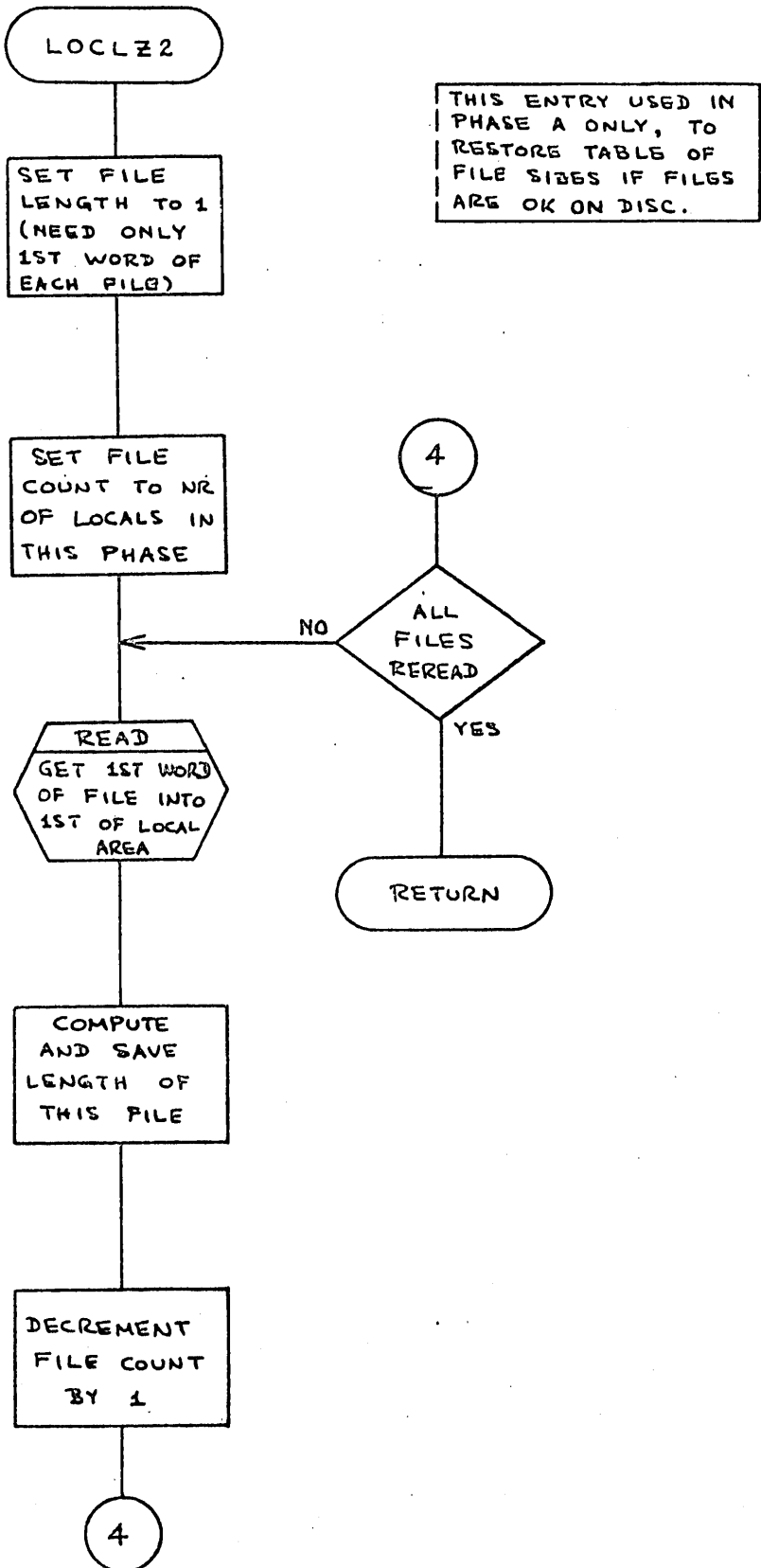
CONTROL DATA CORPORATION

DIVISION

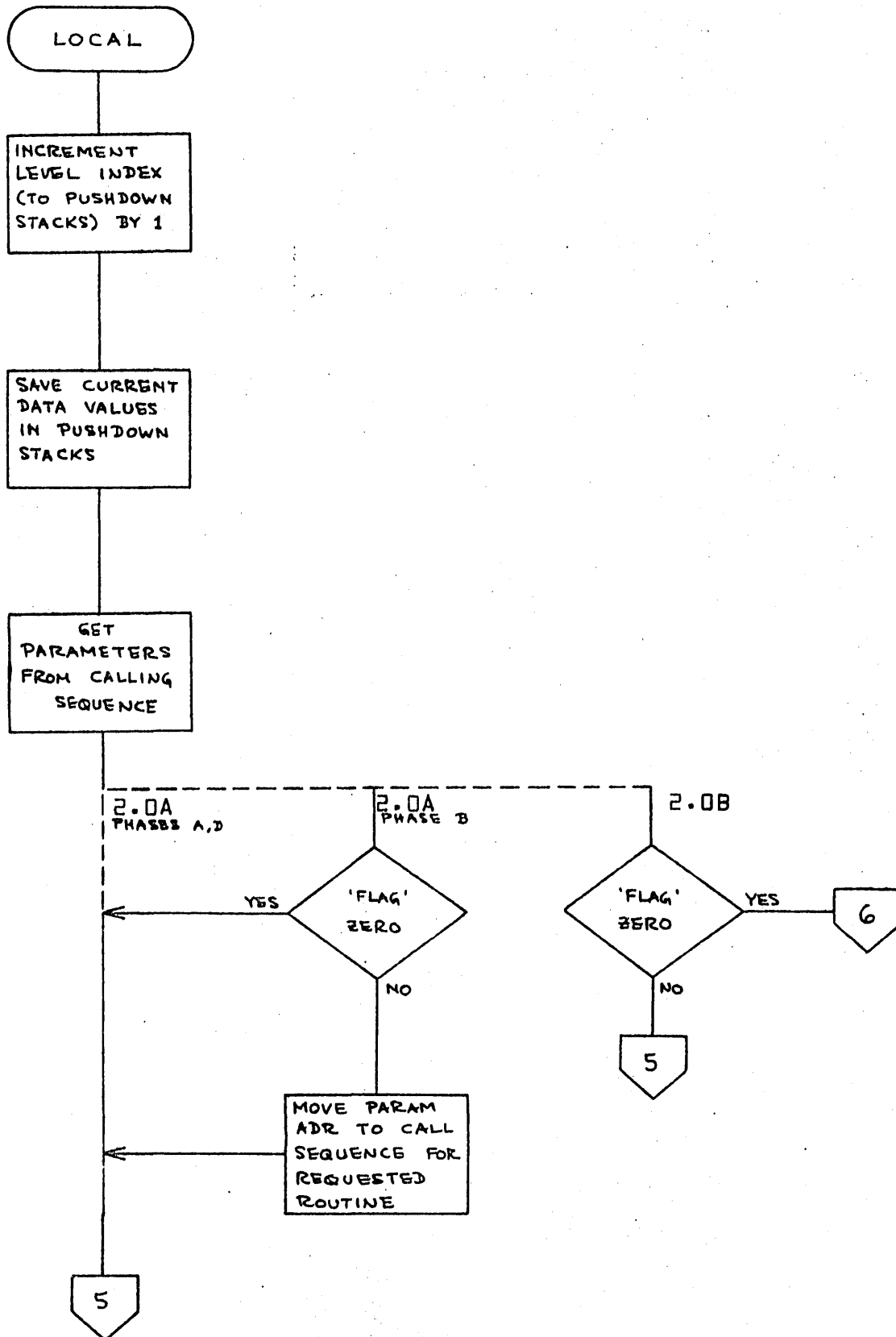
DOCUMENT CLASS IMS PAGE NO. 2-16  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. COO5 V.2.0 MACHINE SERIES 1700



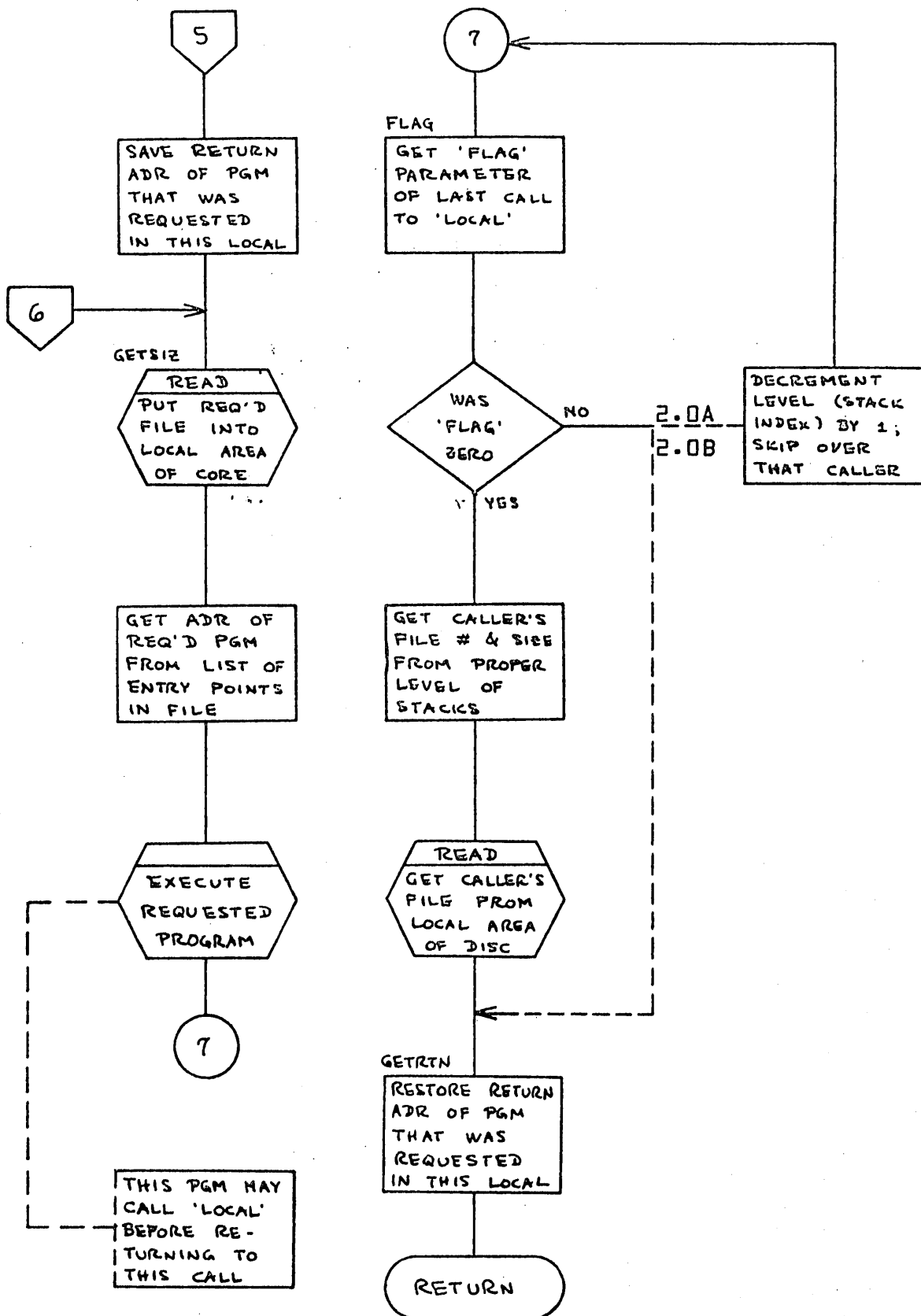
DOCUMENT CLASS IMS PAGE NO. 2-17  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. COO5 V.2.0 MACHINE SERIES 1700



DOCUMENT CLASS IMS PAGE NO. 2-18  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005 V.2.0 MACHINE SERIES 1700







DOCUMENT CLASS IMS PAGE NO. 2-20  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005 V.2.0 MACHINE SERIES 1700

## 2.2.2 PHASEA

Phase A operates as follows:

1. Perform PHASEA Initialization.
2. Read a statement.
3. Process the label.
4. Determine statement type.
5. Branch to process the statement type by either
  - a. calling the particular processor if the statement type is not to be processed in PHASEA, or -
  - b. branching to the in-line processing of the statement type in PHASEA.
6. Repeat the procedure from step 2 to step 5 until an END statement is reached. Upon the occurrence of an END statement, exit from PHASEA.

### 2.2.2.0.1 PHASEA Initialization

The initialization procedure for PHASEA operation is described in detail in the flow charts for PHASEA. The Phase A Block Data routine contains all data presets for labelled common.

### 2.2.2.0.2 Reading Statements

Each FORTRAN source statement is assigned a statement number by the PHASEA routine. ISTNO is the name of a register which contains the number of the statement about to be read and processed. ISTNO is set to zero by Initialization. It is then increased by 1 prior to reading each source statement. In this manner ISTNO contains the number of each source statement at the time it is to be read. A source statement is read by a call to the routine whose name is GNST. (See item 2.2.2.1)

DOCUMENT CLASS IMS PAGE NO. 2-21  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. CD05\*3.1 A/B MACHINE SERIES 1700

2.2.2.0.3 Format of Input Entry

The GNST routine will read a source statement from a card and store it in its internal buffer {ISRS}. The characters which make up the source statement are recorded in the ASCII code as follows:

<u>Character</u>	<u>ASCII Code</u>
Digits: 0-9	¢30 - 39
Letters: A-Z	¢41 - ¢5A
 <u>Special Characters</u>	
Dollar Sign ¢	¢24
Period .	¢2E
Plus +	¢2B
Minus -	¢2D
Equal Sign =	¢3D
Left Paren {	¢28
Right Paren }	¢29
Comma ,	¢2C
Slash /	¢2F
Asterisk *	¢2A
Space	¢20
Statement Terminator	¢20
Single Quote '	¢27

The GNST routine will convert the code for each character of the statement from ASCII to Internal FORTRAN. The Internal FORTRAN code is as follows:

<u>Character</u>	<u>Internal FORTRAN Code</u>
Digit: 0-9	0 - 9
Letter: A-Z	10 - 35
 <u>Special Characters</u>	
Dollar Sign	36
Period	37
Plus	38
Minus	39
Equal Sign	40
Left Paren	41
Right Paren	42
Comma	43
Slash	44
Asterisk	45
Space	46
Statement Terminator	47
Single Quote	48

Following the code conversions, the entire statement is transferred to the source buffer {ISORS}. The buffer which is internal to the GNST

DOCUMENT CLASS IMS PAGE NO. 2-22  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

routine and the source buffer are defined in the source language of the compiler by COMMON statements. The name of the source buffer is ISORS, and its length is 208. Upon return from GNST, ISORSX contains the number of words in the source statement.

2.2.2.0.4 Processing Label

The PHASEA subroutine will call the subroutine whose name is PLABEL. This subroutine will determine whether or not there is a label and, if so, process it.

2.2.2.0.5 Identifying Statement Type

The type of the source statement currently being processed is determined by the subroutine whose name is TYPE. Upon return from the subroutine TYPE, an identifying number giving statement type is recorded in the third word of the output buffer. {Refer to items 2.2.2.0.6.2 and 2.2.2.0.6.2.1}.

There are 47 type numbers used to identify FORTRAN source statements:

<u>Type Number</u>	<u>Statement</u>
0	DIMENSION
1	COMMON
2	INTEGER
3	REAL
4	INTEGER FUNCTION
5	REAL FUNCTION
6	PROGRAM
7	SINGLE
8	BYTE
9	SIGNED BYTE
10	EXTERNAL
11	RELATIVE
12	EQUIVALENCE
13	BLOCK DATA
14	SUBROUTINE
15	FUNCTION
16	DATA
17	FORMAT
18	Replacement statement
19	Statement function
20	ASSIGN
21	CALL
22	RETURN

DOCUMENT CLASS IMS PAGE NO. 2-23  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

23	NOT USED
24	UNCONDITIONAL GO TO
25	COMPUTED GO TO
26	ASSIGNED GO TO
27	CONTINUE
28	STOP
29	STOP n
30	PAUSE
31	PAUSE n
32	END
33	END FILE
34	REWIND
35	BACKSPACE
36	READ {unformatted}
37	READ {formatted}
38	WRITE {unformatted}
39	WRITE {formatted}
40	BEGIN DO
41	END DO
42	ARITHMETIC IF
43	LOGICAL IF
44	ASSEM
45	OPEN
46	DOUBLE PRECISION
47	DOUBLE PRECISION FUNCTION

#### 2.2.2.0.6 Branching to Process Statements

Branching to process a statement is accomplished by means of a Computed GO TO statement in which the index ITEMPX contains the type number +1 for the FORTRAN source statement. A jump is made to a particular statement number in the subroutine depending on the value in ITEMPX. After processing each statement, a test is made to see if the statement has a label. If the source statement does have a label, a call is made to the ENDDO subroutine. If this statement label terminates a DO loop, ENDDO will generate the necessary intermediate language for the DO loop termination procedure. Prior to calling ENDDO, the statement label is recorded in ILLABL.

#### 2.2.2.0.6.1 Statements Not Processed in the PHASEA Routine

Each of these statements is processed by a call to its processor subroutine. The call is made from PHASEA subsequent to branching.

DOCUMENT CLASS IMS PAGE NO. 2-24  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. COO5\*3.1 A/B MACHINE SERIES 1700

Source Statement	Subroutine called from PHASEA
DIMENSION	DIMPR
COMMON	COMNPR
SINGLE	TYPEPR
INTEGER	TYPEPR
INTEGER FUNCTION	SUBPPR
REAL FUNCTION	SUBPPR
REAL	TYPEPR
BYTE	BYEQPR
SIGNED BYTE	BYEQPR
EXTERNAL	EXRLPR
RELATIVE	EXRLPR
EQUIVALENCE	BYEQPR
FUNCTION	SUBPPR
SUBROUTINE	SUBPPR
DATA	DATAPR
FORMAT	CHECKF
ASSIGN	ASGNPR
CALL	ARITH
ENDFILE	ERBPR
REWIND	ERBPR
BACKSPACE	ERBPR
READ unformatted	IOSPR
READ formatted	IOSPR
WRITE unformatted	IOSPR
WRITE formatted	IOSPR
DO	BDOPR
ASSEM	ASEMPR
OPEN	IOSPR
DOUBLE PRECISION	TYPEPR
DOUBLE PRECISION FUNCTION	SUBPPR

### 2.2.2.0.6.2 Statements Processed in the PHASEA Routine

Several statement types are processed in the PHASEA subroutine. These are PROGRAM, BLOCK DATA, RETURN, GO TO, IF, CONTINUE, STOP, PAUSE, replacement statements, statement functions, and END.

#### 2.2.2.0.6.2.1 Program

The PHASEA routine verifies the correct format of the statement, then records the name of the program in the symbol table as a program name. In the event of a format error, an indication is made by a call to the DIAG subroutine. The PHASEA subroutine then reads and processes the following source statement.

DOCUMENT CLASS IMS PAGE NO. 2-25  
PRODUCT NAME I700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES I700

#### 2.2.2.0.b.2.2 Block Data

The PHASEA routine verifies the correct format of the statement. In the event of an error, an indication is given by a call to the DIAG routine. The switch named ISUBP is set to a "1" to indicate the compiler will generate object code for COMMON, DATA, BYTE, EQUIVALENCE & DIMENSION statements only.

#### 2.2.2.0.b.2.3 Return

A RETURN statement is legal only if the source program is either a FUNCTION or SUBROUTINE. If the RETURN statement is used legally, the intermediate language is passed on for processing by a subsequent phase.

If the return statement is used illegally, an error diagnostic is produced by a call to the DIAG subroutine. A STOP statement is substituted for the RETURN statement and it is passed on as intermediate language to be processed by a subsequent phase.

#### 2.2.2.0.b.2.4 GO TO

The GO TO type is determined. Depending on the type, the statement is syntax checked and converted to list notation. Computed GO TO causes a call to ARITH.

DOCUMENT CLASS IMS PAGE NO. 2-26  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005 V.2.0 MACHINE SERIES 1700

## 2.2.2.0.6.2.5 IF

For both logical IF and Arithmetic IF statements, the expression in parentheses is processed by a call to the ARITH subroutine. After this call if the statement is an arithmetic IF, PHASEA will process the three statement labels following the arithmetic expression in the source statement. Statement labels following the expression must be separated from each other by commas, and the last one must be followed by an EOS. There must be exactly 3 statement labels in an Arithmetic IF statement. The statement labels are processed as follows:

PHASEA calls the RDLABL subroutine in order to extract the statement label from the source buffer. If the statement label does not appear in the symbol table, it is entered therein by a call to the STORE subroutine. Upon return from STORE, the entry in the symbol table is assigned the classification for a statement label by -

7 → ICLASS(ISYMX)

An entry is made into the output when the pointer to the symbol table entry is recorded as part of the intermediate language by -

(ISYMX) + (ISYMP) → IBUF2(ITEMP1)

(Upon return from ARITH and prior to processing the 1st statement label, the tally register IBUF2X was recorded in ITEMP1. IBUF2X was then increased by three for the three extra words of intermediate language generated in IBUF2).

After all the labels are processed, WORD 1 of the output buffer is set to the word length of the output buffer by -

(IBUF2X) - 1 → IBUF2(1).

The PHASEA subroutine will check for and give diagnostics upon occurrence of any of the following errors:



## CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 2-27  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. CD05 V.2.0 MACHINE SERIES 1700

1. There are either fewer than or more than three statement labels following the expression in an Arithmetic IF statement.
2. A character other than a comma has occurred between two statement labels.
3. A character other than an EOS follows the last statement label.
4. Two commas have occurred consecutively in the source statement with no statement label between.

Each of the above errors will cause processing of the Arithmetic IF statement to be terminated.

The LOGICAL IF statement is broken into two statements by the PHASEA subroutine. Only the first of the two subsequent statements is assigned a statement number. There is an output entry generated for each statement. WORD 2 of the 1st output entry contains the statement number for the LOGICAL IF statement. A switch named LOGIF is at all times set to the zero position except -

1. it is set to a "1" when processing the 1st half of a Logical IF statement, and -
2. it is set to a "2" when processing the second half.

The second of the two statements may be one of the following:

REPLACEMENT  
CALL  
GO TO  
STOP  
PAUSE  
RETURN  
READ  
WRITE  
ENDFILE  
REWIND  
BACKSPACE

The switch named LOGIF is at all times set to zero except when processing a statement whose type number identifies the statement as a Logical IF. Upon receipt of a Logical IF statement, the LOGIF switch is set to a 1. The input entry containing the Logical IF is passed as intermediate language. If the state-

DOCUMENT CLASS IMS PAGE NO. 2-28  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. COO5 V.2.0 MACHINE SERIES 1700

ment has a label, the label is recorded temporarily in ITEMP9, otherwise, (ITEMP9) = 0.

The second half of the logical IF statement is moved to the beginning of the ISORS buffer so that it appears to the PHASEA routine as an independent statement. PHASEA then branches to the point just after the call to GNST.

When the following statement is processed, it is recognized as the 2nd part of a Logical IF by the fact that - (LOGIF)=1.

The input entry for the 2nd part of the Logical IF is passed on as intermediate language for processing by a subsequent phase in the same manner as for the 1st part. Afterward, the switch LOGIF is set to a 2. Then if (ITEMP9)  $\neq$  0, the label in ITEMP9 is placed in ILLABL and a call is made to the ENDDO subroutine. If this statement label is associated with the end of a DO loop, ENDDO will generate the necessary intermediate language for termination of the DO loop procedure. PHASEA will then process the next statement in sequence. If (ITEMP9) = 0, PHASEA proceeds immediately to process the next statement in sequence.

The LOGIF switch will be reset to a zero when processing the next statement in sequence.

#### 2.2.2.0.6.2.6 CONTINUE

A CONTINUE record is output. (See Output Format, Phase A, Chapter 7, section 7.5).

#### 2.2.2.0.6.2.7 STOP

PHASEA determines whether or not the STOP statement is a

STOP n

in which "n" represents an octal integer, followed by a legal terminator (EOS character). One of the following three events will occur:

1. The statement is not a

STOP n

and the intermediate language will be passed

## CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS \_\_\_\_\_ IMS \_\_\_\_\_ PAGE NO. \_\_\_\_\_ 2-29  
 PRODUCT NAME \_\_\_\_\_ 1700 M/B STORAGE FORTRAN \_\_\_\_\_  
 PRODUCT MODEL NO. \_\_\_\_\_ C005 \_\_\_\_\_ 0 \_\_\_\_\_ MACHINE SERIES \_\_\_\_\_ 1700 \_\_\_\_\_

on for processing by a subsequent phase.

2. The statement is a

STOP n

where "n" is an octal integer followed by a legal terminator. The type number in word 3 of the output buffer is changed to type number for a STOP n by -

IBUF2(3)+1  $\rightarrow$  IBUF2(3).

3. The statement is a

STOP n

where either "n" is something other than an octal integer or an octal integer followed by something other than a legal terminator. An error diagnostic is produced by a call to the DIAG subroutine. The intermediate language for the statement is passed on.

#### 2.2.2.0.6.2.8 PAUSE

PHASEA determines whether or not the PAUSE statement is a

PAUSE n

in which "n" represents an octal integer, followed by a legal terminator (EOS character). One of the following three events will occur:

1. The statement is not a

PAUSE n

and the intermediate language will be passed on for processing by a subsequent phase.

2. The statement is a

PAUSE n

DOCUMENT CLASS IMS PAGE NO. 2-30  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005 V.2.0 MACHINE SERIES 1700

wherein "n" is an octal integer followed by a legal terminator. The type number in word 3 of the output buffer is changed to the type number for a PAUSE n by -

IBUF2 (3)+1  $\rightarrow$  IBUF2(3).

The octal integer is recorded as the last word of the output entry for the statement. The intermediate language for the statement will be passed on for processing by a subsequent phase.

3. The statement is a

PAUSE n

wherein either "n" is something other than an octal integer or an octal integer followed by something other than a legal terminator. An error diagnostic is produced by a call to the DIAG subroutine. The intermediate language for the statement is passed on.

#### 2.2.2.0.6.2.9 Replacement Statements and Statement Functions

PHASEA reads the first field of the statement using GETF. If the field is not a variable, array, or unassigned name, an error is output and the statement ignored up to the equal sign.

If the field is a legal name, it is entered in SYMTAB, if not already there.

The field terminator is then checked. If the terminator is neither a left parenthesis nor an equal sign, an error is output and the statement ignored up to the equal sign.

If the terminator is left parenthesis and the name has not yet been classified as a variable or array and we have not yet encountered the first executable statement, the name is assumed a statement function. The initial part of the output entry is set up in IBUF2 accordingly, and the function parameters are entered in SYMTAB.

If a statement function is illegal at this point in the program or the name has previously been determined

DOCUMENT CLASS IMS PAGE NO. 2-31  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. CD05 V.2.0 MACHINE SERIES 1700

a variable or array, PHASEA resets ISORSX so that the first field will be reprocessed and calls ARITH to process the information to the left of the equal sign (ISTOP =2).

After ascertaining in all cases that ISORSX is positioned to the field immediately following the equal sign PHASEA recalls ARITH to process the information in the statement to the right of the equal sign (ISTOP = 0).

Upon return from ARITH, PHASEA voids the SYMTAB entries for statement function parameters if it has been processing a statement function.

#### 2.2.2.0.6.2.10 END

If the program being compiled is a main program, a check is made to see if a 'last executable statement' must be generated. If the type number of the previous statement (stored in LEST) is not one of the following, PHASEA will generate the 4 word output record for a STOP statement:

GO TO  
STOP  
Arithmetic IF

If any DO LOOP remains 'open', that is, if the label terminating any DO loop has not been encountered, a fatal diagnostic is generated. A CONTINUE statement is generated in IBUF2 and a call to ENDDO is made for each missing label.

An output record for an END statement is generated and output. Then CPLOOP is called to assign relative addresses for elements and dimensions in COMMON storage. If a fatal error has been detected in Phase A, or if no assembly or execute options (X,M,A,P) are set, compilation is terminated at this point by a call to SKIPIT. Otherwise, an exit is made from PHASEA.

#### 2.2.2.0.6.3 Format of Phase A Output Records

An output record consists of the information necessary to convert the statement to pseudo-assembler instructions in Phase B. The format of the output

## CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 2-32  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. CD05 V.2.0 MACHINE SERIES 1700

record for each statement type is listed in Chapter 7, section 7.5. The first 4 words are the same for all output records:

Word 1 contains the total number of words in the output entry, including words 1-4.

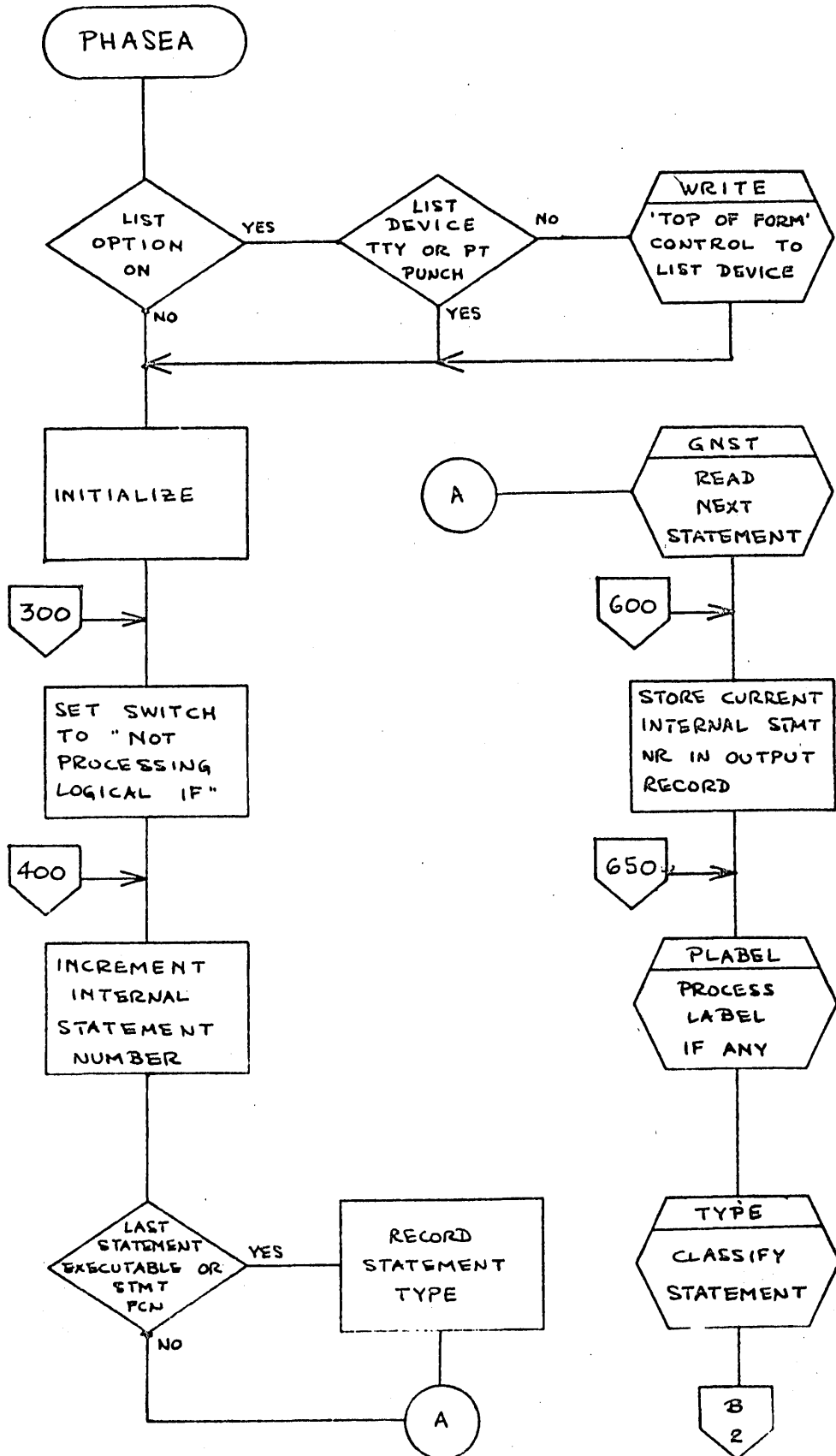
Word 2 contains the statement number for the source statement (ISTNO).

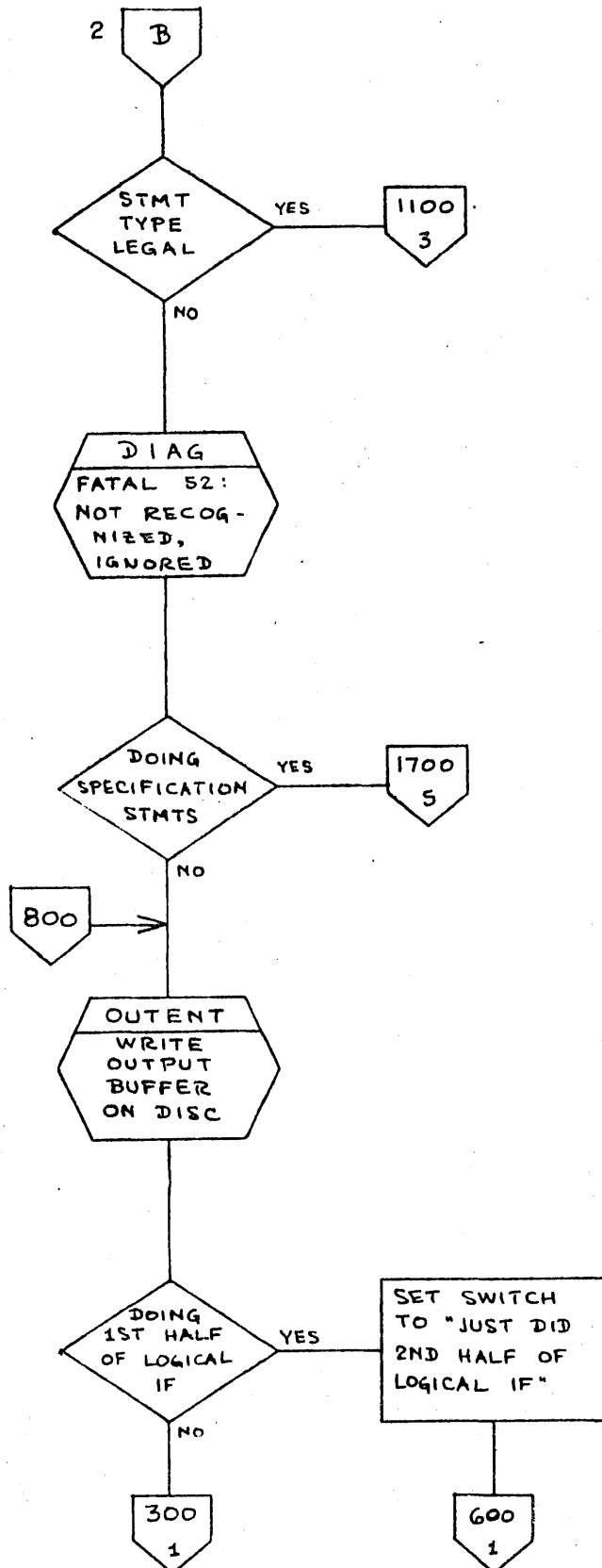
Word 3 contains the statement type.

Word 4 is a switch used to signal the generated statement or statements which comprise the second half of a logical IF.

Word 5. If word 2 is negative, the absolute value of word 2 is the statement number of the statement, and word 5 contains the SYMTAB entry for the statement label of that statement. If word 2 is positive, the coded information appropriate to the statement type begins in word 5.

Output record information is built in IBUF2, and is written on the disk by a call to the OUTENT routine. The starting address for the write operation is specified as the only argument of the subroutine call.



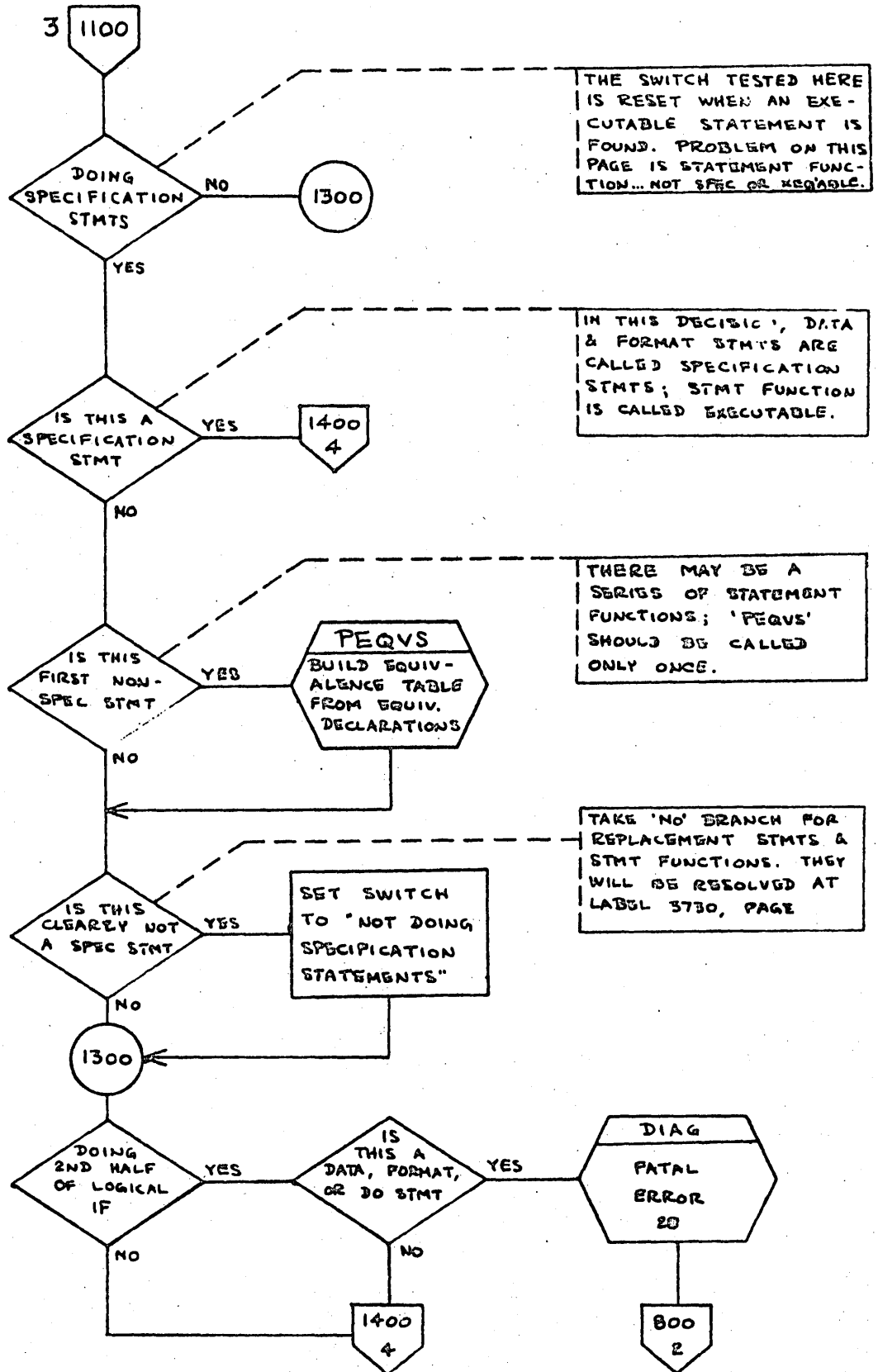




CONTROL DATA CORPORATION

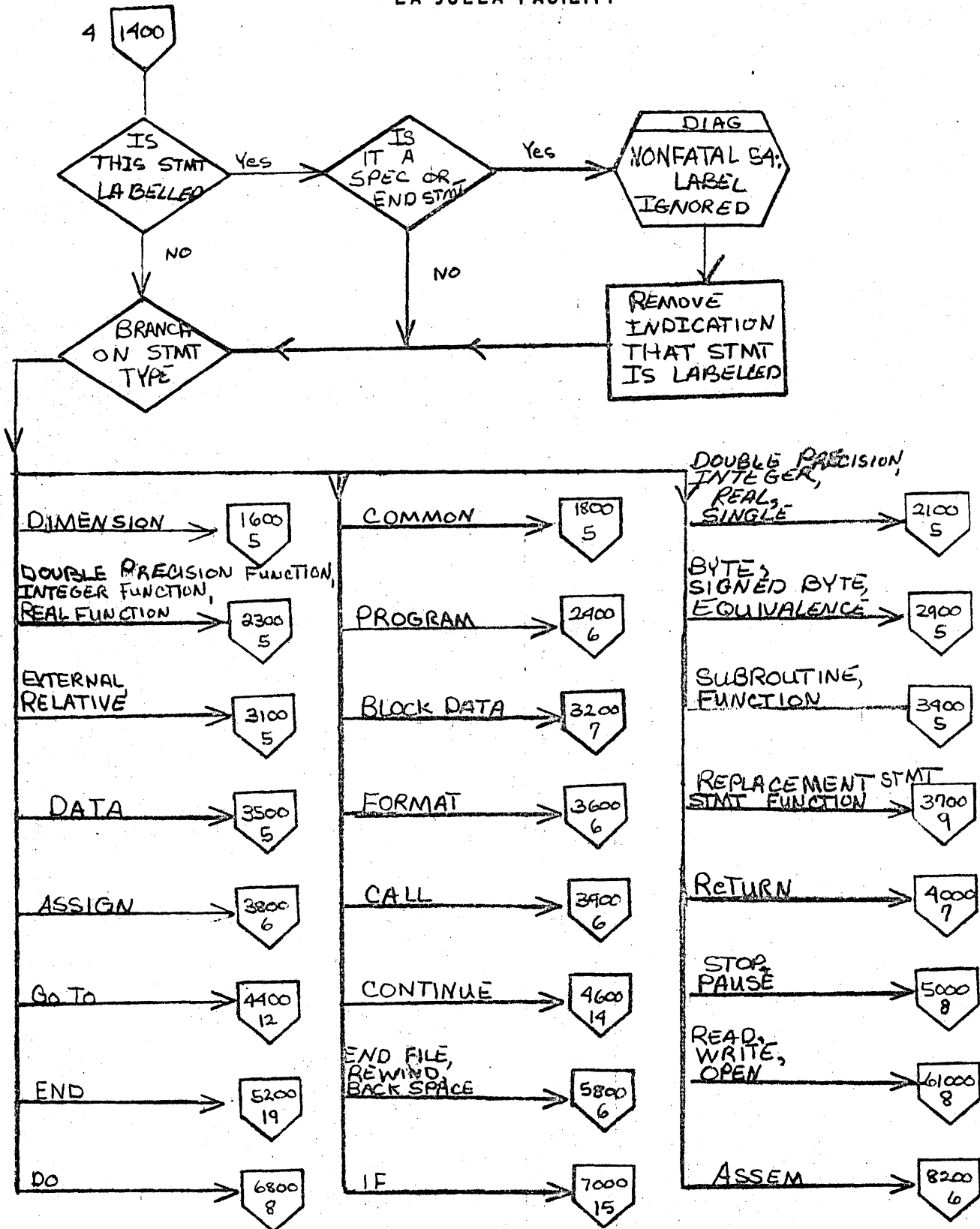
DIVISION

DOCUMENT CLASS IMS PAGE NO. 2-35  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005 V.2.0 MACHINE SERIES 1700





LA JOLLA FACILITY

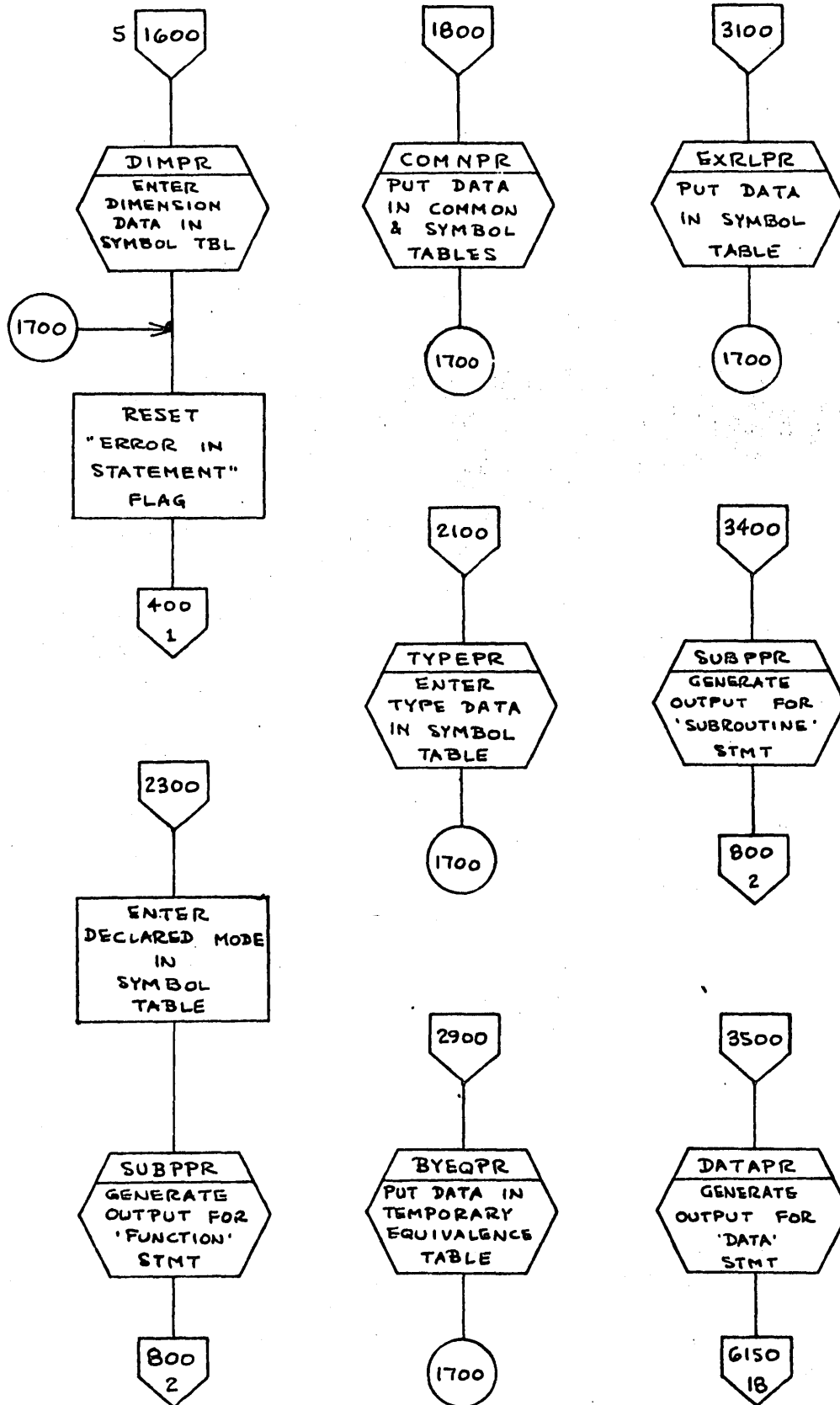


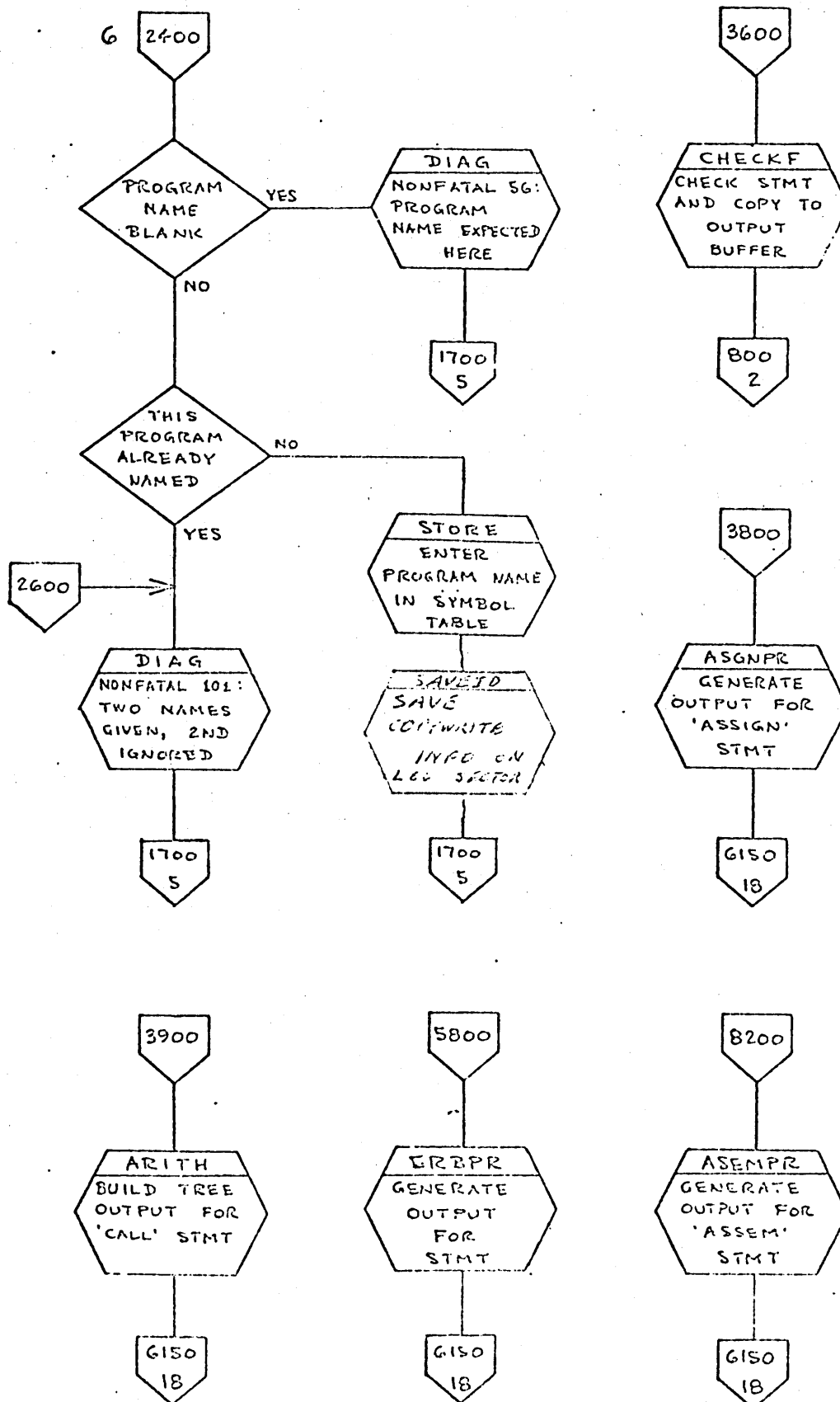
TITLE		PHASE A		DRG. NO.	
DRAWN BY		PROJ.		REVISION	
DATE		SHEET		OF	

CONTROL DATA CORPORATION

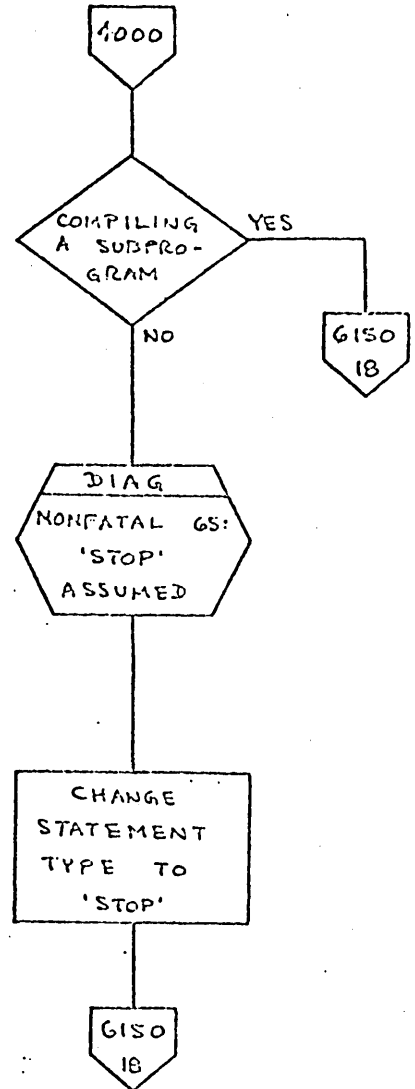
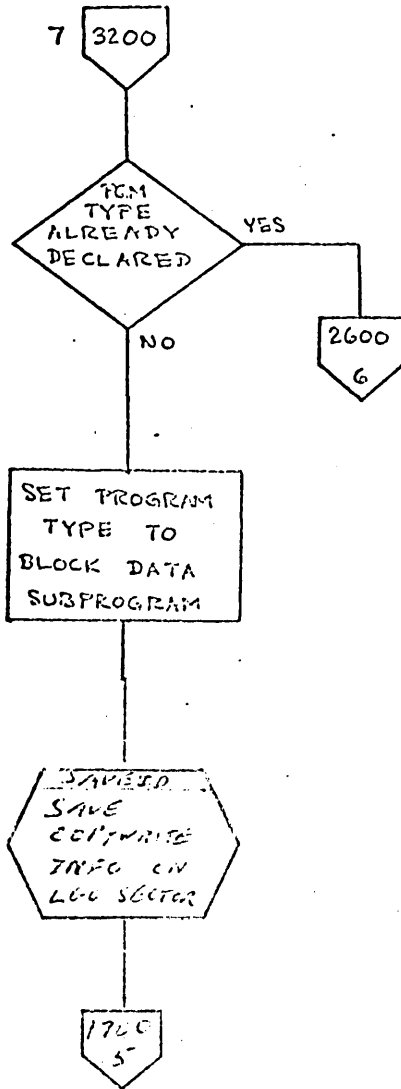
DIVISION

DOCUMENT CLASS IMS PAGE NO. 2-37  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005 V.2.0 MACHINE SERIES 1700





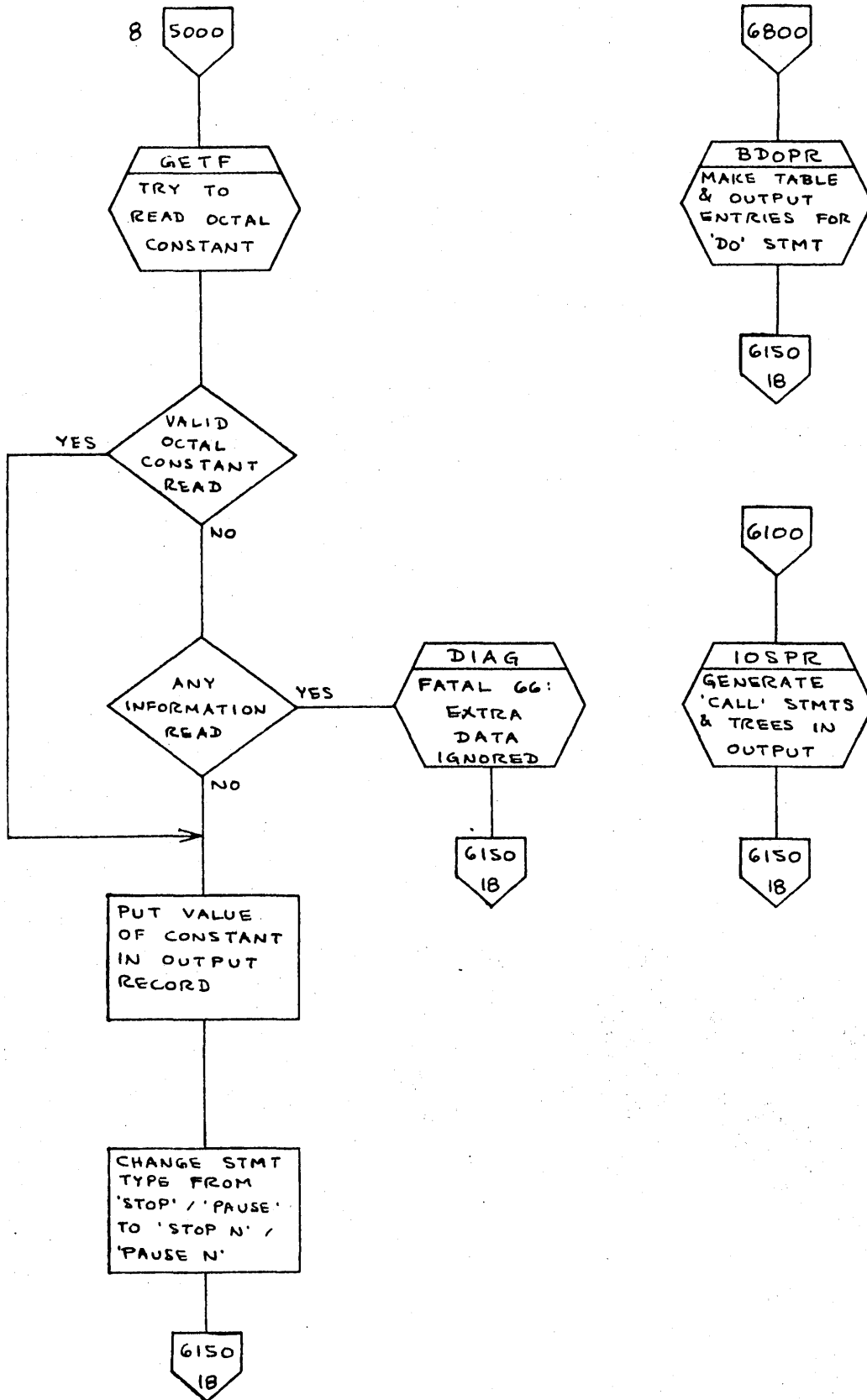
DOCUMENT CLASS IMS PAGE NO. 2-39  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005M3.0 A/B-E006M4.0 MACHINE SERIES 1700



CONTROL DATA CORPORATION

DIVISION

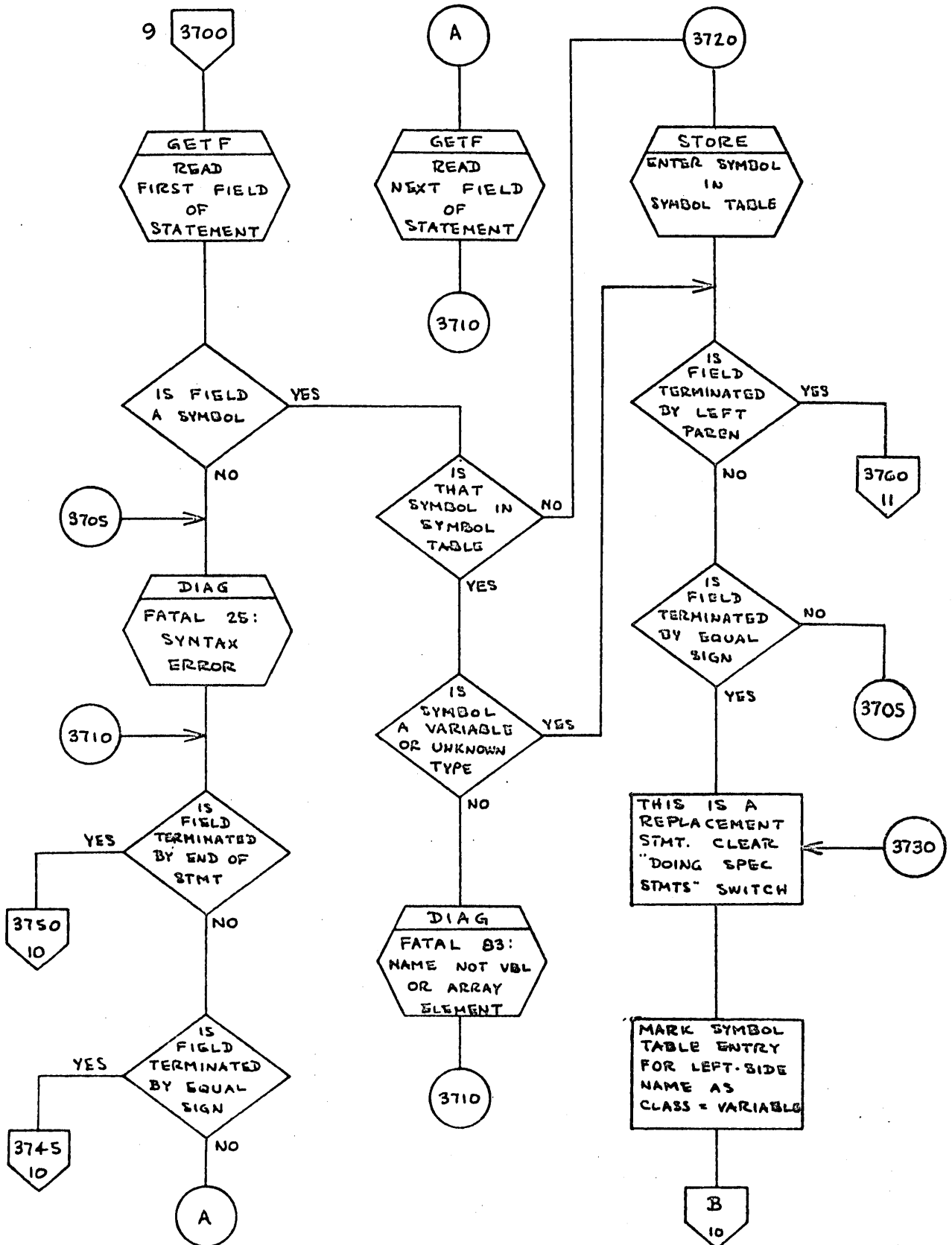
DOCUMENT CLASS IMS PAGE NO. 2-40  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005 V.2.0 MACHINE SERIES 1700

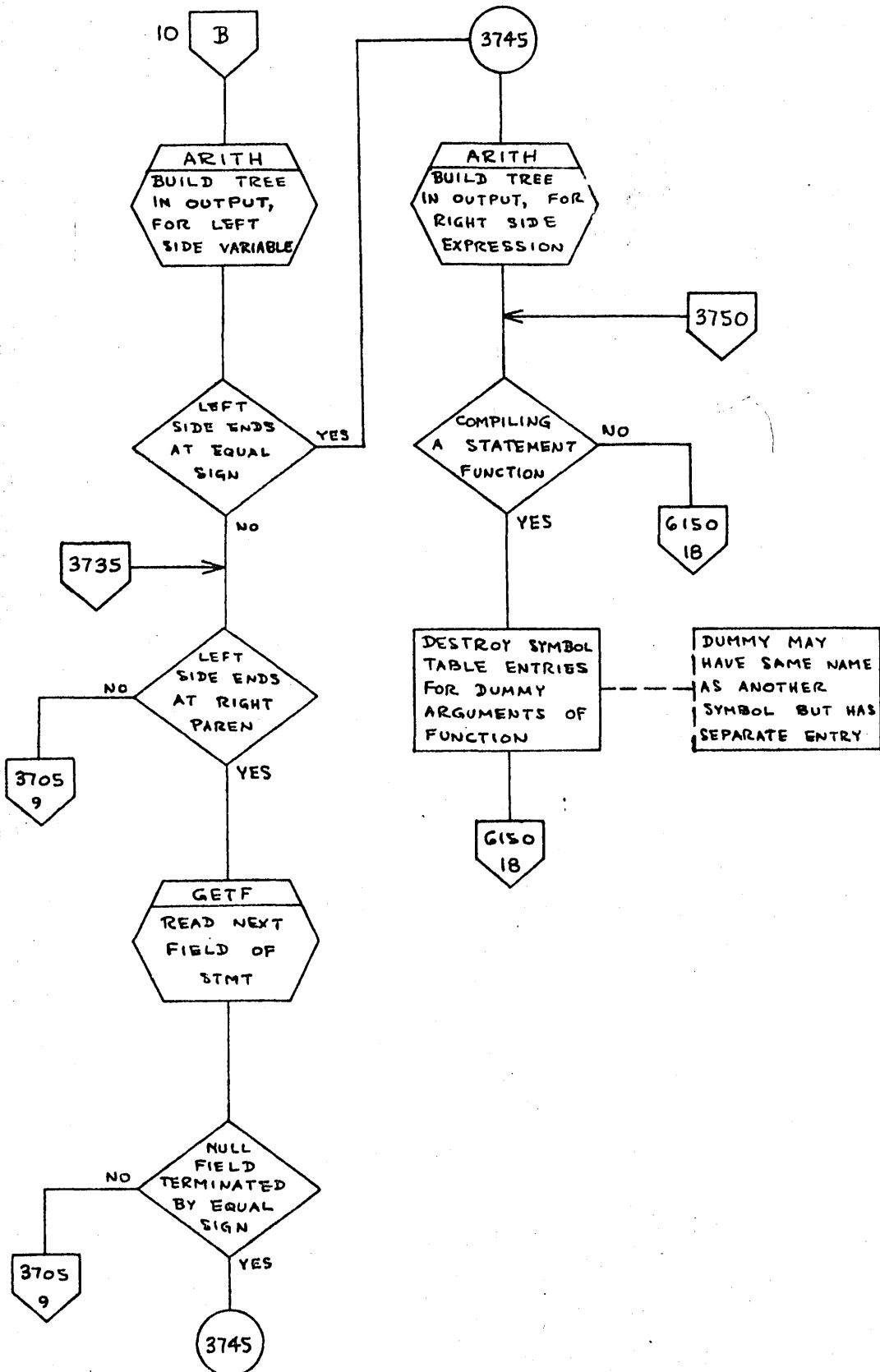


CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 2-41  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. COO5 V.2.0 MACHINE SERIES 1700



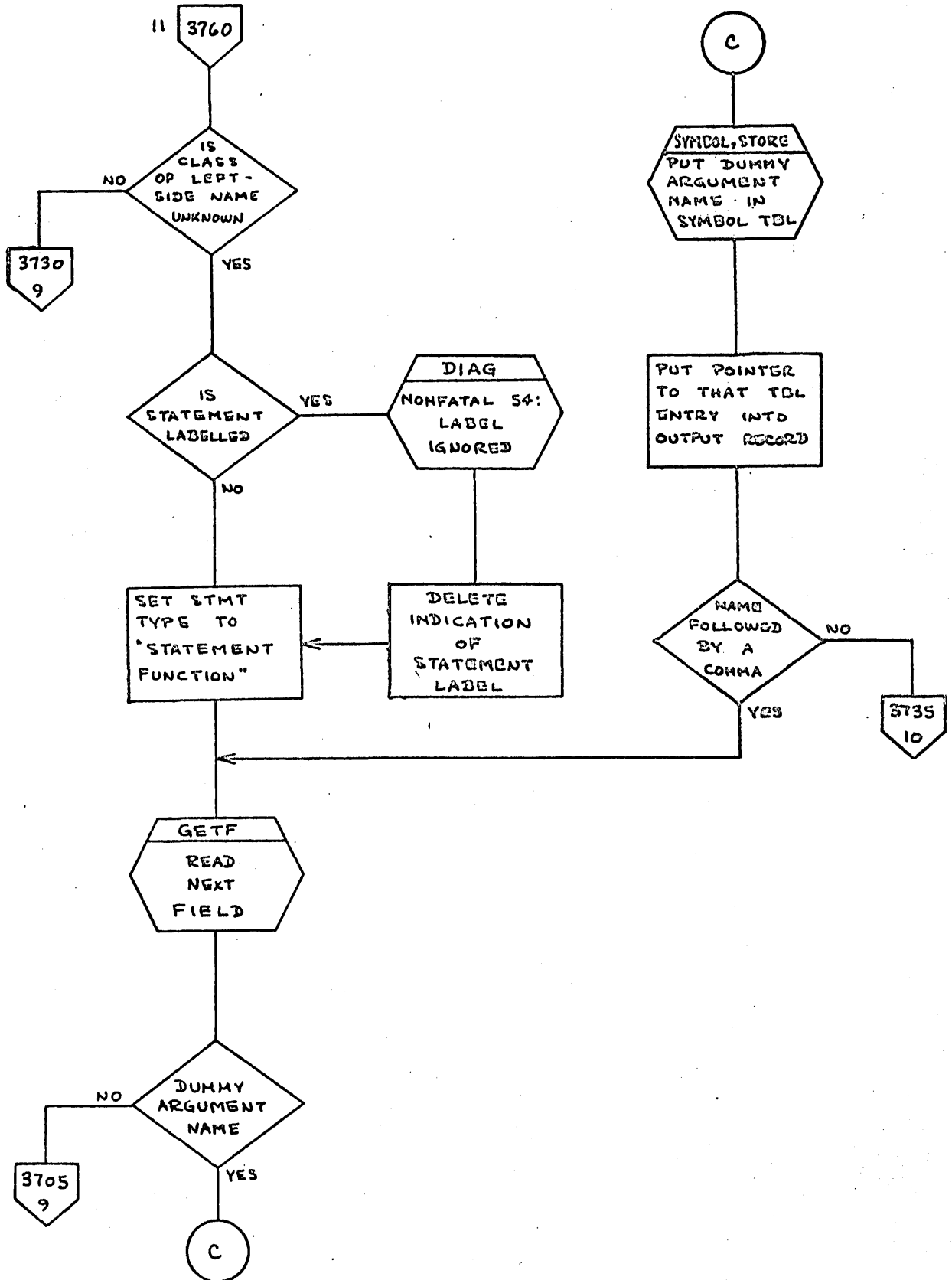


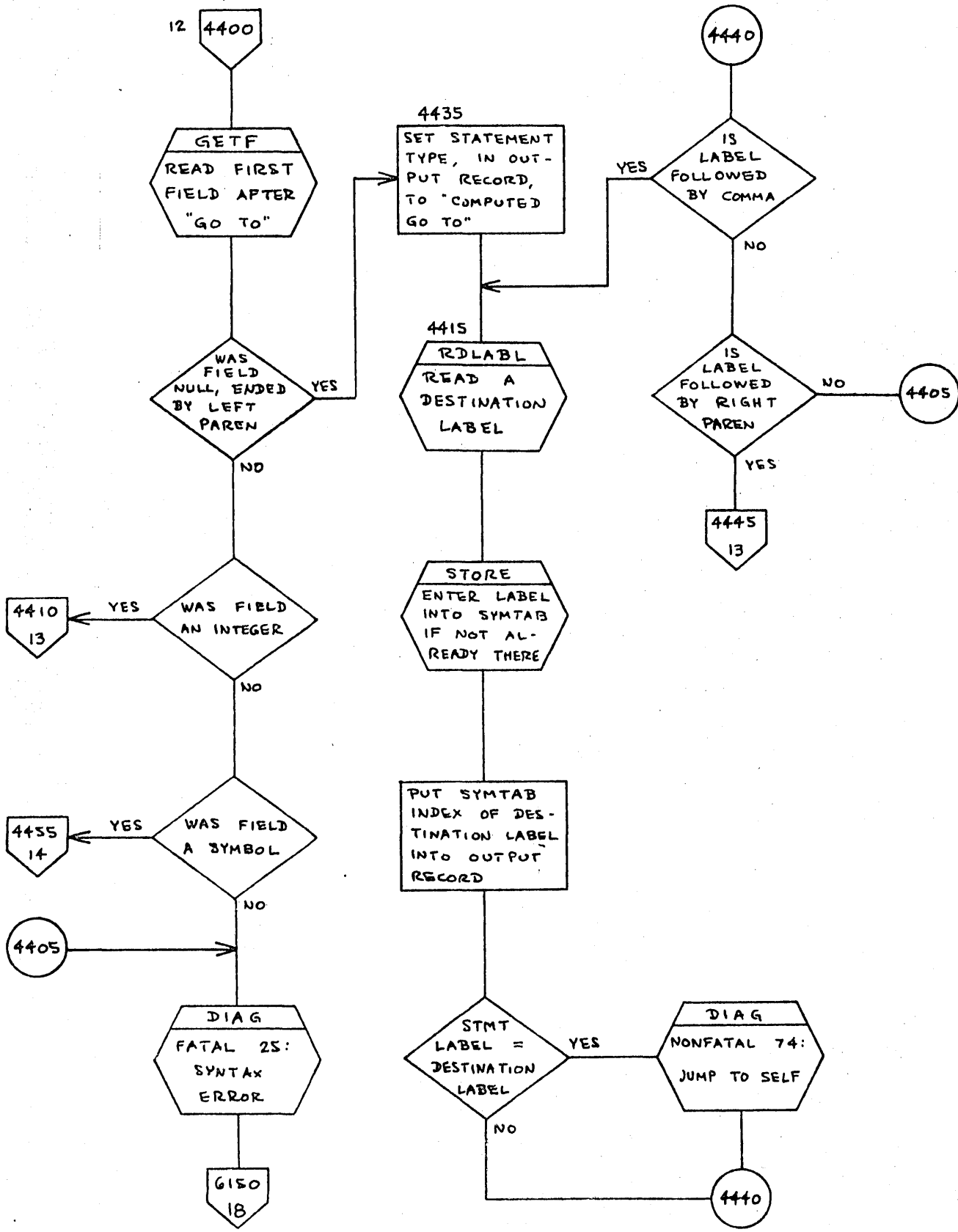


CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS \_\_\_\_\_ IMS \_\_\_\_\_ PAGE NO. \_\_\_\_\_ 2-43  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005 V.2.0 \_\_\_\_\_ MACHINE SERIES 1700

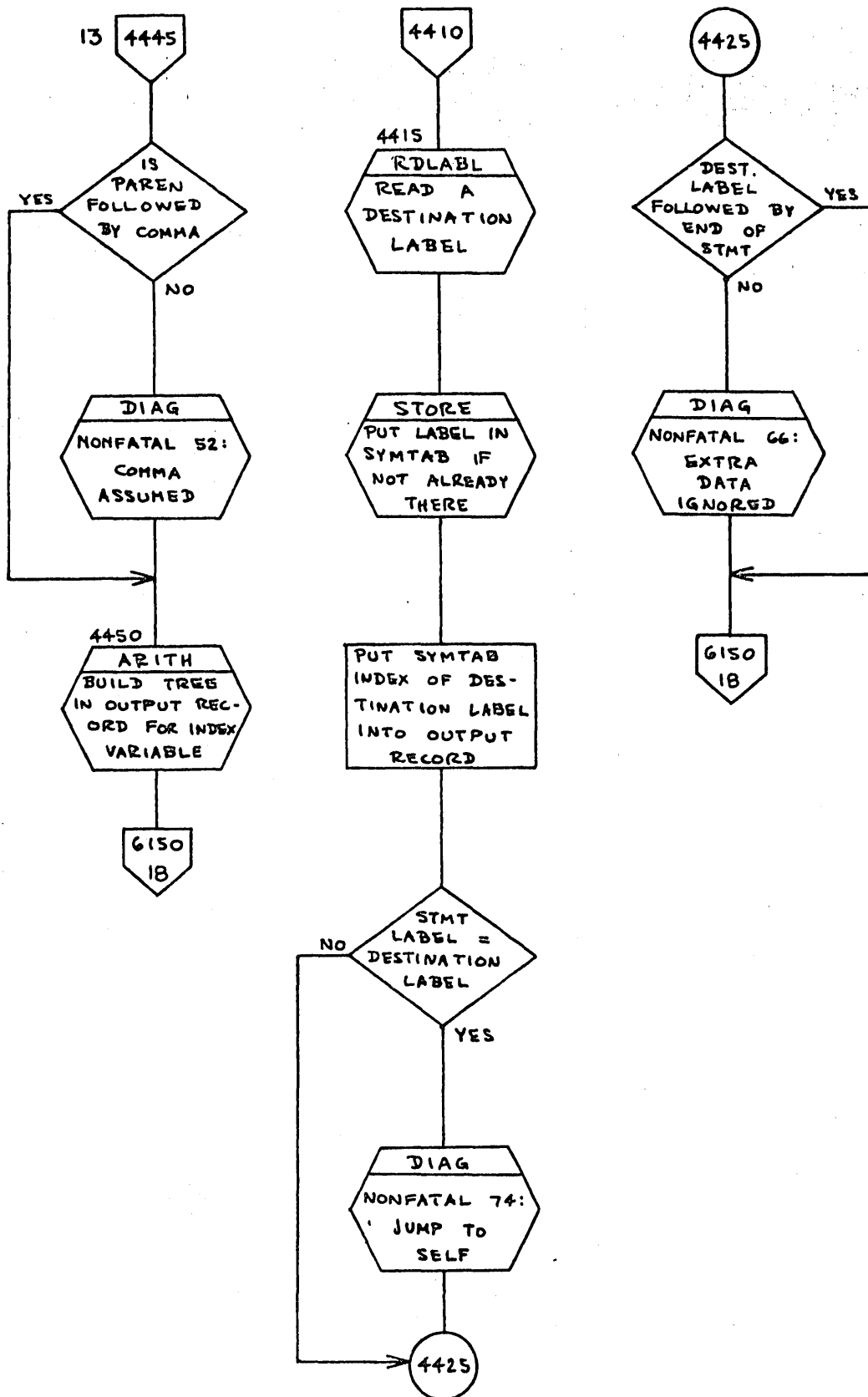




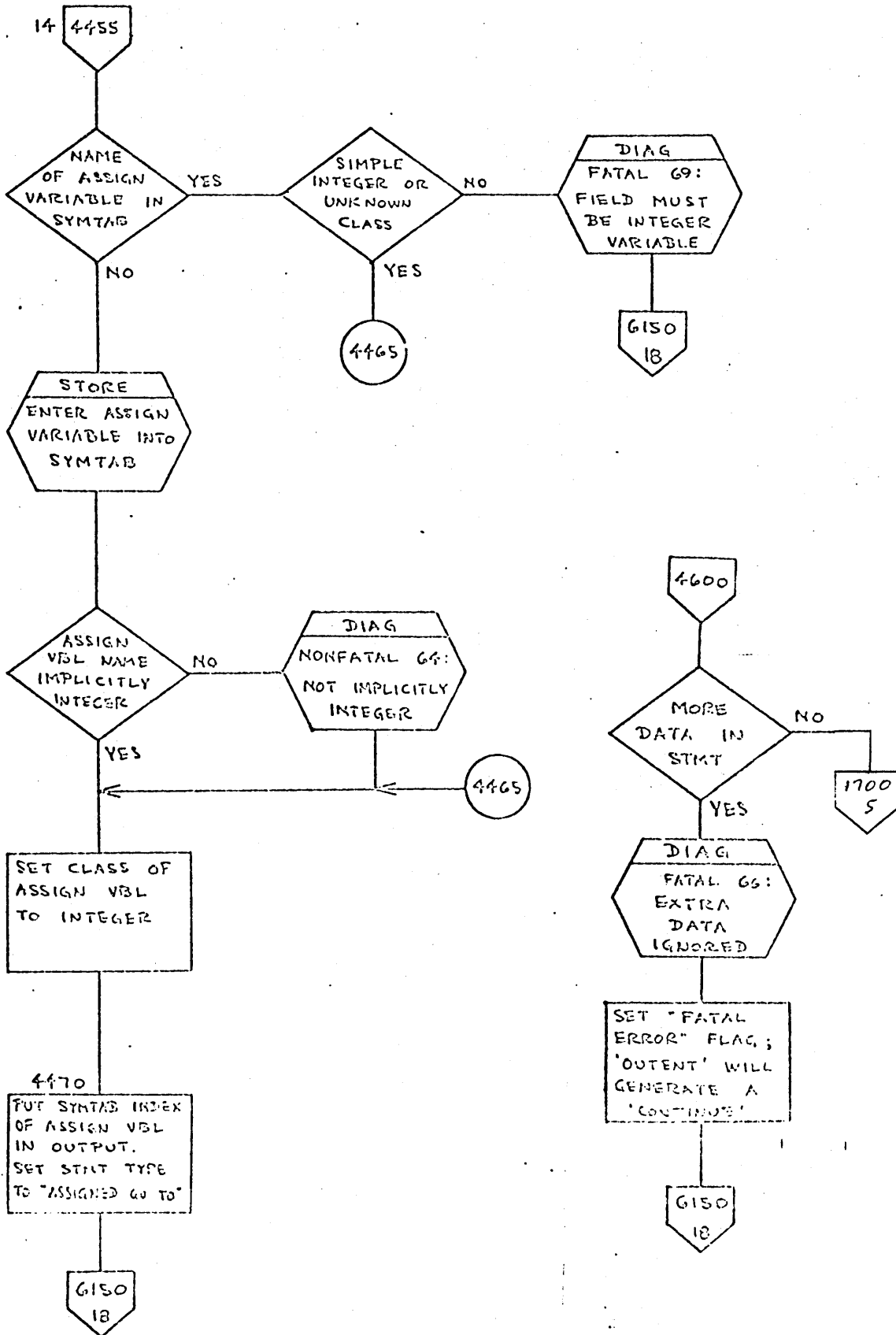
CONTROL DATA CORPORATION

DIVISION

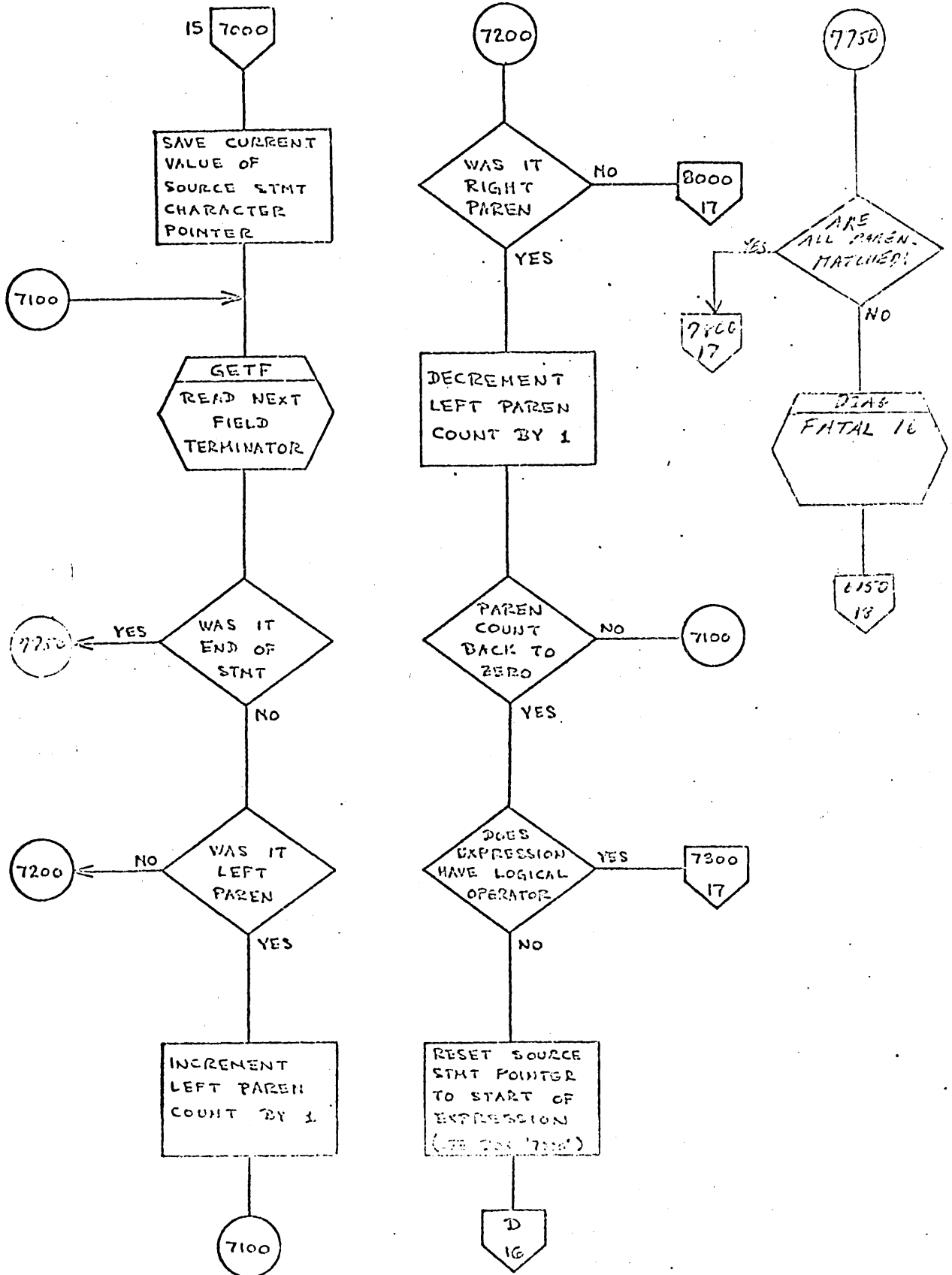
DOCUMENT CLASS IMS PAGE NO. 2-45  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005 V.2.0 MACHINE SERIES 1700



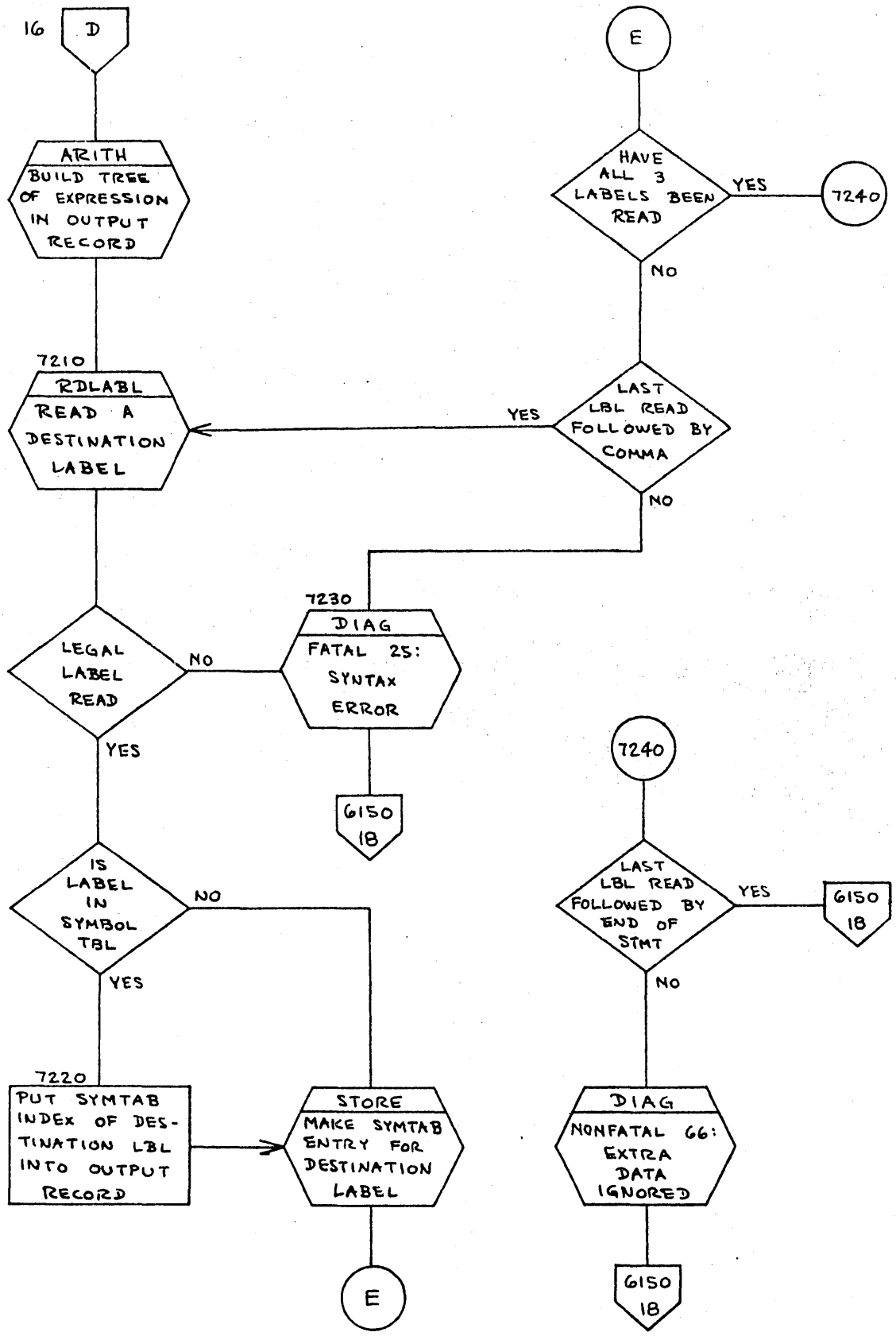
DOCUMENT CLASS IMS PAGE NO. 2-46  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005M3.0 A/B-E006M4.0 MACHINE SERIES 1700



DOCUMENT CLASS IMS PAGE NO. \_\_\_\_\_  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005M3.0 A/B-E006M4.0 MACHINE SERIES 1700



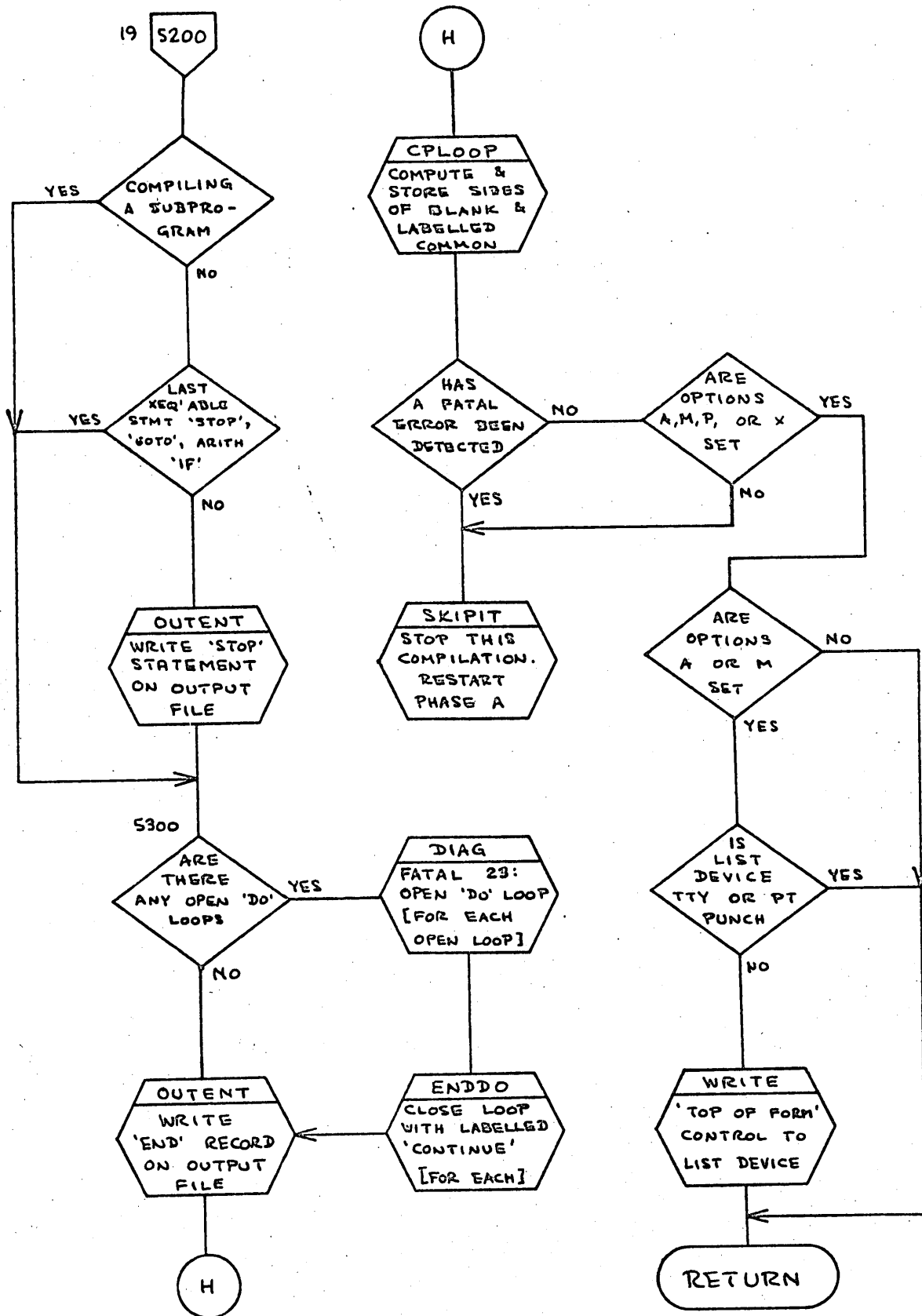
DOCUMENT CLASS IMS PAGE NO. 2-48  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. CD3 V.2.0 MACHINE SERIES 1700











DOCUMENT CLASS IMS PAGE NO. 2-52  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

### 2.2.2.1 Subroutine GNST

The GNST subroutine reads the source statements which are to be processed by the PHASEA subroutine.

With the exception of comment cards, each source card read by GNST is processed in three stages.

#### STAGE 1.

The GNST subroutine calls the READ entry point in IOPRBA to read an 82 character record from unit 1 in the ASCII mode. The input buffer, ISRS, occupies 41 words of Phase A blank common. The GNST routine then scans ISRS from right to left and stores in the variable LENGTH the location of the right-most word containing a non-blank character (not equal to \$2020).

#### STAGE 2.

The contents of the ISRS buffer are printed and transferred to the source buffer ISORS, which occupies 208 words of Phase A blank common.

#### STAGE 3.

The characters stored in the source buffer ISORS are converted from ASCII code to the internal Fortran code (see 2.2.2.0.3).

Continuation records are processed through all three stages during one call to GNST. The initial record of a Fortran statement other than END is passed through stage 1 on one call to GNST, and through stages 2 and 3 on the following call. Because GNST reads at least one record ahead in the source language input, at least two consecutive source records will be processed during a single call to this subroutine.

### 2.2.2.1.1 Normal Flow of Subroutine GNST

This section describes the operations accomplished by a single call to GNST after the first non-comment record has been processed and before the END record has been read. At entry to GNST, the input buffer ISRS will contain the initial record of a Fortran statement that has previously been processed through stage 1. The first task of GNST is to process this ISRS buffer through stage 2;

CONTROL DATA CORPORATION  
LA JOLLA RESOURCE CENTER

DIVISION

DOCUMENT CLASS IMS PAGE NO. 2-53  
PRODUCT NAME L700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. CD05\*3.1 A/B MACHINE SERIES L700

that is, to print the record {if the list option switch, IL, is on} with an internal statement number, and then move the record to ISORS. A full seventy-two characters {36 words} are transferred to ISORS to allow for later continuation cards. The location of the rightmost non-blank word in the record, formerly saved in LENGTH, is transferred to LENSAV.

The subroutine GNST next reads a new input record into ISRS and processes it through state 1, any characters stored as \$FF by the paper tape or magnetic tape drivers are changed to blank {#20} and the location of the rightmost non-blank word is ISRS is stored in LENGTH.

After processing the new input record through stage 1, GNST checks for three special cases - a blank record, a comment record, or a continuation record {not zero or blank in character #6}. These three types of records are immediately processed as follows:

a} Blank record - ignored. A new input record is read and processed through stage 1.

b} Comment record - printed and ignored. If the IL switch is on, the comment record is printed without a statement number and then a new record is read and processed through stage 1.

c} Continuation record - completely processed in one call. The contents of ISRS are printed without a statement number {if IL = 1}. Characters 7 through 72 of the continuation record are added to the end of the statement in the ISORS buffer. ISORSX {set to 1 on entry} is set to the beginning of the added record. The variable LENS AV is reset equal to LENGTH so that LENS AV contains the location of the rightmost non-blank character in the last record added to the ISORS buffer.

If the record in ISRS is the initial line of a Fortran statement other than END, further processing is deferred until the next call to GNST. At this point in the flow, a full Fortran statement {with all continuations} is contained in ISORS. The final step of the GNST subroutine is to convert this statement from ASCII code to internal Fortran code. GNST exits with a full converted Fortran statement in ISORS and the initial line of a Fortran statement in ASCII in ISRS.

DOCUMENT CLASS IMS PAGE NO. 2-54  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. COO5 V.2.0 MACHINE SERIES 1700

### 2.2.2.1.2. First and last source records.

Until the first non comment record is read, stages two and three are skipped in the GNST flow. The variable LENS AV is set to zero by a data statement, and therefore remains zero until a non-comment statement is read. When LENS AV is equal to zero, the printing and transfer of ISRS at the beginning of GNST and the code conversion at the end are not executed. Initial comment cards are printed as usual.

At the end of stage two, after ISRS has been transferred to ISORS, ISRS is tested for the presence of an END record. If the card in ISRS is an END card no more input records are read and the END card is immediately converted to internal code.

### 2.2.2.1.3. Errors

When processing a non-comment statement, GNST tests for the following illegalities.

1. That the first non-comment record is an END statement.
2. That the first non-comment record is a continuation card.
3. That there are more than five continuation cards for a source statement.
4. That characters illegal in Fortran occur in the source statement.

#### 2.2.2.1.3.1. End statement as first non-comment record.

The variable LENS AV is set to zero until the first non-comment card is read and processed. If the END card is read while LENS AV =0, an error diagnostic is given, and further processing of this program is terminated by a call to subroutine SKIPIT.

#### 2.2.2.1.3.2. Continuation card as first non-comment card of deck.

Error flag IERR35 is set, character number 6 is set to blank, and the record is treated as the initial record of a Fortran statement. After the record is fully processed the appropriate error diagnostic is printed.

DOCUMENT CLASS IMS PAGE NO. 2-55  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. CO05 V.2.0 MACHINE SERIES 1700

#### 2.2.2.1.3.3. More than 5 continuation cards.

ITALLY is a tally register for the number of continuation cards per statement. ITALLY is set to zero on entry to GNST and increased by 1 for each continuation card read. When ITALLY exceeds 5 in value the error flag IERR47 is set. Continuation cards in excess of 5 are printed but are not added to ISORS. When the next initial statement card is read, the diagnostic is printed.

#### 2.2.2.1.3.4. Illegal characters

Characters not contained in the Fortran character set are converted to blanks in stage 3, after printing the record. A non-fatal diagnostic is printed for each illegal character.

DOCUMENT CLASS

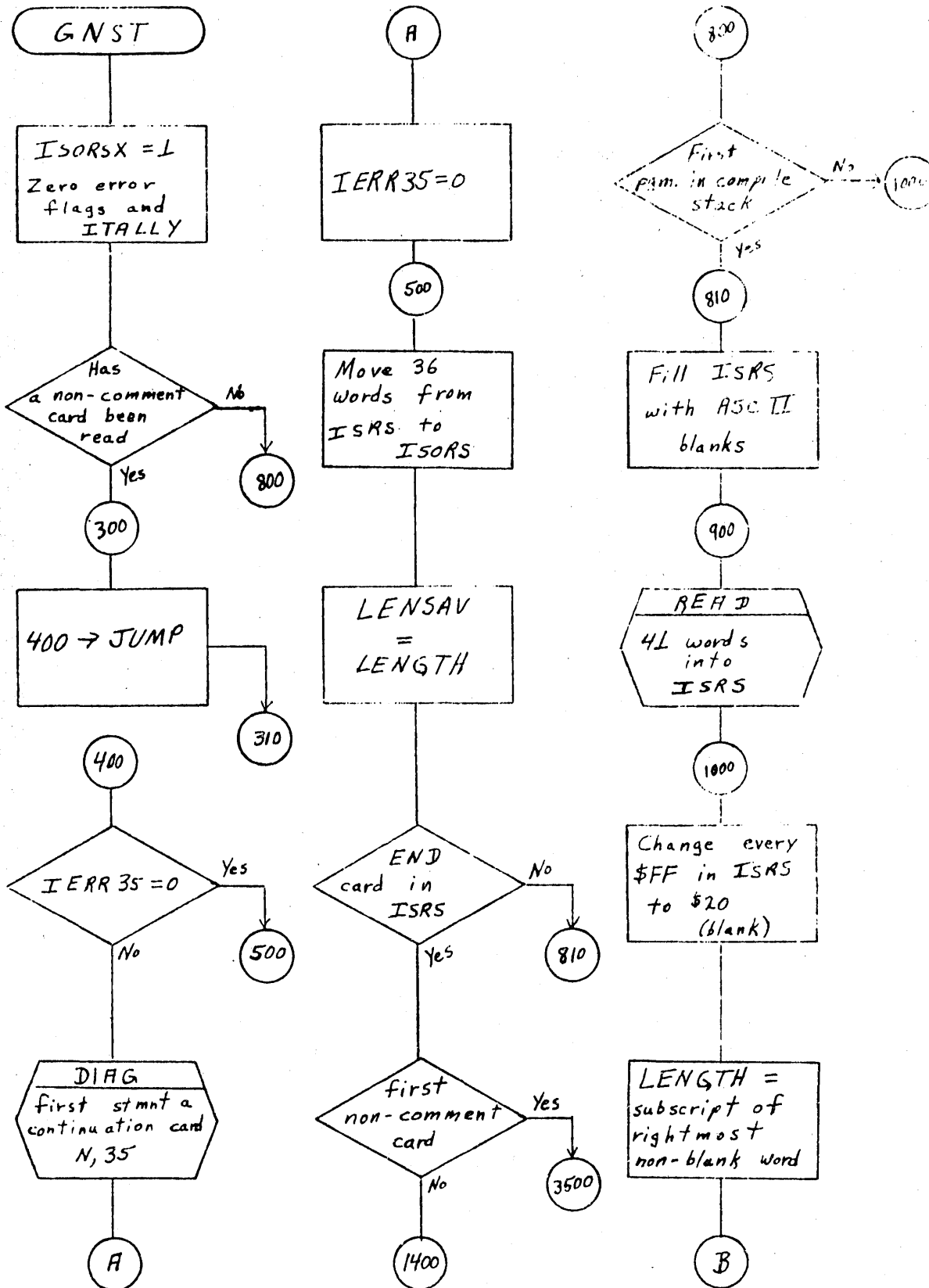
IMS

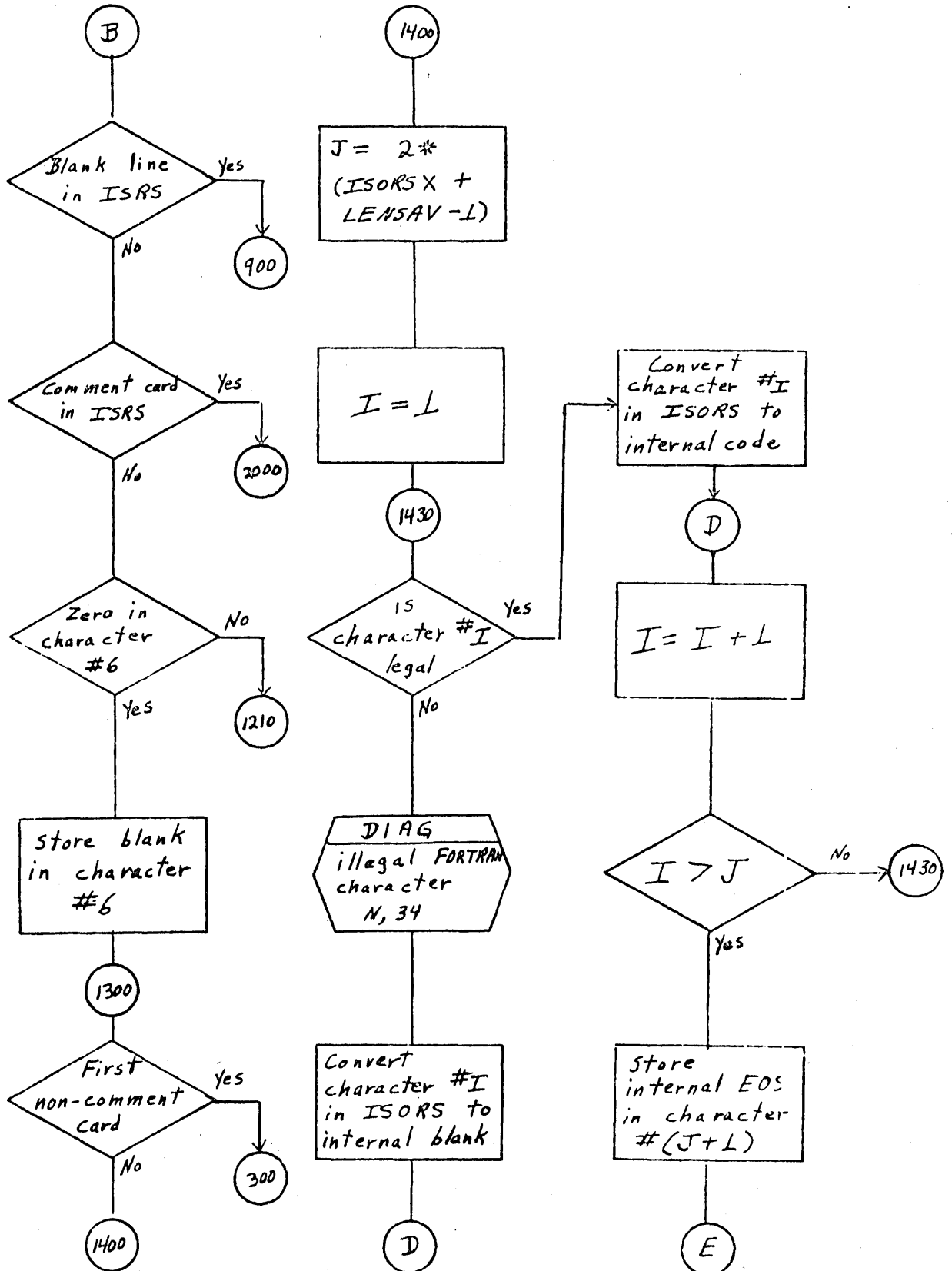
PAGE NO

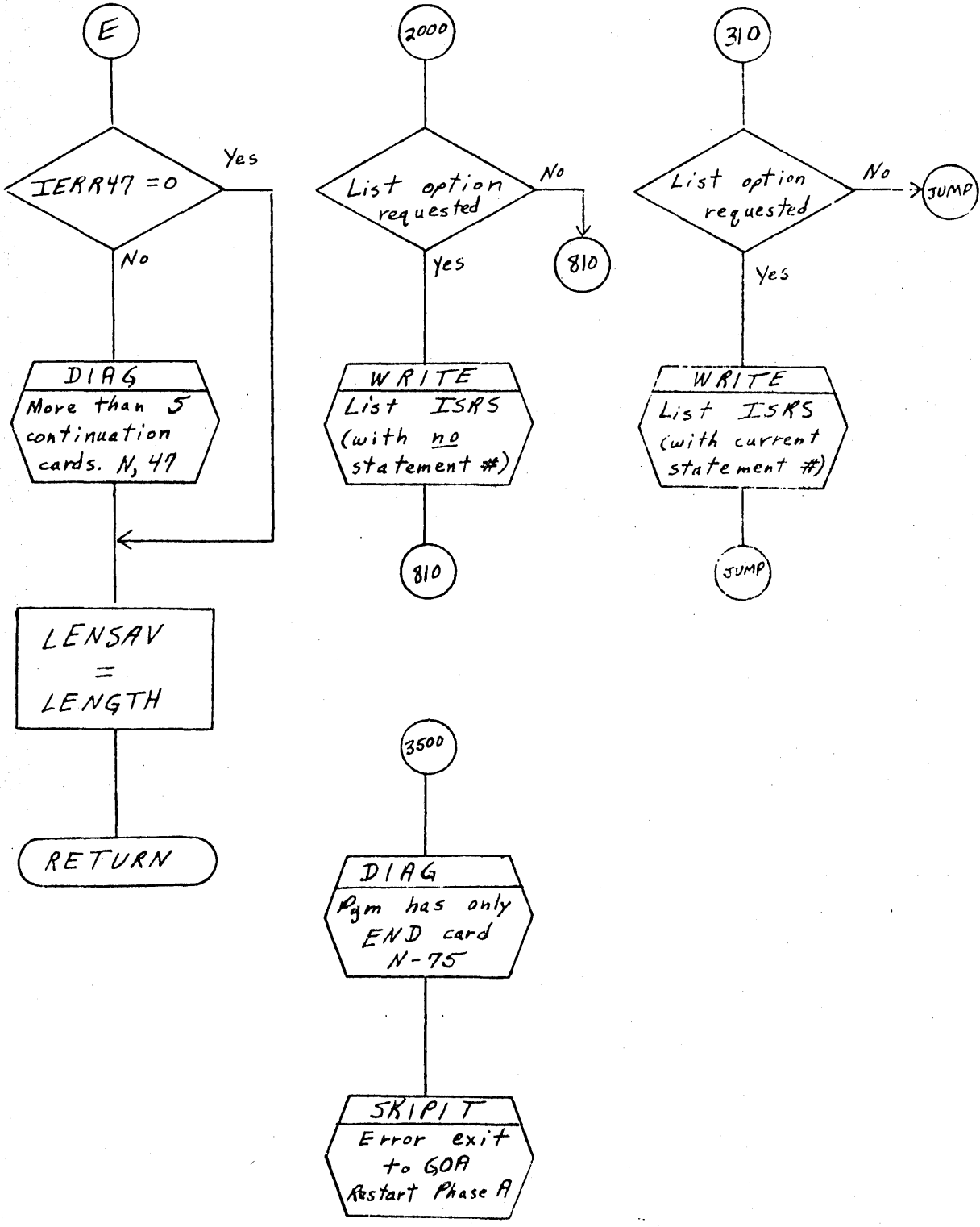
PRODUCT NAME 1700 MASS STORAGE FORTRAN

PRODUCT MODEL NO. CDS V.2.0

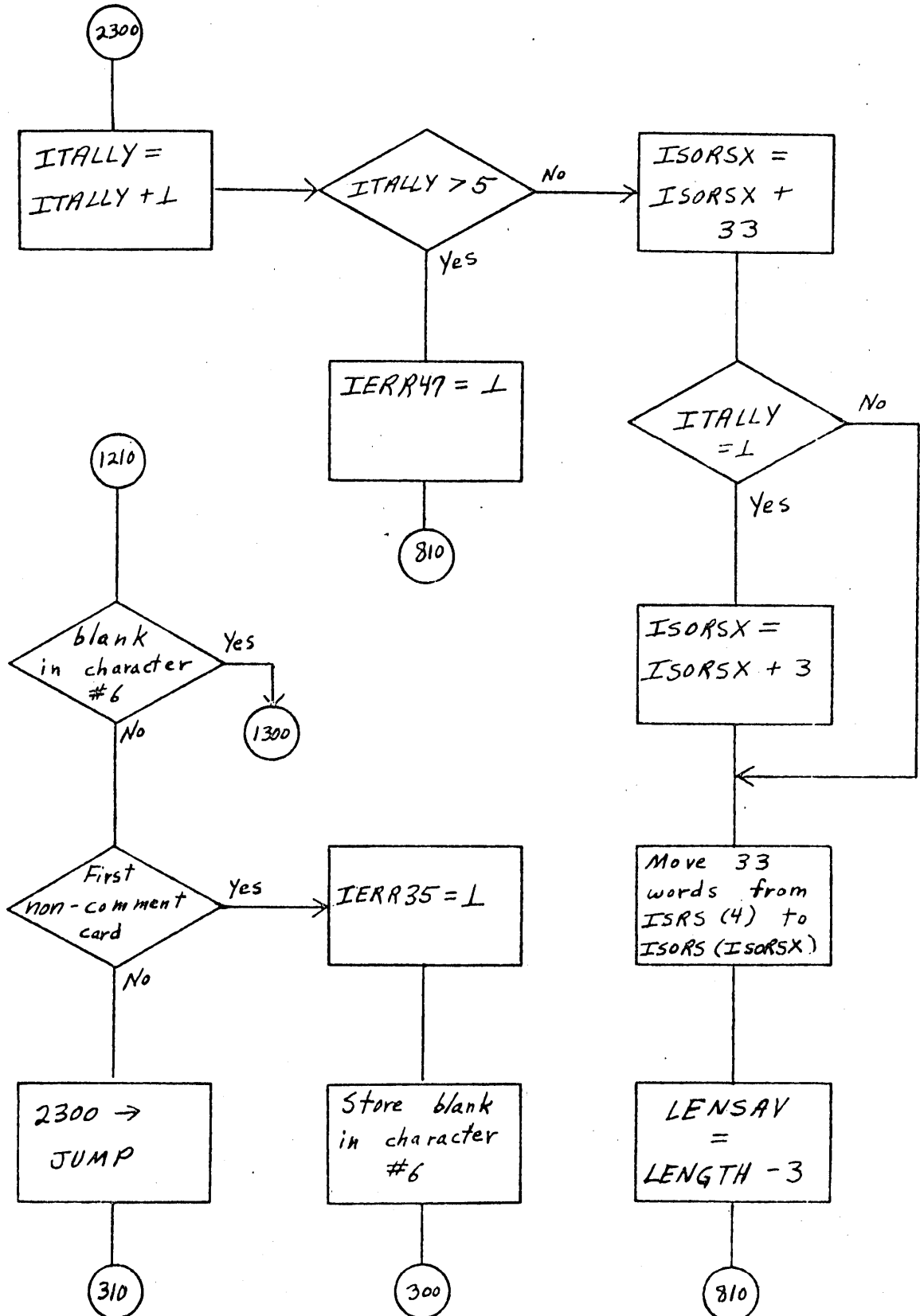
MACHINE SERIES 1700











DOCUMENT CLASS IMS PAGE NO. 2-60  
 PRODUCT NAME L700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005 V2.0 MACHINE SERIES L700

### 2.2.2.2 TYPE

The TYPE subroutine is used by PHASE A to determine the type number for the statement. If the TYPE subroutine is able to determine the type number for the statement, it will place the number in the location IBUF2{3} and exit. If TYPE is unable to identify the statement, it sets an error indicator and returns to the calling routine.

#### 2.2.2.2.1 Entrance to TYPE

Entrance to the TYPE subroutine is made by the following statement:

```
CALL TYPE {error flag}
```

The subroutine statement which leads the TYPE subroutine is as follows:

```
SUBROUTINE TYPE {IX1}
```

where IX1 is the name of the formal parameter corresponding to 'error flag'.

#### 2.2.2.2.2 Initialization for TYPE

Immediately after entrance is made to the TYPE subroutine, the following steps of initialization occur:

1. The error flag is cleared.  
0 → IX1
2. The parentheses level tally register is cleared.  
0 → ITSL
3. The column counter is set to the character position in the source buffer at which scanning of the source statement begins.  
7 → ISORSX

#### 2.2.2.2.3 Identification of Source Statement

If the TYPE subroutine is able to identify the source statement as a replacement statement, it will place the appropriate type number in the holder IBUF2{3} and exit. {See item 2.2.2.2.3.1} If the TYPE subroutine is unable to identify the source statement as a replacement it will search the ISTN table in order to obtain a tape number. {See item 2.2.2.2.3.2}

DOCUMENT CLASS IMS PAGE NO 2-61  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

## 2.2.2.2.3.1 Replacement Statement

The TYPE subroutine begins a scan of the source statement beginning with the 7th character from the beginning. The necessary and sufficient conditions for identifying the statement as a replacement is as follows:

The statement must contain an "=" sign not enclosed in parenthesis which is not followed by a comma not enclosed in parenthesis.

For purposes of scanning, the GETC subroutine is used to extract a single character from the source statement and place it in JCHAR. If the switch JBLANK is set to zero when a call is made to GETC, the space character, whose internal FORTRAN code = 46 is not treated as a significant character. Therefore given a source statement -

D 0 1 0 I = 1 , 5

the TYPE subroutine sees only the significant character and not the intervening spaces which causes it to appear as -

D010I=1,5

This ordinarily would be regarded as a statement in which -

name = expression,

except according to the previously stated rule for a replacement statement, the comma which is not enclosed by parentheses excludes the possibility of this being a replacement statement.

Another special case occurs for the following statement:

FORMAT 10HX=A(1,5)

In this example TYPE encounters an "=" sign which is followed by a comma which is enclosed in parentheses. Ordinarily it would appear to the TYPE subroutine as a

name = expression

type of statement wherein -

1. The name FORMAT10HX is illegal since it is 10 characters long, and -
2. the expression consists of an element of an array, A(1,5).

DOCUMENT CLASS IMS PAGE NO. 2-62  
PRODUCT NAME I700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES I700

However, as a result of scanning, the TYPE subroutine encounters a numeric field terminated with an H. If the value of the numeric field is less than some arbitrarily large number {in TYPE it is 400} an assumption is made by TYPE that it has encountered a FORMAT statement with a Hollerith Field Specification. Therefore, TYPE counts characters according to the number preceding the "H" and beginning with the character following the "H".

In order to count characters, the TYPE subroutine sets the switch JBLANK to 1. It then calls the GETC routine the number of times specified by the numeric field preceding the "H". Each time GETC is called, the column counter is increased by 1. Since {JBLANK} = 1, spaces in the field, following the "H" are regarded as significant characters. When the end of the Hollerith string is encountered, JBLANK is reset to 0 and the column counter is positioned for the character following the Hollerith string.

When slewing the Hollerith string, the TYPE subroutine does not recognize the equal sign it encounters. Therefore, it will not erroneously regard a FORMAT statement as a replacement.

#### 2.2.2.2.3.2 Other Statements

The TYPE subroutine attempts to identify statements other than replacements by searching the ISTN table. An entry in the ISTN table contains the name of a statement stored 1 character to a word, the characters represented by the Internal FORTRAN code. The name of the statement is followed by the type number for the statement, and preceded by a pointer to the start of the next entry in the ISTN table. If the space occupied by the ISTN table is regarded as a one dimensional array, the pointer to the start of the next table entry may be regarded as a subscript with which to reference the element {or word} which is the start position of the next ISTN entry. If the name of a statement has K characters, the ISTN table entry has K+2 elements {or words}. The 1st ISTN table entry is as follows:

{ISTN{1}} = 12    pointer to next table entry; next entry begins at ISTN{12}.

CONTROL DATA CORPORATION  
LA JOLLA RESOURCE CENTER \_\_\_\_\_ DIVISION

DOCUMENT CLASS IMS PAGE NO. 2-63  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

D  
I  
M  
E name  
N  
S  
I  
O  
N  
O type number; type number is stored  
at {ISTN{1}}-1.

The 1st ISTN table entry occupies storage reserved for elements ISTN{1} through ISTN{11}. The second ISTN entry begins at ISTN{12}.

The final ISTN table entry is for the end statement. It appears as follows:

341  
E  
N  
D  
32 = type number

If the flag {ISPEC} = 0 enabling PHASEA to process specification statements, the entire ISTN table is searched in order to identify the source statement. If {ISPEC} = 1 indicating that no subsequent specification statements will be processed, the search of the ISTN table begins with the 1st entry containing a non-specification statement name.

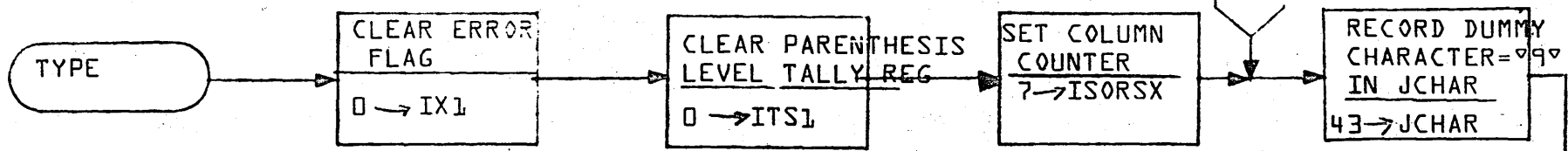
If the TYPE subroutine is able to identify a source statement as a result of an ISTN table search, the type number is read from the last word of the matching entry, placed in IBUF2{3}. An exit is made from TYPE.

If the entry containing the END statement is encountered and no match is made, an error flag is set by -

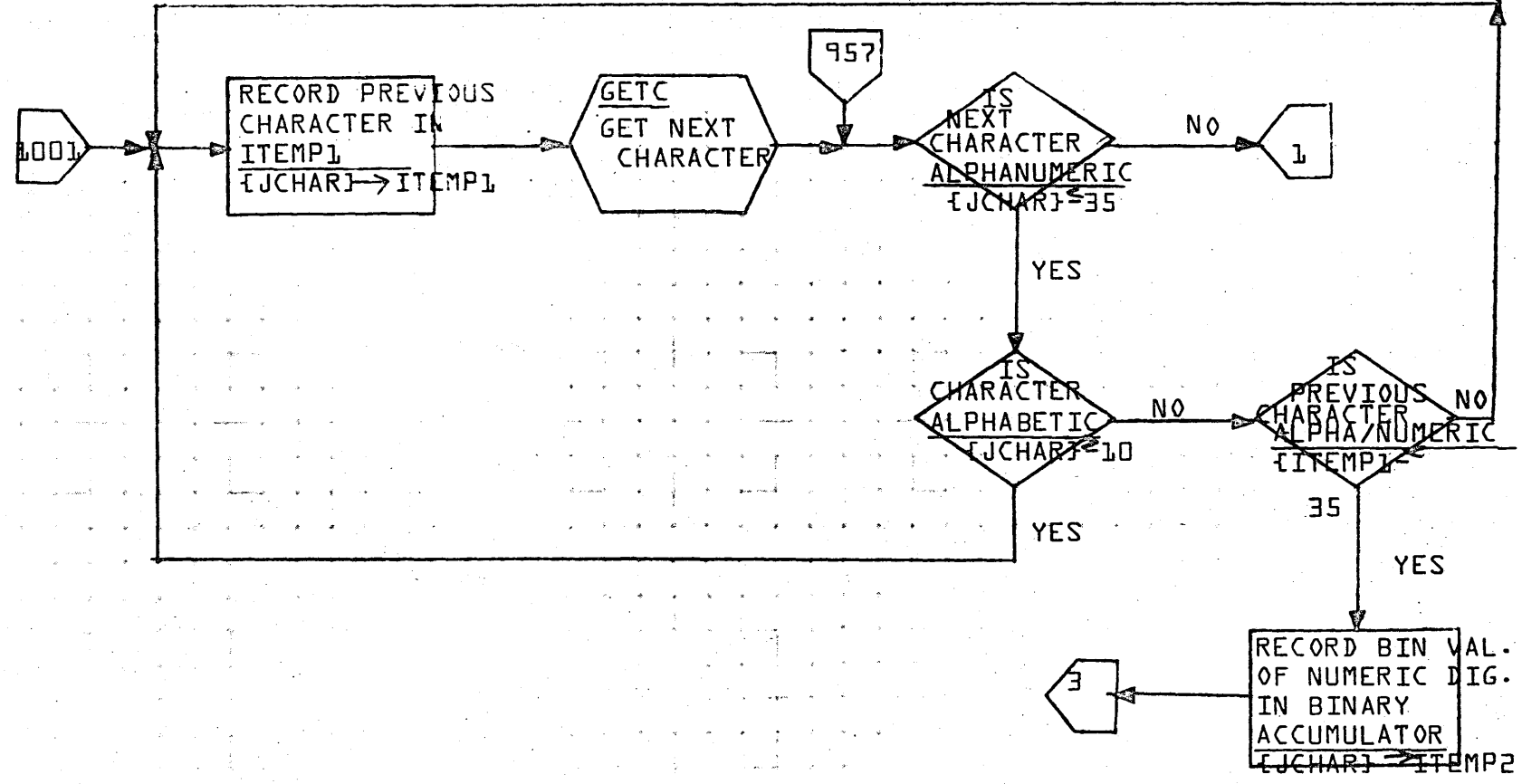
1 → IX1,

and an exit is made from the TYPE subroutine.

A



B



C

D

<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	TYPE	PAGE 1 OF 8		PROJECT MGR.			
	NUMBER	ISSUE DATE			PROJECT NAME			
	DRAWN BY	DATE			TASK NO.			
					TASK NAME			

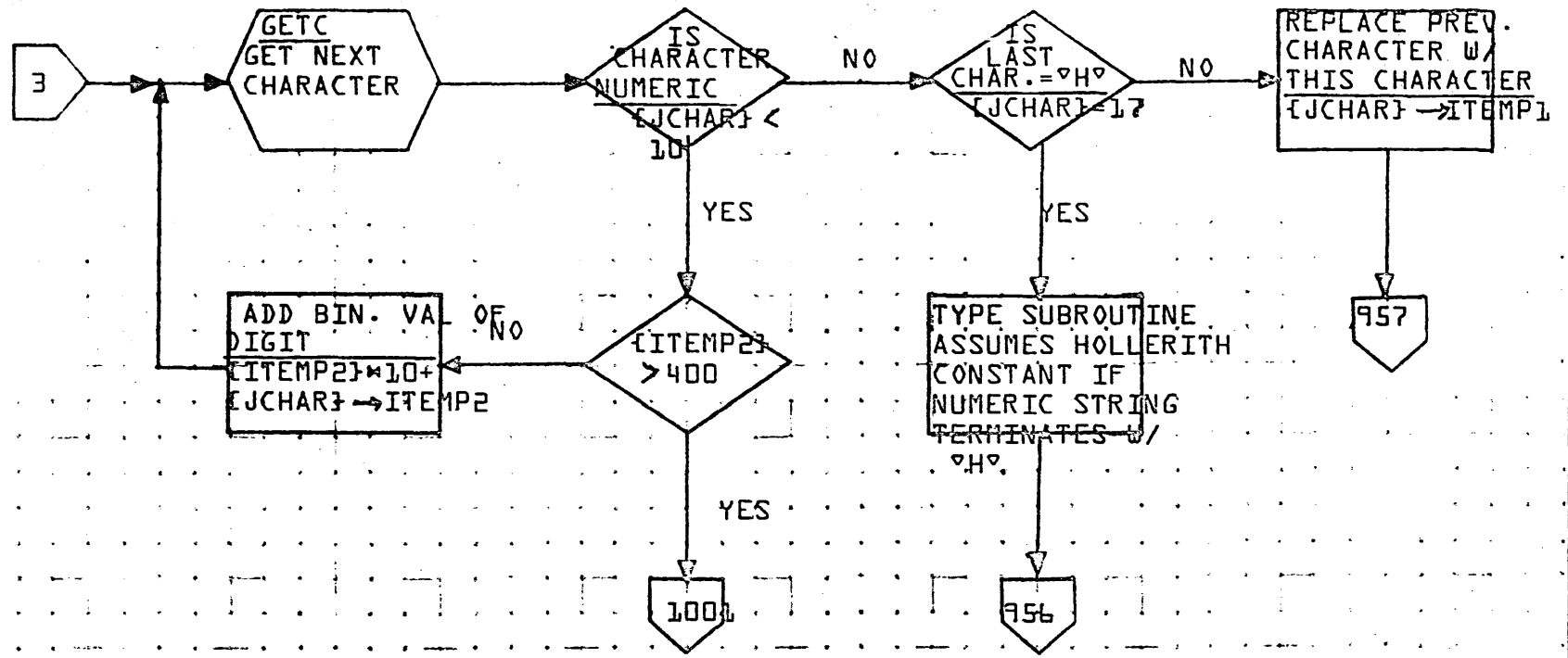
1

2

3

4

5

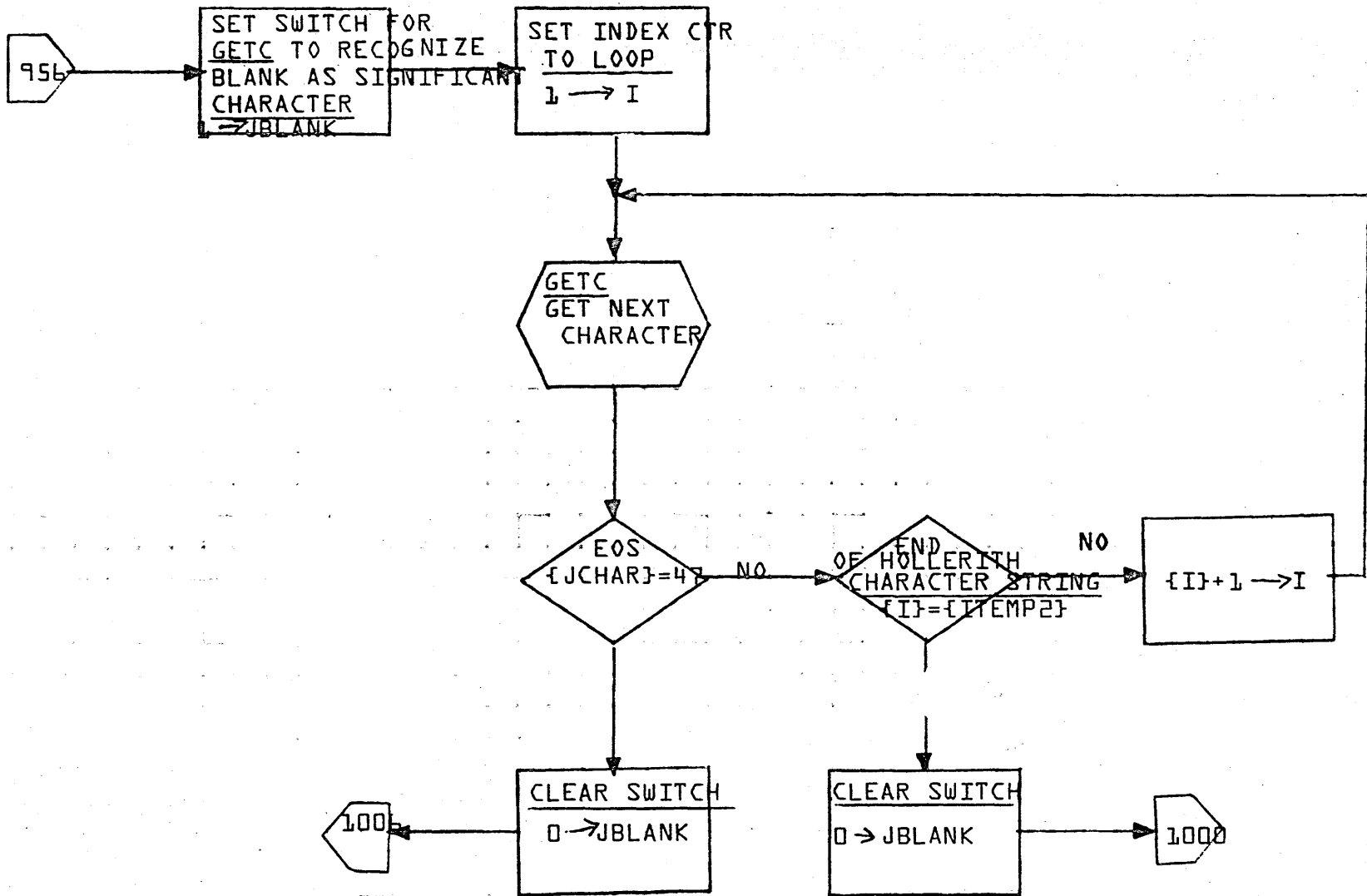


CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

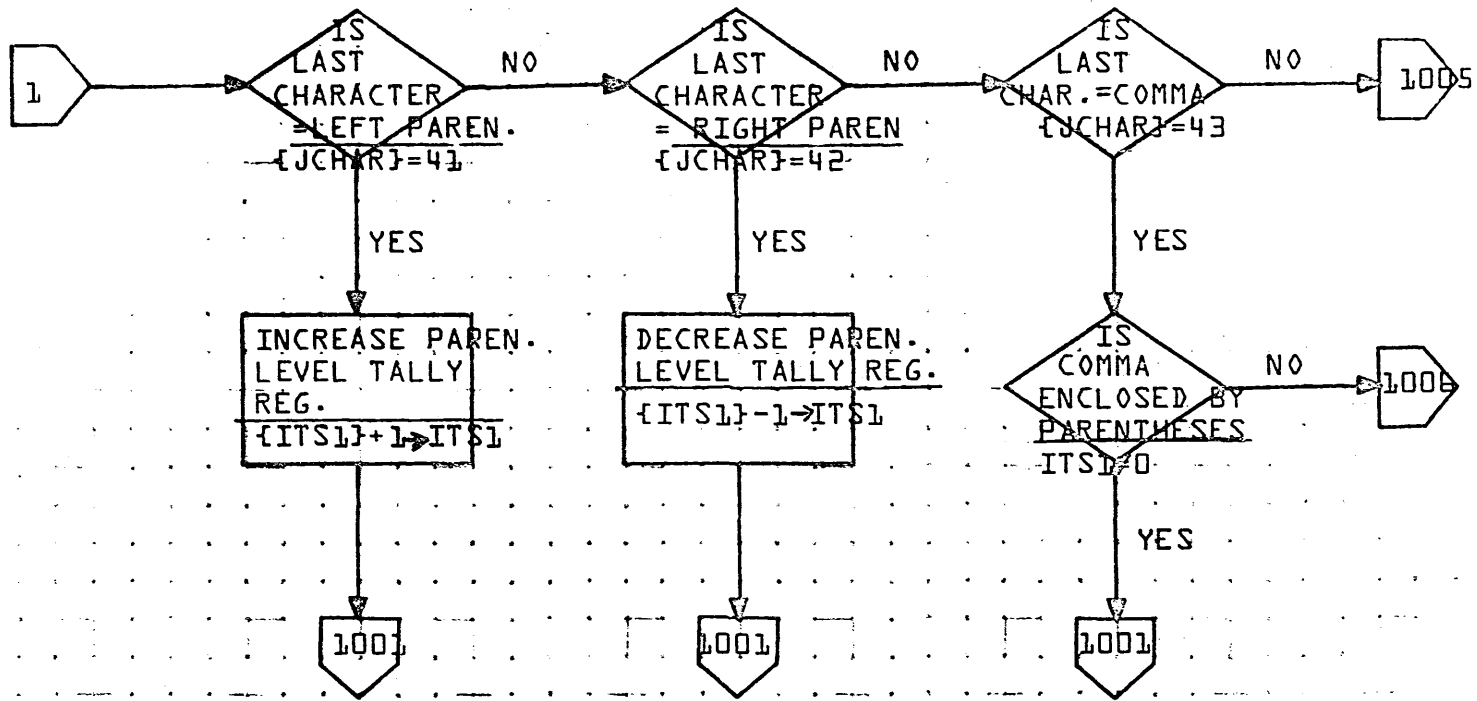
DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.		APPROVED	DATE
DOCUMENT TITLE	TYPE			PROJECT MGR.			
		PAGE	2 OF 8	PROJECT NAME			
NUMBER		ISSUE DATE		TASK NO.			
DRAWN BY		DATE		TASK NAME			

2-65



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	TMS	MACH. TYPE	1200	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	TYPE	PAGE 3 OF 8		PROJECT MGR.			
	NUMBER	ISSUE DATE			PROJECT NAME			
	DRAWN BY	DATE			TASK NO.			
					TASK NAME			



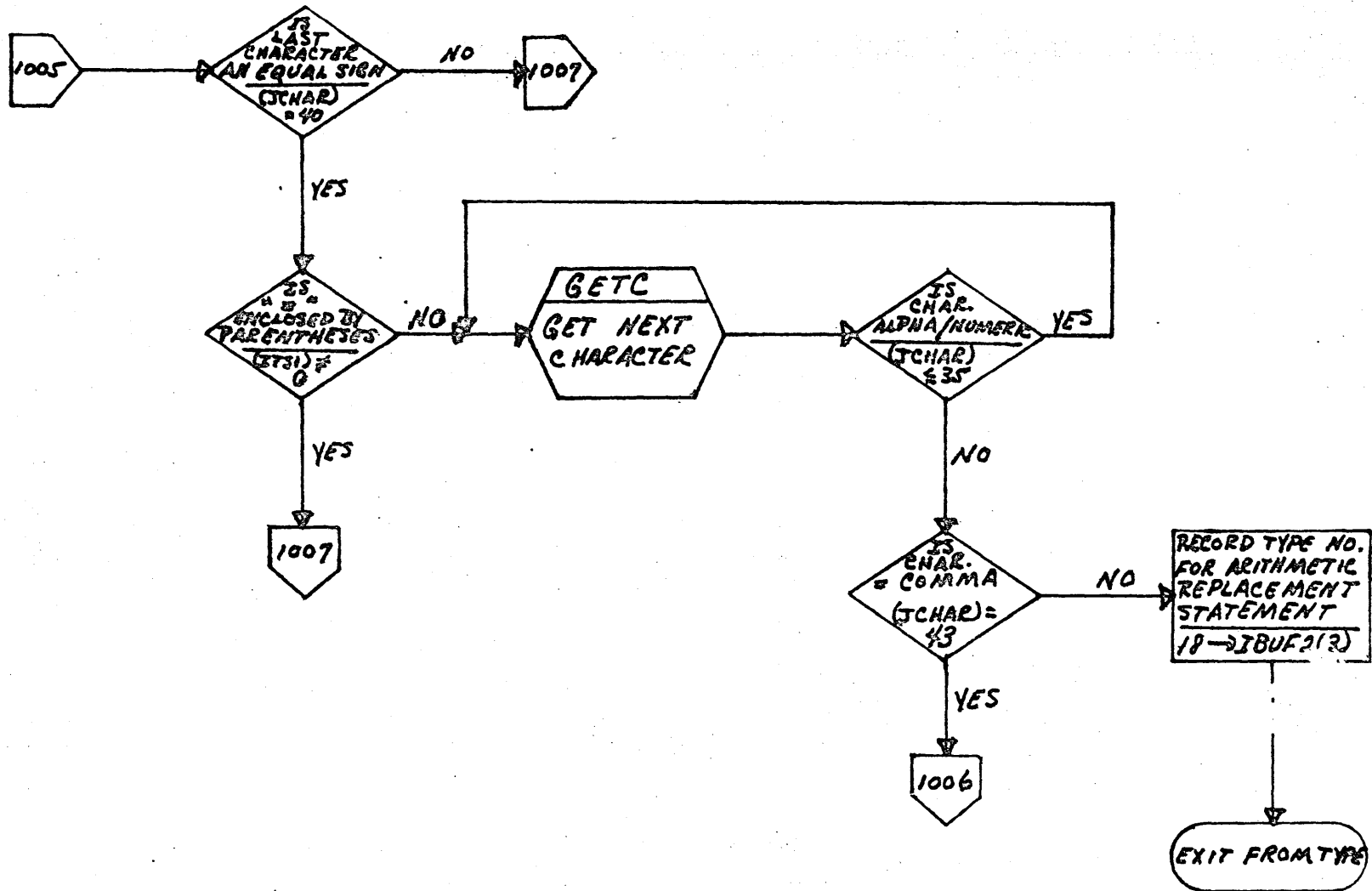


**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS <b>IMS</b>	MACH. TYPE <b>1700</b>	PROJECT NO.	REV.	APPROVED	DATE
DOCUMENT TITLE <b>TYPE</b>		PROJECT MGR.			
	PAGE <b>4</b> OF <b>8</b>	PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.			
DRAWN BY	DATE	TASK NAME			

2-67

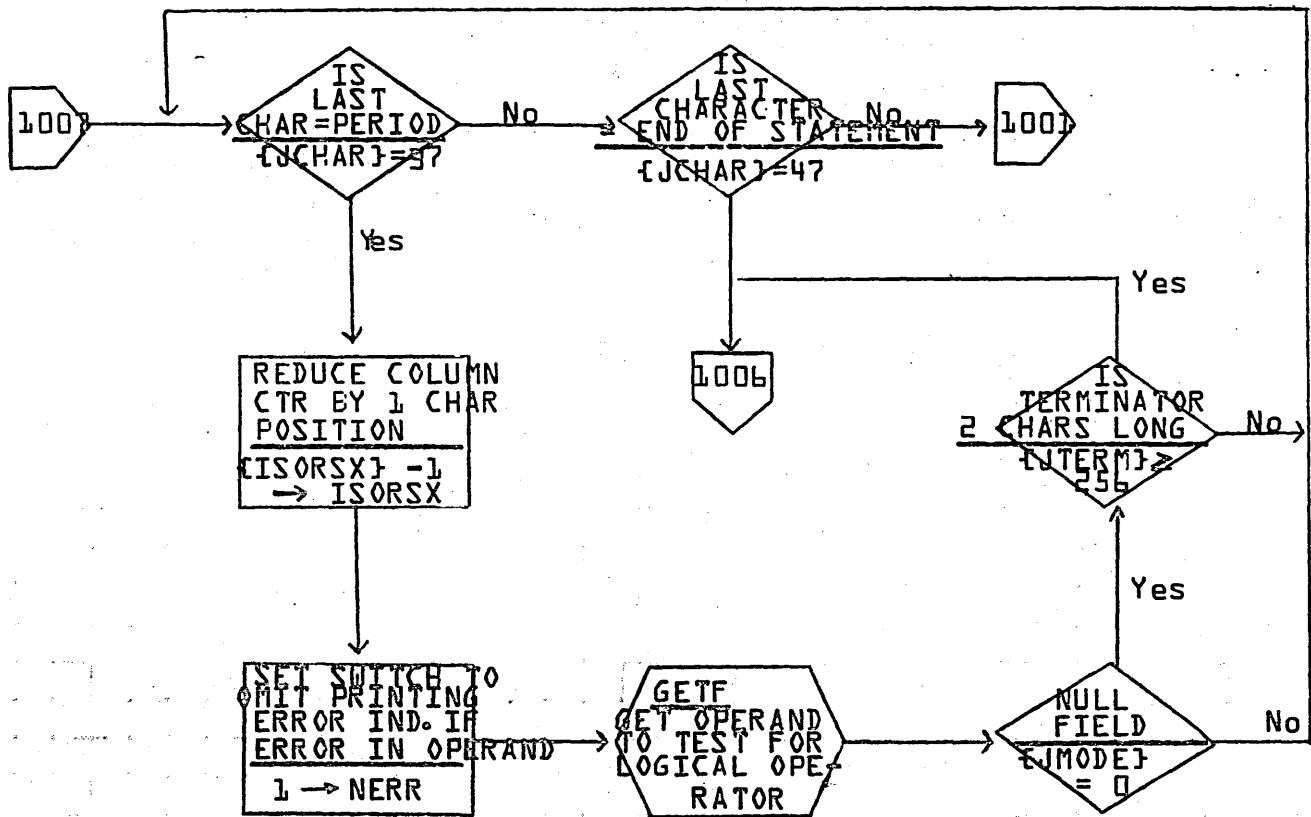


CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

DOCUMENT CLASS	<b>IMS</b>	MACH. TYPE	<b>1700</b>
DOCUMENT TITLE	<b>TYPE</b>		
PAGE <b>5</b> OF <b>8</b>			
NUMBER	ISSUE DATE		
DRAWN BY	DATE		

PROJECT NO.	REV	APPROVED	DATE
PROJECT MGR.			
PROJECT NAME			
TASK NO.			
TASK NAME			



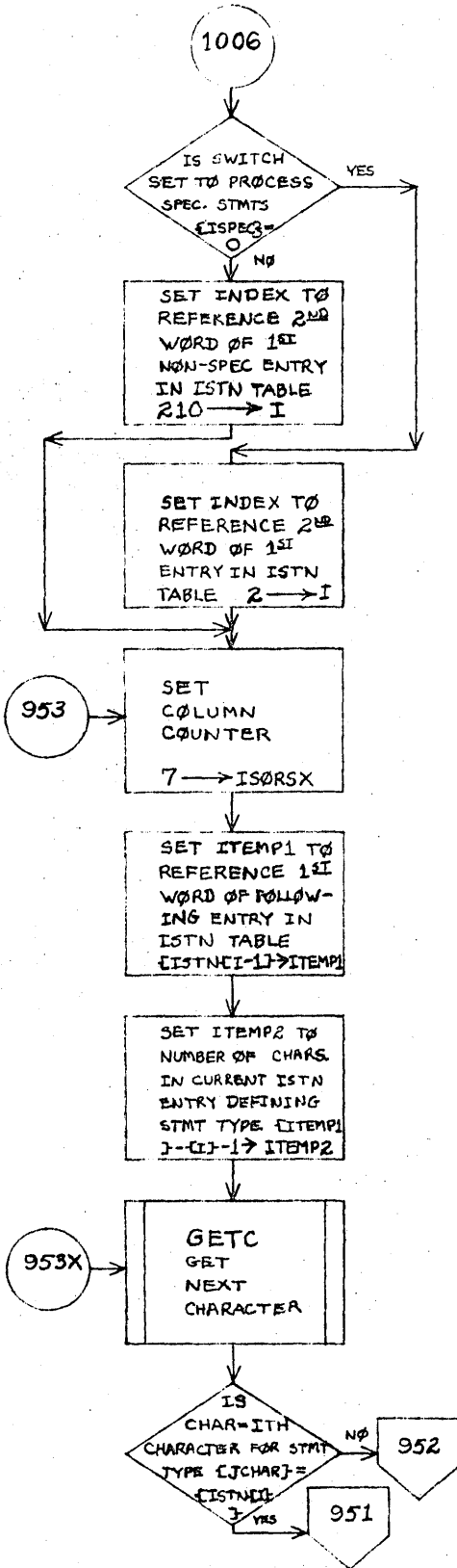
**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

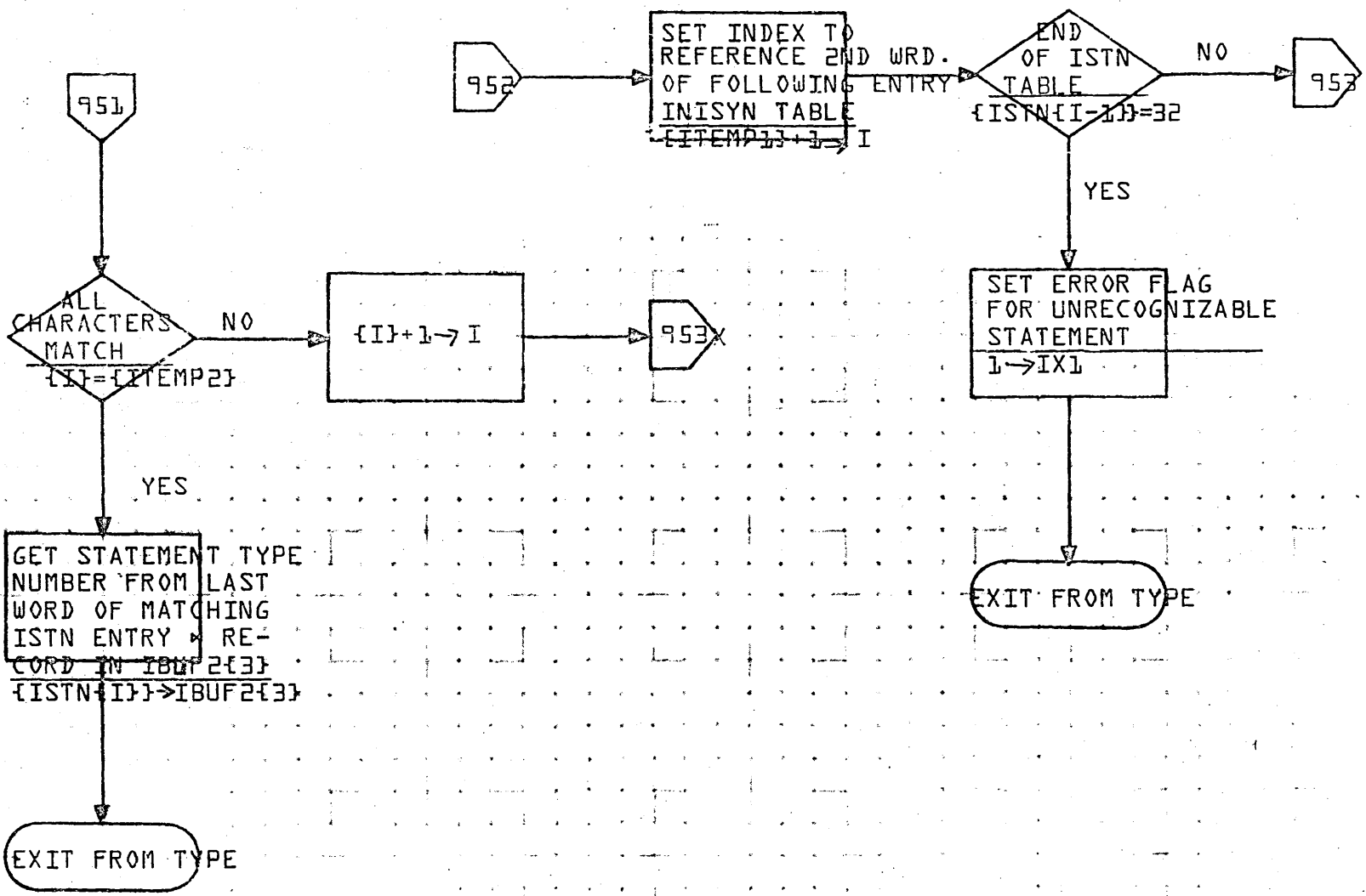
DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	TYPE			PROJECT MGR.			
PAGE 6 OF 8				PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.					
DRAWN BY	DATE	TASK NAME					



LA JOLLA FACILITY



TITLE		TYPE		DRG. NO.	
DRAWN BY		PROJ.		REVISION	
DATE		SHEET		OF	



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE

FLOWCHART

DECISION TABLE

OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700
DOCUMENT TITLE	TYPE		
PAGE A OF A			
NUMBER	ISSUE DATE		
DRAWN BY	DATE		

PROJECT NO.	REV	APPROVED	DATE
PROJECT MGR			
PROJECT NAME			
TASK NO.			
TASK NAME			

DOCUMENT CLASS IMS PAGE NO. 2-72  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. COO5\*3.1 A/B MACHINE SERIES 1700

2.2.2.3 PEQVS - The Equivalence Table Builder

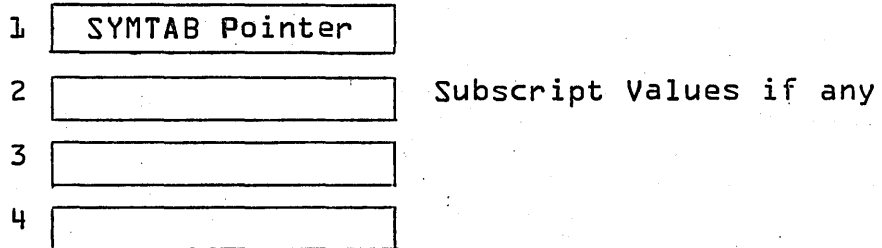
This routine is called by Phase A upon encountering an executable statement or the END line. EQUIVALENCE statements, which have been saved, in a different form, in the buffer KEQV, are merged into equivalence classes and stored, in a shorthand form, in the equivalence table KEQV. Each entry in KEQV is four words. A symbol table pointer, followed by three words containing subscript values, if any.

This routine has three logical parts or phases. In the first part it checks the subscripts for validity, and calculates a single value for the subscript, if any.

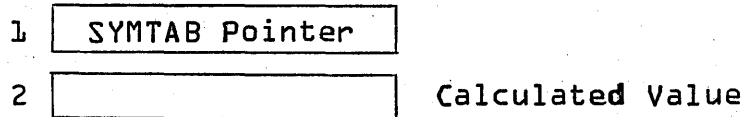
In the second part, all equivalenced variable or array parameters {called a chain} plus the single calculated value are moved into an intermediate table MEQV by chain. Each entry in MEQV has two words, a pointer to the name in the symbol table, and the single calculated value.

In the third part, each chain is sorted by increasing increment size, and the ordered chains are then moved to IEQV. Each entry in IEQV is two words and has the same format as MEQV.

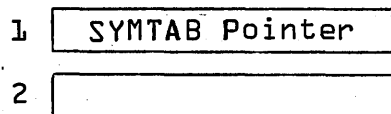
KEQV



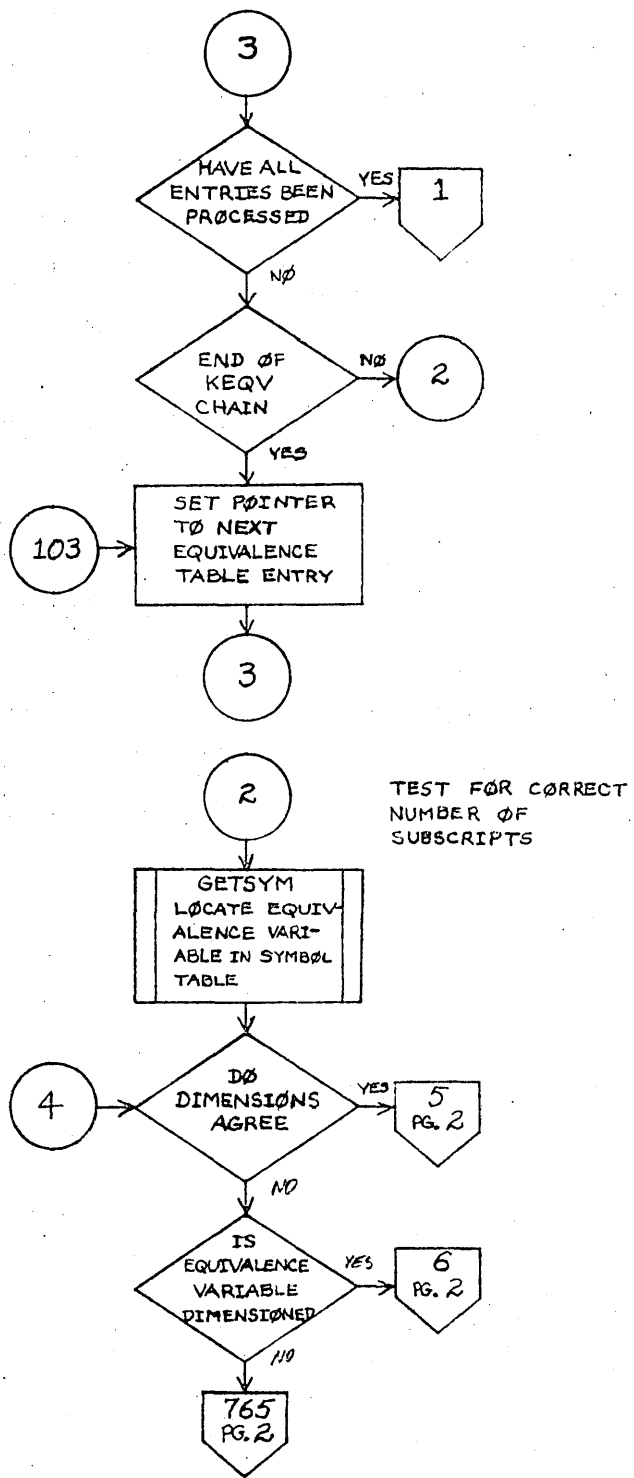
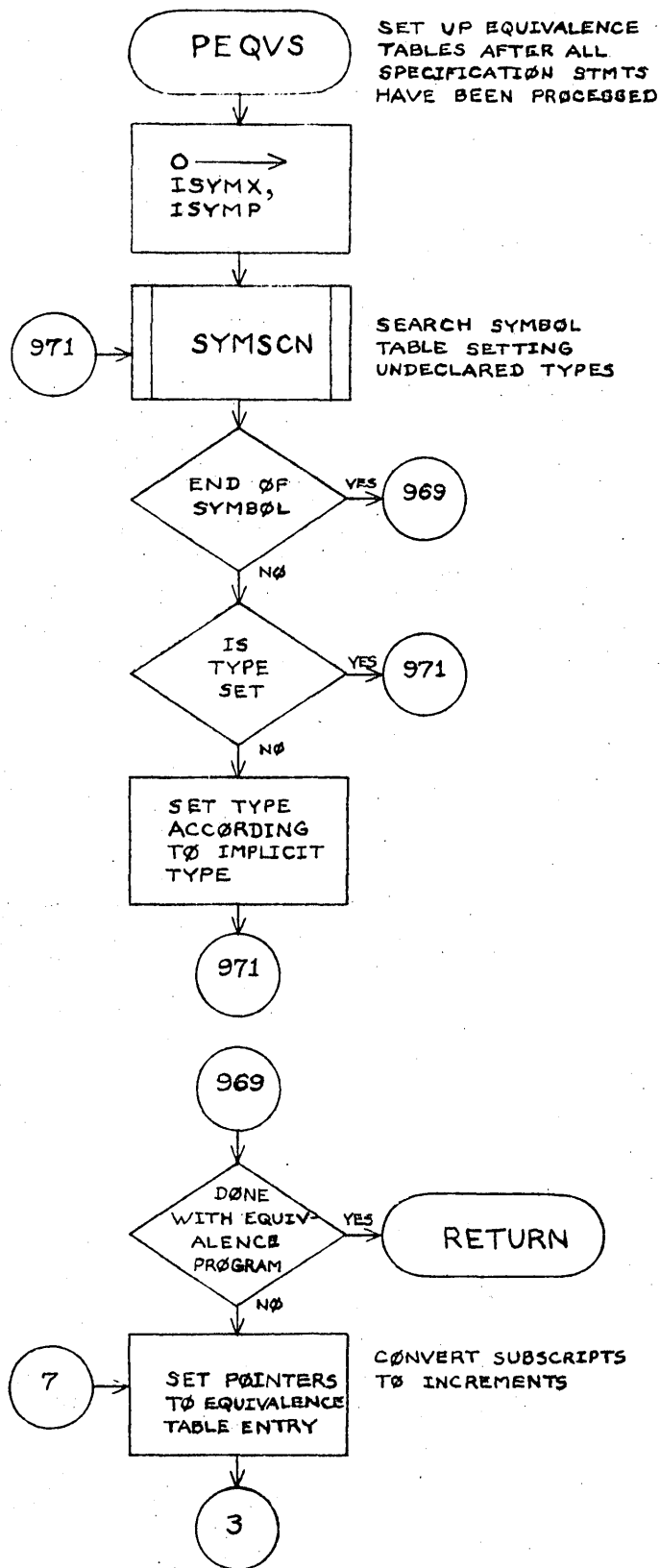
MEQV



IEQV {Sorted}



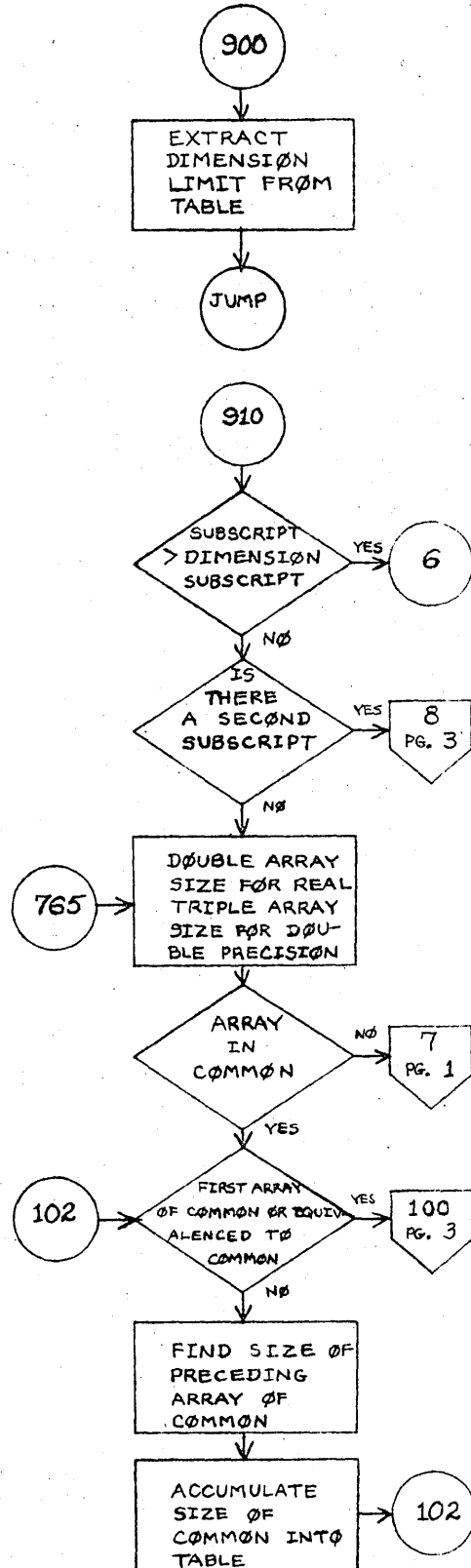
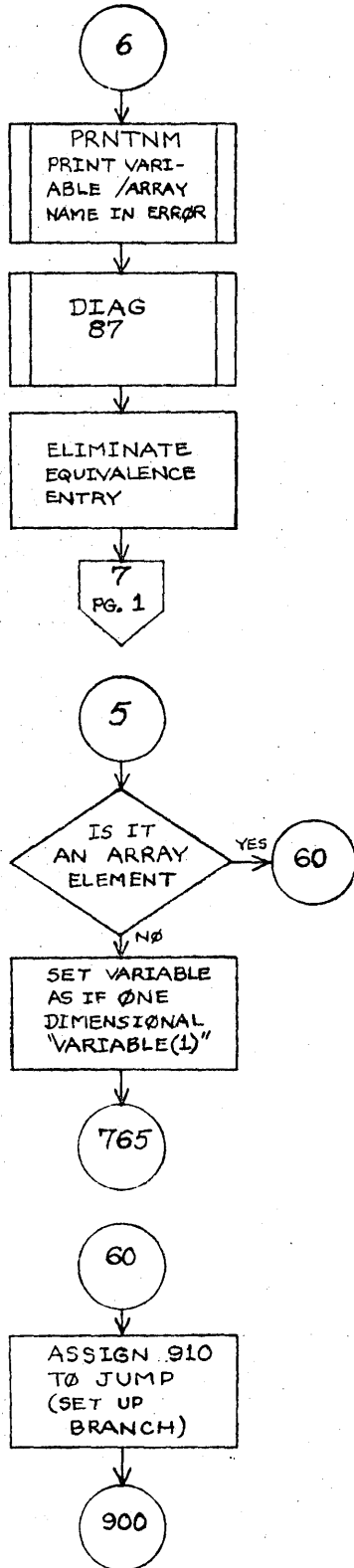
LA JOLLA FACILITY



TITLE		PEQVS		DRG. NO.	
DRAWN BY		PROJ.		REVISION	
DATE		SHEET 1		OF	



LA JOLLA FACILITY

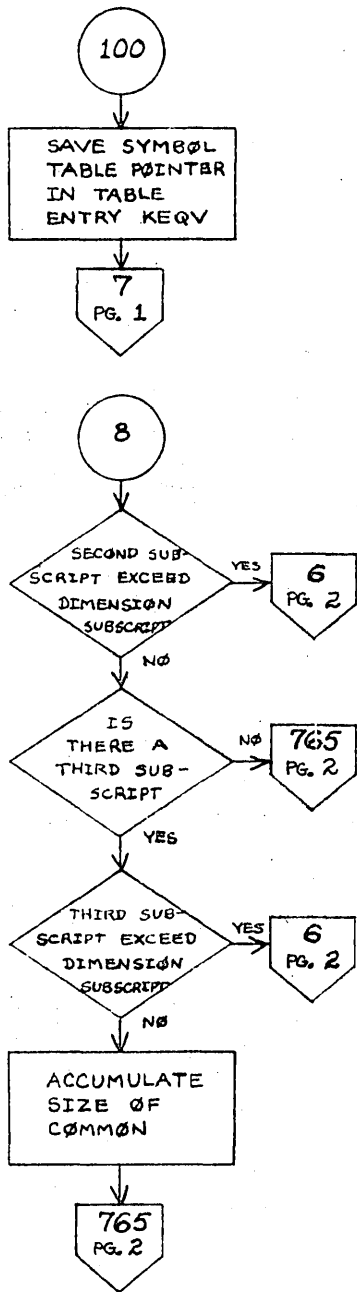


TITLE		PEQVS		DRG. NO.	
DRAWN BY		PROJ.	DATE	REVISION	
SHEET		OF		100	





LA JOLLA FACILITY



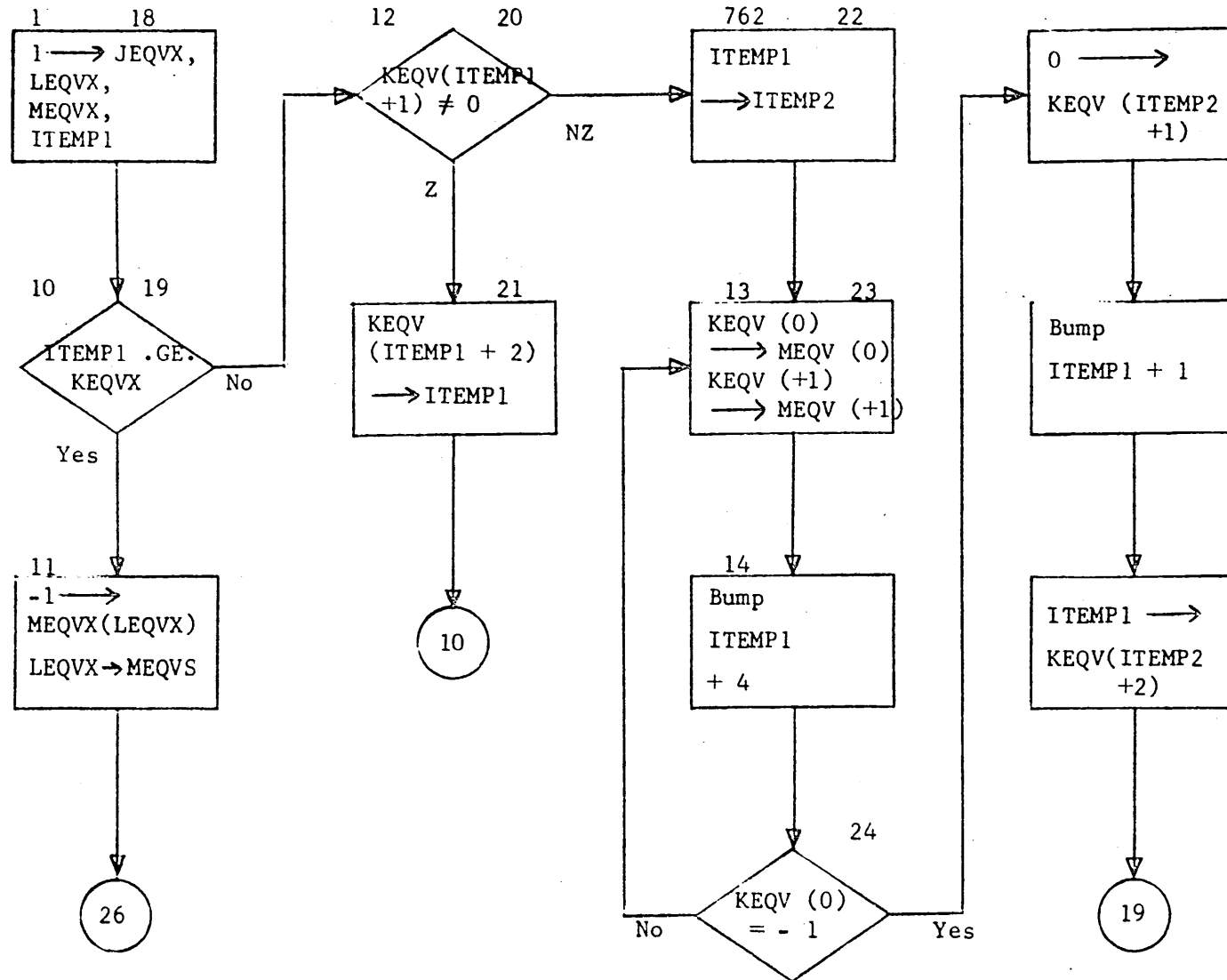
TITLE		PEQVS		DRG. NO.	
				REVISION	
DRAWN BY	PROJ.	DATE	SHEET	OF	

CONTROL DATA CORPORATION  
LA JOLLA RESOURCE CENTER DIVISION

DOCUMENT CLASS IMS PAGE NO. 2-7b  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

{THIS PAGE INTENTIONALLY LEFT BLANK}

18. Merge sets of equivalence chains into equivalence classes. Start by finding unused chain for new equivalence class.
19. Jump if end of entries to YES leg.
20. Jump if this chain has not previously been removed.
21. Move to next chain.
22. Move chain to MEQV, deleting illegal entries.
23. ITEMP1 is index to KEQV, LEQVX is index to MEQV.
24. Back to 13 if not end of chain.



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE

FLOWCHART

DECISION TABLE

OTHER

DOCUMENT CLASS	<i>IMS</i>	MACH. TYPE	<i>1700</i>	PROJECT NO.		REV		APPROVED	DATE
DOCUMENT TITLE	<i>PEQVS</i>			PROJECT MGR.					
		PAGE	<i>5 of 16</i>	PROJECT NAME					
NUMBER		ISSUE DATE		TASK NO.					
DRAWN BY		DATE		TASK NAME					

25. Jump if there are MEQV class elements left.

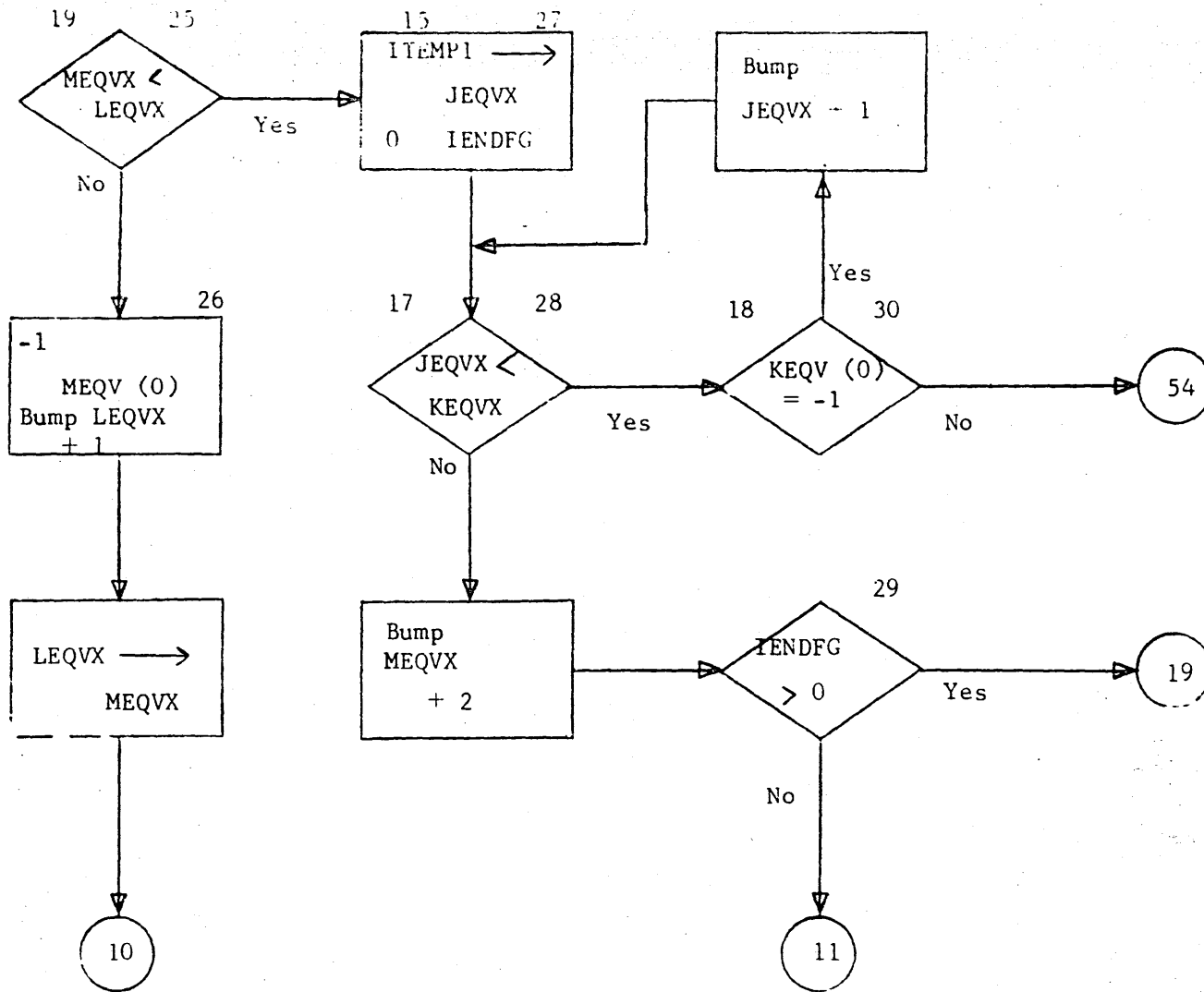
26. Set end of class mark.

27. Try to find equivalent chain, equivalence established by MEQV.

28. Go to 18 if not end of entries.

29. Jump to 19 if something left in KEQV.

30. Jump to 54 if not end of chain.



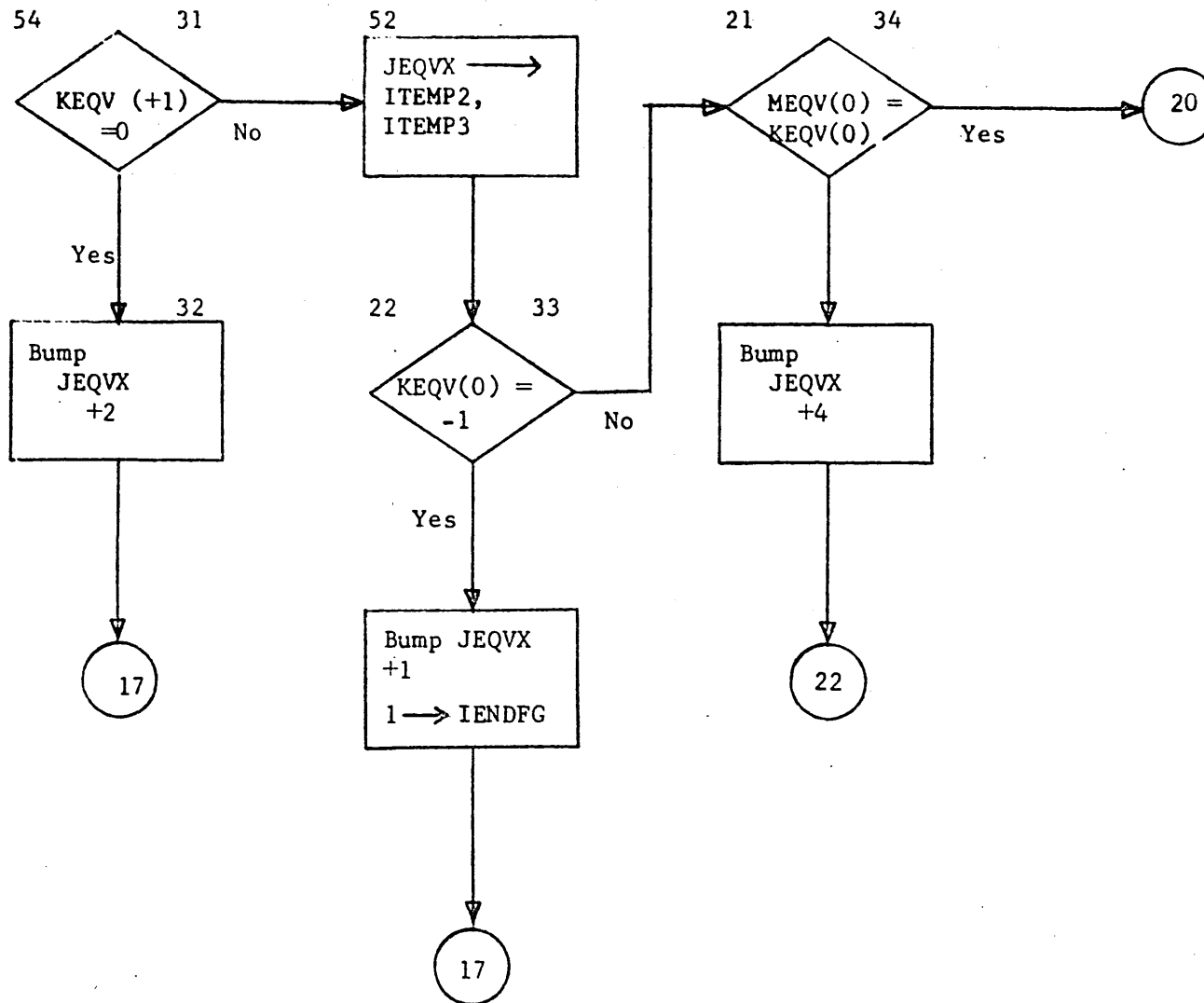
CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE   
FLOWCHART   
DECISION TABLE   
OTHER

DOCUMENT CLASS	<i>IMS</i>	MACH. TYPE	<i>1700</i>
DOCUMENT TITLE	<i>REQV</i>	PAGE <i>6</i> OF <i>16</i>	
NUMBER	ISSUE DATE	TASK NO.	
DRAWN BY	DATE	TASK NAME	

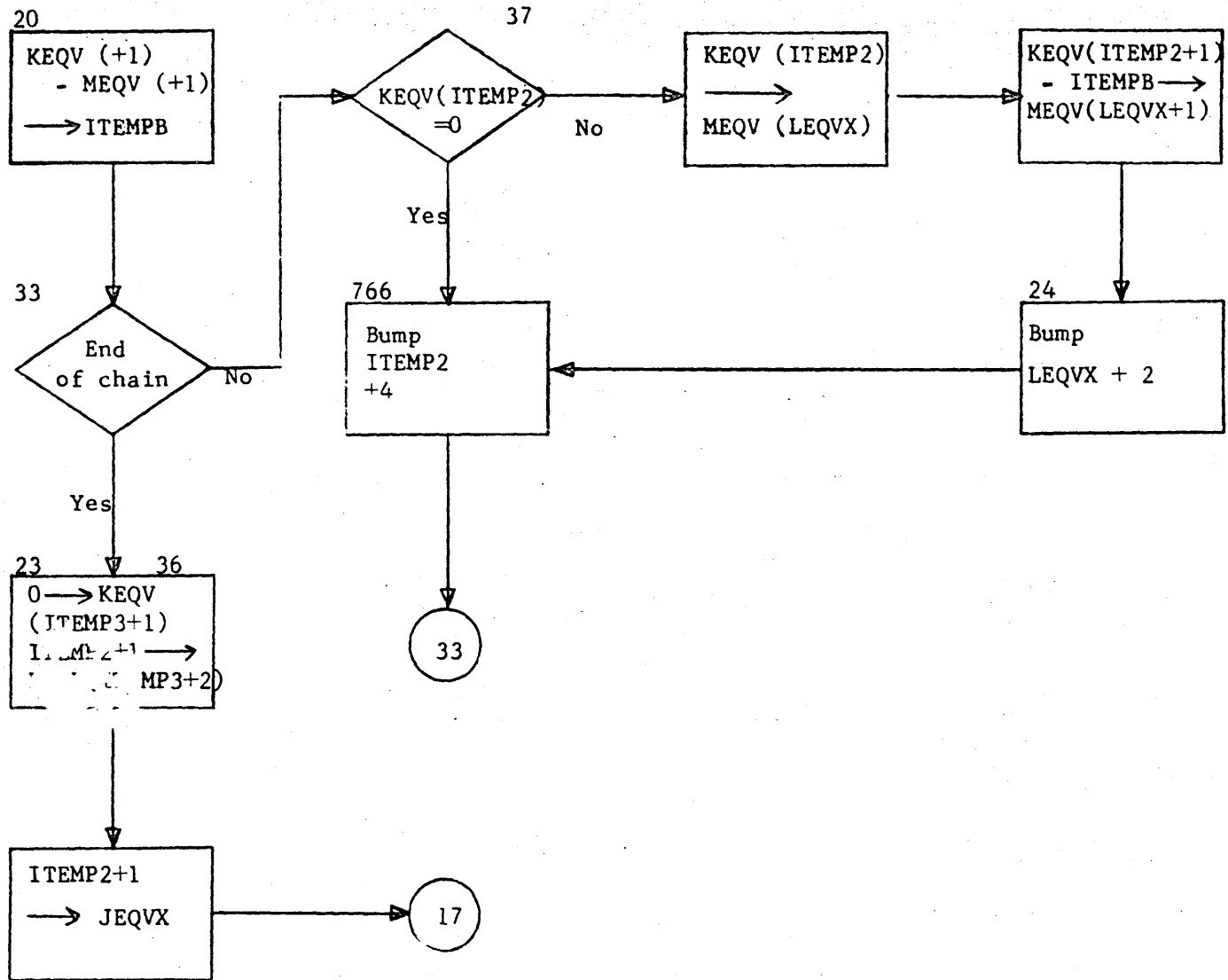
PROJECT NO.	REV	APPROVED	DATE
PROJECT MGR			
PROJECT NAME			
TASK NO.			
TASK NAME			

- 31. Jump to 52 if chain has not been assigned to class.
- 32. Jump to next chain.
- 33. Jump to 21 if not end of chain.
- 34. Is MEQV (MEQVX) an element in this chain. Jump to 20 if so.



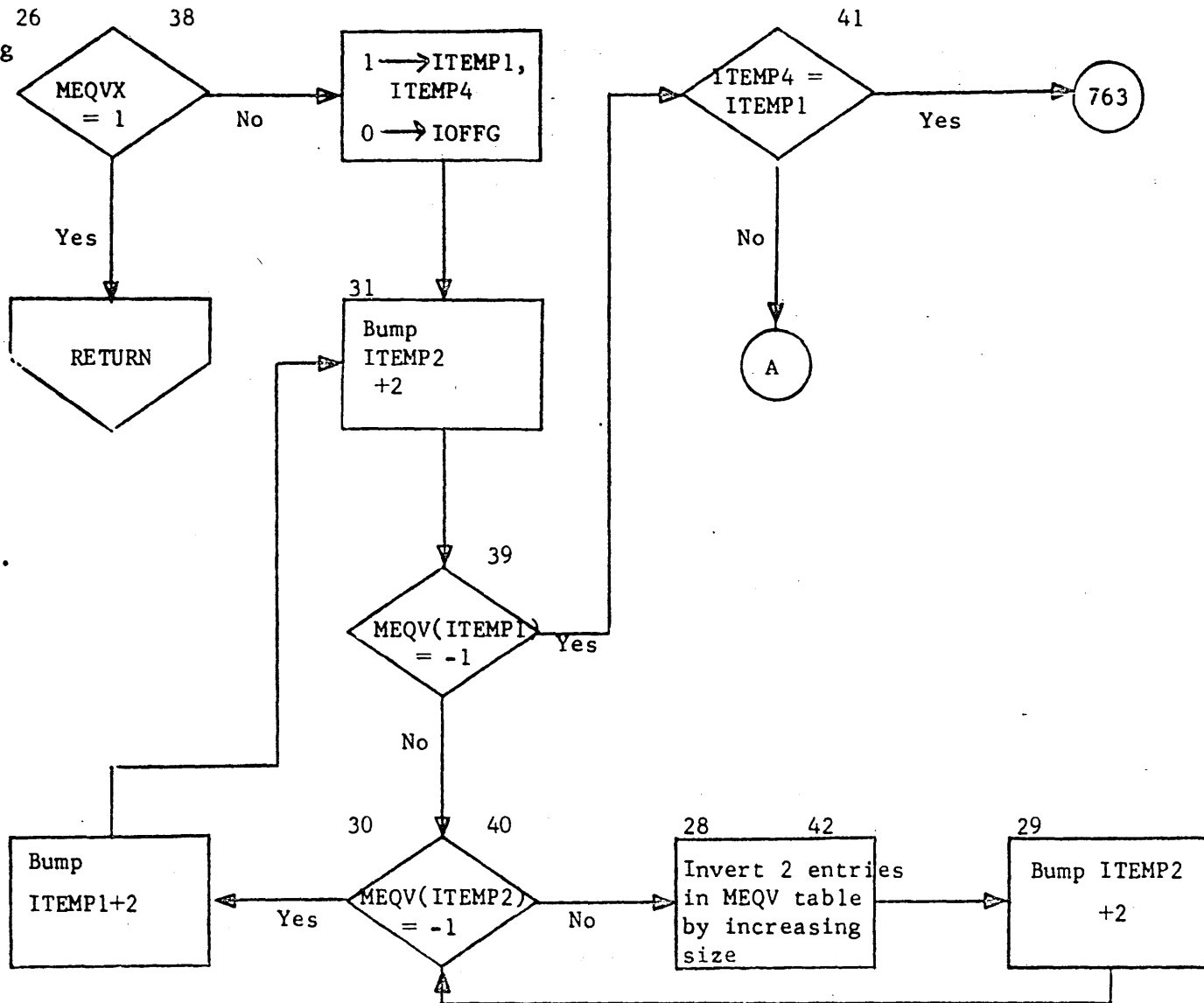
<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	MACH. TYPE	PROJECT NO.	REV.	APPROVED	DATE
	DOCUMENT TITLE		PROJECT MGR.			
		PAGE	PROJECT NAME			
	NUMBER	ISSUE DATE	TASK NO.			
	DRAWN BY	DATE	TASK NAME			

35. Integrate chain into partial class. Must adjust common increment. At this point KEQV indexed by JEQVX, MEQV indexed by MEQVX.
36. Remove this chain from KEQV.
37. Jump to 766 if illegal entry.



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH-TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	PEQV5	PAGE 8 OF 16		PROJECT MGR.			
	NUMBER	ISSUE DATE			PROJECT NAME			
	DRAWN BY	DATE			TASK NO.			
					TASK NAME			

- 38. Set up IEQV by ordering increments correctly and deleting illegalities. Sort class by increasing increment size.
- 39. Not end chain for primary scan, go to 30.
- 40. Not end of chain for secondary scan, go to 30.
- 41. Go to 763 if class empty.
- 42. ITEMP1 and ITEMP2 are pointers to MEQV entries to be inverted.



**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

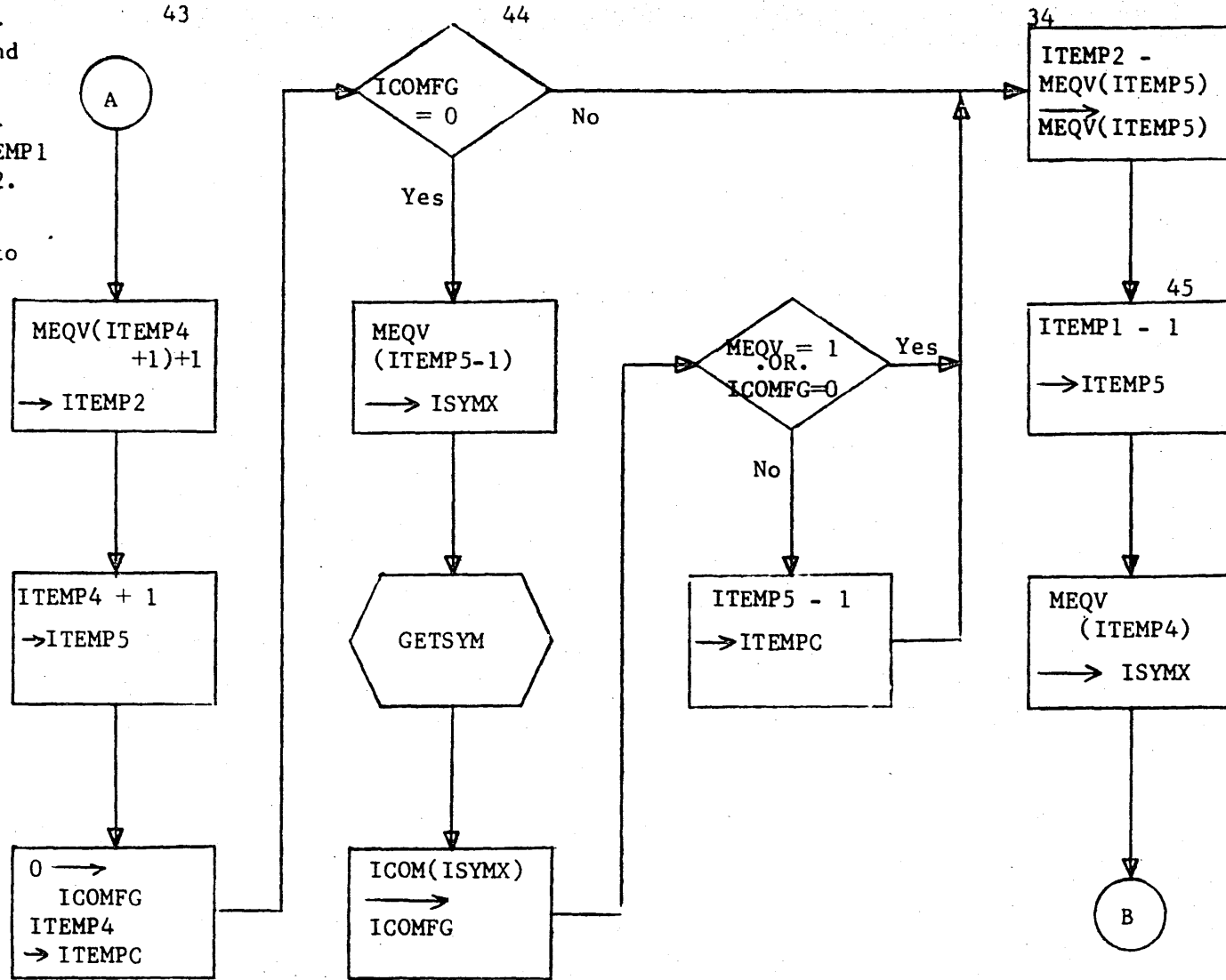
- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS <i>ZMS</i>	MACH. TYPE <i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE <i>PERIS</i>		PROJECT MGR.			
	PAGE <i>9 of 16</i>	PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.			
DRAWN BY	DATE <i>3</i>	TASK NAME			

43. Set up class as increment over base and find first common block.

44. DO loop from to statement 34, ITEMP5 to ITEMP1 times incremented by 2.

45. Compare all pairs for legality and enter into IEQV.



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE

FLOWCHART

DECISION TABLE

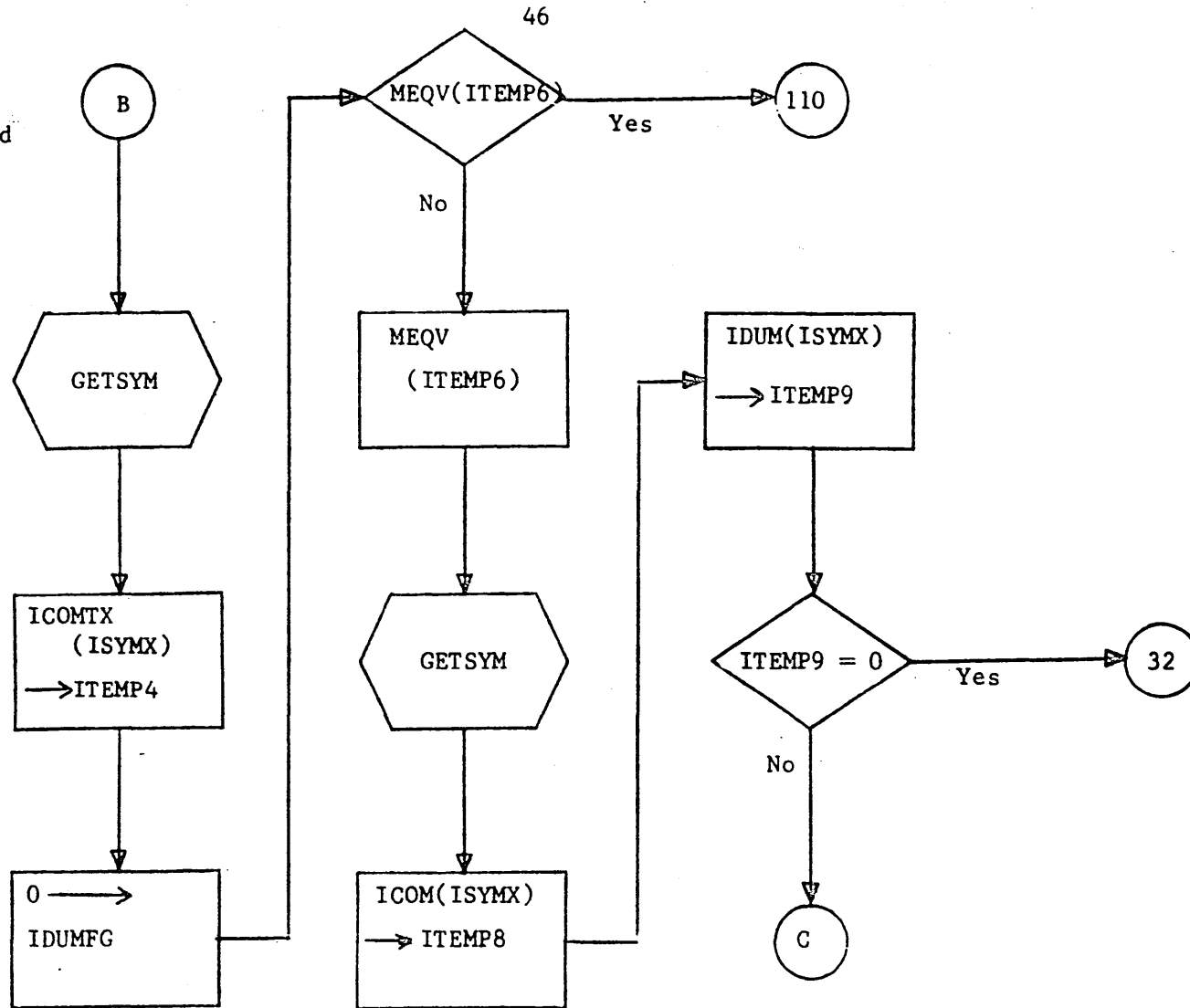
OTHER

DOCUMENT CLASS <i>ZHS</i>	MACH. TYPE <i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE <i>REQVS</i>		PROJECT MGR.			
	PAGE <i>10</i> OF <i>16</i>	PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.			
DRAWN BY	DATE <i>3/6/66</i>	TASK NAME			

E-82



46. DO loop to 110.  
 ITEMP6 = ITEMPC,  
 ITEMP5,2. Jump if  
 pair has been deleted  
 to DO loop counter.



CONTROL DATA CORPORATION  
 SOFTWARE DOCUMENT

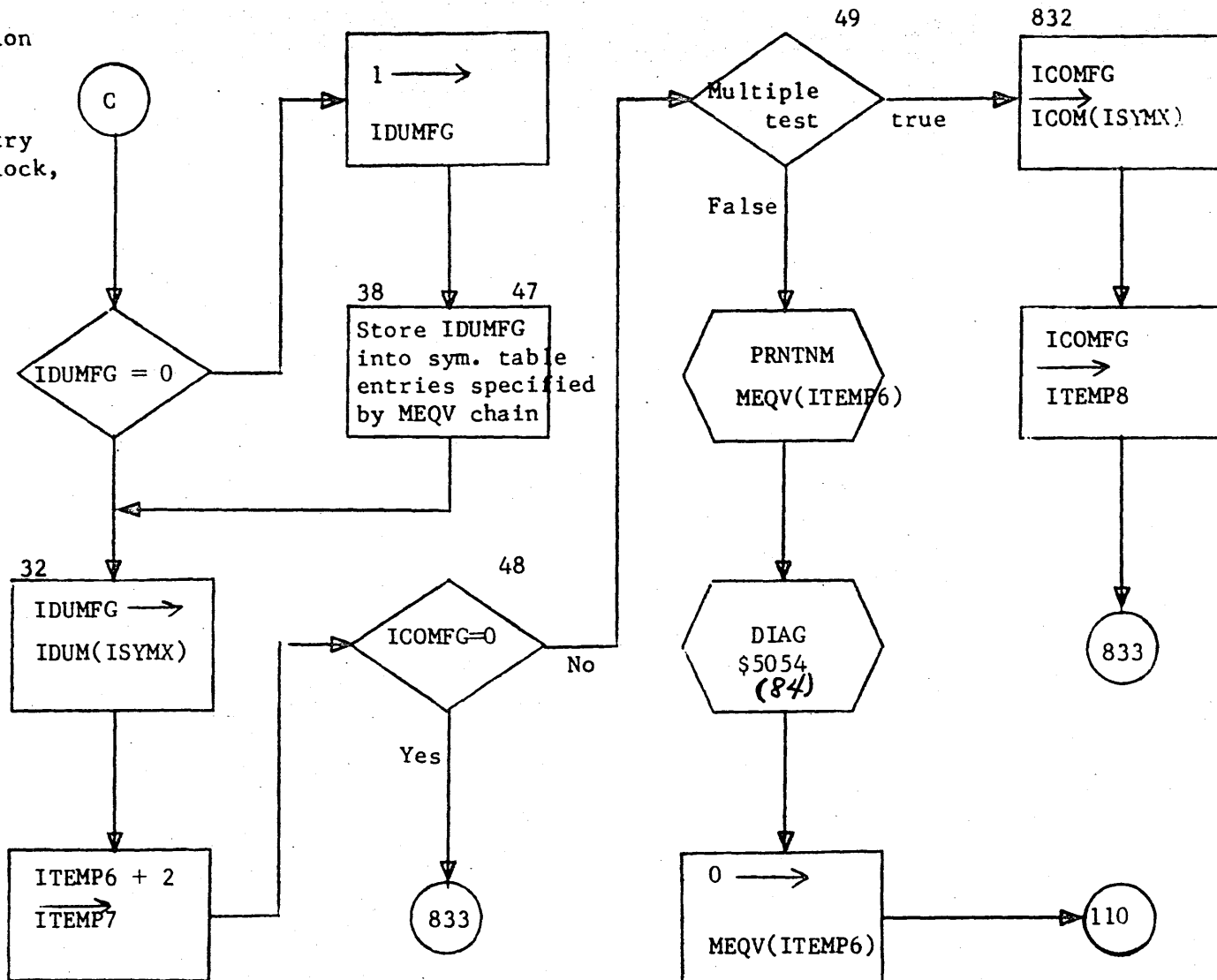
- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS <i>JMS</i>	MACH. TYPE <i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE <i>PEQ15</i>	PAGE <i>11</i> OF <i>16</i>	PROJECT MGR.			
NUMBER	ISSUE DATE	PROJECT NAME			
DRAWN BY	DATE <i>3-1-77</i>	TASK NO.			
		TASK NAME			

47. DO loop with induction variables ITEMPD = ITEMPC, ITEMP6, 2

48. Set symbol table entry to correct common block, if any.

49. Multiple test is:  
 ITEMP8 = 0  
 .OR.  
 ITEMP8 = ICOMFG  
 .AND.  
 ITEMP9 = 0



**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS <i>IMS</i>	MACH. TYPE <i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE <i>PERVS</i>		PROJECT MGR.			
	PAGE <i>12 of 16</i>	PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.			
DRAWN BY	DATE <i>5/1/71</i>	TASK NAME			

50. DO loop to 35 with induction variables:

ITEMP7=ITEMP7,.  
ITEMP5, 2.

Jump if item been deleted to 35.

51. Test is:

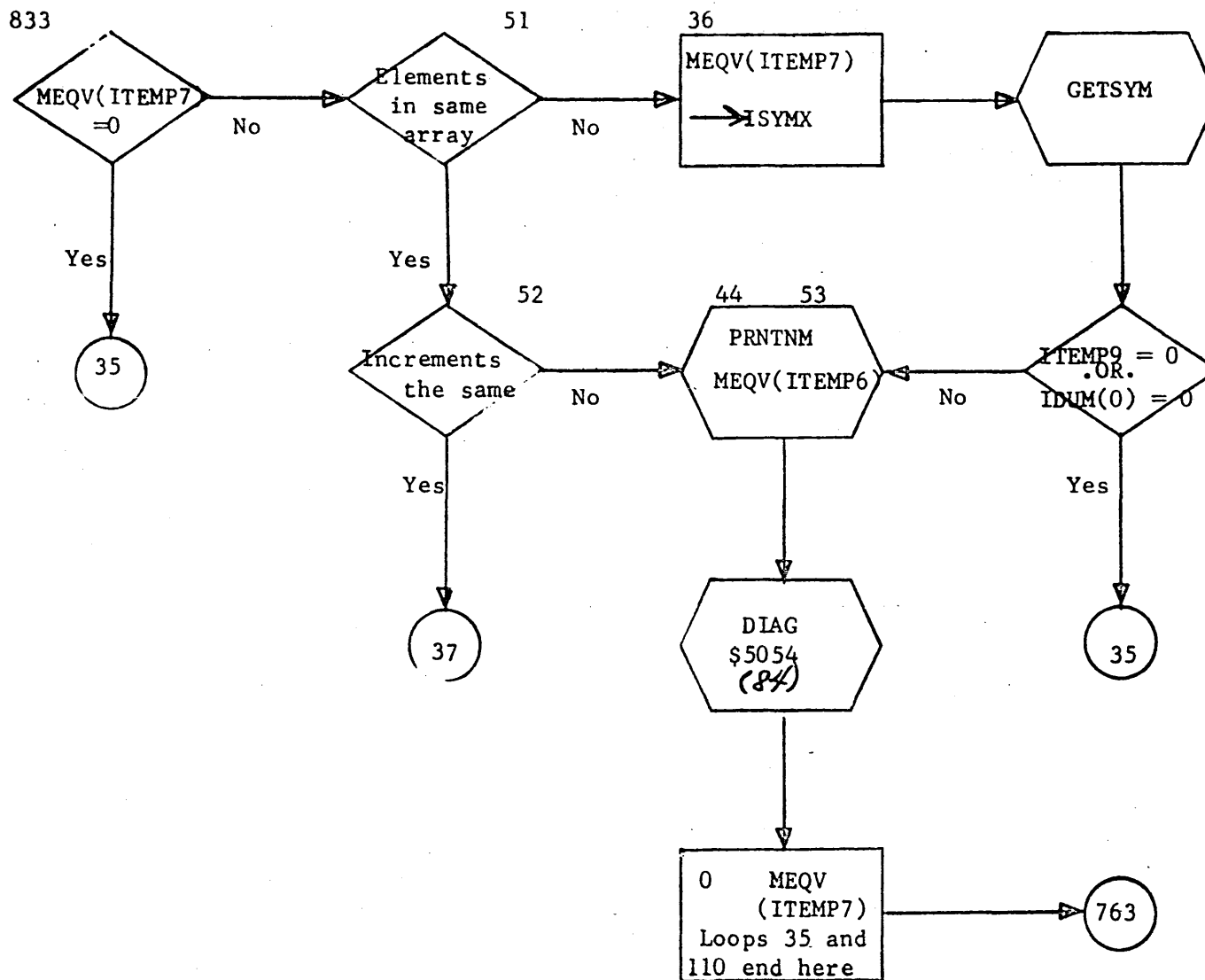
MEQV(ITEMP6) ≠ MEQV(ITEMP7) true means no.

52. Test is:

MEQV(ITEMP6+1) = MEQV(ITEMP7+1) true means yes.

53. Print out names for:

MEQV(ITEMP6) and MEQV(ITEMP7).

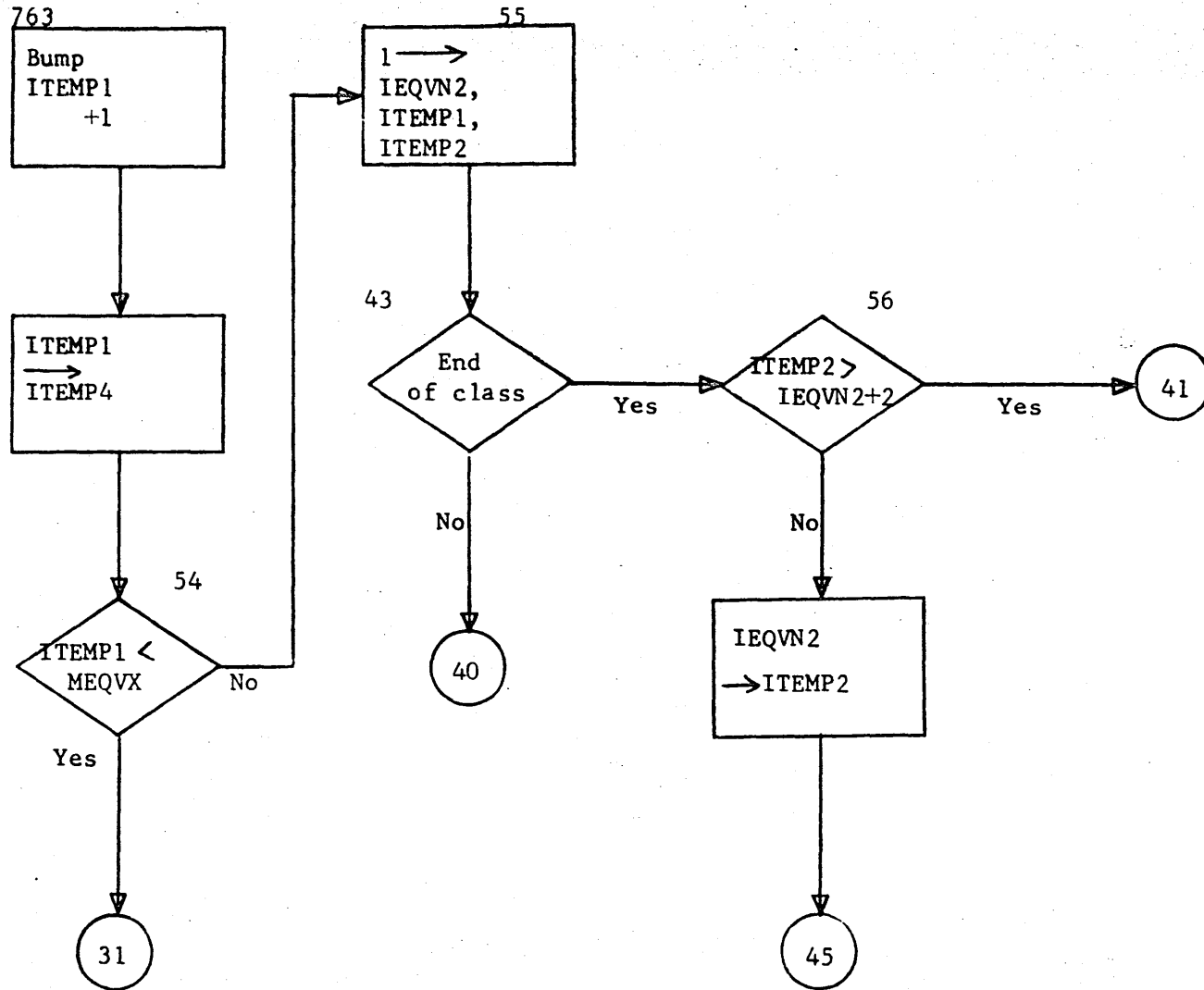


CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE   
FLOWCHART   
DECISION TABLE   
OTHER

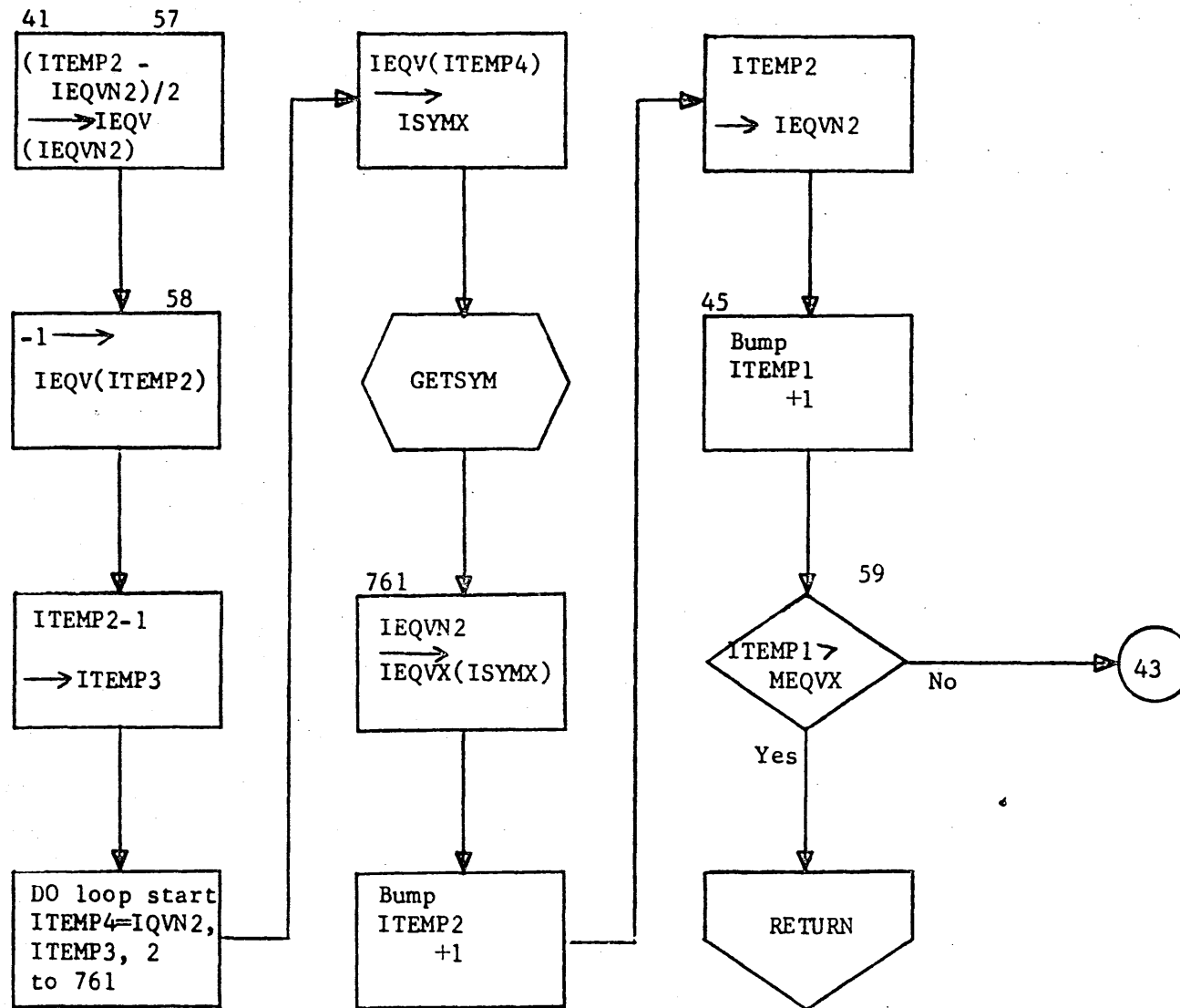
DOCUMENT CLASS <i>INS</i>	MACH. TYPE <i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE <i>PEQ15</i>		PROJECT MGR.			
	PAGE <i>13</i> OF <i>16</i>	PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.			
DRAWN BY	DATE <i>3/67</i>	TASK NAME			

- 54. Jump to 31 to process more classes.
- 55. Prepare to set up IEQV tables.
- 56. Jump to 41 if class contains two or more names.



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	MACH. TYPE	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE		PROJECT MGR.			
		PAGE 14 OF 67	PROJECT NAME			
	NUMBER	ISSUE DATE	TASK NO.			
	DRAWN BY	DATE = 6.7	TASK NAME			

57. Set up number of elements in this class.
58. Set end of class flag.
59. Return if all equivalence entries have been processed.

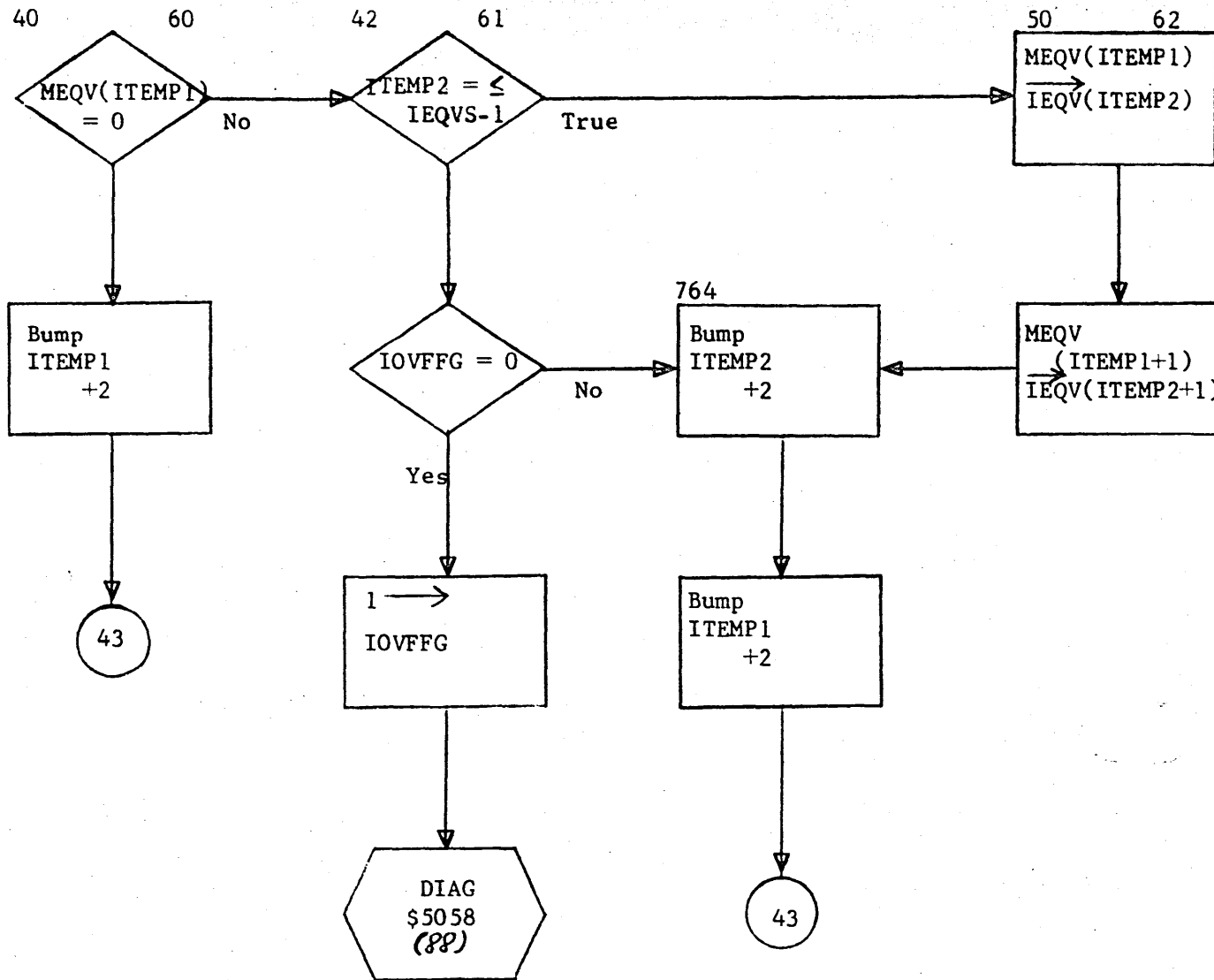


CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE   
FLOWCHART   
DECISION TABLE   
OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	REQS			PROJECT MGR.			
		PAGE	158 of 16	PROJECT NAME			
NUMBER		ISSUE DATE		TASK NO.			
DRAWN BY		DATE	3-11	TASK NAME			

- 60. Jump if entry is not to be deleted.
- 61. Check for table overflow. Jump if none.
- 62. Store name into equivalence table.



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS <i>IMS</i>	MACH. TYPE <i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE <i>PERVS</i>		PROJECT MGR.			
		PAGE <i>16</i> OF <i>16</i>	PROJECT NAME			
	NUMBER	ISSUE DATE	TASK NO.			
	DRAWN BY	DATE	TASK NAME			

DOCUMENT CLASS IMS PAGE NO. 2-89  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005 V.2.0 MACHINE SERIES 1700

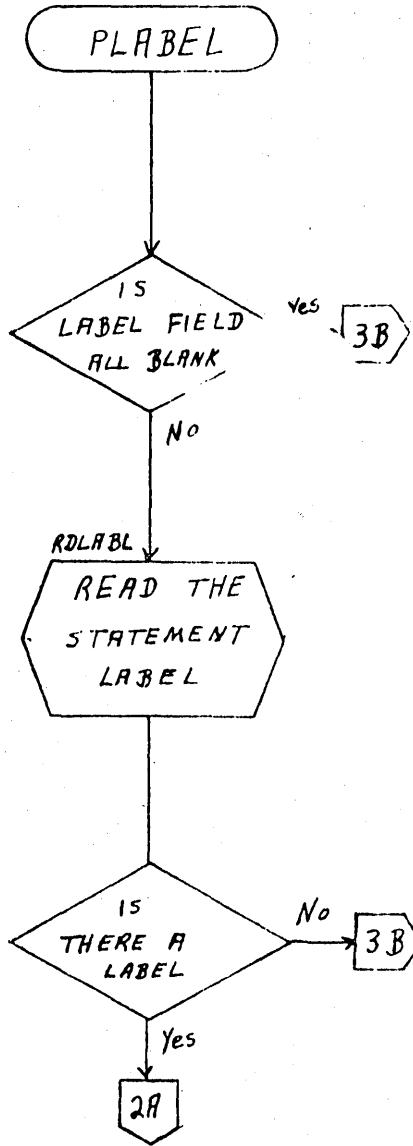
#### 2.2.2.4 Subroutine PLABEL

PLABEL is called by PHASEA to process the label field of an input statement. If the label field is blank, the routine returns immediately with ISORSX = 7.

Otherwise, it uses the routine RDLABL to read the field. If the field contains a label, a check is made to see if this label is currently in the symbol table (SYMTAB). If it is not, it is entered there by a call to STORE. If it is, a check is made to see if the label has been previously defined. If it has, an error is output and the routine exits with ILLABL = 0 and ISORSX = 7.

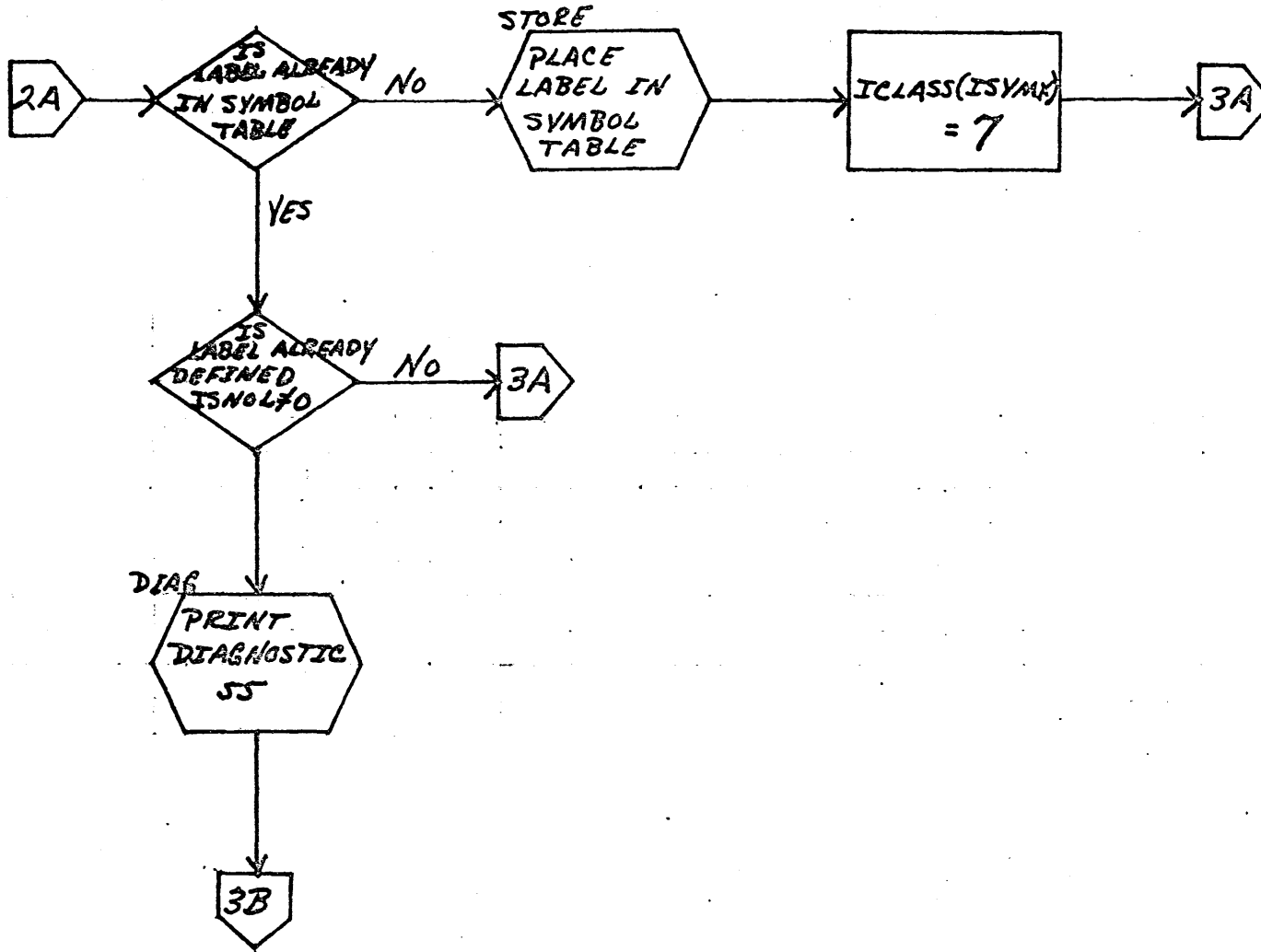
Otherwise, ISNOL of the SYMTAB entry of this label is set to the current statement number, thus setting the label as defined.

The routine then exits with ILLABL equal to the SYMTAB index of the label entry, ISORSX = 7 (the next column to process), IBUF2(5) = the SYMTAB index for this label, and IBUF2(2) set to indicate a labelled statement (negative statement number).





PLABEL

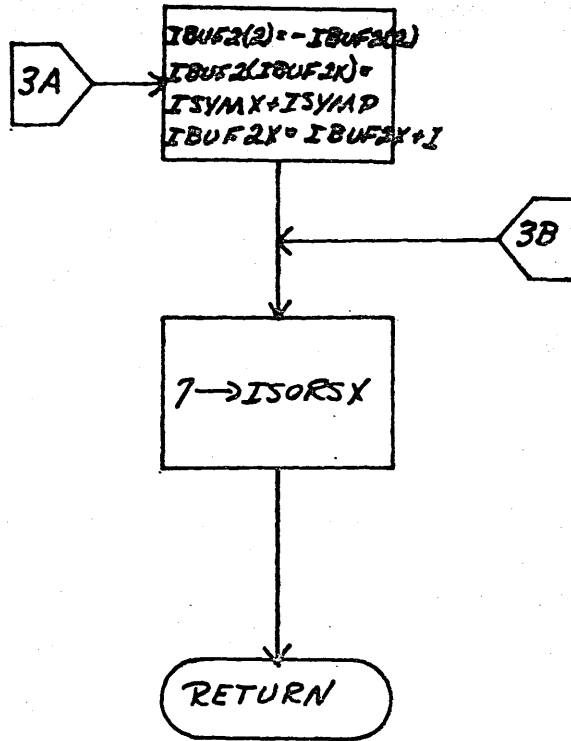


CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	<i>IMS</i>	MACH. TYPE	<i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	<i>PLABEL</i>	PAGE <i>2</i> OF <i>3</i>		PROJECT MGR.			
NUMBER		ISSUE DATE		PROJECT NAME			
DRAWN BY		DATE		TASK NO.			
				TASK NAME			

PLABEL



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700
DOCUMENT TITLE	PLABEL		
PAGE		3 of 3	
NUMBER	ISSUE DATE	TASK NO.	
DRAWN BY	DATE	TASK NAME	

REV	APPROVED	DATE

2-92

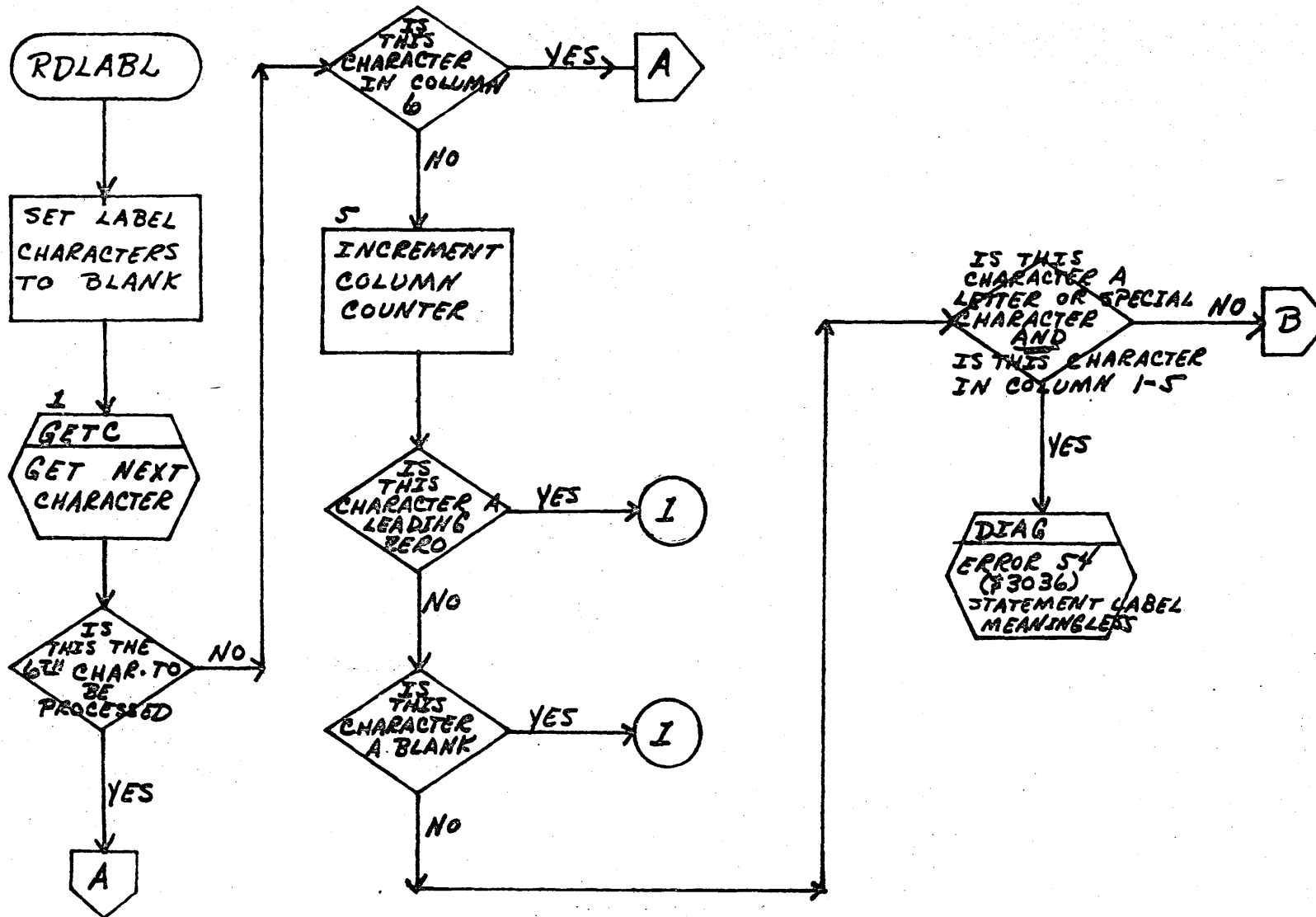
CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 2-93  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005 2.0 MACHINE SERIES 1700

2.2.2.4.1 Subroutine RDLABL

RDLABL is called to read a label. It reads the label and checks to see if it is a number. If it is not, it prints out diagnostic 54. If the label is legal, RDLABL packs the label into JSYM and returns.



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS <b>IMS</b> MACH. TYPE <b>1700</b>	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE <b>RDLABL</b>	PROJECT MGR.			
	PAGE <b>1</b> OF <b>3</b>	PROJECT NAME			
	NUMBER _____ ISSUE DATE _____	TASK NO.			
	DRAWN BY _____ DATE _____	TASK NAME			

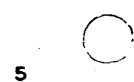


2

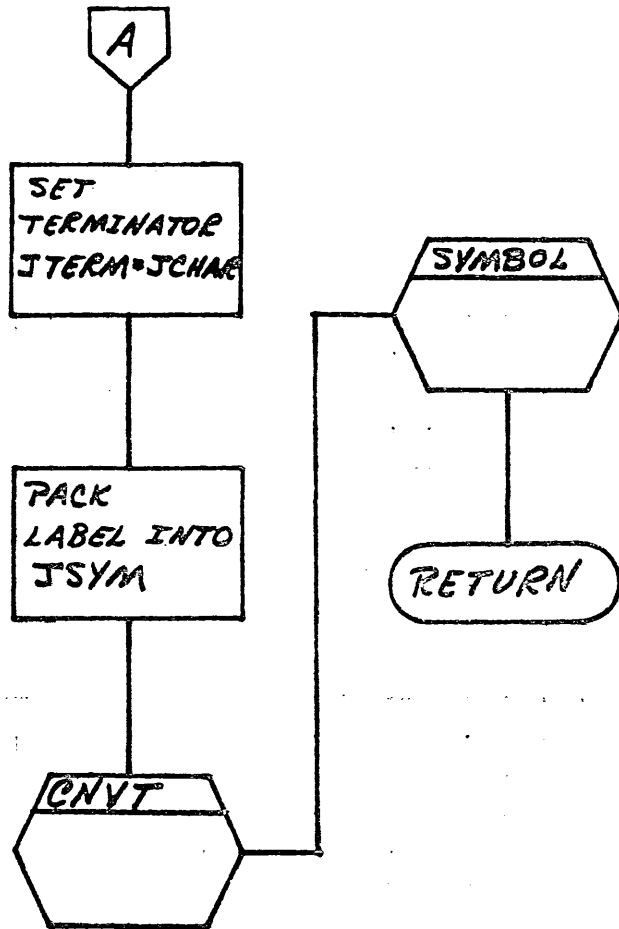


3

4



5



**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	<b>IMS</b>	MACH. TYPE	<b>1700</b>	PROJECT NO.	REV.	APPROVED	DATE
DOCUMENT TITLE	<b>RDLABL</b>	PAGE <b>2</b> OF <b>3</b>		PROJECT MGR.			
NUMBER	ISSUE DATE			PROJECT NAME			
DRAWN BY	DATE			TASK NO.			
				TASK NAME			

A

B

C

D

B

2  
 STUFF  
 CHARACTER  
 IN LABEL  
 ARRAY

SET FLAGS TO  
 INDICATE A  
 LABEL & NO MORE  
 LEADING ZEROS  
 OR BLANKS  
 ILLABL = 1  
 ILEAD = 1

INCREMENT  
 CHARACTER  
 COUNTER

1

CONTROL DATA CORPORATION  
 SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	RDLABL	PAGE 3 OF 3		PROJECT MGR.			
NUMBER	ISSUE DATE			PROJECT NAME			
DRAWN BY	DATE			TASK NO.			
				TASK NAME			

2-96

DOCUMENT CLASS IMS PAGE NO. 2-97  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005x3.0 A/B-E006x4.0 MACHINE SERIES 1700

### 2.2.2.5 Option Selection - Subroutine OPTIONS

OPTIONS is the routine, written in 1700 assembly language, which processes option selection in 1700 Mass Storage Fortran. It is entered from GOA. Upon entrance, OPTIONS clears the option selection holders in the DATA block {IR, IK, IP, IA, IL, IM, IXLG0}, backgrounds its input buffer and solicits input from the standard input device. If there was no  $\nabla$  OPT  $\nabla$  card, the standard options LPX are assumed and the option selection holders in the DATA block IP, IL, and IXLG0 are set #0. If no options are listed on the  $\nabla$  OPT  $\nabla$  card, OPTIONS solicits input from the input comment medium by outputting OPTIONS and waiting for a carriage return.

Upon receipt of a carriage return or finding options on the  $\nabla$  OPT  $\nabla$  card, OPTIONS interrogates each character input. If it is recognizable {R, K, P, A, L, M, X} the corresponding holder is marked selected {#0}. Unrecognizable characters are ignored.

OPTIONS then returns to GOA.

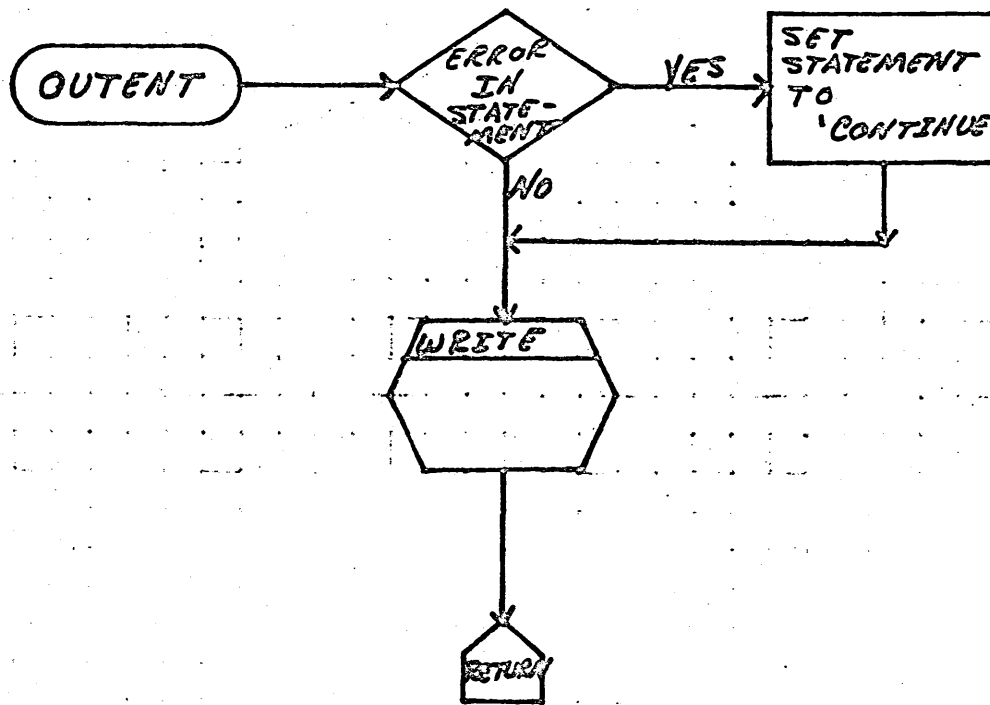
DOCUMENT CLASS TMS PAGE NO. 2-98  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

2.2.2.6 Subroutine OUTENT

OUTENT is the subroutine used to output a record to the scratch output device. If an error has occurred in the statement, it is changed to a CONTINUE and the error-in-statement switch {IERR} is cleared.

OUTENT calls subroutine IOPR {entry point WRITE} to perform the output.





**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	<i>IMS</i>	MACH. TYPE	<i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	<i>OUTENT</i>			PROJECT MGR.			
PAGE <i>1</i> OF <i>1</i>				PROJECT NAME			
NUMBER	ISSUE DATE		TASK NO.				
DRAWN BY	DATE		TASK NAME				

DOCUMENT CLASS IMS PAGE NO. 2-100  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

### 2.2.3.2 Subroutine ENDD0 - The D0 Loop Termination Routine

ENDD0 is called by PHASEA upon completion of processing of a labeled statement and by IOSPR to terminate D0 loops and complete the L0OPT table.

If there are no D0 loops open, ENDD0 exits. If the label of the current statement terminates the innermost open D0, an ENDD0 entry is output and the L0OPT for this D0 loop is completed. This D0 entry is then erased from the L0OPT table {D0 nesting} and the check of label vs. innermost D0 is repeated.

When ENDD0 encounters an open innermost D0 which does not terminate at this statement label, the routine searches the rest of the L0OPT table to ascertain that no outer D0 loops terminate here. If any do, an error is output and the erroneous entry is erased from L0OPT.

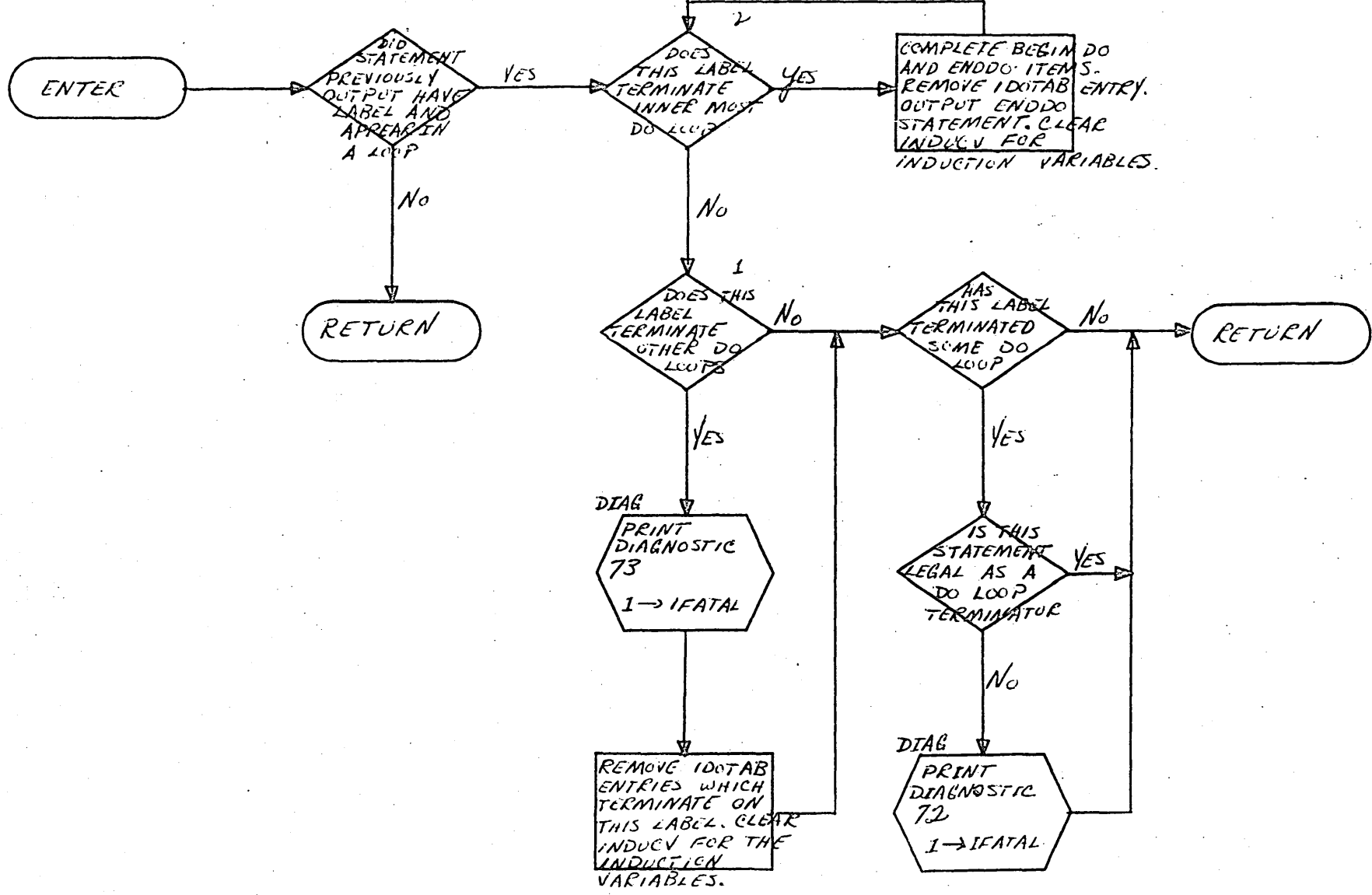
If any D0 loops have terminated at the current statement label, ENDD0 checks the statement associated with this label. If the statement is of a type illegal for D0 termination an error is output and the statement changed to a CONTINUE.

### 2.2.3.3 Subroutine SAVEID

SAVEID was written to allow the use of up to 47 characters of comments {copyright statement} on the source routine identification card which will be output in a format similar to Macro Assembler produced binary NAM blocks.

The format of the binary NAM card includes up to 47 characters of comments preceded by a slash from the source routine identification card {PROGRAM, SUBROUTINE, FUNCTION, or BLOCK DATA} on the binary NAM card. If no slash and no comments appear on the source card, the binary NAM card will be punched entirely with blanks. If there is no slash, but comments do appear on the card, diagnostic #66 will be issued, and the binary NAM card will contain blanks. A continuation card may also be used.

ENDDO



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	ENDDO	PAGE 1 OF 1		PROJECT MGR.			
NUMBER	ISSUE DATE			PROJECT NAME			
DRAWN BY	DATE	3/1/71		TASK NO.			
				TASK NAME			

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS IMS PAGE NO. 2-102  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. COO5\*3.1 A/B MACHINE SERIES 1700

2.3 General Pass 1 Routines

2.2.1 Symbol Table Manipulating Routines

2.3.1.1 SYMBOL

To locate a name in the symbol table, store the name in the three cell array, JSYM. If the name is not a constant, pack it into JSYM{1} and JSYM{2} by calling CNVT. The name may be a statement label. If the name is an integer constant, the constant's value should be in JSYM{1} and the constant 2424<sub>16</sub> should be in JSYM{3}.

If the name is a real constant, the constant's value should be in JSYM{1} and JSYM{2} and the constant 2525<sub>16</sub> should be in JSYM{3}. If the name is a double precision constant, the constant's value should be in JSYM{1}, JSYM{2}, and JSYM{3}. The common variable JMODE should equal 6 for double precision constant. Issue the statement

CALL SYMBOL

Upon exit, if the common variable ISYMD is zero, the symbol is not in the table. If ISYMD is non-zero, the COMMON variable ISYMX indexes the symbol entry, that is, each item in that symbol's entry may be obtained as

item name {ISYMX}

If ISYMD is zero, ISYMX points to the next available symbol table entry. There are two versions of SYMBOL, as follows:

Phase A: If SYMTAB paging is necessary, only the current page is written on disk before being overlaid; later pages are simply overlaid. SYMBOL accomplishes this by setting ISYMX negative before calling GETSYM after the current page has been written.

Phase B: If it is known that the name is not in the symbol table, SYMBOL may be called with ISYMX negative. In this case SYMBOL does not search the symbol table but returns with ISYMX pointing to the next available entry.

2.3.1.2 Subroutine STORE

Assume that JSYM{1}, JSYM{2}, and JSYM{3} are filled, probably by GETF, but possible by the programmer himself. Also assume that ISYMX is pointing to the next available

DOCUMENT CLASS IMS PAGE NO. 2-103  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. COO5\*3.1 A/B MACHINE SERIES 1700

entry in the symbol table. SYMBOL will set ISYMX in this manner if an entry is not in the table. Then the symbol in JSYM will be inserted into the symbol table. ISYMN {current size of symbol table} will be incremented. If JSYM{3} is 2424<sub>16</sub> or 2525<sub>16</sub>, or if JMODE is 6 for double precision constants, ICLASS and ITYPE will be set. Issue the statement

CALL STORE

Thus, the normal sequence of events leading to an insertion in the symbol table is

CALL GETF

which then does

CALL SYMBOL

to see if the field is in the symbol table. Upon return from GETF, the calling program does

CALL STORE

to insert the field in the symbol table if it is not in it already.

### 2.3.1.3 Subroutine GETSYM

GETSYM insures that the symbol table page requested by ISYMX is in core, updates the current symbol table page holder {ISYMPC} if necessary, and adjusts ISYMX to the in-core subscript of the desired symbol table entry {ISYMX = ISYMX - ISYMP}.

There are five versions of GETSYM, distinguished as follows:

Phases A and B:

If ISYMX is positive on entry and paging is necessary, the current symbol table page is written on disk before the desired page is read. If ISYMX is negative, it is set positive and the current symbol table page is not saved.

The symbol table page size is 960 words. The current ISTABX page {96 words} is read and written along with the SYMTAB page, therefore eleven sectors are read and written for each page.

Phase C and Phase D/E 2.0B version.

The symbol table page size is 2400 words. When reading and writing the large page every eleventh sector {formerly an ISTABX page} is skipped. The current symbol

DOCUMENT CLASS \_\_\_\_\_ IMS \_\_\_\_\_ PAGE NO. \_\_\_\_\_ 2-104  
PRODUCT NAME \_\_\_\_\_ 1700 MASS STORAGE FORTRAN \_\_\_\_\_  
PRODUCT MODEL NO. \_\_\_\_\_ C005 V.2.0 \_\_\_\_\_ MACHINE SERIES \_\_\_\_\_ 1700 \_\_\_\_\_

table page is always saved on disk before a new page is read.

Phase D local 1, 2.0A version:

This version reads and writes the compressed symbol table with a page size of 1000 words (see Assembler Output Phase, section 5.2). The current symbol table page is always saved.

Phase D local 2, 2.0A version:

This version reads the normal symbol table with a page size of 480 words. Every eleventh disk sector is skipped. The current symbol table page is never saved.

Phase E, 2.0A version:

The symbol table page size is 480 words. Every eleventh disk sector is skipped. The current symbol table page is always saved.

#### 2.3.1.4 Subroutine CNVT

CNVT packs a six-character variable name into two words for storage in the symbol table. Only alphanumeric characters and blanks can be packed. CNVT packs the leftmost three characters into the first word of JSYM and the rightmost three characters into the second word as follows:

Let  $x$ ,  $y$ ,  $z$ ,  $a$ ,  $b$ , and  $c$  be the internal Fortran codes for  $X$ ,  $Y$ ,  $Z$ ,  $A$ ,  $B$ , and  $C$ . Then the name  $XYZABC$  is stored as:

$$\begin{aligned} \text{JSYM (1)} &= 1521x + 39y + z \\ \text{JSYM (2)} &= 1521a + 39b + c \end{aligned}$$

Blanks, which are 46 in internal Fortran code, are charged to 38 before packing.

#### 2.3.1.5 Subroutine SYMSCN

SYMSCN gets the entry following the current entry from the symbol table. Upon entrance,  $\text{ISYMX} = 0$  requests the first symbol table entry, and  $\text{ISYMX} = (\text{an in-core symbol table subscript})$  requests the entry  $(\text{ISYMX} + \text{ISYMP} + \text{ISYMF})$ . Upon return,  $\text{ISYMP} = 1$  says there are no more entries. GETSYM is called just before return.

CONTROL DATA CORPORATION

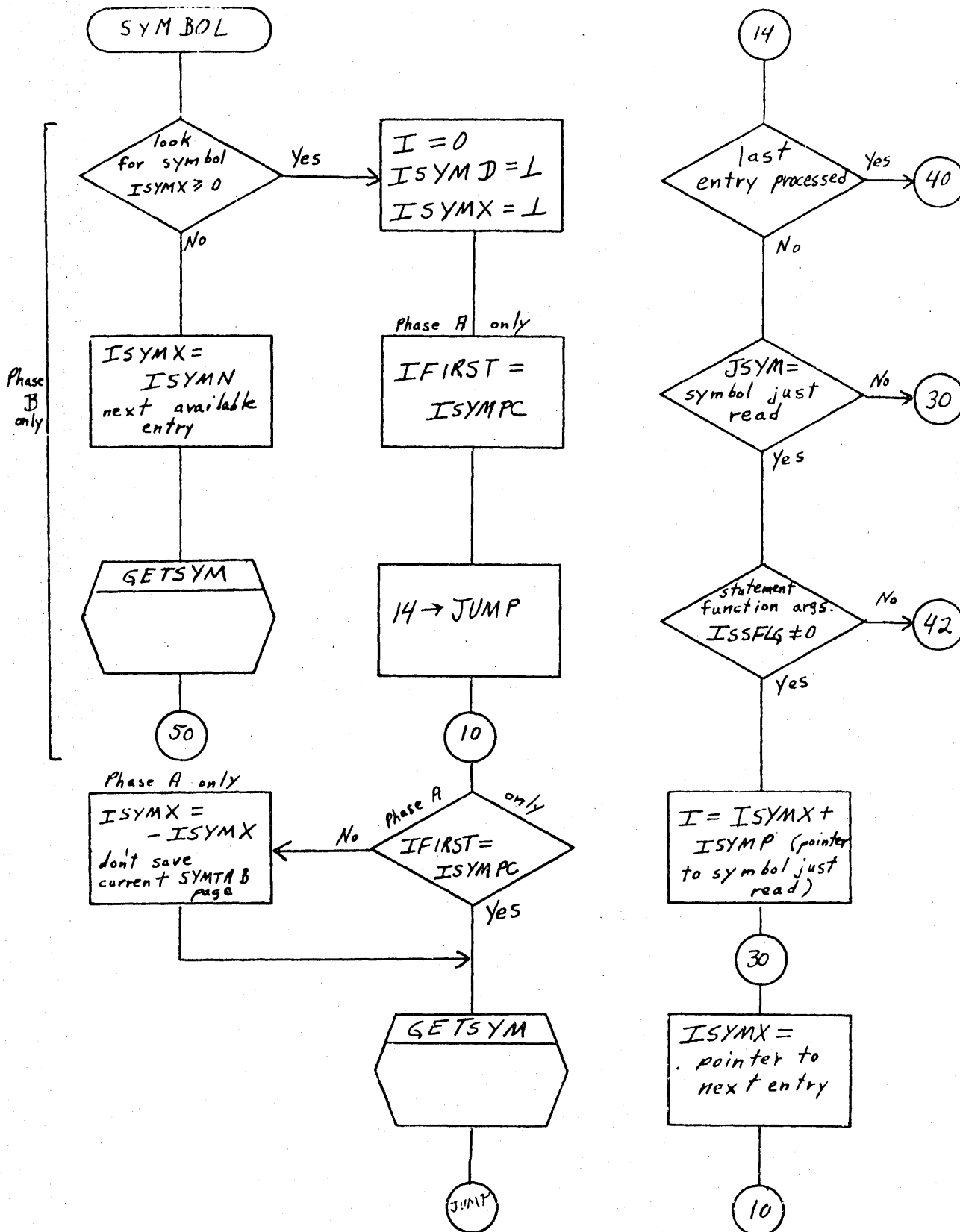
DIVISION

DOCUMENT CLASS IMS PAGE NO. 2-105  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005 V.2.0 MACHINE SERIES 1700

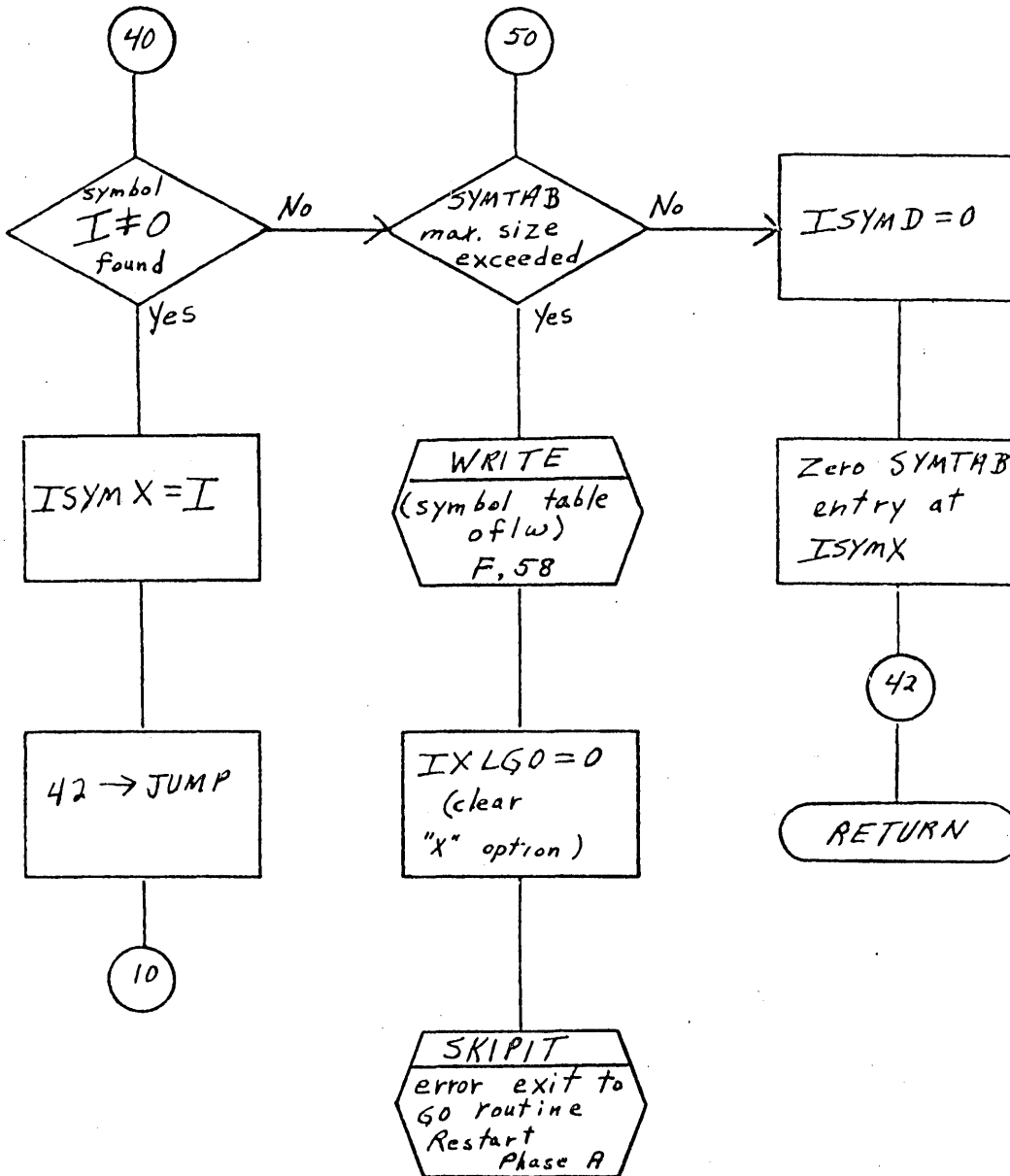
Input parameters: ISYMX (in-core subscript), ISYMP set to current entry.

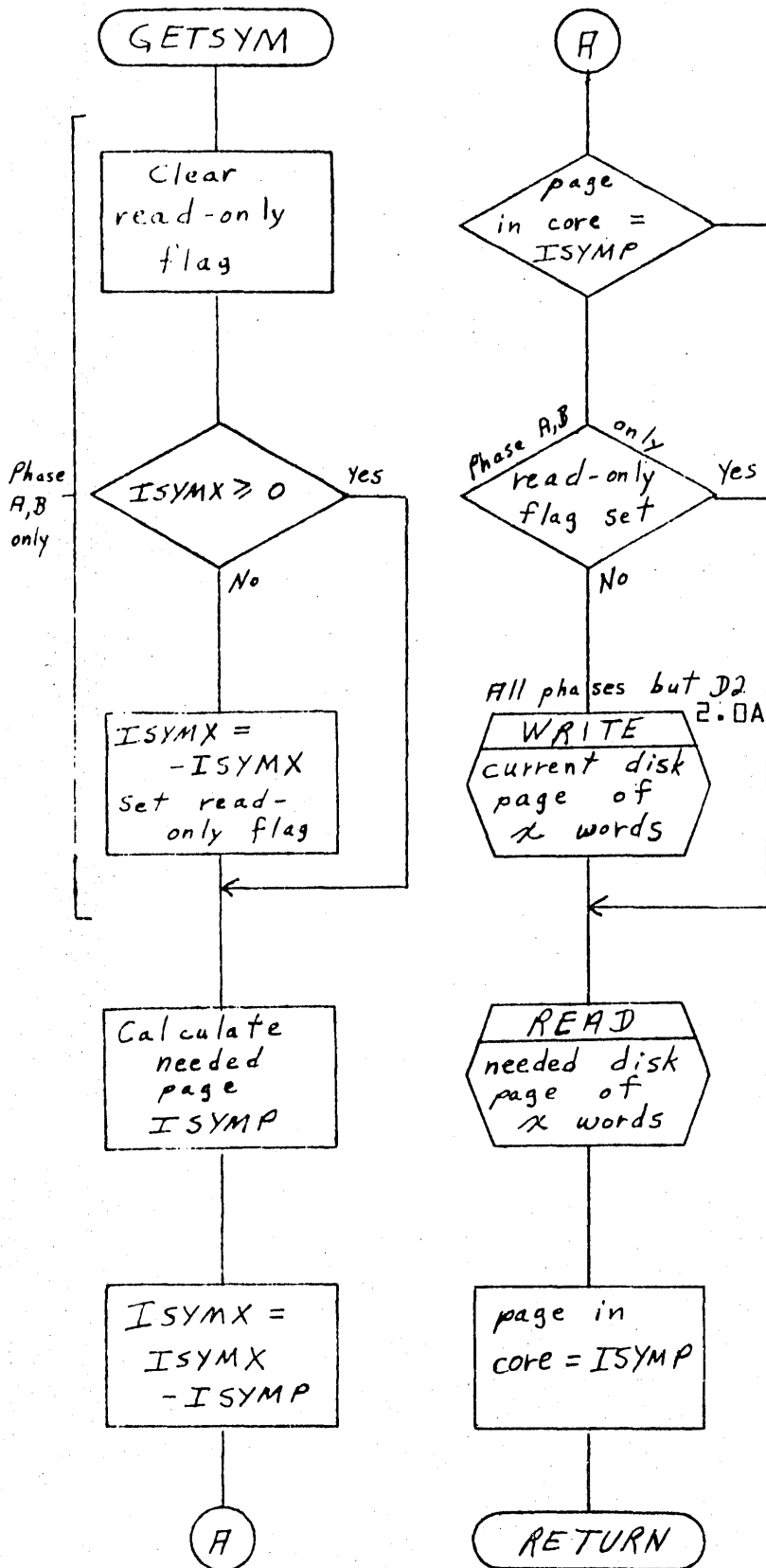
Output parameters: ISYMX (in-core subscript), ISYMP updated to next entry.

There are four versions of SYMSCN. The phase D local 1 2.0A version references the compressed symbol table (see Assembly Output Phase, 5.2). The other three versions differ only in the dimension of the array SYMTAB in common.







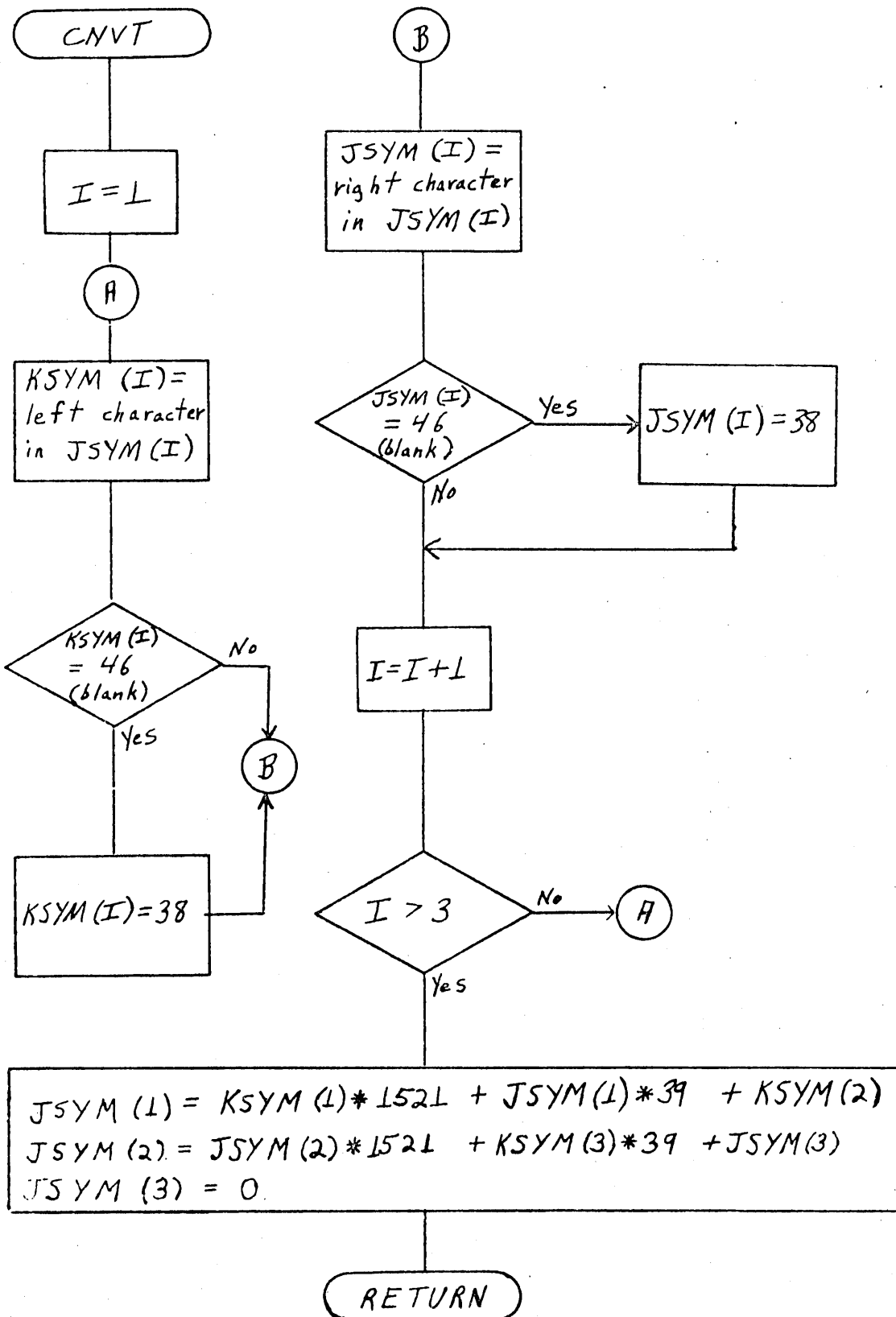


Phases A and B  
 $x = 960 + 96$   
 (SYMTAB and ISTABX)

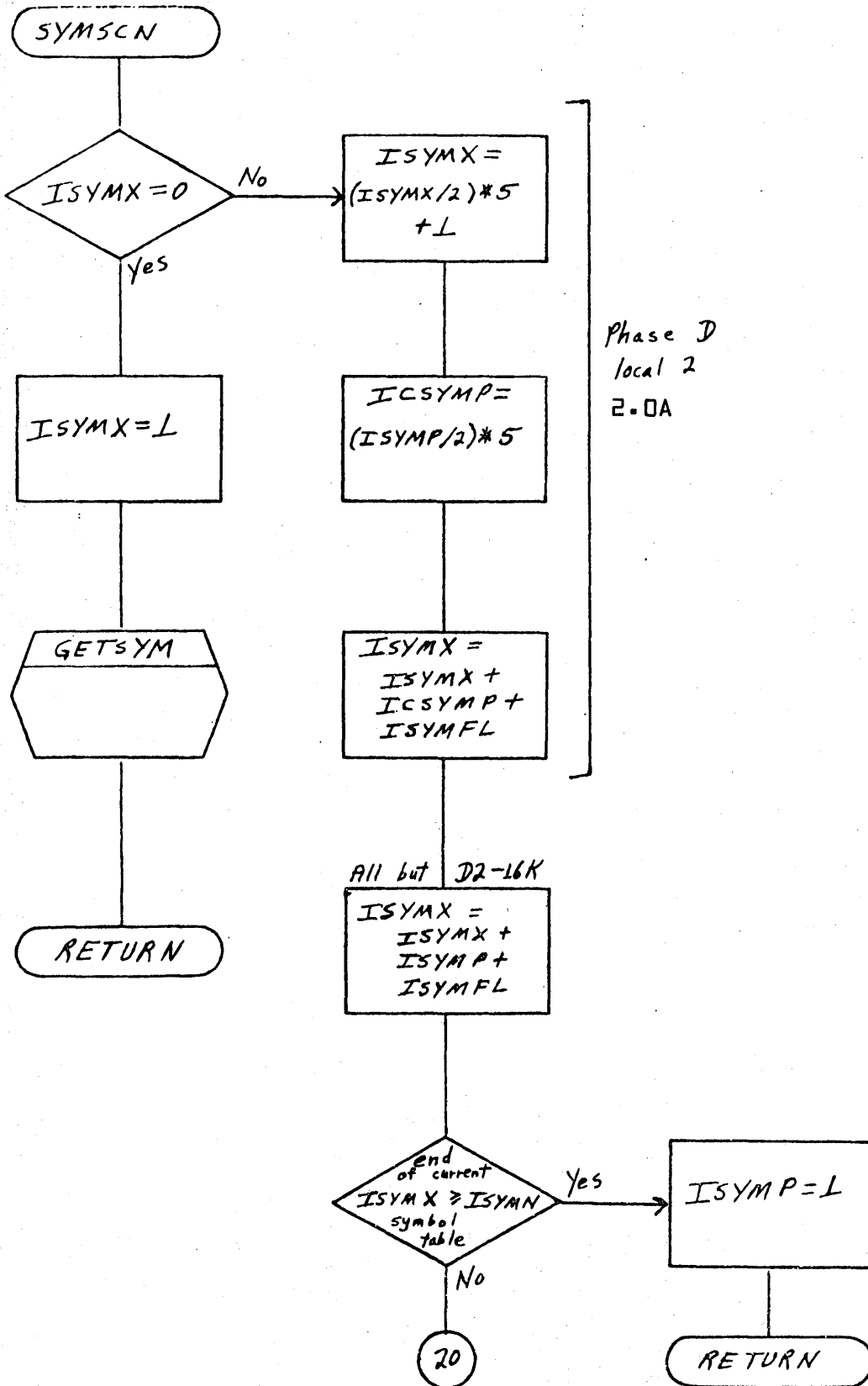
Phases C and D/E 2.0B  
 $x = 2400$   
 (skip every 11th sector)

Phases D2, E 2.0A  
 $x = 480$   
 (skip every 11th sector)

Phase D local I 2.0A  
 $x = 1000$   
 Read and write  
 CSYMTAB  
 (compressed symbol  
 table)

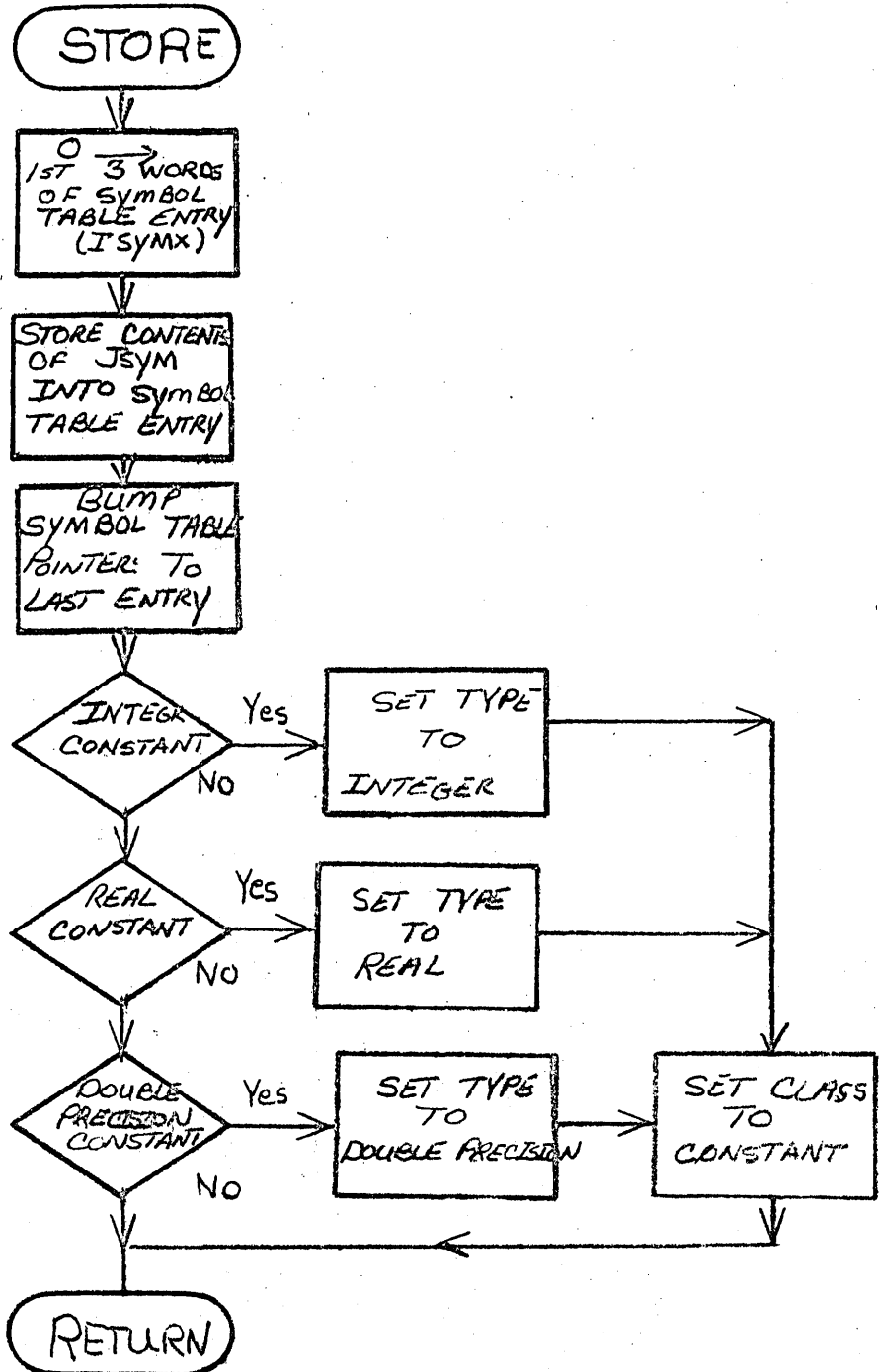


DOCUMENT CLASS: 1700 MASS STORAGE FORTRAN PAGE NO. 2-110  
PRODUCT NAME: CO05 V.2.0 MACHINE SERIES 1700





LA JOLLA FACILITY



TITLE	IMS STORE		DRG. NO.	
			REVISION	
DRAWN BY J. HARRINGTON	PROJ.	DATE 1-16-73	SHEET	1 OF 1

DOCUMENT CLASS IMS PAGE NO. 2-112  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

2.3.2 Field Processing Routines

2.3.2.1 Subroutine GETF

GETF is the subroutine which extracts a field from the source input in ISORS\*. The column at which to start is given in ISORSX\*. If the field is alphabetic {first character is A-Z}, the field is placed left-justified in JSYM and JMODE set to 2. JESWT is set to indicate the implicit type of the name {1 = integer, 2 = real}. If the field is a fixed, floating point, or double precision floating point constant, it is converted and placed in JSYM. JMODE is set to indicate the mode of the constant. Integer constants are placed in JSYM{1} converted as if they were decimal unless OCT ≠ 0 upon entrance to GETF, or if the constant is preceded by \$. If OCT ≠ 0, an integer constant is considered octal and non-integer fields are considered errors. OCT is set to zero upon exit from GETF. If the constant is preceded by \$, the constant is considered hexadecimal and the only permissible digits are 0-9, A-F. For integer constants: JSYM{2} = 0, JSYM{3} = 2424<sub>16</sub>. Real constants are placed in JSYM{1} and JSYM{2} and JSYM{3} is set to 2525<sub>16</sub>. Double precision floating point constants are placed in JSYM{1}, JSYM{2}, and JSYM{3} and common variables JMODE and JFLOT are set to 6 and 3 respectively.

If the field is null, JMODE is set = 0.

SYMBOL is called for alphabetic or numeric fields if NERR = 0 upon entrance to GETF.

The field terminator is placed in JTERM.

JTERM settings for 3 character logical operators are:

- .AND. = AD
- .NOT. = NT

Error messages will be printed as errors are encountered if NERR = 0 upon entrance to GETF.

NERR is set to zero upon exit from GETF.

- JERR = 0 - no error encountered in field switch
- JERR ≠ 0 - error encountered

\*If AND {IFLAGS, #10} is not zero, input is in IBUF2 {IBUF2X}. In 2.0A, there are two versions of GETF.

DOCUMENT CLASS IMS PAGE NO. 2-113  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. COO5\*3.1 A/B MACHINE SERIES 1700

GETF switches

JESWT	constant analysis indicator 0 digits only encountered 1 decimal point encountered 2 E or D encountered 3 exponent is + 4 exponent is -
JFLOT	mode of constant 0 - integer 2 - real 3 - double precision
JOCHAR	character count
JOL	overflow indicator for constant conversion
JCHAR	character being looked at
JE	fraction count
JEL	exponent count
JCT	JSYM character indicator
JCTL	JSYM word count

If an error is encountered and NERR = 0, an error message is output and depending on the severity of error, JERR may be set  $\neq$  0. If JERR  $\neq$  0 upon exit from GETF, the contents of JSYM and JMODE are meaningless. NERR is set = 0 upon exit from GETF.

INPUT Parameters

OCT	convert integer constant as octal if OCT $\neq$ 0. If field is not an octal integer, an error is assumed.
ISORSX	column of input at which to start
ISORS	source input table
NERR	$\neq$ 0 says don't print errors, don't call SYMBOL.

OUTPUT Parameters

JSYM	3 word holder for field
JTERM	1 word holder for field terminator
JMODE	mode of JSYM 0 = null 2 = alphabetic 3 = integer - only JSYM{1} is significant 5 = real - JSYM{1} and {JSYM{2} contain value 6 = double precision - JSYM{1}, JSYM{2}, JSYM{3} contain value

DOCUMENT CLASS IMS PAGE NO. 2-114  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

ISORSX updated to reflect start of next field  
 or end of input  
 JERR error switch  
 JESWT for JMODE = 2: implicit type of JSYM  
 1 = integer  
 2 = real

1700 INTERNAL CODE

CHARACTER		CODE {Decimal}
digits	0-9 . . . . .	0-9
letters	A-Z . . . . .	10-35
dollar sign	\$ . . . . .	36
period	. . . . .	37
plus sign	+ . . . . .	38
minus sign	- . . . . .	39
equal sign	= . . . . .	40
left parenthesis	{ . . . . .	41
right parenthesis	} . . . . .	42
comma	, . . . . .	43
slash	/ . . . . .	44
asterisk	* . . . . .	45
blank	△ . . . . .	46
End of statement	△ . . . . .	47
single quote	' . . . . .	48

2.3.2.2 Subroutine GETC

GETC gets one character from ISORS. The column to get is given in ISORSX. If JBLANK is zero upon entrance to GETC, blank characters are skipped. If AND {IFLAGS, #10} is not zero, the character is from IBUF2 {IBUF2X}. In 2.0A there are two versions. JBLANK is not initialized upon exit.

INPUT Parameters

ISORSX card column at which to start  
 ISORS source input table  
 JBLANK present blanks switch. 0 = skip blanks;  
 ≠ 0 - present blanks



DOCUMENT CLASS IMS PAGE NO. 2-114A  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. COO5\*3.1 A/B MACHINE SERIES 1700

OUTPUT Parameters

JCHAR next significant character {right  
justified}  
ISORSX updated

2.3.2.3 GPUT

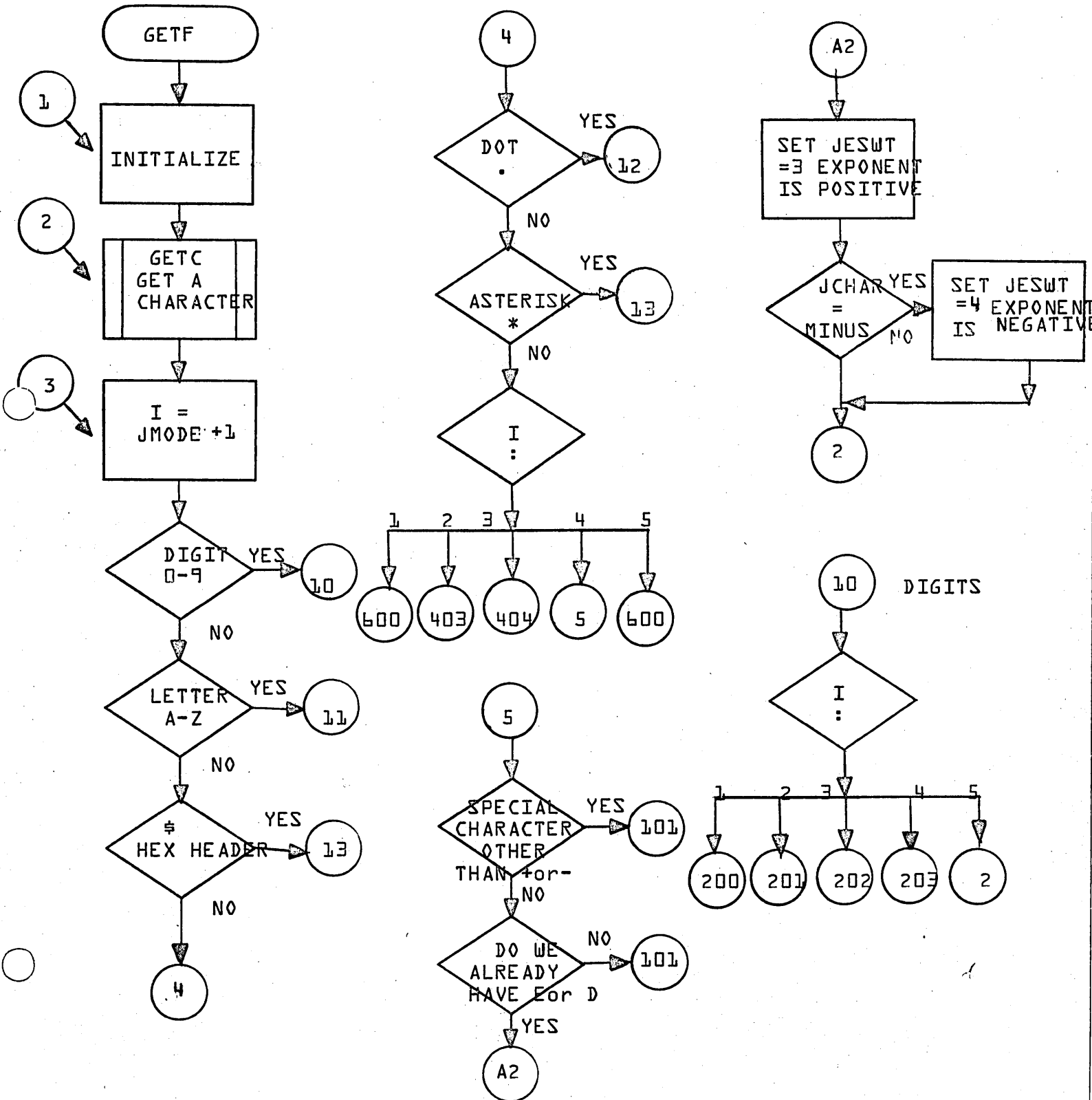
GPUT packs the character in JCHAR into the three word buffer JSYM into the 8 bit byte signaled by the current setting of JCT1 {character number}.

The characters are packed two per word.



DOCUMENT CLASS IMS PAGE NO. 2-115  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

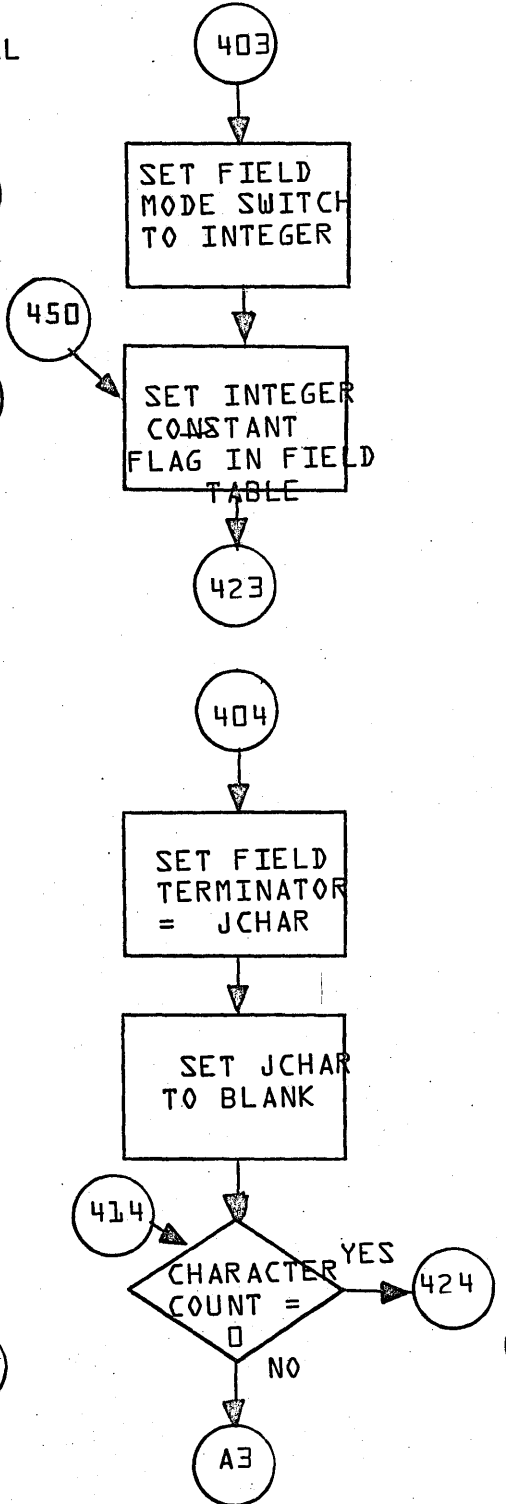
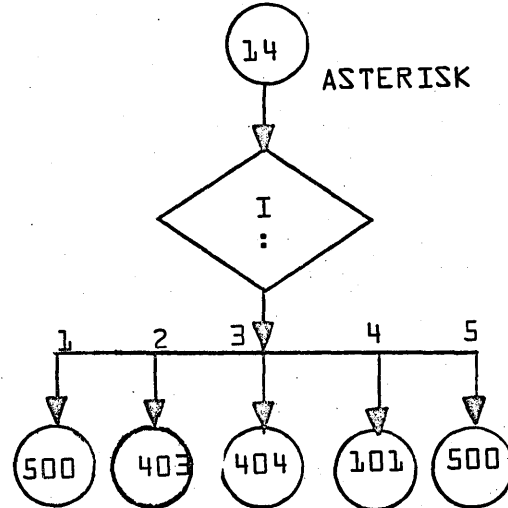
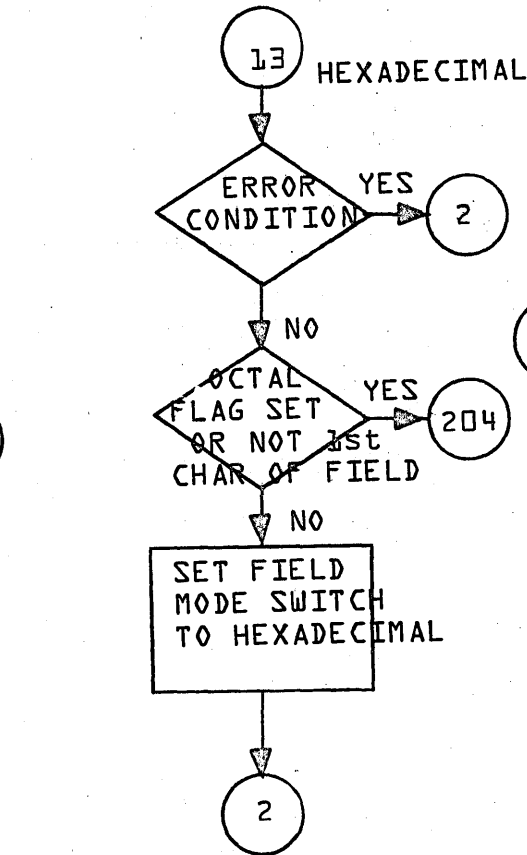
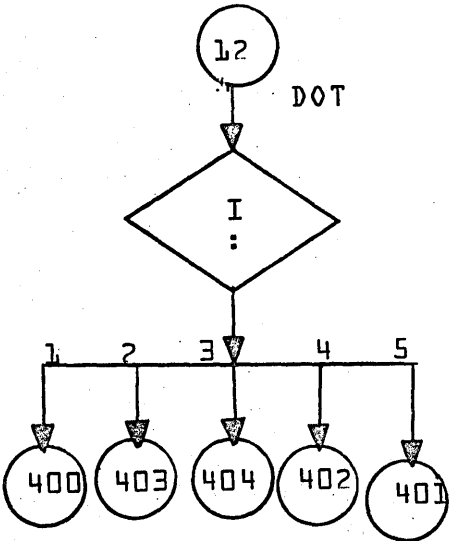
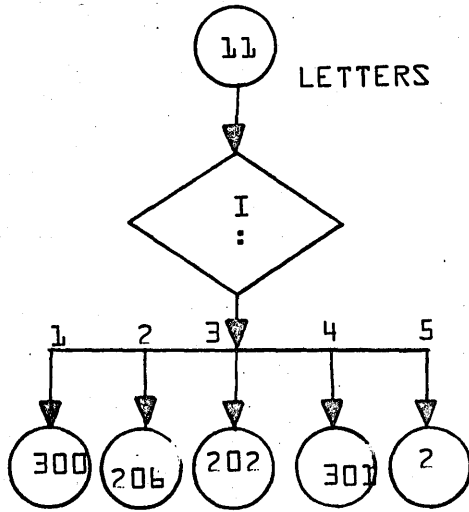
GETF  
 Page 1 of 14



**CONTROL DATA CORPORATION**  
**LA JOLLA RESOURCE CENTER** \_\_\_\_\_ **DIVISION**

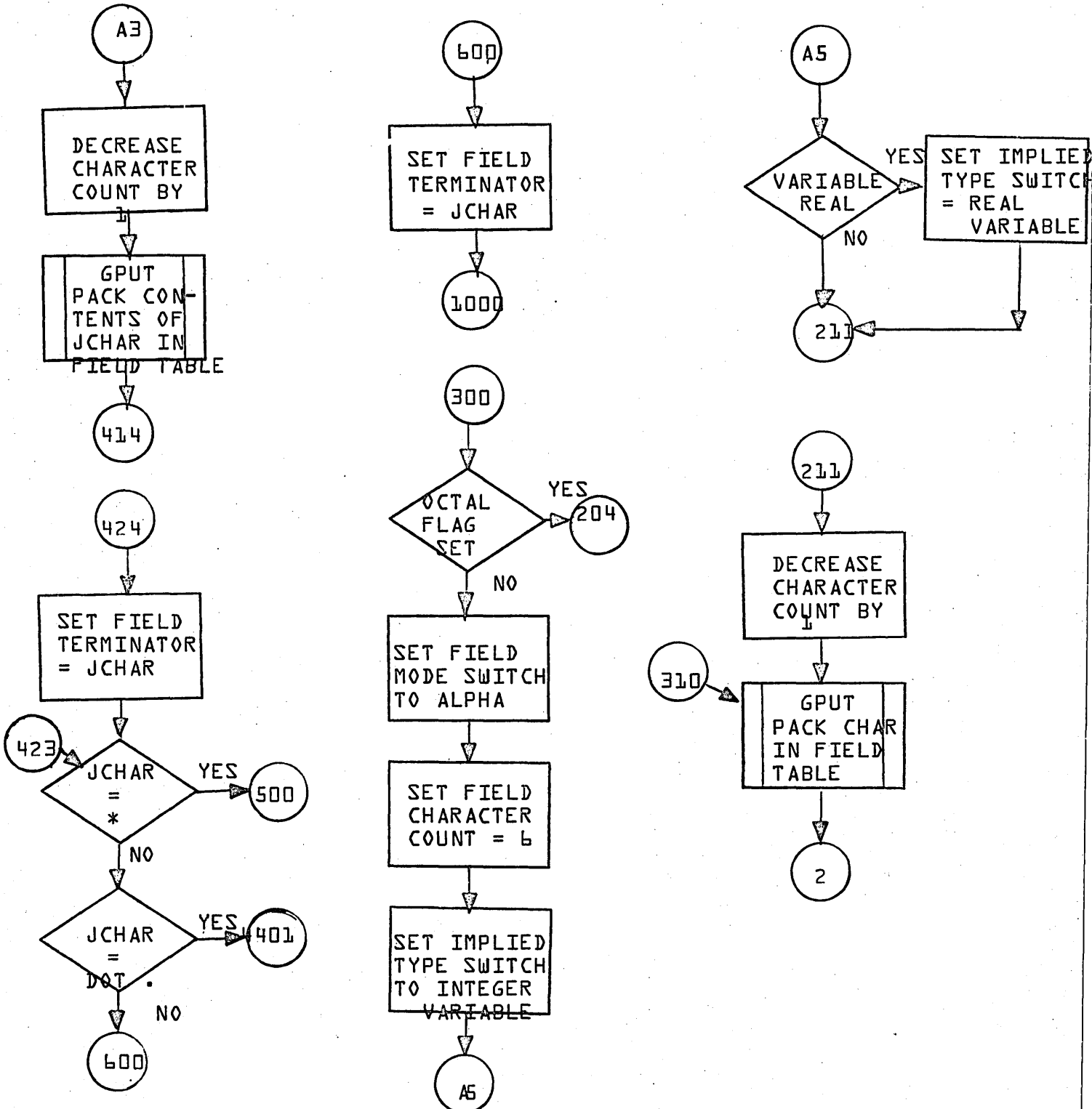
DOCUMENT CLASS IMS PAGE NO. 2-116  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

GETF  
Page 2 of 14



DOCUMENT CLASS IMS PAGE NO. 2-117  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

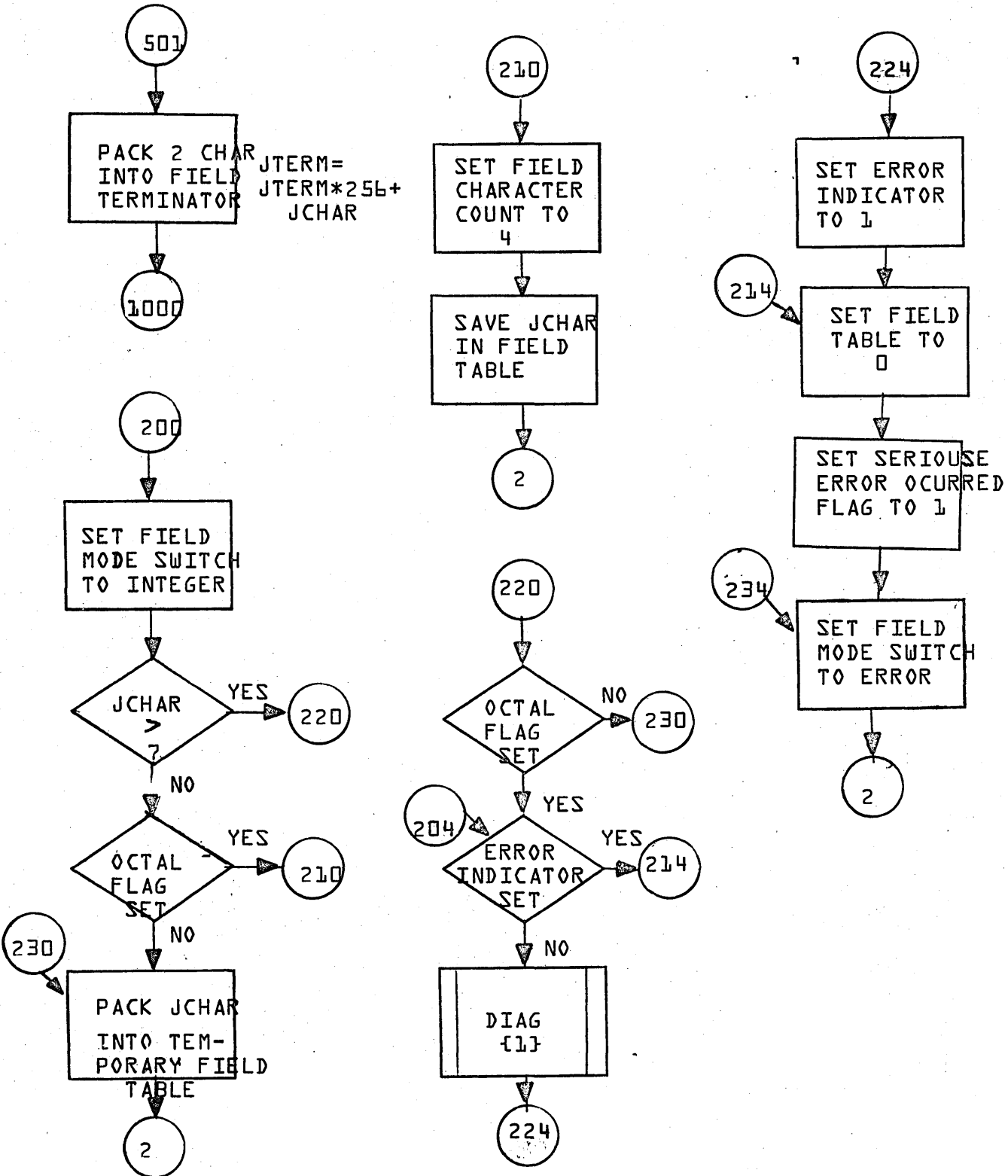
GETF  
 Page 3 of 14



CONTROL DATA CORPORATION  
 LA JOLLA RESOURCE CENTER DIVISION

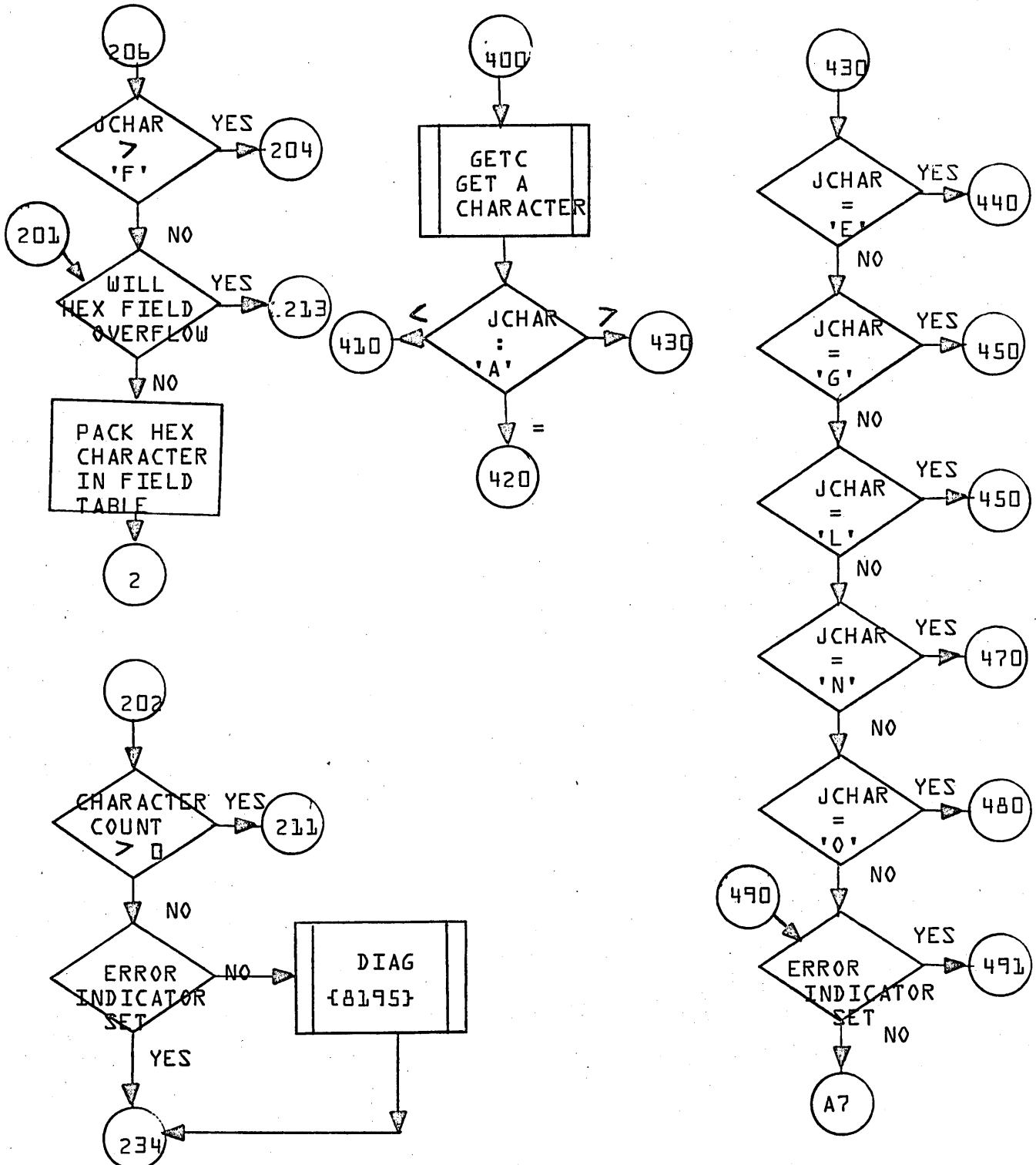
DOCUMENT CLASS IMS PAGE NO. 2-118  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

GETF  
 Page 4 of 14



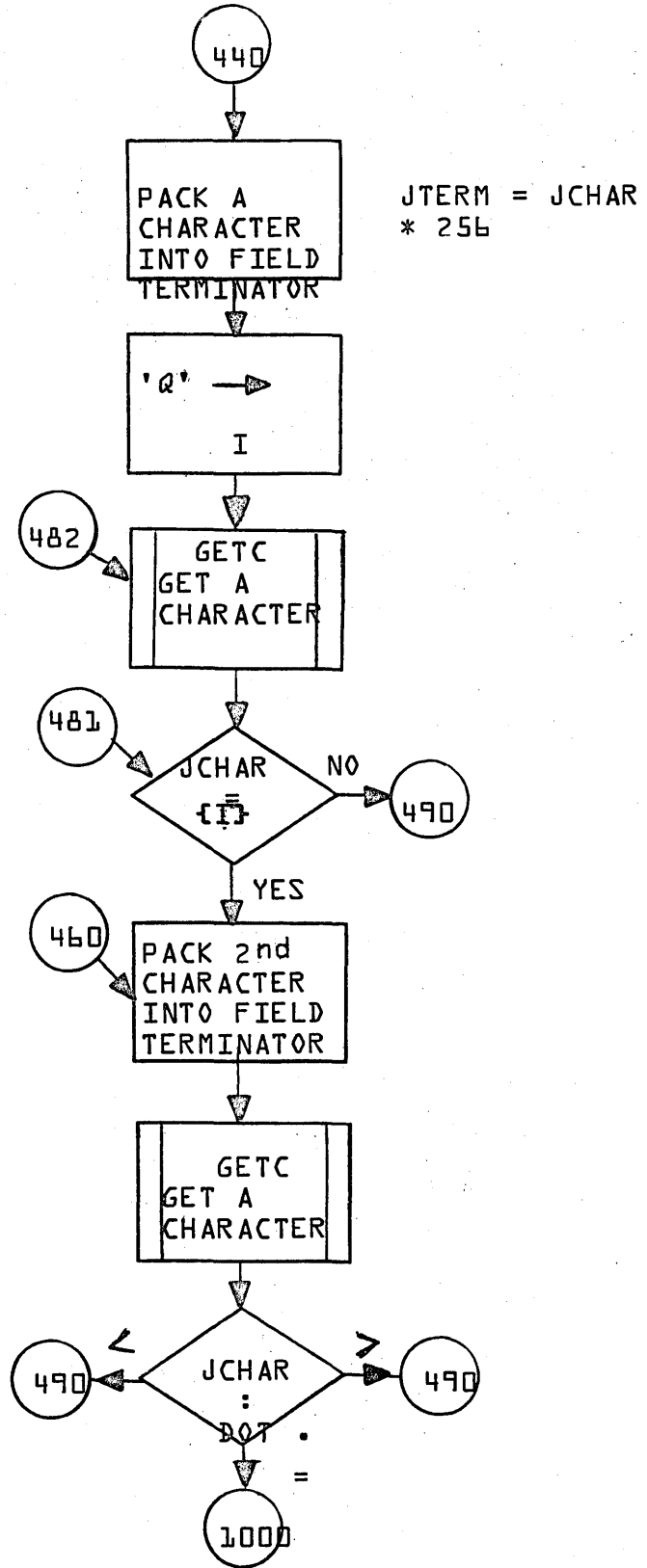
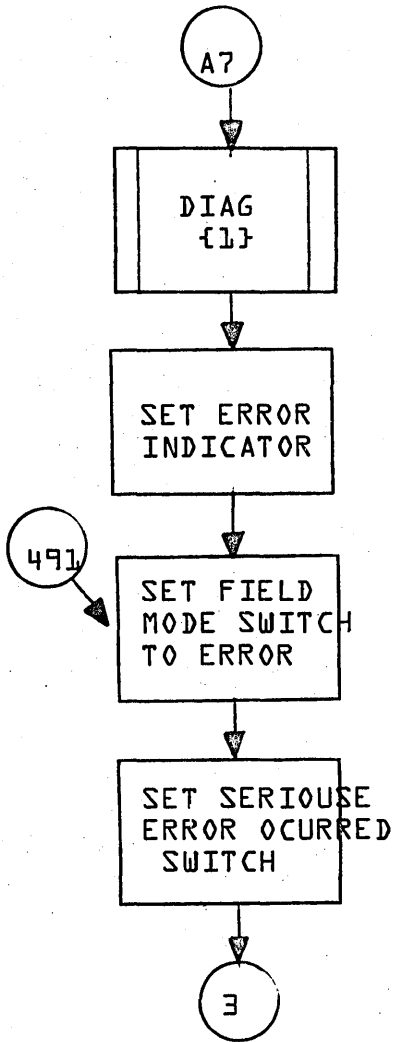
DOCUMENT CLASS IMS PAGE NO. 2-119  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

GETF  
 Page 5 of 14



DOCUMENT CLASS IMS PAGE NO. 2-120  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

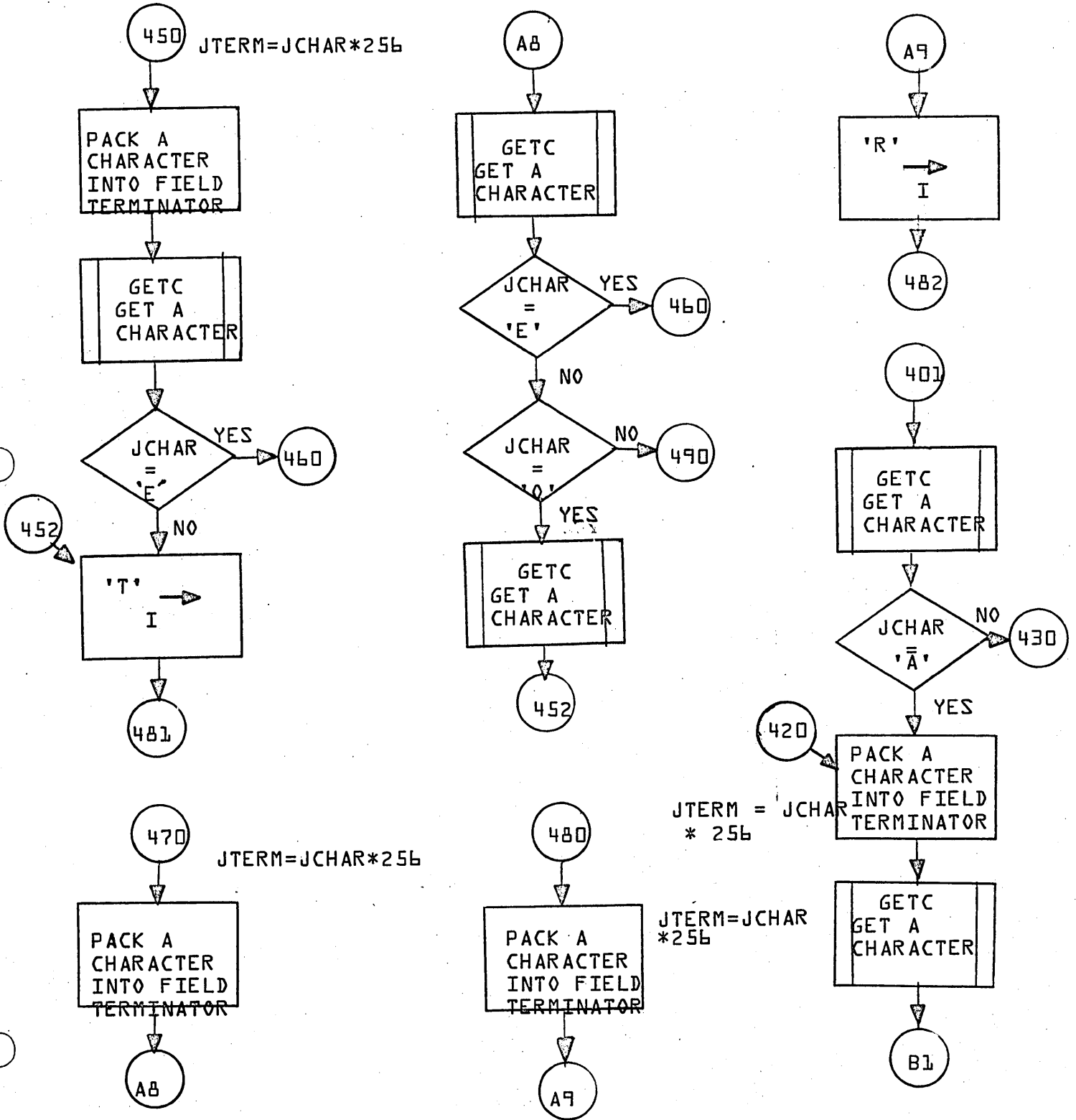
GETF  
 Page 6 of 14





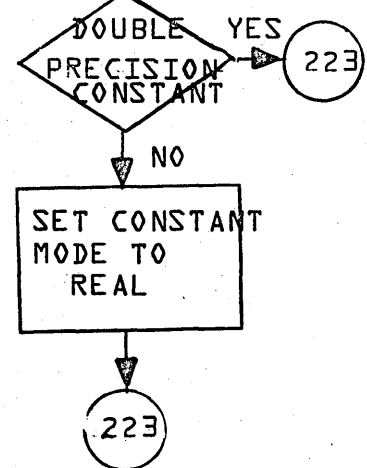
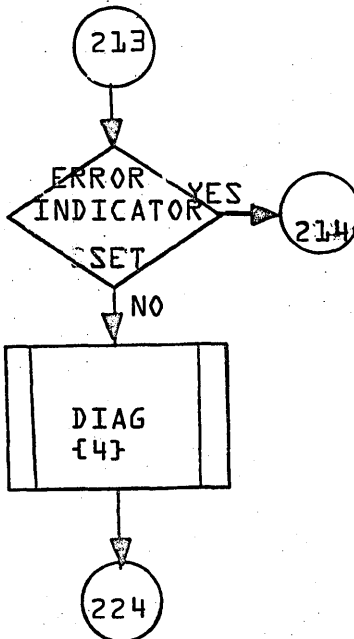
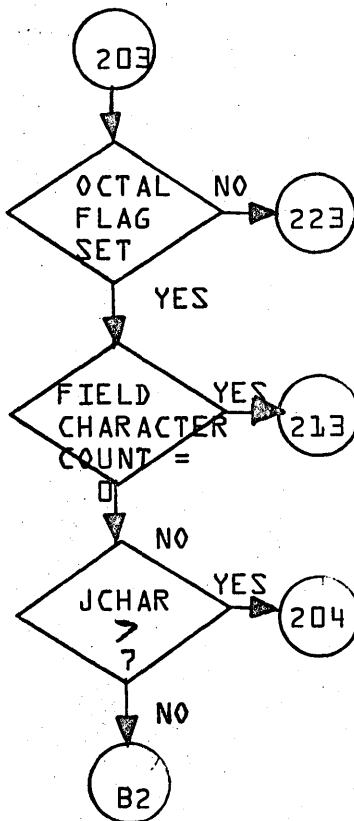
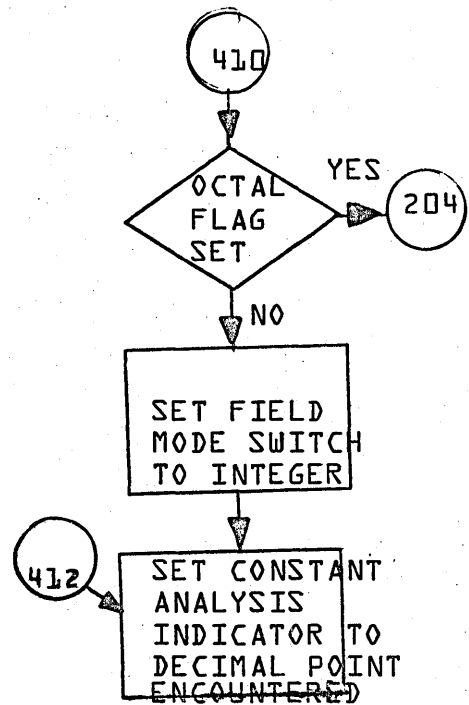
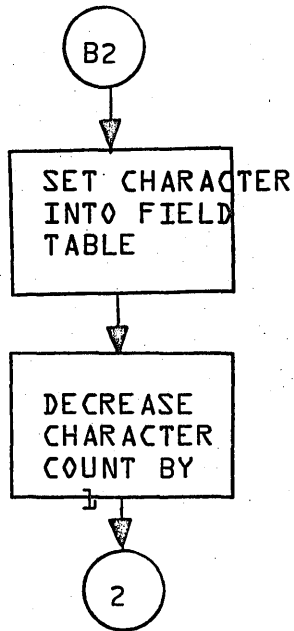
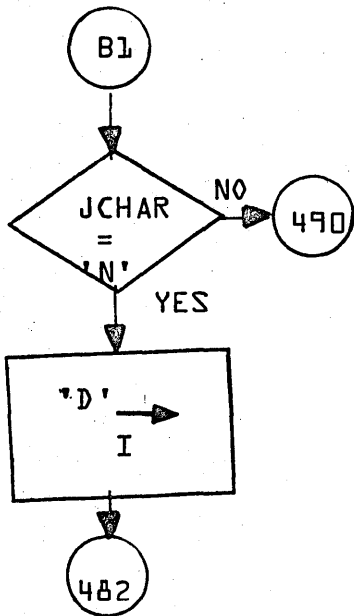
DOCUMENT CLASS IMS PAGE NO. 2-121  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. CO05\*3.1 A/B MACHINE SERIES 1700

GETF  
Page 7 of 14



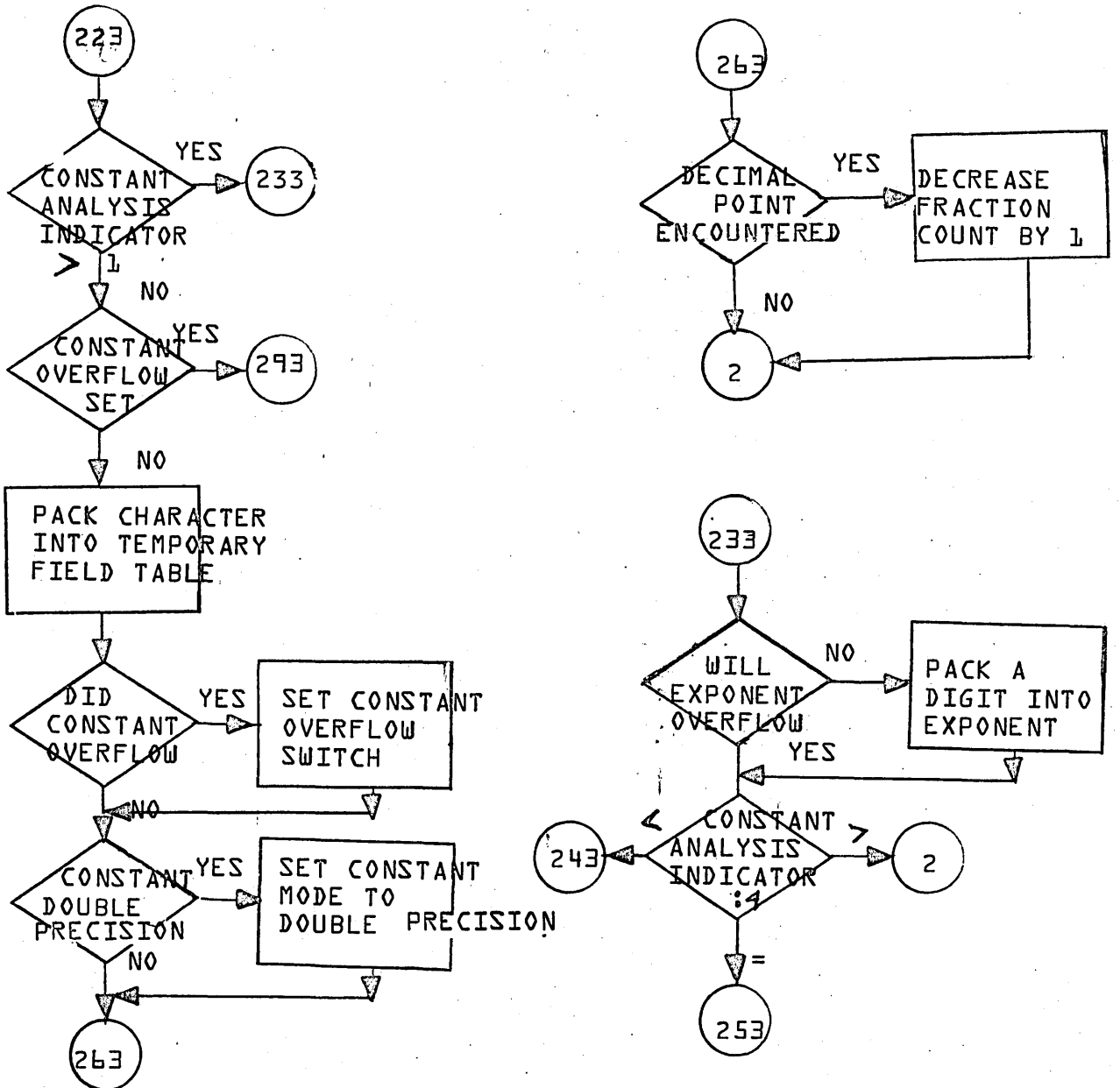
DOCUMENT CLASS IMS PAGE NO. 2-122  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

GETF  
 Page 8 of 14



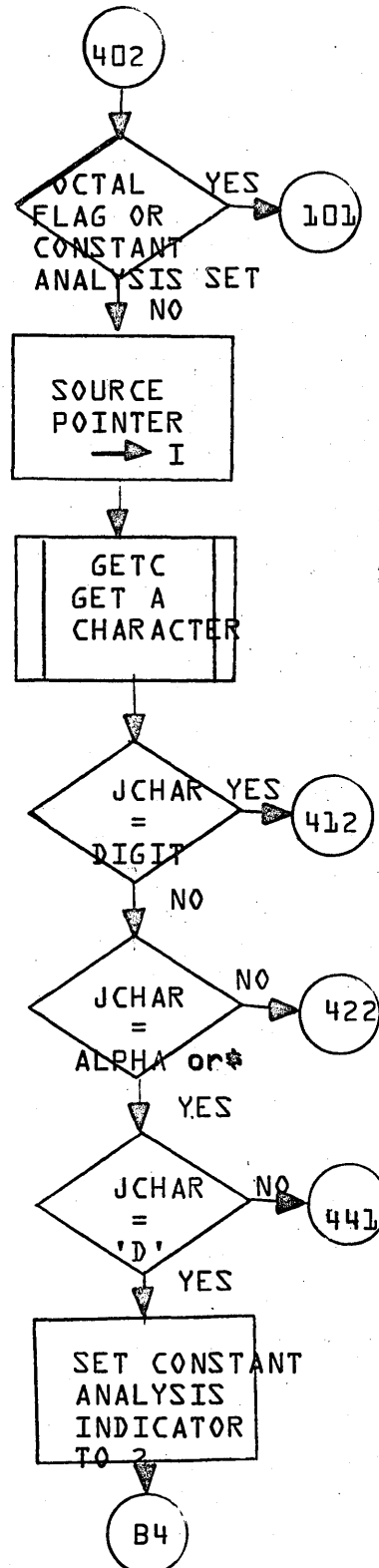
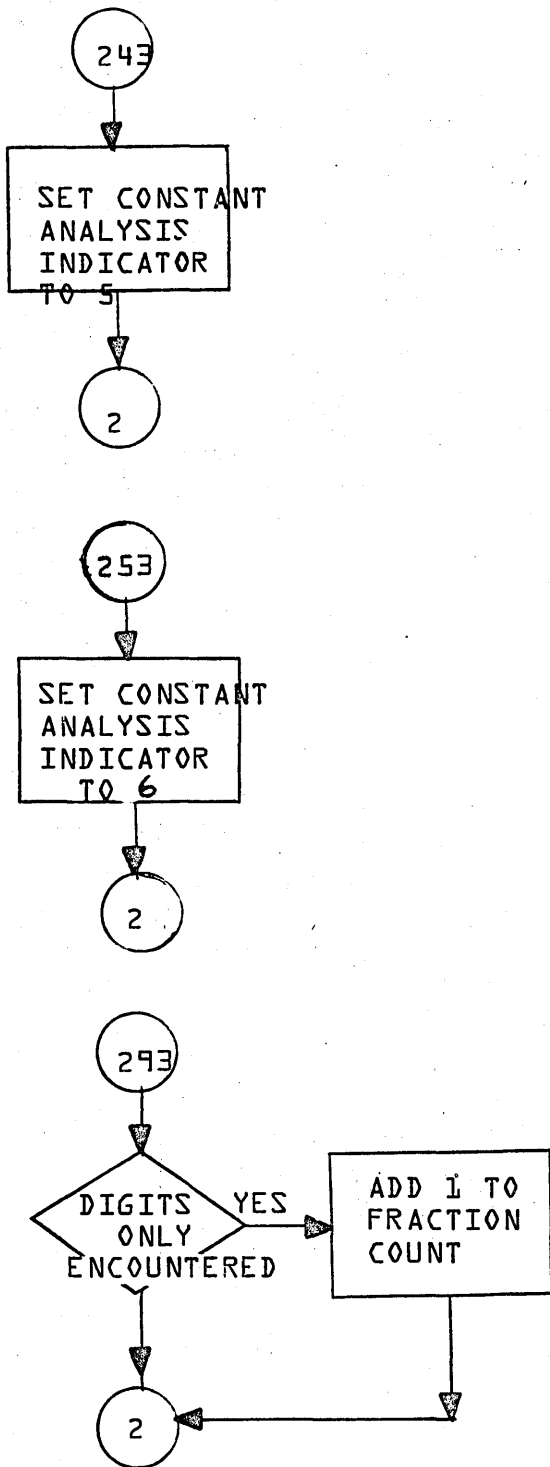
DOCUMENT CLASS IMS PAGE NO. 2-123  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

GETF  
 Page 9 of 14



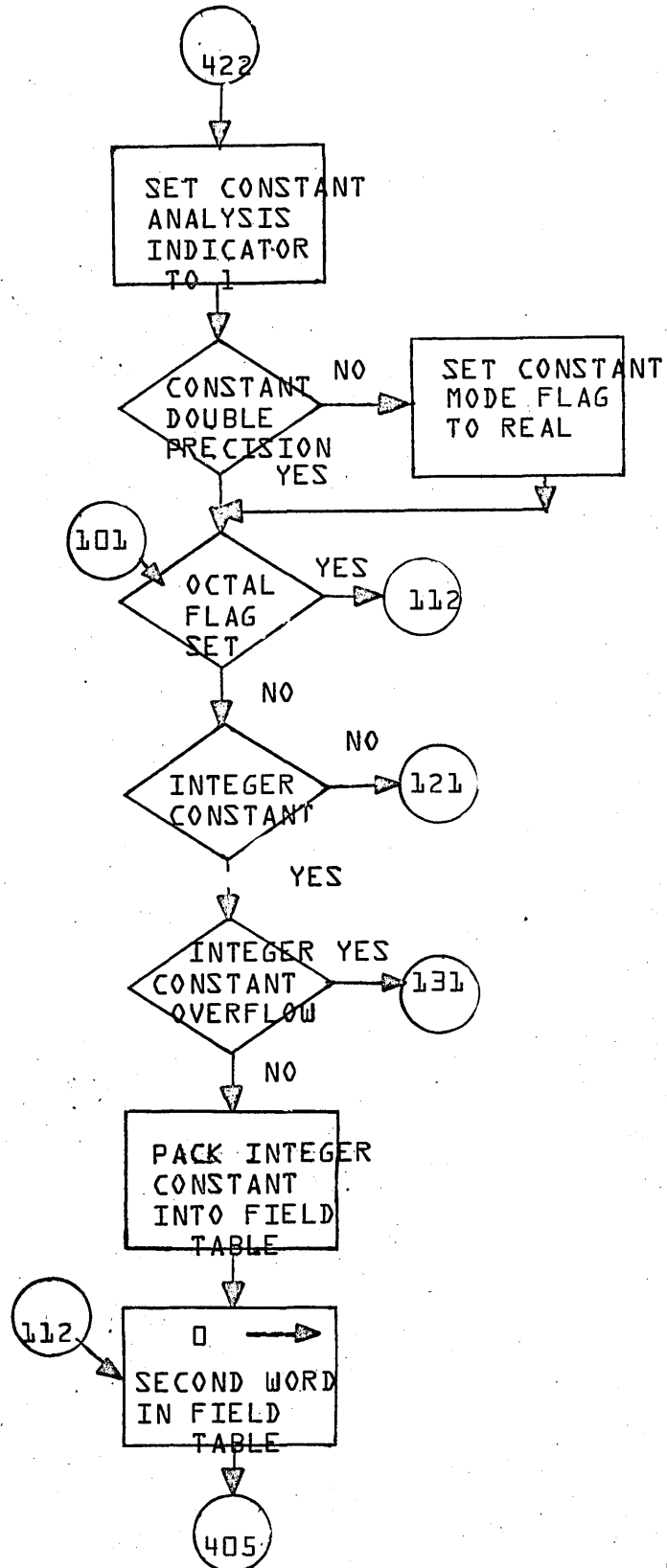
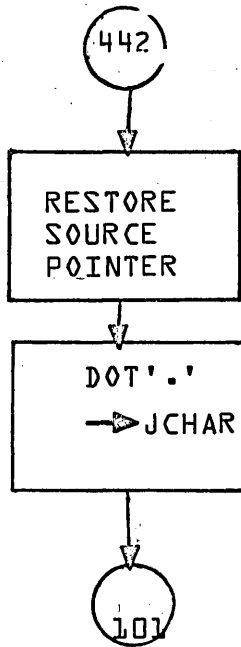
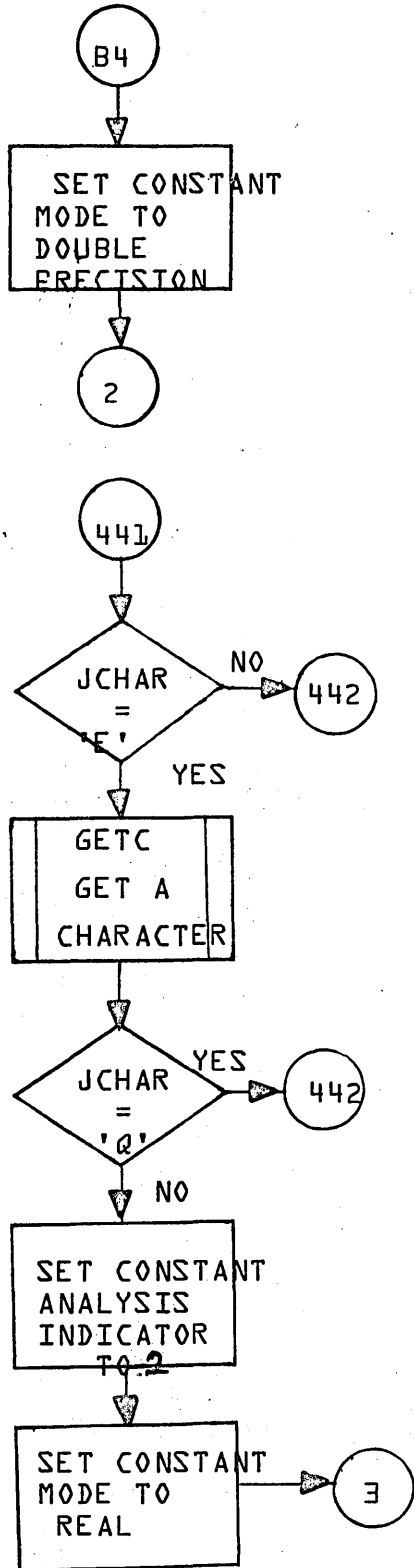
DOCUMENT CLASS IMS PAGE NO. 2-124  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

GETF  
 Page 10 of 14



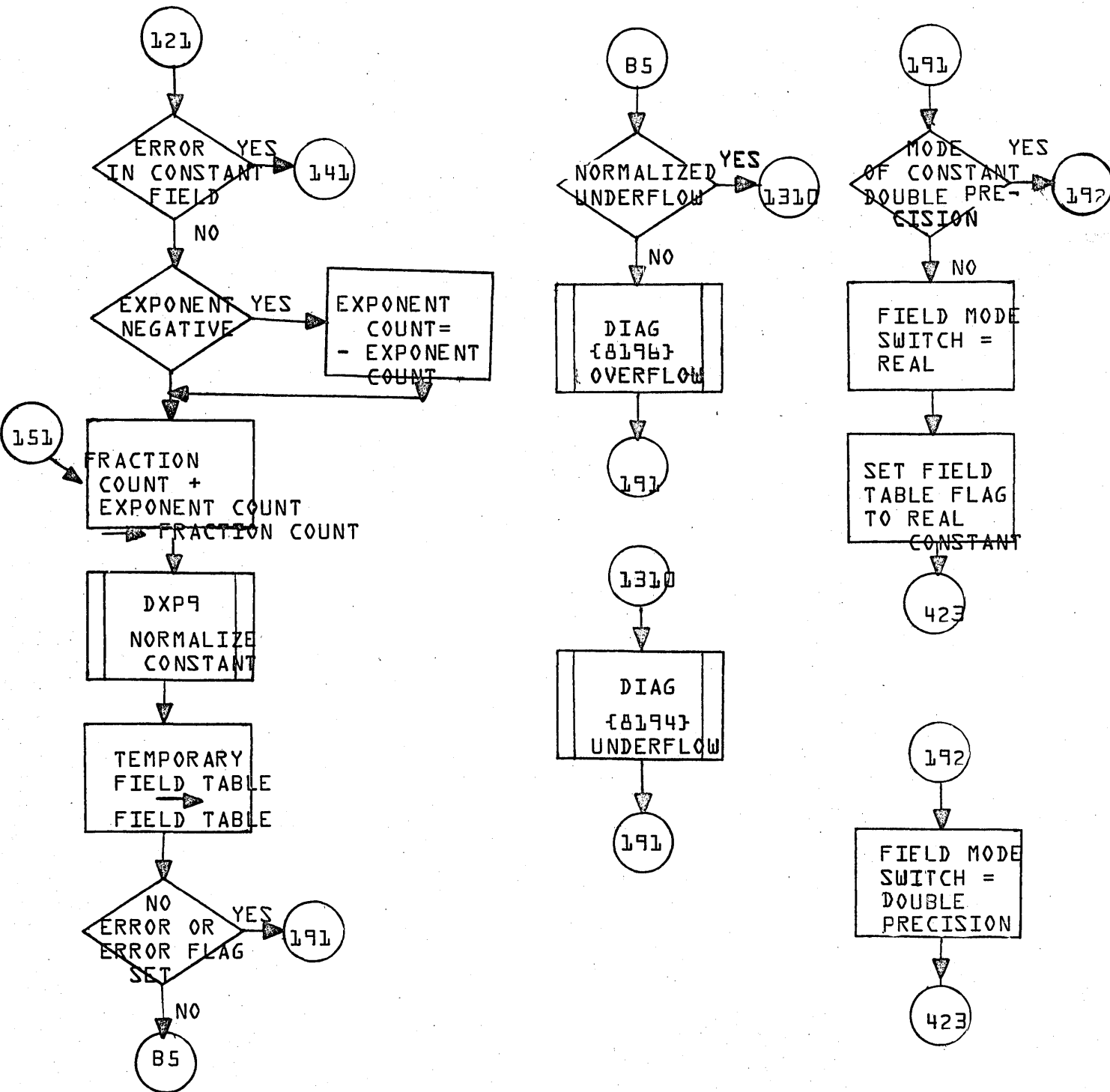
DOCUMENT CLASS IMS PAGE NO. 2-125  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. COO5\*3.1 A/B MACHINE SERIES 1700

GETF  
 Page 11 of 14



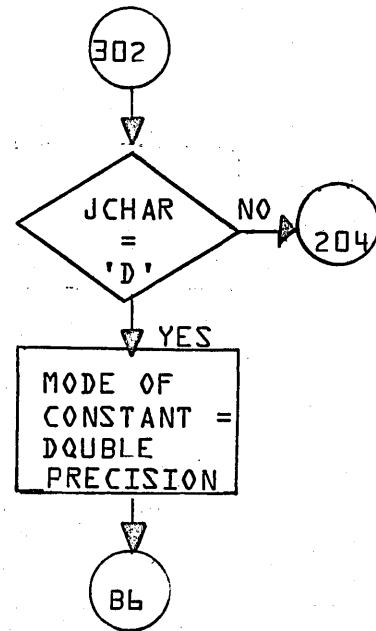
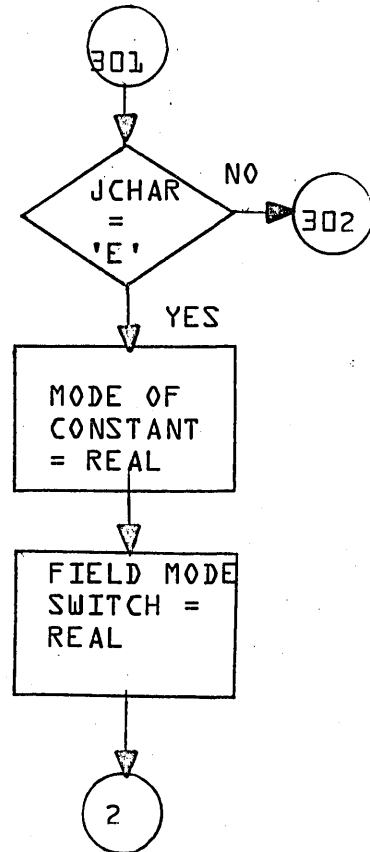
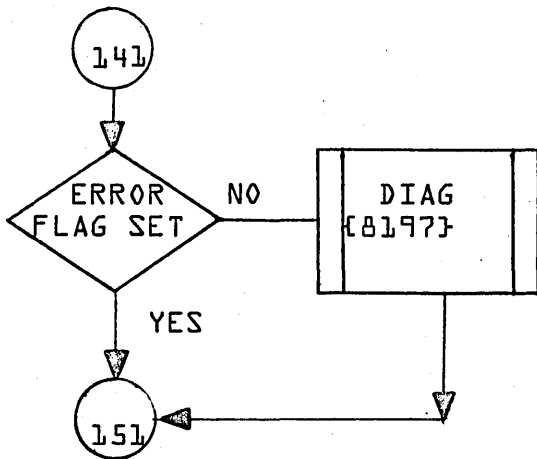
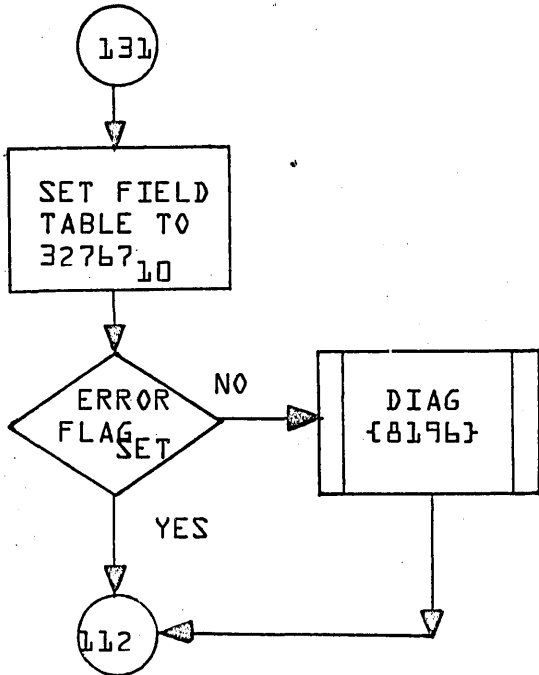
DOCUMENT CLASS IMS PAGE NO. 2-126  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

GETF  
 Page 12 of 14



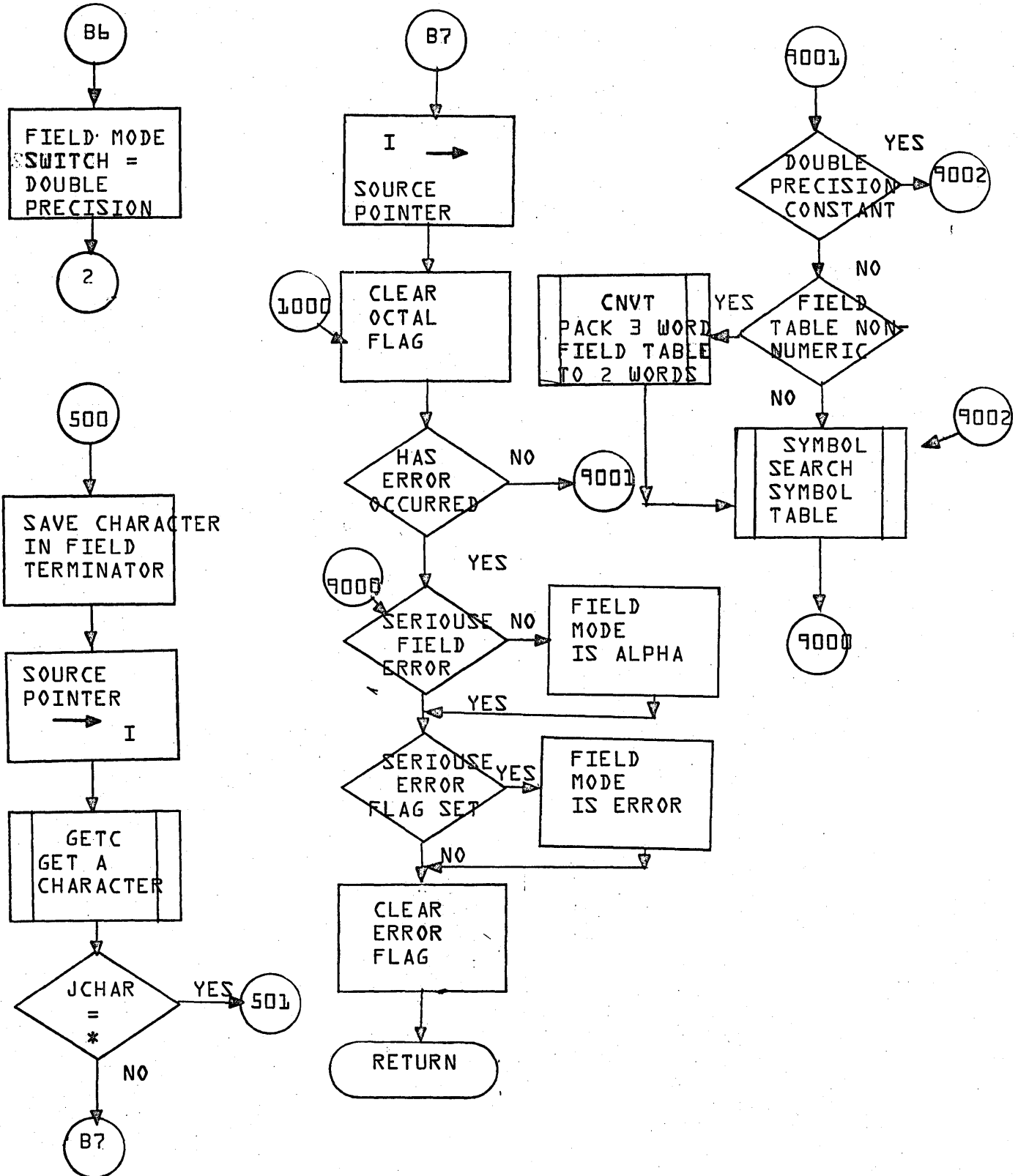
DOCUMENT CLASS IMS PAGE NO. 2-127  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

GETF  
 Page 13 of 14



DOCUMENT CLASS IMS PAGE NO. 2-128  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

GETF  
 Page 14 of 14

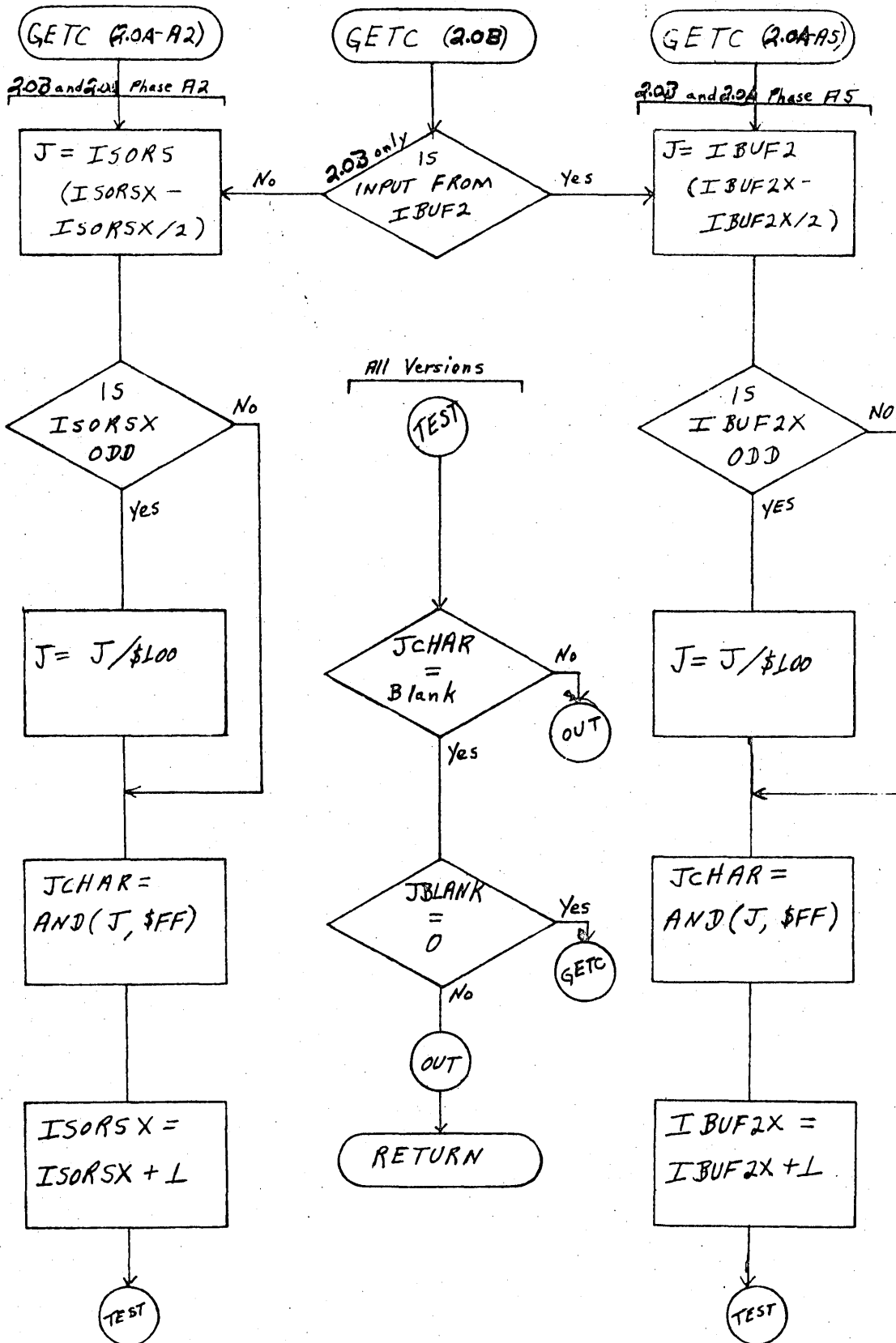




CONTROL DATA CORPORATION  
LA JOLLA RESOURCE CENTER DIVISION

DOCUMENT CLASS IMS PAGE NO. 2-129 thru 2-135  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

FAGES 2-129 THRU 2-135  
ARE INTENTIONALLY LEFT BLANK



DOCUMENT CLASS \_\_\_\_\_ PAGE NO. 2-137  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT NO. C00 5 VERSION 2.0 MACHINE SERIES 1700

2.3.2.4 Function IGETCF

IGETCF is used by subroutine GNST to get a specific character from an input string.

Input parameters:

IMAGE - location of start of string

ICOLMN - character within string

2.3.2.5 Subroutine STCHAR

STCHAR is used by subroutine GNST to store an eight bit character into a character string.

Input parameters:

IMAGE - location of start of string

ICOLMN - character within string

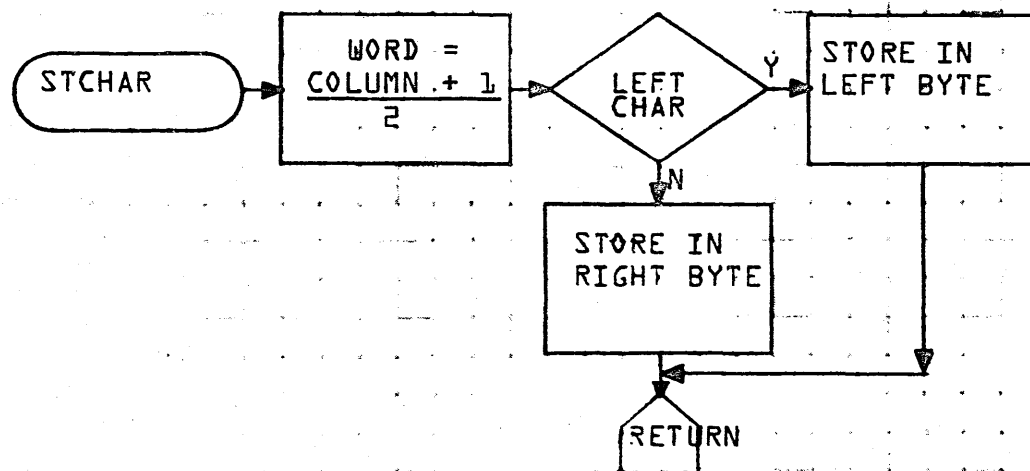
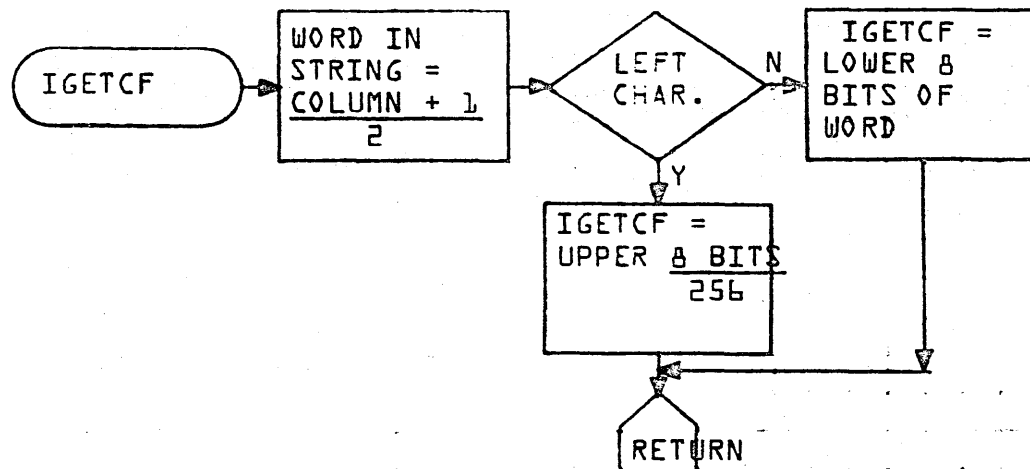
ICHARC - character to store

A

B

C

D



## CONTROL DATA CORPORATION

SOFTWARE DOCUMENT

SAMPLE CODE FLOWCHART DECISION TABLE OTHER 

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	IGETCF, STCHAR			PROJECT MGR.			
		PAGE	1 OF 1	PROJECT NAME			
NUMBER		ISSUE DATE		TASK NO.			
DRAWN BY		DATE		TASK NAME			

DOCUMENT CLASS IMS PAGE NO. 2-139  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

### 2.3.3 Special Field Handling Routines

#### 2.3.3.1 CONSUB

Given an element of an array, the CONSUB subroutine is used to compute the increment from the subscripts of the element in order to reference the location of the element in the array. The CONSUB subroutine requires that subscripts be integer constants. It computes the nth element of an array.

Given the array defined by

DIMENSION A {I,J,K},

and the element

A {I1, J1, K1}

the increment is computed and recorded in IVCFLG as follows:

IVCFLG = {K1-1}\*I\*J+{J1-1}\*I+I1

The resulting value in IVCFLG is such that

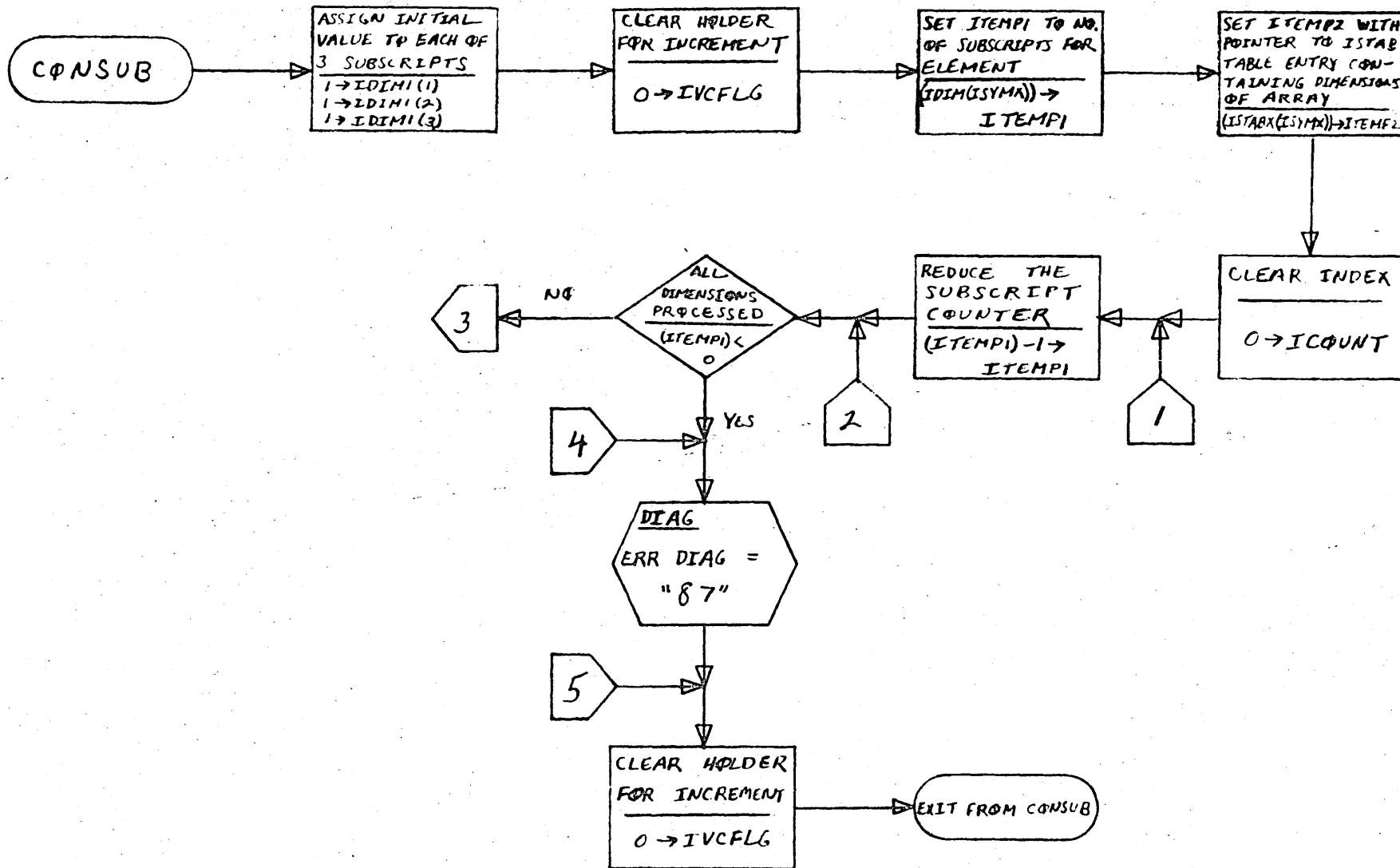
A{I1, J1, K1} = the IVCFLGth element in the array A{I, J, K}. The original dimensions of the array A {I, J, K} are obtained from an entry in the ISTAB table.

#### 2.3.3.1.1 Errors

The CONSUB subroutine gives diagnostics for the following errors:

1. An element with no subscripts,
2. an element with more subscripts than the array of which it is a member has dimensions,
3. an element with a subscript which exceeds the value of the corresponding dimension for the array of which it is a member,
4. an element with a subscript which is something other than an integer, and
5. a format error in which the subscripts are not separated by commas and/or not enclosed in parentheses.

Errors 1, 2, 4 & 5 will cause CONSUB to terminate operation. Following the error indication, the holder IVCFLG is set to zero, and an exit is made from CONSUB. Error 3 will not cause the CONSUB operation to terminate following the error indication.

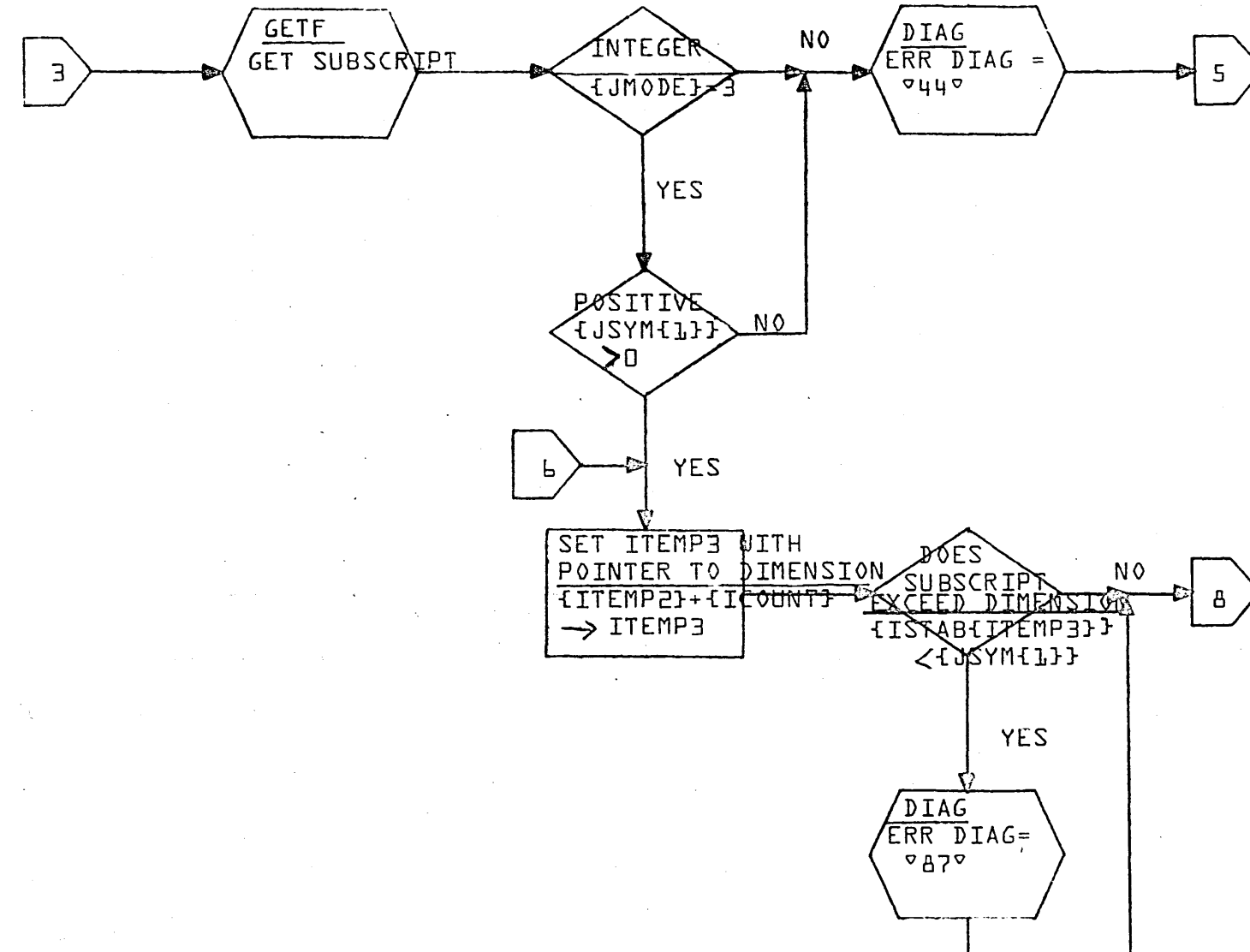


**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	<i>IMS</i>	MACH. TYPE	<i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	<i>CONSUB</i>	PAGE <i>1</i> OF <i>3</i>		PROJECT MGR.			
NUMBER	ISSUE DATE			PROJECT NAME			
DRAWN BY	DATE <i>3-67</i>			TASK NO.			
				TASK NAME			

2-140

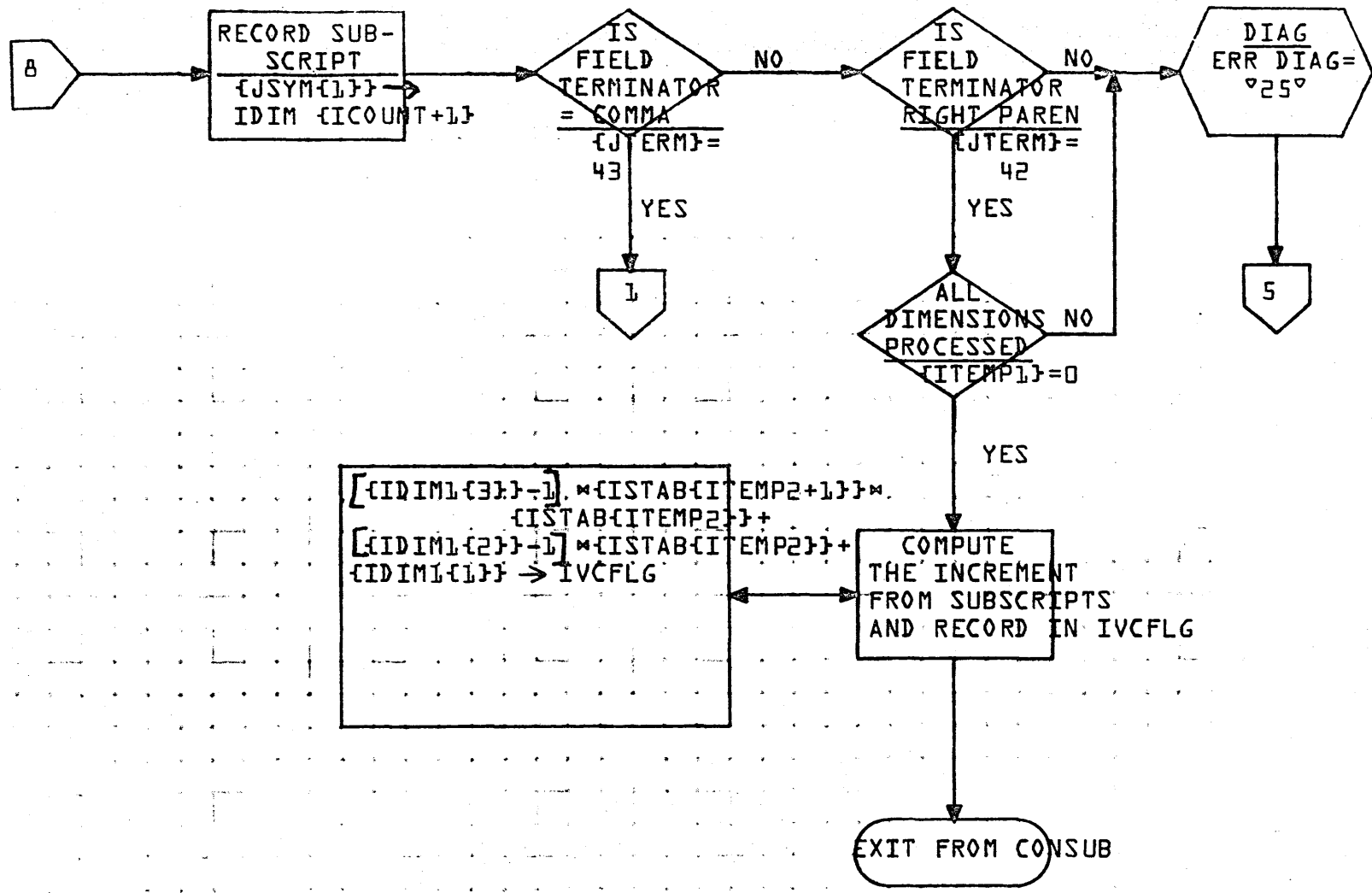


CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	CONSUB			PROJECT MGR.			
	PAGE 2 OF 5			PROJECT NAME			
NUMBER		ISSUE DATE		TASK NO.			
DRAWN BY		DATE		TASK NAME			

T4T-2



**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	CONSUB			PROJECT MGR.			
		PAGE	3 OF 3	PROJECT NAME			
NUMBER		ISSUE DATE		TASK NO.			
DRAWN BY		DATE		TASK NAME			



DOCUMENT CLASS TMS PAGE NO. 2-143  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT NO. COC5 VERSION 2.0 MACHINE SERIES 1700

2.3.3.2 CFIVOC - Checks for Integer Variable or Integer Constant

CFIVOC reads the next field in ISORS. If the field is an integer variable or constant, the field is stored in the symbol table (if necessary), the type set to integer and the class set appropriately. IVCFLG is set to signal the type of field

0 = not integer variable or constant  
1 = constant  
2 = variable

2.3.3.3 CKIVC - Checks for Integer Variable or Constants and Provides Diagnostics

CKIVC calls CFIVOC to read next ISORS field. If the field is neither an integer variable or an integer constant (IVCFLG = 0), diagnostic 63 is output.

2.3.3.4 CKNAME - Checks for Integer Name

CKNAME calls CFIVOC to read the next ISORS field. If the field is not an integer name (IVCFLG = 2), diagnostic 64 is output.

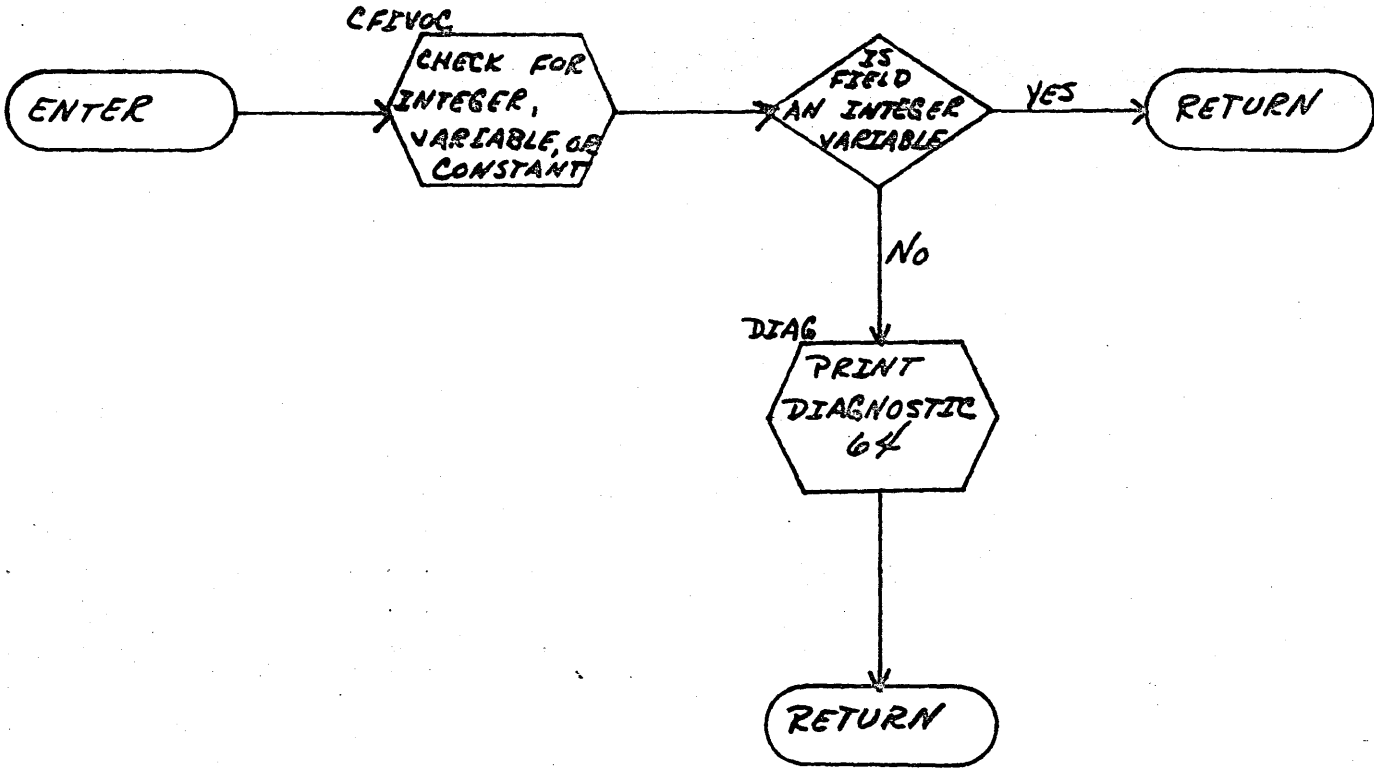
CKNAME

A

B

C

D



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE

FLOWCHART

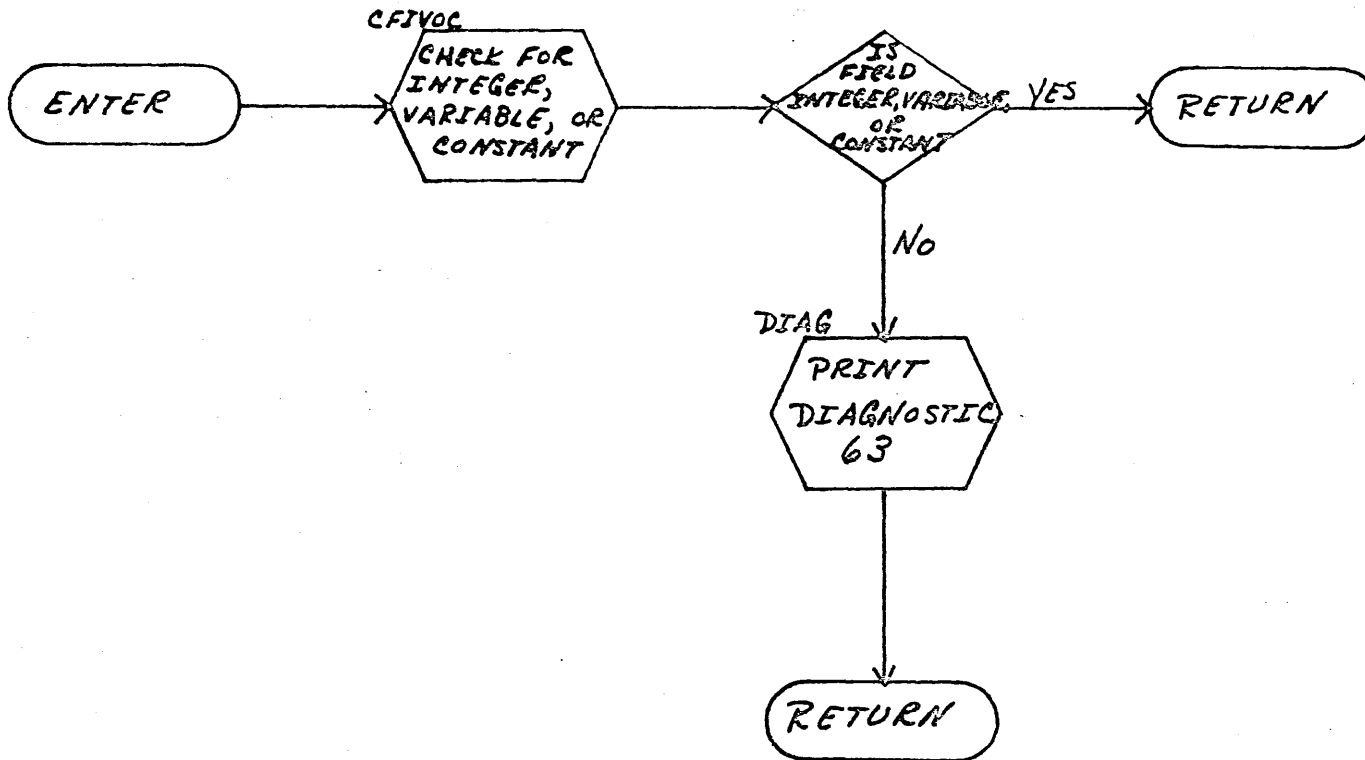
DECISION TABLE

OTHER

DOCUMENT CLASS	<i>I MS</i>	MACH. TYPE	<i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	<i>CKNAME</i>	PAGE <i>1</i> OF <i>1</i>		PROJECT MGR.			
NUMBER	ISSUE DATE			TASK NO.			
DRAWN BY	DATE			TASK NAME			

2-144

CKIVC



A  
B  
C  
D

<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS <b>IMS</b>	MACH. TYPE <b>1700</b>	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE <b>CKIVC</b>	PAGE <b>1</b> OF <b>1</b>		PROJECT MGR.		
	NUMBER	ISSUE DATE	TASK NO.			
	DRAWN BY	DATE	TASK NAME			

54-145

DOCUMENT CLASS IMS PAGE NO. 2-146  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005 V.2.0 MACHINE SERIES 1700

## 2.3.4 Diagnostic Handling Routine

### 2.3.4.1 Subroutine DIAG

DIAG is a multiphase subroutine which outputs error diagnostics. The error message output is of the form:

\*C E,S           xxxxxx

Where C is F or N for fatal or non-fatal  
E is the error number  
S is the offending statement number  
xxxxxx are six characters from the source  
input buffer surrounding the current  
setting of ISORSX

The input to DIAG is one packed parameter giving the error number in bits 11-0 and flags in bits 14, 13, 12.

The flags are:

Bit 14    0 = print statement number  
          1 = don't print statement number

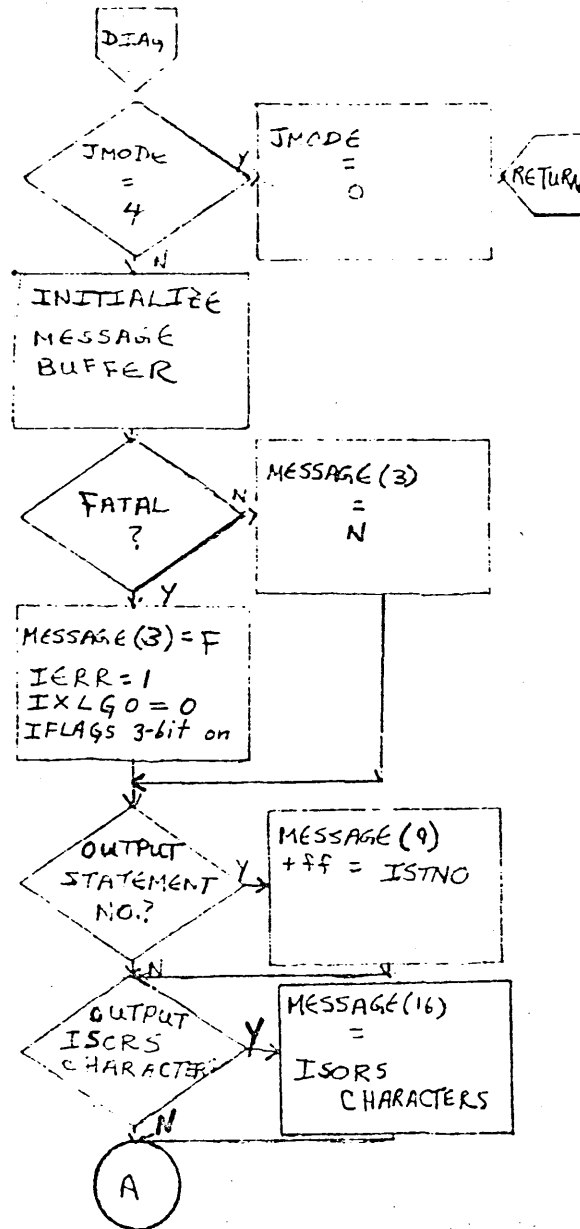
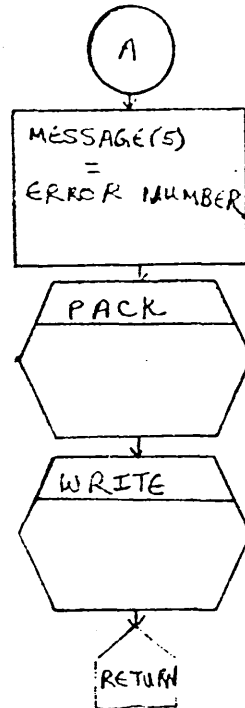
Bit 13    0 = fatal error  
          1 = non-fatal error

Bit 12    0 = Print ISORS characters  
          1 = Don't print ISORS characters

If this is a fatal error, IERR is set  $\neq$  0.

If DIAG is entered with JMODE = 4. (error mode in GETF), no error is output and JMODE is set = 0. This is to minimize the output of redundant errors.

If this is a fatal error, the "X" option flag IXLGO is cleared and bit 3 is set in IFLAGS.



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE  
FLOWCHART   
DECISION TABLE  
OTHER

DOCUMENT CLASS <i>IMS</i>	MACH. TYPE <i>1700</i>	PROJECT NO.	REV.	APPROVED	DATE
DOCUMENT TITLE <i>DIAG</i>		PROJECT MGR.			
	PAGE <i>1</i> OF <i>1</i>	PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.			
ISSUED BY	DATE	TASK NAME			

2-147

DOCUMENT CLASS IMS PAGE NO 2-148  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

#### 2.3.4.2 Subroutine CONV

CONV converts a binary integer to external code (ASCII) and puts the result in an output buffer, one character per word. Its input parameters are:

IX1 number to convert

IX2 output buffer - six words long

CONV is coded in assembler. The version of CONV used in PHASE A cannot convert negative numbers, while the version used in PHASE E can convert any binary integer.

CONV

INITIALIZE  
OUTPUT  
BUFFER TO  
BLANKS

Phase E only

NUMBER < 0

NUMBER =  
- NUMBER  
SWITCH = ON

CONVERT  
NUMBER

SWITCH ON

PUT IN  
-

RETURN

CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE   
FLOWCHART   
DECISION TABLE   
OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.		REV	APPROVED	DATE
DOCUMENT TITLE	CONV			PROJECT MGR.				
		PAGE	1 OF 1	PROJECT NAME	FORTRAN			
NUMBER		ISSUE DATE		TASK NO.				
DRAWN BY		DATE	3-67	TASK NAME				

2-749

DOCUMENT CLASS IMS PAGE NO. 2-150  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. 0005 V.2.0 MACHINE SERIES 1700

#### 2.3.4.3 Subroutine PACK

PACK packs a buffer which comes to it one character per word into the form of two characters per word.

Input parameters:

IX1 Location of buffer

IX2 number of words to pack

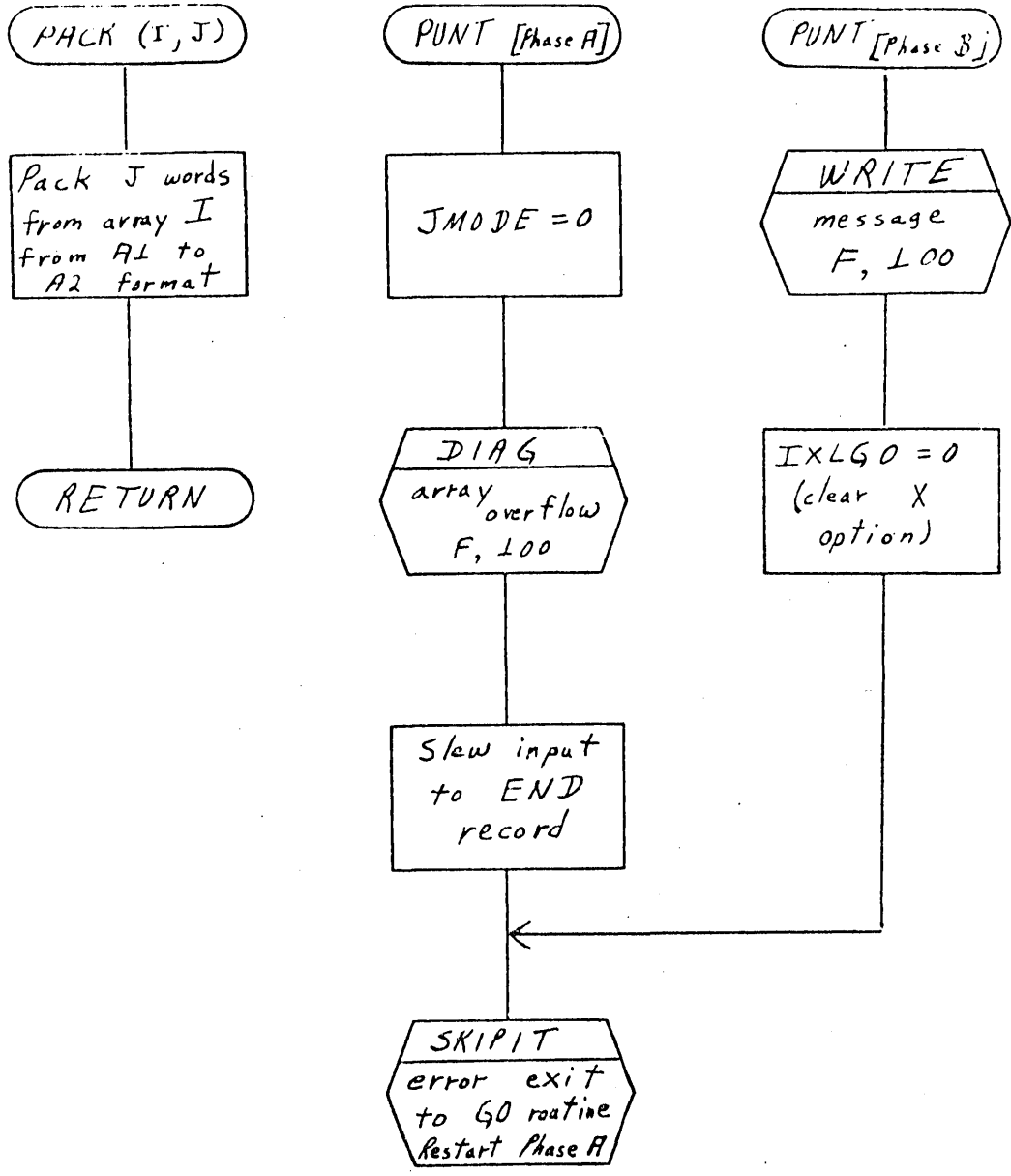
#### 2.3.4.4 Subroutine PUNT

PUNT is called when a catastrophic table overflow occurs in the compiler. It outputs error diagnostic 100 and calls SKIPIT. The Phase A version slews the input to an END card before calling SKIPIT.

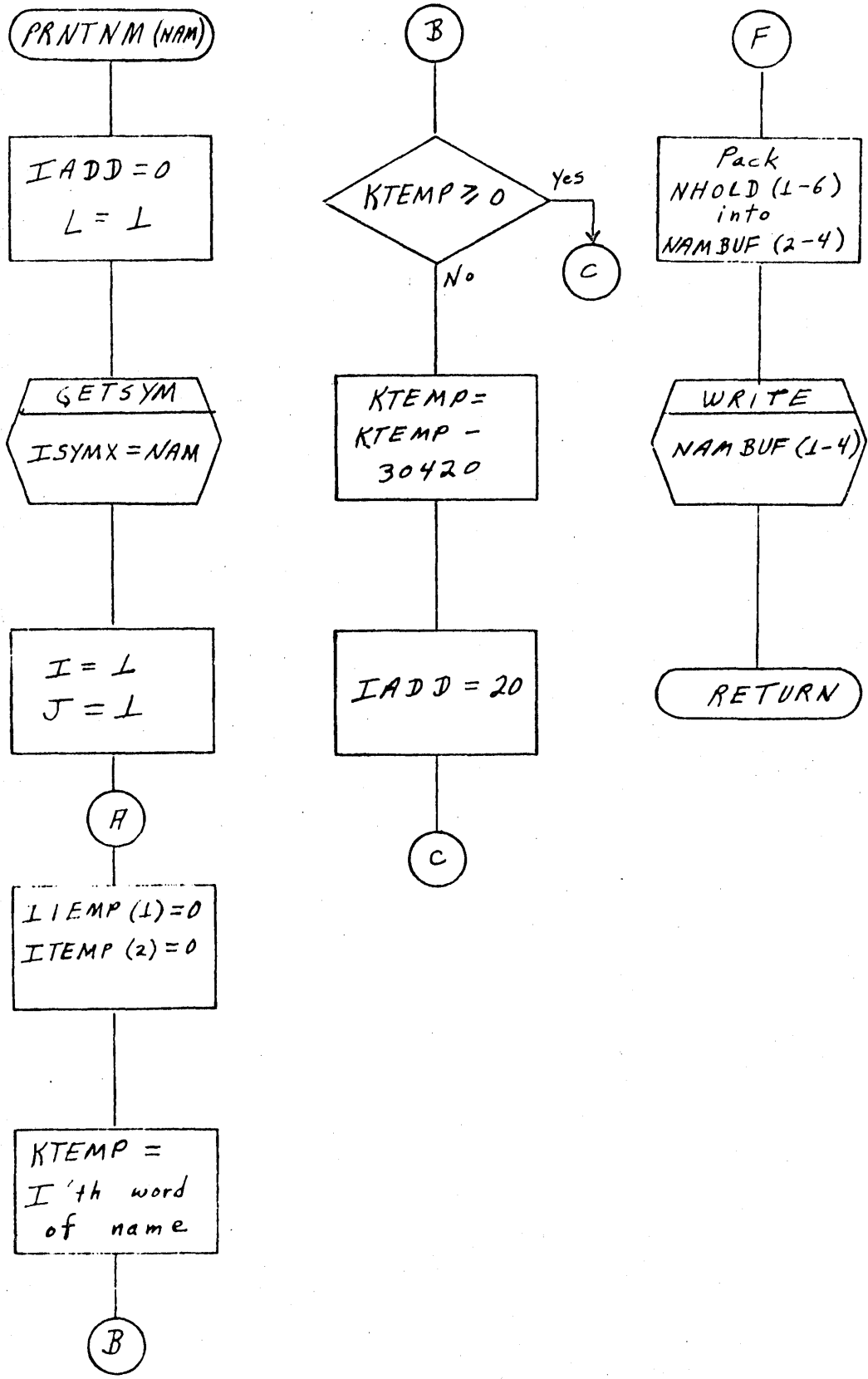
#### 2.3.4.5 Subroutine PRNTNM

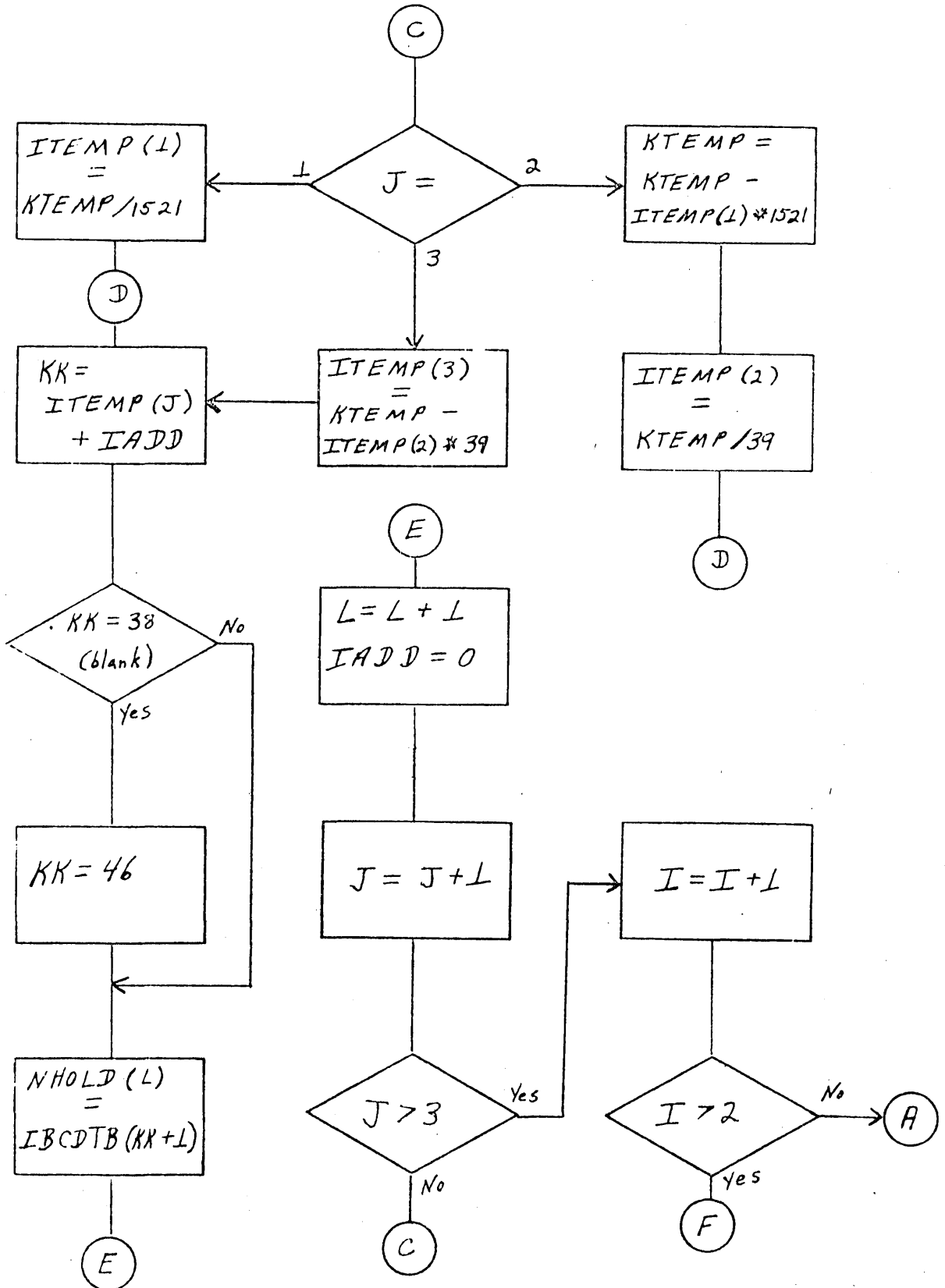
PRNTNM is called by subroutine PEQVS when an illegal equivalence chain is encountered. PRNTNM converts the name of the illegal variable to external code (ASCII) and outputs it through the use of subroutine IOPR (entry point WRITE).





DOCUMENT CLASS IMS PAGE NO. 2-152  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. COO5 V.2.0 MACHINE SERIES 1700





PROGRAM CLASS

IMS

PAGE NO.

2-154

PROGRAM NAME

1700 MASS STORAGE FORTRAN

PROGRAM MODEL NO.

0005 V.E.O

MACHINE SERIES

1700

### 2.3.4.6 Subroutine IOPR

This assembly-language routine performs all I/O for the compiler. In order to optimize core and disk use, there is a separate version of IOPR for each phase of each {2.0A and 2.0B} compiler; but all versions share the following functional description.

#### Entry points

IOPR is the entry called by GO at the start of each phase. It allocates the disc scratch area, initializes pointers, and begins input.

READ and WRITE perform both unit record and mass storage I/O. Unit record I/O in phase A is buffered; records on the scratch files are packed by WRITE and unpacked by READ.

RESET is available in phase D/E; it performs a logical "rewind" of a scratch file.

EXIT issues an EXIT request to MSOS.

#### Disc allocation

The speed of the compiler is highly dependent on the placement of its files within the MSOS-allotted disk scratch area. This area is allocated starting at the first sector of the first cylinder beyond the current top of the load-and-go file; seek operations are minimized by never allowing a file to "break" between two cylinders.

The first portion of this area is allocated for local files. (Even though the present phase may not have locals, this area must be stepped over in order to find the symbol table and scratch files.) The partitioning of this area is "customized" for each phase, generally leaving each file several sectors more than it requires to allow for later expansion.

Following the local files, an area is assigned to the symbol table. Since the table is organized in eleven-sector "pages", this area starts 11n sectors from the bottom of a cylinder and overlaps onto the next cylinder. The disk allocation is such that only programs with a very large number of symbols will use any of the SYMTAB pages on the next cylinder, and so that the most-used local files will share the "prime" SYMTAB cylinder.

DOCUMENT CLASS IMS PAGE NO. 2-155  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005 V.2.0 MACHINE SERIES 1700

After the symbol table, the remainder of the scratch area is available for the two scratch files. These files, like the symbol table, are organized in "pages" of several sectors; in order to minimize length of seeks, pages of file 1 are interleaved with pages of file 2. No effort is made to keep scratch pages from crossing cylinder boundaries, since the "broken" page would be referred to only once per phase.

#### Scratch file packing

To minimize disk usage, the two scratch files are packed into streams of logical records without regard for sector boundaries. (The first word of each logical record contains the total record length.) The files are then broken up into "pages" of several sectors; a "page" being as much of the file as will fit into core. (In several cases, a phase keeps only part of a page in core at a time; the page size is chosen for maximum efficiency over all four phases of a compiler version.) A call to READ causes copying of a logical record from the unpacking area to the user-designated address; if the end of the unpacking area is reached before the entire record is moved, then the next page of the input file is read into the unpacking area and the copying proceeds. A call to WRITE similarly loads the output packing area, which is similarly emptied when full. (A special call is provided to force the output onto disk at the end of each phase.)

It is important to note that on scratch-file I/O, the logical record length is controlled only by word 1 of the record; The record-length parameter of the call to READ or WRITE is ignored.

#### Unit record buffering

Phase A unit record I/O...source statement input and listing... is buffered, so that phase A execution time is overlapped as much as possible. Space is provided in the phase A IOPR to hold one line of input and one or two lines of output; each line is associated with an FREAD or FWRITE request. When the IOPR entry is called, at the start of a phase, it activates the FREAD request that is to fill the input buffer. Thereafter, a call to READ will: (1) wait until the request is unthreaded; (2) copy the record just read, from the buffer

## CONTROL DATA CORPORATION

DIVISION

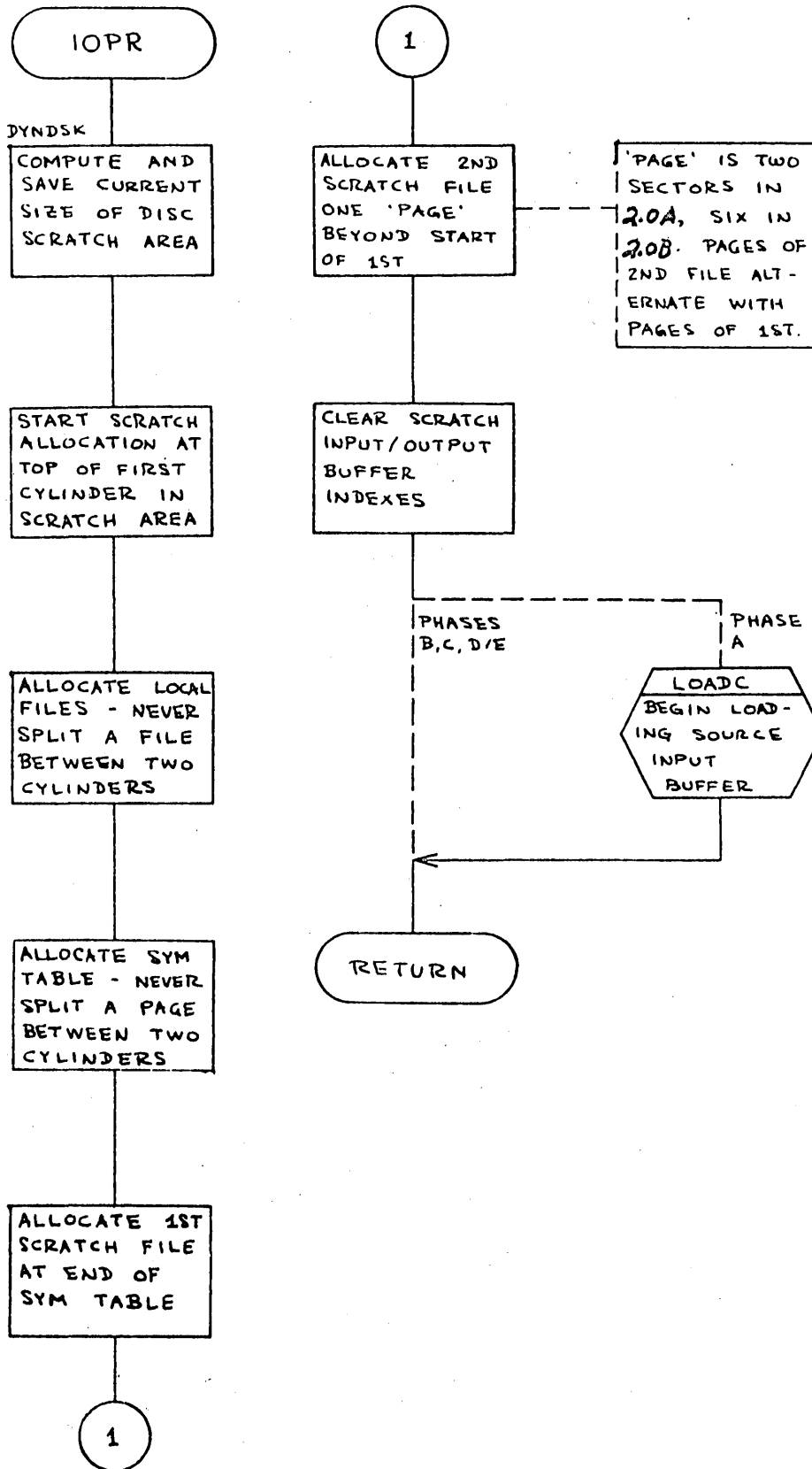
DOCUMENT CLASS \_\_\_\_\_ IMS \_\_\_\_\_ PAGE NO. \_\_\_\_\_ 2-156  
PRODUCT NAME \_\_\_\_\_ 1700 MASS STORAGE FORTRAN \_\_\_\_\_  
PRODUCT MODEL NO. \_\_\_\_\_ C005 V.2.0 \_\_\_\_\_ MACHINE SERIES \_\_\_\_\_ 1700 \_\_\_\_\_

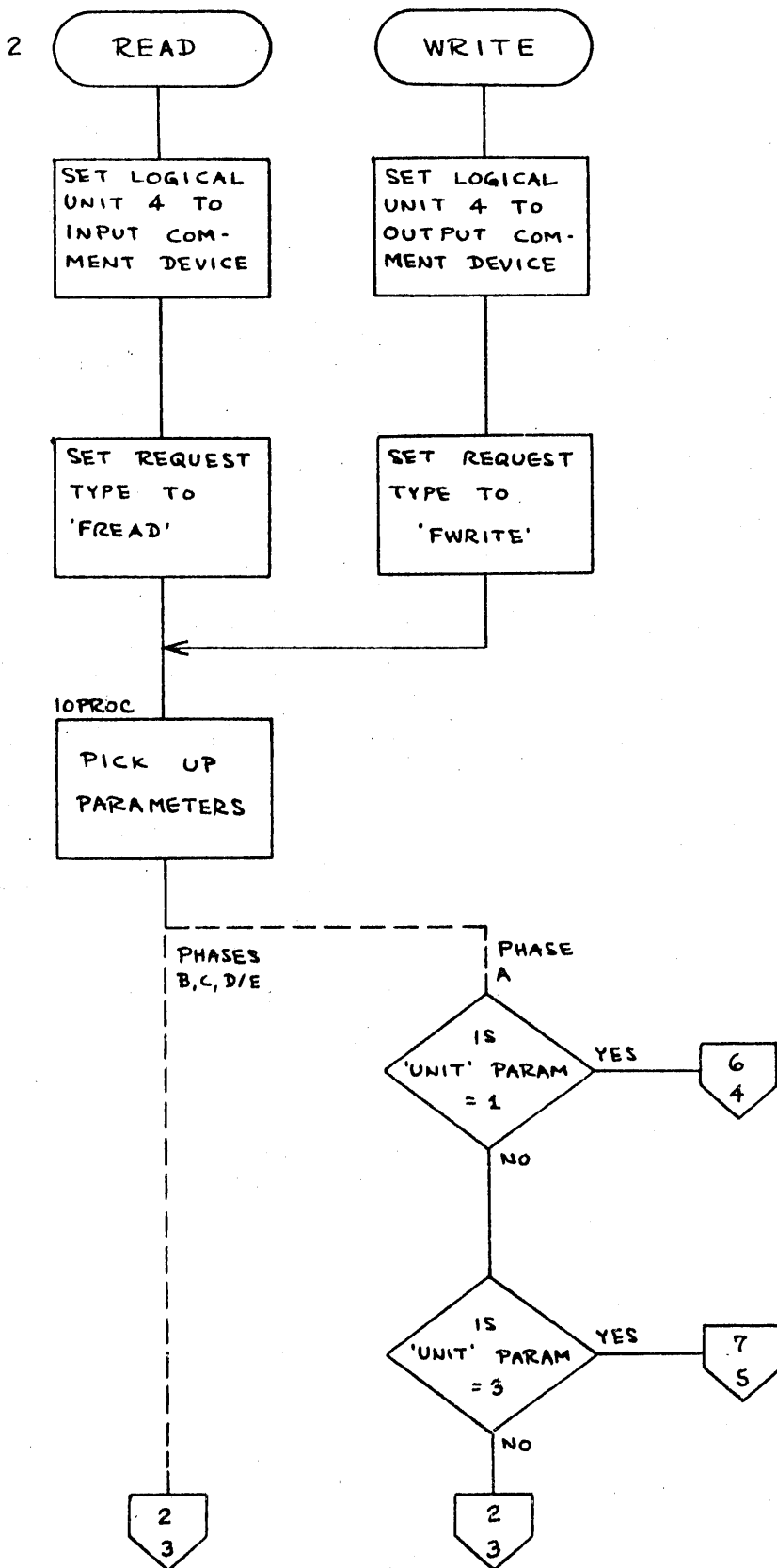
to the user's area; (3) reissue the FREAD request. Similarly, a call to WRITE waits for a buffer to be available, fills it, issues an FWRITE request for it, and returns.

The READ routine does not reissue the FREAD request if the buffer contains an END or MON card. The buffer is blanked before the request is issued. The requested record length is 41 words so that, regardless of actual record length, the operating system will store the actual last-word-address-plus-one outside the area to be processed by Fortran.

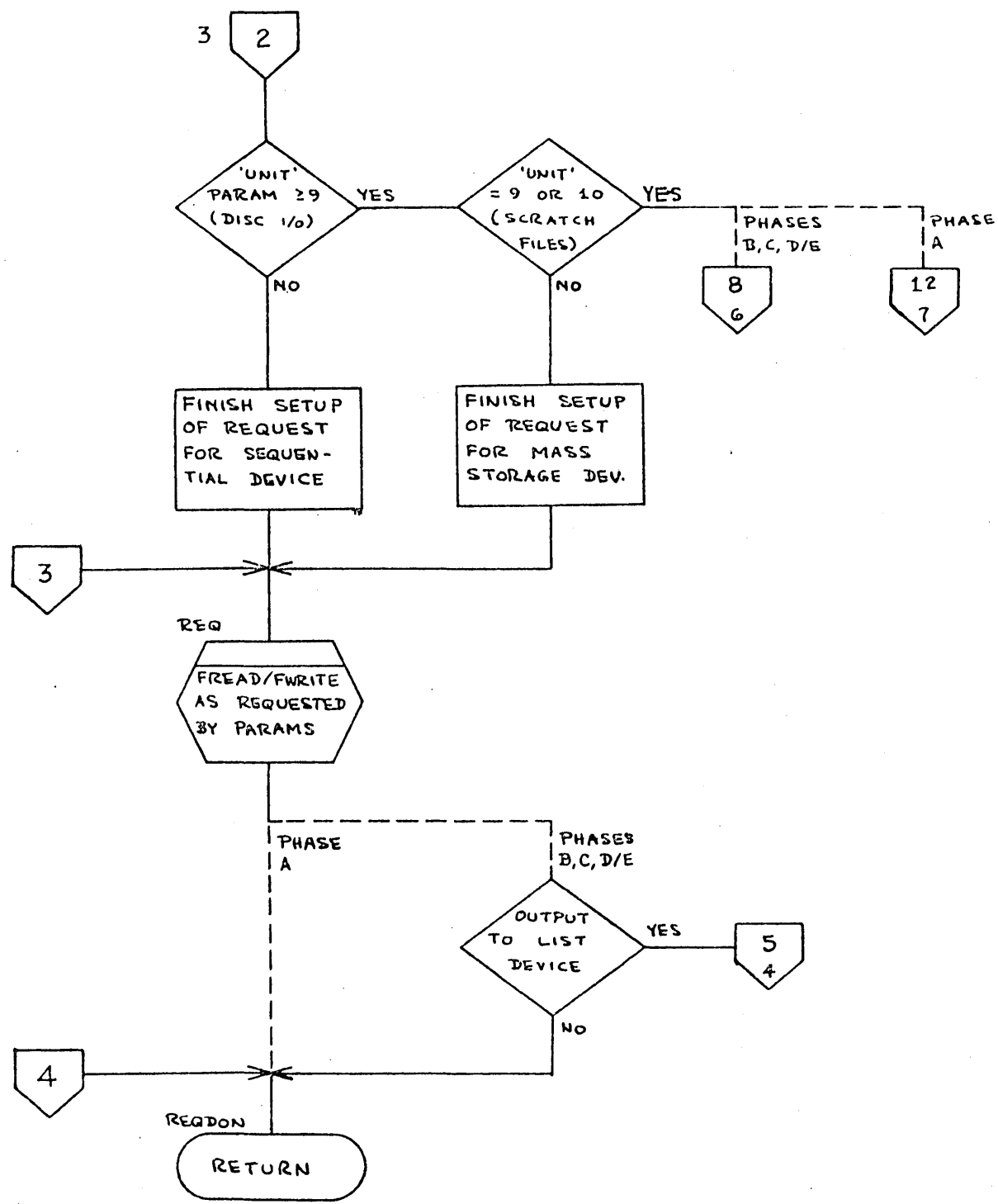
#### Page ejection

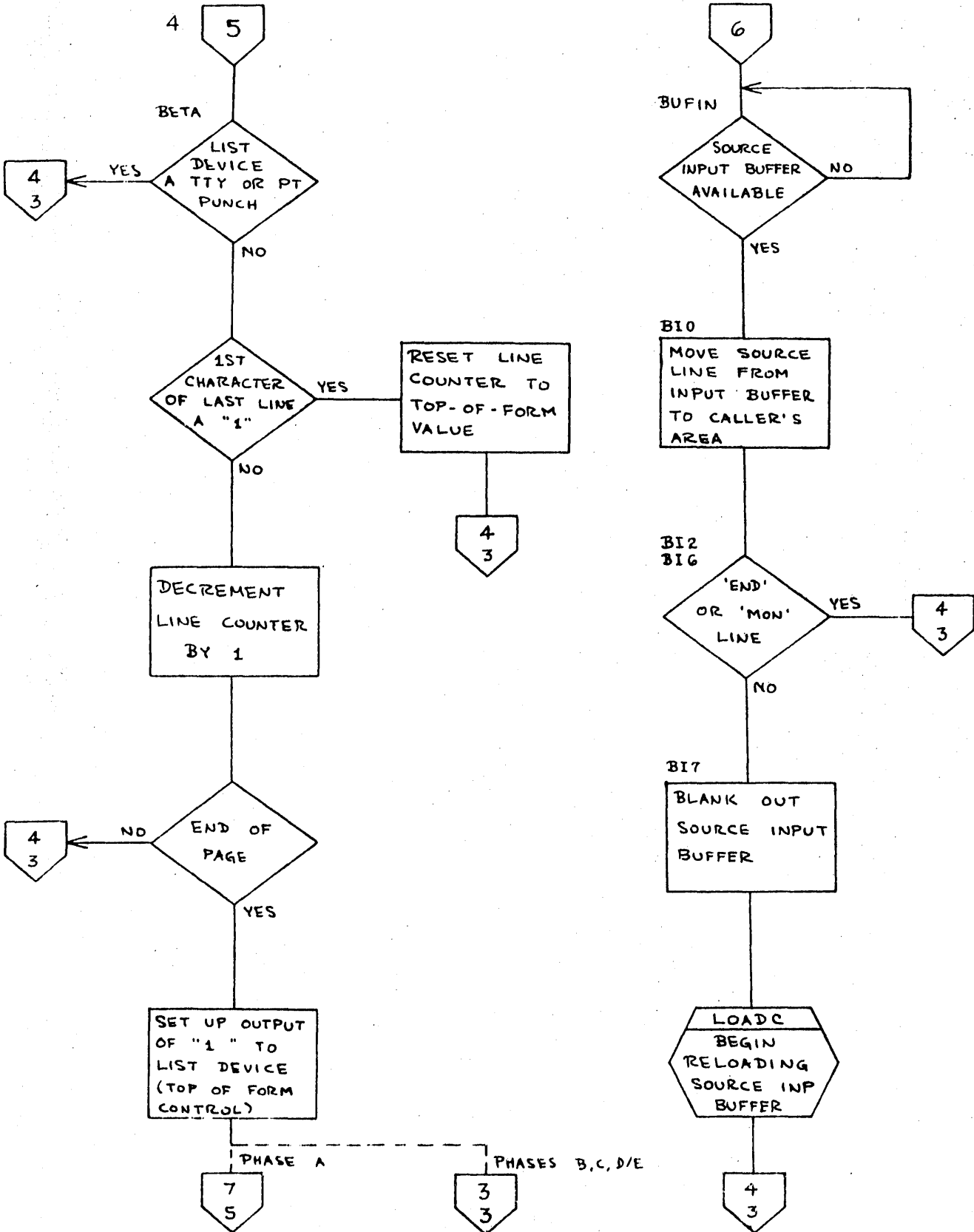
If the list output device is not a teletype or paper tape punch, IOPR issues a page-eject control (\$3120) after each 60 lines of output. The maximum and current number of lines per page are held in labelled common and are set by the phase A block data subprogram and subroutine PHASEA, respectively. PHASEA is responsible for issuing page ejects before the source and assembly listings.



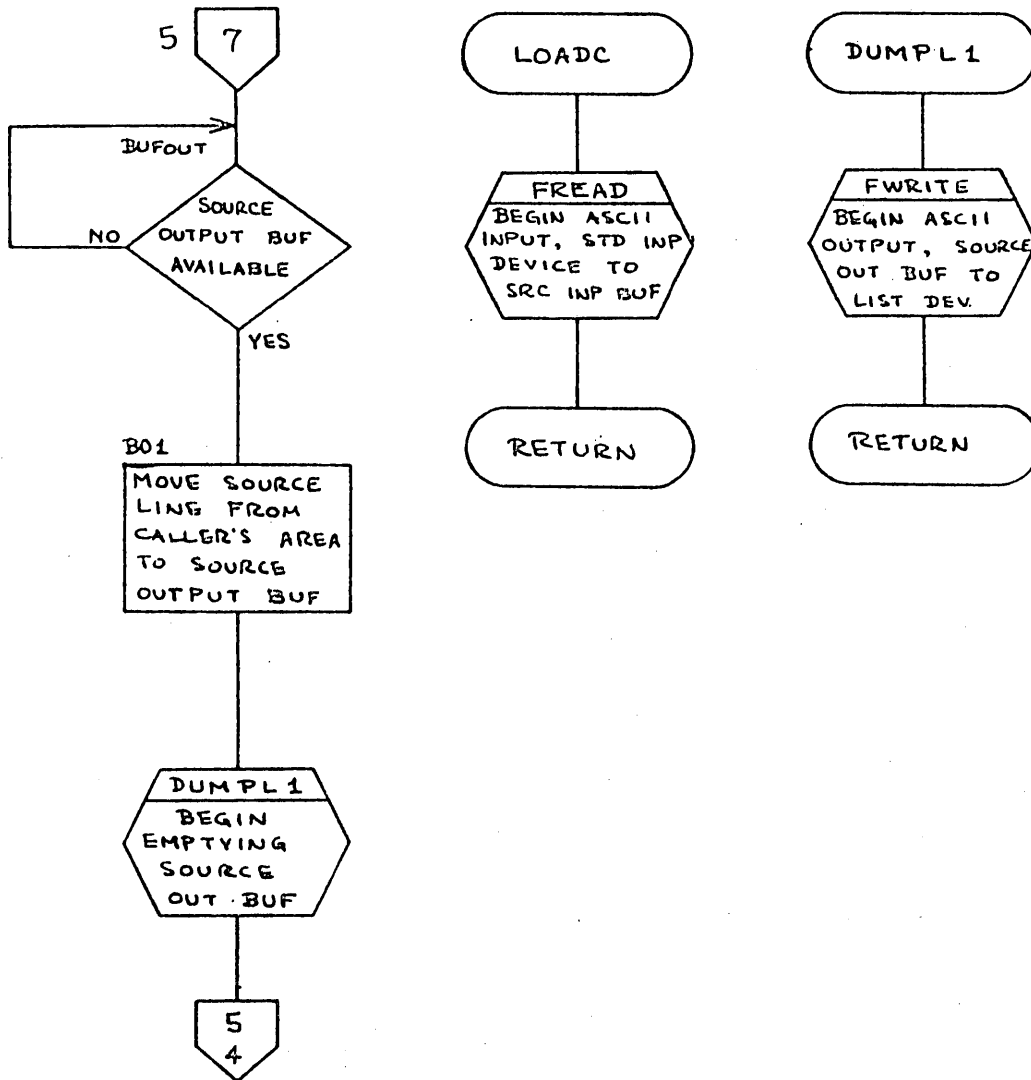


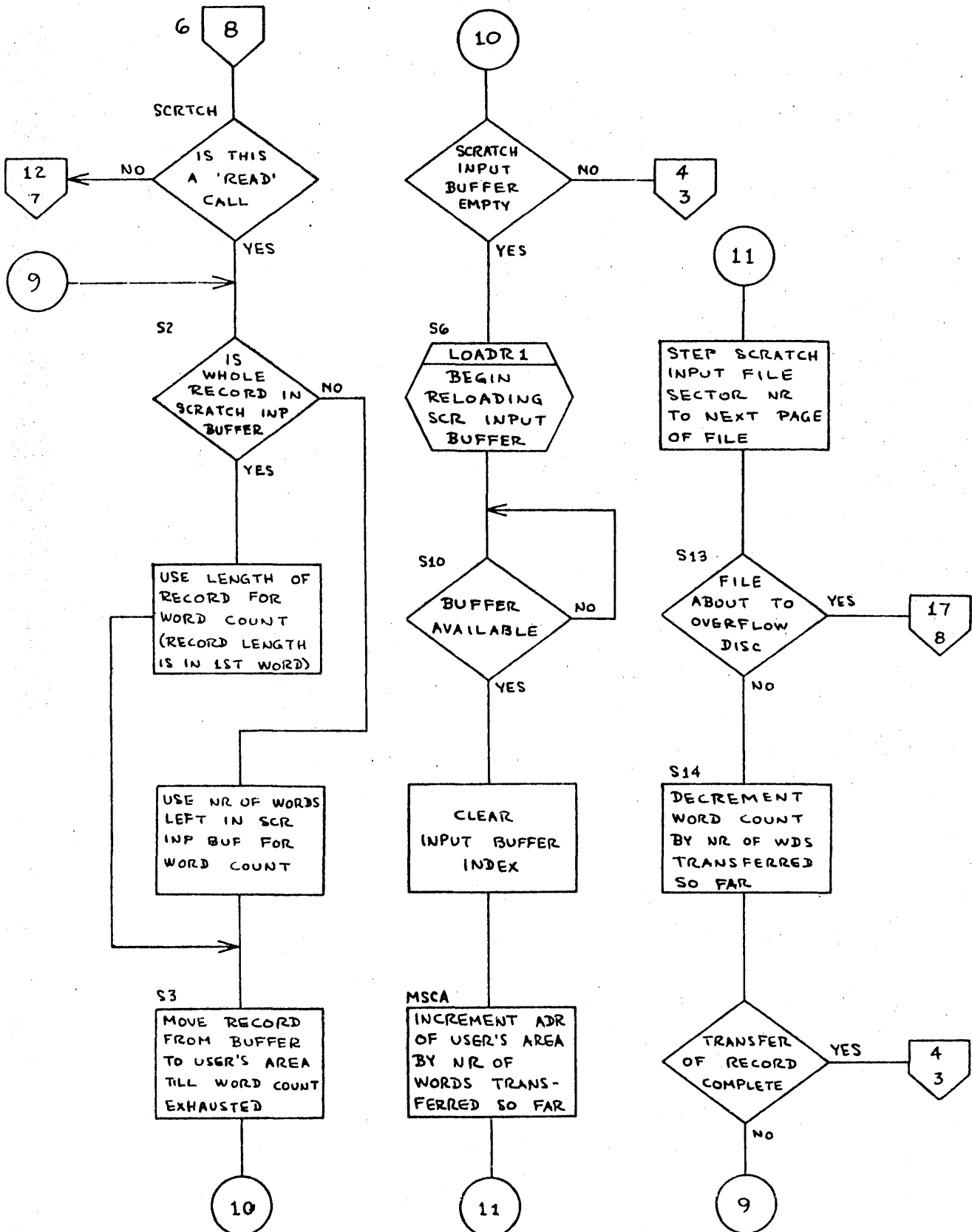


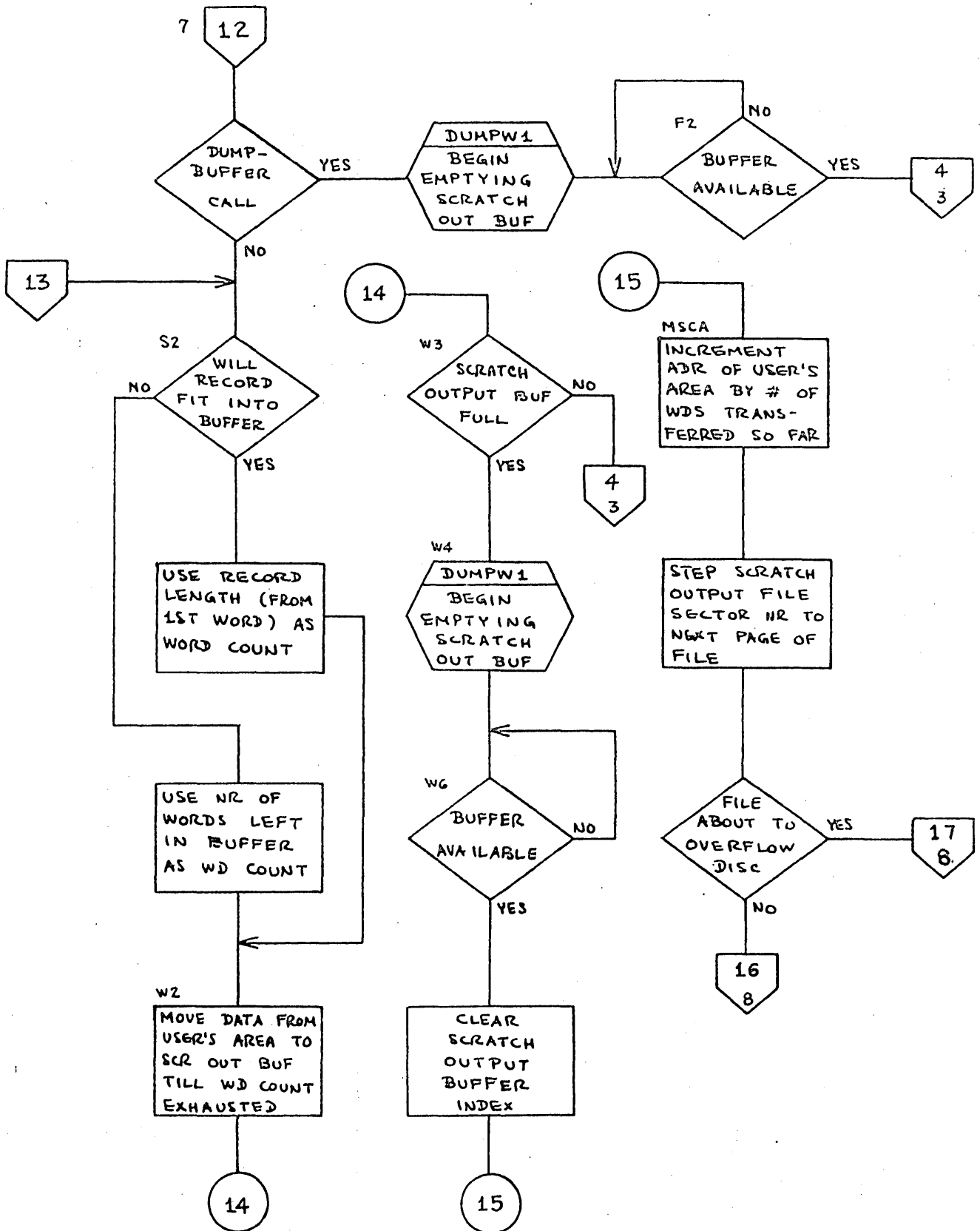




DOCUMENT CLASS IMS PAGE NO. 2-161  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005 V.2.0 MACHINE SERIES 1700



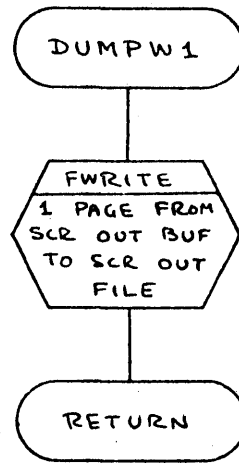
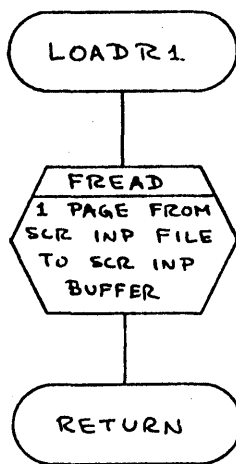
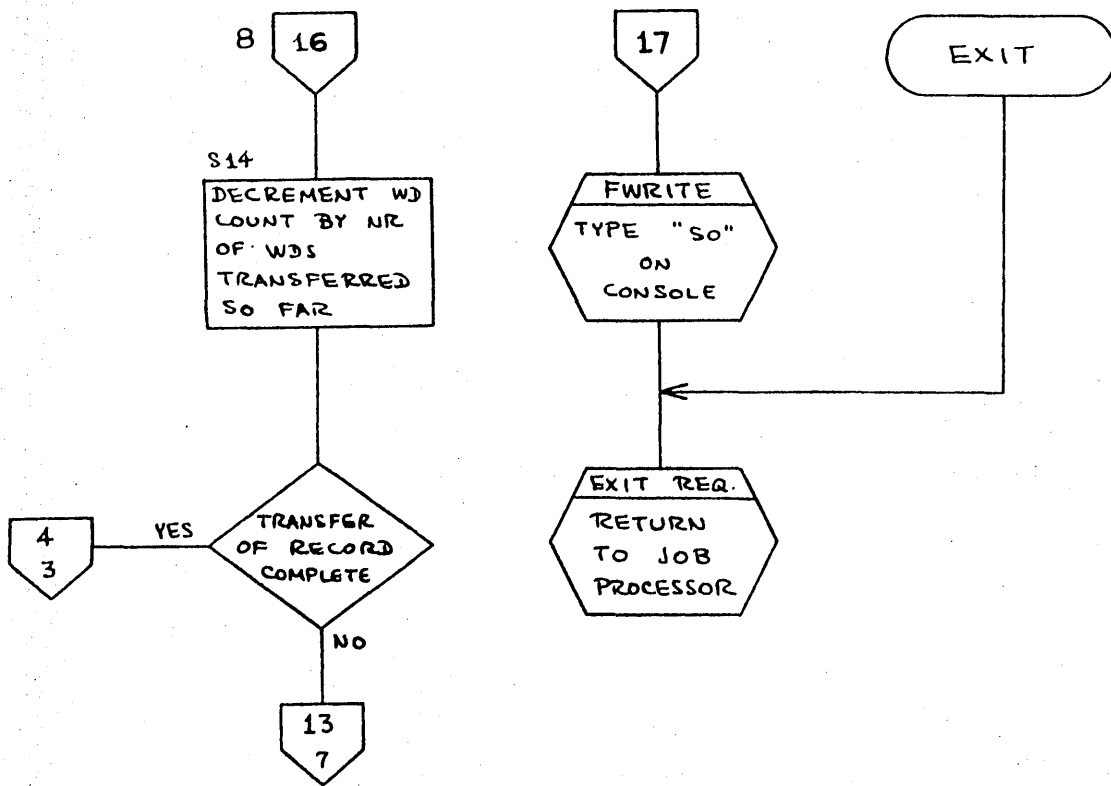




CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS \_\_\_\_\_ IMS \_\_\_\_\_ PAGE NO. 2-164  
PRODUCT NAME L700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005 V.2.0 \_\_\_\_\_ MACHINE SERIES 1700



9 RESET

RS2  
SET CURRENT  
SECTOR # OF  
SPECIFIED FILE  
TO START OF  
THAT FILE

THIS ENTRY IN  
PHASE D/E ONLY.  
CALLED BY  
'NPUNCH' TO  
BUILD LOAD-&-GO.

FILE  
TO BE USED  
NEXT FOR  
OUTPUT

NO

RESET1

LOADR1  
BEGIN LOAD-  
ING SCRATCH  
INPUT  
BUFFER

THIS BRANCH  
ALWAYS TAKEN  
IN PRESENT  
COMPILERS.

YES

CLEAR  
SCRATCH  
OUTPUT  
BUFFER  
INDEX

STEP SCRATCH  
INPUT FILE  
SECTOR NR  
TO NEXT PAGE  
OF FILE

4  
3

RESETP

CLEAR  
SCRATCH  
INPUT  
BUFFER  
INDEX

INPUT  
BUFFER  
AVAILABLE

NO

YES

4  
3

DOCUMENT CLASS IMS PAGE NO. 2-166  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. CO05\*3.1 A/B MACHINE SERIES 1700

## 2.4 STATEMENT SPECIFIC PHASEA ROUTINES

### 2.4.1 DIMENSION Statements - Subroutine DIMPR

Subroutine DIMPR processes dimension information. It will be called directly by PHASE A to process DIMENSION statements or as an auxiliary routine when processing COMMON or Type statements.

Input to DIMPR is the ISORS image of the statement being processed and the current setting of ISORSX within this buffer.

DIMPR builds entries in the symbol table {SYMTAB}, defining the arrays it processes. Entries are set to dimensioned arrays {ICLASS = 1, IDIM = no. dimensions, ISTABX = index into ISTAB to get dimensions}. The dimensions are entered in the ISTAB table. If a COMMON statement is being processed, ICOM {common block indicator} and ICOMTX {thread of variables and arrays in common block} are set in the symbol table and ICOMBX in the common table {ICOMT} is updated to reflect the current thread of the chain of variables and arrays in a specific common block.

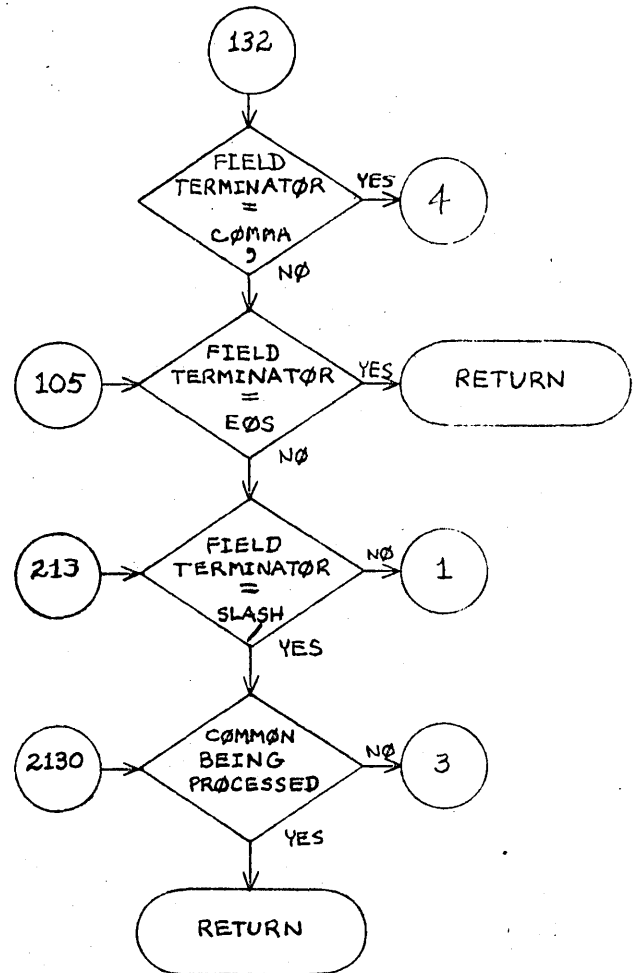
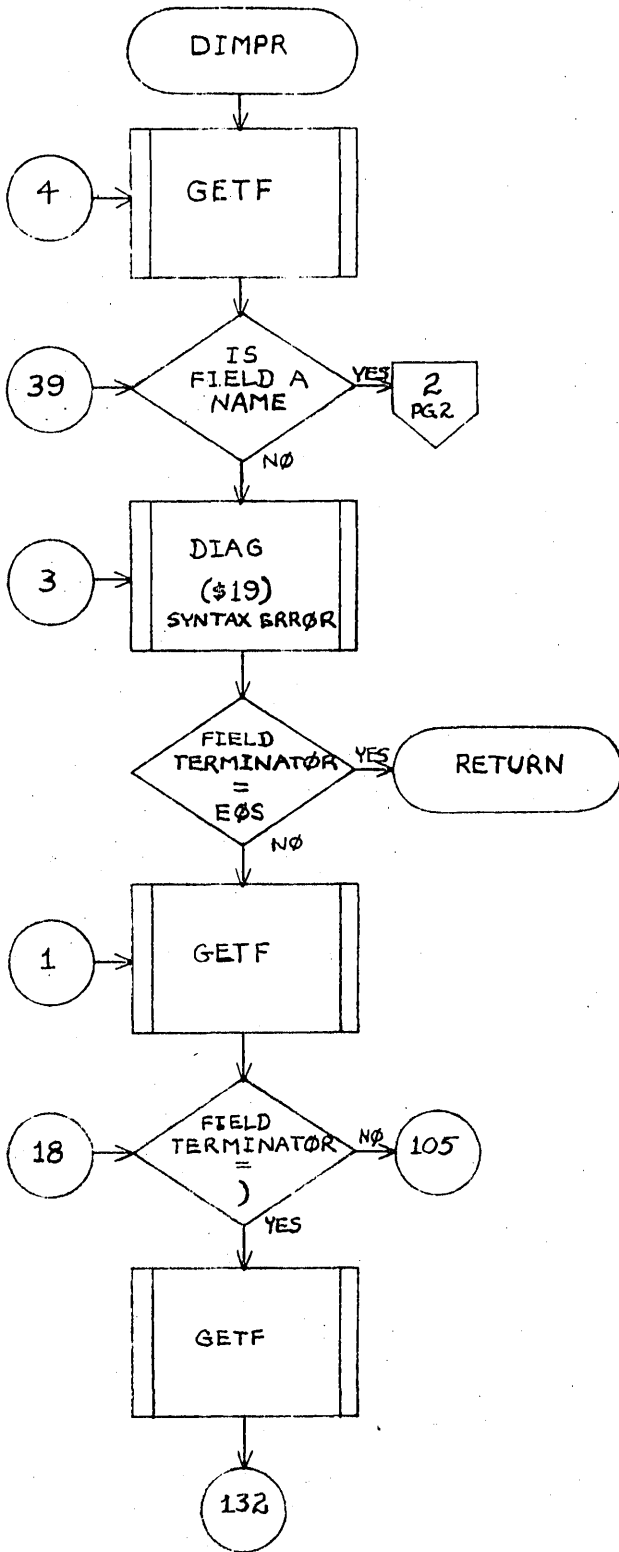
If a Type statement is being processed, the symbol table entry is set to the appropriate type {ITYPE = 1 for INTEGER or SINGLE, 2 for REAL, 3 for DOUBLE, ISNGL = 1 for SINGLE}.

Throughout its processing, DIMPR checks the syntax of the statement it is analyzing and outputs diagnostics if errors are encountered. It also checks for conflicting information, and outputs diagnostics if such information is encountered.

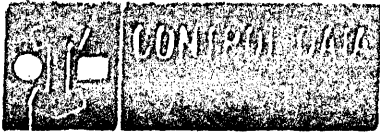




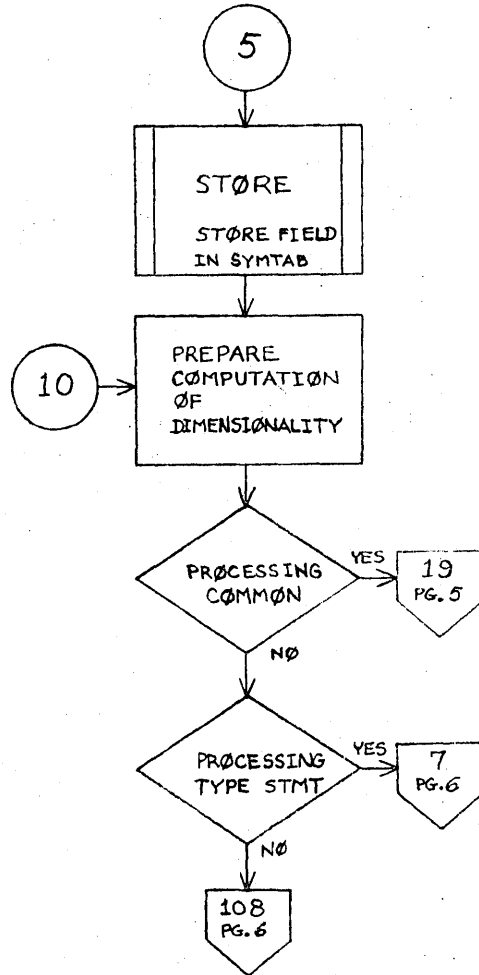
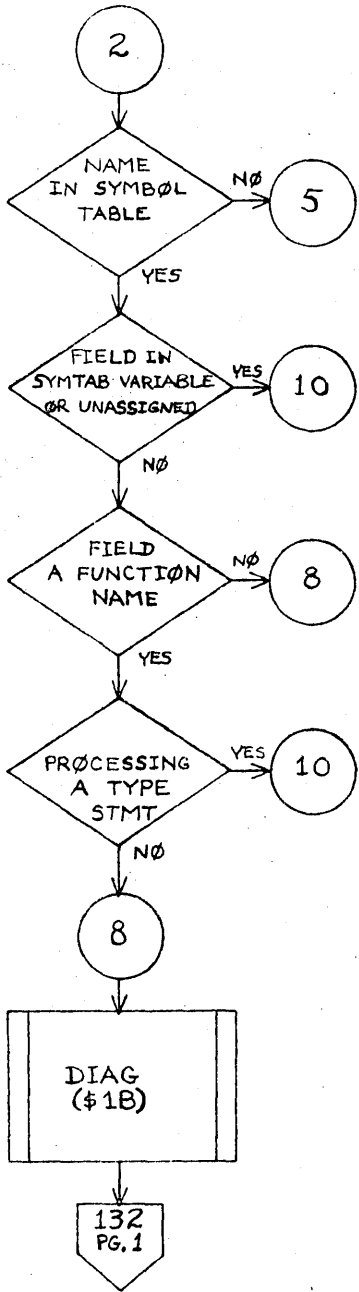
LA JOLLA FACILITY



TITLE		DIMPR		DRG. NO.	
				REVISION	
DRAWN BY	PROJ.	DATE	SHEET	1	OF 7
CSD 203					



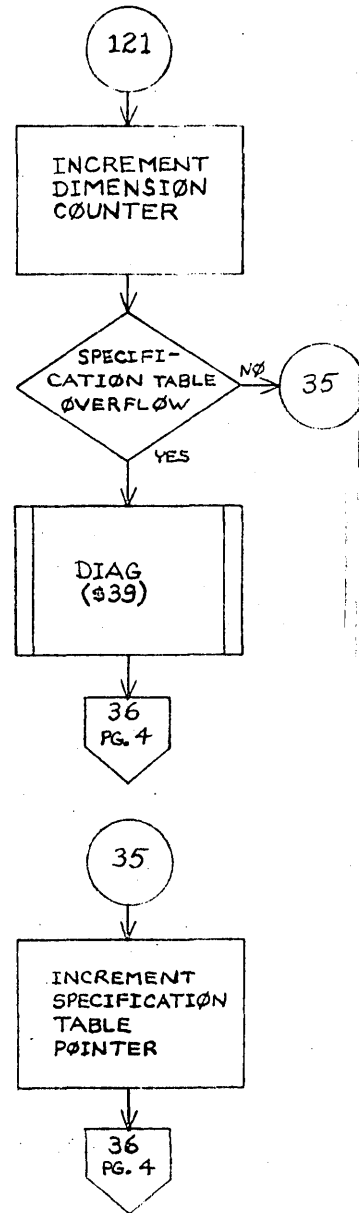
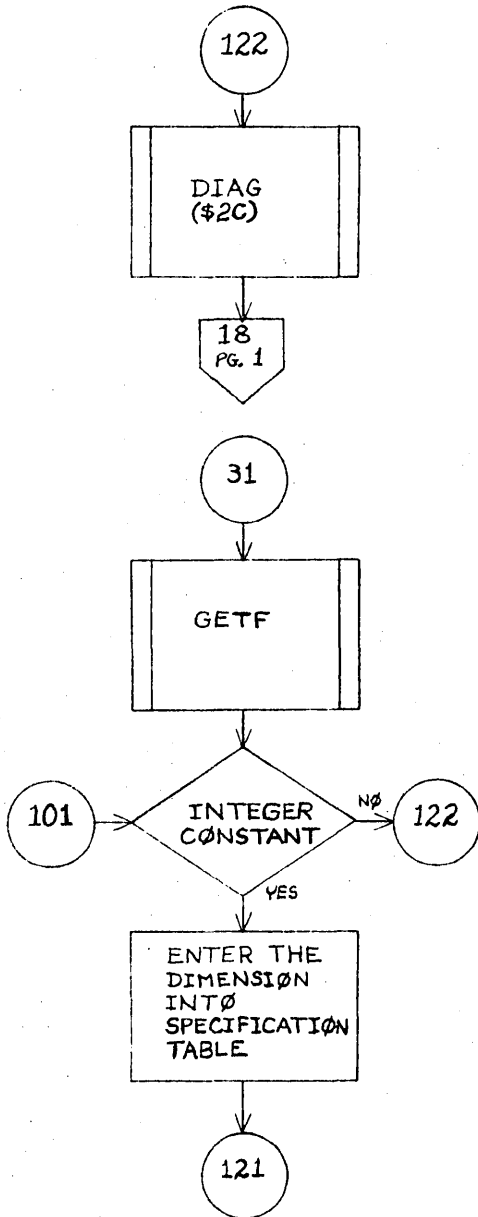
LA JOLLA FACILITY



TITLE DIMPR			DRG. NO.	
			REVISION	
DRAWN BY	PROJ.	DATE	SHEET 2	OF 7



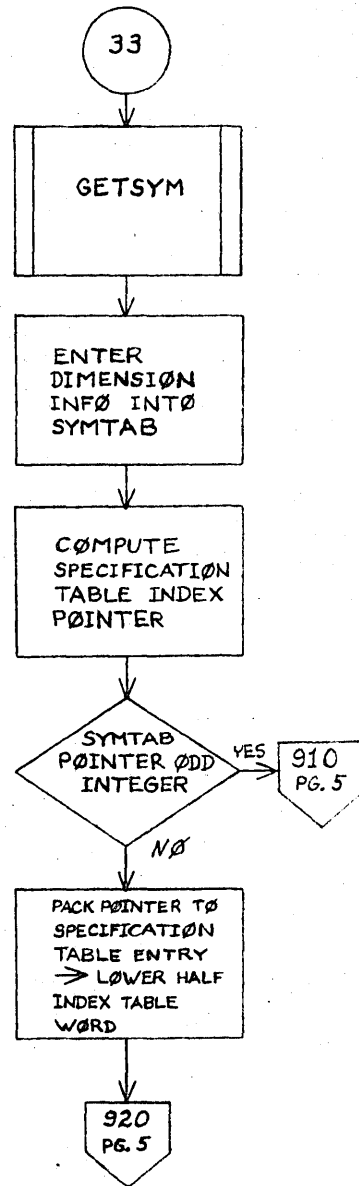
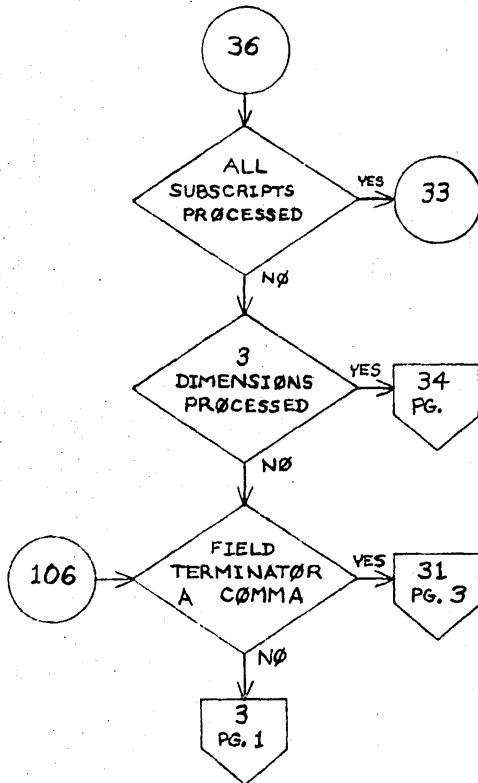
LA JOLLA FACILITY



TITLE		DIMPR		DRG. NO.	
DRAWN BY		PROJ.		REVISION	
CSD 203		DATE		SHEET 3 OF 7	



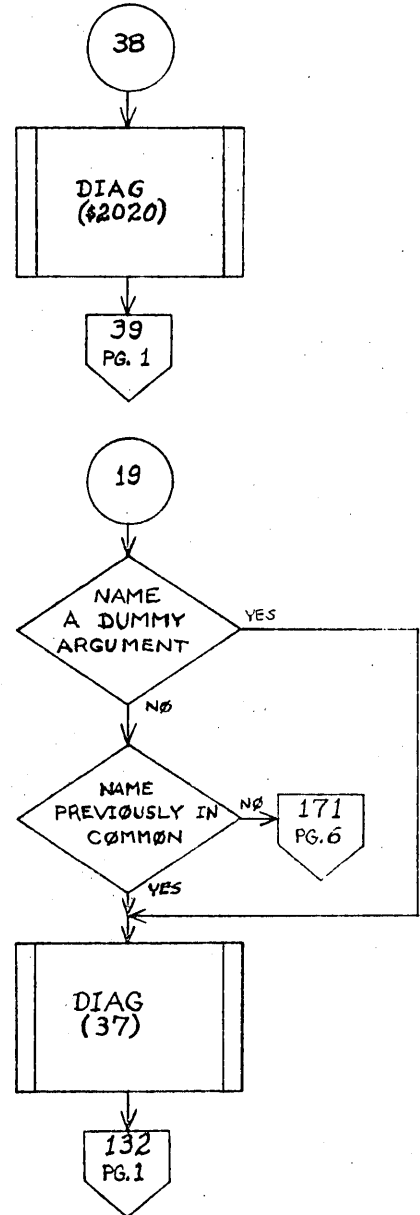
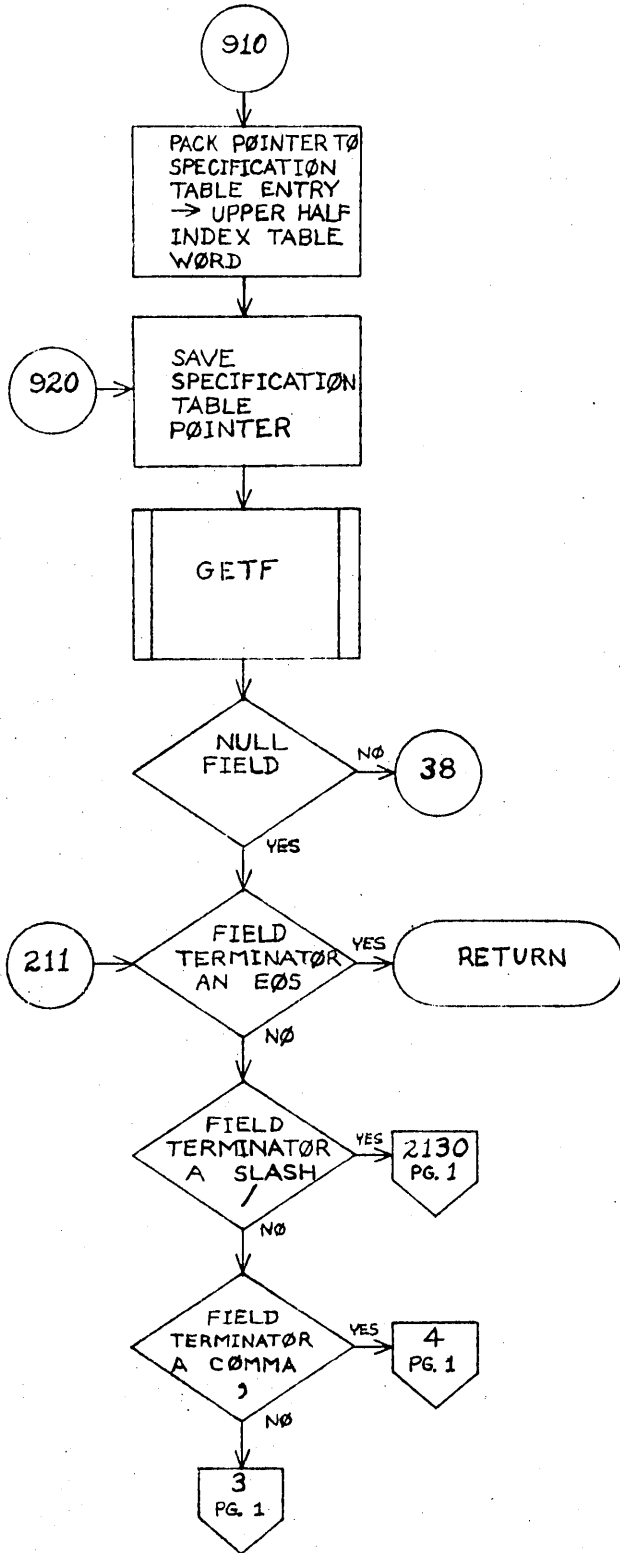
LA JOLLA FACILITY



TITLE DIMPR			DRG. NO.	
			REVISION	
DRAWN BY	PROJ.	DATE	SHEET	4 OF 7



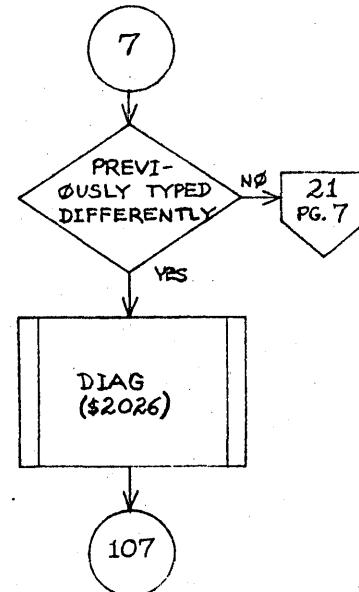
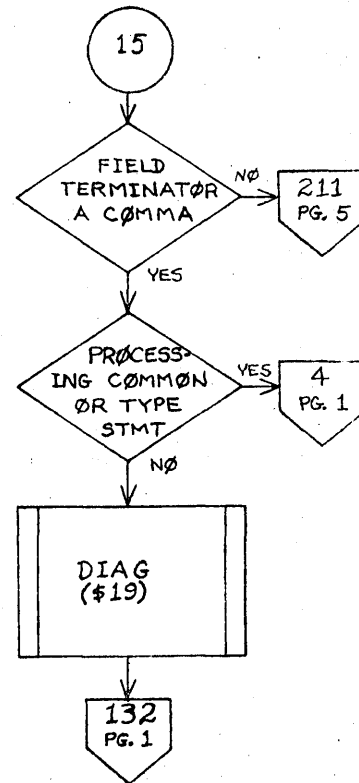
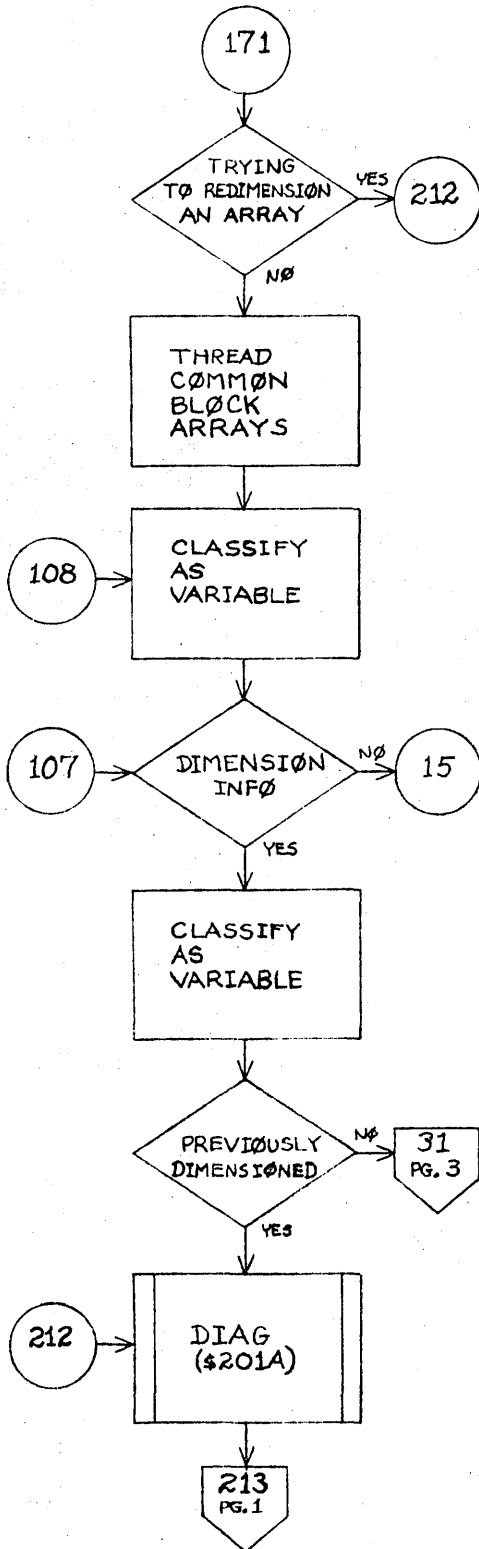
LA JOLLA FACILITY



TITLE			DIMPR		DRG. NO.	
DRAWN BY			PROJ.	DATE	REVISION	
CSD 203					SHEET 5 OF 7	



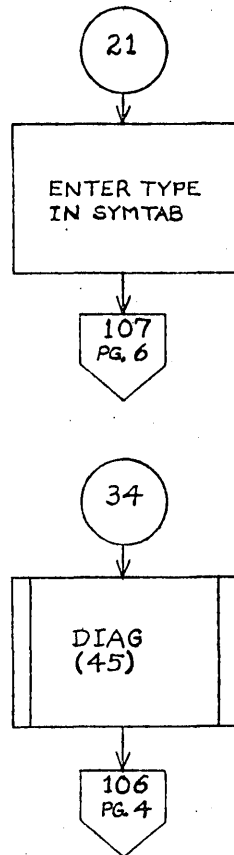
LA JOLLA FACILITY



TITLE		DIMPR		DRG. NO.	
DRAWN BY		PROJ.	DATE	REVISION	
CSD 203				SHEET	6 OF 7



LA JOLLA FACILITY



TITLE		DIMPR		DRG. NO.	
DRAWN BY		PROJ.		REVISION	
CSD 203		DATE		SHEET 7 OF 7	

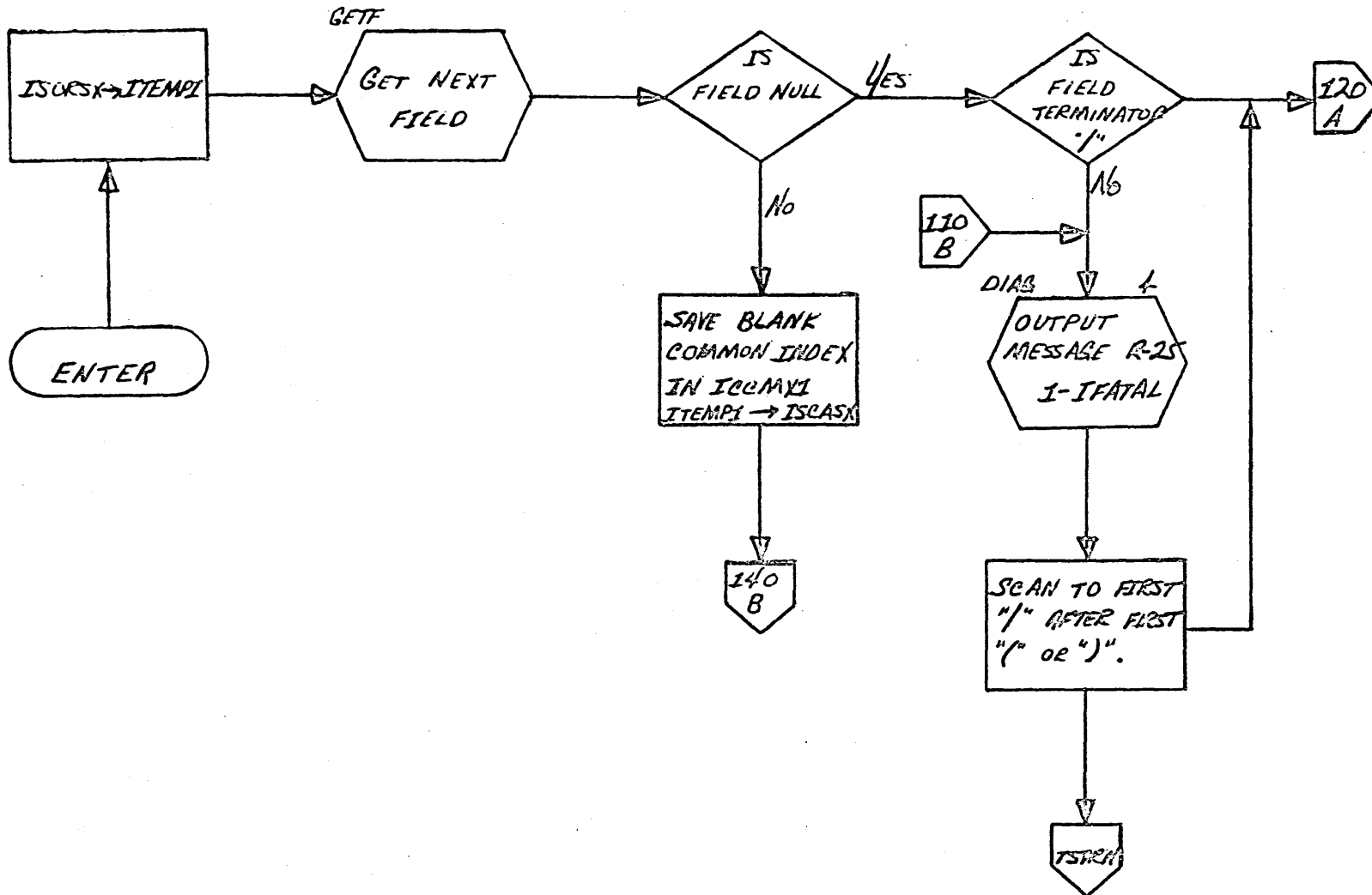
DOCUMENT CLASS IMS PAGE NO 2-174  
PRODUCT NAME 1700, MASS STORAGE FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

2.4.2 COMMON Statements - Subroutine COMNPR

COMMON Statements are processed by the routine COMNPR. The common block is entered in ICOMT, if it is not already there. DIMPR is called. DIMPR returns to COMNPR to search for another common block.



COMMON PROCESSOR

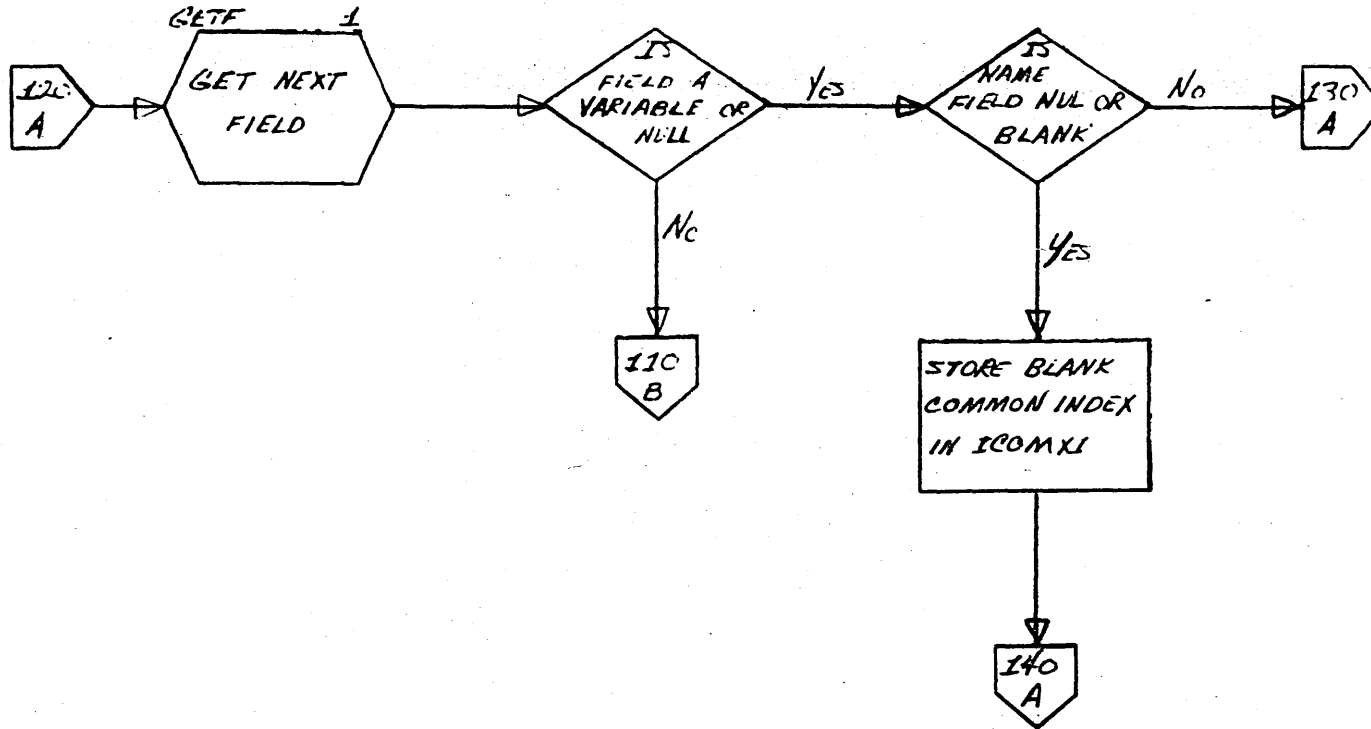


CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE   
FLOWCHART   
DECISION TABLE   
OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.		APPROVED	DATE
DOCUMENT TITLE	COMPR			PROJECT MGR.			
		PAGE	1 OF 4	PROJECT NAME			
NUMBER		ISSUE DATE		TASK NO.			
DRAWN BY		DATE	5/7	TASK NAME			

COMMON PROCESSOR

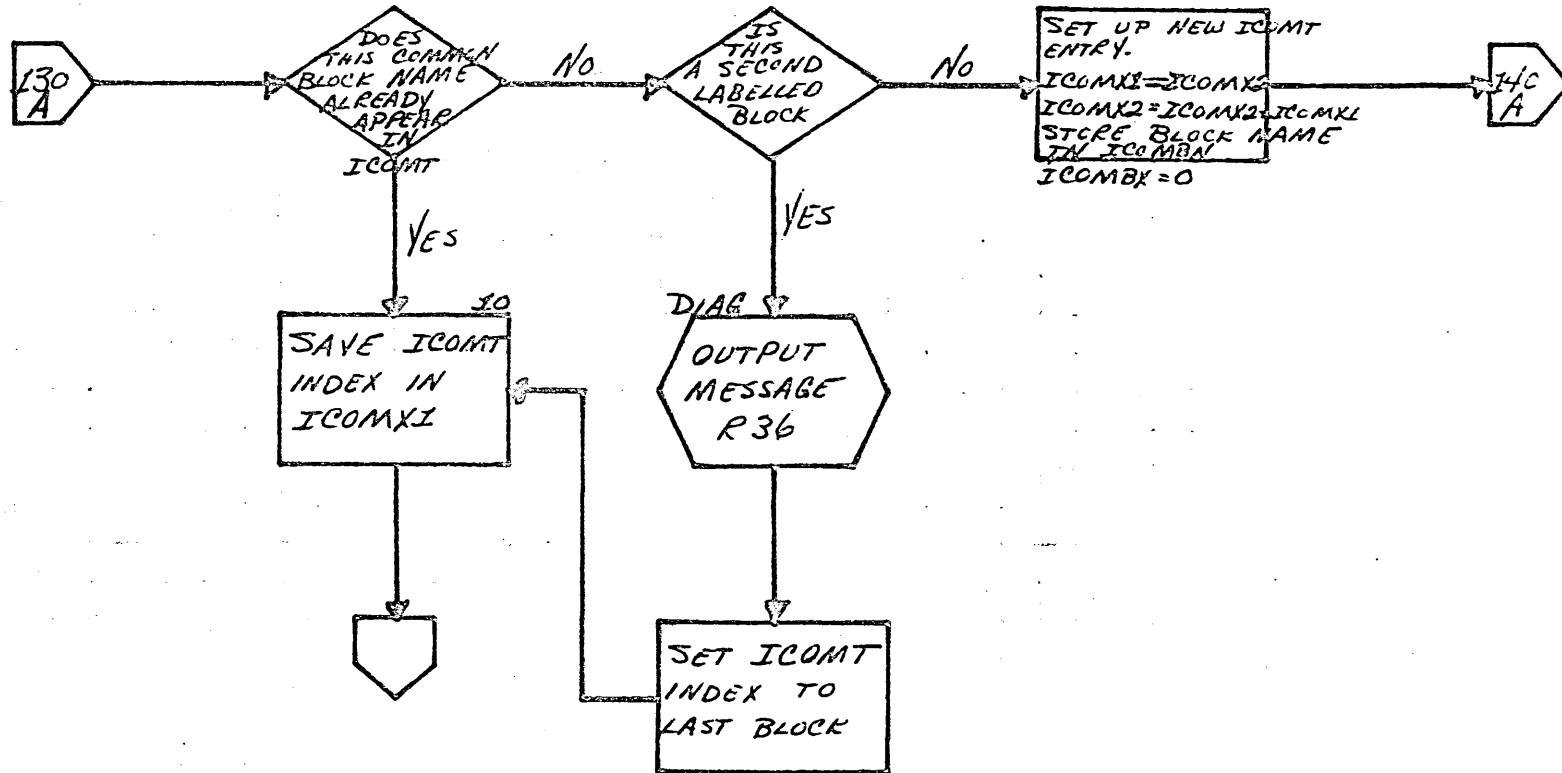


CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	<i>IMS</i>	MACH. TYPE	<i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	<i>COMMONPR</i>	PAGE <i>2</i> OF <i>4</i>		PROJECT MGR.			
NUMBER	ISSUE DATE			PROJECT NAME			
DRAWN BY	DATE	<i>5-6</i>		TASK NO.			
				TASK NAME			

COMMON PROCESSOR



CONTROL DATA CORPORATION

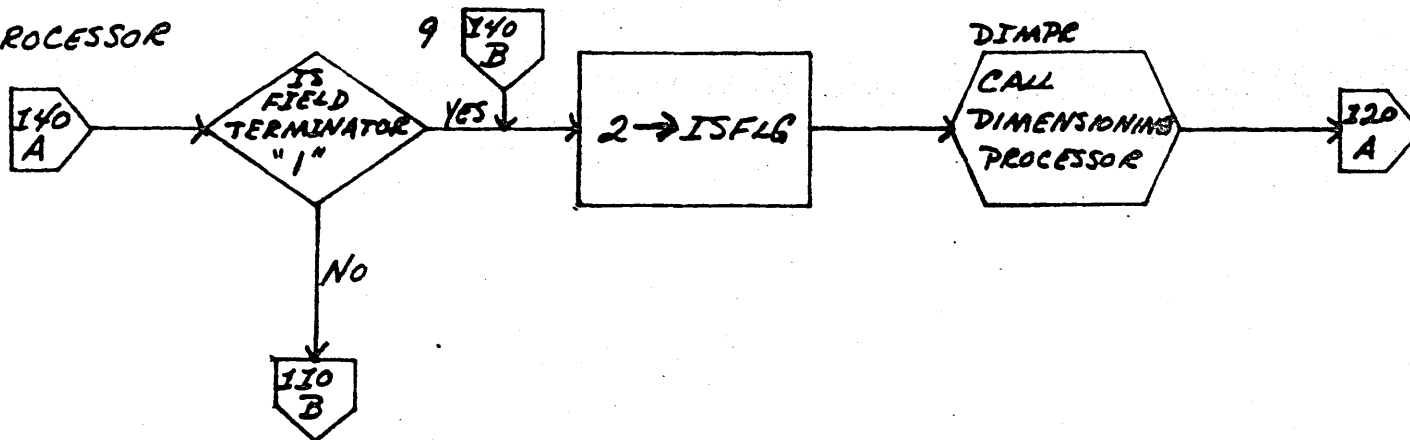
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.		REV	APPROVED	DATE
DOCUMENT TITLE	COMMONPR	PAGE 3 OF 4		PROJECT MGR.				
NUMBER		ISSUE DATE		PROJECT NAME				
DRAWN BY		DATE		TASK NO.				
				TASK NAME				

2-177

COMMON PROCESSOR



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

DOCUMENT CLASS	<i>IMS</i>	MACH. TYPE	<i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	<i>COMMNR</i>	PAGE <i>4</i> OF <i>4</i>		PROJECT MGR.			
NUMBER		ISSUE DATE		PROJECT NAME			
DRAWN BY		DATE		TASK NO.			
				TASK NAME			

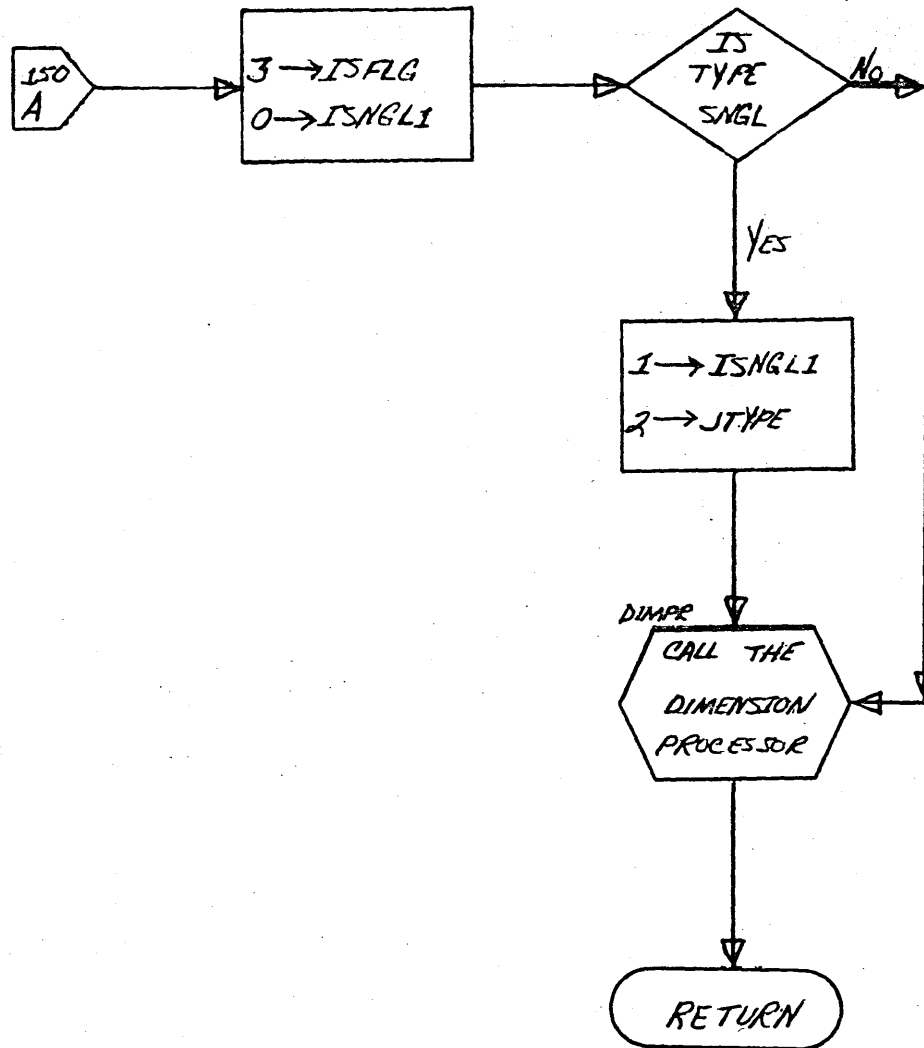
DOCUMENT CLASS TMS PAGE NO. 2-179  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

2.4.3

TYPE Statements - Subroutine TYPEPR

The routine TYPEPR is entered. This routine, in turn, calls DIMPR. DIMPR enters the symbols, with their appropriate type, into the symbol table.

TYPE PROCESSOR



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS <i>INS</i>	MACH. TYPE <i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE <i>TYPE PR</i>		PROJECT MGR.			
		PAGE <i>1</i> OF <i>1</i>	PROJECT NAME			
	NUMBER	ISSUE DATE	TASK NO.			
	DRAWN BY	DATE <i>3-6-71</i>	TASK NAME			

091-2

DOCUMENT CLASS IMS PAGE NO. 2-181  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT NO. Q015 VERSION 2.0 MACHINE SERIES 1700

#### 2.4.4 BYTE, EQUIVALENCE Statements - Subroutine BYEQPR

The BYEQPR subroutine is used to process BYTE, SIGNED BYTE and EQUIVALENCE statements. Upon entry to BYEQPR, the type number for the statement being processed is contained in IBUF2(3) as follows:

(IBUF2(3)) = 8 for BYTE statement.  
 (IBUF2(3)) = 9 for SIGNED BYTE statement.  
 (IBUF2(3)) = 10 for EQUIVALENCE statement.

##### 2.4.4.1 KEQV Table

BYEQPR generates entries for the KEQV table as a result of processing either of the three statements previously mentioned. Prior to entering BYEQPR, KEQVX contains the word length for the information in KEQV, and KEQVS contains the storage capacity for KEQV. The KEQV table along with KEQVX are initially cleared to zero as part of the initialization procedure of the PHASE 1 subroutine.

LEQVX is set to the value in KEQVX immediately upon entry to BYEQPR. Entries are made to KEQV using the storage index LEQVX. There are four words per entry to KEQV. For each entry made to KEQV, LEQVX is increased by the number of words per entry --

$(LEQVX) + 4 \rightarrow LEQVX.$

Entries made to the KEQV table have the following format:

WORD1: pointer to symbol table entry containing name of variable  
 WORD2: 1st subscript or 0 if subscript not specified  
 WORD3: 2nd subscript or 0 if subscript not specified  
 WORD4: 3rd subscript or 0 if subscript not specified

Consecutive entries made to KEQV correspond to related elements in the source statement. (Refer to item 2.4.4.2 for an EQUIVALENCE statement and to item 2.4.4.3 for a BYTE statement.) The last of these entries is followed by an end-of-chain mark which is a word containing a value of -1. When an entry is made to KEQV to indicate the end of chain KEQVX is increased by the number of words added to the KEQV table by -

$(LEQVX) + 1 \rightarrow KEQVX.$

When processing begins and the next chain of related elements in the source statement, LEQVX will be set to the current word length of KEQV information by -

$(KEQVX) \rightarrow LEQVX.$

DOCUMENT CLASS IMS PAGE NO. 2-182  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT NO. 0005 VERSION 2.0 MACHINE SERIES 1700

Upon exit from BYEQPR, KEQVX will contain the word length of information stored in the KEQV table.

#### 2.4.4.2 Processing the EQUIVALENCE Statement

Equivalence statements of the form:

$$\text{EQUIVALENCE } (a_1, a_2, \dots, a_i), (b_1, b_2, \dots, b_i), \dots, (c_1, c_2, \dots, c_i)$$

are processed as follows:

A chain of related elements are enclosed in parentheses. The chains separated by commas. The last chain is followed by an EOS. Within the chain, the elements are separated by commas. A four word entry is made to KEQV in the manner described by 2.4.4.1 for each element in a chain. The last entry is followed by an end-of-chain mark. The 1st entry made for the next chain follows the end-of-chain mark for the preceding entry, etc. A chain must contain at least 2 elements.

#### 2.4.4.2.1 Processing Errors for EQUIVALENCE Statements

Diagnostics will be given for the following errors while processing EQUIVALENCE statements:

1. A chain is not enclosed in parenthesis.
2. An element is not alphanumeric.
3. An element within a chain is already in the symbol table, and either -
  - a. classed as something other than a variable name, or -
  - b. the name appears either on a RELATIVE or an EXTERNAL statement.
4. An element has more than three subscripts.
5. An element has a subscript  $\leq 0$ .
6. An element has a non-integer subscript.
7. The subscripts of an element are not separated by commas.
8. The subscripts are not enclosed in parentheses.
9. A chain contains fewer than two elements.

After the error diagnostic is printed, the entries made to the KEQV table are erased in the following manner:

Using the value in KEQVX to locate the starting point, each KEQV word is set to zero by -

0 → KEQV(KEQVX),  
 0 → KEQV(KEQVX+1),  
 ⋮  
 and 0 → KEQV(LEQVX-1).



DOCUMENT CLASS IMS PAGE NO. 2-183  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT NO. COO 5 VERSION 2.0 MACHINE SERIES 1700

Once these entries are erased, BYEQPR will slew the remaining portion of the chain. While slewing, an exit will be made from BYEQPR when it encounters in the source buffer, either an EOS character or a right parenthesis followed by an EOS. If, while slewing, BYEQPR encounters a right parenthesis followed by a comma, it will proceed to process the next field.

## 2.4.4.3

## Processing BYTE Statements

BYTE or SIGNED BYTE statements of the form:

$$\left. \begin{array}{l} \text{BYTE} \\ \text{SIGNED BYTE} \end{array} \right\} (a_1, b_1, (c_1=d_1)), \dots, (a_i, b_i, (c_i=d_i))$$

are processed as follows:

For each byte name processed, two entries are made into the KEQV table, each in the manner described by item 2.4.4.1. The 1st entry is made for the byte name  $a_i$ , followed by the entry for the second variable  $b_i$ . This entry is followed by an end-of-chain mark of "-1". The  $b_i$  variable may or may not be subscripted. However, the  $a_i$  variable must not be subscripted. Therefore, the 2nd, 3rd and 4th words of the KEQV entry for  $a_i$  must contain zeros.

If  $(c_i, b_i, (c_i=d_i))$

may be regarded as a chain, the chain of the BYTE or SIGNED BYTE statement must be separated by commas and the last followed by an EOS.

The byte limits  $c_i$  and  $d_i$  must both be integer constants and maintain the relationship -

$$15 \geq c_i \geq d_i \geq 0.$$

An indicator is set in the symbol table entry containing the name  $a_i$  to show that it is a byte name by either -

- a. 1  $\rightarrow$  IPART(ISYMX) if  $a_i$  is not a signed byte, or -
- b. 2  $\rightarrow$  IPART(ISYMX) if  $a_i$  is a signed byte.

The byte limits are recorded in the symbol table entry as follows:

1.  $c_i \rightarrow$  IPARTL(ISYMX), and -
2.  $d_i \rightarrow$  IPARTR(ISYMX).

DOCUMENT CLASS \_\_\_\_\_ IMS \_\_\_\_\_ PAGE NO. 2-184  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT NO. COG 5 VERSION 2.0 MACHINE SERIES 1700

#### 2.4.4.3.1 Processing Errors for BYTE and SIGNED BYTE Statements

Diagnostics will be given for the following errors while processing BYTE and SIGNED BYTE statements:

1. A chain (as defined in item 2.4.4.3) is not enclosed in parenthesis.
2. An element, either  $a_i$  or  $b_i$ , is not alphanumeric.
3. Some byte name  $a_i$  appears in the symbol table either -
  - a. typed as something other than an integer variable,
  - b. indicated to be a dummy variable, or -
  - c. previously specified as a byte.
4. The byte name  $a_i$  is subscripted.
5. The second variable  $b_i$  is not alphanumeric.
6. The second variable  $b_i$  has more than three subscripts.
7. The second variable  $b_i$  has a non-integer subscript.
8. The second variable  $b_i$  has a subscript 0.
9. The subscripts for  $b_i$  are not enclosed within parentheses.
10. Either of the byte limits  $c_i$  or  $d_i$  is not an integer constant.
11. The byte limits  $c_i$  and  $d_i$  are not enclosed in parentheses and/or the limits  $c_i$  and  $d_i$  are separated by something other than equal sign.
12. The byte limits  $c_i$  and  $d_i$  do not satisfy the condition -
$$15 > c_i > d_i \geq 0.$$

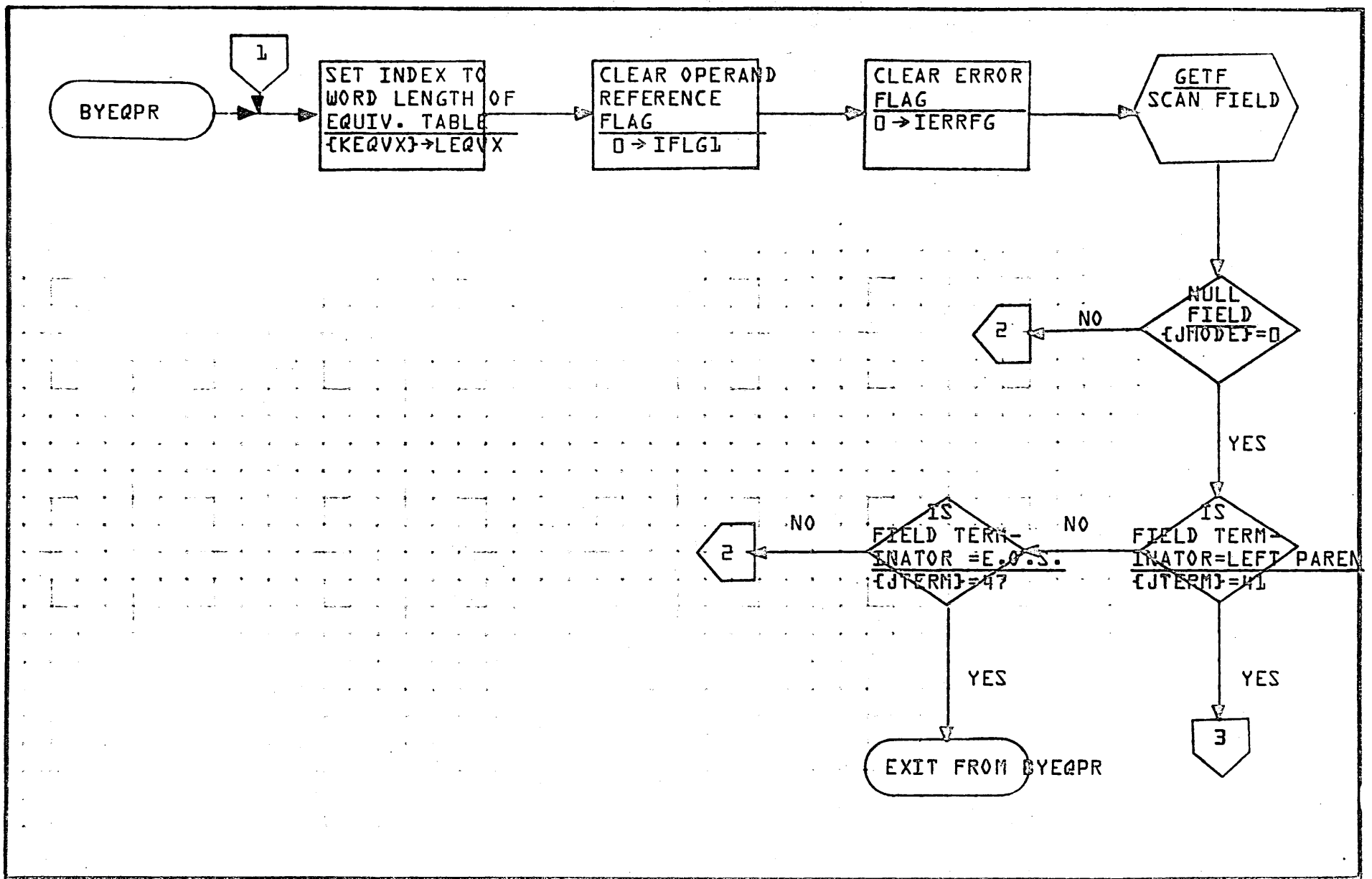
Following the diagnostic output, the procedure followed for errors 1-11 is the same as the procedure described in item 2.4.4.2.1 concerning errors in EQUIVALENCE statements. After printing the diagnostic, the procedure for error 12 is to substitute a 15 for  $c_i$  and a 0 for  $d_i$  and proceed as if no error had occurred.

A

B

C

D



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	BYEQPR	PAGE 1 OF 15		PROJECT MGR.			
	NUMBER	ISSUE DATE			PROJECT NAME			
	DRAWN BY	DATE			TASK NO.			
					TASK NAME			

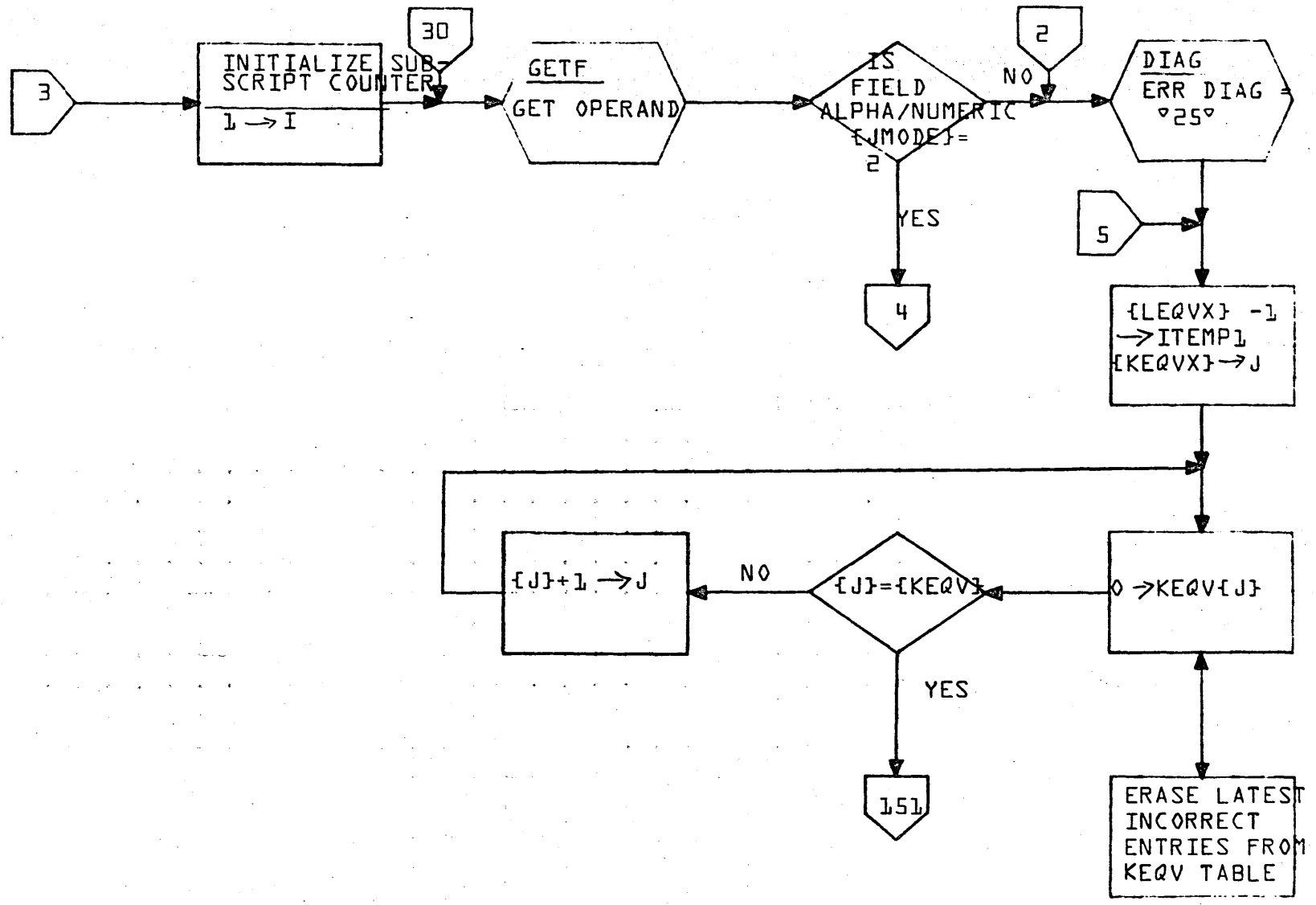
2-185

A

B

C

D



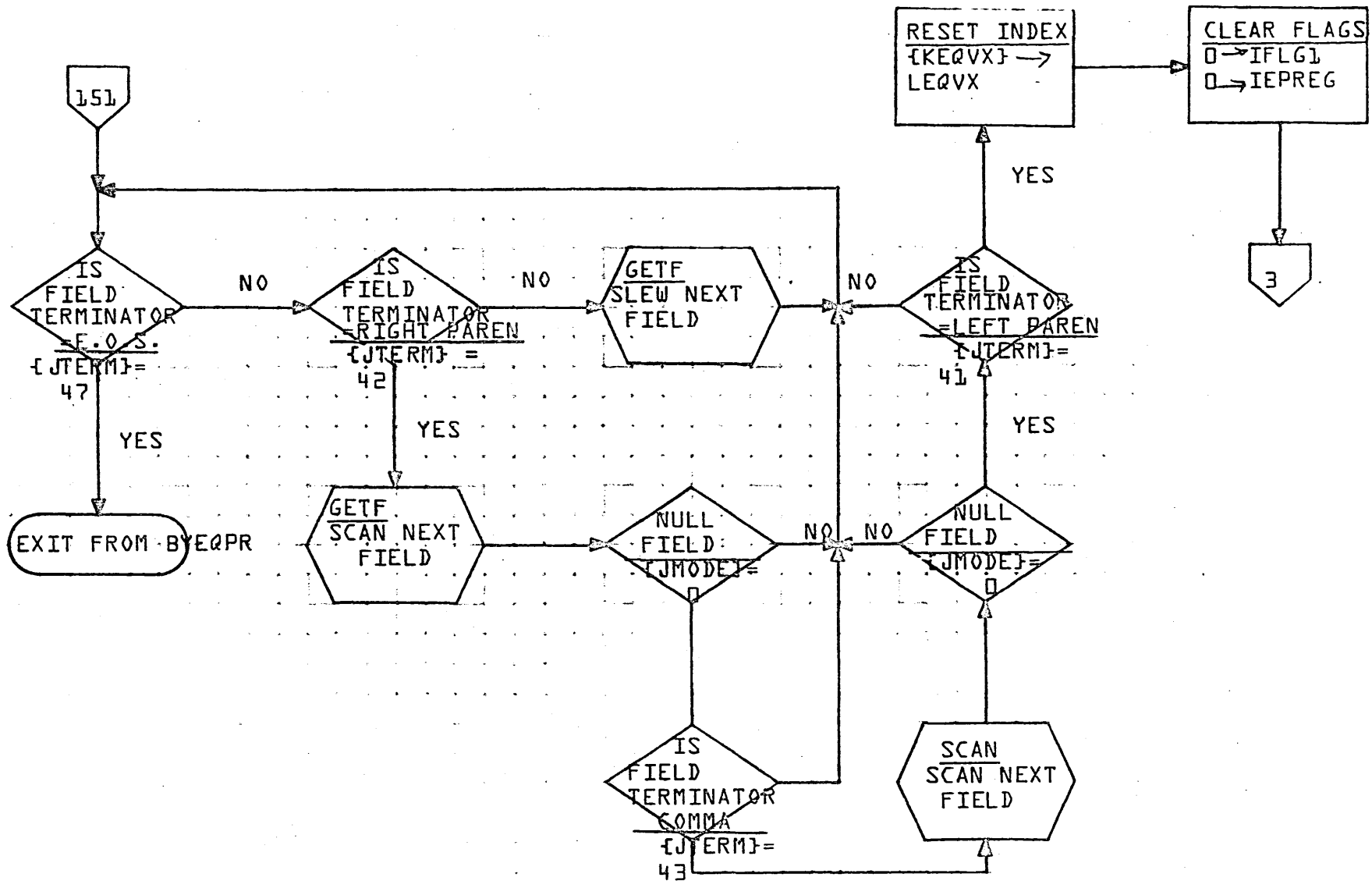
<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	BYEQPR	PAGE 2 OF 15		PROJECT MGR.			
	NUMBER	ISSUE DATE			PROJECT NAME			
	DRAWN BY		DATE		TASK NO.			
					TASK NAME			

A

B

C

D



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	BYEQPR	PAGE 3 OF 15		PROJECT MGR.			
	NUMBER	ISSUE DATE	DATE		PROJECT NAME			
	DRAWN BY	DATE		TASK NO.				
			DATE		TASK NAME			

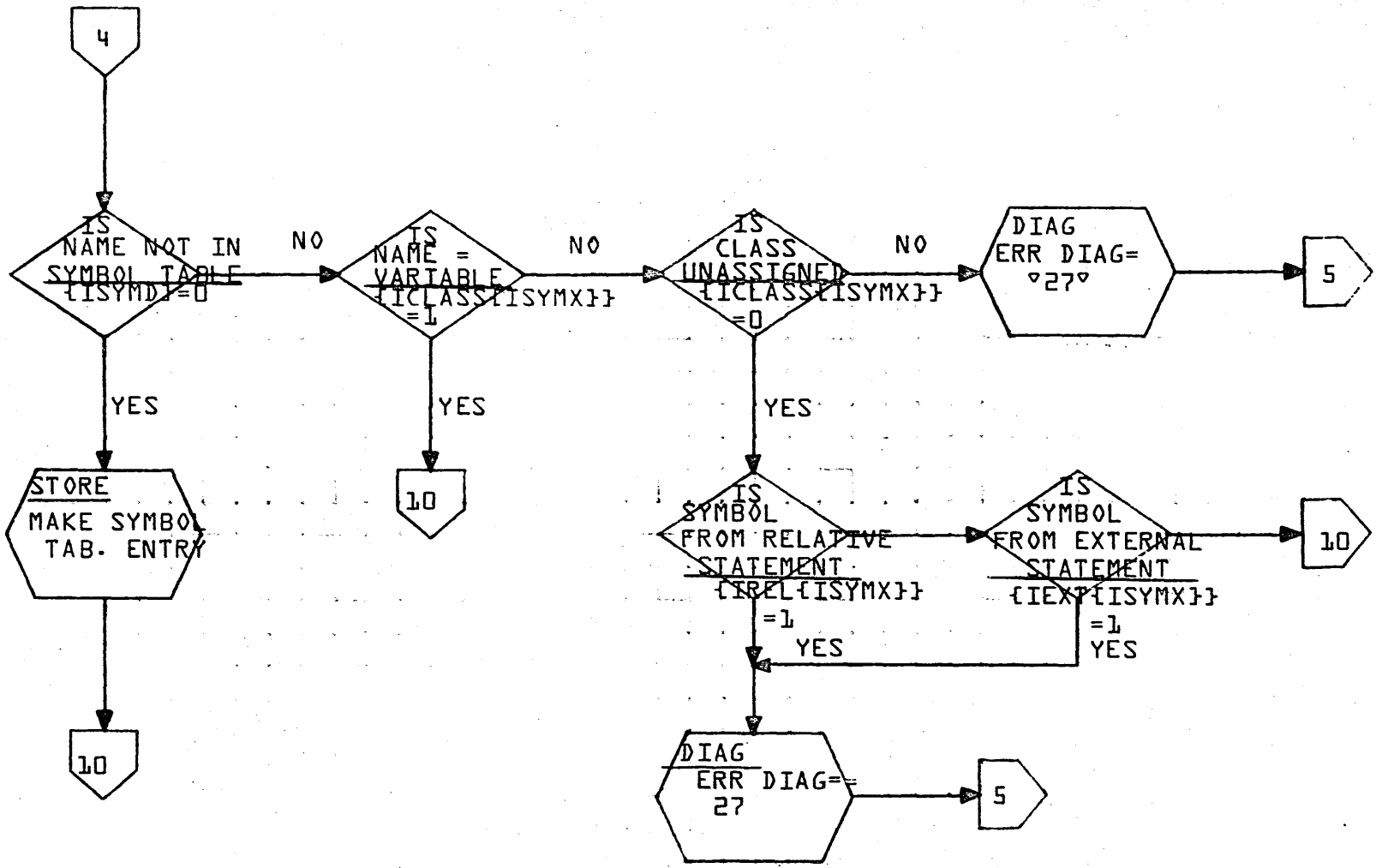
2-187

A

B

C

D

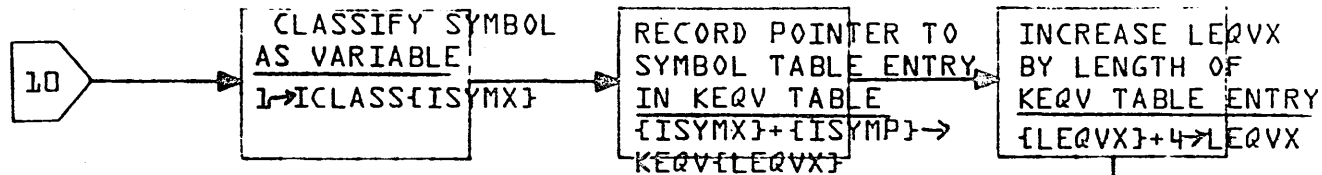


**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

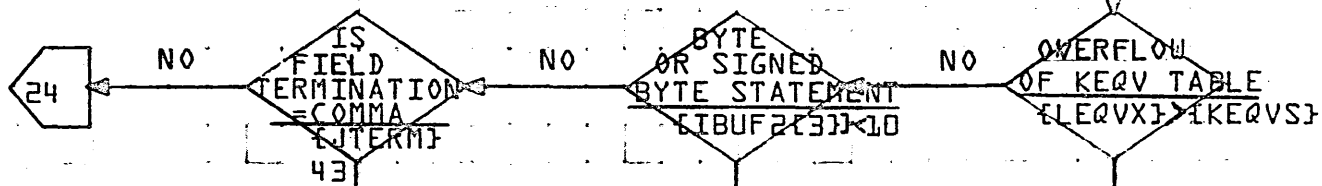
- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	BYEQPR	PAGE OF 15		PROJECT MGR.			
NUMBER	ISSUE DATE			PROJECT NAME			
DRAWN BY	DATE			TASK NO.			
				TASK NAME			

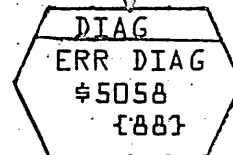
A



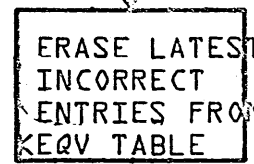
B



C



D



**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	BYEQPR	PAGE 5 OF 15		PROJECT MGR.			
NUMBER	ISSUE DATE			PROJECT NAME			
DRAWN BY	DATE			TASK NO.			
				TASK NAME			

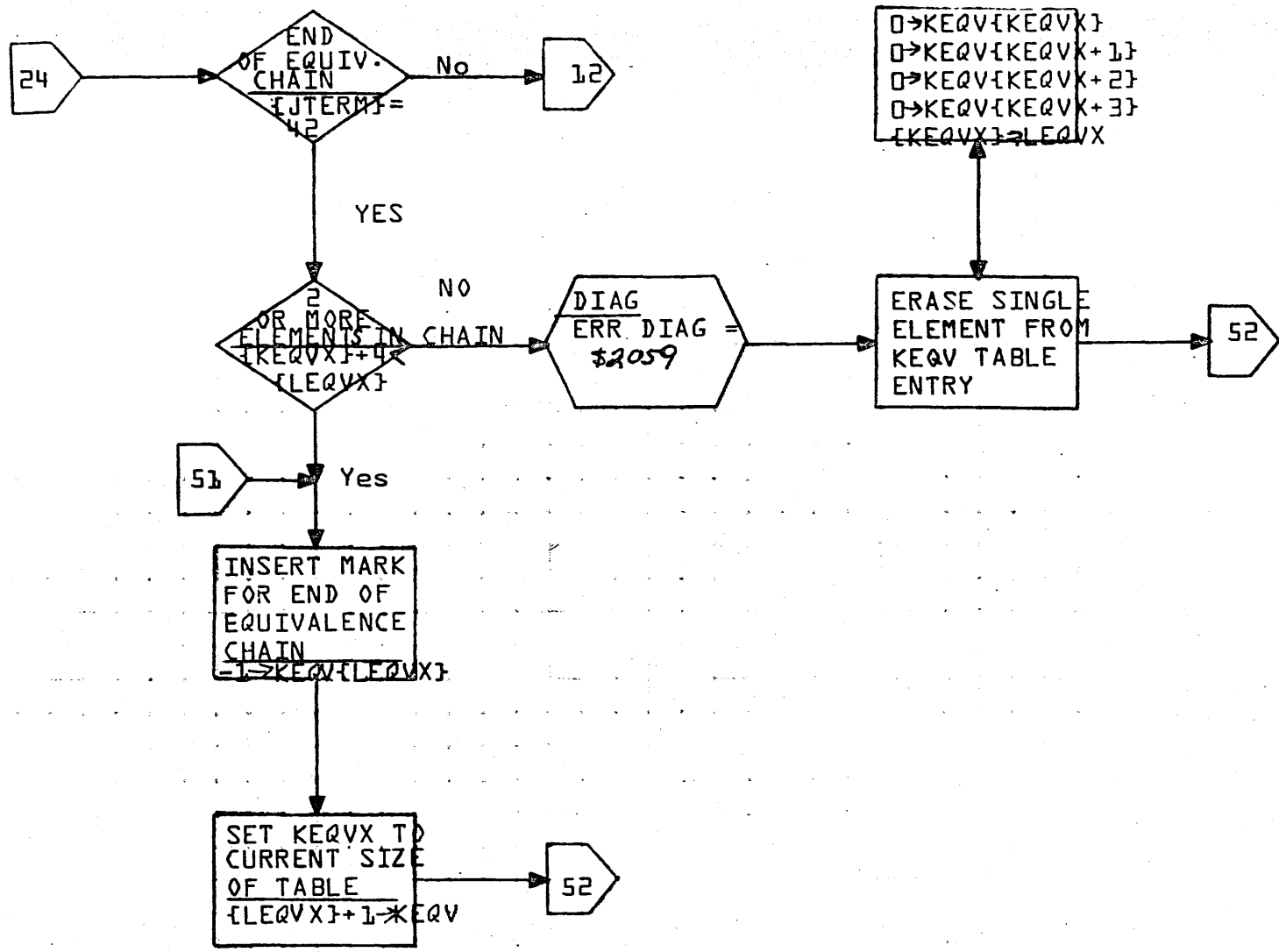
2-189

A

B

C

D



**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	BYEQPR	PAGE 5 OF 5		PROJECT MGR.			
NUMBER	ISSUE DATE	TASK NO.					
DRAWN BY	DATE	TASK NAME					

06T-2

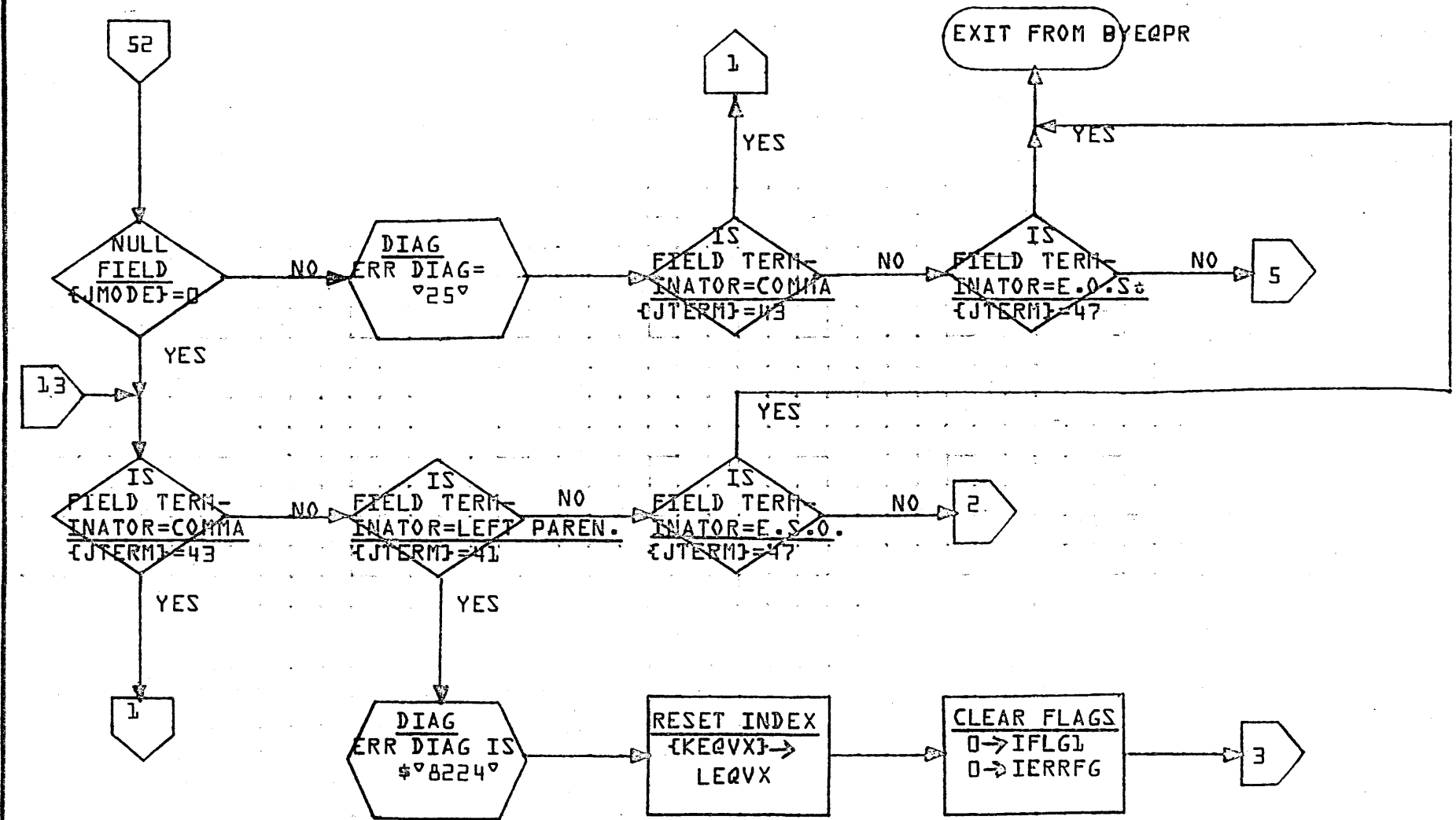


A

B

C

D



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	BYEQPR	PAGE 7 OF 15		PROJECT MGR.			
	NUMBER	ISSUE DATE	DATE		PROJECT NAME			
	DRAWN BY	DATE		TASK NO.	TASK NAME			

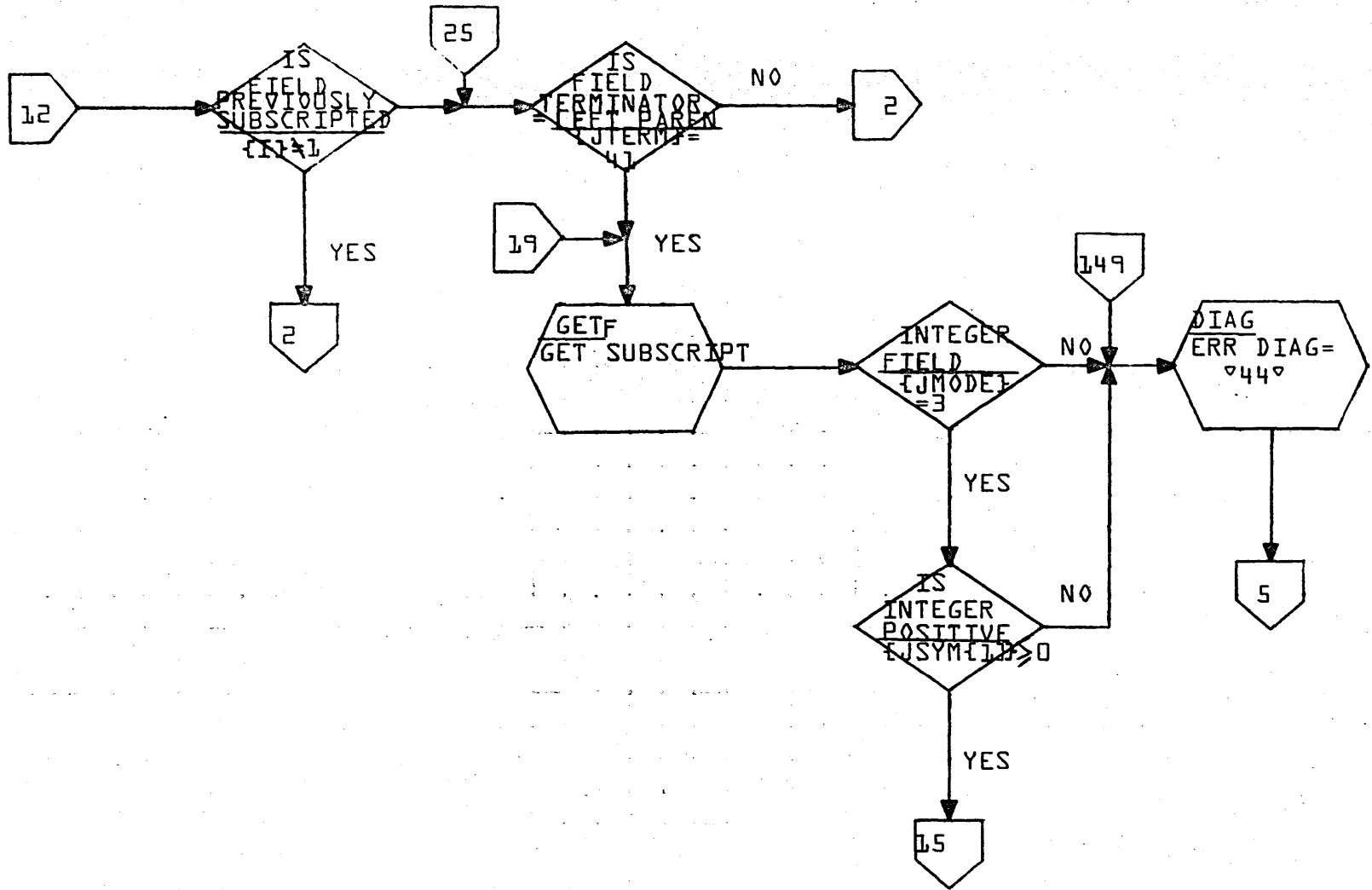
2-191

A

B

C

D



**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

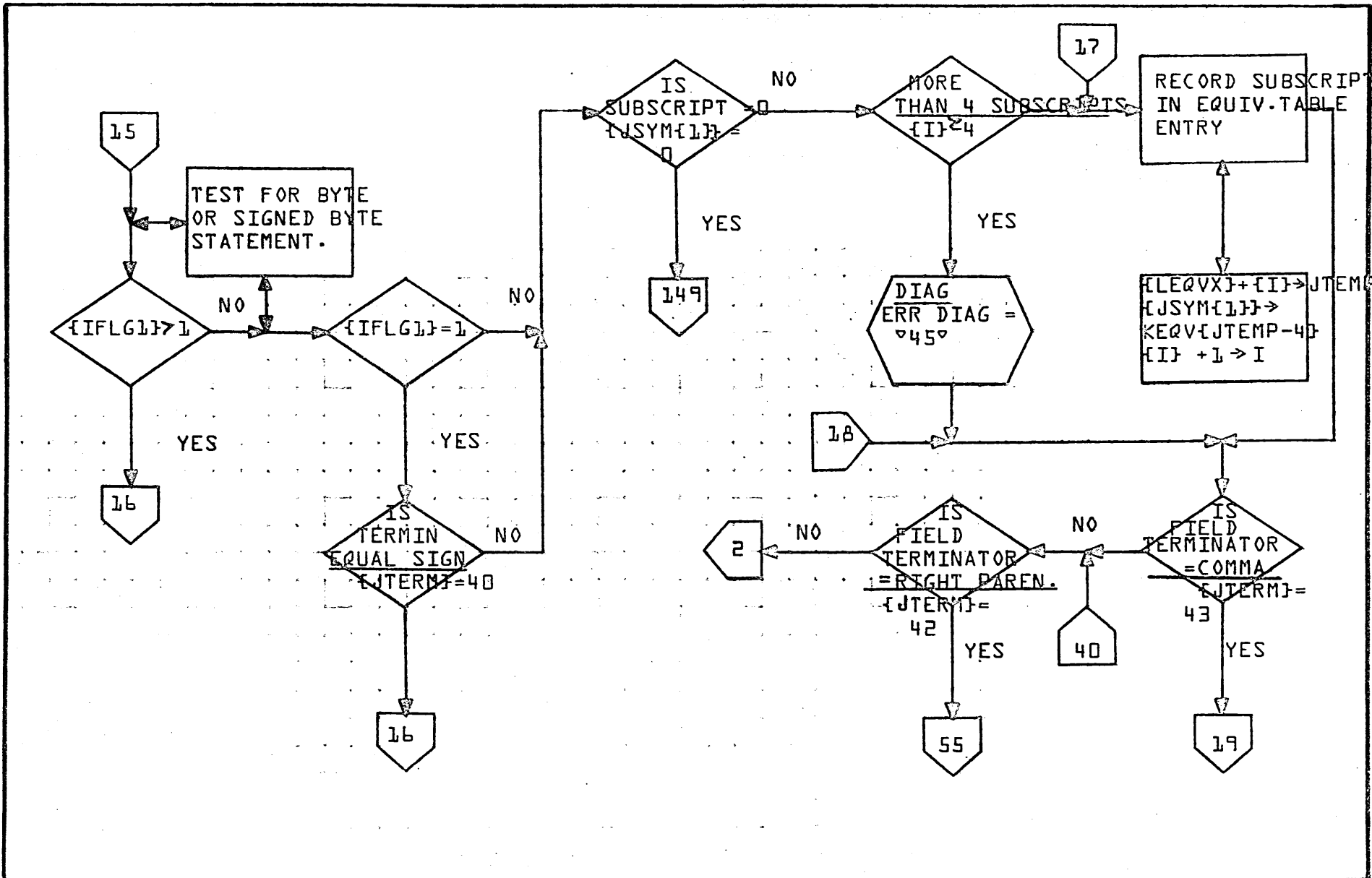
DOCUMENT CLASS	TMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	BYEQPR	PAGE 8 OF 15		PROJECT MGR.			
NUMBER	ISSUE DATE			PROJECT NAME			
DRAWN BY	DATE			TASK NO.			
				TASK NAME			

A

B

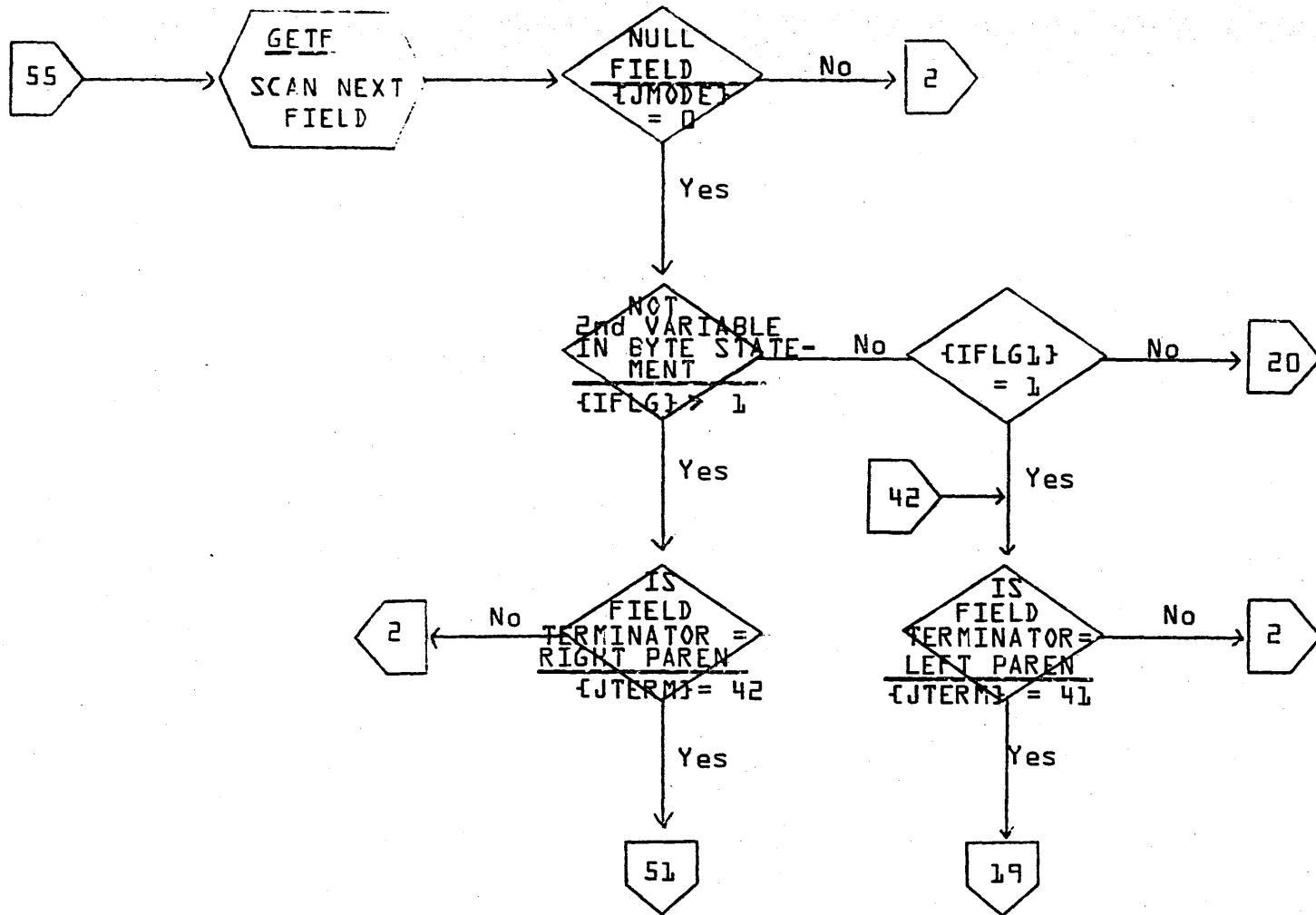
C

D



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	BYEQPR	PAGE 9 OF 15		PROJECT MGR.			
	NUMBER	ISSUE DATE			PROJECT NAME			
	DRAWN BY	DATE			TASK NO.			
					TASK NAME			

2-173



**CONTROL DATA CORPORATION**

SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

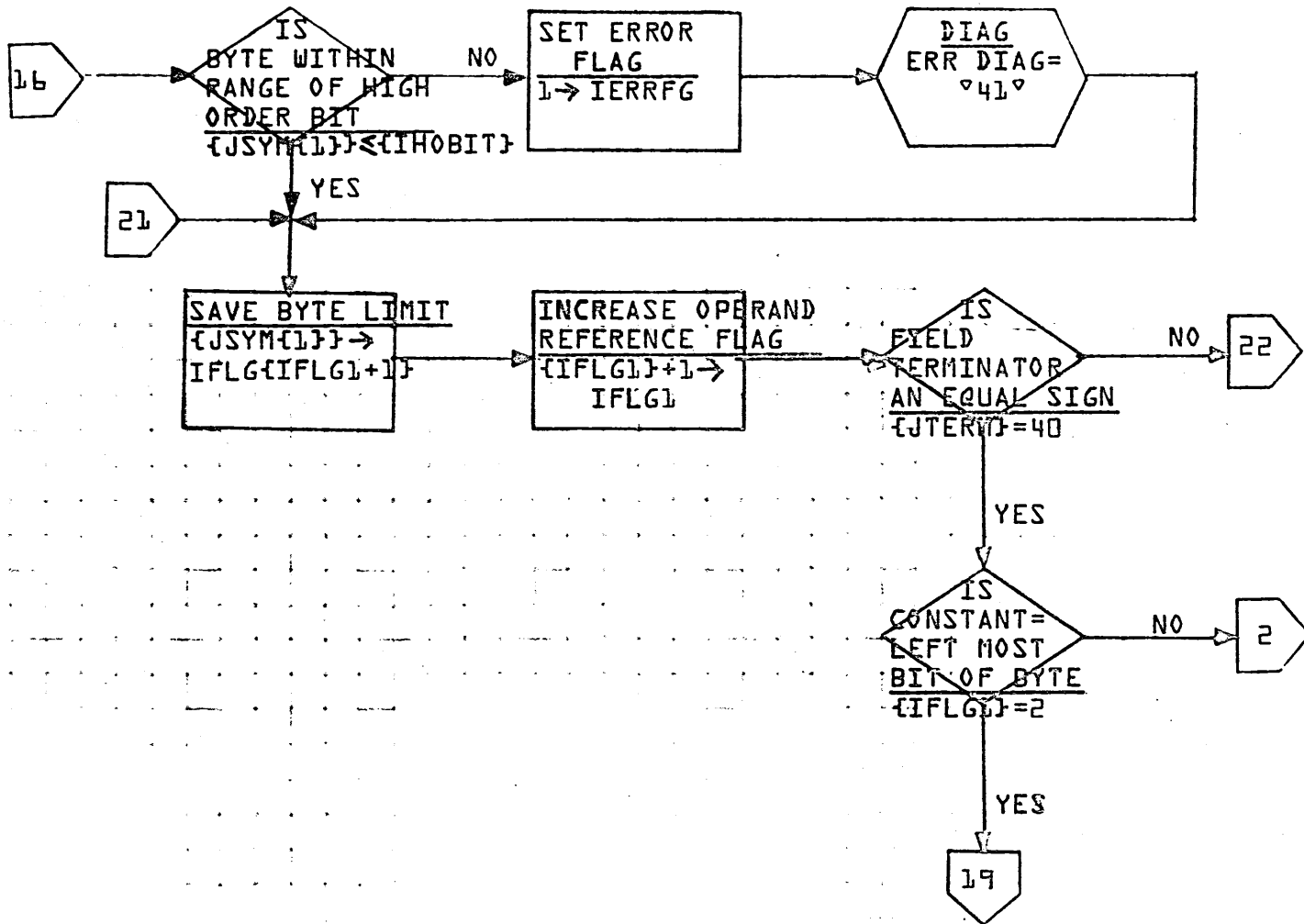
DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	BYEQPR	PAGE 1 OF 15		PROJECT MGR.			
NUMBER	ISSUE DATE	TASK NO.		PROJECT NAME			
DRAWN BY	DATE	TASK NAME					

A

B

C

D



**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.		REV		APPROVED		DATE	
DOCUMENT TITLE	BYEQPR			PROJECT MGR.							
			PAGE 1 OF 15	PROJECT NAME							
NUMBER		ISSUE DATE		TASK NO.							
DRAWN BY		DATE		TASK NAME							

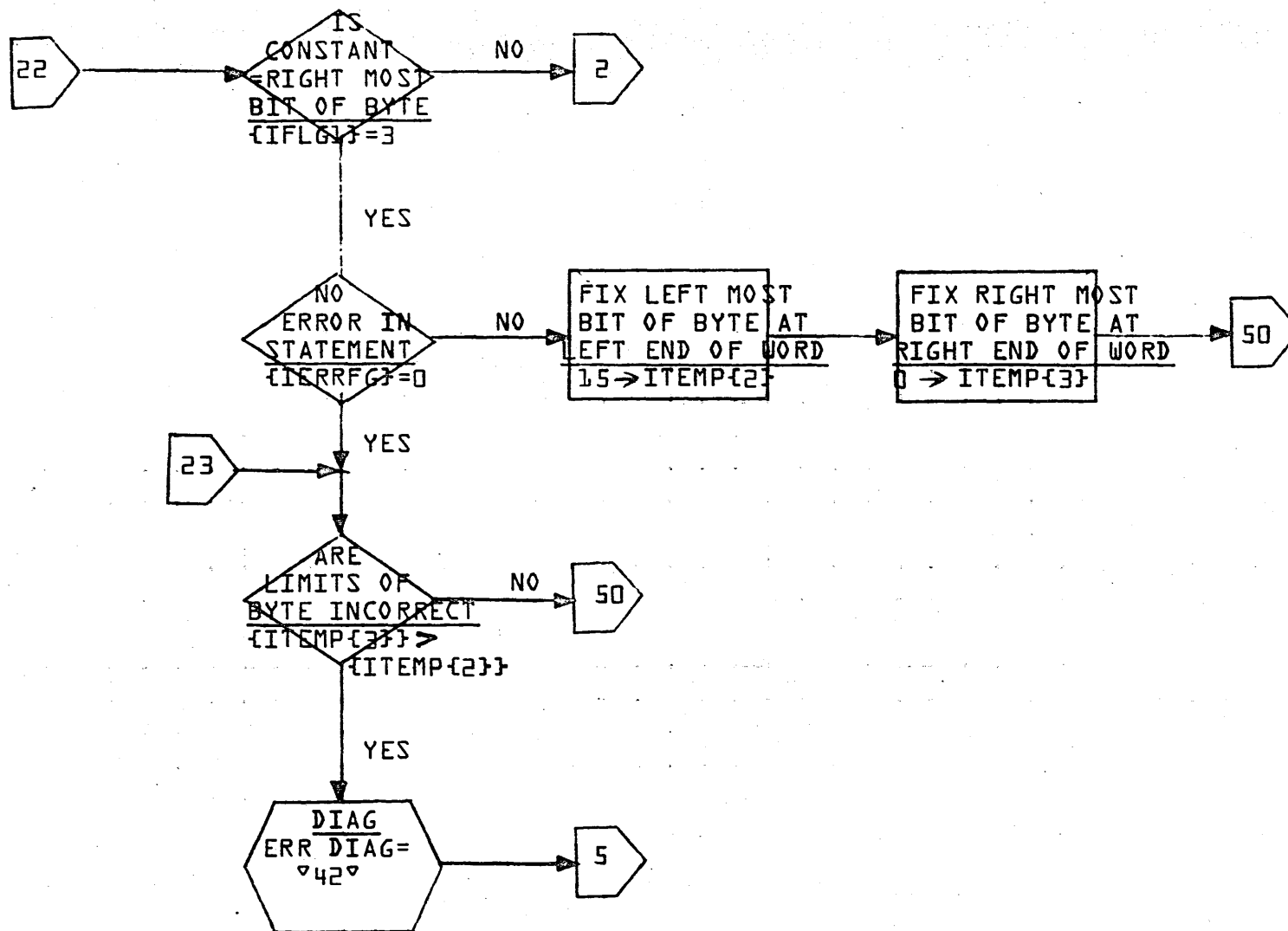
2-195

A

B

C

D


**CONTROL DATA CORPORATION**  
 SOFTWARE DOCUMENT

 SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER 

DOCUMENT CLASS	IMS	MACH. TYPE	1700
DOCUMENT TITLE	BYEQPR	PAGE	12F 15
NUMBER	ISSUE DATE	DRAWN BY	DATE

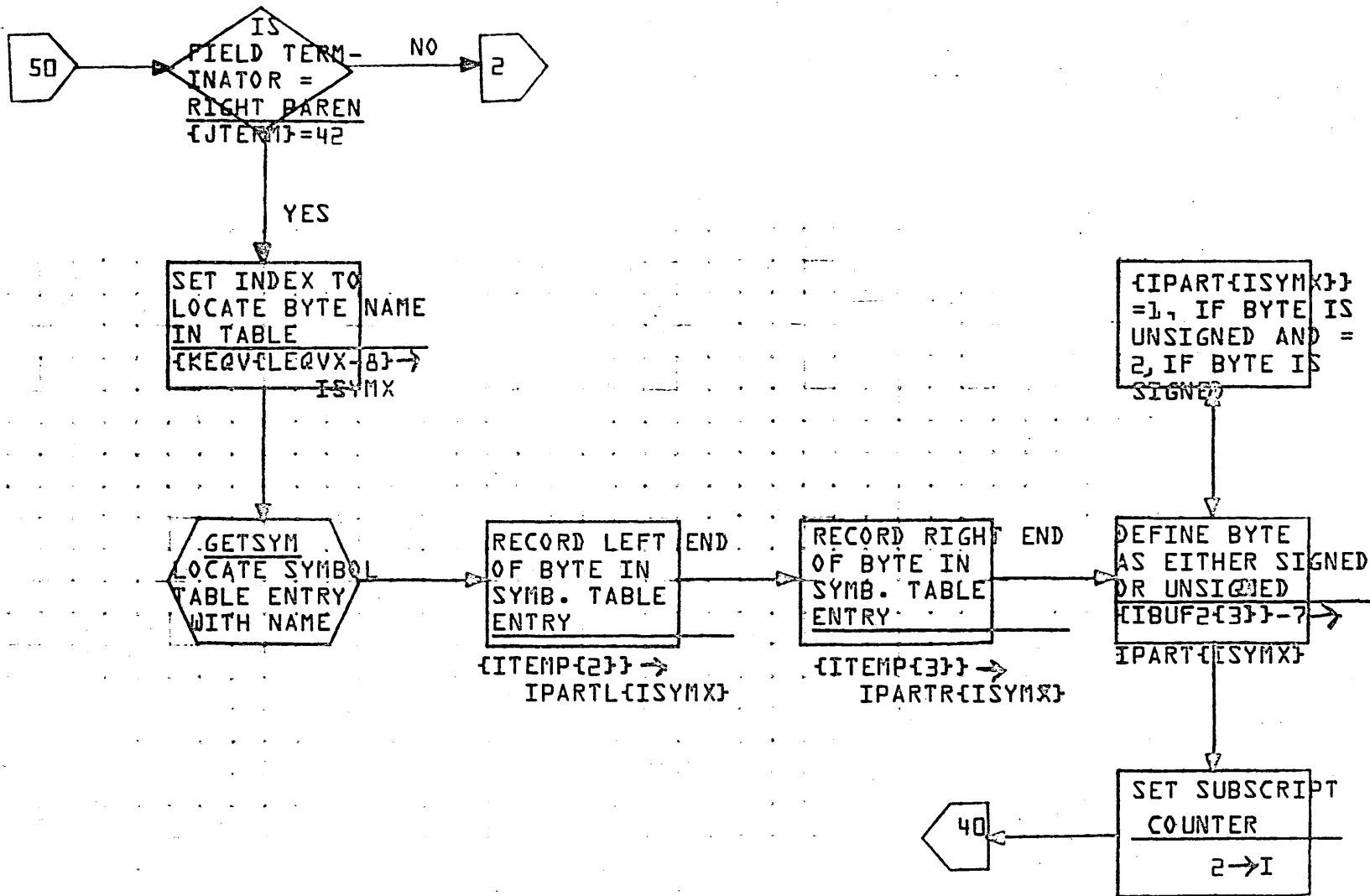
PROJECT NO.	REV	APPROVED	DATE
PROJECT MGR.			
PROJECT NAME			
TASK NO.			
TASK NAME			

A

B

C

D

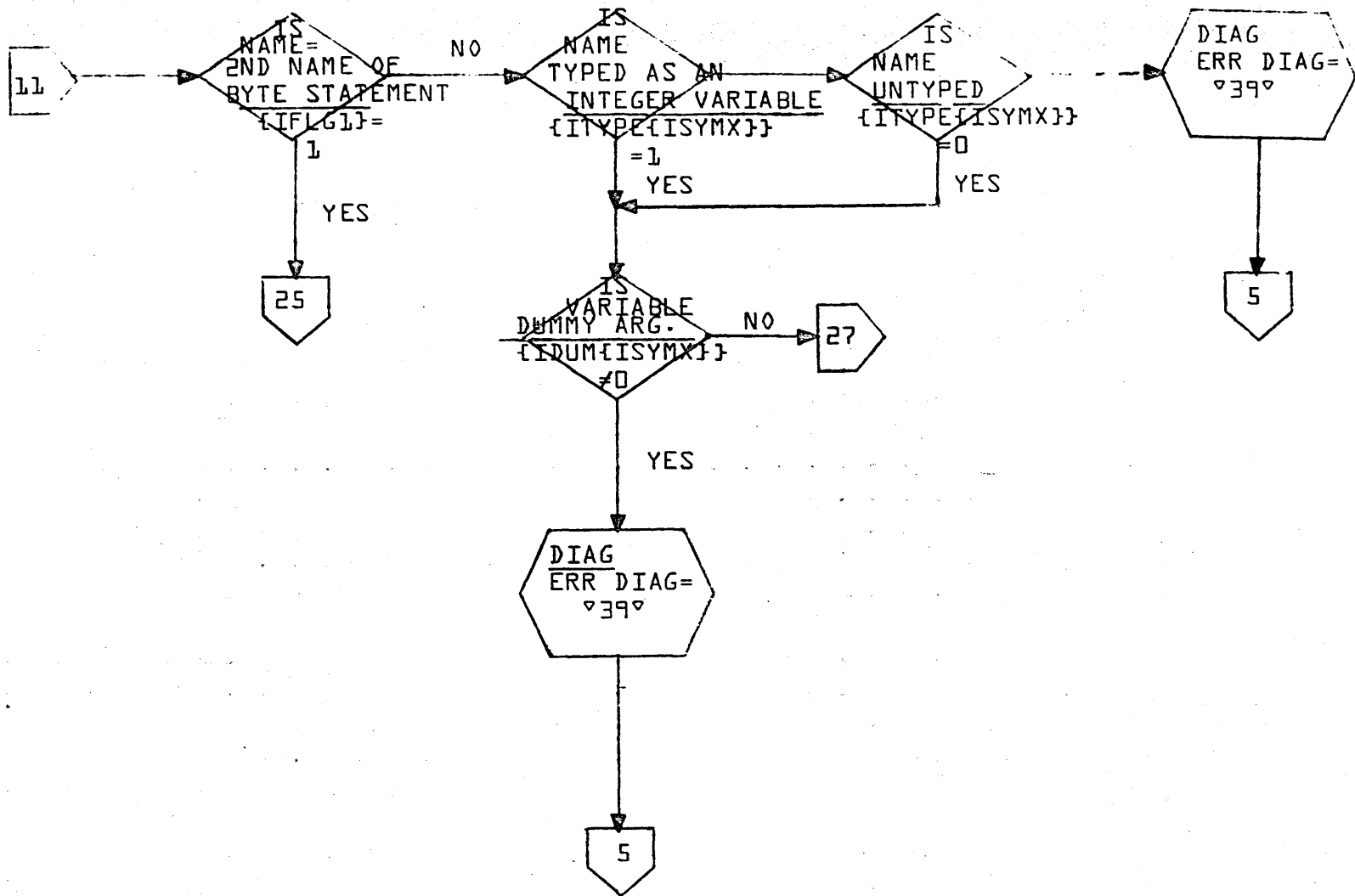


**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.		APPROVED	DATE
DOCUMENT TITLE	BYEQPR			PROJECT MGR.			
		PAGE	13F 15	PROJECT NAME			
NUMBER		ISSUE DATE		TASK NO.			
DRAWN BY		DATE		TASK NAME			

2-197



**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	BYEQPR	PAGE 14 OF 15		PROJECT MGR.			
NUMBER	ISSUE DATE			PROJECT NAME			
DRAWN BY	DATE			TASK NO.			
				TASK NAME			

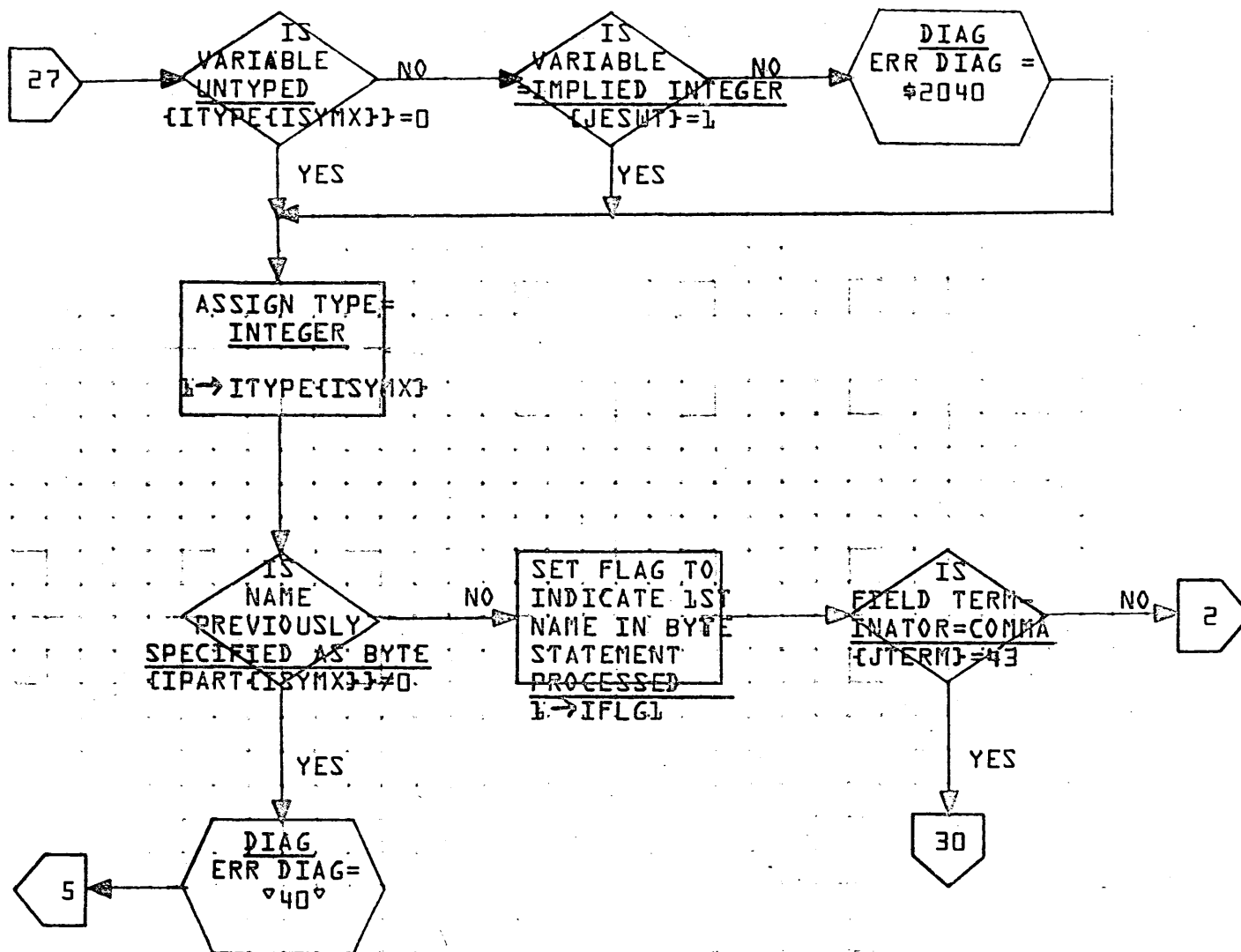


A

B

C

D



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS IMS	MACH. TYPE 1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE BYEQPR		PROJECT MGR.			
	PAGE 1 OF 1,5	PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.			
DRAWN BY	DATE	TASK NAME			

2-179

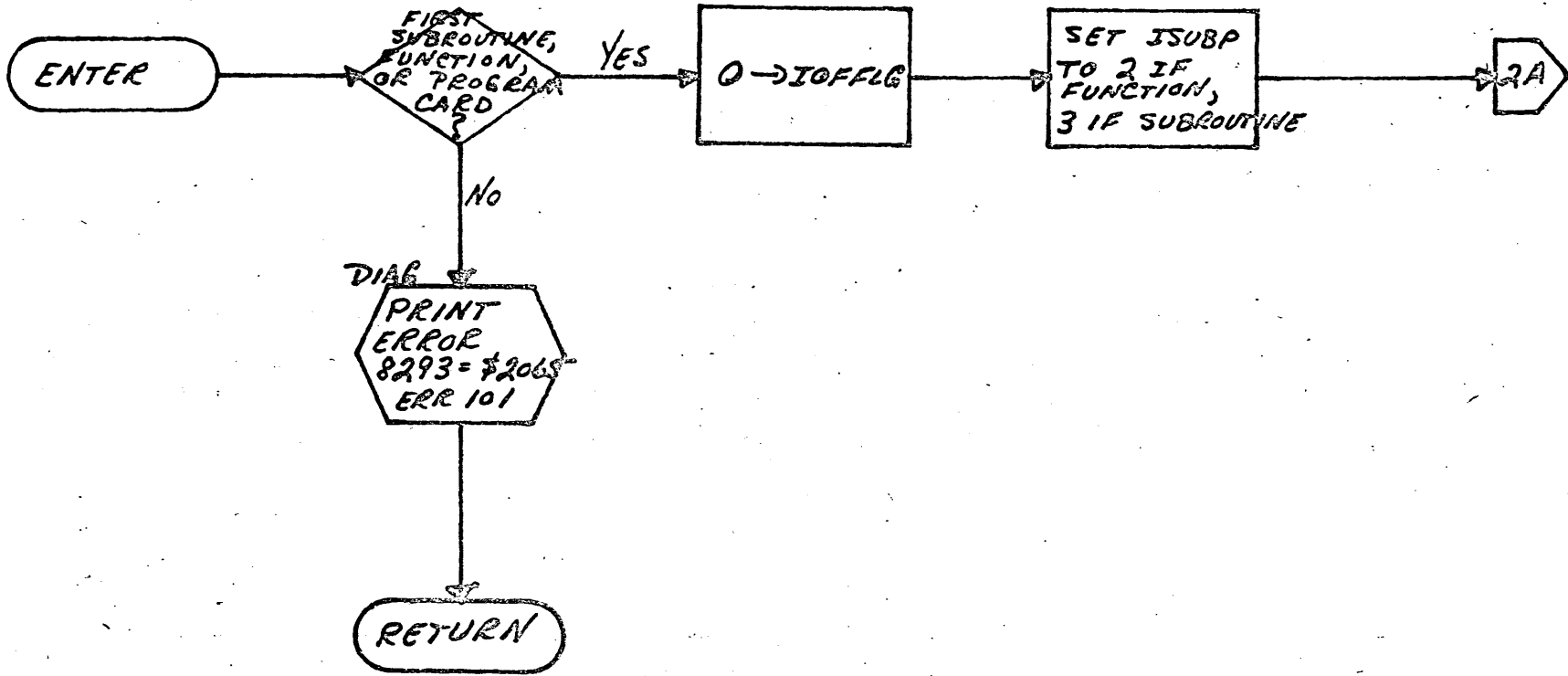
DOCUMENT CLASS IMS PAGE NO. 2-200  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

2.4.5 SUBROUTINE or FUNCTION Statement - Subroutine SUBPPR

SUBPPR processes SUBROUTINE or FUNCTION statements and is called directly by PHASE A. It sets a flag to distinguish between a subroutine and a function. (2 = function, 3 = subroutine.)

The next field is checked to see if it is a name (JMODE = 2). The name is checked for legality. If legal, it is put in the symbol table. If not, a diagnostic is printed. The terminator is checked. If not a left parenthesis, SUBPPR returns to PHASE A. If left parenthesis, fields are checked until a right parenthesis is found. Each name encountered is checked for legality and entered in the symbol table. The symbol table pointer goes to output buffer.

SUBPPR



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

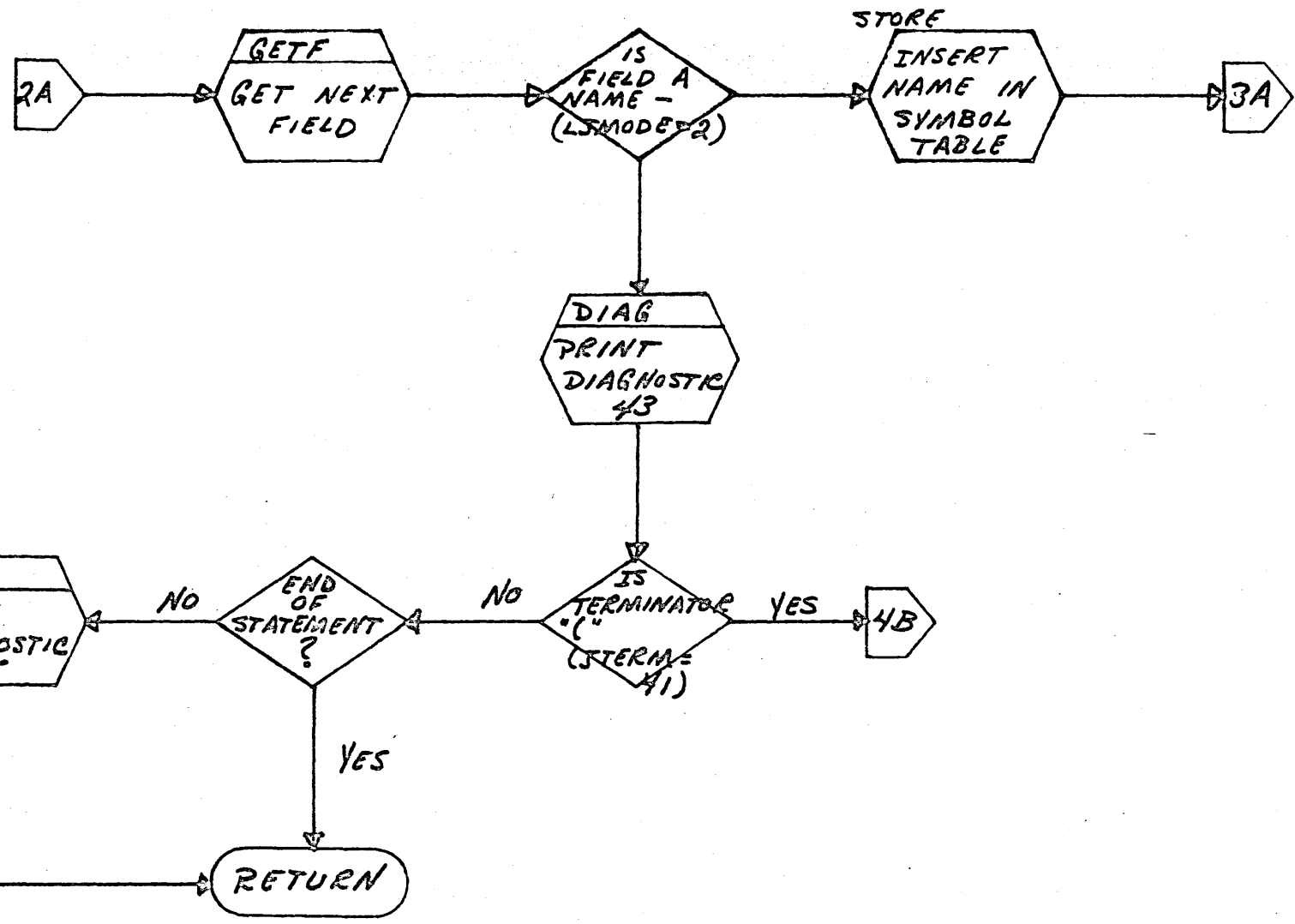
DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV.	APPROVED	DATE
DOCUMENT TITLE	SUBPPR	PAGE 1 of 6		PROJECT MGR.			
NUMBER		ISSUE DATE		TASK NO.			
DRAWN BY		DATE		TASK NAME			

A

B

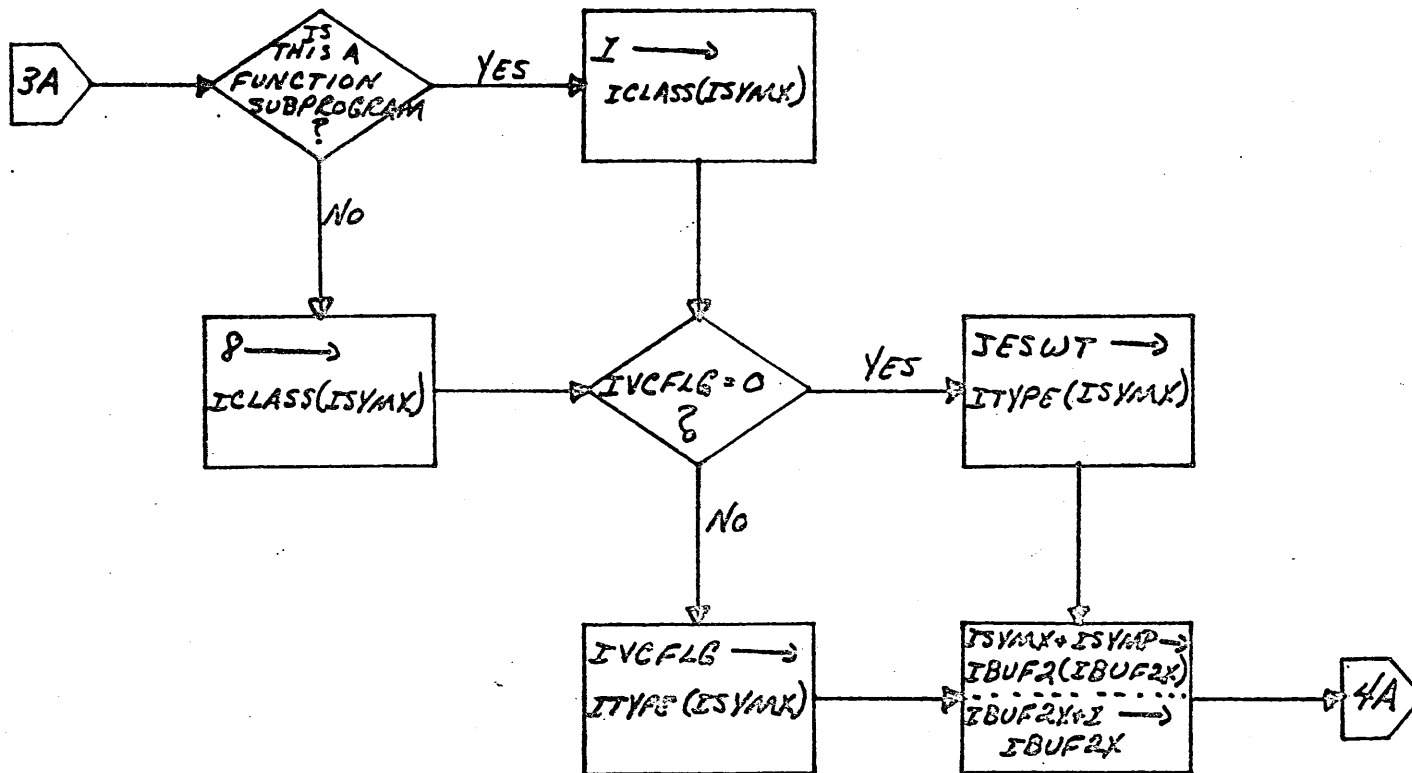
C

D



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	SUBPPR		PAGE	2 OF 6	PROJECT MGR.		
	NUMBER	ISSUE DATE			PROJECT NAME			
	DRAWN BY	DATE			TASK NO.			
					TASK NAME			

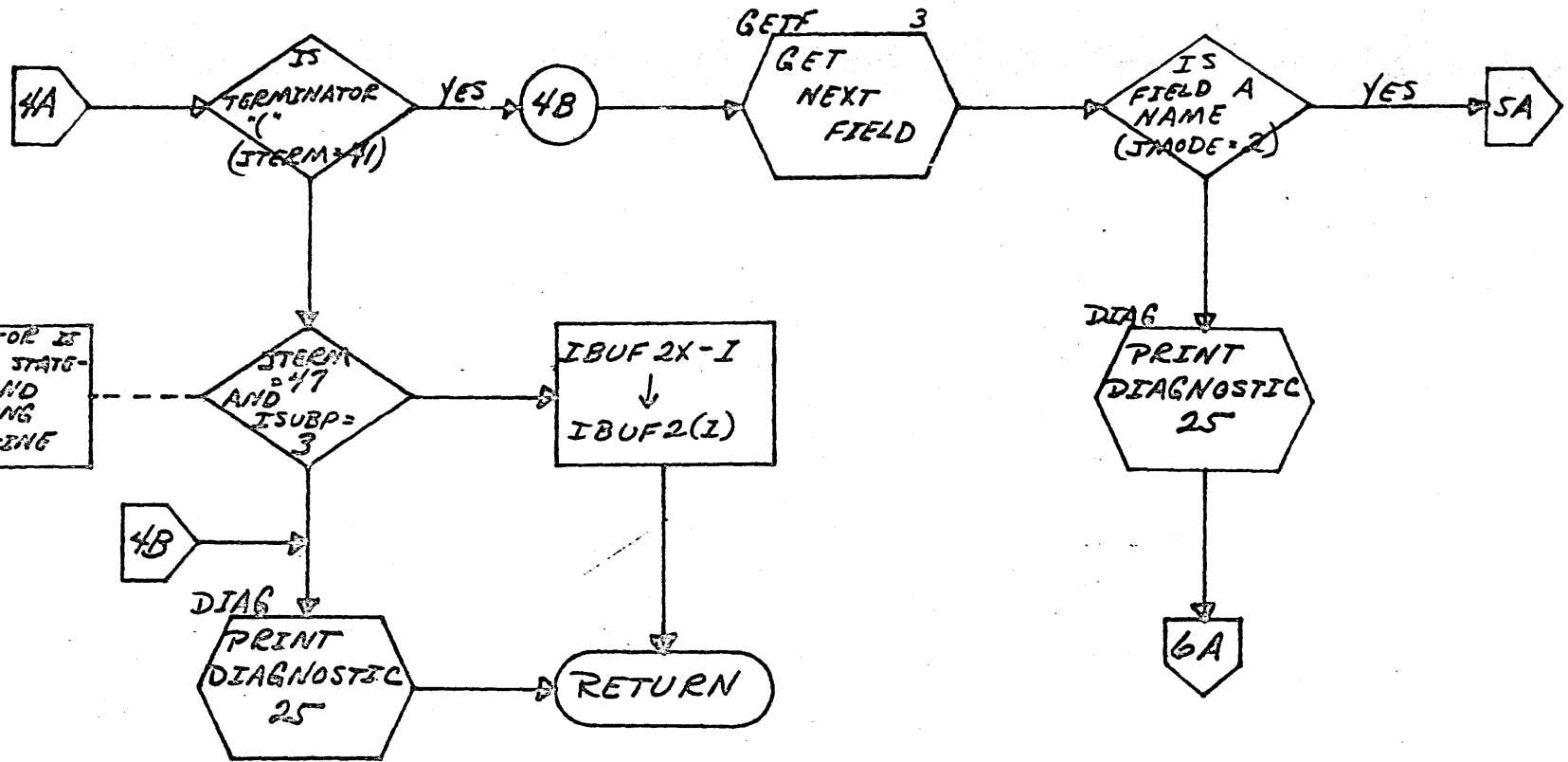
A  
B  
C  
D



**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	SUBPPR	PAGE 3 OF 6		PROJECT MGR.			
NUMBER		ISSUE DATE		PROJECT NAME			
DRAWN BY		DATE		TASK NO.			
				TASK NAME			

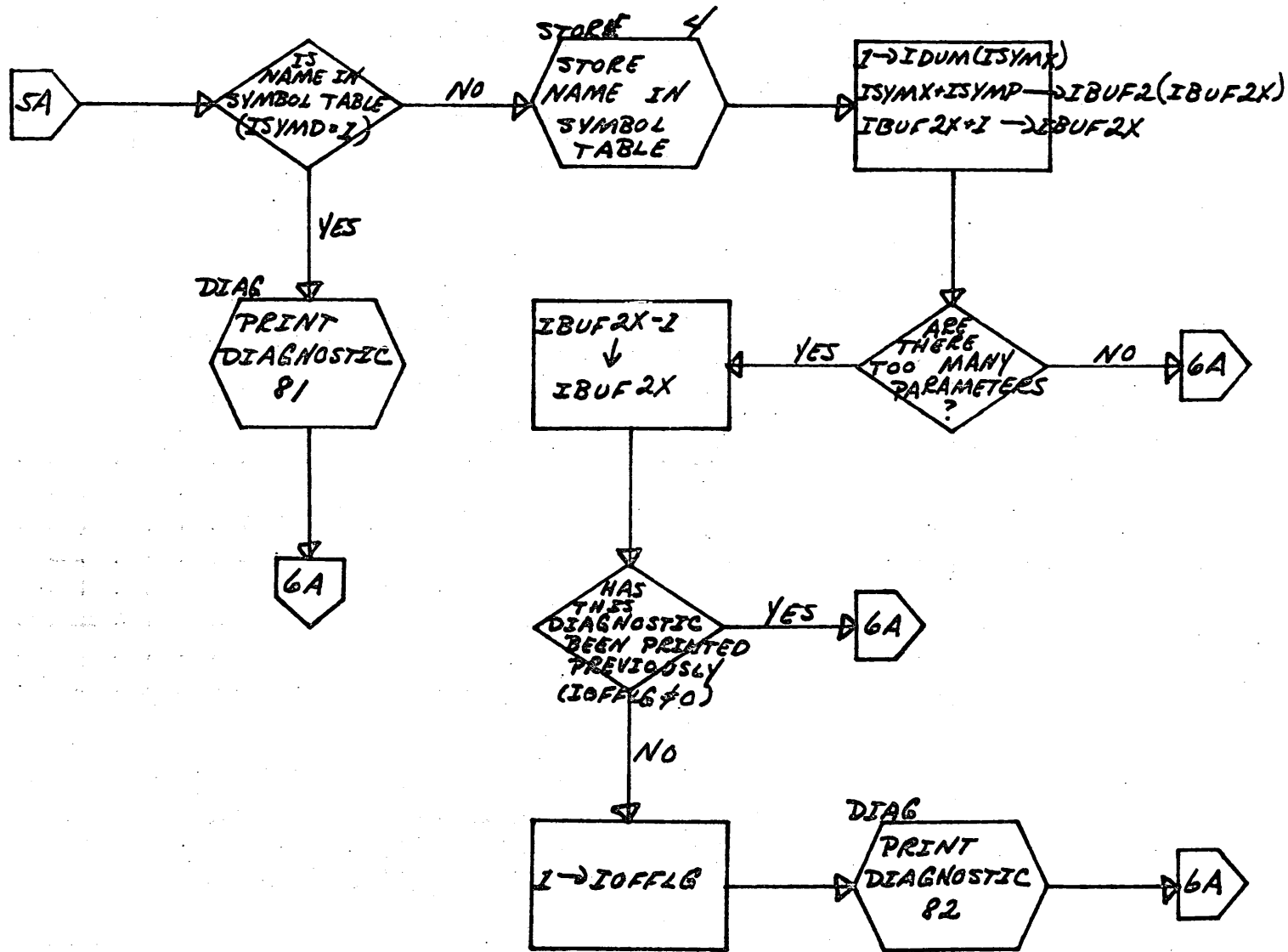


CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE   
FLOWCHART   
DECISION TABLE   
OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700
DOCUMENT TITLE	SUBPPR	PAGE 4 OF 6	
NUMBER	ISSUE DATE		
DRAWN BY	DATE		

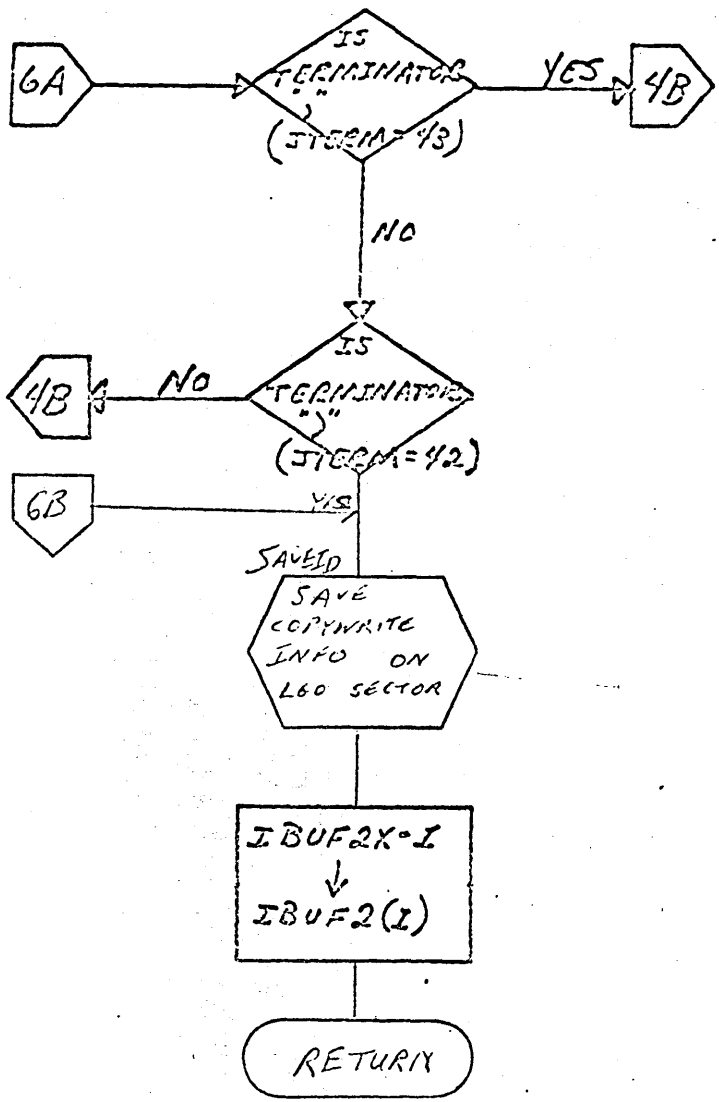
PROJECT NO.	REV	APPROVED	DATE
PROJECT MGR.			
PROJECT NAME			
TASK NO.			
TASK NAME			



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	SUBPPR	PAGE 5 OF 6		PROJECT MGR.			
NUMBER	ISSUE DATE			PROJECT NAME			
DRAWN BY	DATE			TASK NO.			
				TASK NAME			



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	SUEPPR	PAGE 6 of 6		PROJECT MGR.			
NUMBER	ISSUE DATE	TASK NO.					
DRAWN BY	DATE	TASK NAME					



DOCUMENT CLASS IMS PAGE NO. 2-207  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

2.4.6 DATA Statements - Subroutine DATAPR

The DATAPR subroutine is used to process DATA statements. Intermediate language will be produced in the output buffer for the names and constants which appear in the source statement. The output buffer is named IBUF2. The tally register that holds the word count for the information placed in the output buffer is IBUF2X. IBUF2X is set to an initial value prior to entering the DATAPR subroutine. Intermediate language generated for each member of a name list in the source statement is inserted in the output buffer beginning at

IBUF2{IBUF2X}.

The tally register IBUF2X is increased by the number of words of intermediate language generated for the member being processed. Upon exit from DATAPR, IBUF2X contains the word length of the output entry.

2.4.6.1 Processing Name List

The DATAPR subroutine calls the GETF subroutine to extract a member of the name list from the source statement. A member of the name list must be alphanumeric. If the name does not appear in the symbol table, it is recorded therein by a call to the STORE subroutine. If the name is untyped, an indicator is set in the symbol table entry to the type number for the name:

{JEWST} → ITYPE{ISYMX}

where JEWST is set as a result of the call to GETF. Another indicator is set in the symbol table entry to show that this name appeared on a data statement:

1 → IDATAS{ISYMX}

If the symbol is unclassified, the indicator in the symbol table entry is set to show that this symbol is classified as a variable name:

1 → ICLASS{ISYMX}

If the entry containing this name indicates that the name is classified, it must be classified as a variable name. The variable may not be used as a byte, a dummy variable, or variable assigned to blank common.

If the name is an element of an array, a call is made to the CONSUB routine in order to compute the increment from the subscripts of the element. Upon return from

DOCUMENT CLASS IMS PAGE NO. 2-208  
PRODUCT NAME I700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES I700

CONSUB, the value in IVCFLG indicates that the element is the IVCFLGth member of the array. If each element of the array occupies one storage cell,

{IVCFLG} - 1

is the increment in words which must be added to the base address of the array in order to reference the element in the array. Each element of the array occupies one storage cell if either -

1. The name is typed as an integer, or
2. the ASA option has been chosen and the name is typed as a single.

In other words

either {ITYPE{ISYMX}} = 1, or {IK} ≠ 0 and {ISNGL{ISYMX}} ≠ 0.

If each element of the array occupies 2 storage cells -

{IVCFLG}\*2-2

is the increment in words which must be added to the base address of the array in order to reference the element in the array. Each element of the array occupies 2 storage cells if either -

1. The name is typed as real, or
2. the ASA option is chosen but the integer element is not typed as a SINGLE.

In other words -

ITYPE{ISYMX} = 2 or {ITYPE{ISYMX}} = 1 and either {IK} = 0, or {IK} ≠ 0 and {ISNGL{ISYMX}} = 0

If each element of the array occupies 3 storage cells -

{IVCFLG}\*3-3

is the increment in words which must be added to the base address of the array in order to reference the element in the array. Each element of the array occupies 3 storage cells if it is double precision regardless of the ASA option.

The members of the list of names must be separated by commas and the last name in the list must be followed by a slash. When DATAPR encounters the slash, it terminates processing the list of names and proceeds to process the list of constants. {Refer to item 2.4.6.2} Members of the name list may be variables, array elements, arrays, and implied DO's. When an array name is found, it is reduced into a series of array elements. When an

DOCUMENT CLASS IMS PAGE NO. 2-209  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

implied D0 is analyzed, a series of array elements is also generated.

#### 2.4.6.1.1 Intermediate Language Generated as a Result of Processing the Name List

The starting position in IBUF2 of the intermediate language is recorded in ITEMP1 prior to processing the 1st name in the list. As each member of the list of names is processed, the following intermediate language is generated:

WORD 1: {IBUF2{IBUF2X}} = pointer to the name in the symbol table.

WORD 2: {IBUF2{IBUF2X+1}} =

1. 0 if the variable is not an element of an array,
2. {IVCFLG}-1 if the variable is an element of an array where each element of the array occupies 1 storage cell,
3. IVCFLG\*2-2 if the variable is an element of an array where each element of the array occupies 2 storage cells, and
4. IVCFLG\*3-3 if the variable is an element of an array where each element of the array occupies 3 storage cells.

WORD 3: {IBUF2{IBUF2X+2}} = the number of storage cells the variable occupies {according to its type}.

The tally register is increased by the number of words of intermediate language to be generated for this name

{IBUF2X}+2+{IBUF2{IBUF2X+2}} → IBUF2X

When the DATAPR subroutine encounters a slash indicating the end of the name list, an end of output mark is generated in the intermediate language buffer by

-1 → IBUF2{IBUF2X}

#### 2.4.6.1.2 Error Checking While Processing Name List

Diagnostics will be given for the following variables:

1. A member of the name list is not alphanumeric.
2. A member of a name list appears in the symbol table as something other than a class 1 variable.

DOCUMENT CLASS IMS PAGE NO. 2-210  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. CD05\*3.1 A/B MACHINE SERIES 1700

3. The member of the name list appears in the symbol table, but an indicator is set to show this symbol is a byte.
4. The member of the name list appears in the symbol table, but an indicator is set to show this symbol is either in numbered common or used as a dummy variable.
5. Either the members of the name list are not separated from each other by commas, or the final member in the list is not followed by a slash.

Errors 1 and 5 will cause processing of the source statement to be terminated by an exit from DATAPR. For the remaining errors, processing of the source statement is resumed following the output of the error indication.

#### 2.4.6.2 Processing Constant List

In a DATA statement, there must be a one to one correspondence between the members of the constant list and the members of the preceding name list. The members of the constant list may be numeric and literal. A literal constant must be contained by single quotes. Numeric constants may be double, real, or integer and may have a leading algebraic sign of "+" or "-". If the value of the constant is to be assigned to consecutive variables in the preceding name list, it may be preceded by a repeat count equal to the number of times it is to be used. The repeat count for the constant must be an integer, and it must be separated from the constant by an asterisk. In order to comply with the one to one correspondence rule as stated above, the following condition must be satisfied when a member of a constant list is preceded by a repeat count:

$$C+RC=V$$

where

1. C represents the number of constants in the list without repeat counts,
2. RC represents the sum of the repeat counts occurring in the constant list, and
3. V represents the number of variables in the preceding name list.

When using literal constants, the number of characters must be taken into consideration. If the variable type is:

DOCUMENT CLASS IMS PAGE NO. 2-211  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. COO5\*3.1 A/B MACHINE SERIES 1700

Single or Integer	it holds	2 characters
Real	it holds	4 characters
Double Precision	it holds	6 characters

Trailing blanks are used to fill a variable entry that has been started and an end of literal quote is encountered, that is:

D is a double precision variable.

DATA D/'ABC'/

The variable D will be initialized as if the DATA statement were:

DATA D/'ABCbbb'/' b = blanks

Since a numeric constant may be double precision, real or integer, it must agree in type with the variable in the preceding name list to which it is assigned.

The members of the constant list must be separated from each other by commas. The final member of the list must be followed by a slash which is followed by either a comma or an EOS. If the list terminates with a slash and an EOS, processing of DATA statement terminates when processing of the current constant list terminates. If the list terminates with a slash and a comma, DATAPR proceeds to process the list of names which follow in the manner describe by item 2.4.6.1.

#### 2.4.6.2.1 Completing the Intermediate Language in the Output Buffer

The tally register IBUF2X is set to the value recorded in ITEMP1 prior to processing the list of names preceding the list of constants. {See item 2.4.6.1.1} The DATAPR subroutine calls GETF to extract a member of the list of constants from the source buffer. Prior to calling GETF, the repeat ICOUNT is set to 1. Upon return from GETF, one of the following conditions has occurred:

CONDITION 1: The operand is not preceded by a repeat count, or a quote, or a leading algebraic sign. Upon return from GETF -

JTERM	=	comma, slash, or EOS
JMODE	=	3 for integer constant, 5 for real constant and 6 for double precision constant
JSYM{1}	=	value of integer constant
JSYM{1} and JSYM{2}	=	value of real constant
JSYM{1}, JSYM{2} and JSYM{3}	=	value of double precision constant

DOCUMENT CLASS: IMS PAGE NO: 2-212  
 PRODUCT NAME: 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. COO5\*3.1 A/B MACHINE SERIES: 1700

The numeric constant and the name from the preceding list to which the constant is assigned must agree in type such that -

CONSTANT	JMODE		IBUF2{IBUF2X+2}
double precision	6	agrees with	3
real	5	agrees with	2
integer	3	agrees with	1

The constants are entered into the intermediate language by -

```
integer      JSYM{1} → IBUF2{IBUF2X+2}
real         JSYM{1}, JSYM{2} → IBUF2{IBUF2X+2},
             IBUF2{IBUF2X+3}
double precision JSYM{1}, JSYM{2}, JSYM{3} → IBUF2
             {IBUF2X+2}, IBUF2{IBUF2X+3}, IBUF2
             {IBUF2X+4}
```

The tally register is increased by 3, 4, 5 accordingly.

CONDITION 2: The operand is preceded by a leading algebraic sign. Upon return from GETF -

JTERM = either "+" or "-"  
 JMODE = 0 indicating a null field

A second call is made to GETF to extract the numeric constant from the source statement. If the algebraic sign was a "-", the value for the constant is complemented by:

```
- JSYM{1} → JSYM{1},
- JSYM{2} → JSYM{2},
and - JSYM{3} → JSYM{3}.
```

If the leading algebraic sign was a "+", the sign is ignored. The constant is then processed as if it had no leading algebraic sign. {See condition 1}

CONDITION 3: The constant is preceded by a repeat count. The repeat count must be an integer separated from the constant by an asterisk. Upon return from GETF -

JTERM = "\*", JMODE = 3, and JSYM{1} = value of repeat count.

The repeat count is recorded in ICOUNT by -

```
JSYM{1} → ICOUNT
```

A second call is made to GETF to obtain the constant. If it is a literal constant, it is processed as condition 5. The numeric constant is assigned to a number of

DOCUMENT CLASS IMS PAGE NO. 2-213  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

consecutive names from the preceding name list until ICOUNT = 0. The constant must match "type wise" each of the consecutive names to which it is to be assigned. The constant is inserted into the intermediate language in IBUF2 for each of the names in the manner described by condition 1. IBUF2X is increased by the number of words of intermediate language for each entry made in IBUF2. The repeat count is reduced by 1 for each entry made into IBUF2. When ICOUNT = 0, the repeat conditions are satisfied. The repeat count must not be so large that it is not zero when the end of the name list is encountered in IBUF2. The end of the intermediate language is marked by an entry in which:

IBUF2{IBUF2X} = -1

A flag IVBUF is checked to see if the name list has been exhausted. If IVBUF = 1 then more intermediate language will be generated and IBUF2 will be written out and re-filled. If IVBUF = 0 and the end of the intermediate language is encountered, then the end of the name list will have been reached. {See item 2.4.b.1.1}

CONDITION 4: The numeric constant is preceded both by a repeat count and has a leading algebraic sign. First, the repeat count is processed according to condition 3. Then the leading algebraic sign is processed according to condition 2.

CONDITION 5: The operand is preceded by a single quote, indicating a literal constant. Literal constants must have a terminating single quote. Upon return from GETF -

JTERM = "'" single quote  
 JMODE = 0 indicating a null field

The routine GETC is used to extract characters from the source statement. Characters are extracted until a terminating quote is encountered.

The characters are entered into the intermediate language according to the following conditions.

Type of NAME	No. of CHARACTERS
Integer	2
Real	4
Double precision	6

DOCUMENT CLASS IMS PAGE NO. 2-213A  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. COO5\*3.1 A/B MACHINE SERIES 1700

Characters are assigned to consecutive names until the literal is terminated. If assignment has begun and the literal terminates, the remainder of the name is assigned with blanks. {That is, if a name, whose type is real, was assigned a character and the end of literal was encountered 3 blanks are assigned to fill up the entry.} The intermediate language pointer is updated after each name assignment by:

Type of Name → IENTRY  
IBUF2X+IENTRY+2 → IBUF2X

#### 2.4.6.2.2 Error Conditions Check for While Processing Constant List

Diagnostics will be given for the following errors:

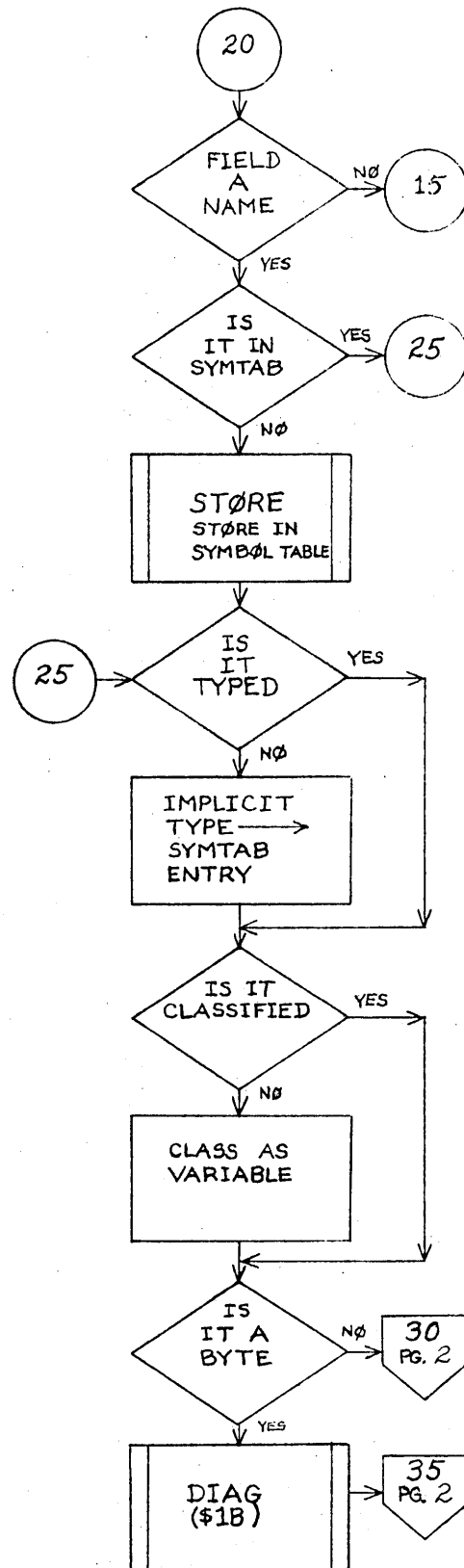
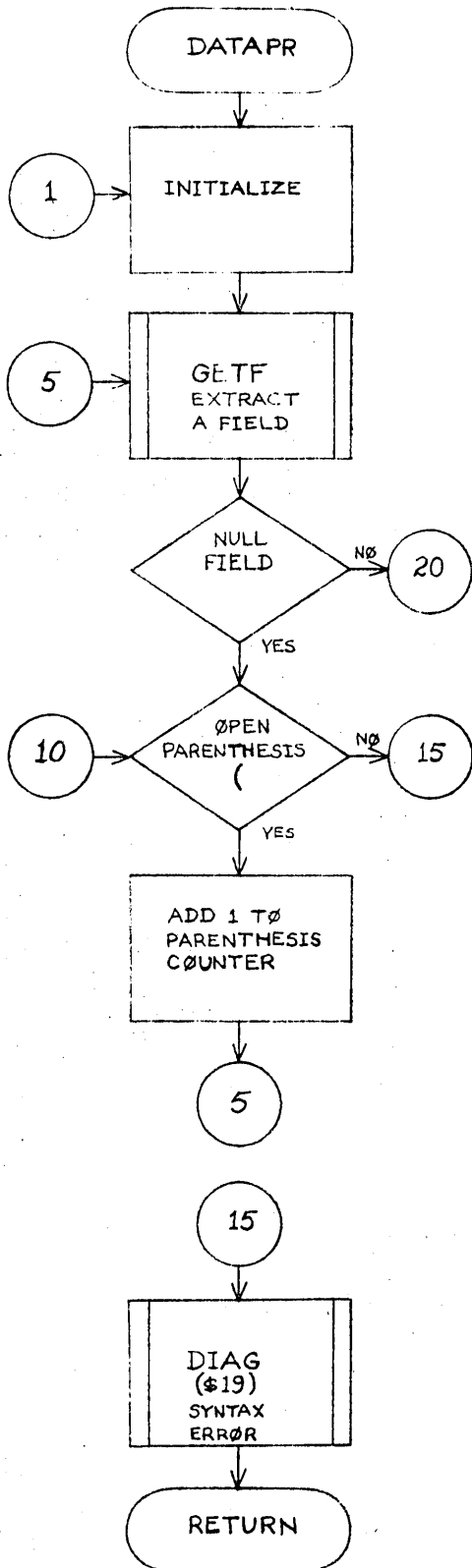
1. The repeat counter is either non-integer or is  $\leq 0$  or both.
2. A member of the constant list is both non-numeric and not a literal.
3. The numeric constant does not agree in type with the name {from the preceding name list} to which it is assigned.
4. There is not a one to one correspondence between the members of the name list and the members of the constant list. This could be caused by a repeat count which is excessive in value, it could also be caused by an incorrect number of characters in the literal constant.
5. The last member of the constant list is not followed by either a "/" or a "/EOS".
6. The members of the constant list are not separated by commas.
7. A member of the constant list is followed by an EOS not preceded by a slash.

Errors 1, 4, 5, 6 and 7 will cause processing of the source statement to be terminated by an exit from DATAPR. Processing of the source statement will be resumed following the printing of the error indication for errors 2 and 3. For illegal numeric constants, zeros will be substituted in their place.





LA JOLLA FACILITY

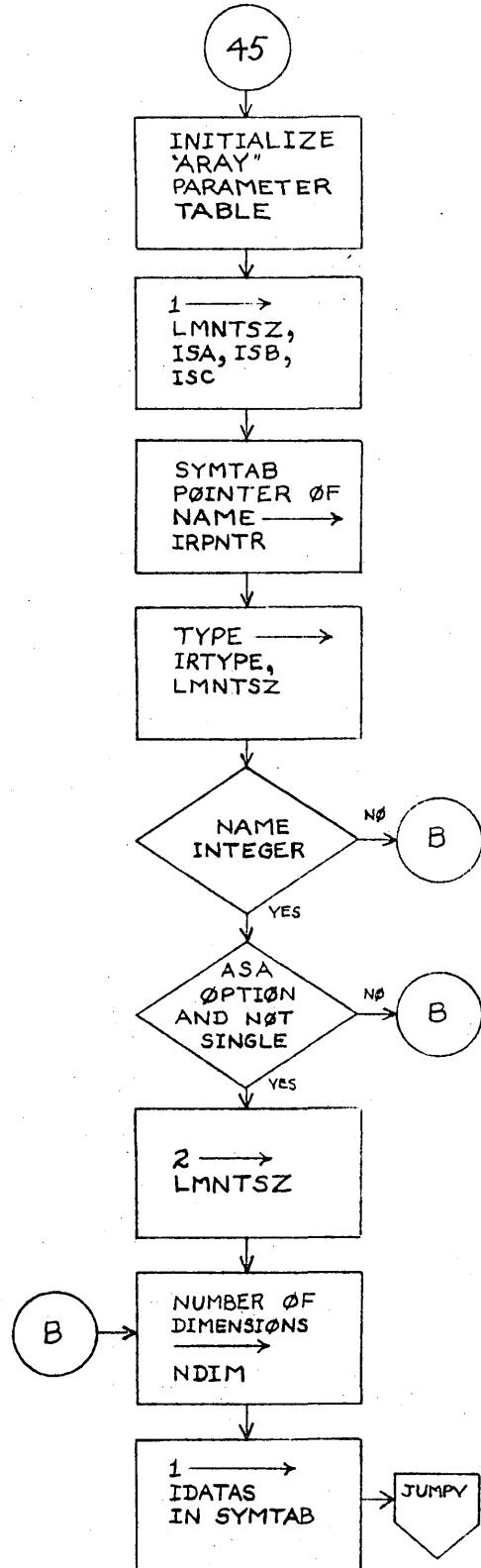
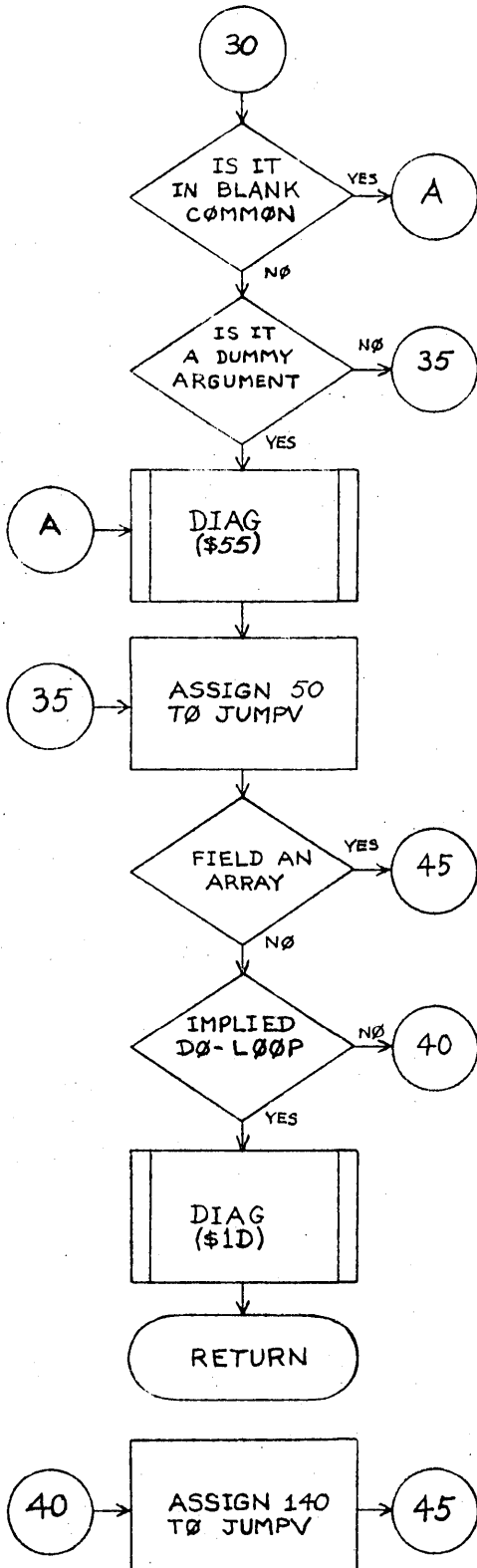


TITLE			DRG. NO.	
DATAPR				
DRAWN BY			REVISION	
PROJ.	DATE	SHEET 1	OF 11	





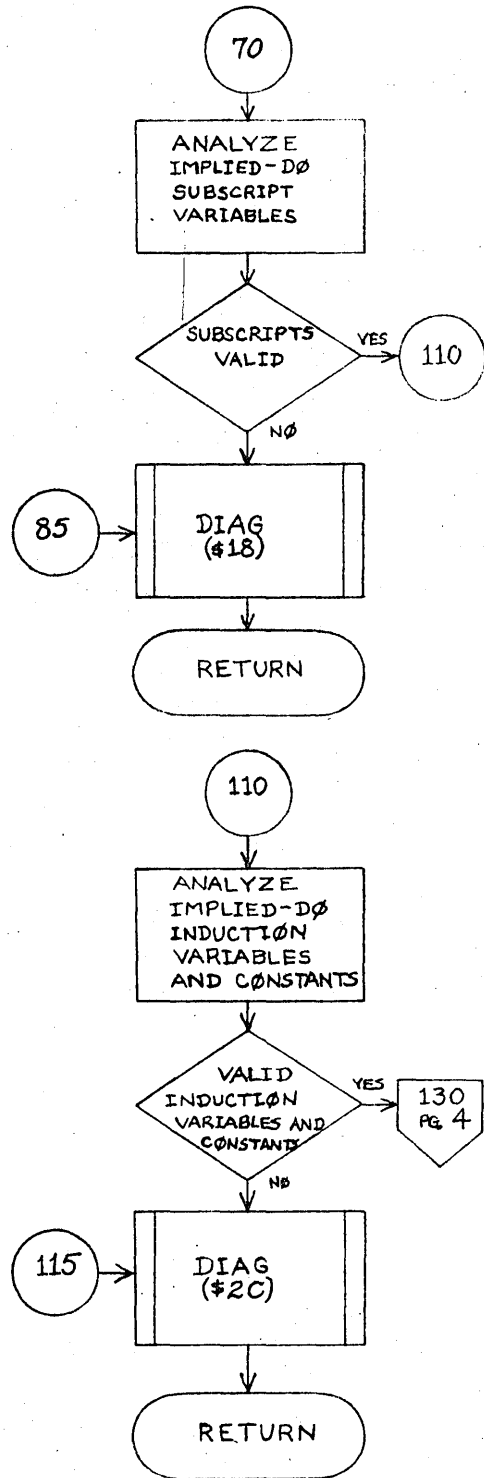
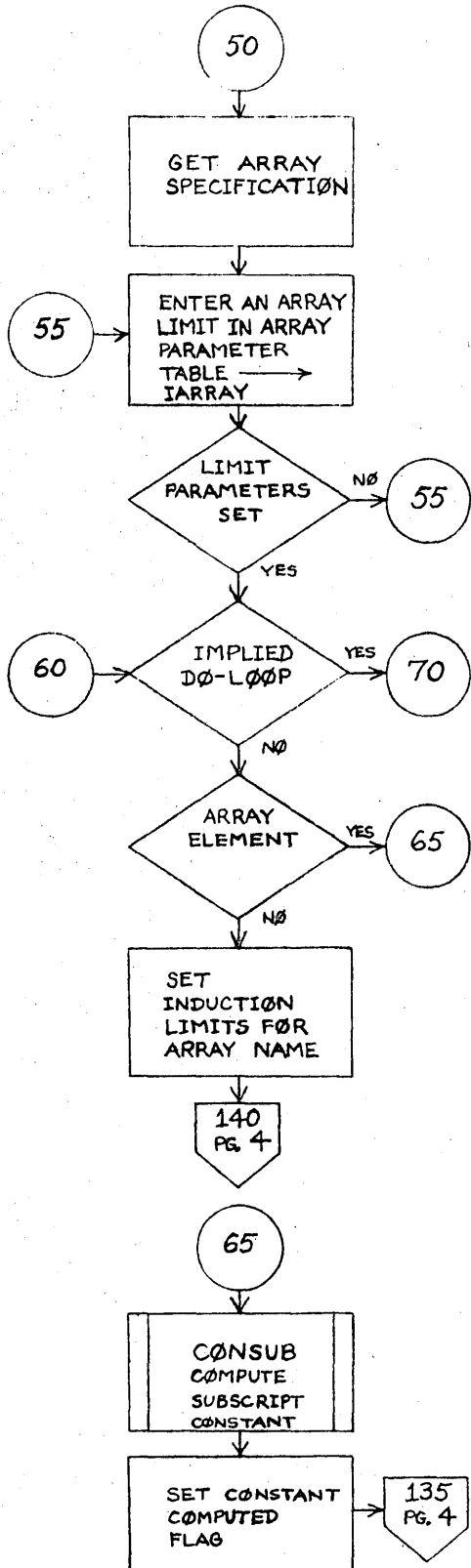
LA JOLLA FACILITY



TITLE		DATAPR		DRG. NO.	
				REVISION	
DRAWN BY	PROJ.	DATE	SHEET	2	OF 11



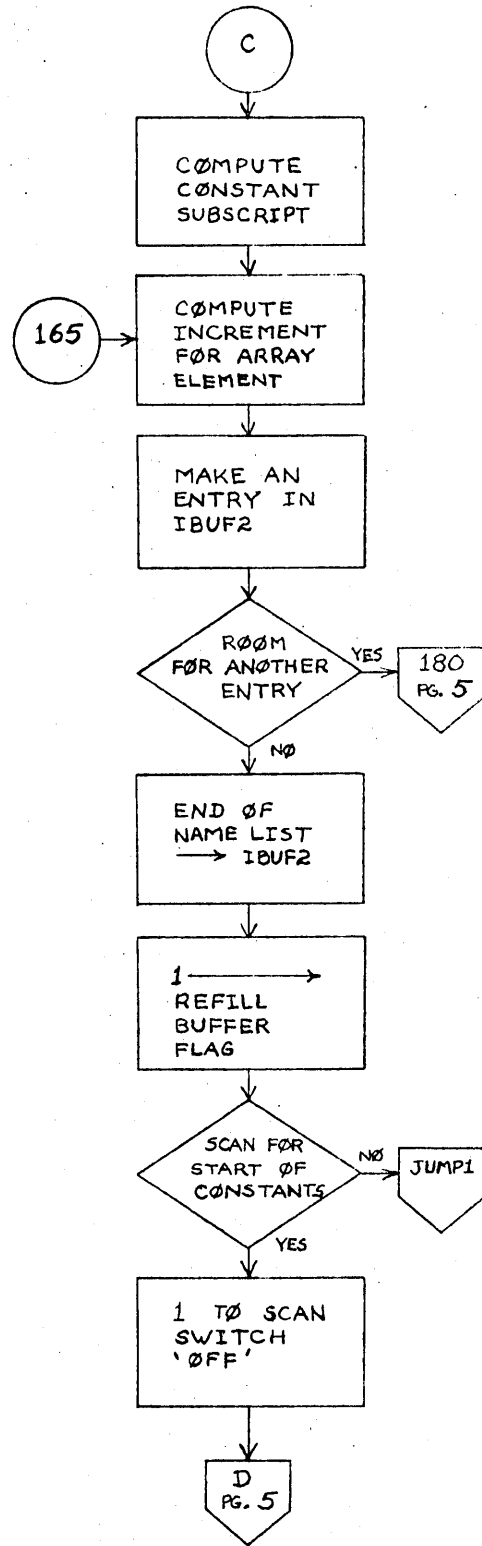
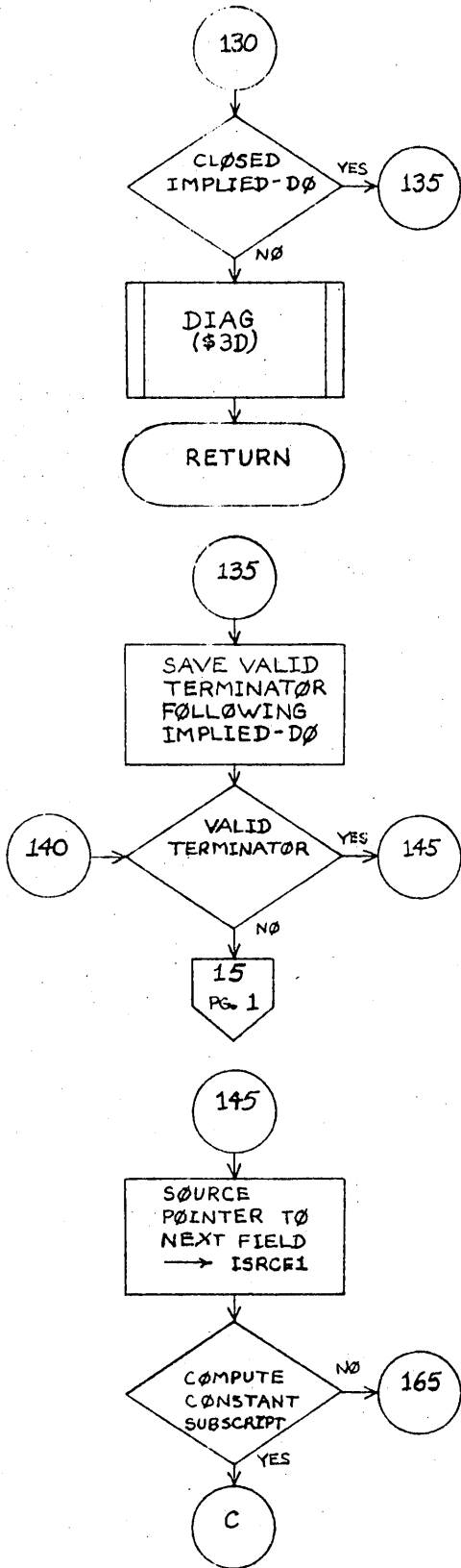
LA JOLLA FACILITY



TITLE			DRG. NO.	
DATAPE			REVISION	
DRAWN BY	PROJ.	DATE	SHEET 3	OF 11



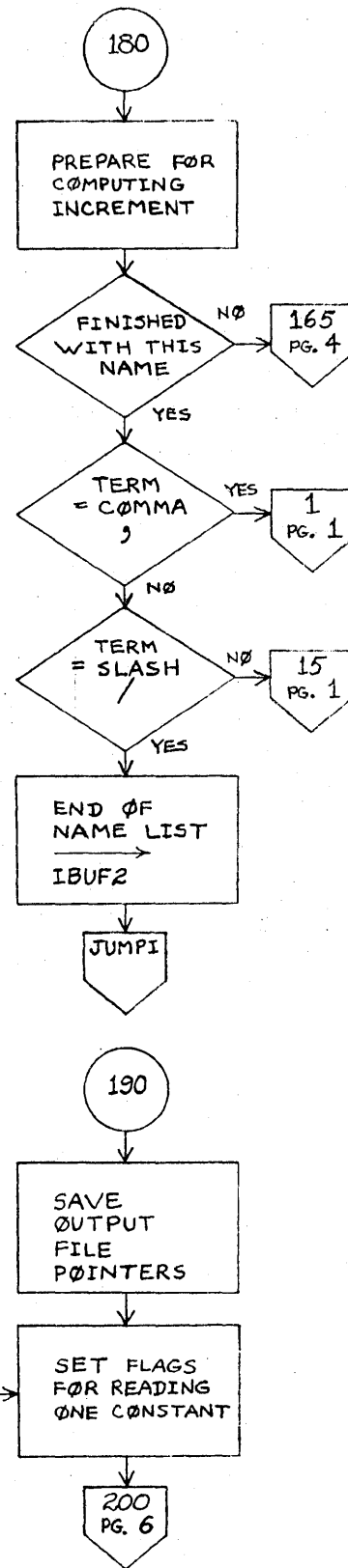
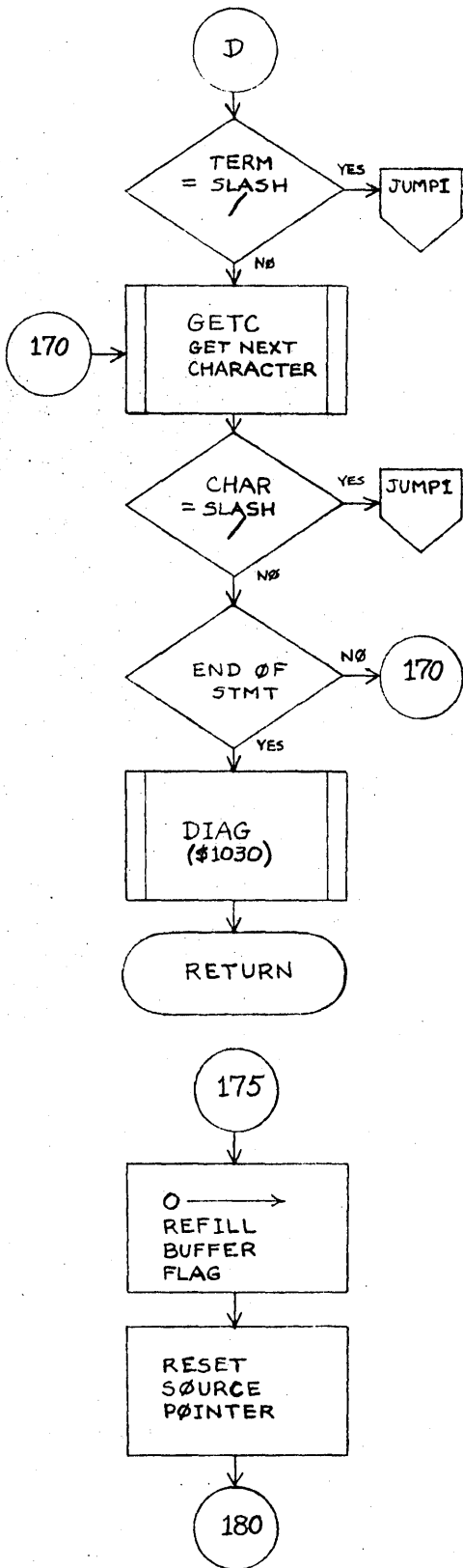
LA JOLLA FACILITY



TITLE		DATAPR		DRG. NO.	
				REVISION	
DRAWN BY	PROJ.	DATE	SHEET	4	OF 11



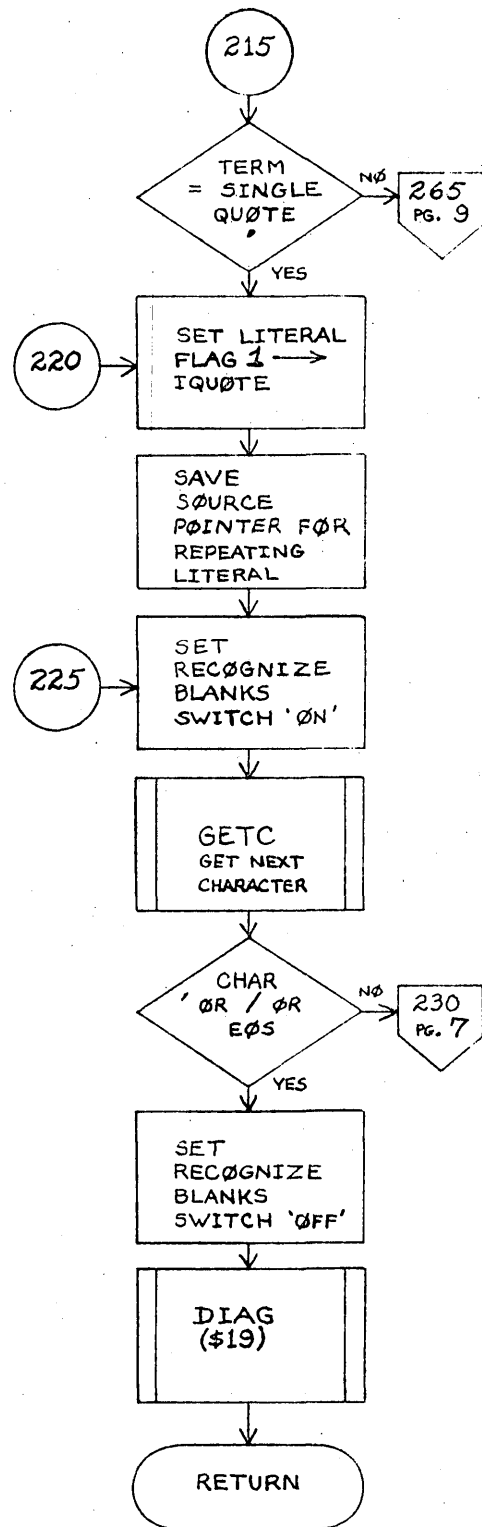
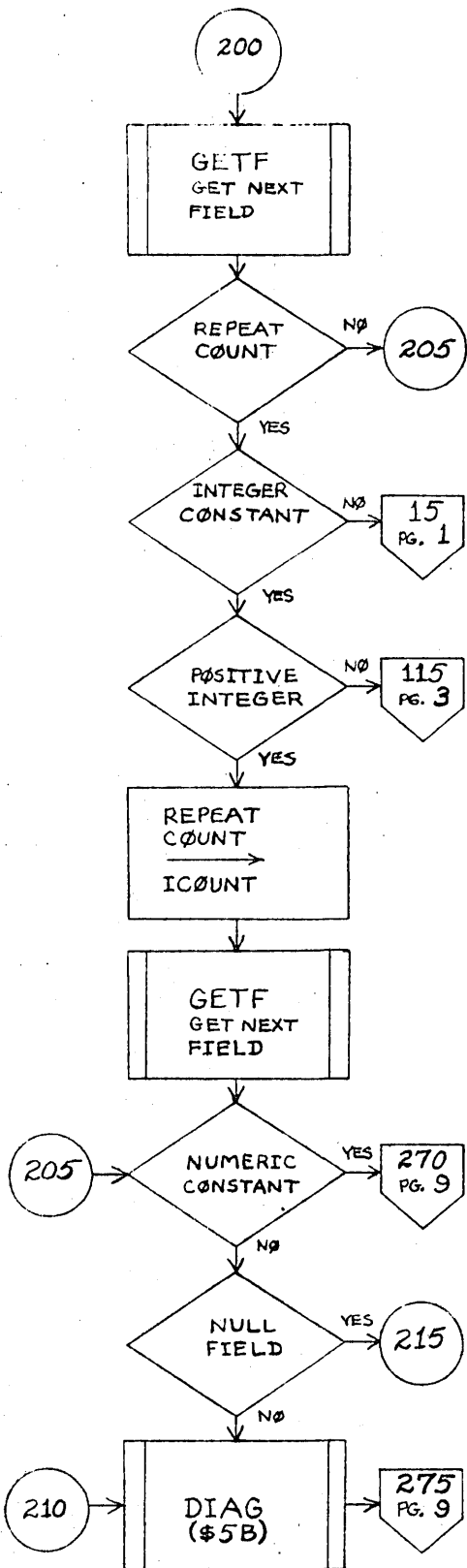
LA JOLLA FACILITY



TITLE			DRG. NO.	
DATAPR			REVISION	
DRAWN BY	PROJ.	DATE	SHEET 5	OF 11



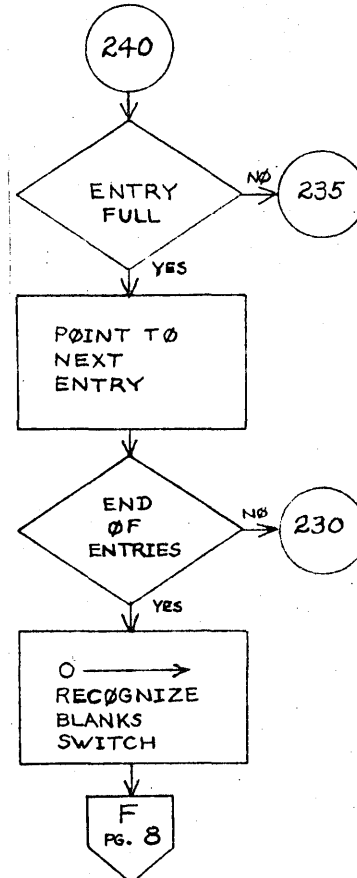
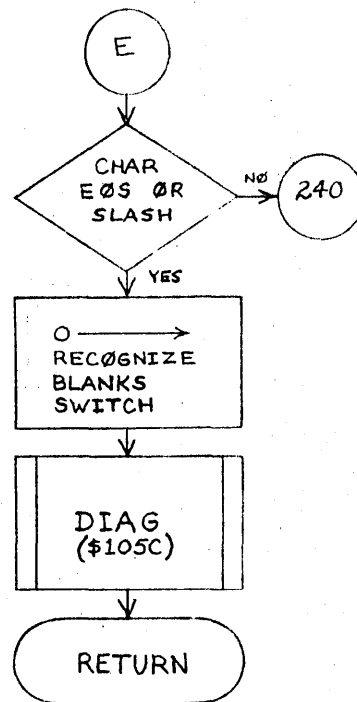
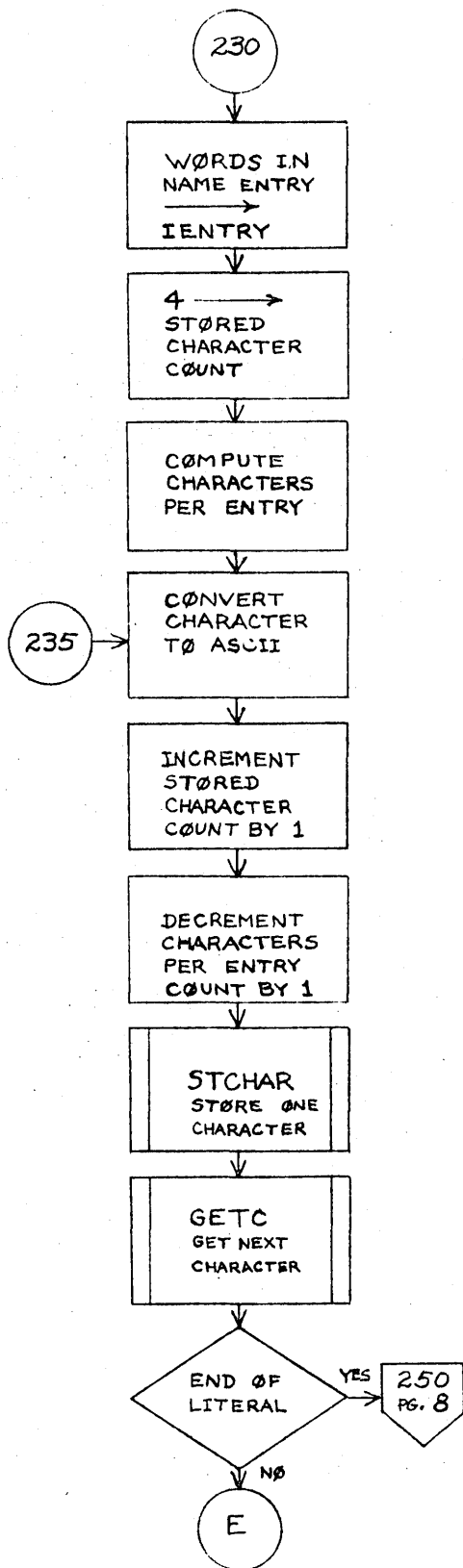
LA JOLLA FACILITY



TITLE			DRG. NO.	
DATAPR			REVISION	
DRAWN BY	PROJ.	DATE	SHEET 6	OF 11
CSD 203				



LA JOLLA FACILITY

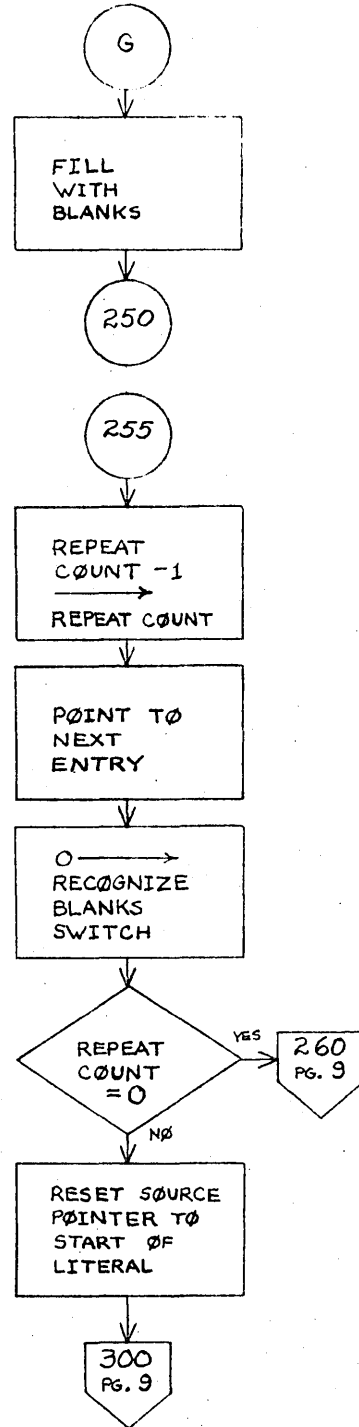
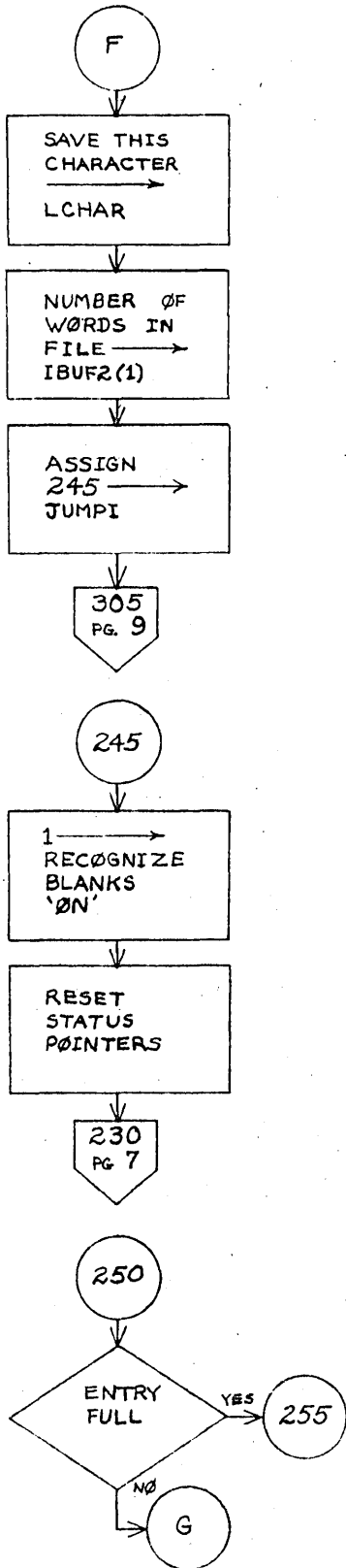


TITLE			DRG. NO.	
DATA PR			REVISION	
DRAWN BY	PROJ.	DATE	SHEET 7	OF 11





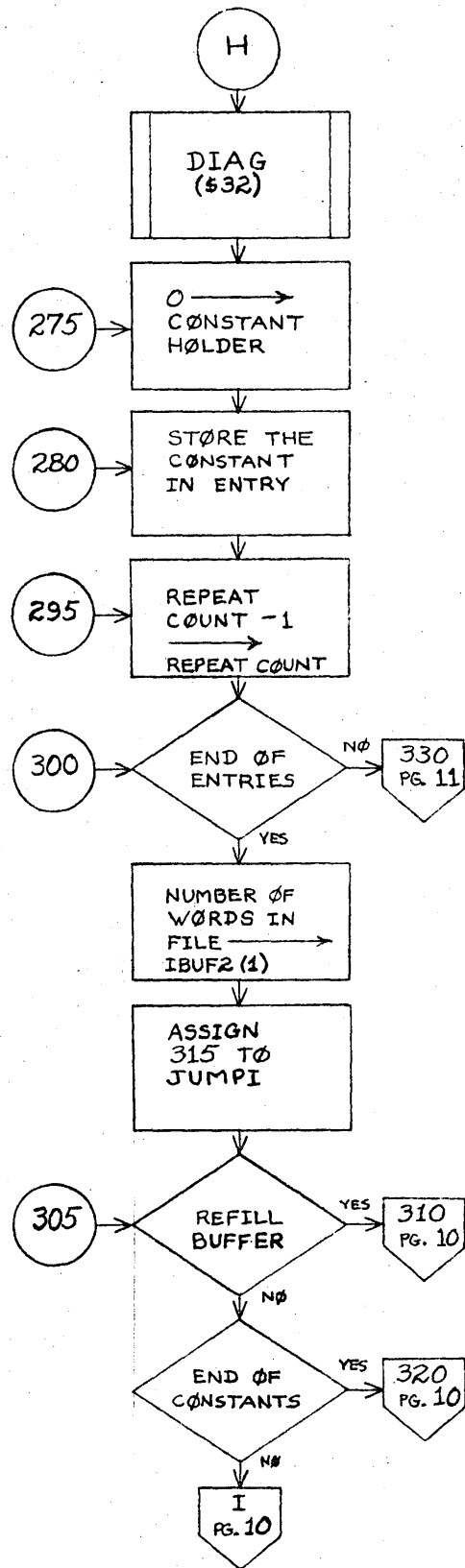
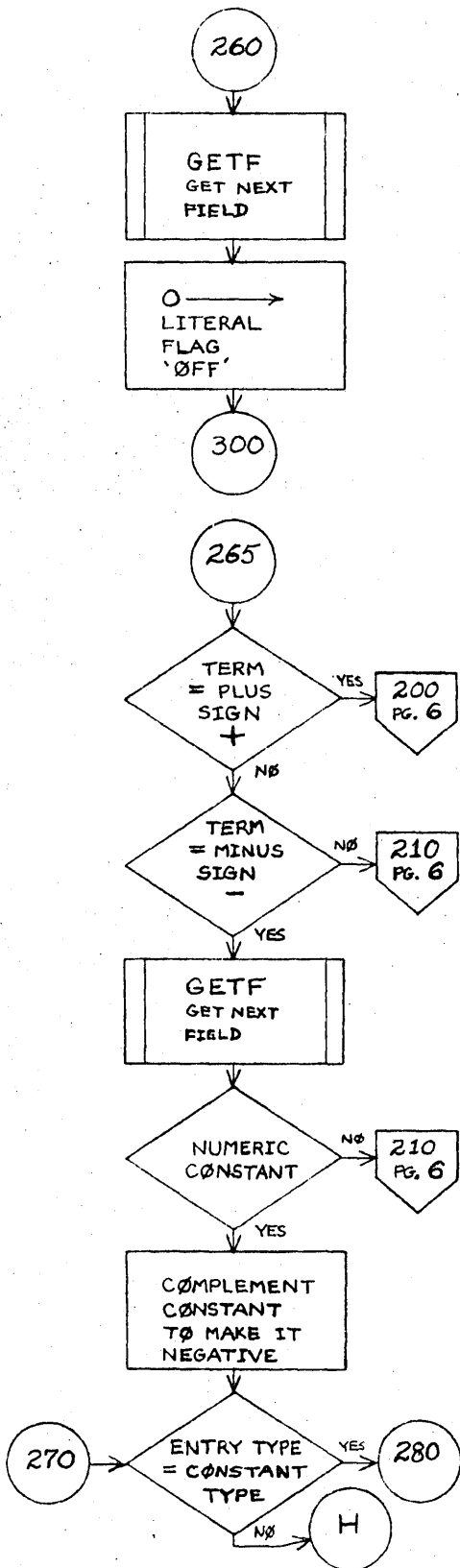
LA JOLLA FACILITY



TITLE		DRG. NO.	
DATAPR		REVISION	
DRAWN BY	PROJ.	DATE	SHEET 8 OF 11
CSD 203			



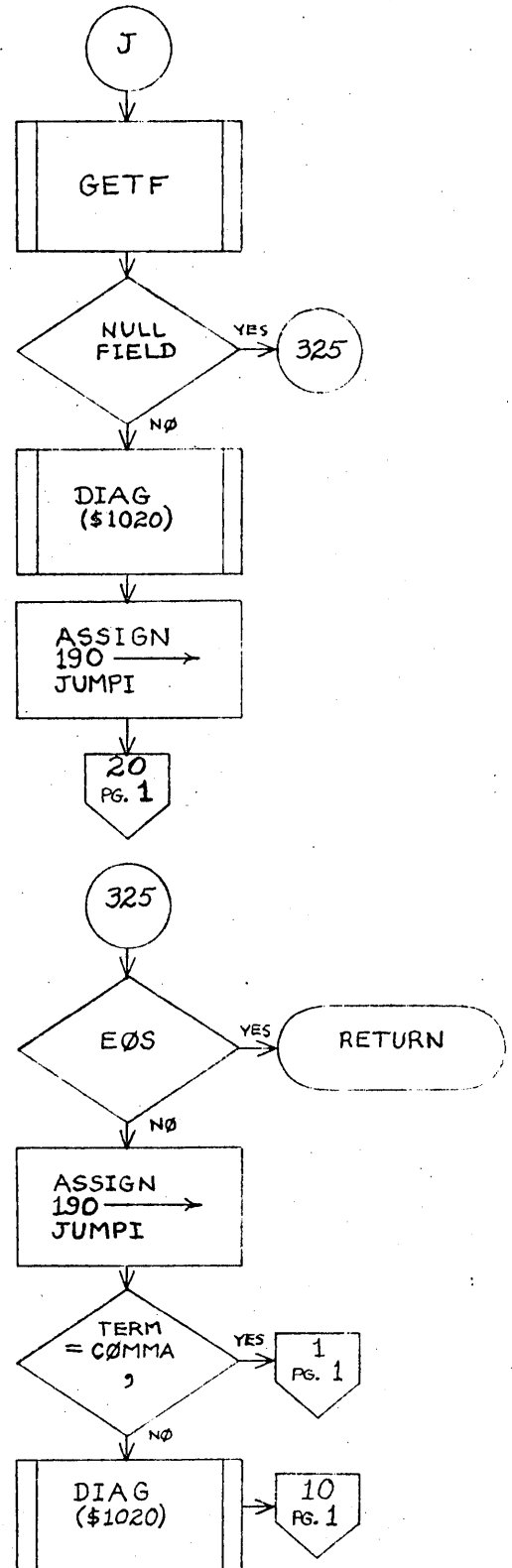
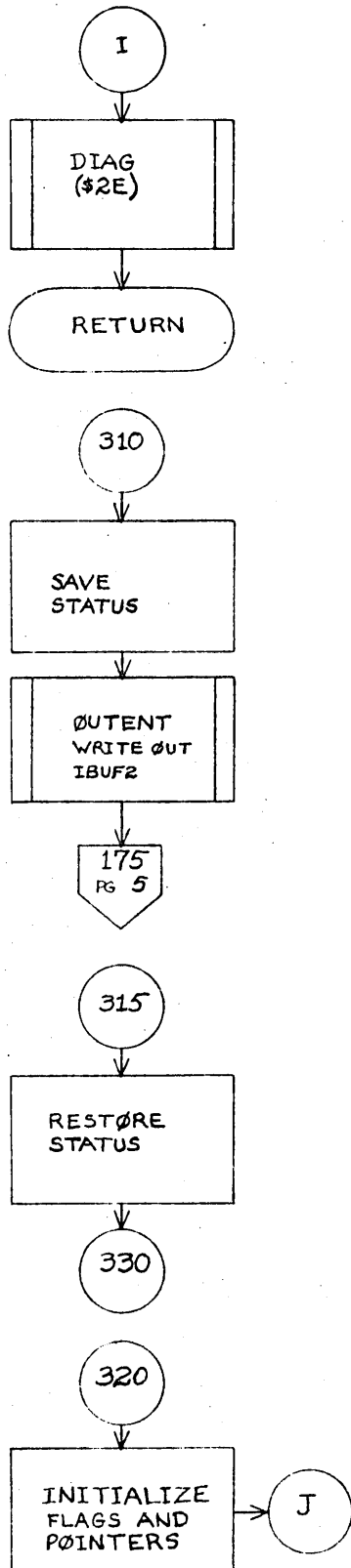
LA JOLLA FACILITY



TITLE			DRG. NO.		
DATAPR			REVISION		
DRAWN BY	PROJ.	DATE	SHEET	9	OF 11



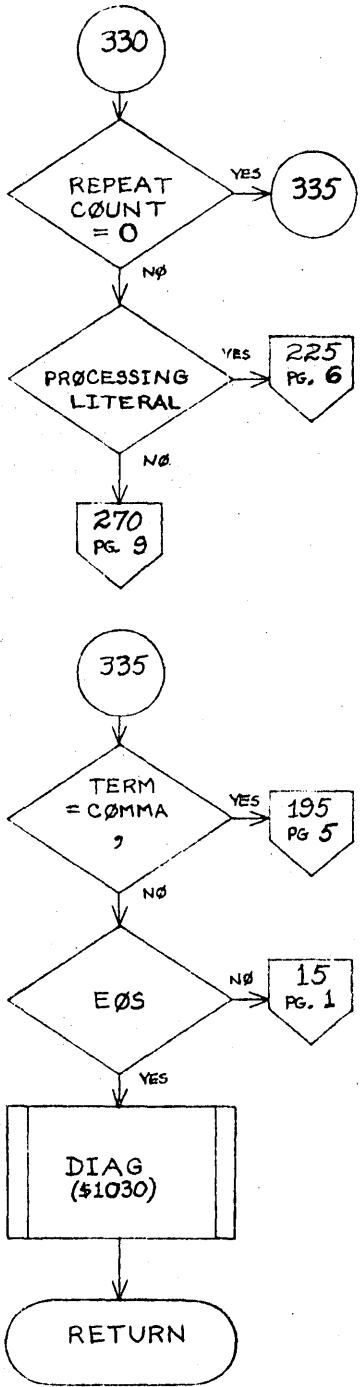
LA JOLLA FACILITY



TITLE		DATAPP.		DRG. NO.	
				REVISION	
DRAWN BY	PROJ.	DATE	SHEET	10	OF 11
CSD 203					



LA JOLLA FACILITY



TITLE		DATAPK		DRG. NO.	
				REVISION	
DRAWN BY	PROJ.	DATE	SHEET	11	OF 11

DOCUMENT CLASS IMS PAGE NO. 2-224  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

#### 2.4.7 FORMAT Statements

##### 2.4.7.1 Subroutine CHECKF

The routine CHECKF is entered. The statement is syntax checked, but remains a string of ASCII characters. It is noted whether E or F conversion is used so entry points for these conversions will or will not be created.

CHECKF is a routine to syntax check a FORMAT statement. If it discovers an error in the format specifications CHECKF calls DIAG, the diagnostic routine.

#### INPUT

CHECKF uses the FORK routine to check fields in the FORMAT statement. The first field it receives should always be null with a ( terminator (the leftmost left parenthesis). The field preceding the last must always terminate with ); the last field must be null with an end of statement terminator. CHECKF assumes that each successive call to FORK picks up the next field scanning left to right, in the FORMAT statement.

#### OUTPUT

As the check proceeds FORK (using FGETC) converts the FORMAT statement to external code (BCD or ASCII) and places it into IBUF2 starting at IBUF2(6). The last character placed in IBUF2 (banning an error) will be the terminating right parenthesis. The FORTRAN statement placed in IBUF2 will not include extraneous blanks or the EOS character.

Upon exit IBUF2(X) will be at the number of words in IBUF2.

Subroutines used: FORK, FGETC, GETC, DIAG

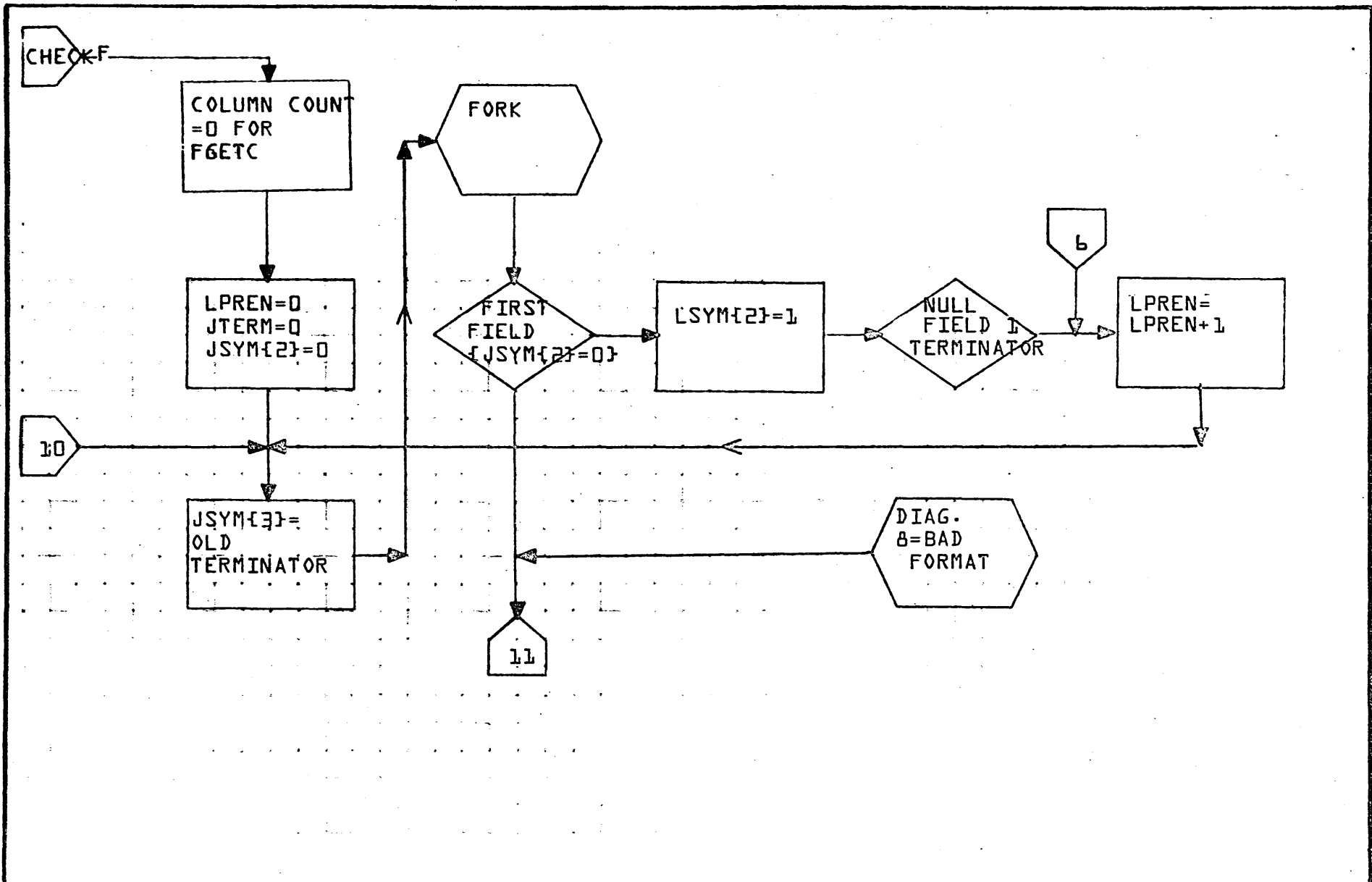


A

B

C

D



**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

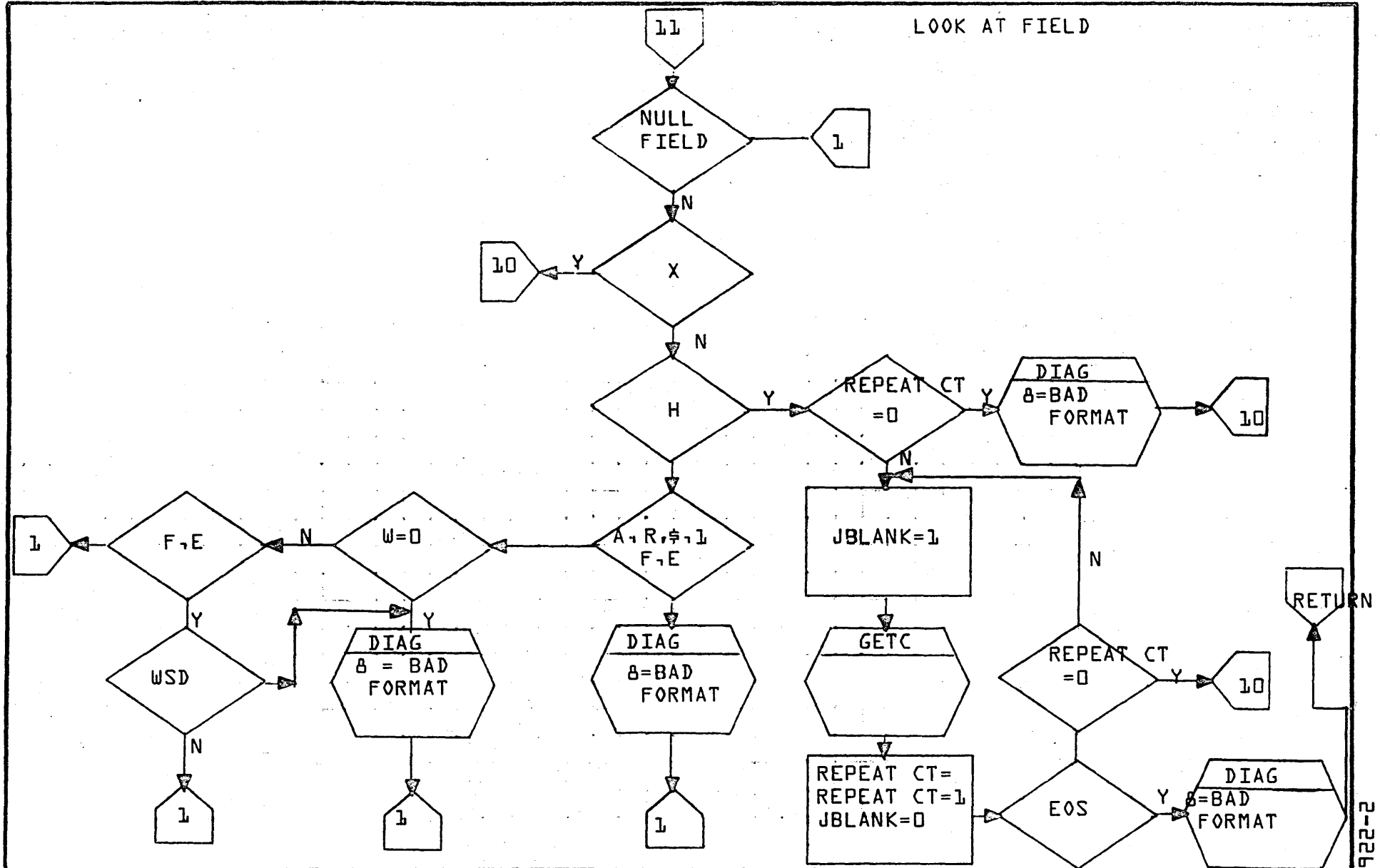
SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	CHECKF	PAGE 1 OF 3		PROJECT MGR.			
NUMBER	ISSUE DATE			PROJECT NAME			
DRAWN BY	DATE			TASK NO.			
				TASK NAME			

2-225

LOOK AT FIELD

A  
B  
C  
D



2-226

<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	CHECKF	PAGE 2 OF 3		PROJECT MGR.			
	NUMBER	ISSUE DATE			PROJECT NAME			
	DRAWN BY	DATE			TASK NO.			
					TASK NAME			





DOCUMENT CLASS IMS PAGE NO 2-228  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

2.4.7.2 Format Handling Routines - FORK

Syntax checks a format statement. Input parameters: holders for output.

Output parameters:

1. I1 field type

- 1 = null or repeat count only
- 2 = X
- 3 = H
- 4 = A
- 5 = R
- 6 = \$
- 7 = I
- 8 = f
- 9 = E
- 10 = Error

2. I2 field terminator

- 0 = none (X,H fields)
- 1 = (
- 2 = )
- 3 = ,
- 4 = /
- 13 = E05

3. I3 repeat count

4. I4 = W

5. I5 = D

Modes: 1 = initial  
2 = repeat  
3 = f, E           aux switch = 5,6  
4 = A,R,\$,I  
5 = error           aux switch = 0,1,2,3

Subroutines used: FGETC

FORK

IL=1 I2=0  
I3=0 I4=0  
I5=0  
IAUX=0 IDEC=0

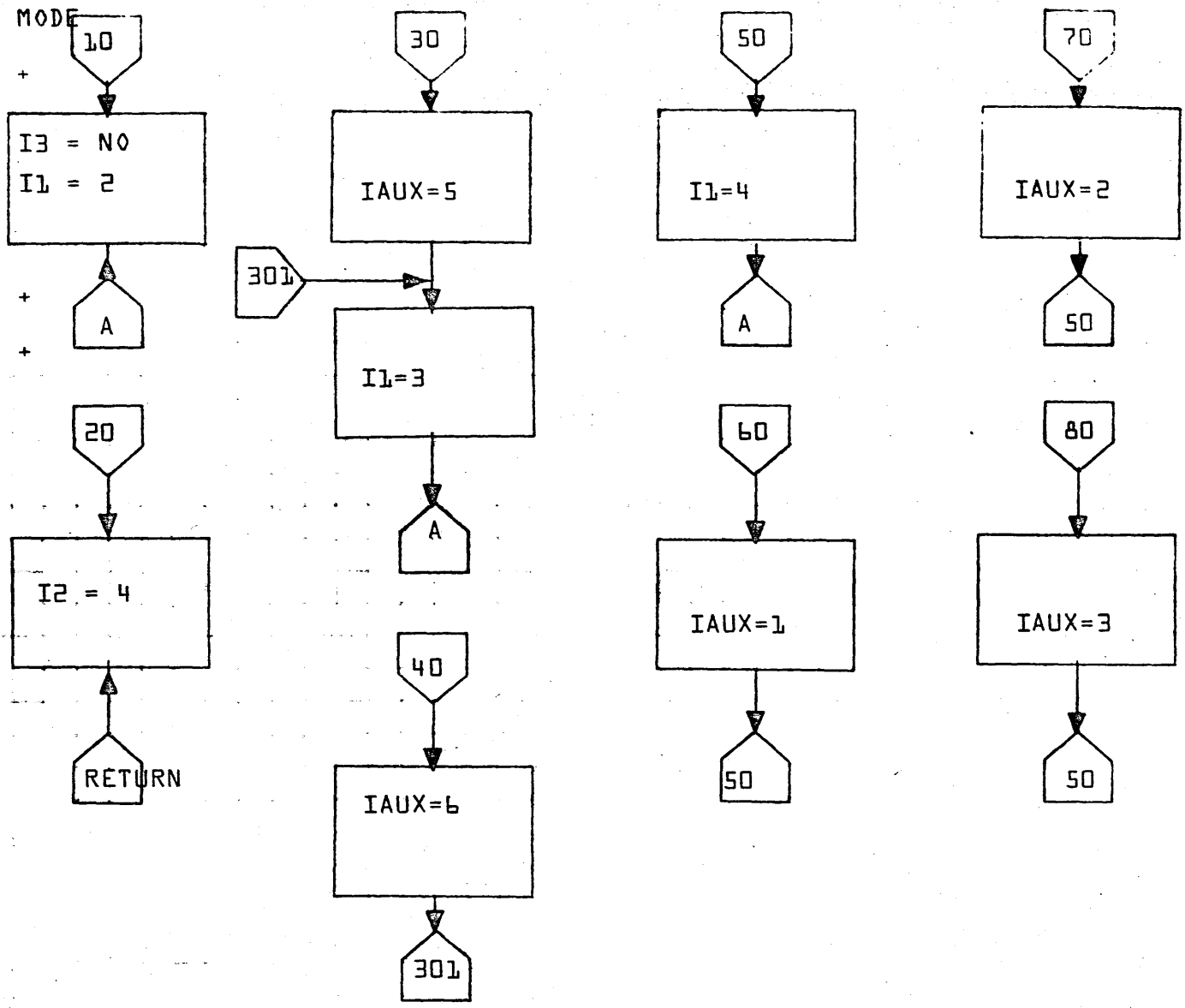
A

FGETC  
get next  
non-blank  
character

I=L5

0-9	EOS	F	E	A	R		\$	I	X	H	(	)	,	
10	20	30	40	50	60	INITIAL (IL=1)	70	80	90	100	120	130	140	H
150	160	30	40	50	60	REPEAT (IL=2)	70	80	90	100	120	130	180	190
200	220	120	120	120	120	F, E (IL=3)	120	120	120	120	260	270	280	290
240	230	120	120	120	120	A, R, \$, I (IL=4)	120	120	120	120	120	130	140	210
A	230	A	A	A	A	ERROR (IL=5)	A	A	A	A	A	130	140	210

- 10. NUMBER IN INITIAL MODE
- 20. / IN INITIAL MODE
- 30. A IN INITIAL MODE + REPEAT MODE
- 40. E IN INITIAL MODE + REPEAT MODE
- 50. A IN INITIAL MODE + REPEAT MODE.
- 60. R IN INITIAL MODE + REPEAT MODE
- 70. \$ IN INITIAL MODE + REPEAT MODE
- 80. I IN INITIAL MODE + REPEAT MODE



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	FORK		PAGE 2 OF 8	PROJECT MGR.			
	NUMBER		ISSUE DATE		PROJECT NAME			
	DRAWN BY		DATE		TASK NO.			
					TASK NAME			

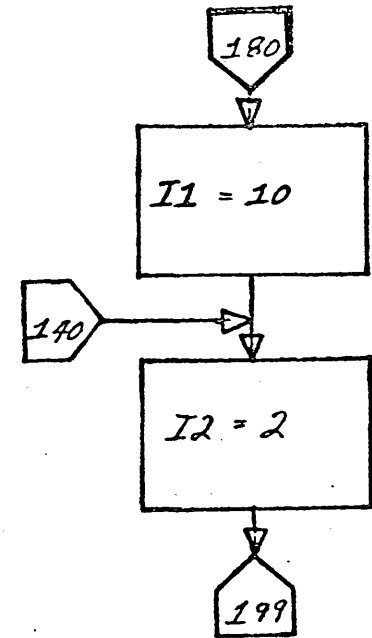
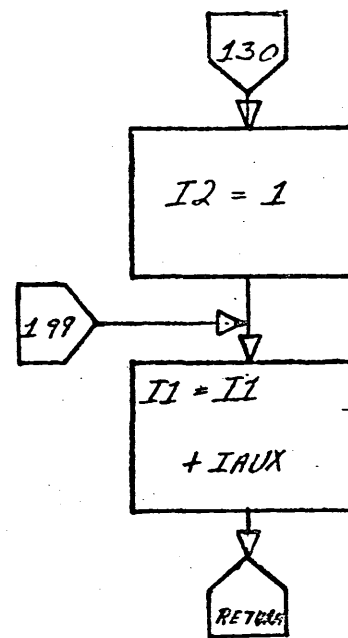
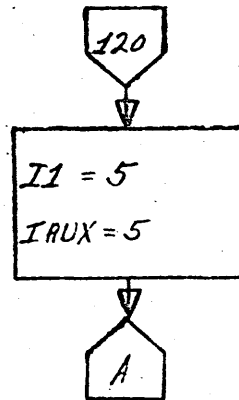
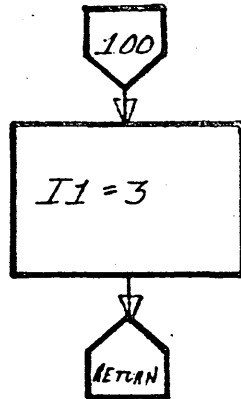
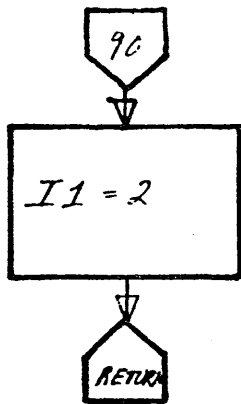
90. X IN INITIAL MODE +  
REPEAT MODE

100. H IN INITIAL MODE +  
REPEAT MODE

120. ERROR

130. ( ALL MODES EXCEPT  
F, E

180. ) REPEAT MODE  
ALL OTHERS EXCEPT F, E

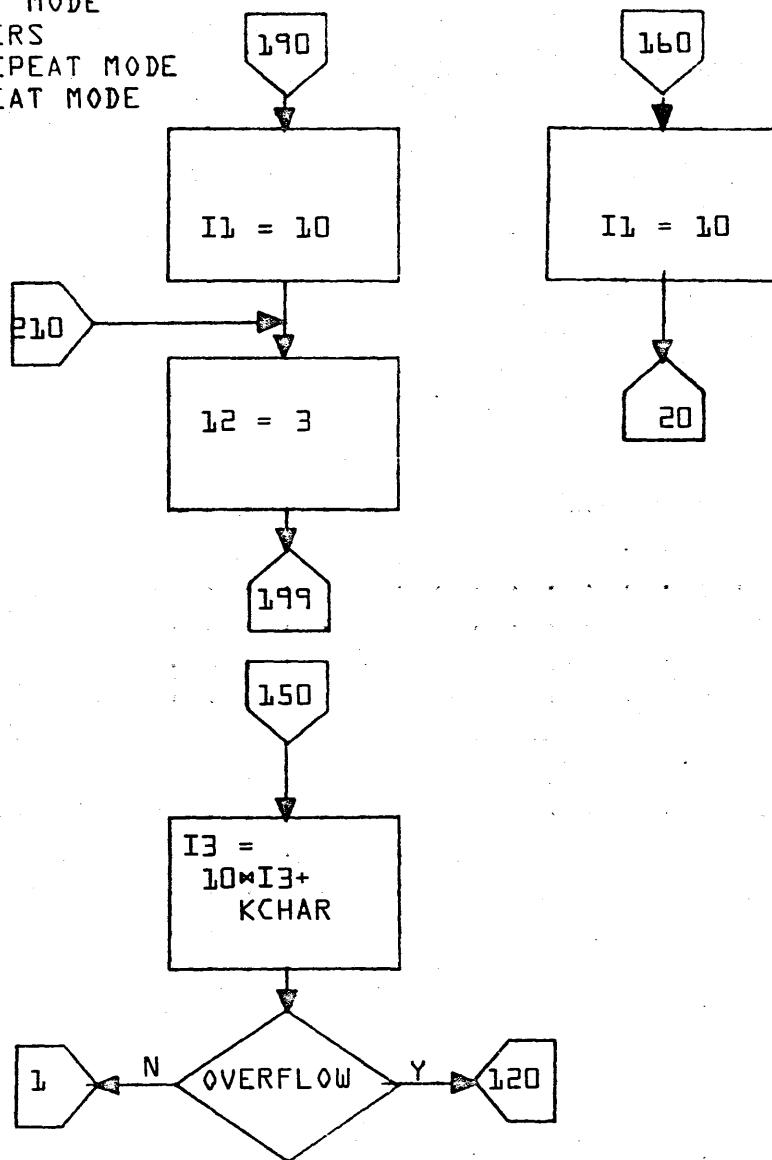


CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS <i>I MS</i>	MACH. TYPE <i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE <i>FORK</i>		PROJECT MGR			
	PAGE <i>3</i> OF <i>8</i>	PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.			
DRAWN BY	DATE <i>3-7</i>	TASK NAME			

190. } IN REPEAT MODE  
 } ALL OTHERS  
 150. 0-9 IN REPEAT MODE  
 160. / IN REPEAT MODE



CONTROL DATA CORPORATION  
 SOFTWARE DOCUMENT

SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

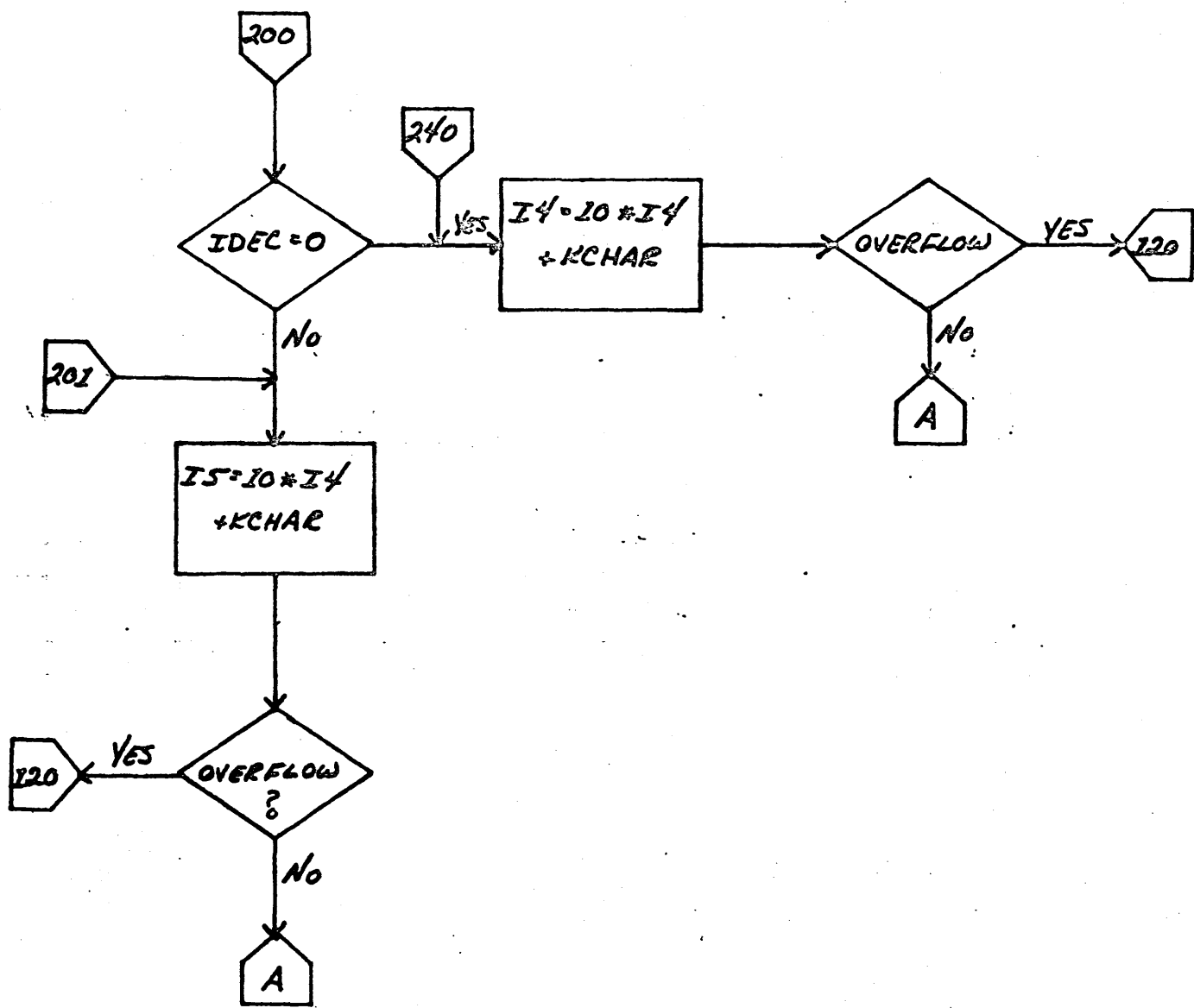
DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	FORK			PROJECT MGR.			
PAGE 4 of 8				PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.					
DRAWN BY	DATE	TASK NAME					

A

B

C

D



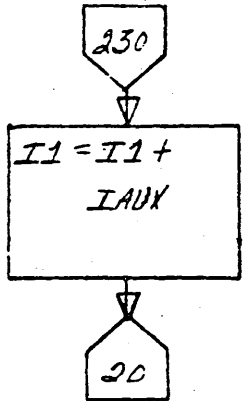
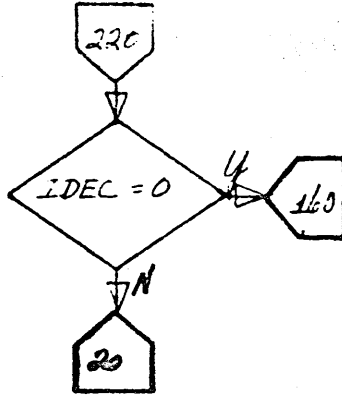
CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	<i>IMS</i>	MACH. TYPE	<i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	<i>FORK</i>	PAGE <i>5</i> OF <i>8</i>		PROJECT MGR.			
NUMBER	ISSUE DATE			PROJECT NAME			
DRAWN BY	DATE			TASK NO.			
				TASK NAME			

220. / IN F, E & EOS IN F, E

230. / IN A, R, S, I AND ERROR  
EOS in A, R, S, I, and ERROR



**CONTROL DATA CORPORATION**

SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	<i>I15</i>	MACH. TYPE	<i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	<i>FOR</i>			PROJECT MGR.			
		PAGE	<i>6 of 8</i>	PROJECT NAME			
NUMBER		ISSUE DATE		TASK NO.			
DRAWN BY		DATE	<i>3-67</i>	TASK NAME			

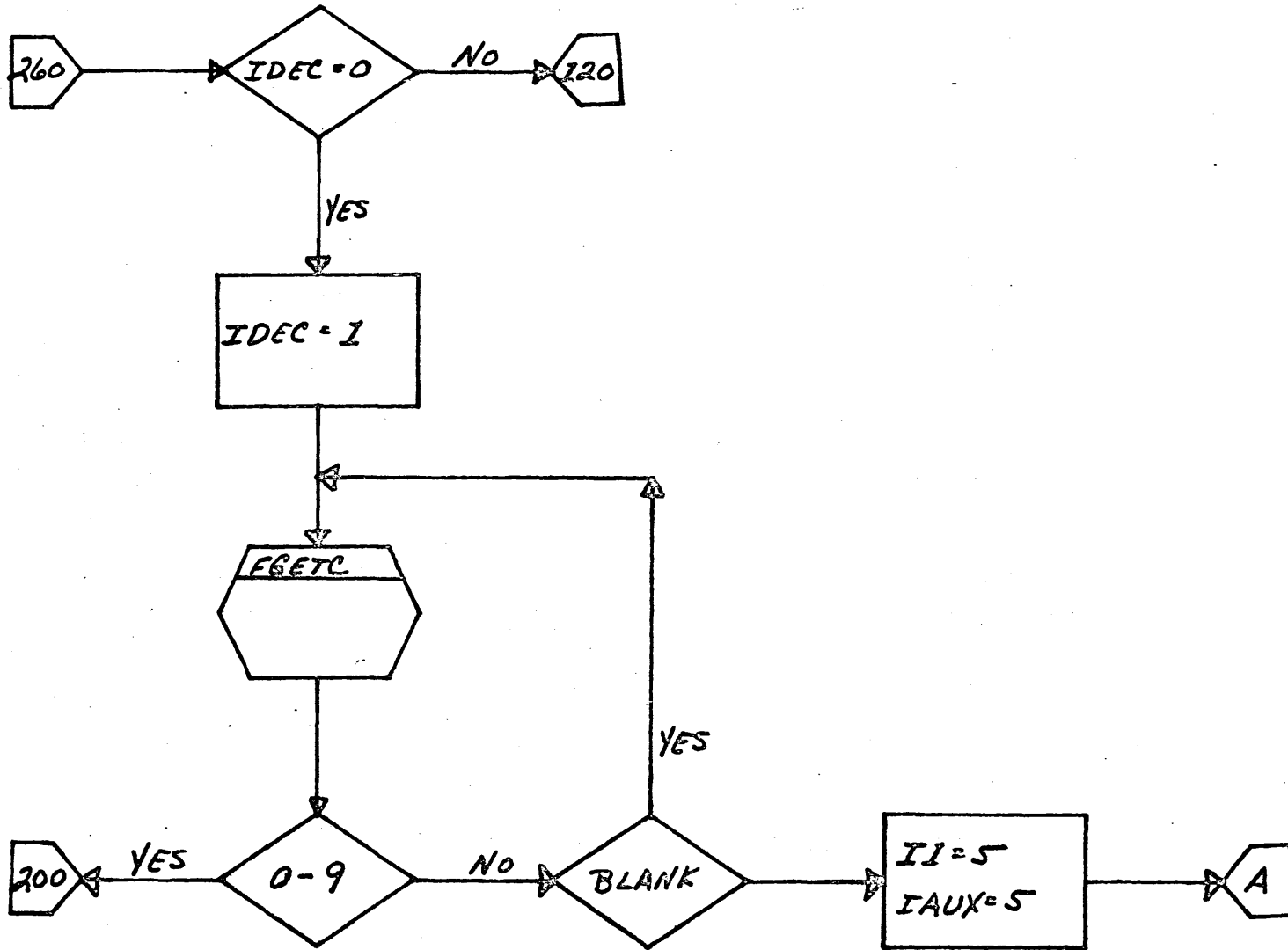


A

B

C

D

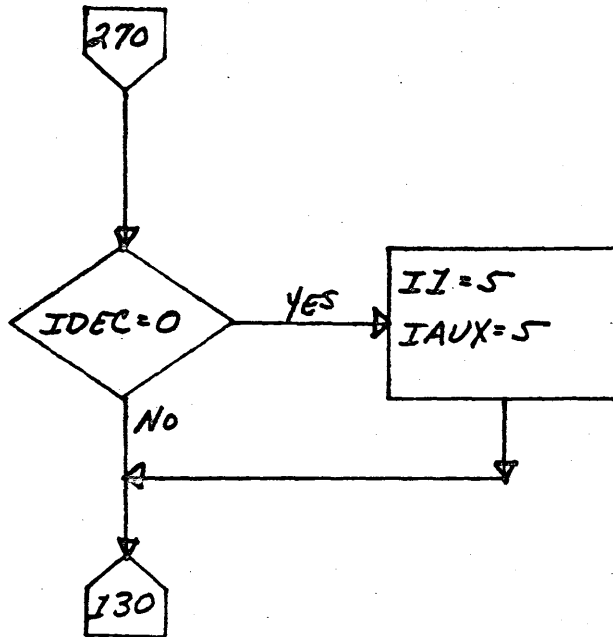


**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	<i>IMS</i>	MACH. TYPE	<i>1700</i>	PROJECT NO.		REV		APPROVED		DATE	
DOCUMENT TITLE	<i>FORK</i>	PAGE <i>7</i> OF <i>8</i>		PROJECT MGR.							
NUMBER		ISSUE DATE		PROJECT NAME							
DRAWN BY		DATE		TASK NO.							
				TASK NAME							

270. ( IN FILE



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

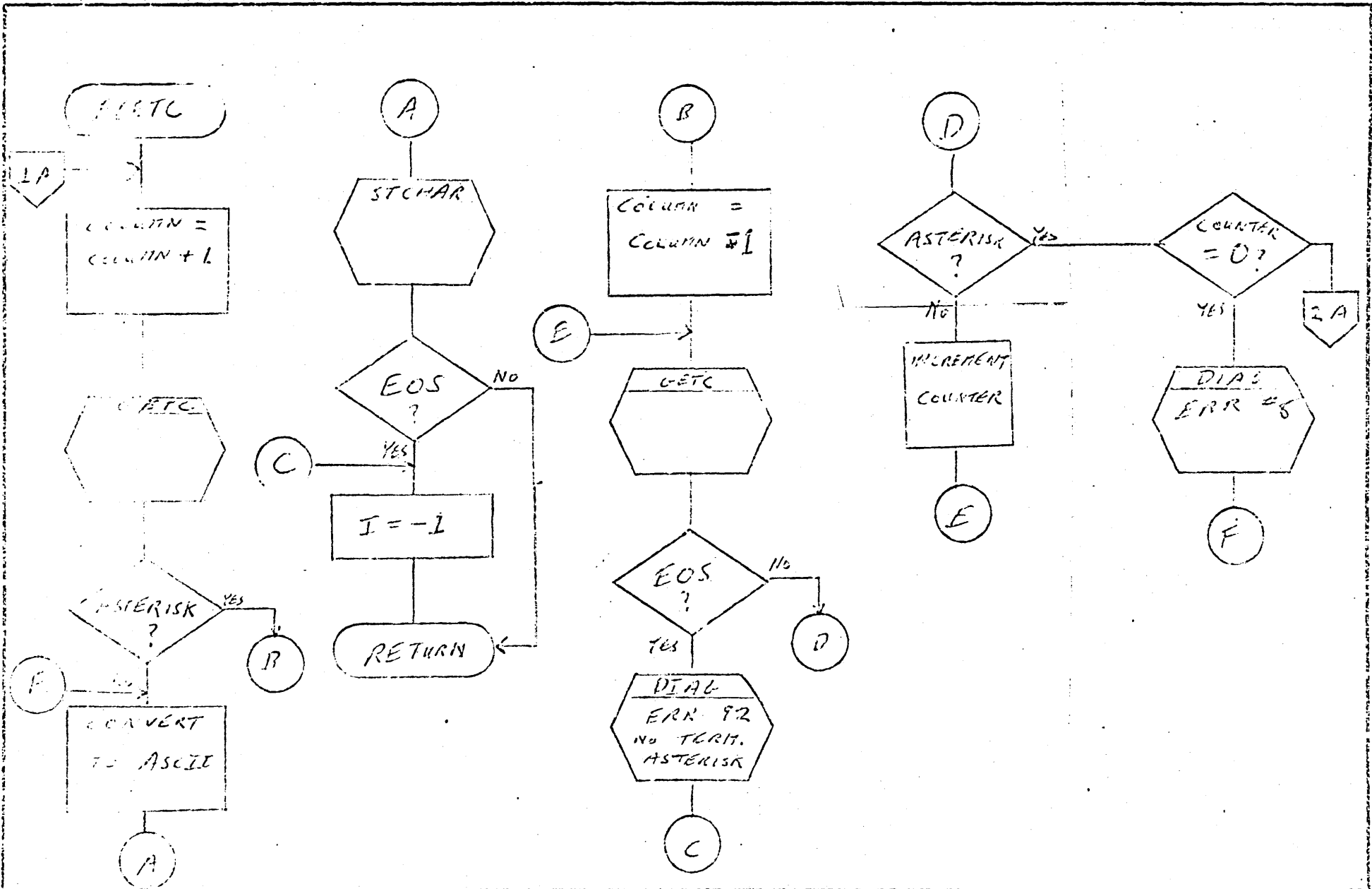
DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV.	APPROVED	DATE
DOCUMENT TITLE	FORK			PROJECT MGR.			
		PAGE	8 OF 8	PROJECT NAME			
NUMBER		ISSUE DATE		TASK NO.			
DRAWN BY		DATE		TASK NAME			

2-236

DOCUMENT CLASS IMS PAGE NO. 2-237  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005x3.0 A/B-E006x4.0 MACHINE SERIES 1700

### 2.4.7.3 Subroutine FGETC

Subroutine FGETC usually delivers one character from ISORS. The character is converted to ASCII and output. However, if the character is an asterisk indicating a 'free field' Hollerith string, FGETC counts the number of characters in the literal string and makes up an appropriate Hollerith declaration.

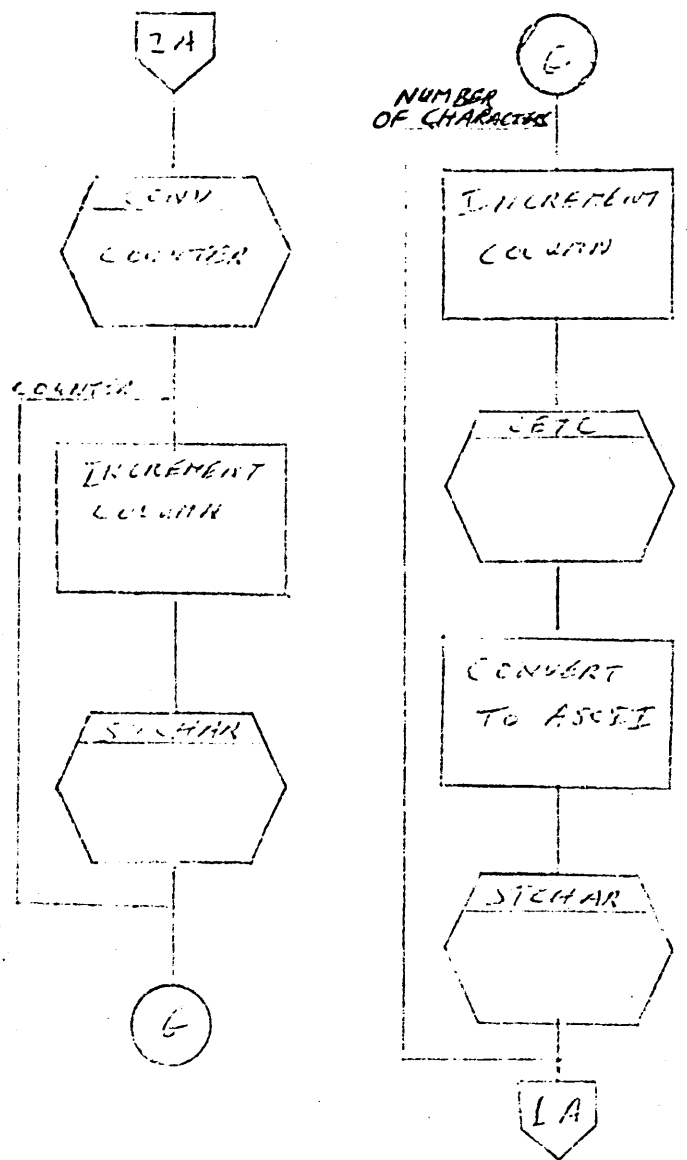


CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV.	APPROVED	DATE
DOCUMENT TITLE	FBETC	PAGE 1 OF 1		PROJECT MGR.			
NUMBER		ISSUE DATE		PROJECT NAME			
DRAWN BY		DATE		TASK NO.			
				TASK NAME			

A  
B  
C  
D



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	MM'S	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	FLETC			PROJECT MGR.			
	PAGE 1 OF 2				PROJECT NAME			
	NUMBER	ISSUE DATE		TASK NO.				
	DRAWN BY	DATE		TASK NAME				
	<div style="text-align: right; font-size: small;">1-2-58</div>							

C

C

C

DOCUMENT CLASS IMS PAGE NO. 7-239  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT NO. COO 5 VERSION 2.0 MACHINE SERIES 1700

#### 2.4.8 ASSEM Statements - Subroutine ASEMPR

The ASEMPR subroutine will generate intermediate language for each member of a parameter list in an ASSEM statement. The meaning of the intermediate language generated for a particular constant depends on the type of constant the ASEMPR subroutine encounters in the parameter list. The constants may be either numeric constants, address constants of statement labels.

##### 2.4.8.1 Output of Intermediate Language

The intermediate language is generated in the output buffer named IBUF2. The tally register used for the word length of the output entry is IBUF2X. IBUF2X is set initially prior to entering ASEMPR, and is increased by the number of words of intermediate language generated for a parameter when the parameter is processed. An i.l. entry generated for a parameter begins at IBUF2 (IBUF2X).

##### 2.4.8.2 Numeric Constants

A numeric constant must be an integer of the form:

\$hhhh,

implying a hexadecimal number. The GETF subroutine is called to extract the numeric constant from the source buffer. If, upon return from GETF, the operand is other than an integer, the flag JMODE will not contain a "3". An error diagnostic will be made followed by an exit from ASEMPR.

If the numeric constant is legal, the following 2 words of intermediate language will be generated in the output buffer:

WORD 1: (IBUF2(IBUF2X)) = 1; = the number which identifies the parameter as a numeric constant.

WORD 2: (IBUF2(IBUF2X+1)) = the value for the numeric constant.

##### 2.4.8.3 Address Constants

An address constant may be either an alphanumeric operand in which case it is a name or it may be a numeric operand in which case it is regarded as a statement label. The name or label is extracted from the source buffer by a call to GETF. If the name or label does not appear anywhere in the symbol table, a call is made to the store routine to read the operand and its class.

CONTROL DATA CORPORATION  
LA JOLLA RESOURCE CENTER DIVISION

DOCUMENT CLASS IMS PAGE NO. 2-240  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. COO5\*3.1 A/B MACHINE SERIES 1700

Either name or label is assumed to be used with the relative mode of addressing unless the operand is preceded by a "+". It is then regarded as to be used with the absolute mode of addressing. If either operand is to have an increment, the increment must follow the operand and be enclosed with parentheses. The increment must be an integer constant. The GETF subroutine is used to extract the increment from the source buffer. If, upon return from GETF, either -

1. {JMODE} ≠ 3 indicating that the constant is not an integer, or
2. {JTERM} ≠ 42 indicating that the increment is preceded by a left parenthesis but not followed by a right parenthesis, an error diagnostic will begin.

Upon return from GETF, the value of the increment is recorded in JSYM{L}: If {L} the switch IK is set non-zero indicating that the ASA option is selected, {2} the parameter is a name, and {3} the name does not appear in the symbol table typed as SINGLE, the value of the increment is doubled. This is to accomodate the fact that if the ASA option is selected, integer and real names require 2 storage cells each while SINGLE names require 1. In other words: If {IK} ≠ 0, and ISNGL {ISYMX} = 0,

{JSYM{L}}\*2 → JSYM{L}.

Triple size of increment if double precision element:

{JSYM{L}}\*3 → JSYM{L}

The following 2 words of intermediate language will be generated for an address constant using absolute addressing and having no increment:

WORD 1: {IBUF2{IBUF2X}} = 2

WORD 2: {IBUF2{IBUF2X+1}} = pointer to name or label  
in symbol table.

The following 3 words of intermediate language will be generated for an address constant using the absolute mode of addressing and having an increment:

WORD 1: {IBUF2{IBUF2X}} = 4

WORD 2: {IBUF2{IBUF2X+1}} = pointer to name or label  
in symbol table.



CONTROL DATA CORPORATION  
LA JOLLA RESOURCE CENTER DIVISION

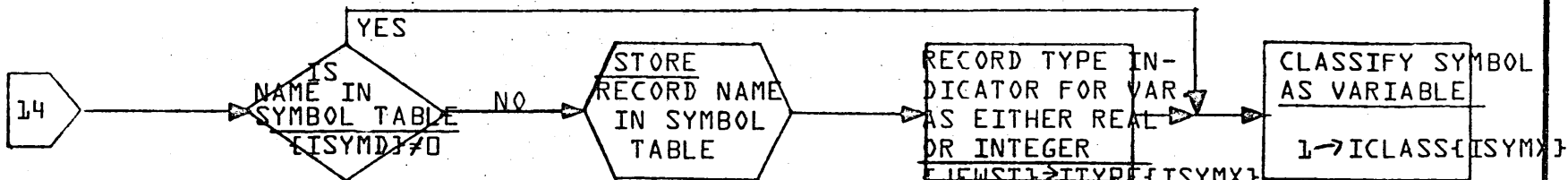
DOCUMENT CLASS IMS PAGE NO. 2-240A  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

WORD 3: {IBUF2{IBUF2X+2}} = value for increment re-  
corded in JSYM{1}.

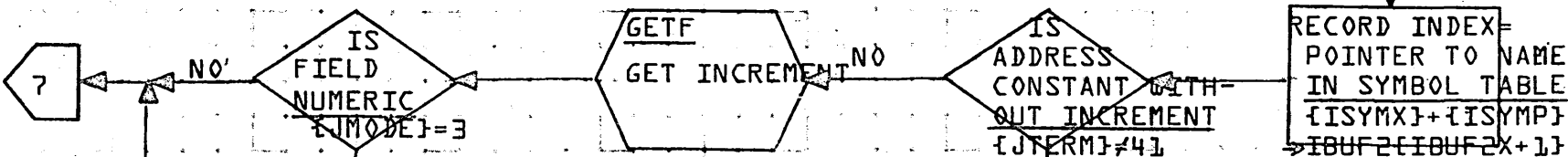
The following 3 words of intermediate language will be  
generated for an address constant using the relative  
mode of addressing and having an increment:



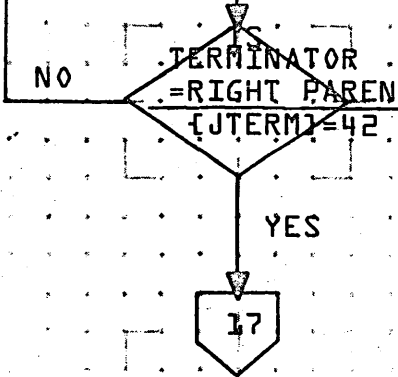
A



B



C



D

**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

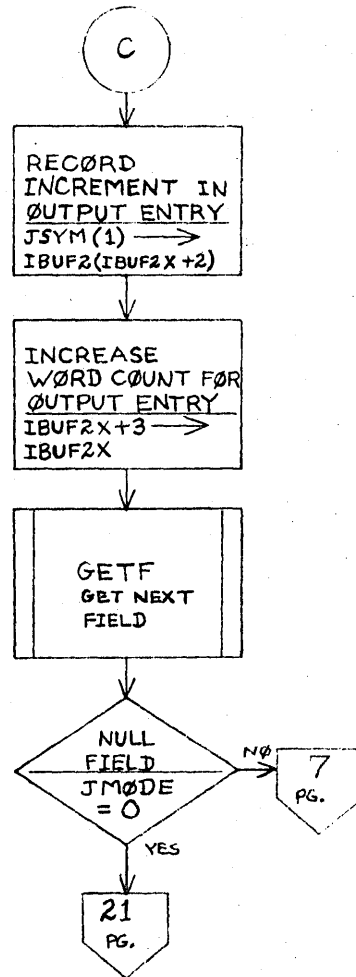
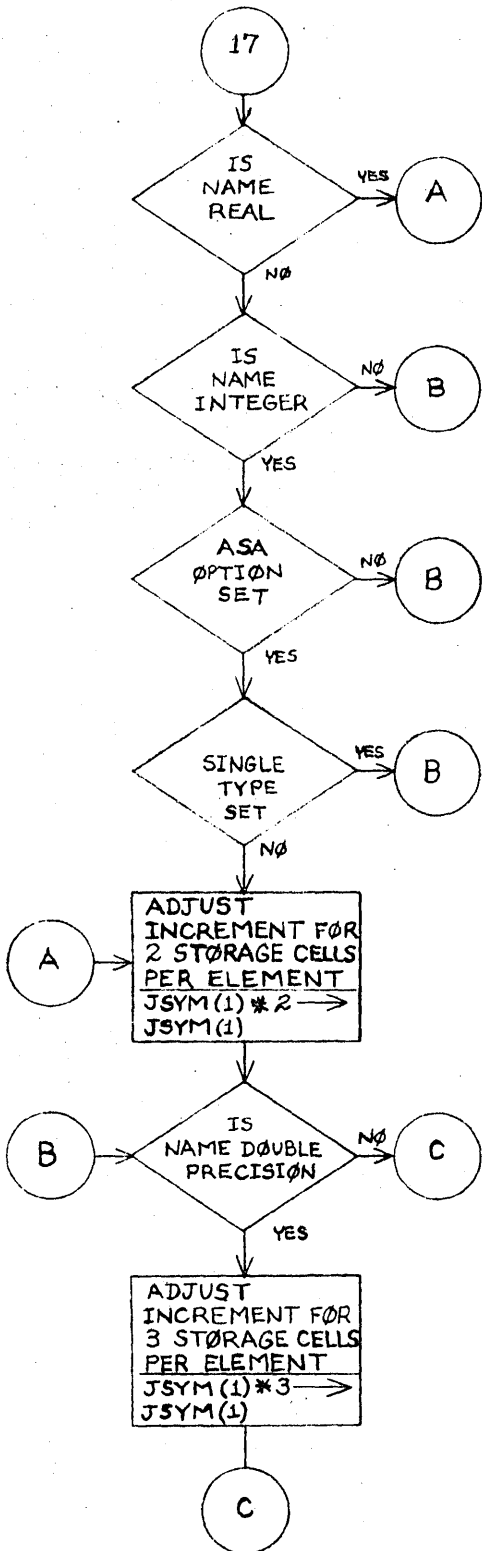
SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

DOCUMENT CLASS TMS	MACH. TYPE 1,200	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE ASEMPR	PAGE 2 OF 8	PROJECT MGR.			
NUMBER	ISSUE DATE	PROJECT NAME			
DRAWN BY	DATE	TASK NO.			
		TASK NAME			

2-243



LA JOLLA FACILITY



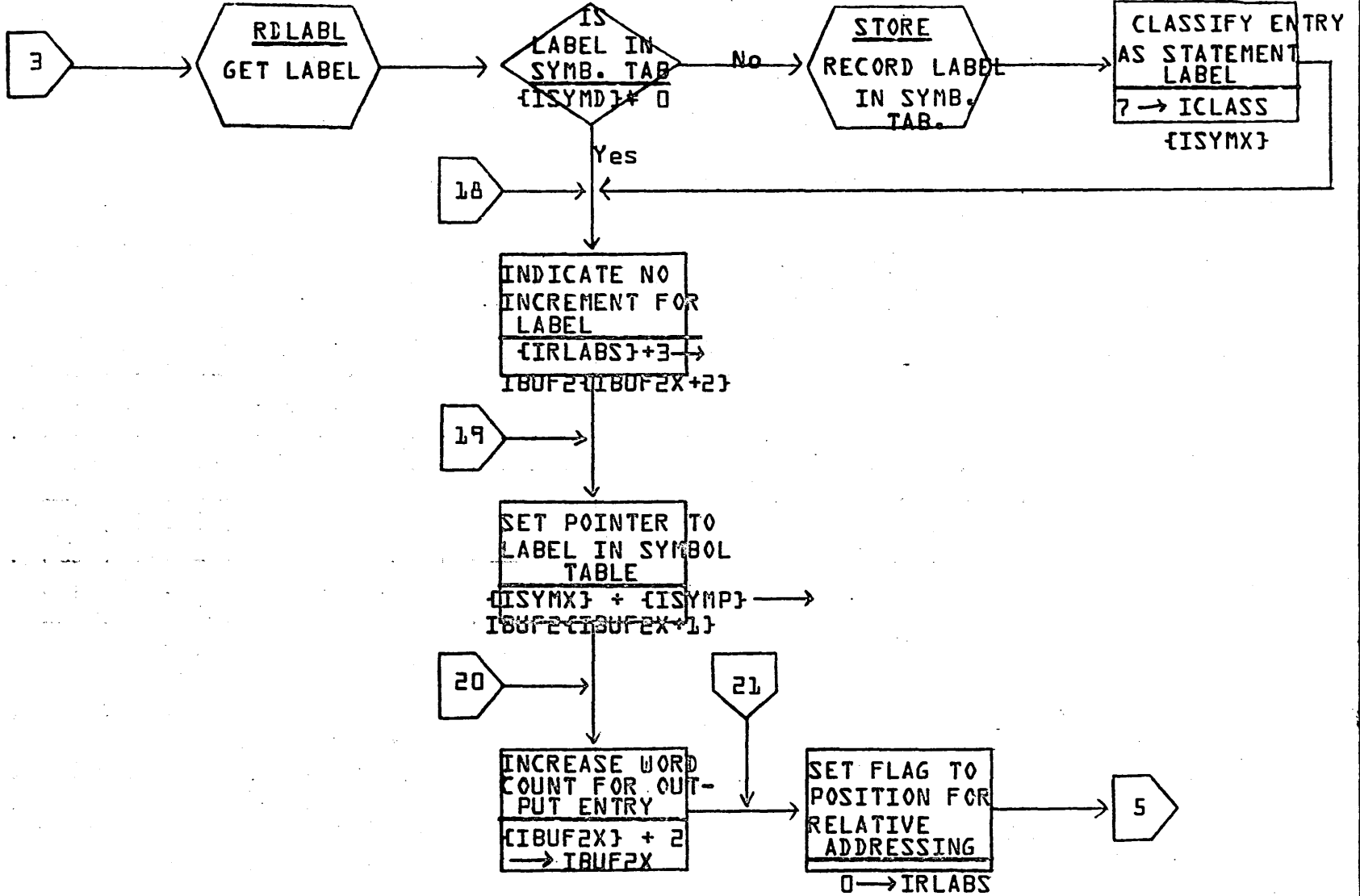
TITLE			DRG. NO.		
ASEMPR			REVISION		
DRAWN BY	PROJ.	DATE	SHEET	3	OF 8

A

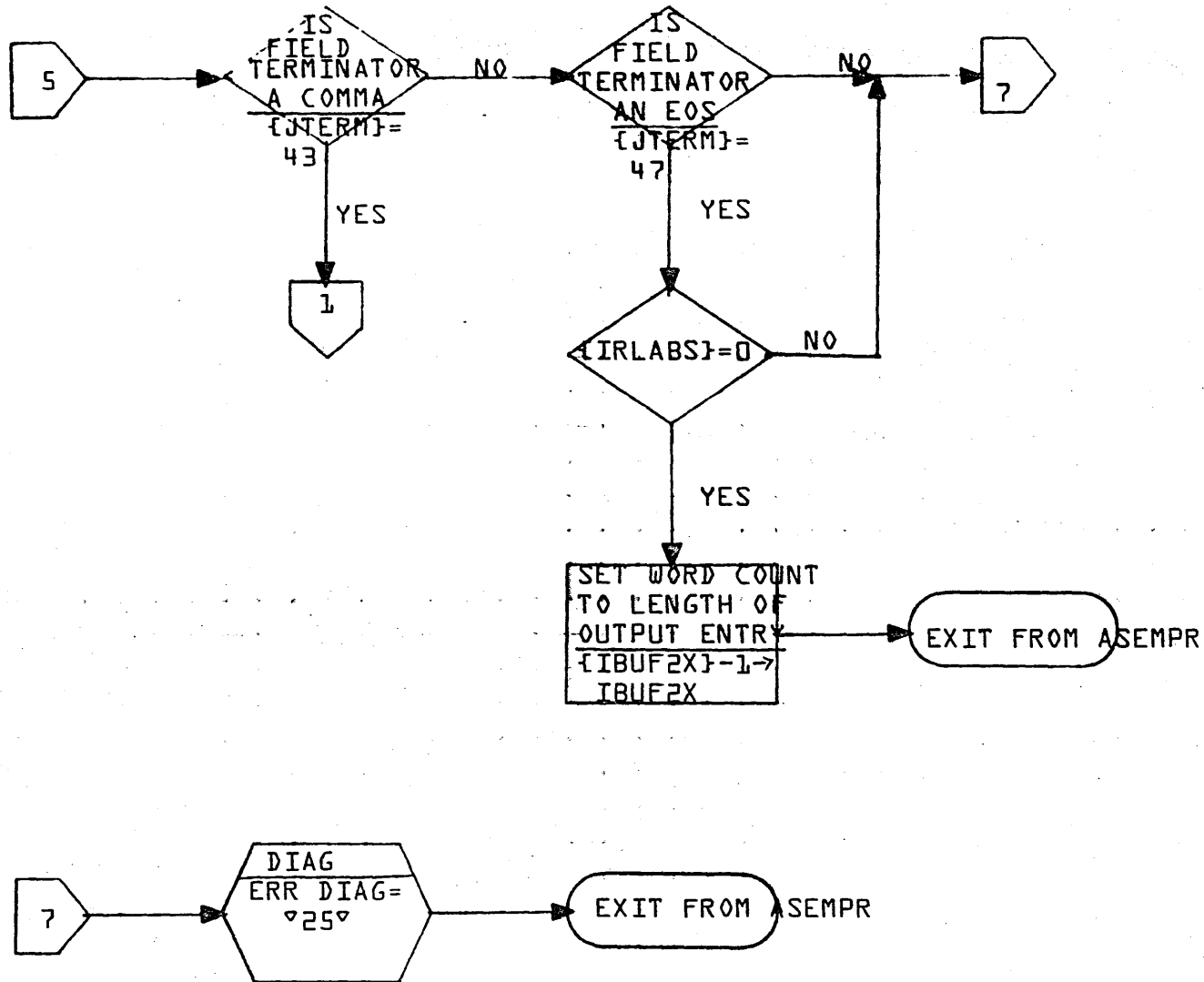
B

C

D



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	ASEMPR			PROJECT MGR.			
	PAGE 4 OF 8				PROJECT NAME			
	NUMBER	ISSUE DATE		TASK NO.				
	DRAWN BY	DATE		TASK NAME				



**CONTROL DATA CORPORATION**

SOFTWARE DOCUMENT

SAMPLE CODE

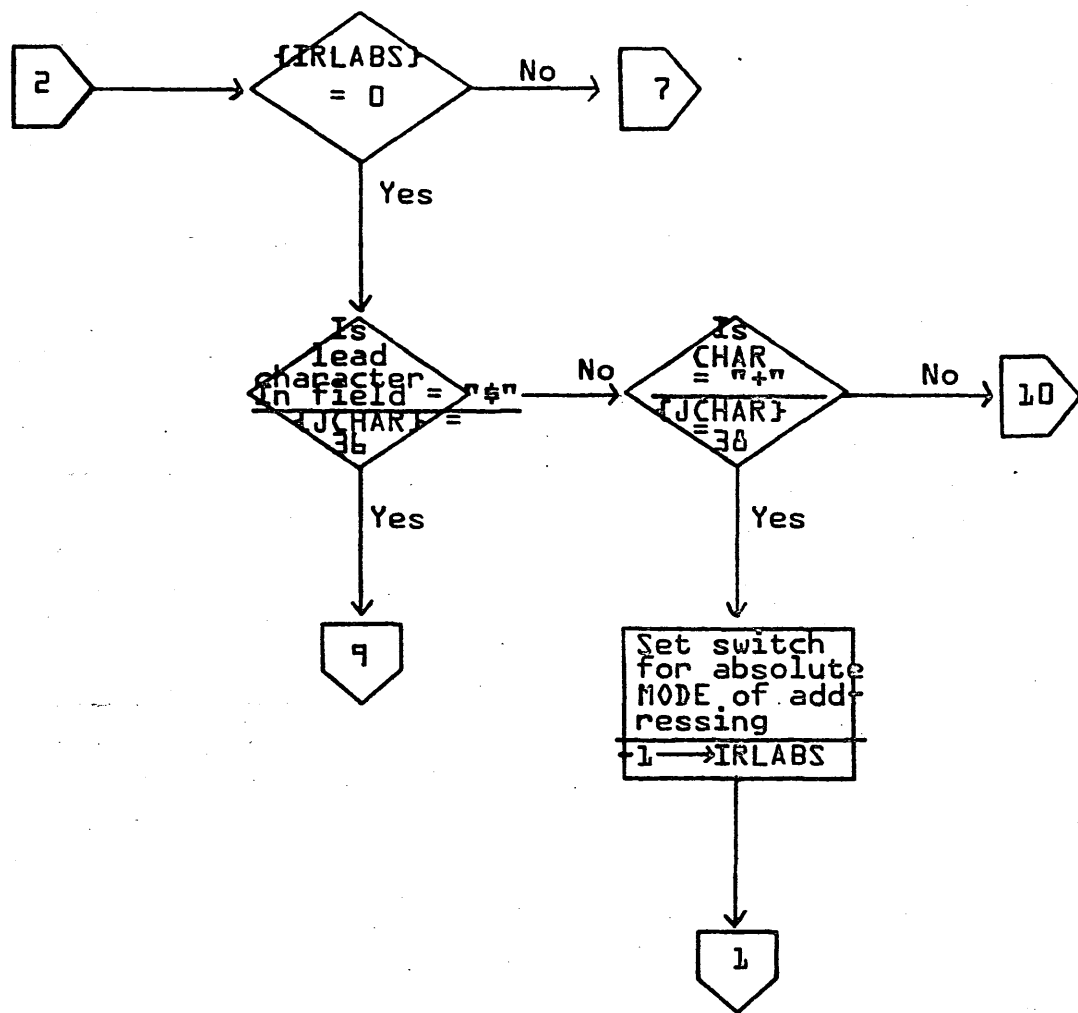
FLOWCHART

DECISION TABLE

OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1,200	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	ASEMPR	PAGE 5 OF 8		PROJECT MGR.			
NUMBER	ISSUE DATE			PROJECT NAME			
DRAWN BY	DATE			TASK NO.			
				TASK NAME			

A  
B  
C  
D



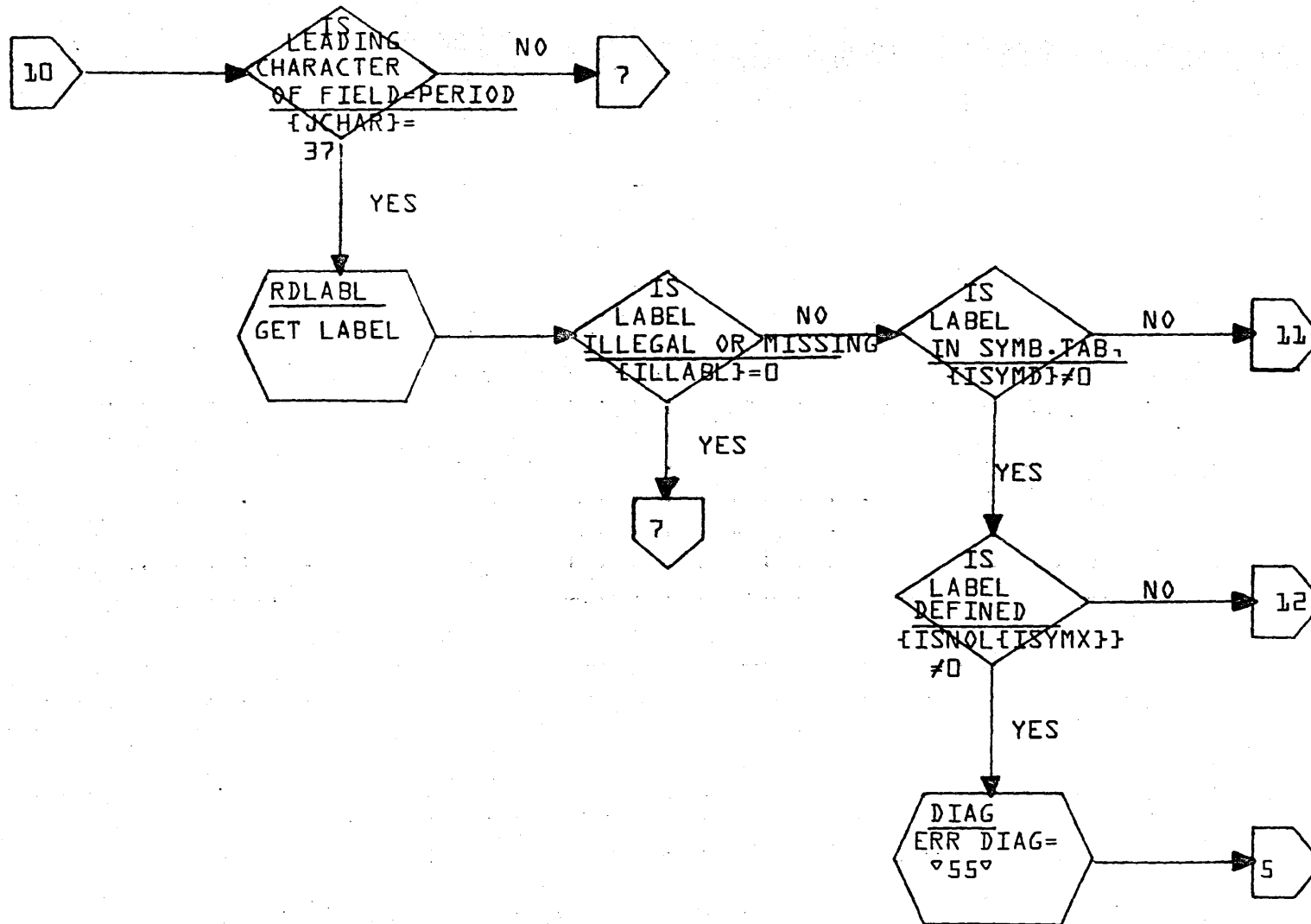
**CONTROL DATA CORPORATION**  
 SOFTWARE DOCUMENT

SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700
DOCUMENT TITLE	ASEMPR		
NUMBER		ISSUE DATE	
DRAWN BY		DATE	

PROJECT NO.	REV	APPROVED	DATE
PROJECT MGR.			
PROJECT NAME			
TASK NO.			
TASK NAME			

2-247



**CONTROL DATA CORPORATION**

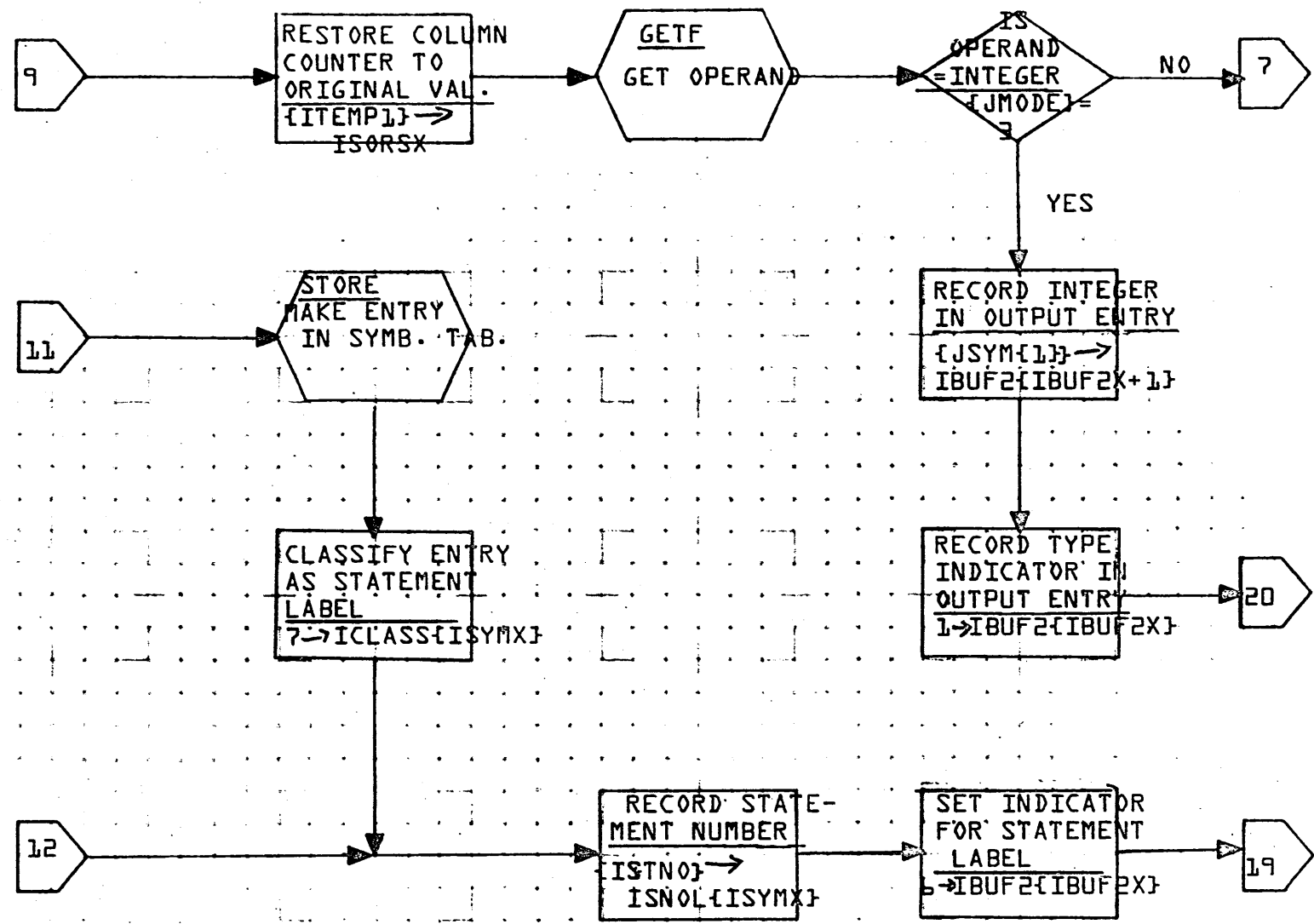
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	ASEMPR			PROJECT MGR.			
		PAGE	7	PROJECT NAME			
		OF	8	TASK NO.			
NUMBER		ISSUE DATE		TASK NAME			
DRAWN BY		DATE					



A  
B  
C  
D



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS IMS	MACH. TYPE 1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE ASEMPR	PAGE 8 OF 8	PROJECT MGR.			
NUMBER	ISSUE DATE	PROJECT NAME			
DRAWN BY	DATE	TASK NO.			
		TASK NAME			

DOCUMENT CLASS IMS PAGE NO 2-250  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

#### 2.4.9 ASSIGN Statement - Subroutine ASGNPR

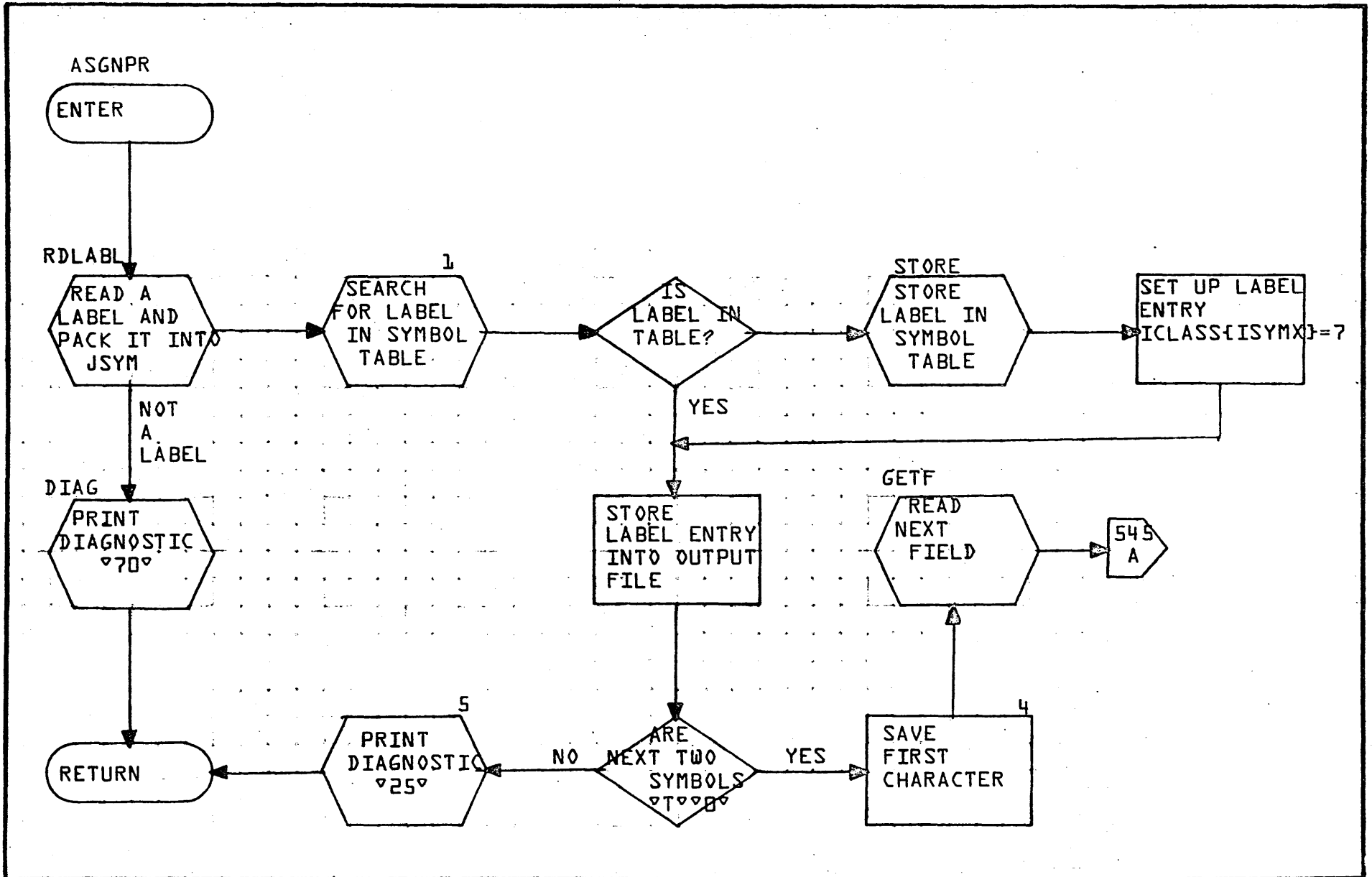
ASGNPR is called directly by PHASEA to process ASSIGN statements. It reads the next field in ISORS using RDLABL and ascertains that it is a label. ASGNPR locates the label in the symbol table and places this symbol table pointer into the output buffer. The next two characters of ISORS are checked for T, O - if they are not, a diagnostic is printed. ASGNPR reads the next field and checks that it is an integer variable name. If so, the name is located in the symbol table and its symbol table pointer is placed in the output buffer. The number of words in the record is put into the first word of the output record and ASGNPR exits to PHASEA.

A

B

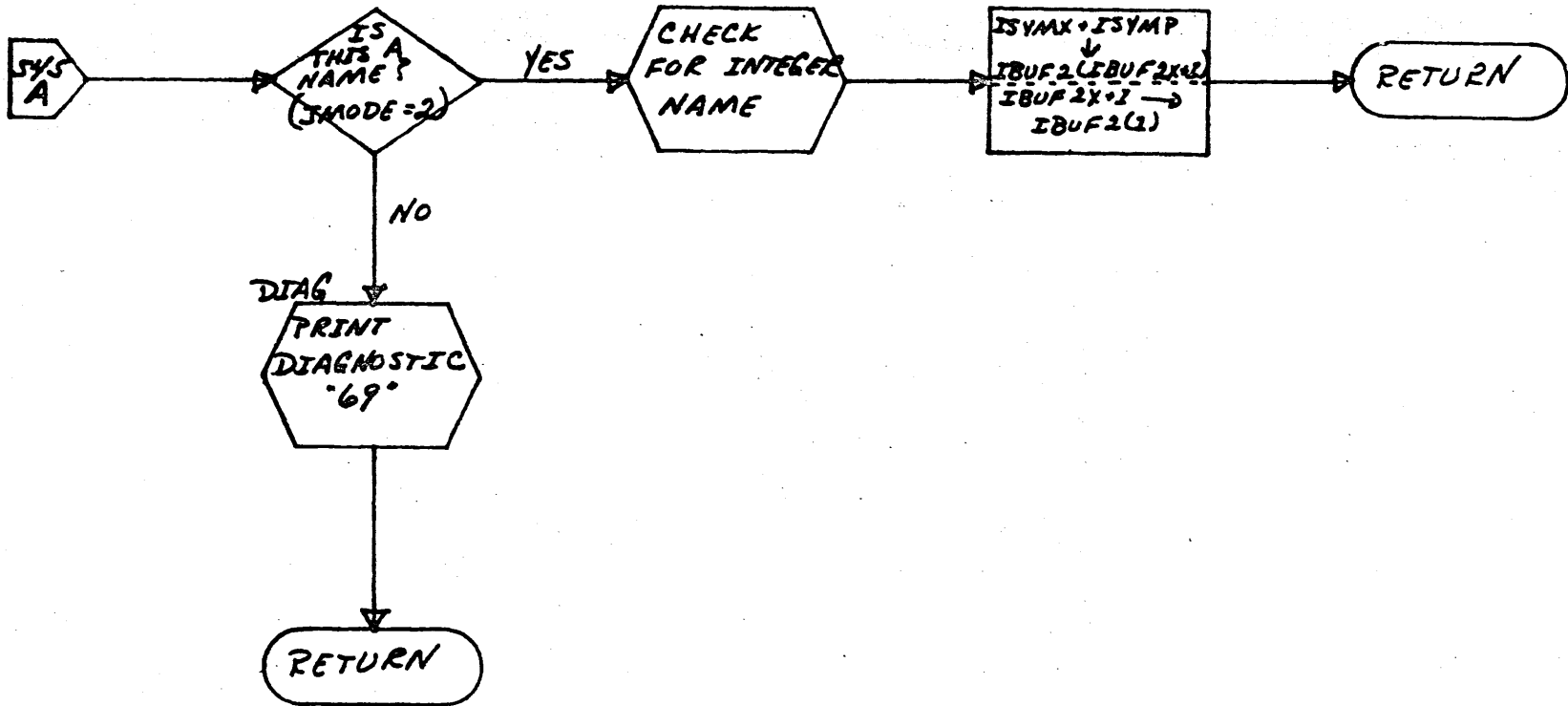
C

D



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	ASGNPR	PAGE 1 OF 2		PROJECT MGR.			
	NUMBER	ISSUE DATE			PROJECT NAME			
	DRAWN BY	DATE			TASK NO.			
					TASK NAME			

2-251



**CONTROL DATA CORPORATION**

SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	<b>ZMS</b>	MACH. TYPE	<b>1700</b>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	<b>ASGNPR</b>	PAGE <b>2</b> OF <b>2</b>		PROJECT MGR.			
NUMBER		ISSUE DATE		TASK NO.			
DRAWN BY		DATE		TASK NAME			

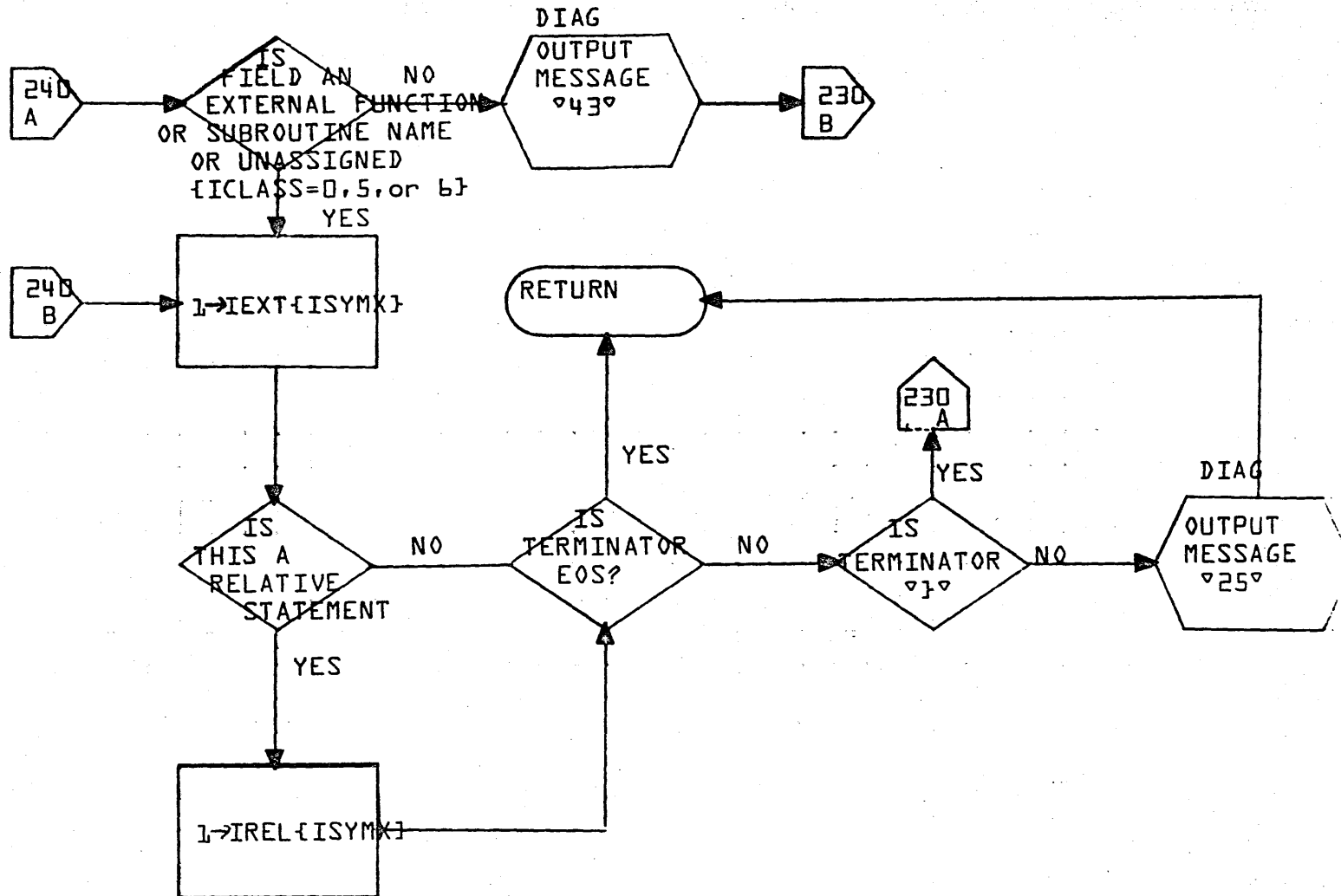
2-252

DOCUMENT CLASS IMS PAGE NO. 2-253  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

#### 2.4.10 EXTERNAL and RELATIVE Statements - Subroutine EXRLPR

EXRLPR is called by PHASEA<sup>4</sup> to process EXTERNAL and RELATIVE statements. It scans ISORS, checking the legality of each name it picks up, putting the name in the symbol table, and checking whether it has previously been declared as other than an external name. If it has been declared as other than an external, an error is output. If not, IEXT in the symbol table is set to one and, if EXRLPR is processing a relative statement, IREL is also set to one. Upon encountering the end of the statement, EXRLPR exits to PHASEA .

EXRLPR



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE   
FLOWCHART   
DECISION TABLE   
OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	EXRLPR	PAGE 2 OF 2		PROJECT MGR.			
NUMBER	ISSUE DATE	TASK NO.					
DRAWN BY	DATE	TASK NAME					

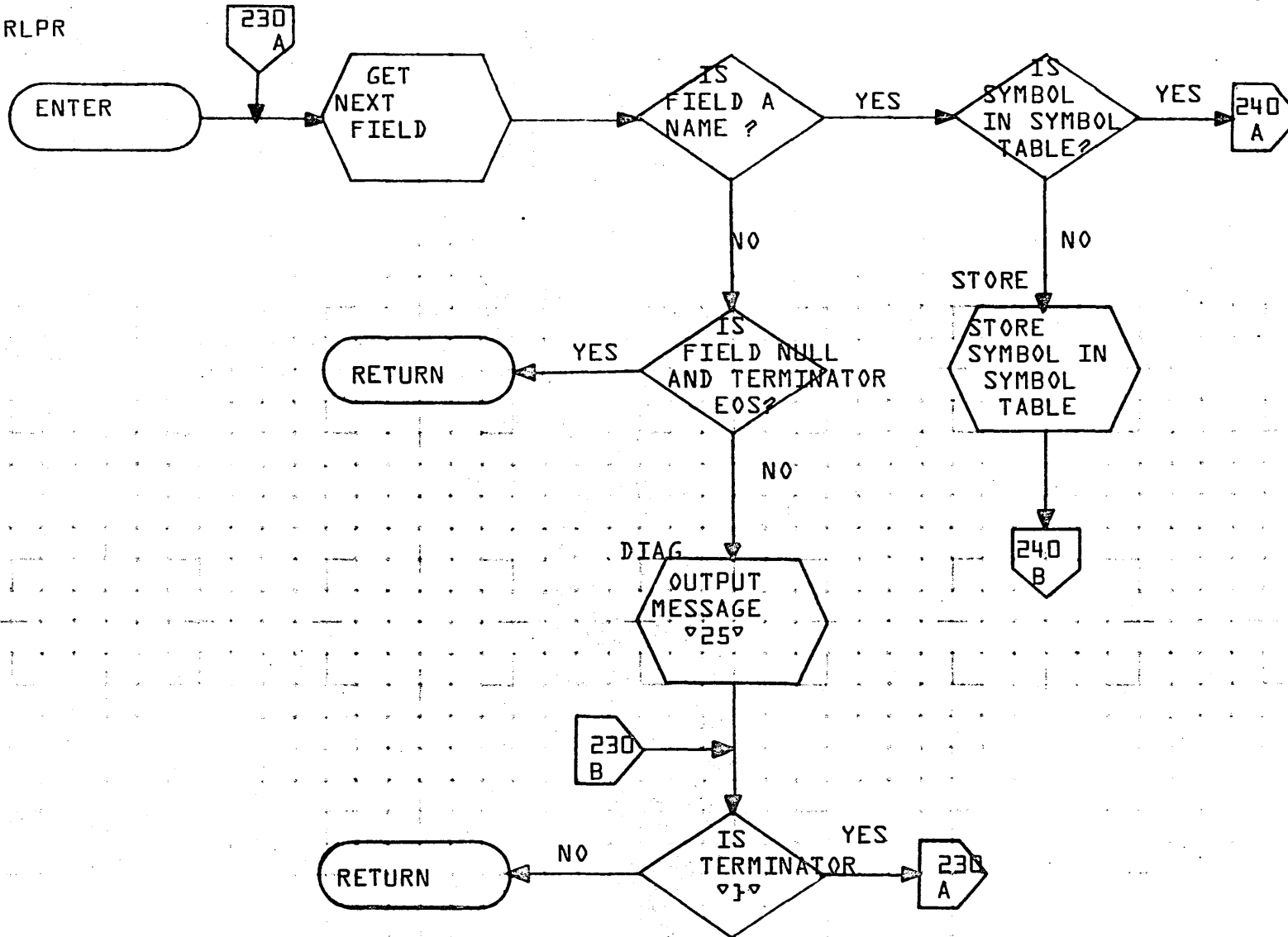
EXRLPR

A

B

C

D



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	EXRLPR	PAGE 1 OF 2		PROJECT MGR.			
NUMBER	ISSUE DATE			PROJECT NAME			
DRAWN BY	DATE			TASK NO.			
				TASK NAME			

## CONTROL DATA CORPORATION

DIVISION

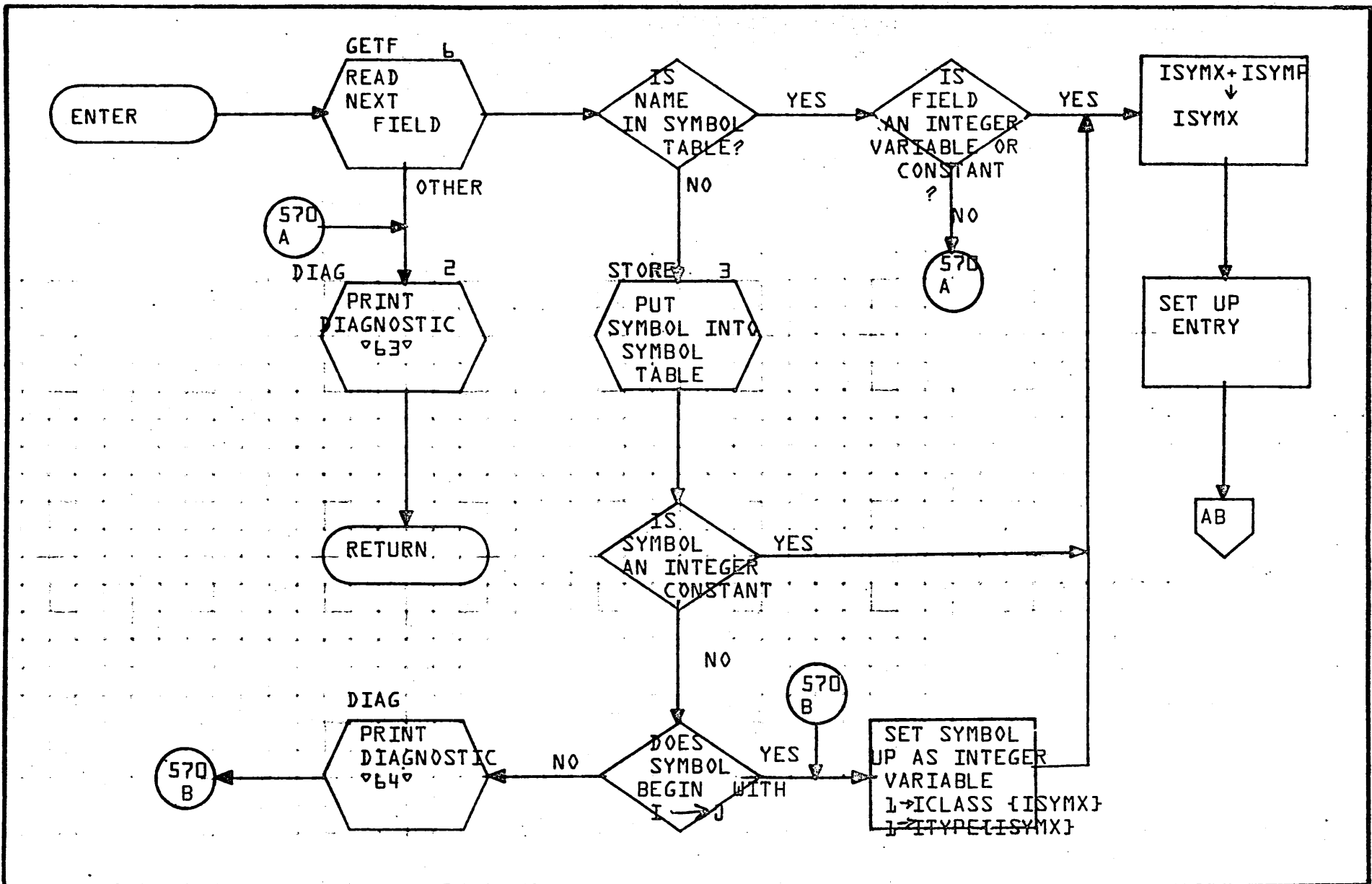
DOCUMENT CLASS IMS PAGE NO. 2-256  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C075 3.0A/B -E006 4.0 MACHINE SERIES 1700

## 2.4.11 REWIND, ENDFILE, BACKSPACE Statements - Subroutine ERBPR

ERBPR is called directly by PHASEA to process REWIND, BACKSPACE, or ENDFILE statements. It reads the next field of ISORS and verifies that it is an integer constant or variable name. After verification ERBPR sets up the output entry and return.

In the 3.0A compiler the length of ERBPR is crucial. Any increase in ERPBR must be accompanied by a corresponding decrease in the BSS buffer at the end of DUMYAB. {See page 2-9}

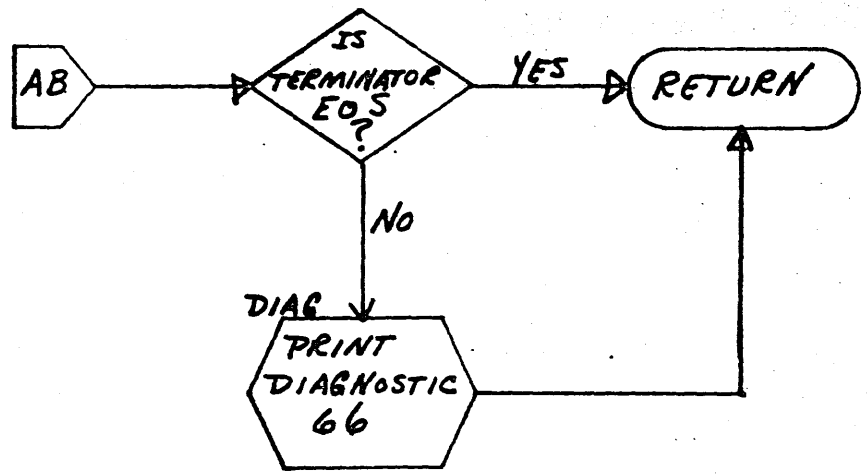




<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	ERBPR	PAGE 1 OF 2		PROJECT MGR.			
	NUMBER	ISSUE DATE			PROJECT NAME			
	DRAWN BY		DATE		TASK NO.			
					TASK NAME			

2-257

A  
B  
C  
D



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	ERBPR		PAGE 2 OF 2				
	NUMBER	ISSUE DATE	TASK NO.					
	DRAWN BY	DATE	TASK NAME					

2-258

DOCUMENT CLASS IMS PAGE NO. 2-259  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

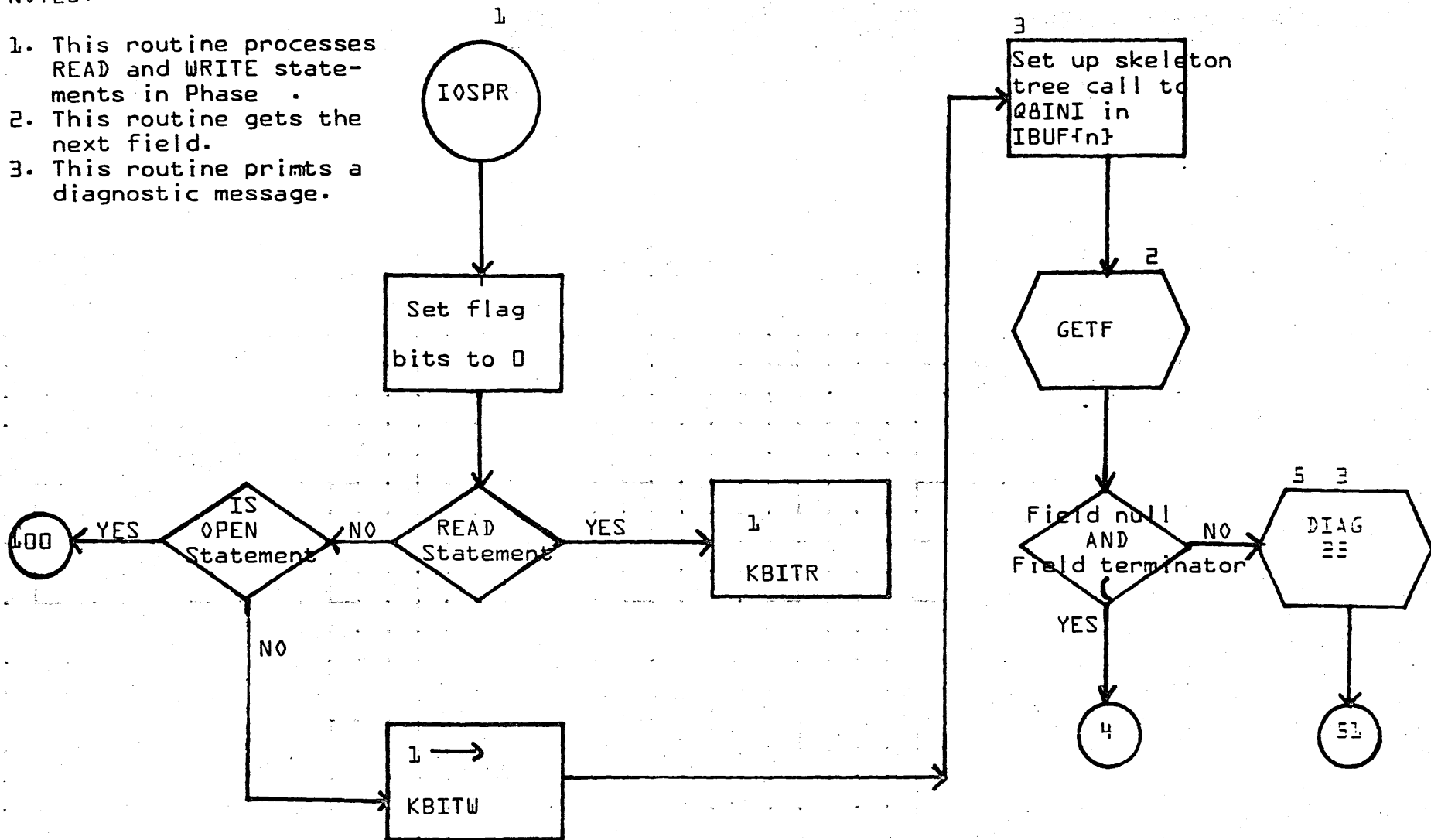
#### 2.4.12 IOSPR - Tree Builder for I/O Statements

This routine is called by PHASE<sup>A</sup> upon encountering a READ or WRITE. It builds the trees for the calls to Q8QINI, Q8QX, and Q8QEND, as required in IBUF2. Calls to Q8QX are built as new source statements which will be processed by ARITH. Other calls are built in the form of ARITH output and will be processed next by PHASEB

For a READ, WRITE statement the first tree generated is for Q8QINI. If there is a list, a Q8QX call is generated for each element, and a call to Q8QEND is generated after the last element.

NOTES:

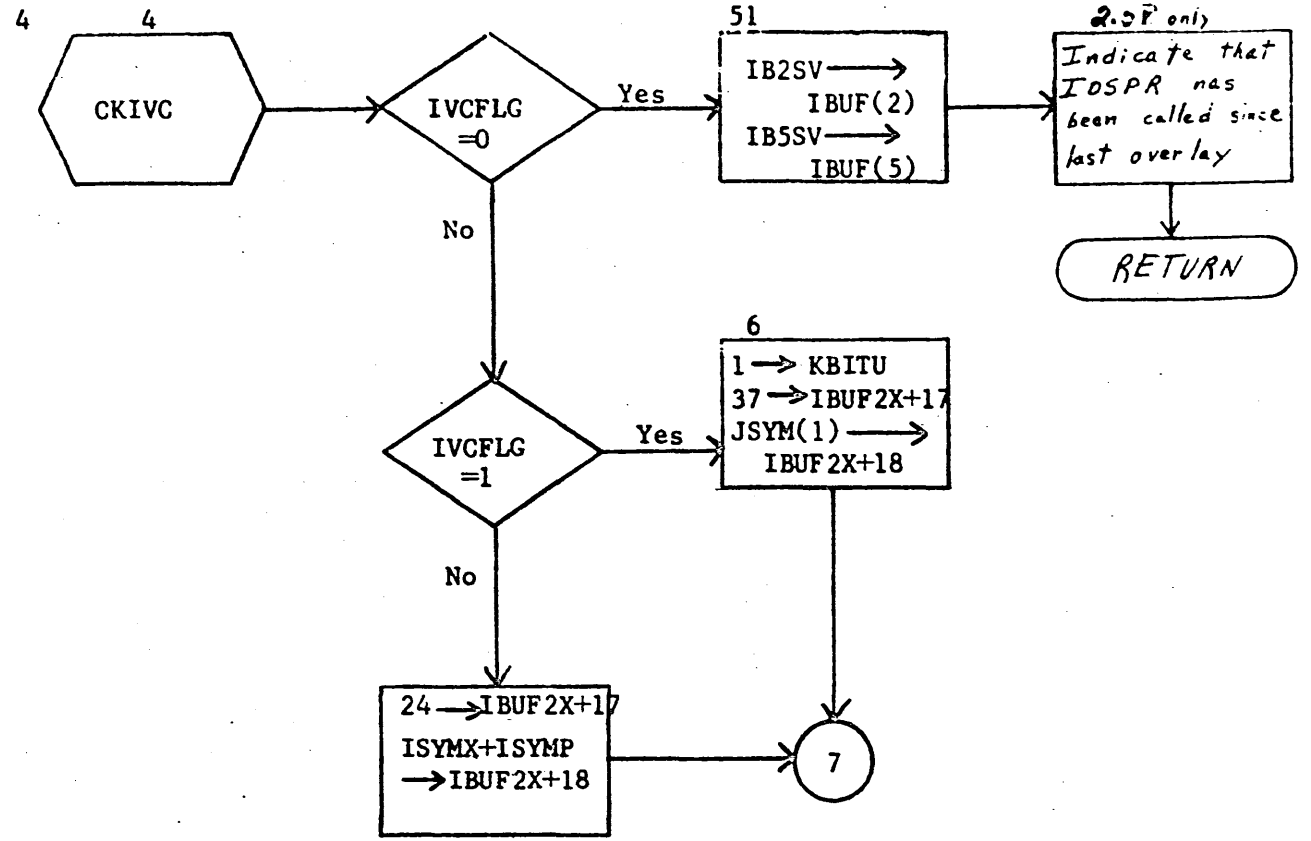
1. This routine processes READ and WRITE statements in Phase .
2. This routine gets the next field.
3. This routine prints a diagnostic message.



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	<i>IMS</i>	MACH. TYPE	<i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	<i>IOSPR</i>	PAGE <i>1</i> OF <i>16</i>		PROJECT MGR.			
	NUMBER	ISSUE DATE			PROJECT NAME			
	DRAWN BY	DATE			TASK NO.			
					TASK NAME			

NOTES:

4. Checks for integer variable or constant - prints DIAG 63 if not.



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE  
FLOWCHART  
DECISION TABLE  
OTHER

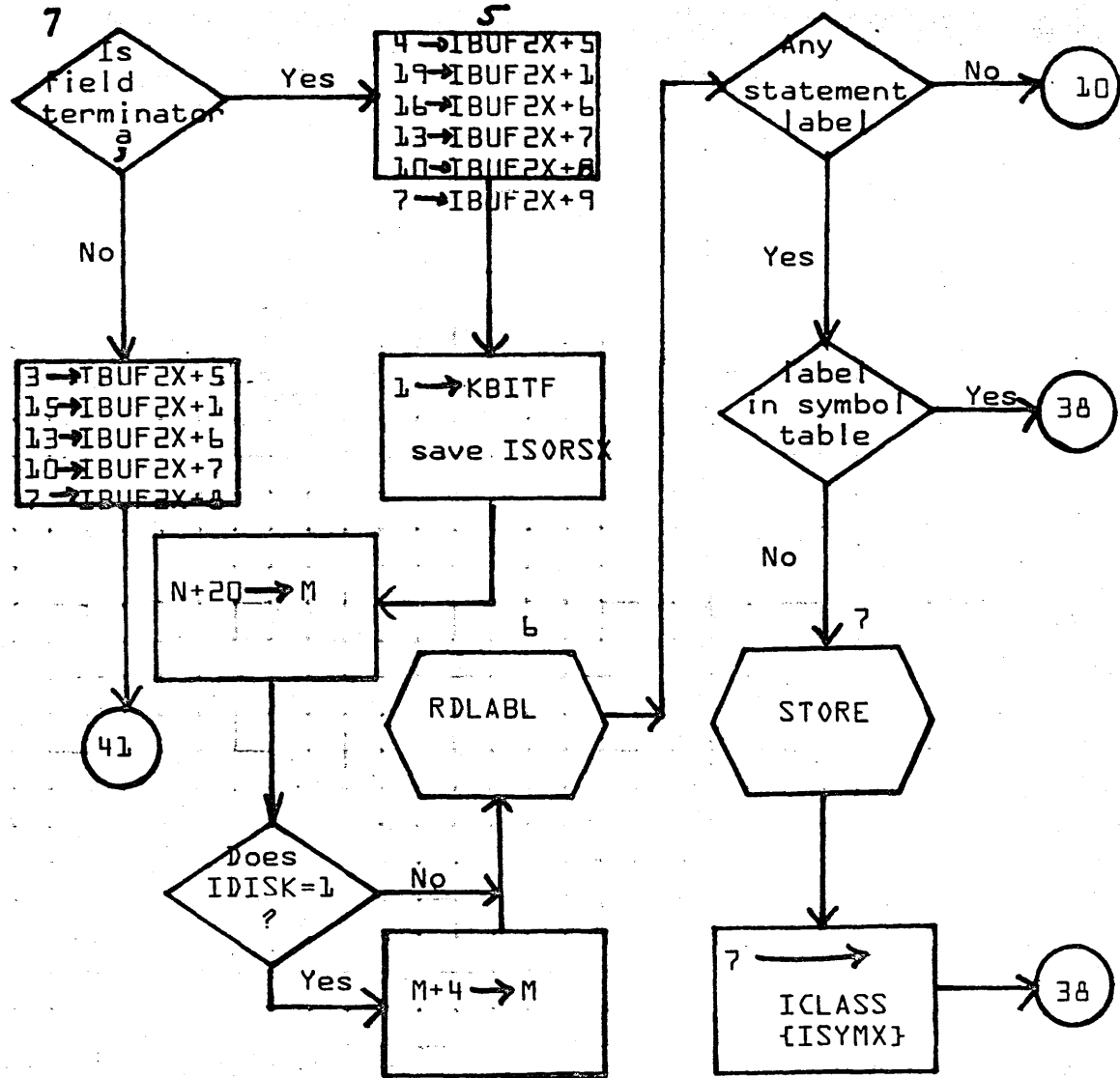
DOCUMENT CLASS	<i>ZMS</i>	MACH. TYPE	<i>1700</i>
DOCUMENT TITLE	<i>IOSPR</i>	PAGE <i>2</i> OF <i>16</i>	
NUMBER	ISSUE DATE	TASK NO.	TASK NAME
DRAWN BY	DATE		

PROJECT NO.	REV	APPROVED	DATE
PROJECT MGR			
PROJECT NAME			
TASK NO.			
TASK NAME			

2-251

NOTES:

- 5. Set up formatted I/O.
- 6. Read statement label.
- 7. Put label in symbol table.
- 8. For IBUF2X+5 means IBUF2{IBUF2X+5}.



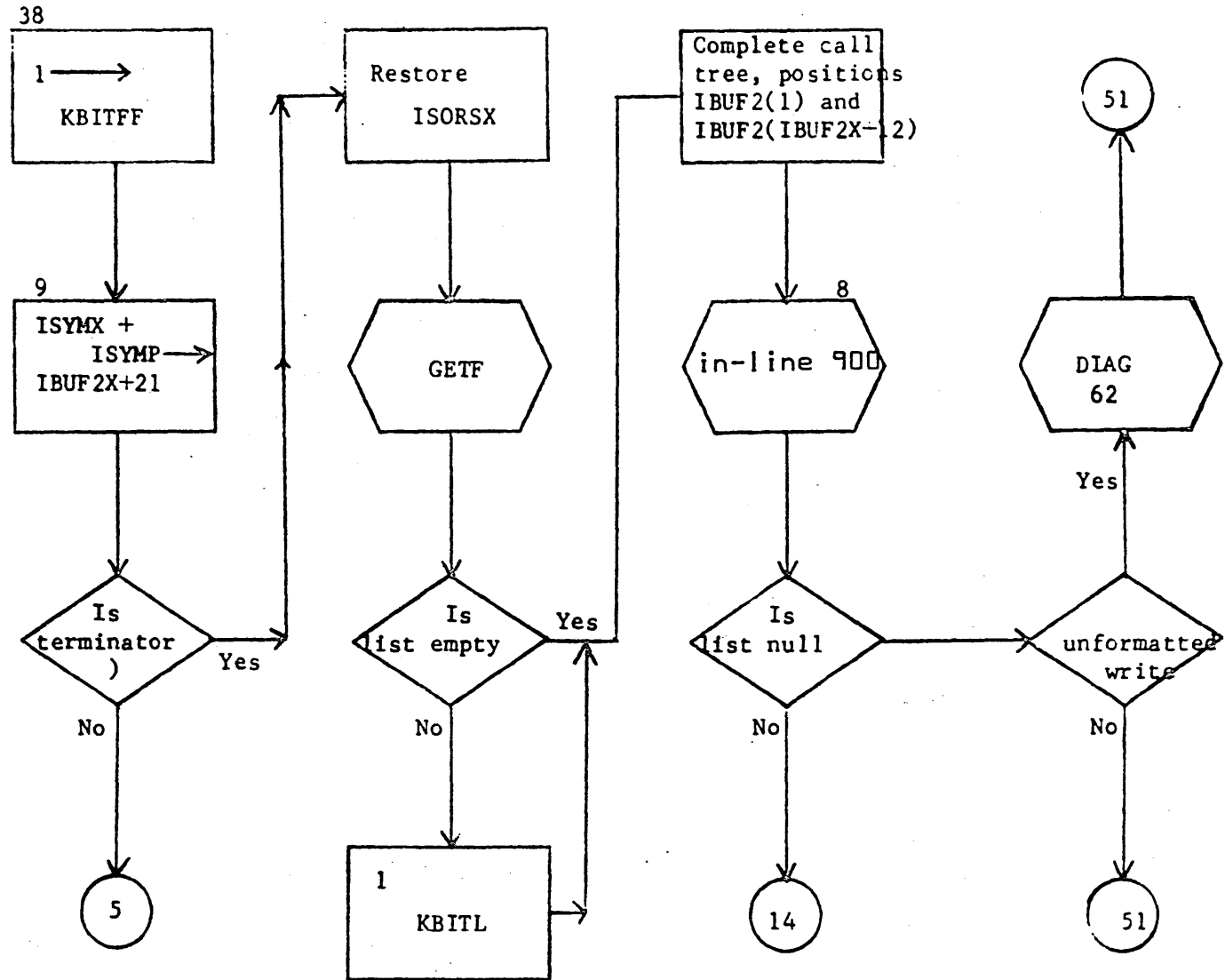
CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	<i>IMS</i>	MACH. TYPE	<i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	<i>IOSPR</i>	PAGE <i>3</i> OF <i>16</i>		PROJECT MGR.			
NUMBER	ISSUE DATE			PROJECT NAME			
DRAWN BY	DATE			TASK NO.			
				TASK NAME			

NOTES:

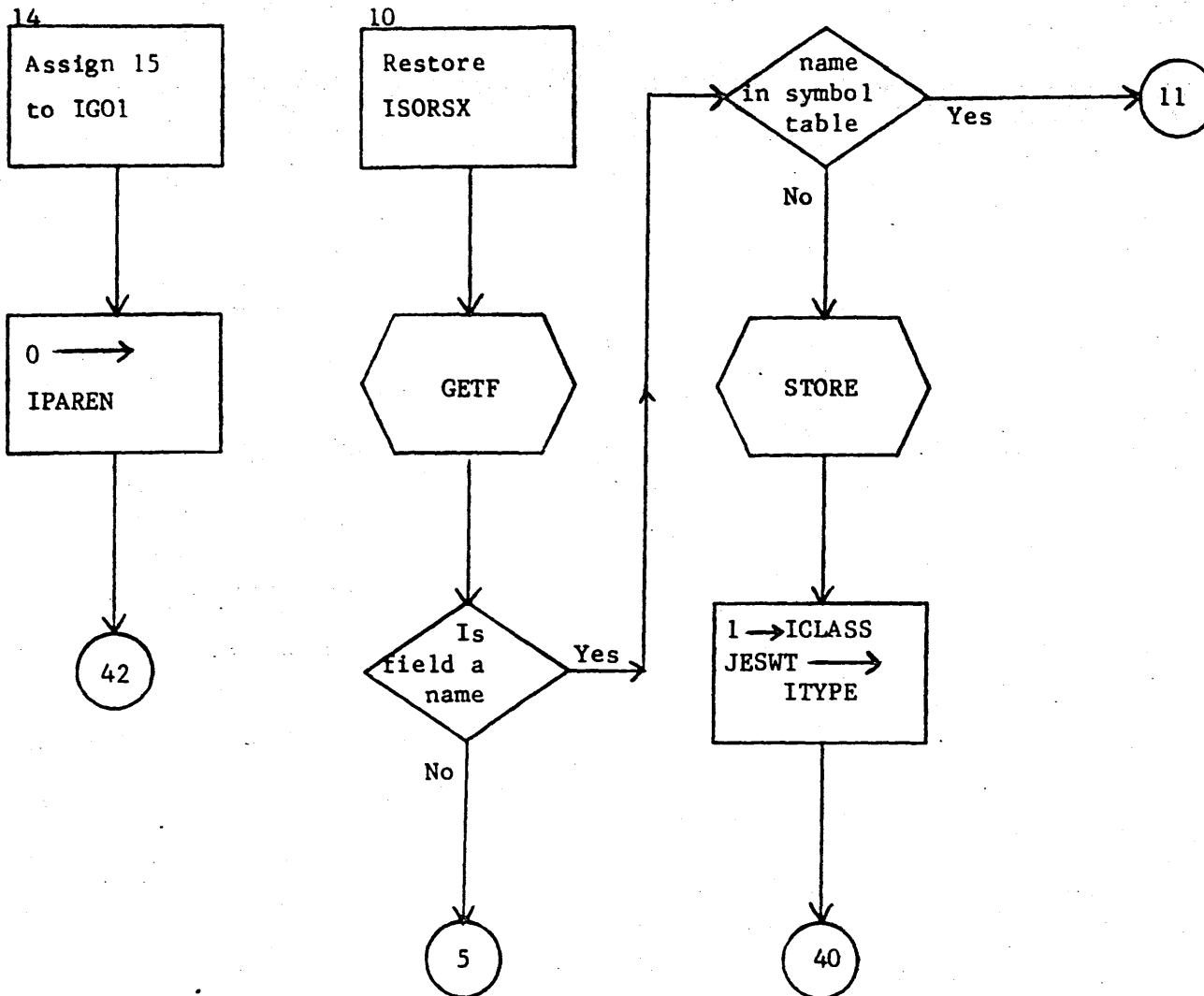
8. Output the statement.



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE  
FLOWCHART  
DECISION TABLE  
OTHER

DOCUMENT CLASS	<i>IMS</i>	MACH. TYPE	<i>1700</i>	PROJECT NO.		REV	APPROVED	DATE
DOCUMENT TITLE	<i>TOSAP</i>			PROJECT MGR				
		PAGE	<i>4</i> OF <i>16</i>	PROJECT NAME				
NUMBER		ISSUE DATE		TASK NO.				
DRAWN BY		DATE	<i>5-1-7</i>	TASK NAME				



**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

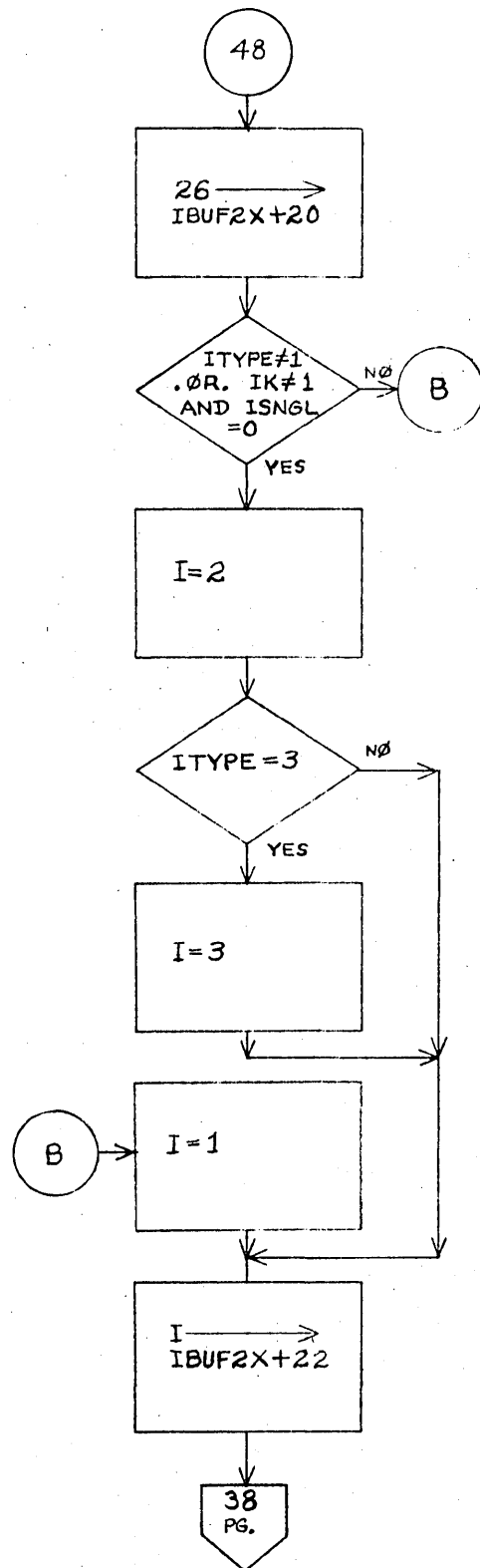
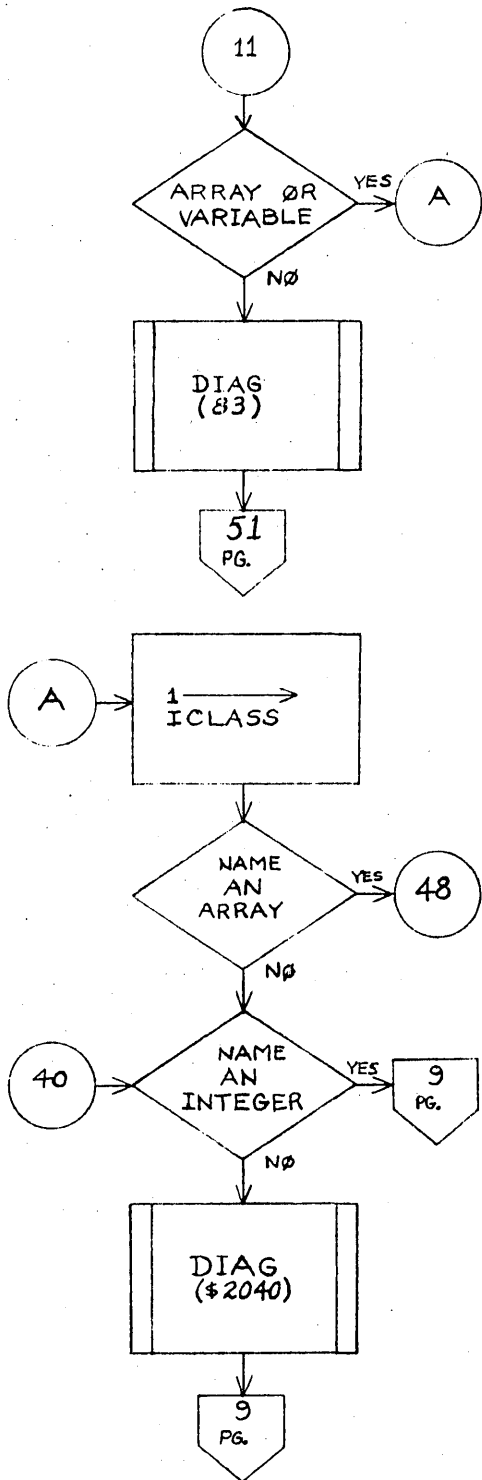
- SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

DOCUMENT CLASS	<i>IMS</i>	MACH. TYPE	<i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	<i>IOSAP</i>			PROJECT MGR.			
		PAGE	<i>5 OF 16</i>	PROJECT NAME			
NUMBER		ISSUE DATE		TASK NO.			
DRAWN BY		DATE	<i>3-6-7</i>	TASK NAME			

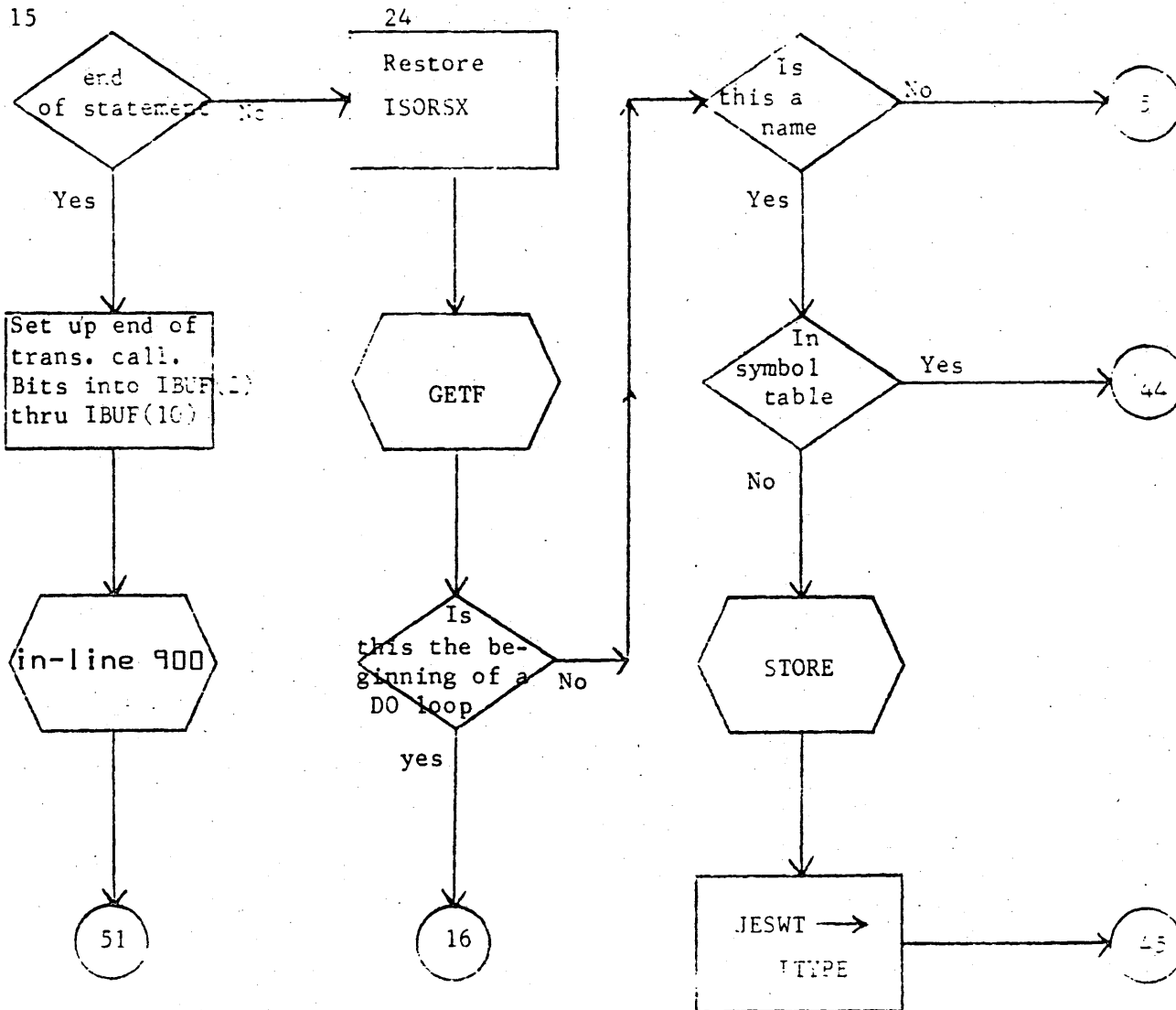




LA JOLLA FACILITY

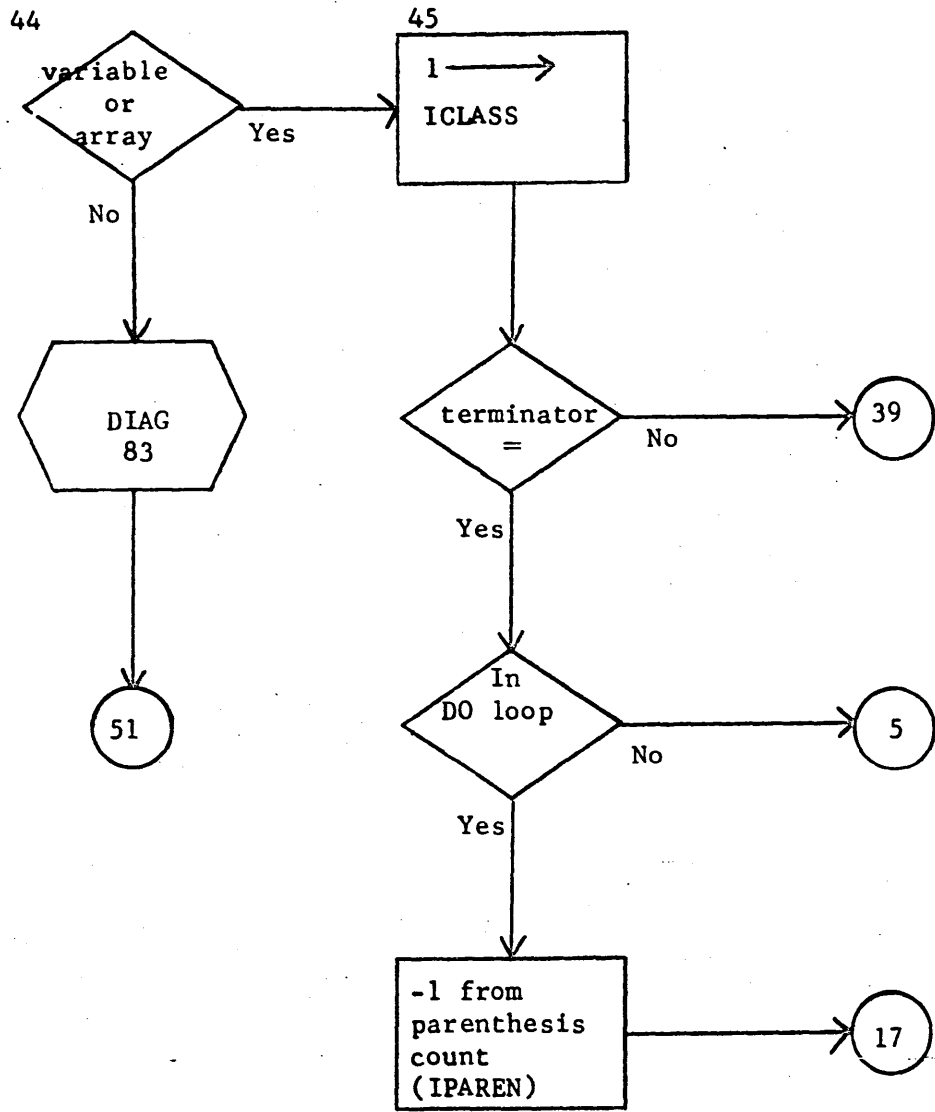


TITLE			ASEM PR			DRG. NO.		
						REVISION		
DRAWN BY		PROJ.		DATE		SHEET		4 OF 8
CSD 203								



CONTROL DATA CORPORATION SOFTWARE DOCUMENT  SAMPLE CODE FLOWCHART DECISION TABLE OTHER	DOCUMENT CLASS	ZMS	MACH TYPE	1700	PROJECT NO.		REV	APPROVED	DATE
	DOCUMENT TITLE	ZOSPR			PROJECT MGR				
			PAGE	7 OF 16	PROJECT NAME				
	NUMBER		ISSUE DATE		TASK NO				
	DRAWN BY		DATE		TASK NAME				

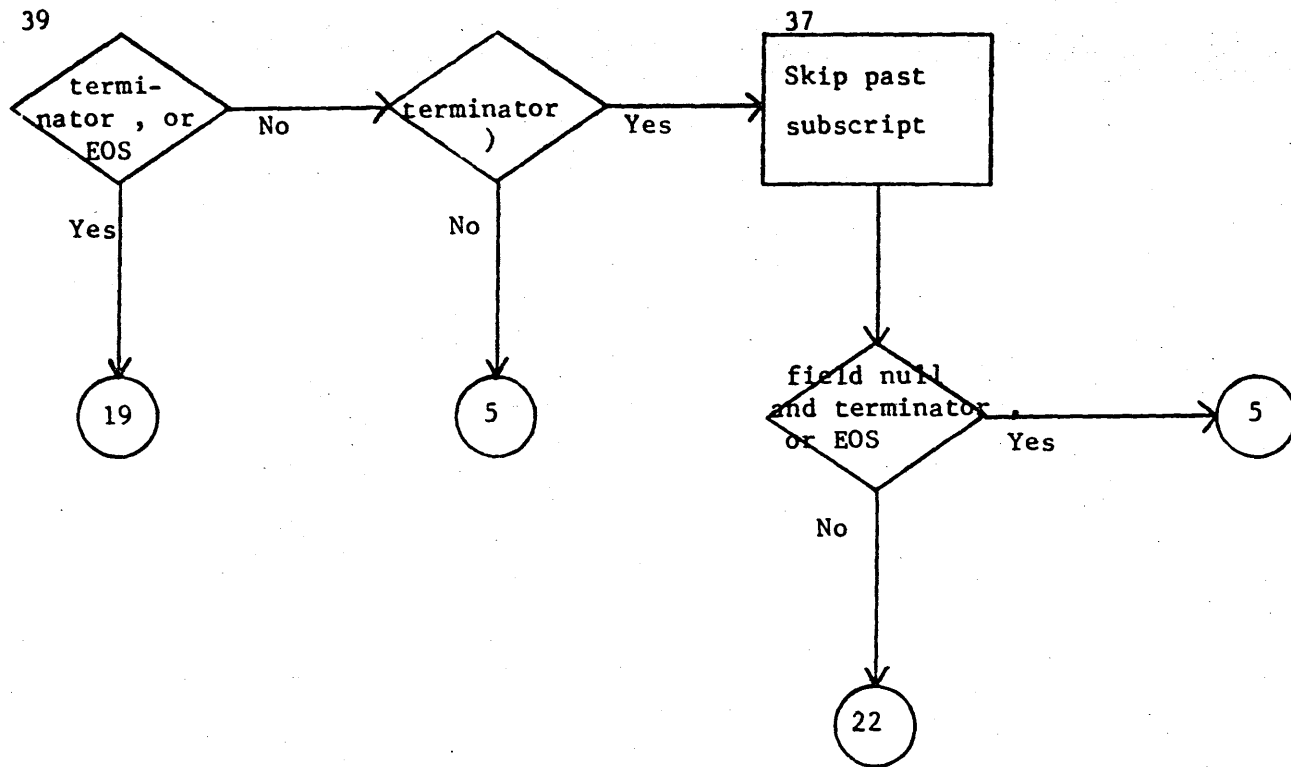
2-266



**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	<i>IMS</i>	MACH. TYPE	<i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	<i>TOSP</i>	PAGE <i>8</i> OF <i>16</i>		PROJECT MGR.			
NUMBER		ISSUE DATE		PROJECT NAME			
DRAWN BY		DATE		TASK NO.			
				TASK NAME			



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	MACH. TYPE	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE		PROJECT MGR.			
		PAGE	PROJECT NAME			
	NUMBER	ISSUE DATE	TASK NO.			
	DRAWN BY	DATE	TASK NAME			

992-2

22  
Move Q8QX  
calling sequence  
parameters in-  
to IBUF2

GETC

Bypass  
blanks from  
ITEMP3 to  
ITEMP4

Is  
it a  
blank

field  
done

43  
Store ) and  
EOS using  
STCHAR

46  
STCHAR  
IBUF2,J,JCHAR

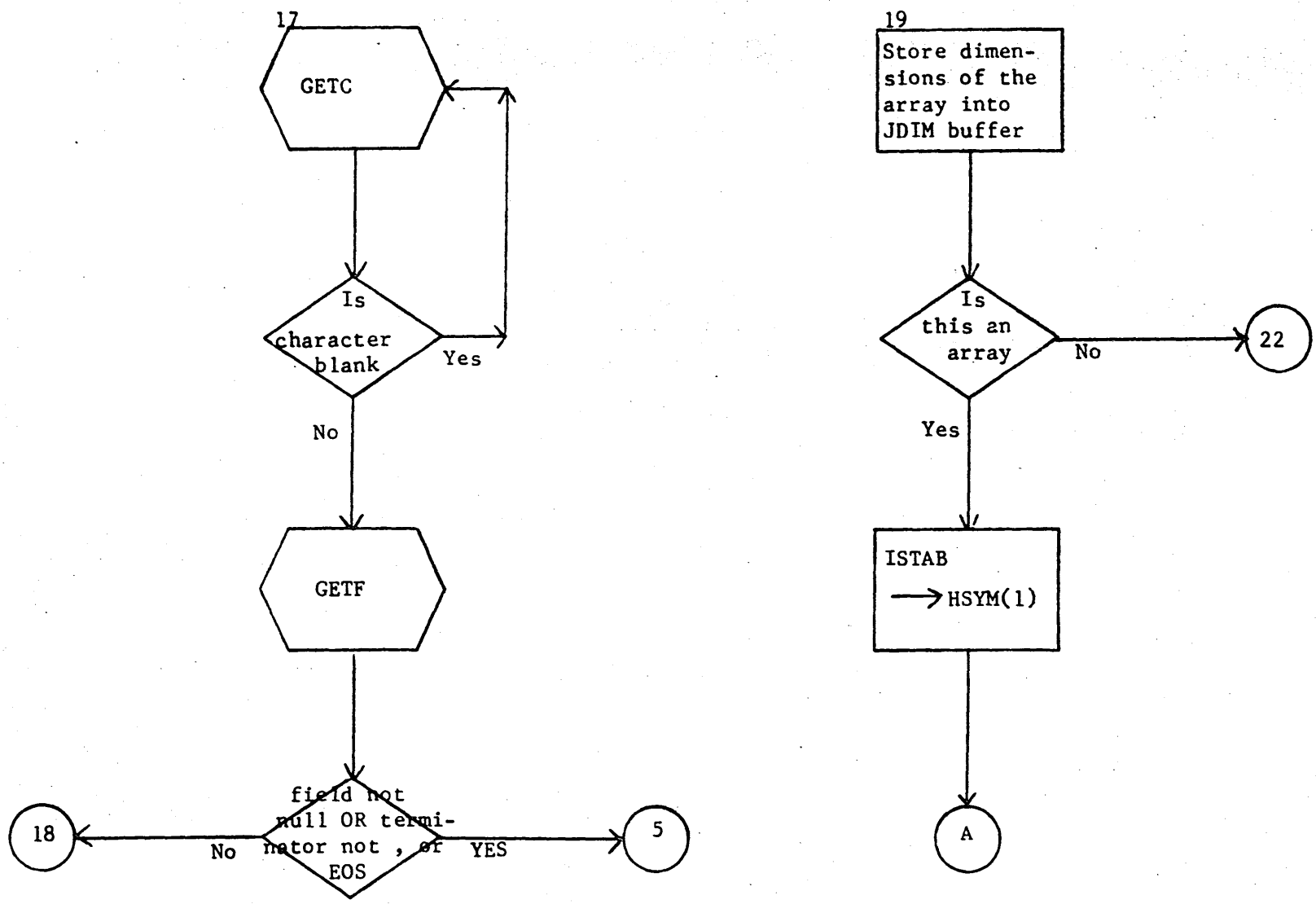
in-line 900

IG01

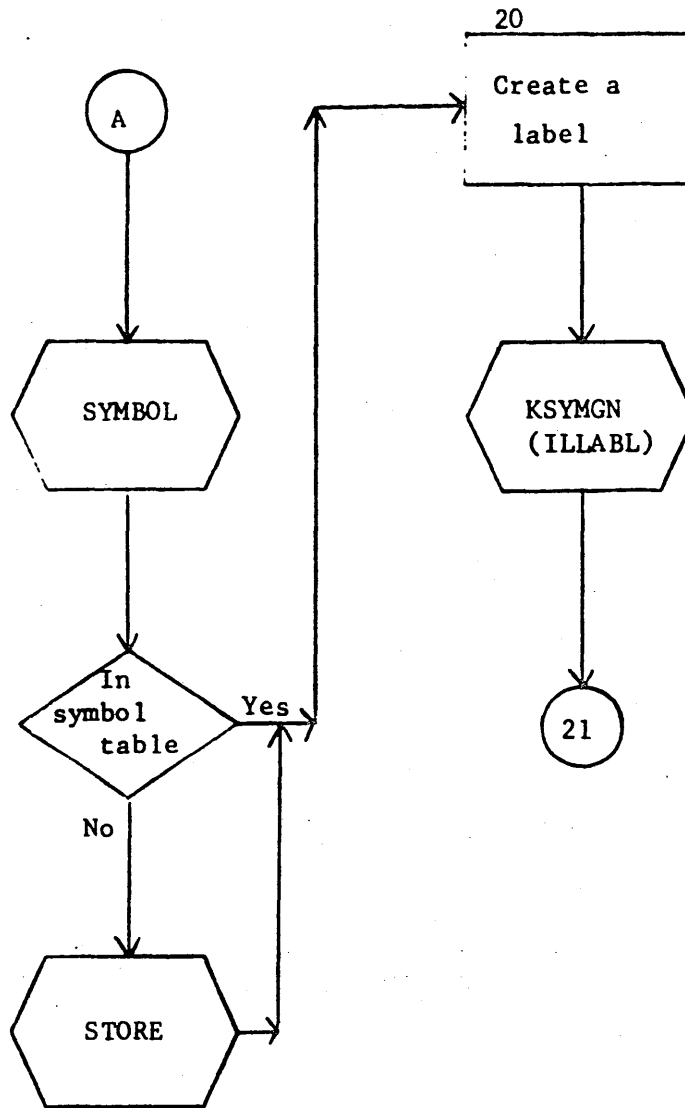
<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE _____ FLOWCHART _____ DECISION TABLE _____ OTHER _____	DOCUMENT CLASS	<i>JAS</i>	MACH. TYPE	<i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	<i>JOSCP</i>	PAGE <i>10</i> OF <i>16</i>		PROJECT MGR			
	NUMBER	ISSUE DATE	TASK NO		PROJECT NAME			
	DRAWN BY	DATE	TASK NAME					

2-269

A  
B  
C  
D



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS <i>IMS</i>	MACH. TYPE <i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE <i>TOSAP</i>		PROJECT MGR.			
		PAGE <i>11</i> OF <i>16</i>	PROJECT NAME			
	NUMBER	ISSUE DATE	TASK NO.			
	DRAWN BY	DATE <i>5 67</i>	TASK NAME			



**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

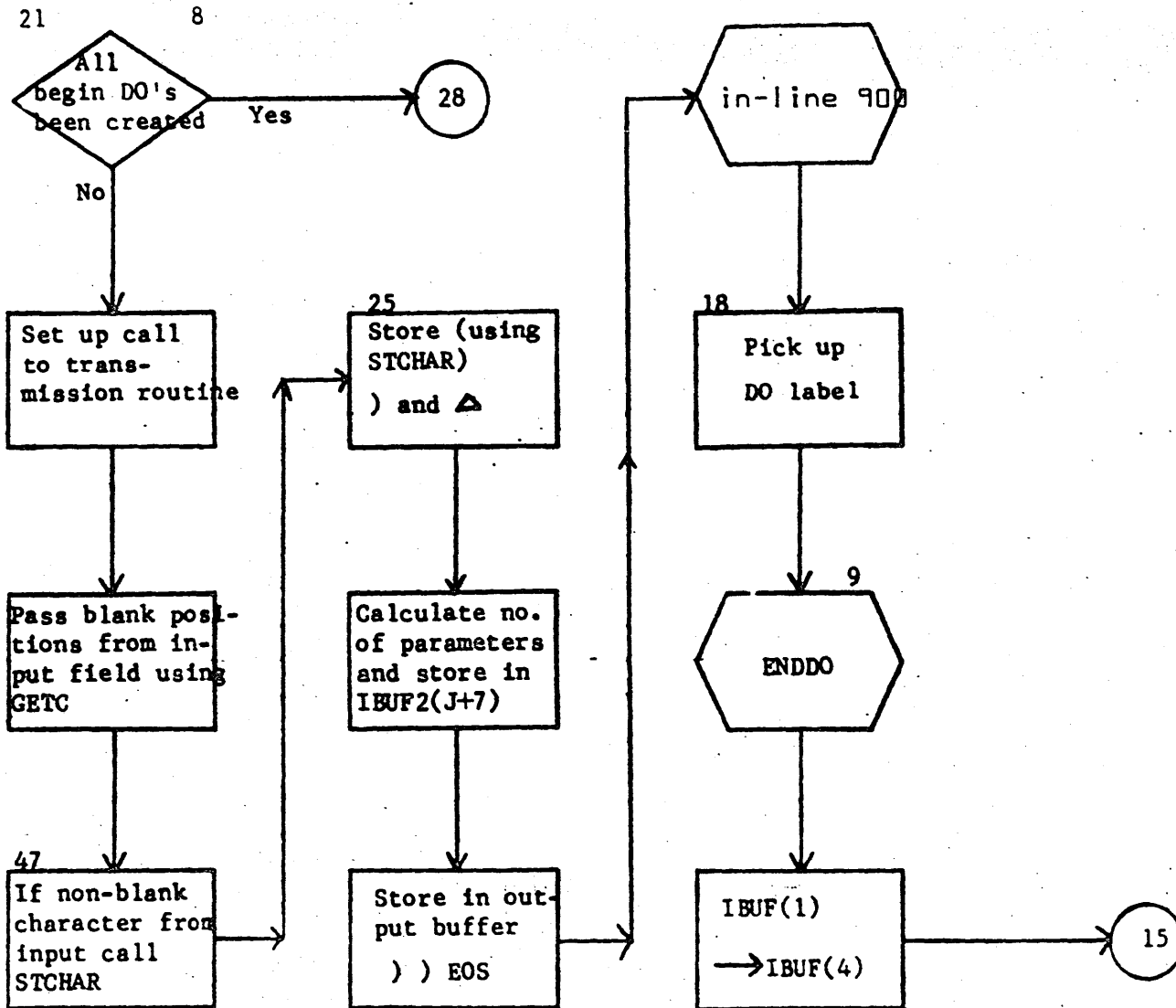
- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS <i>JMS</i>	MACH. TYPE <i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE <i>JOSAP</i>		PROJECT MGR.			
	PAGE <i>129</i> OF <i>16</i>	PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.			
DRAWN BY	DATE <i>5/1/77</i>	TASK NAME			

NOTES:

8. ITEMPB > 0

9. Create an end DO constraint.



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS <i>INS</i>	MACH. TYPE <i>1700</i>	PROJECT NO.	REV.	APPROVED	DATE
	DOCUMENT TITLE <i>IOSAP</i>		PROJECT MGR.			
		PAGE <i>13 of 16</i>	PROJECT NAME			
	NUMBER	ISSUE DATE	TASK NO.			
	DRAWN BY	DATE <i>3-6-7</i>	TASK NAME			

2-272



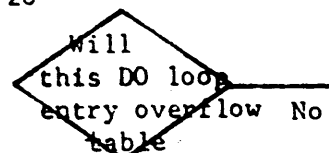
A

B

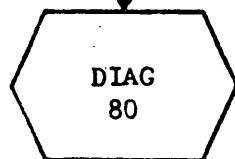
C

D

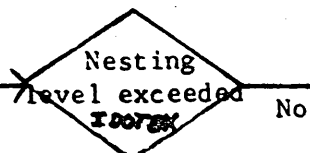
28



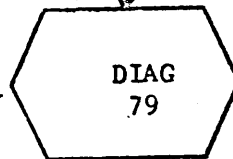
Yes



51



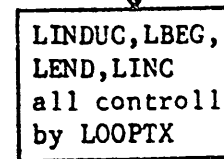
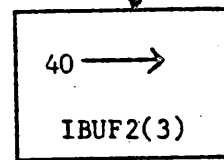
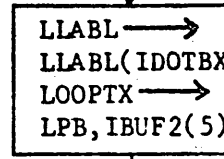
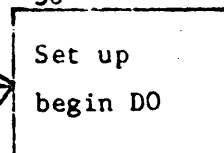
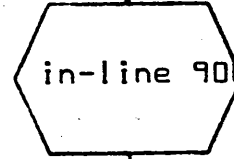
Yes



51

No

30

5  
IBUF2(1)

21

CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE   
FLOWCHART   
DECISION TABLE   
OTHER

DOCUMENT CLASS

IMS

MACH. TYPE

1700

PROJECT NO.

REV

APPROVED

DATE

DOCUMENT TITLE

TOSPR

PROJECT MGR.

PAGE 12 OF 16

PROJECT NAME

NUMBER

ISSUE DATE

TASK NO.

DRAWN BY

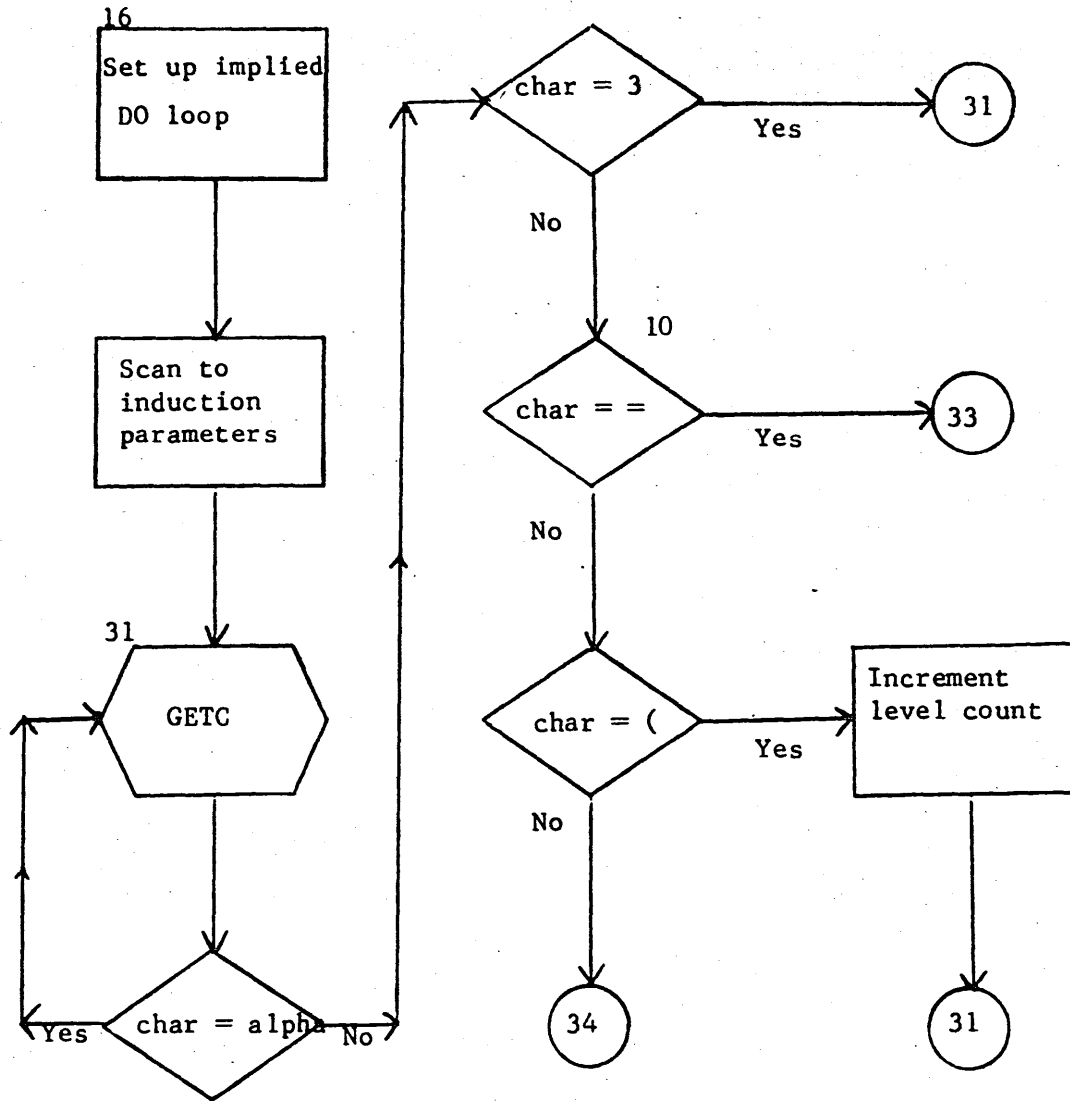
DATE

TASK NAME

2-273

NOTES:

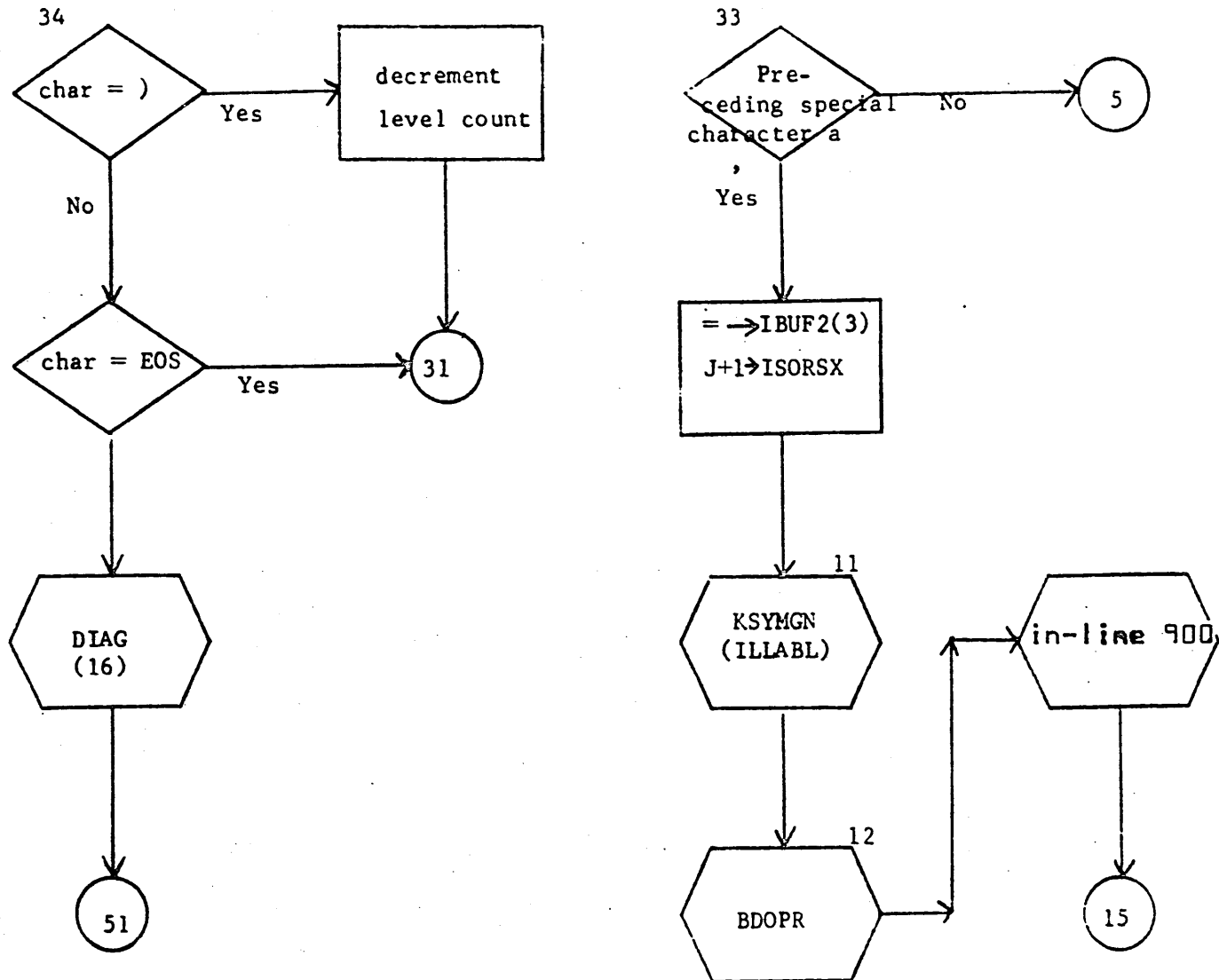
10. At zero parenthesis level.



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS <i>IMS</i>	MACH. TYPE <i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE <i>IOSPR</i>		PROJECT MGR.			
		PAGE <i>15</i> OF <i>16</i>	PROJECT NAME			
	NUMBER	ISSUE DATE	TASK NO.			
	DRAWN BY	DATE	TASK NAME			

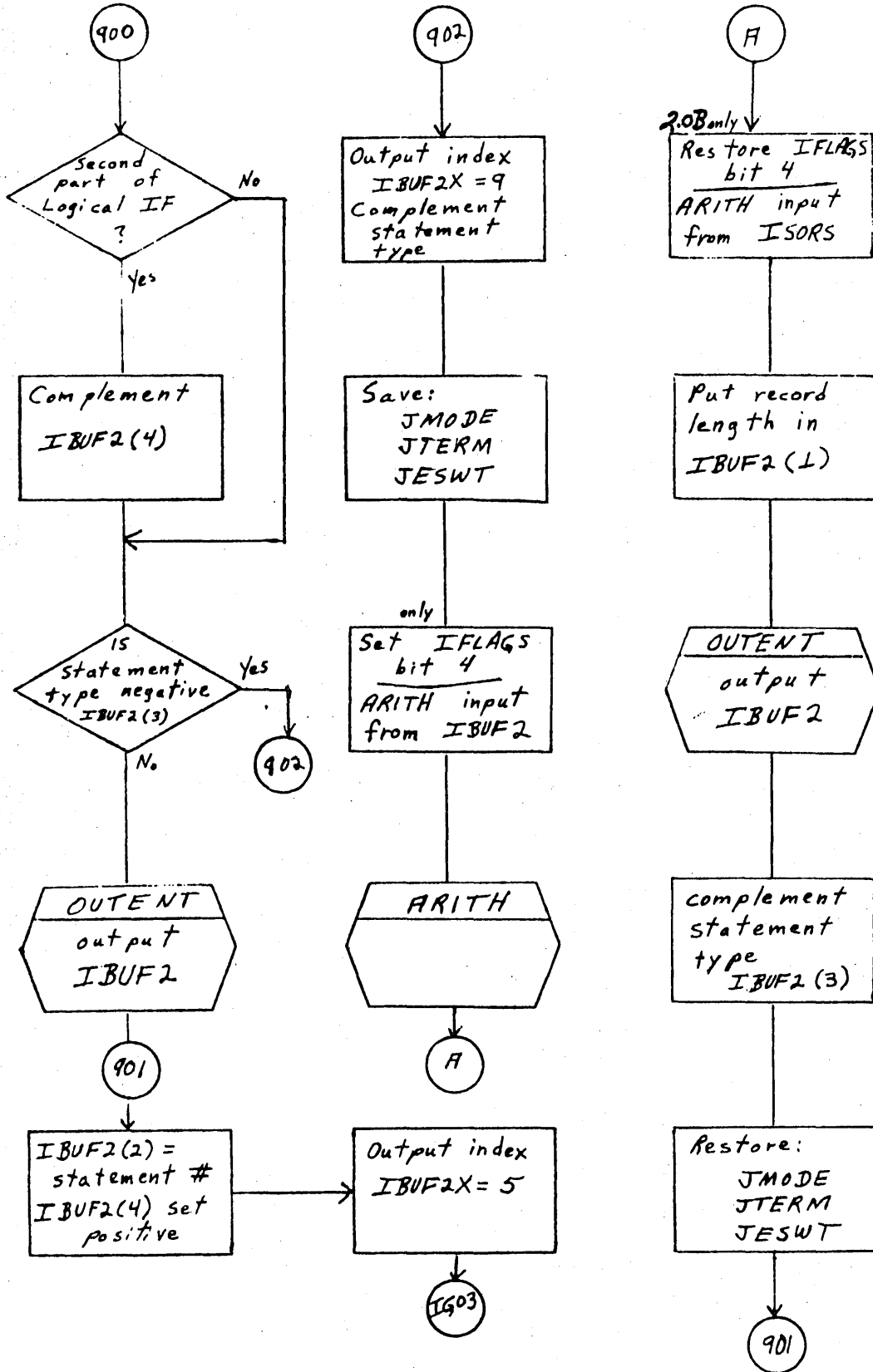
NOTES:

- 11. Allocate one symbol table entry.
- 12. Create begin DO.



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE FLOWCHART DECISION TABLE OTHER	DOCUMENT CLASS	MACH. TYPE	PROJECT NO.	REV.	APPROVED	DATE
	DOCUMENT TITLE		PROJECT MGR.			
		PAGE	PROJECT NAME			
	NUMBER	ISSUE DATE	TASK NO.			
	DRAWN BY	DATE	TASK NAME			

2-275



DOCUMENT CLASS IMS PAGE NO. 2-277  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

## 2.4.13 BDOPR

The BDOPR subroutine is called to process DO statements of the form:

DO label induc = lbeg, lend, incr

The parameters for this statement are as follows:

1. "label" is the statement label assigned to the final statement of the do loop.
2. "induc" is the induction parameter.
3. "lbeg" is the beginning parameter.
4. "lend" is the end parameter.
5. "incr" is the increment parameter.

## 2.4.13.1 "label"

A call is made to the RDLABL subroutine in order to extract the label from the source language input. Upon return from RDLABL the following is true:

1. Either  $(ILLABL) = 0$  if either -
  - a. the label is missing, or -
  - b. the label is illegal,
 or  $(LLABL) \neq 0$  if the label is present and legal.
2. Either  $(ISYMD) = 0$  if the label does not appear in the symbol table, or  $(ISYMD) \neq 0$  if an entry for the label is present. If  $(ISYMD) = 0$ , a call is made to the STORE routine to insert this label in the symbol table.
3. If  $(ISYMD) \neq 0$ , the statement label must either be undefined, or it must occur subsequent to the DO statement. If -
 

$(ISNOL(ISYMX)) = 0$ ,

 the label is undefined, and if
 

$(ISNOL(ISYMX)) > (ISTNO)$

 the label occurs subsequent to the DO statement where -
 

$(ISTNO) =$  the statement number for the source statement currently being processed.

## 2.4.13.2 "induc"

The restrictions on the induction parameter are as follows:

DOCUMENT CLASS IMS PAGE NO 2-278  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

1. It must be an integer variable.
2. It may not be preceded by an algebraic sign.
3. It must be followed by an equal sign.
4. This parameter must not be used as any parameter in another DO statement if the do-loop for that statement contains the do-loop of the statement currently being processed.

A call is made to the CKNAME subroutine to extract the parameter from the source language input.

Upon return from CKNAME either -

1. (IVCFLG) = 2 if the parameter is an integer variable, or -
2. (IVCFLG)  $\neq$  2 if not.

If 2 is the case an error diagnostic is given prior to the return from CKNAME. Therefore, following the return from CKNAME, only an exit is made from BDOPR.

#### 2.4.13.3

"lbeg"

The restrictions on the use of the beginning parameter are as follows:

1. It must be either an integer variable or an integer constant.
2. It may not be preceded by an algebraic sign.
3. It must be separated from the preceding parameter by an equal sign.
4. It must be followed by a comma.

A call is made to the CKIVC subroutine to extract this parameter from the source buffer. Upon return from CKIVC, either -

1. (IVCFLG) = 0 if the parameter is non integer,
2. (IVCFLG) = 1 if the parameter is an integer constant, or -
3. (IVCFLG) = 2 if the parameter is an integer variable.

If (IVCFLG) = 0, an error diagnostic is given by the CKIVC subroutine. Therefore, after returning from CKIVC, only an exit is made from BDOPR.

#### 2.4.13.4

"lend"

The restrictions for the use of this parameter are the same as those for the "lbg" parameter. The CKIVC subroutine is used to extract this parameter from the source buffer the same as for the "lbg" parameter.

DOCUMENT CLASS TMS PAGE NO. 2-279  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. COOS V. 2.0 MACHINE SERIES 1700

### 2.4.13.5 $\nabla$ incr $\nabla$

The restrictions for the use of this parameter are the same as for the  $\nabla$ lbeg $\nabla$  parameter with one exception:

The increment parameter may be preceded by a minus sign. This parameter is also followed by an EOS. This parameter is optional and may be omitted. If so, the  $\nabla$ lend $\nabla$  parameter is followed by an EOS. A value of 1 is substituted for the increment parameter.

### 2.4.13.6 Output of BDOPR

The BDOPR subroutine will generate a 5 word output entry for the do-loop table and a 1 word output entry for the output buffer.

### 2.4.16.1 Do-Loop Table Entry

$\nabla$ LOOP $\nabla$  = name of table  
 {LOOP $\nabla$ TS} = capacity of the table  
 {LOOP $\nabla$ TB} = length of a table entry  
 {LOOP $\nabla$ TX} = pointer to the entry in the table

A DO-LOOP table entry is five words long. The format of the DO-LOOP table is as follows:

WORD1:           LEND       = bits 0-15 of LOOP $\nabla$ {4}.  
                   {LEND}   = pointer to symbol table entry  
                                   containing the  $\nabla$ lend $\nabla$   
                                   parameter.

WORD2:   1.   LID        = bit 15 of LOOP $\nabla$ {5}.  
                   {LID}   = 1 if the  $\nabla$ incr $\nabla$  parameter is  
                                   preceded by a minus sign.

                  2.   LINDUC = bits 0-15 of LOOP $\nabla$ {1}.  
                           {LINDUC}= pointer to the symbol 1 table  
                                   entry containing the  $\nabla$ induc $\nabla$   
                                   parameter.

WORD3:           LBEG       = bits 0-15 of LOOP $\nabla$ {2}.  
                   {LBEG}   = pointer to the symbol table  
                                   entry containing the  $\nabla$ lbeg $\nabla$   
                                   parameter.

WORD4:           LINC       = bits 0-15 of LOOP $\nabla$ {3}.

DOCUMENT CLASS ... IMS ... PAGE NO. 2-280  
PRODUCT NAME: 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. COO5 V. 2.0 ... MACHINE SERIES 1700

{LINC} = pointer to symbol table entry  
containing the 'incr'  
parameter.

WORDS: LLABL = bits 0-14 of ID0TB{5}.  
{LLABL} = pointer to symbol table entry  
containing the 'label'  
parameter.



DOCUMENT CLASS IMS PAGE NO. 2-281  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. COO5 V. 2.0 MACHINE SERIES 1700

### 2.4.13.6.3 Output Buffer Entry

An entry made in the output buffer consists of 1 word, a pointer to the LOOP entry for this DO statement.

{LOOPTX} → IBUF2{IBUF2X}

### 2.4.13.7 Errors

Error diagnostics are given by BDOPR for the following:

1. An illegal label or a missing label.
2. The label occurs prior to the DO statement.
3. Overflow of the DO-LOOP table.
4. Overflow of the NESTED-DO-LOOP table.
5. The induction parameter is currently in use as an induction variable in a do-loop containing the do-loop for the statement currently being processed.
6. Illegal characters separating the parameters in the DO statement.

DOCUMENT CLASS \_\_\_\_\_ IMS \_\_\_\_\_ PAGE NO. 2-282  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

Errors 1, 2 & 5 will cause termination of the DO statement processing, and an exit will be made from BDOPR following the printout of the error diagnostic. Following the printout of the error diagnostic for errors 3, 4 & 6, processing of the DO statement will be resumed.

#### 2.4.13.8 Induction Variables

Indicators are to be set in each of the symbol table entries containing the "lbeg", "lend", "induc" and "incr" parameters to show that these are induction variables. These indicators are reset when the statement label is encountered that marks the end of the do-loop. Such an indicator is set by -

1 → INDUCV(ISYMX)

#### 2.4.13.9 Do-Loops Originating in I/O Lists

A DO statement in the source buffer is terminated by an EOS. A do loop may originate in an I/O list such as -

WRITE (unit, format statement label)(A(I),I=1,10),...

If so, an indicator is set prior to entering BDOPR. This indicator is recorded in ITEMPX. If the origin of the do loop is in an I/O list, (ITEMPX) = -1. If this is the case, a right parenthesis replaces an EOS as the end of the statement in the processor.

A

B

C

D

BDOPR

SAVE FLAG  
{IVCFLG} →  
ITEMPX

IS  
DO-LOOP ORIGIN  
IN LIST OF I/O  
STATEMENT  
{ITEMPX} =  
-1

17

2

DIAG  
ERR DIAG =  
°25°

EXIT FROM BDOPR

IS  
LABEL ILLEGAL  
OR MISSING  
{LLABL} =  
0

IS  
LABEL IN  
SYMB.TAB  
{ISYMD}  
≠ 0

STORE  
MAKE SYMBOL  
TABLE ENTRY

17

RDLABL  
GET LABEL FOR  
END OF DO-  
LOOP

IS  
LABEL UNDEFINED  
{ISNOL}{ISYMX}  
= 0

4

DOES  
LABEL OCCUR  
AFTER DO STATEMENT  
{ISNOL}{ISYMX}  
> {ISTNO}

4

DIAG  
ERR DIAG =  
°78°

5

CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE

FLOWCHART

DECISION TABLE

OTHER

DOCUMENT CLASS IMS MACH. TYPE 1700

DOCUMENT TITLE BDOPR

PAGE 1 OF 8

NUMBER ISSUE DATE

DRAWN BY DATE

PROJECT NO.	REV	APPROVED	DATE
PROJECT MGR.			
PROJECT NAME			
TASK NO.			
TASK NAME			

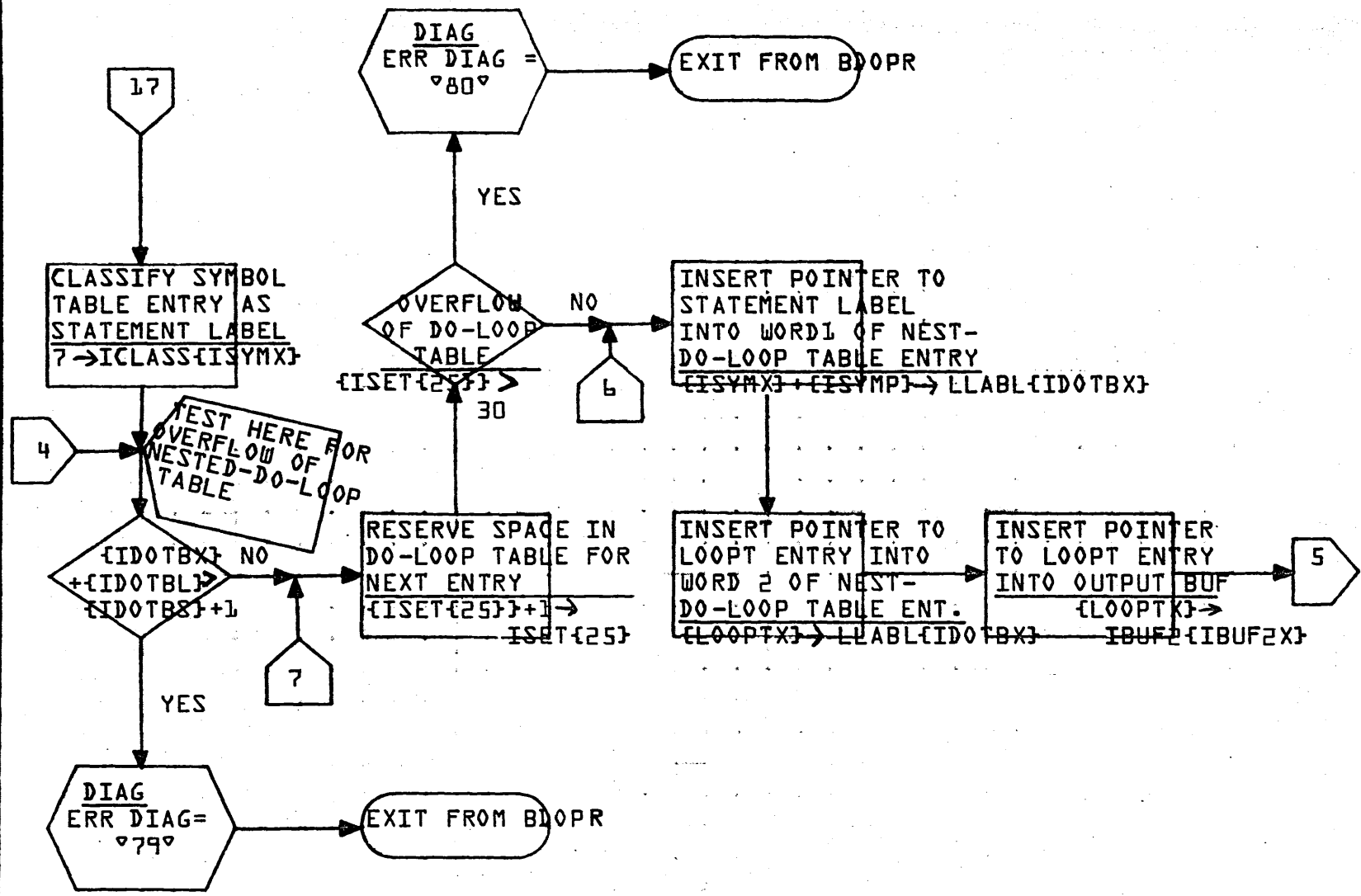
2-203

A

B

C

D



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE

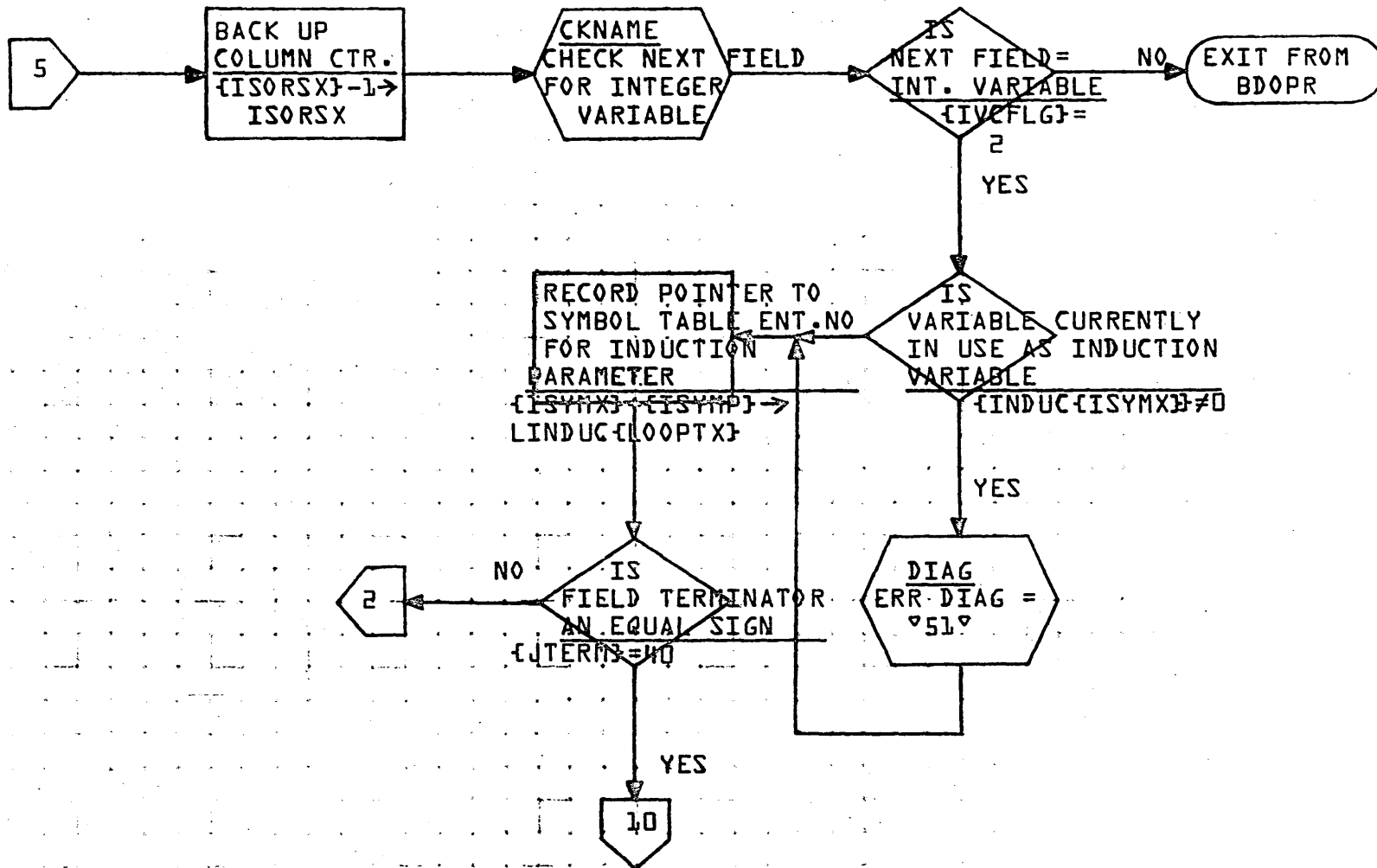
FLOWCHART

DECISION TABLE

OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700
DOCUMENT TITLE	BDOPR	PAGE 2 OF 8	
NUMBER	ISSUE DATE	TASK NO.	
DRAWN BY	DATE	TASK NAME	

PROJECT NO.	REV	APPROVED	DATE
PROJECT MGR.			
PROJECT NAME			
TASK NO.			
TASK NAME			



2-285

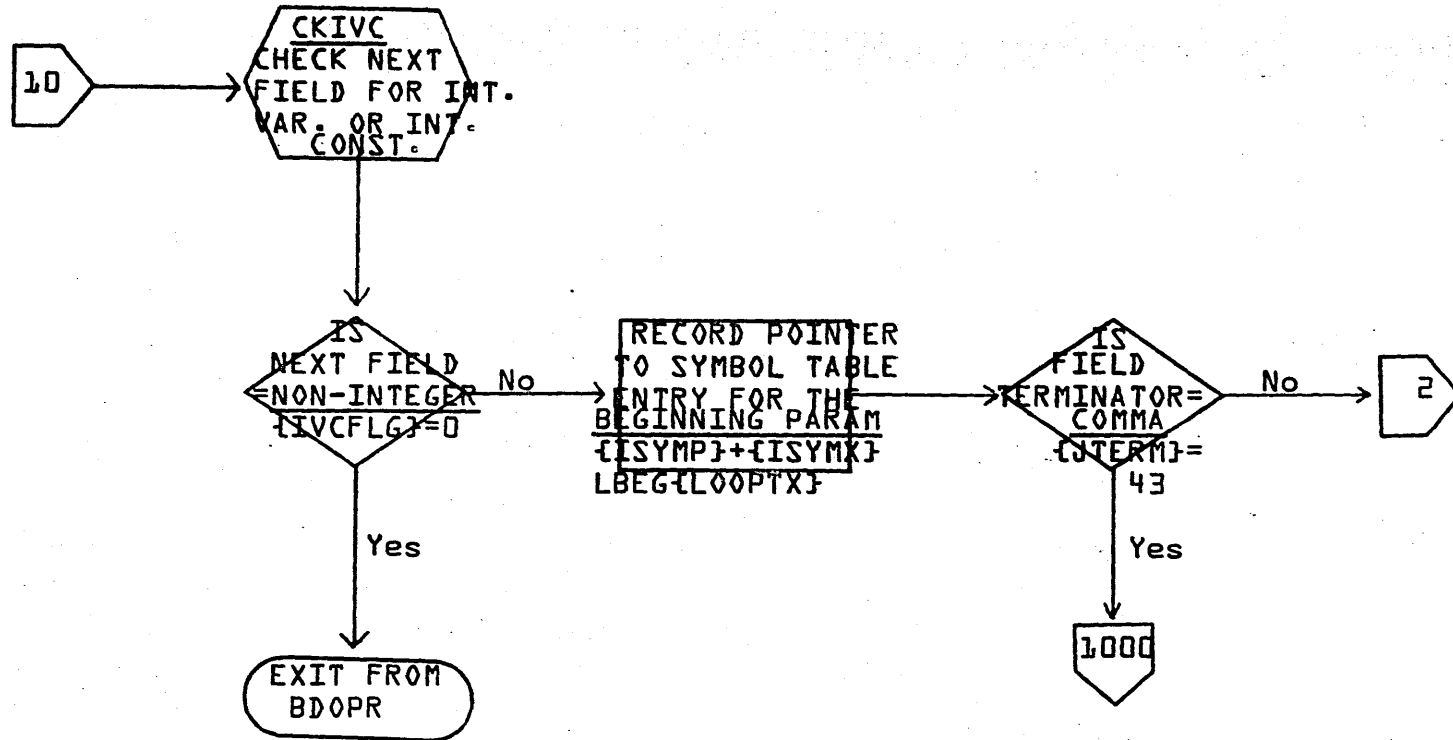
<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	BDOPR			PROJECT MGR.			
		PAGE 3 OF 8			PROJECT NAME			
	NUMBER		ISSUE DATE		TASK NO.			
	DRAWN BY		DATE		TASK NAME			

A

B

C

D

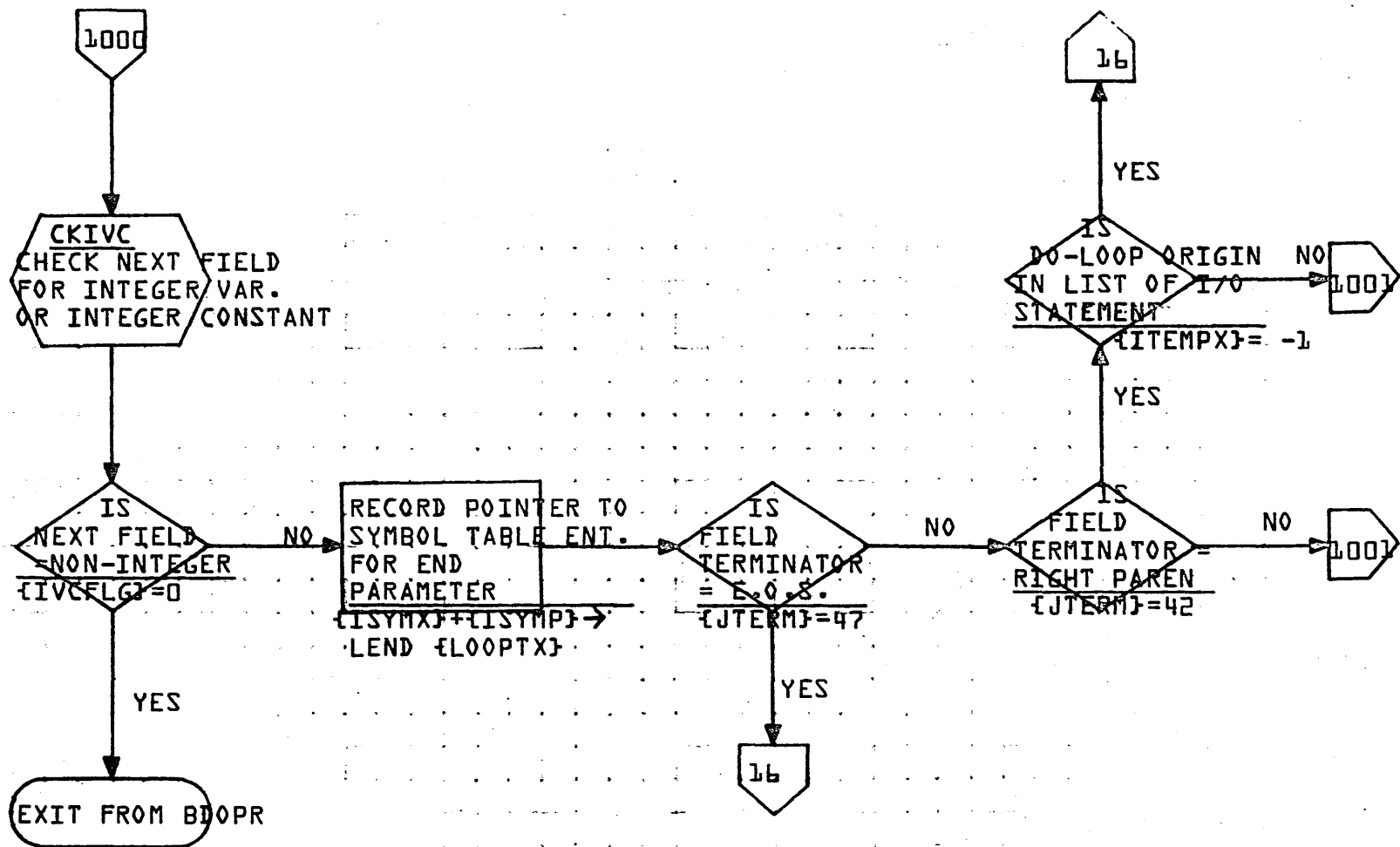


**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

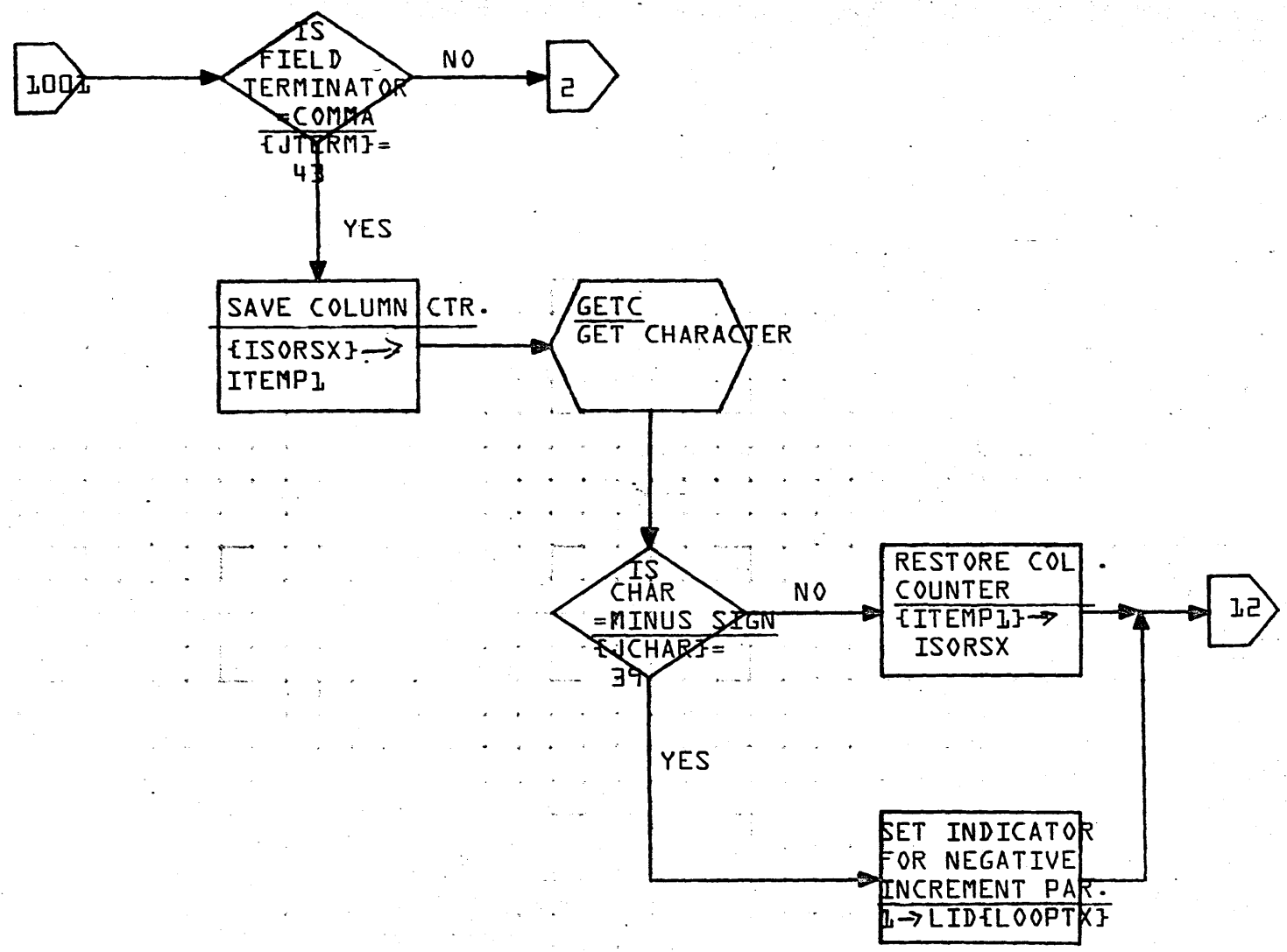
DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	BDOPR	PAGE 4 OF 8		PROJECT MGR.			
NUMBER	ISSUE DATE			PROJECT NAME			
DRAWN BY	DATE			TASK NO.			
				TASK NAME			

A  
B  
C  
D



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	BDOPR	PAGE 5 OF 8		PROJECT MGR.			
	NUMBER	ISSUE DATE			PROJECT NAME			
	DRAWN BY	DATE			TASK NO.			
					TASK NAME			

A  
B  
C  
D



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	BDOPR		PAGE 6 OF 8		PROJECT MGR.		
	NUMBER	ISSUE DATE	TASK NO.					
	DRAWN BY	DATE	TASK NAME					

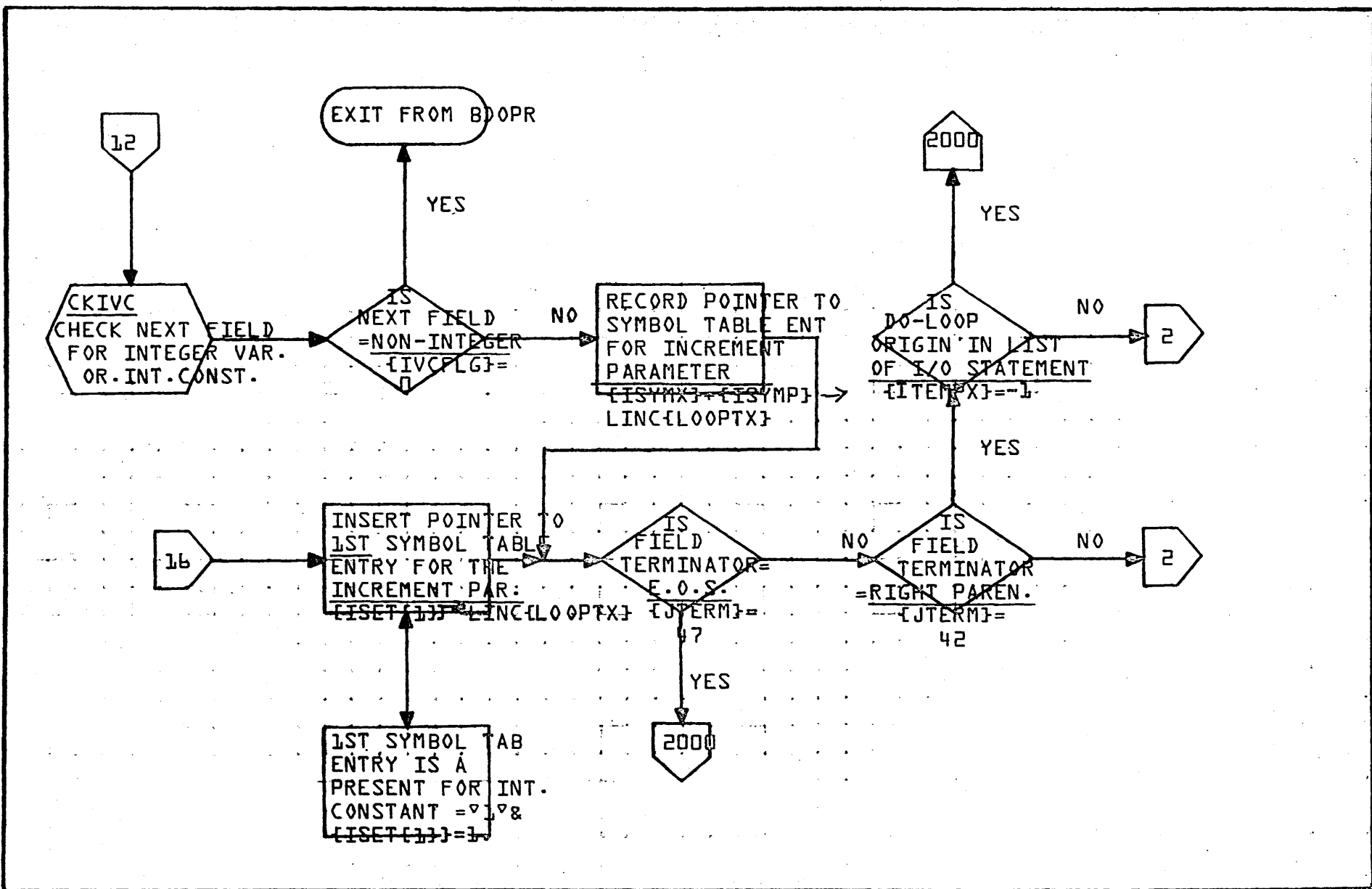


A

B

C

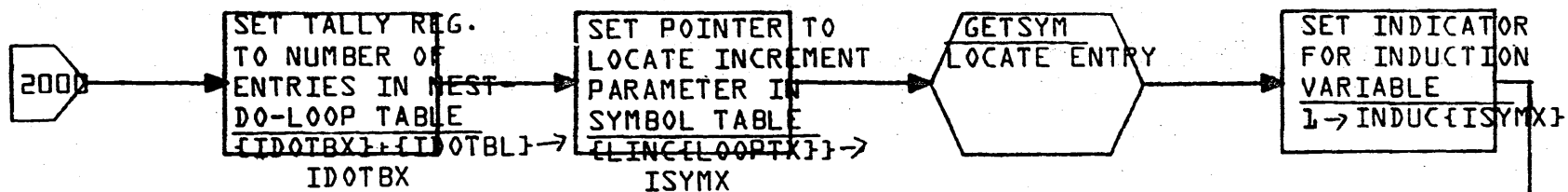
D



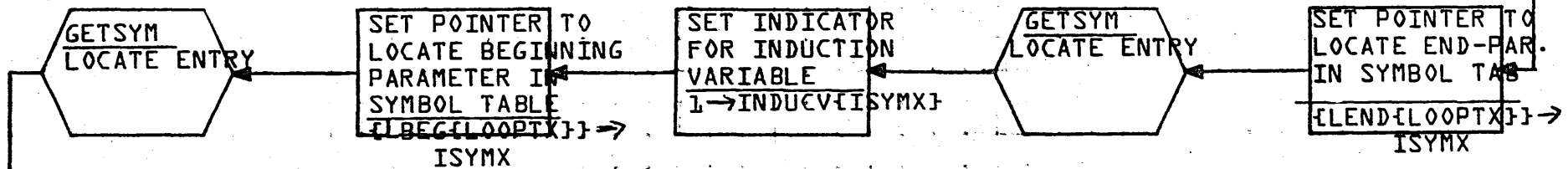
<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	BDOPR	PAGE 7 OF 8		PROJECT MGR.			
	NUMBER	ISSUE DATE			PROJECT NAME			
	DRAWN BY	DATE			TASK NO.			
					TASK NAME			

2-289

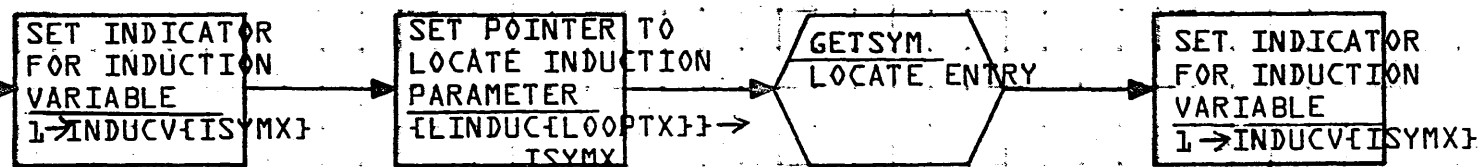
A



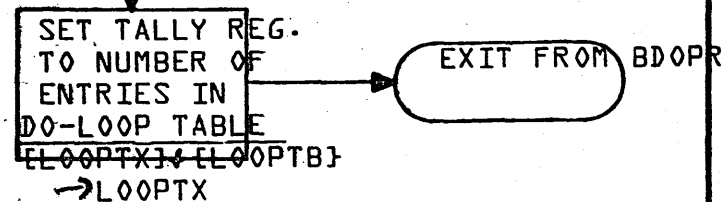
B



C



D



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE

FLOWCHART

DECISION TABLE

OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	BDOPR	PAGE 8 OF 8		PROJECT MGR.			
NUMBER	ISSUE DATE			PROJECT NAME			
DRAWN BY	DATE			TASK NO.			
				TASK NAME			

2-290

DOCUMENT CLASS IMS PAGE NO. 3-1  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. COO5\*3.1 A/B MACHINE SERIES 1700

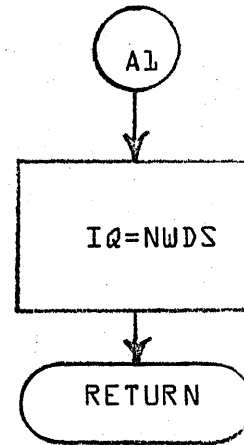
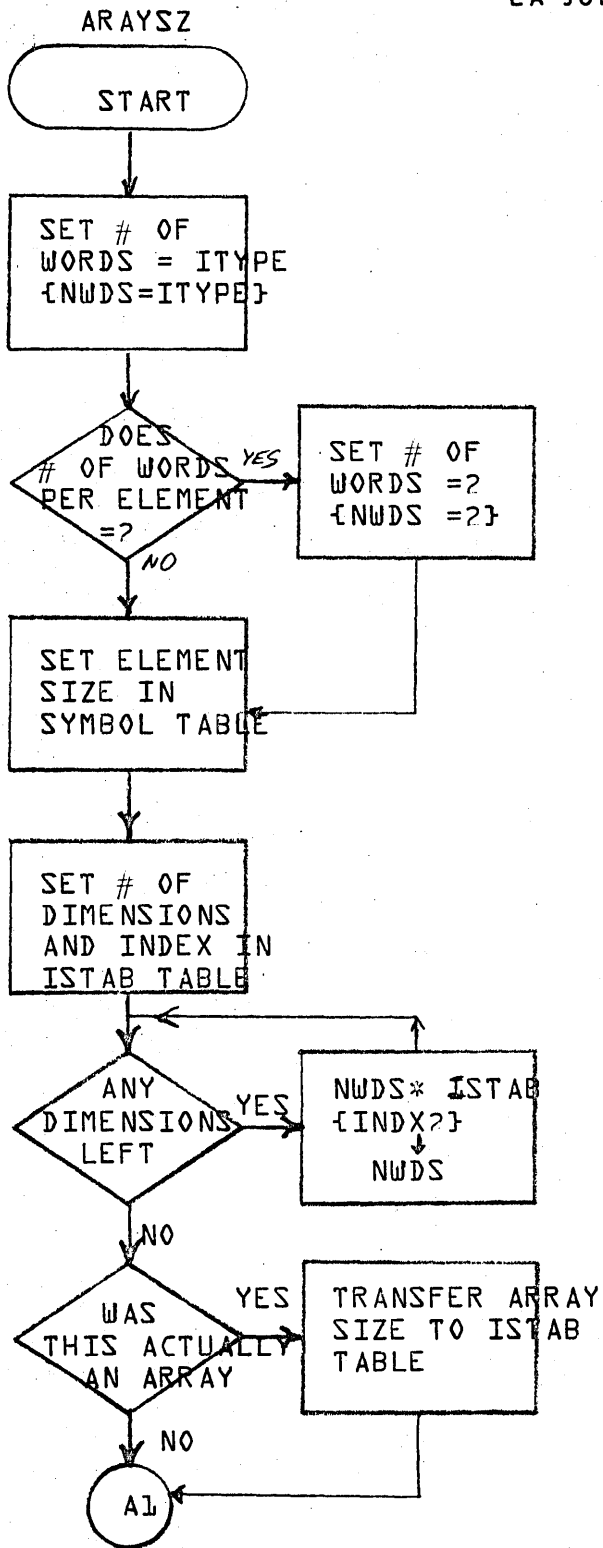
3.3 Special Subroutines for Common Allocation

3.3.1 Subroutine ARAYSZ

ARAYSZ is called to determine the number of words in any given array. The number of words is determined by the number of elements in the array times the number of words per element. If the array is real or integer with ASA options selected, the number of words per element will equal 2; if the array is double precision, the number of words per element will equal 3. ARAYSZ determines the number of words per element and sets the element size in the symbol table {KELSIZ}. It then picks up the number of dimensions and the index into the ISTAB table. The number of words in the given array is calculated and the array size is transferred to the ISTAB table at the first entry for the array.



LA JOLLA FACILITY

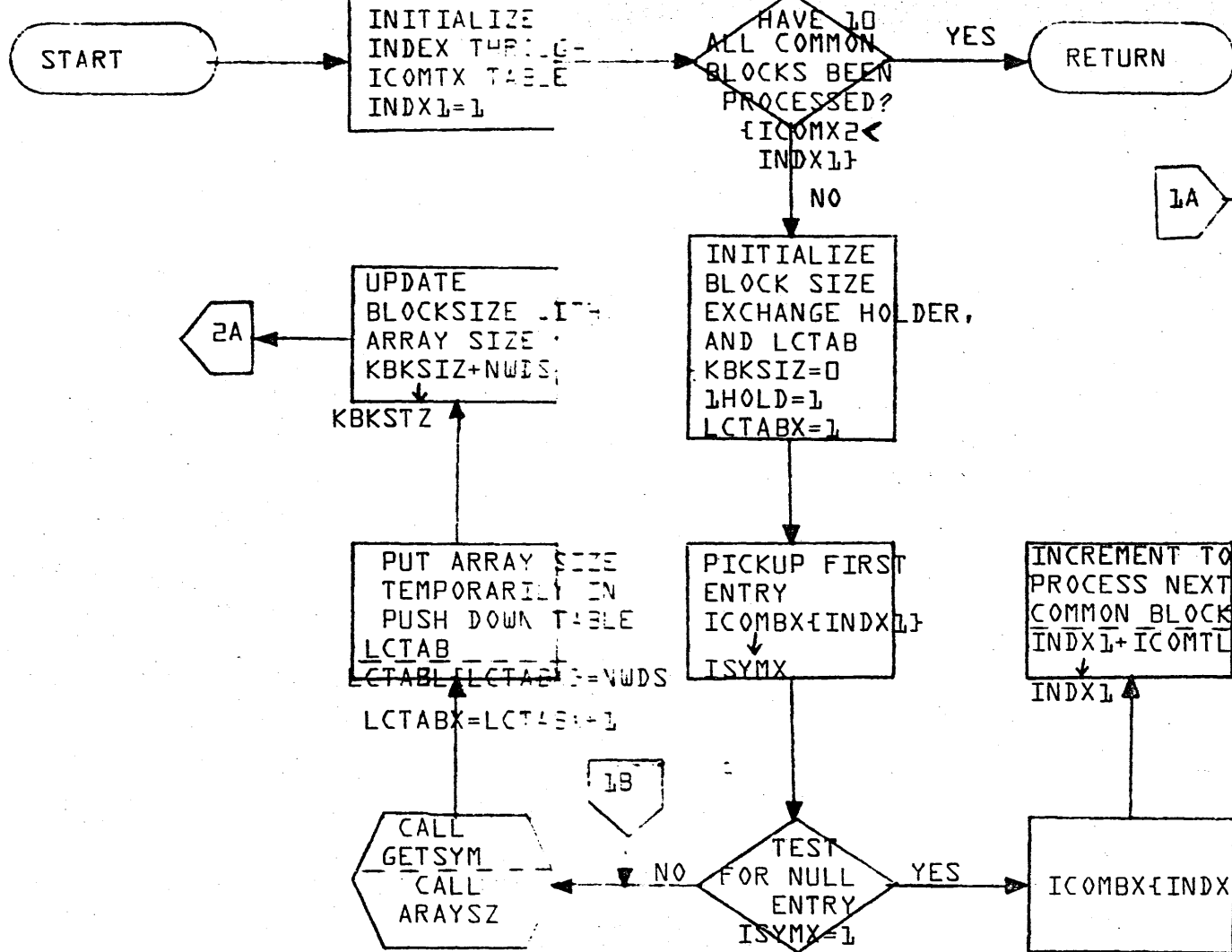


TITLE			DRG. NO.
ARAYSZ			REVISION
DRAWN BY	PROJ.	DATE	SHEET / OF 2

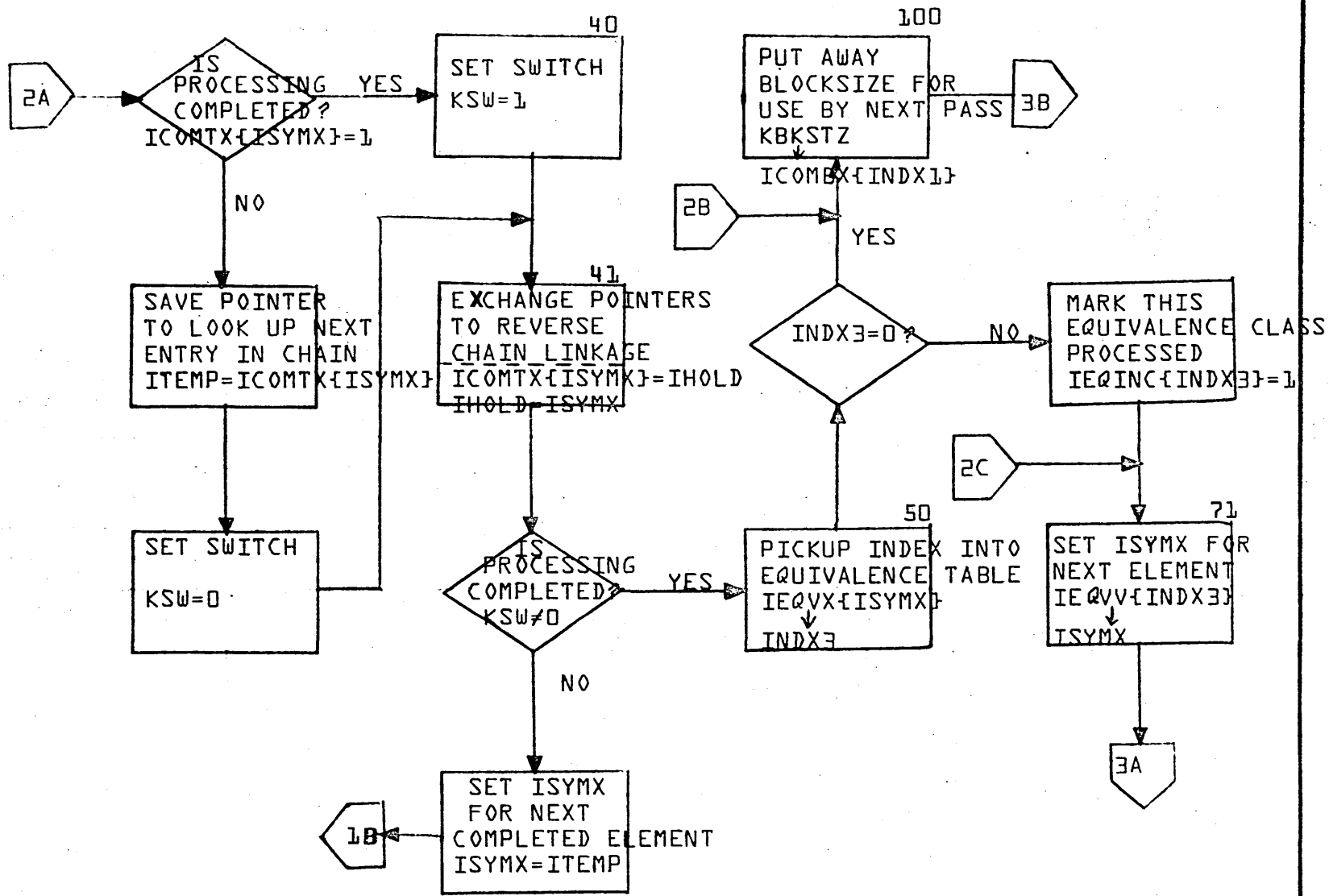
DOCUMENT CLASS IMS PAGE NO 3-3  
PRODUCT NAME 170 MASS STORAGE FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

3.3.2 Subroutine CPLOOP

CPLOOP is called by PHASEA. It allocates, in consecutive order, the space needed for each common block. CPLOOP operates in two passes, the first of which determines the size of each array in common, and the second of which allocates space. Two passes are necessary because common is strung in the reverse direction from which the COMMON statements came in. CPLOOP takes into account any EQUIVALENCE statements to variables and/or arrays in common, and lengthens the COMMON BLOCK if necessary based on these EQUIVALENCES.



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	CPLCCE			PROJECT MGR.			
	PAGE 1 OF 4				PROJECT NAME			
	NUMBER	ISSUE DATE			TASK NO.			
	DRAWN BY	DATE			TASK NAME			



**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	CPL00P	PAGE 2 OF 4		PROJECT MGR.			
NUMBER	ISSUE DATE			PROJECT NAME			
DRAWN BY	DATE			TASK NO.			
				TASK NAME			

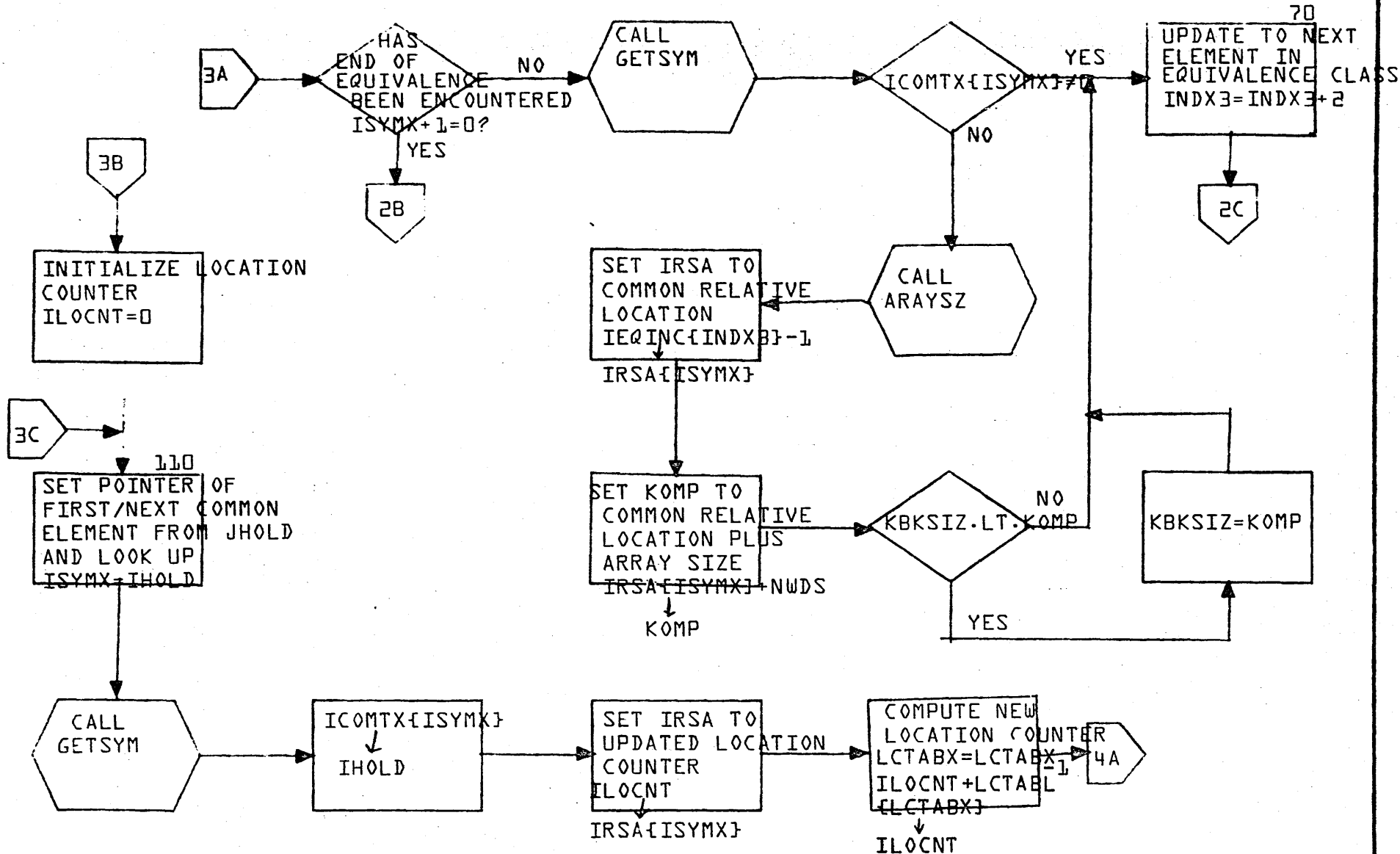
1-5

A

B

C

D



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE

FLOWCHART

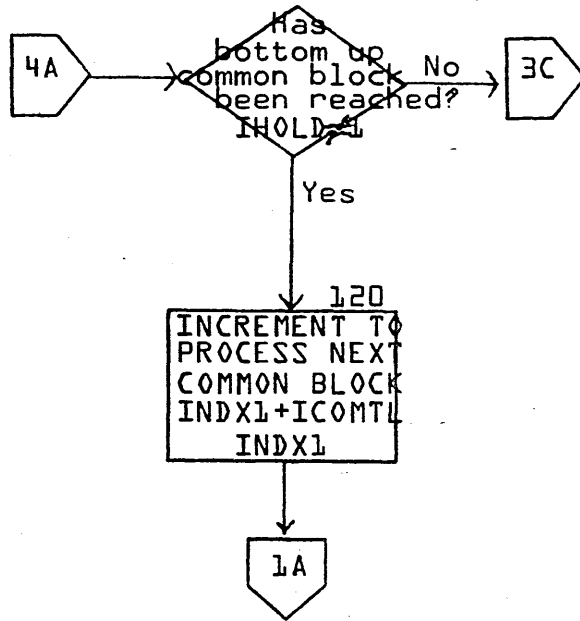
DECISION TABLE

OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	CLOOP			PROJECT MGR.			
PAGE 3 OF 4				PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.					
DRAWN BY	DATE	TASK NAME					

P-1





<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	CPLOOP			PROJECT MGR			
	PAGE 4 OF 4				PROJECT NAME			
	NUMBER	ISSUE DATE			TASK NO.			
	DRAWN BY	DATE			TASK NAME			

DOCUMENT CLASS \_\_\_\_\_ IMS \_\_\_\_\_ PAGE NO. 3-8  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN \_\_\_\_\_  
 PRODUCT MODEL NO. C005 V.2.0 \_\_\_\_\_ MACHINE SERIES 1700 \_\_\_\_\_

### 3.4.0 Arithmetic Expression Processors

#### 3.4.1 Subroutine ARITH

ARITH is the Phase A subroutine which itemizes and syntax checks arithmetic expressions.

ARITH processes an expression found in ISORS (ISORSX)\* to either end of statement or zero parenthesis level, dependent on the setting of ISTOP.

The itemization and syntax check are done concurrently and an itemized reconfiguration of the source input is built in IBUF1. If syntax errors are encountered, error diagnostics are output.

If a subscript of a variable is encountered, a call to SUBSCR is made to expand the subscript to a polynomial.

When the scan of ISORS\* is complete, IERR is checked. If IERR  $\neq$  0, ARITH exits. If IERR = 0, ARITH calls subroutine TREE to reconfigure the items in IBUF1 into a tree structure representing the arithmetic expression. In the P.0A version, ARITH passes a mixed-mode flag to TREE in the common variable LS1.

Upon return from TREE, ARITH exits to its caller. In the P.0B version, ARITH calls MODMER just before exit if mixed mode arithmetic has been encountered.

Input parameters:	ISORS*	buffer in which to find statement to analyze
	ISORSX*	- Indexer into ISORS
	ISTOP	- Switch signalling when to stop analysis 0 = end of statement 1 = zero parenthesis level (If statement 2 = process single variable (READ,WRITE)
	IBUF2	- Set to the next available word (relative to start) in the output buffer (IBUF2)
	ISTNO	- Current statement number

## CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS

IMS

PAGE NO.

3-9

PRODUCT NAME

1700 MASS STORAGE FORTRAN

PRODUCT MODEL NO.

C005 V.2.0

MACHINE SERIES

1700

Output parameters: IBUF2 - Contains the expression  
in tree form  
IBUF2X - Set to next available  
word in IBUF2  
IERR - Switch signalling  
syntax error  
ISORSX - Updated to reflect  
end of expression or  
character following zero  
parenthesis level.

Subroutines used: GETF - Get one field from ISORS  
GETSYM - Get symbol table page  
SYMBOL - Search symbol table  
STORE - Store symbol in symbol  
table

\* If AND (IFLAGS,\$10) is not zero, input is in IBUF2  
(IBUF2X). In 2.0A, there are two versions of  
ARITH.

DOCUMENT CLASS MS PAGE NO 3-10  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

ARITH Switches and Tables

Switches and tables for Syntax check

IBUF1 Table where reconfigured expression is built  
 IBUF1X IBUF1 index  
 LSI Prior symbol  
 0 = beginning of expression  
 -1 = illegal  
 TERR Syntax error switch  
 LPREN Parenthesis counter  
 ISORS Table containing input expression  
 ISORSX ISORS indexer  
 LSWITC Subroutine or function call in effect switch  
 ISTOP Input switch signalling where to stop analysis  
 0 = end of statement  
 1 = zero parenthesis level  
 2 = after processing one variable

MODE Mode of current expression  
 IWORK Mode table for exponentiation or function calls. 2  
 word entry.  
 Word 1 = mode  
 Word 2 = LPREN setting for return to mode in word 1

IMSW IWORK index  
 ITEM Temporary holder  
 IO Logical operator switch  
 LRELRO Relational operator required switch  
 IRELEN Relational operator encountered switch

Operators

GROUP	LEVEL	GROUP MEMBERS
comma	0	comma
OR	1	OR
AND	2	AND
NOT	3	NOT
Relational	4	LT, GT, LE, GE, EQ, NE
..	5	+, -
*	6	*, /
**	7	**
(	8	(, ), function

Function is a special variable treated like a set of parentheses.  
 It and its parameters are not reorderable, and can be looked at only  
 just as it sits.

DOCUMENT CLASS IMS PAGE NO 3-11  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

INVERSE operands

If operator:                      Operand is --  
    \*                      divisor  
    \*\*                     thing to exponentiate (coefficient)

SYNTAX LEGALITY TABLE (P - permissible)

	REL	OR	AND	NOT	VAR	*/**	+-	( )	,	EOS	CONST.	FUNC
RELAT	-	-	-	-	P	-	P	P	-	-	P	P
OR	-	-	-	P	P	-	P	P	-	-	P	P
AND	-	-	-	P	P	-	P	P	-	-	P	P
NOT	-	-	-	-	P	-	P	P	-	-	P	P
VARIABLE	P	P	P	-	-	P	P	P	P	P	-	-
*/**	-	-	-	-	P	-	-	P	-	-	P	P
+-	-	-	-	-	P	-	-	P	-	-	P	P
(	-	-	-	P	P	-	P	P	-	-	P	P
)	P	P	P	-	-	P	P	-	P	P	-	-
,	-	-	-	-	P	-	P	P	-	-	P	P
CONSTANT	P	P	P	-	-	P	P	-	P	P	-	-
BEGINNING	-	-	-	P	P	-	P	P	-	-	P	P
FUNCTION	-	-	-	-	-	-	-	P	P	P	-	-

F  
I  
R  
S  
T

3.4.2 Subroutine SUBSCR

SUBSCR expands a subscript of a variable into one of the following polynomials:

one dimensional array:

$$\text{subscript} = (C1 * X1 \pm D1) * E$$

two dimensional array:

$$\text{subscript} = (C1 * X1 \pm D1 + A * C2 * X2 \pm A * D2 \pm A) * E$$

DOCUMENT CLASS IMS PAGE NO 3-12  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

three dimensional array:

$$\text{subscript} = (C_1 * X_1 + A_1 * C_2 * X_2 + A_1 * A_2 * C_3 * X_3 + D_1 \\ + A_1 * D_2 - A_1 + A_1 * A_2 * D_3 - A_1 * A_2) * E$$

Where  $A_i$  is the dimension of the variable being subscripted,  $E$  is the number of words in an array element, and the permissible form of a subscript dimension is:

$$C_i X_i + D_i$$

to a maximum of three dimensions.

Depending on the type of resulting polynomial when all constant terms are collected, the variable is set to incrementally subscripted (the polynomial has no variable terms), variable subscripted (the polynomial has only a single simple variable), variable and increment subscript (the polynomial has a single simple variable and a constant term), or complex subscript. The subscript is moved to IBUF1.

Subroutines used: SYMBOL  
STORE  
GETF

#### Switches and Tables

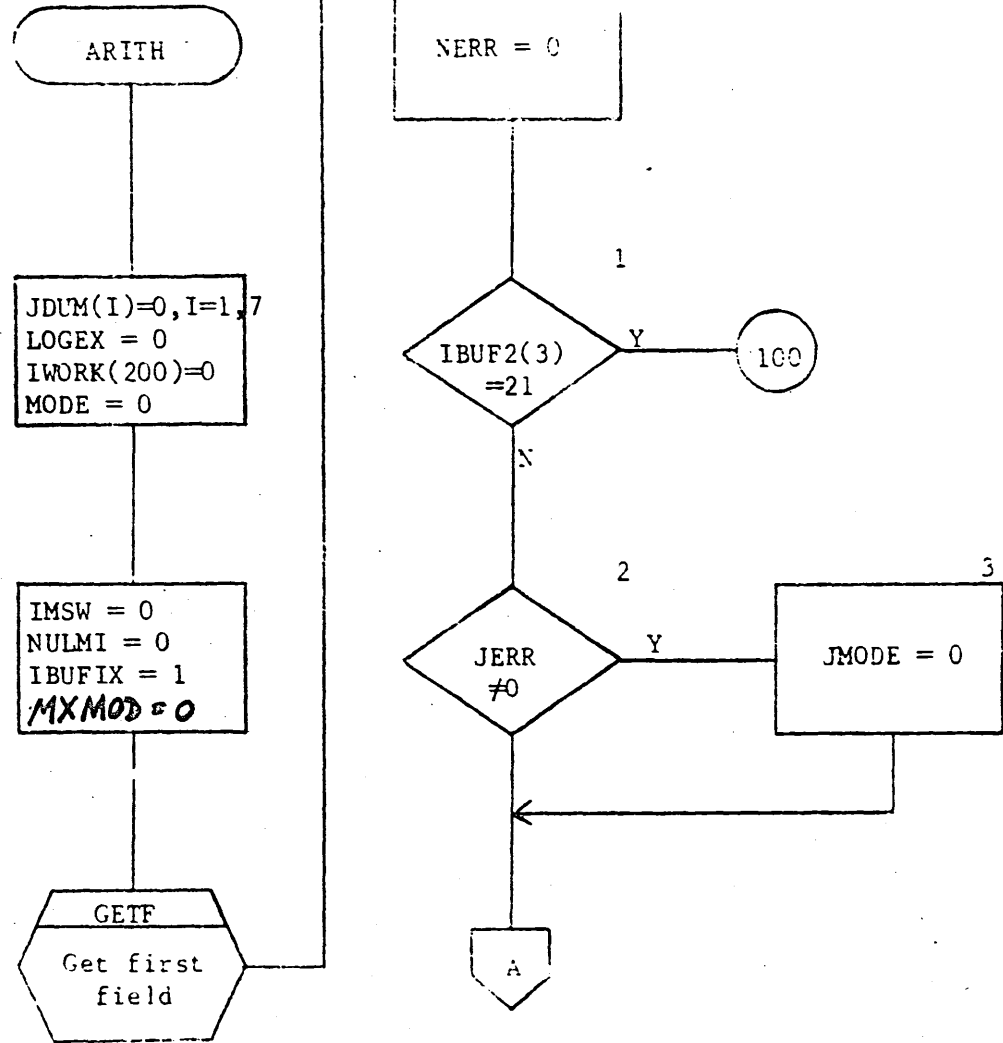
ISS	3 word table containing dimensions of subscripted variable
ISSX	current dimension
IV	IBUF1 index of variable associated with subscript
IVM	Word length of variable associated with subscript
ID1	3 word table of constant terms in subscript
IC1	3 word table of constant multipliers in subscript
IX1	3 word table of variables in subscript
ISFLAG	+, - flag for subscript terms

- 1. CALL statement.
- 2. Error in field.
- 3. Set to null field.

This routine syntax checks the source statement and stores it in pre-tree form in IBUF1 for TREE to process.

IBUF1 Form

- WORD 1 Type
- WORD 2 level
- WORD 3 symbol table pointer (if necessary).



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE  
FLOWCHART  
DECISION TABLE  
OTHER

X

DOCUMENT CLASS	<i>IMS</i>	MACH TYPE	<i>1900</i>	PROJECT NO	REV	APPROVED	DATE
DOCUMENT TITLE	<i>ARITH</i>			PROJECT MGR			
		PAGE	<i>1 of 35</i>	PROJECT NAME			
NUMBER		ISSUE DATE		TASK NO			
DRAWN BY		DATE		TASK NAME			





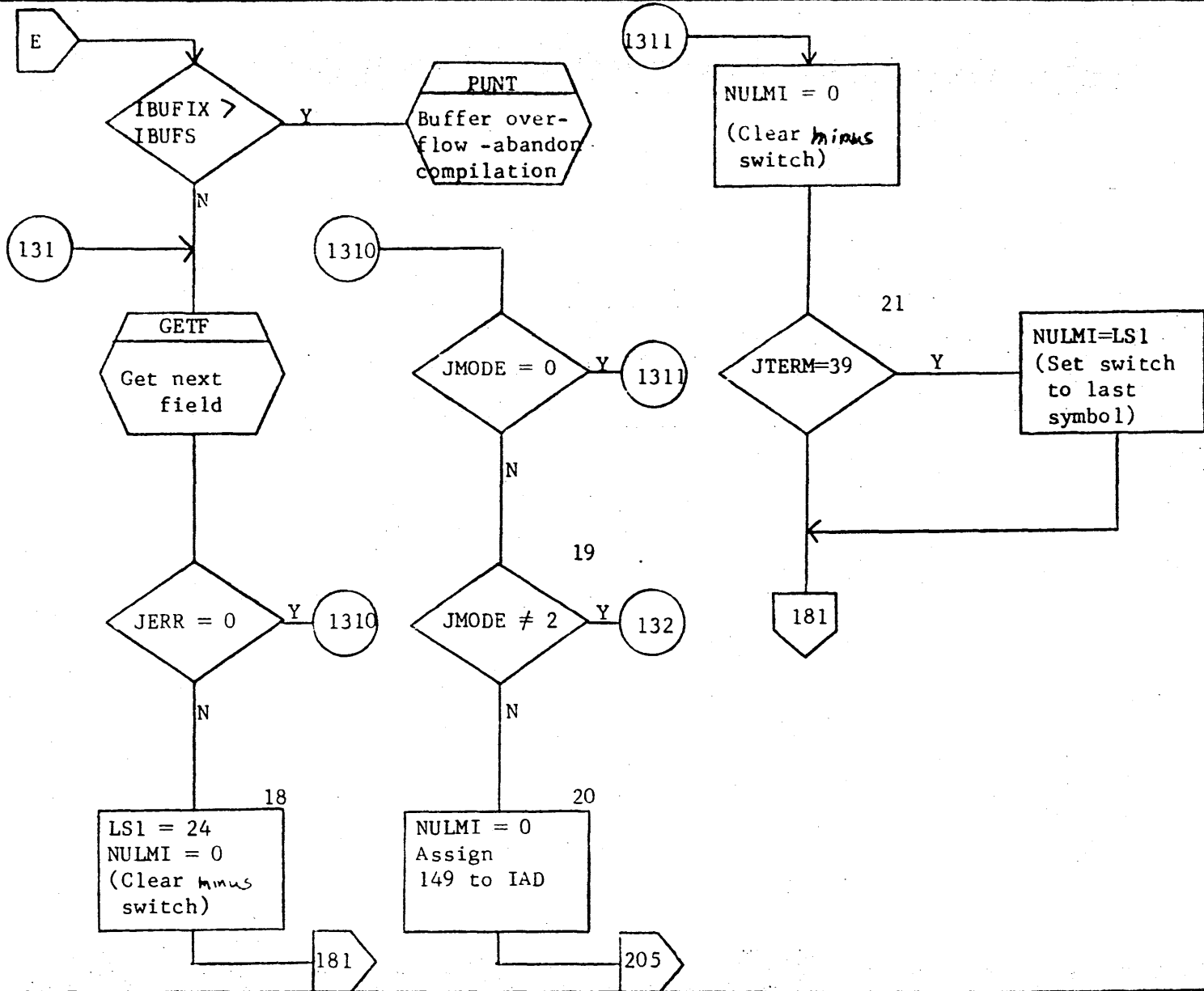


18. If error set last symbol to nonsubscripted variable and go to identify terminator.

19. Must be symbol field or go to constant processor.

20. Put symbol in symbol table if not already there.

21. Check for "-" terminator..



CONTROL DATA CORPORATION

SOFTWARE DOCUMENT

SAMPLE CODE

FLOWCHART

DECISION TABLE

OTHER

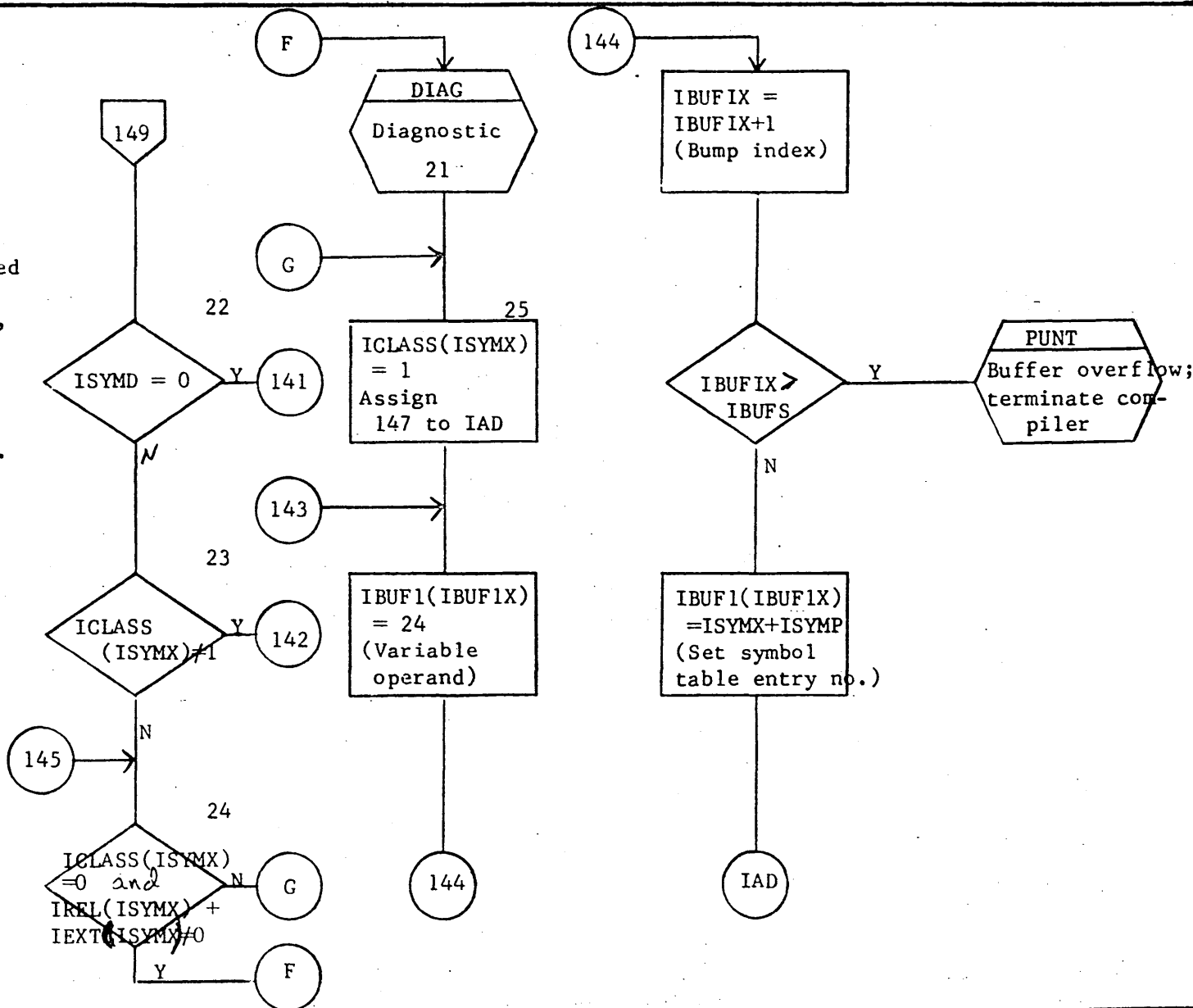
DOCUMENT CLASS	<i>IMS</i>	MACH TYPE	<i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	<i>ARITH</i>	PAGE <i>4</i> OF <i>35</i>		PROJECT MGR.			
NUMBER	ISSUE DATE			PROJECT NAME			
DRAWN BY	DATE			TASK NO.			
				TASK NAME			

22. If the field is not in SYMTAB, check for "(".

23. If class is not a variable, must be "(", ")", or ",".

24. If class is unassigned and symbol is either relative or external, a variable has been used as a subprogram name.

25. Set class = variable.



**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

SAMPLE CODE

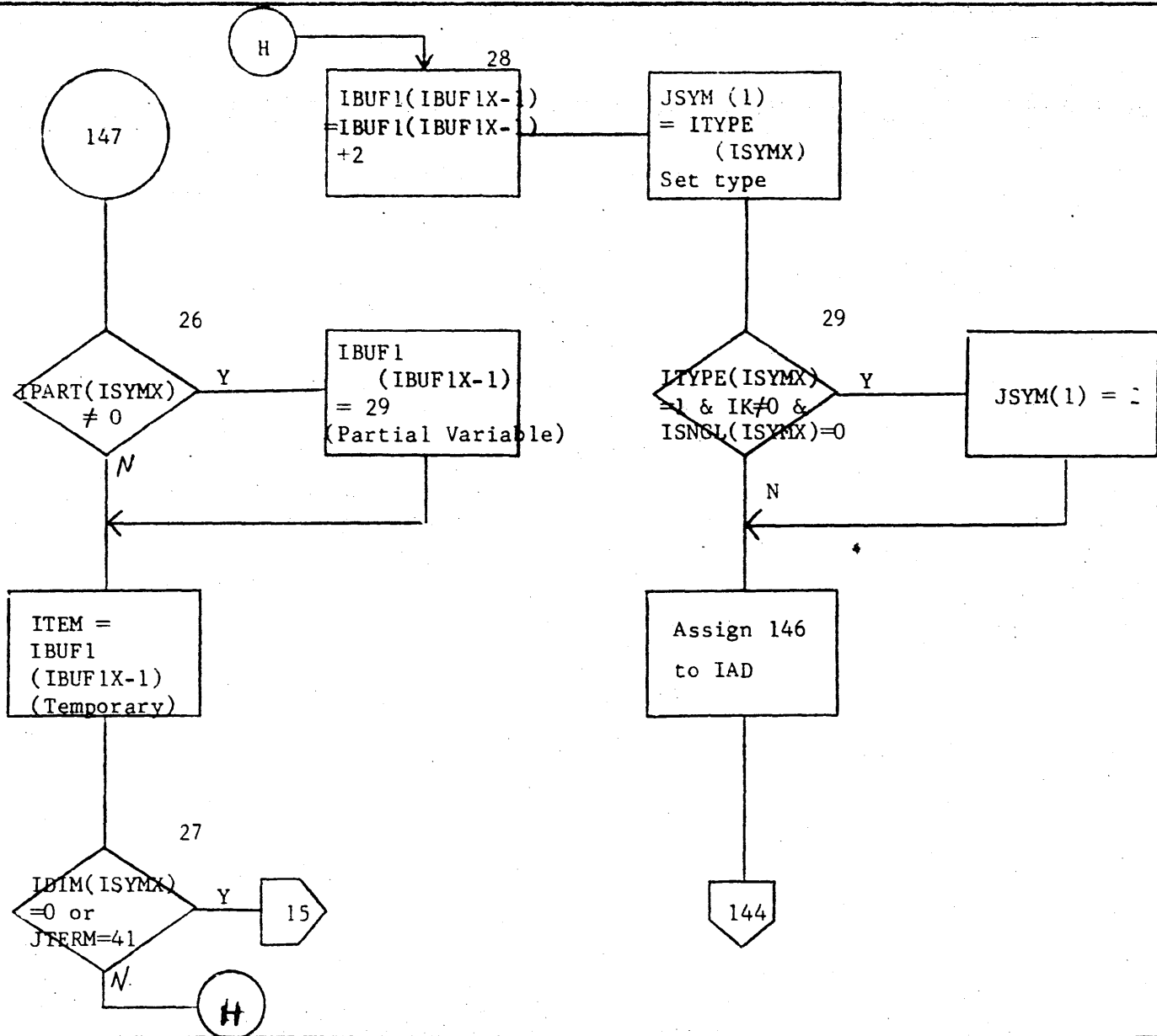
FLOWCHART

DECISION TABLE

OTHER

DOCUMENT CLASS <i>IMS</i>	MACH. TYPE <i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE <i>ARITH</i>		PROJECT MGR.			
	PAGE <i>5</i> OF <i>35</i>	PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.			
DRAWN BY	DATE <i>5/1/77</i>	TASK NAME			

26. Check partial variable.
27. If dimensional or "(" go check predecessor.
28. Dimensioned so set to increment only variable.
29. Two words/entry if non-integer or integer and ASA selected and SINGLE not selected.



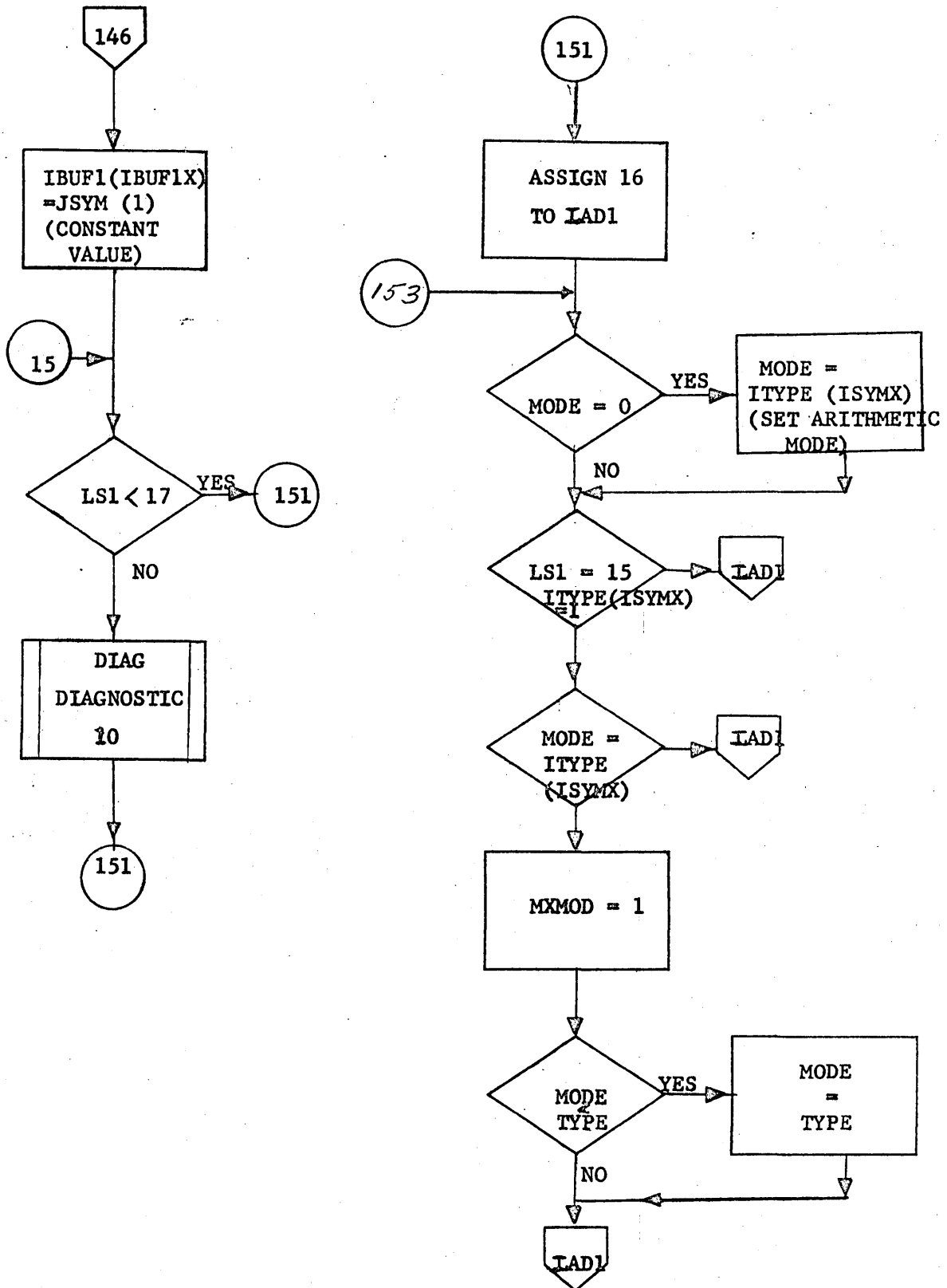
CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	ARITH	PAGE 6 OF 35		PROJECT MGR.			
NUMBER	ISSUE DATE			PROJECT NAME			
DRAWN BY		DATE 3-6-1		TASK NO.			
				TASK NAME			

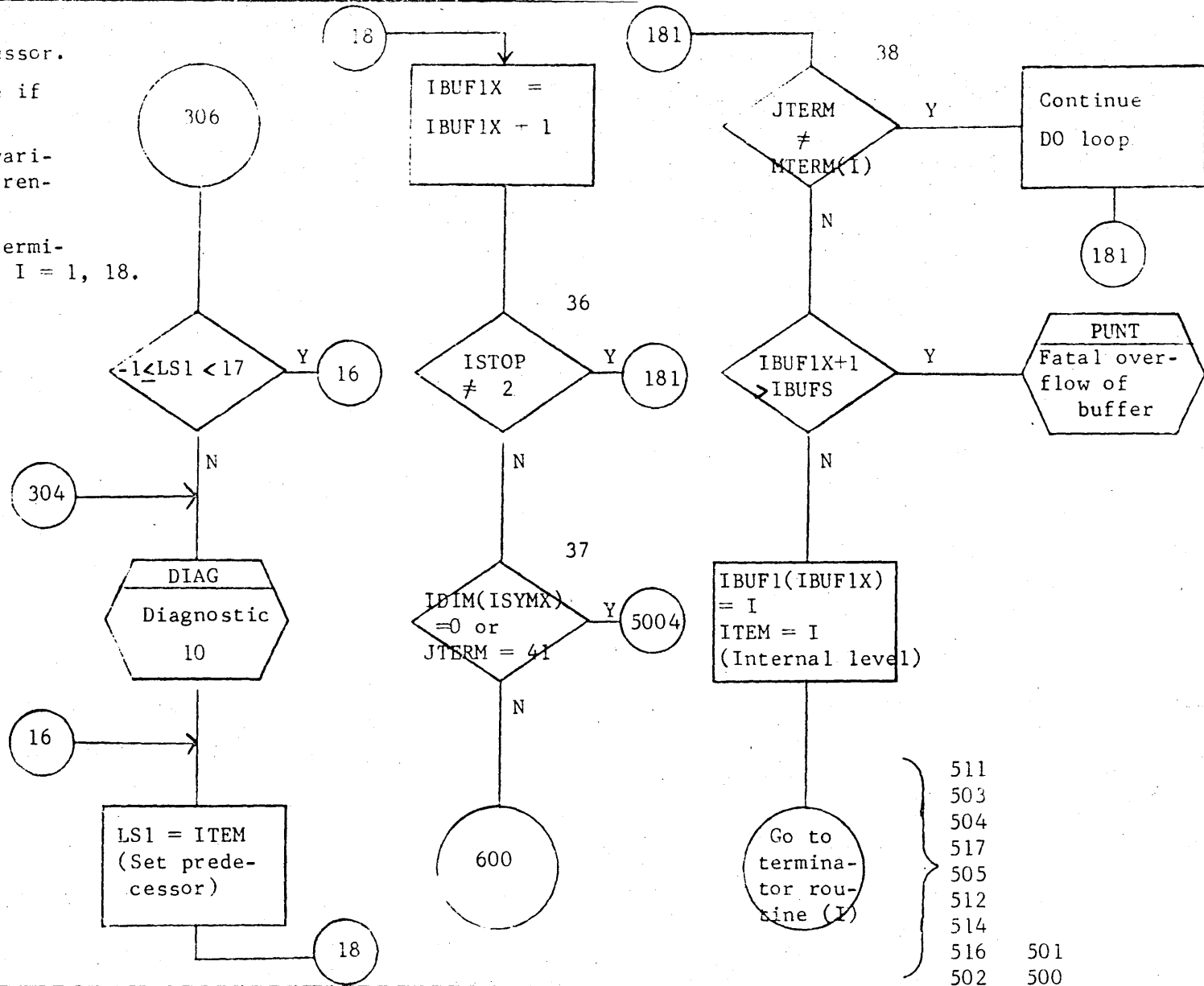


LA JOLLA FACILITY



TITLE		ARITH		DRG. NO.	
DRAWN BY		PROJ.	DATE	REVISION	
CSD 203				SHEET 2	OF 3

- 35. Illegal predecessor.
- 36. Single variable if ISTOP=2.
- 37. Undimensioned variable or left parenthesis.
- 38. Check against terminator table for I = 1, 18.



CONTROL DATA CORPORATION

SOFTWARE DOCUMENT

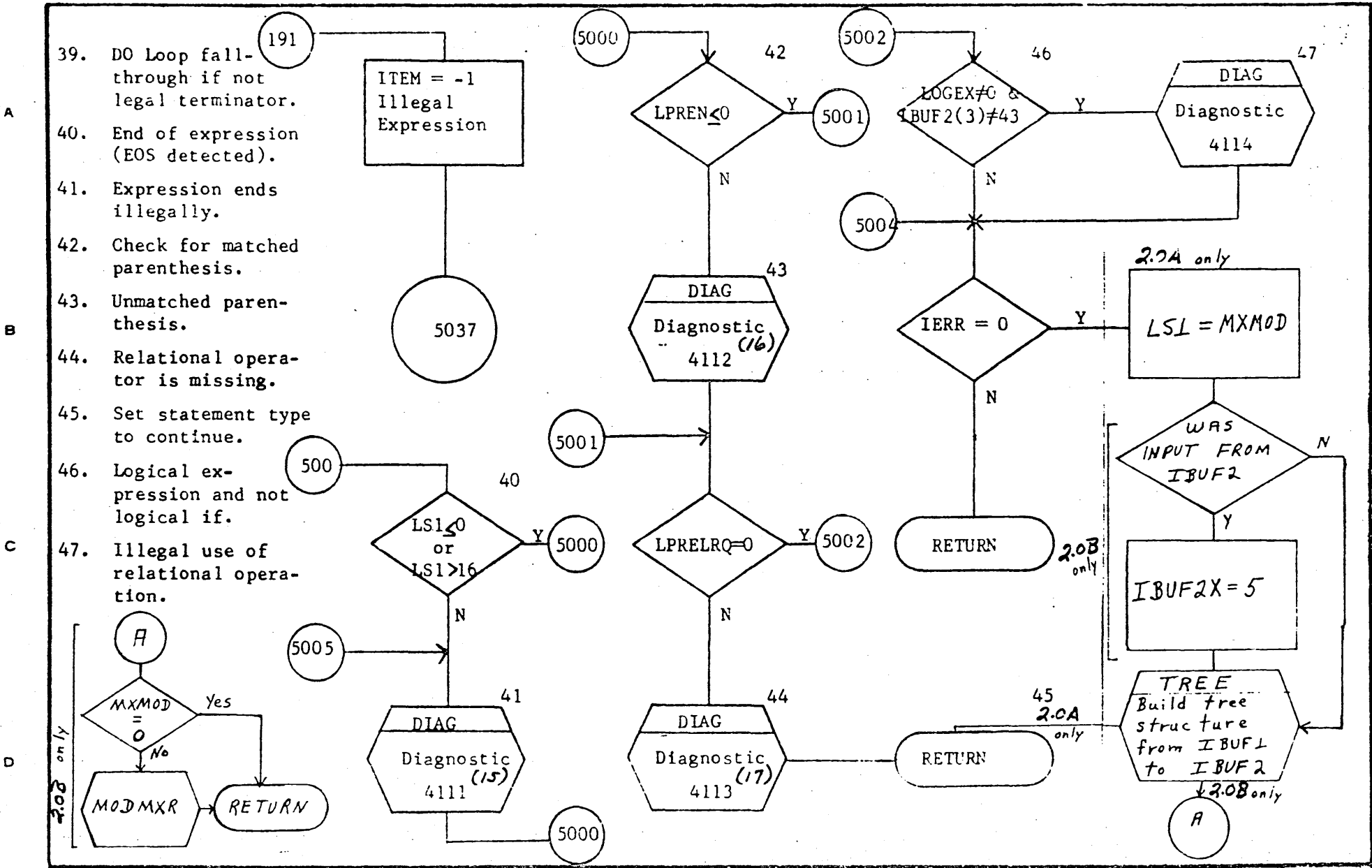
SAMPLE CODE

FLOWCHART

DECISION TABLE

OTHER

DOCUMENT CLASS	IMS	MACH TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	ARITH	PAGE 8 OF 35		PROJECT MGR.			
NUMBER	ISSUE DATE			PROJECT NAME			
DRAWN BY		DATE		TASK NO.			
				TASK NAME			



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

DOCUMENT CLASS *ZMS* MACH. TYPE *1100*

PROJECT NO.

REV. APPROVED DATE

SAMPLE CODE  
PLA CHART  
DEFINITION TABLE

DOCUMENT TITLE *ARITH*

PAGE *9* OF *35*

NUMBER ISSUE DATE

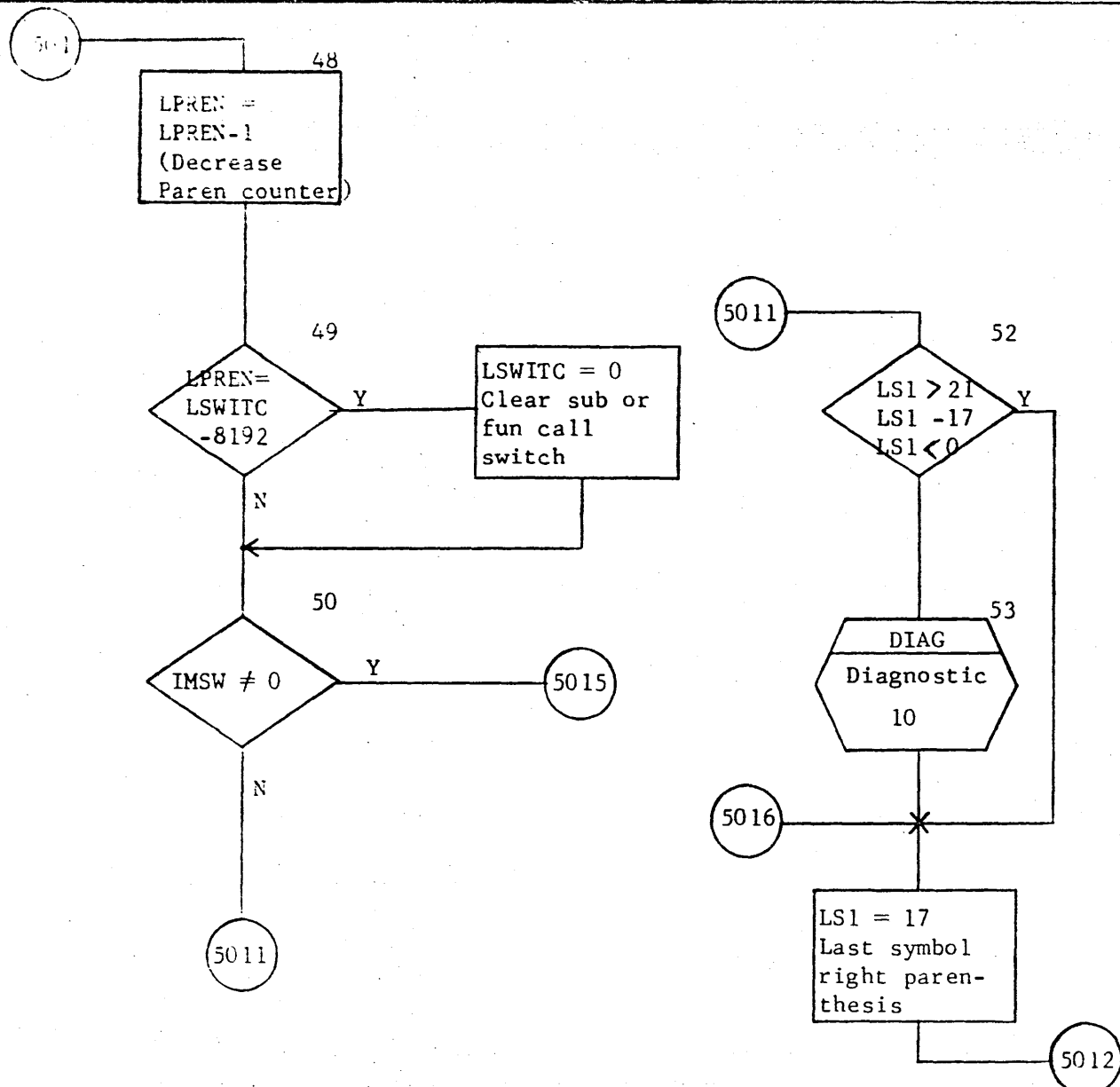
PROJECT MGR

PROJECT NAME

TA - 100

- 48. ")" detected.
- 49. Parenthesis level the same?
- 50. No entries in mode control table?

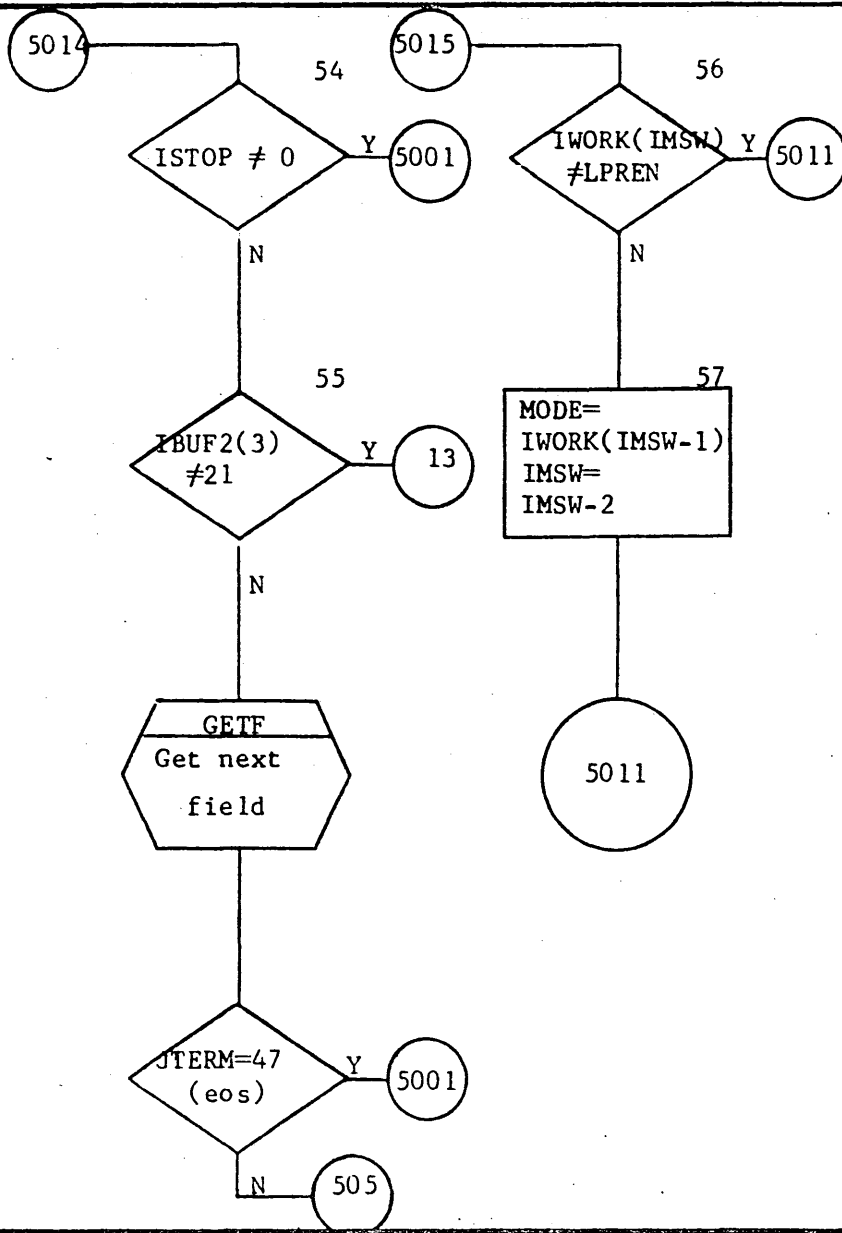
- 52. Check predecessor, variable, constant, no parameter function or subroutine legal.
- 53. Illegal operator or operand.



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS <i>IMS</i>	MACH. TYPE <i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE <i>ARITH</i>		PROJECT MGR.			
		PAGE <i>10</i> OF <i>35</i>	PROJECT NAME			
	NUMBER	ISSUE DATE	TASK NO.			
	DRAWN BY	DATE <i>6-7</i>	TASK NAME			



54. Test for end of statement.  
 55. IBUF(3)=21 means CALL statement.  
 56. Parenthesis level the same?  
 57. Yes. Set mode and decrease IWORK index.

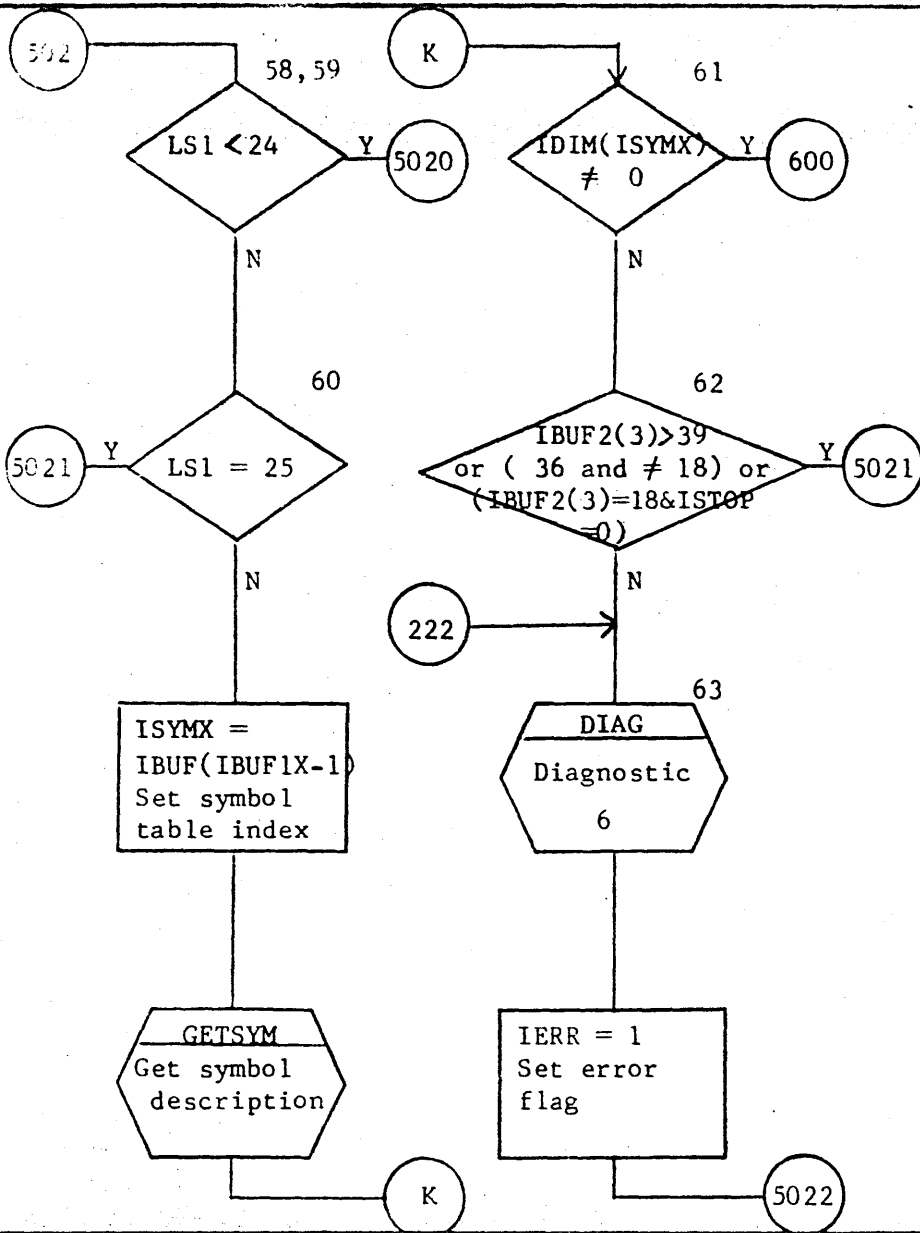


**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

DOCUMENT CLASS <i>IMS</i>	MACH. TYPE <i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE <i>ARITH</i>		PROJECT MGR.			
	PAGE <i>11</i> OF <i>35</i>	PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.			
DRAWN BY	DATE <i>1-1</i>	TASK NAME			

58. Left parenthesis processor.  
 59. Check predecessor a variable.  
 60. Predecessor is a variable. Is it a numeric constant?  
 61. Dimensioned variable?  
 62. No. Check if not read, write, or replacement, or replacement and processing to end of statement.  
 63. Undimensioned subscripted variable.



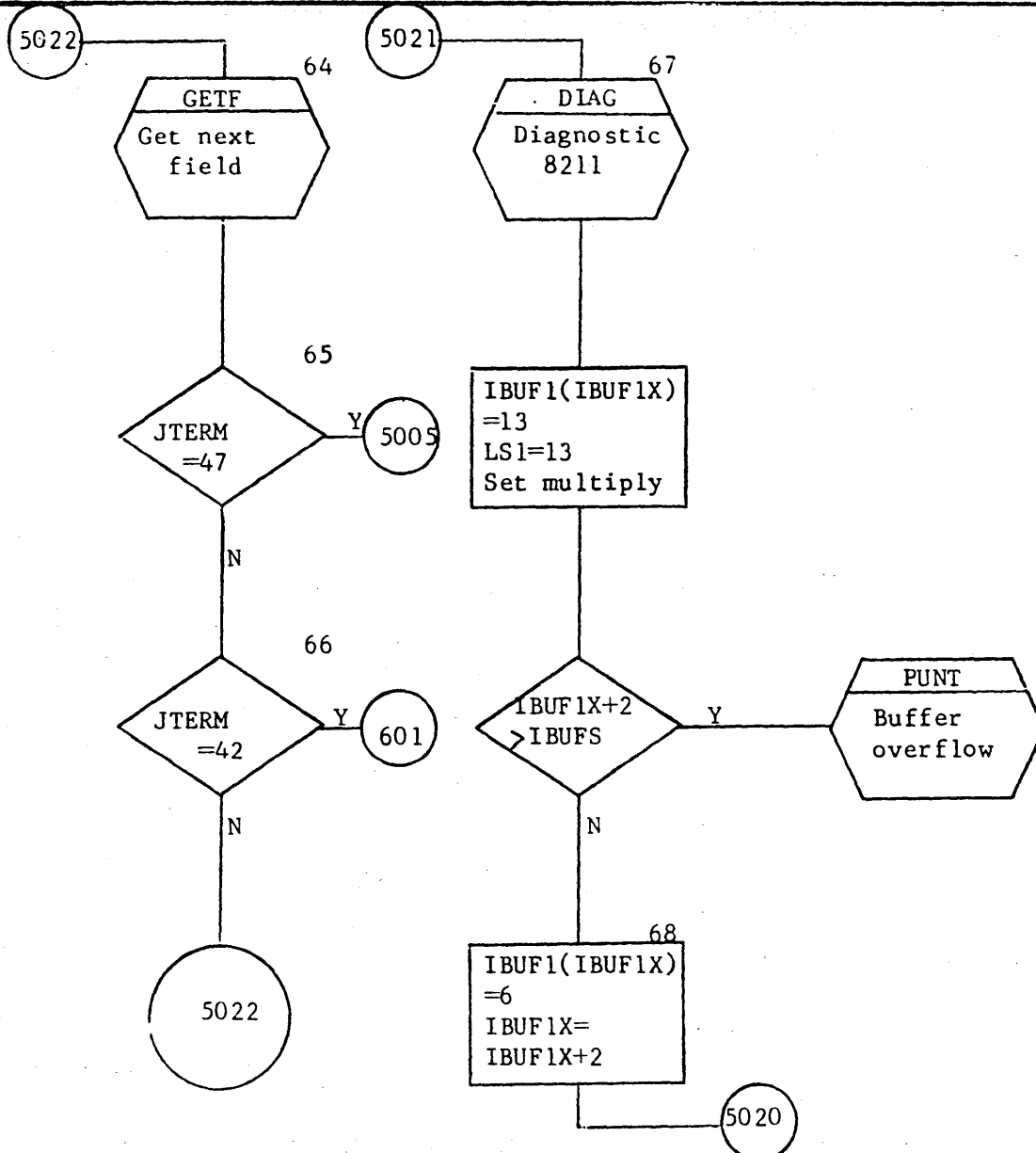
**CONTROL DATA CORPORATION**

SOFTWARE DOCUMENT

- SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

DOCUMENT CLASS <i>IMS</i>	MACH. TYPE <i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE <i>ARITH</i>		PROJECT MGR.			
	PAGE <i>12</i> OF <i>35</i>	PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.			
DRAWN BY	DATE	TASK NAME			

- 64. Slew to right parenthesis.
- 65. End of statement?
- 66. Right parenthesis.
- 67. \* assumed.
- 68. Set level 6 and bump index.



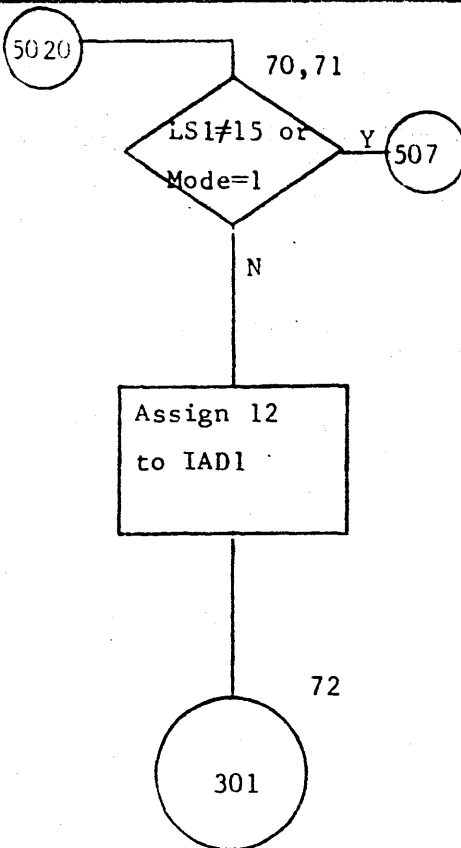
<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	MACH. TYPE	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE		PROJECT MGR.			
		PAGE	PROJECT NAME			
	NUMBER	ISSUE DATE	TASK NO.			
	DRAWN BY	DATE	TASK NAME			

69.

70. Entry if nonvariable  
predecessor.

71. Check predecessor is  
\*\* or mode is integer.

72. Go make IWORK entry.



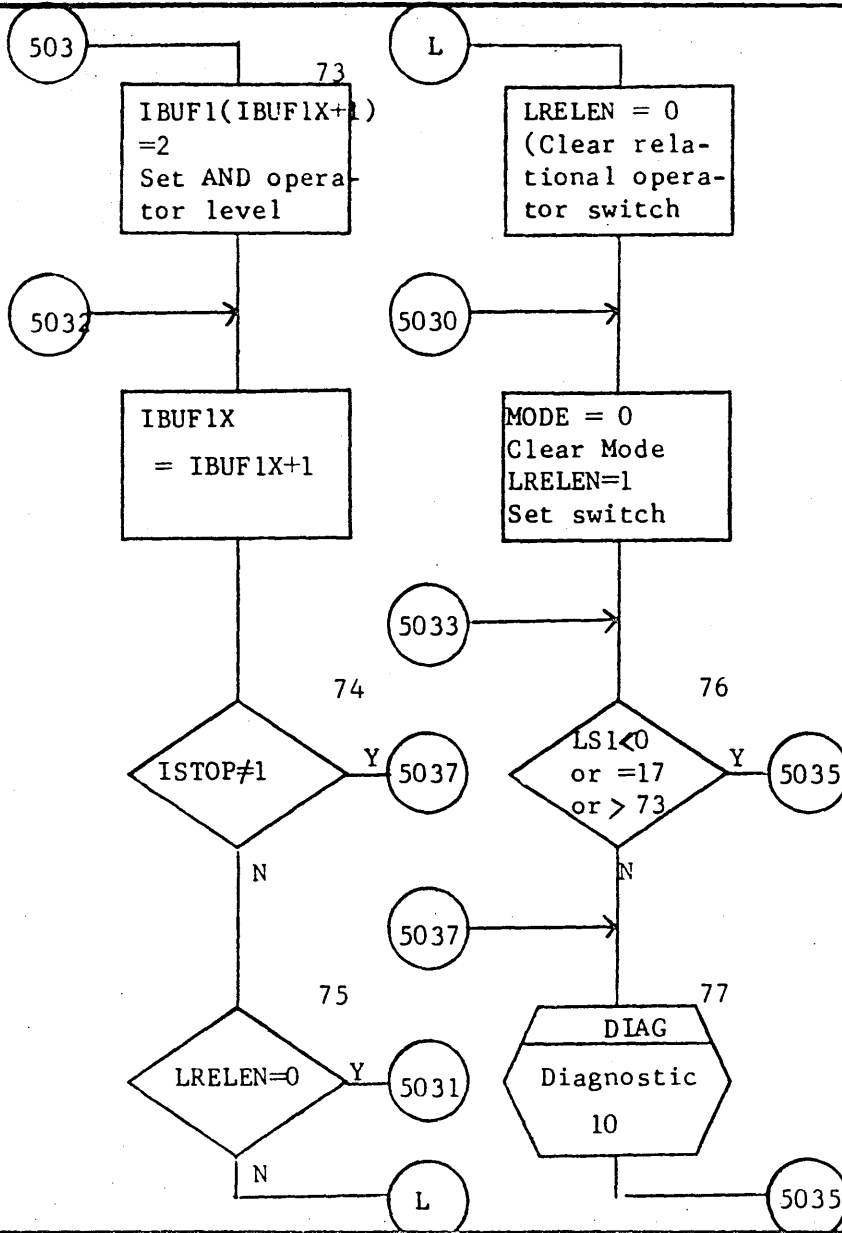
**CONTROL DATA CORPORATION**

SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS <i>TMS</i>	MACH. TYPE <i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE <i>ARITH</i>		PROJECT MGR.			
PAGE <i>14</i> OF <i>35</i>		PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.			
DRAWN BY	DATE <i>5-6-7</i>	TASK NAME			

- 73. AND processor.
- 74. Only OK in logical if.
- 75. Relational operator encountered switch on?
- 76. Check if last symbol ")" or constant or variable.
- 77. Illegal operator or operand.

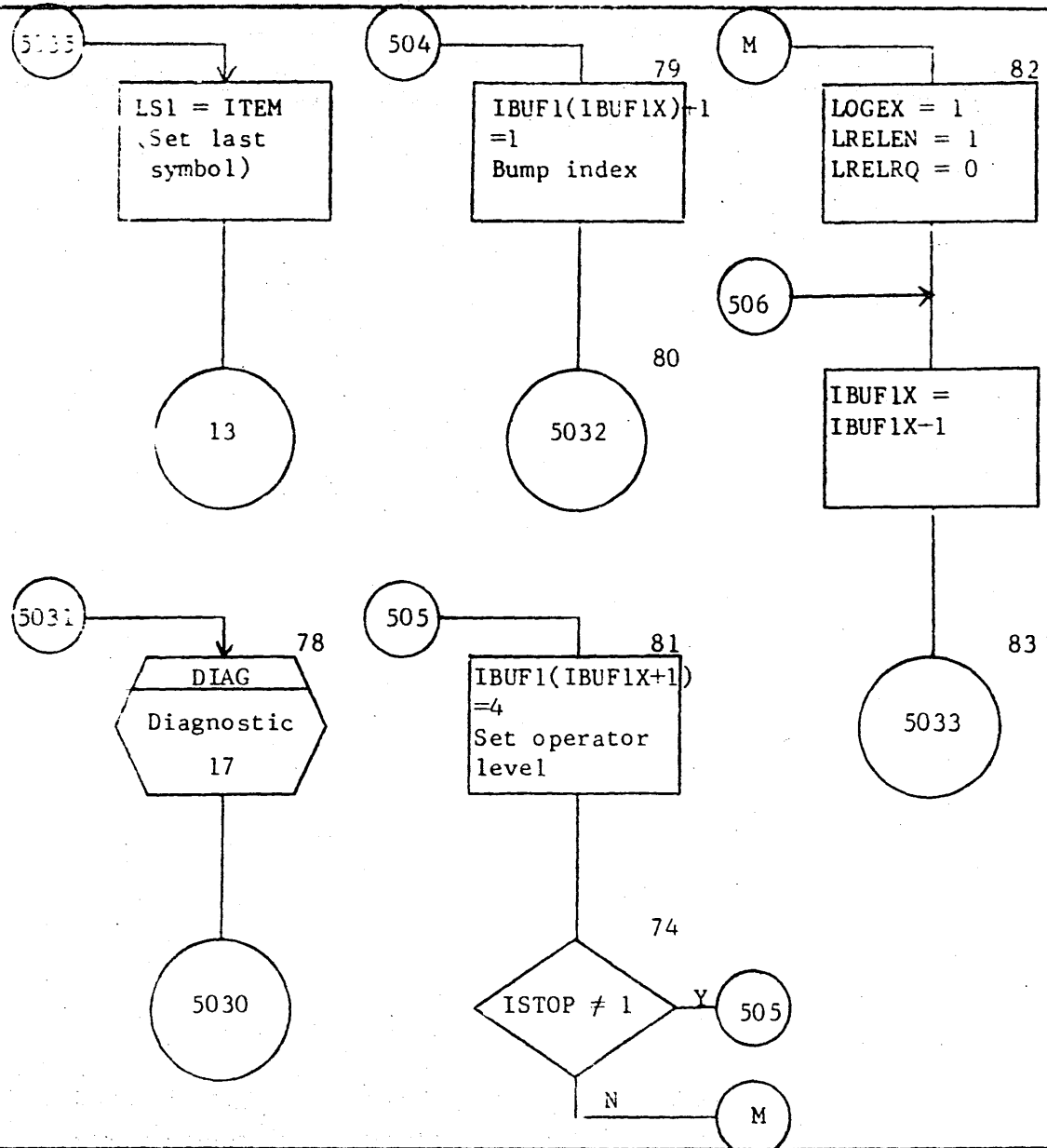


**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS <i>IMS</i>	MACH. TYPE <i>1100</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE <i>ARITH</i>		PROJECT MGR.			
	PAGE <i>15</i> OF <i>35</i>	PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.			
DRAWN BY	DATE	TASK NAME			

- 78. Relational operator missing.
- 79. OR processor.
- 80. Same as AND.
- 81. >, <, ≤, ≥, =, ≠ Processor.
- 82. Set logical switch, Relational operator encountered switch. Clear relational operator required switch.
- 83. Same as AND and OR at this point.

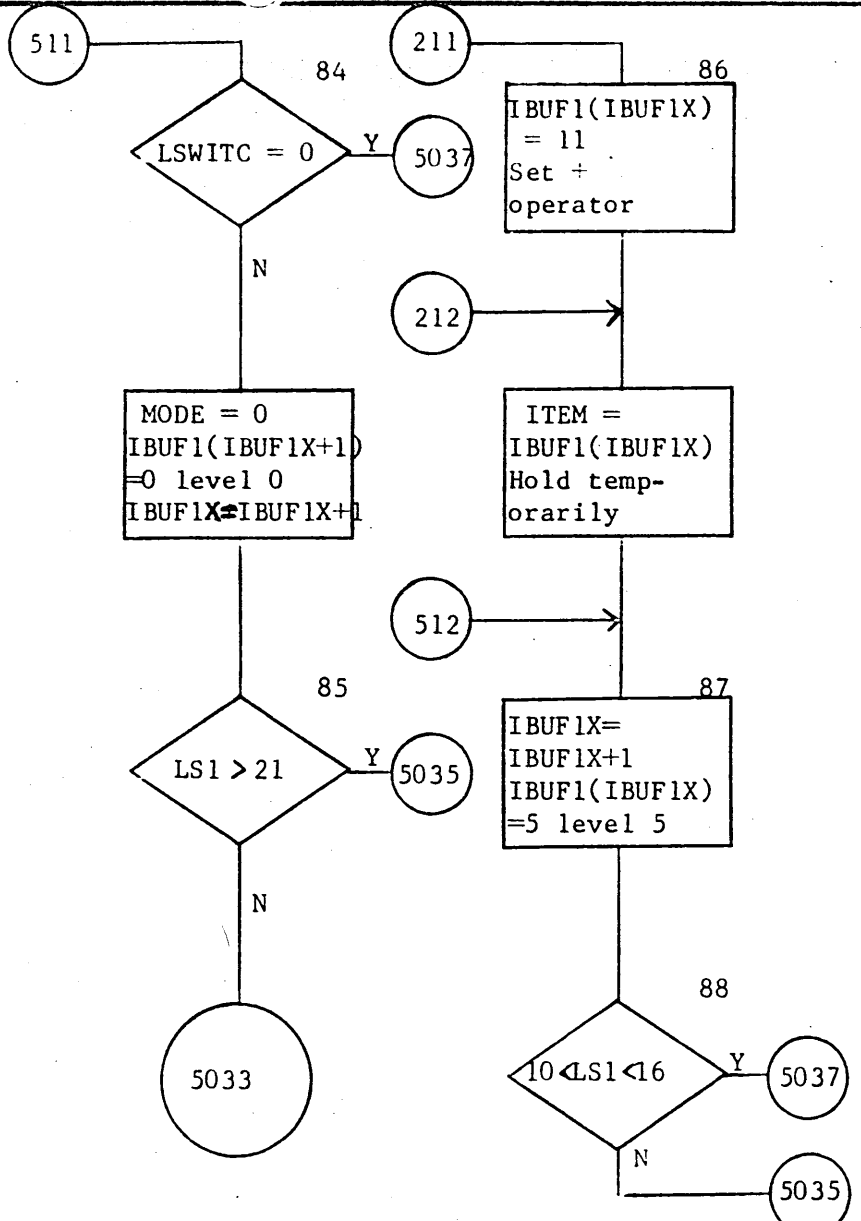


**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	MAC- TYPE	PROJECT NO.	REV	APPROVED	DATE
IMS	1700				
DOCUMENT TITLE		PROJECT MGR.			
ARITH					
	PAGE 16 OF 35	PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.			
DRAWN BY	DATE	TASK NAME			

- 84. Comma Processor -  
Illegal if not function  
or subroutine.
- 85. Check for function or  
subroutine without  
argument.
- 86. Plus processor - first  
field.
- 87. "+, -" processor.
- 88. Check illegal predecessors  
- +, -, \*\*, \*, /.

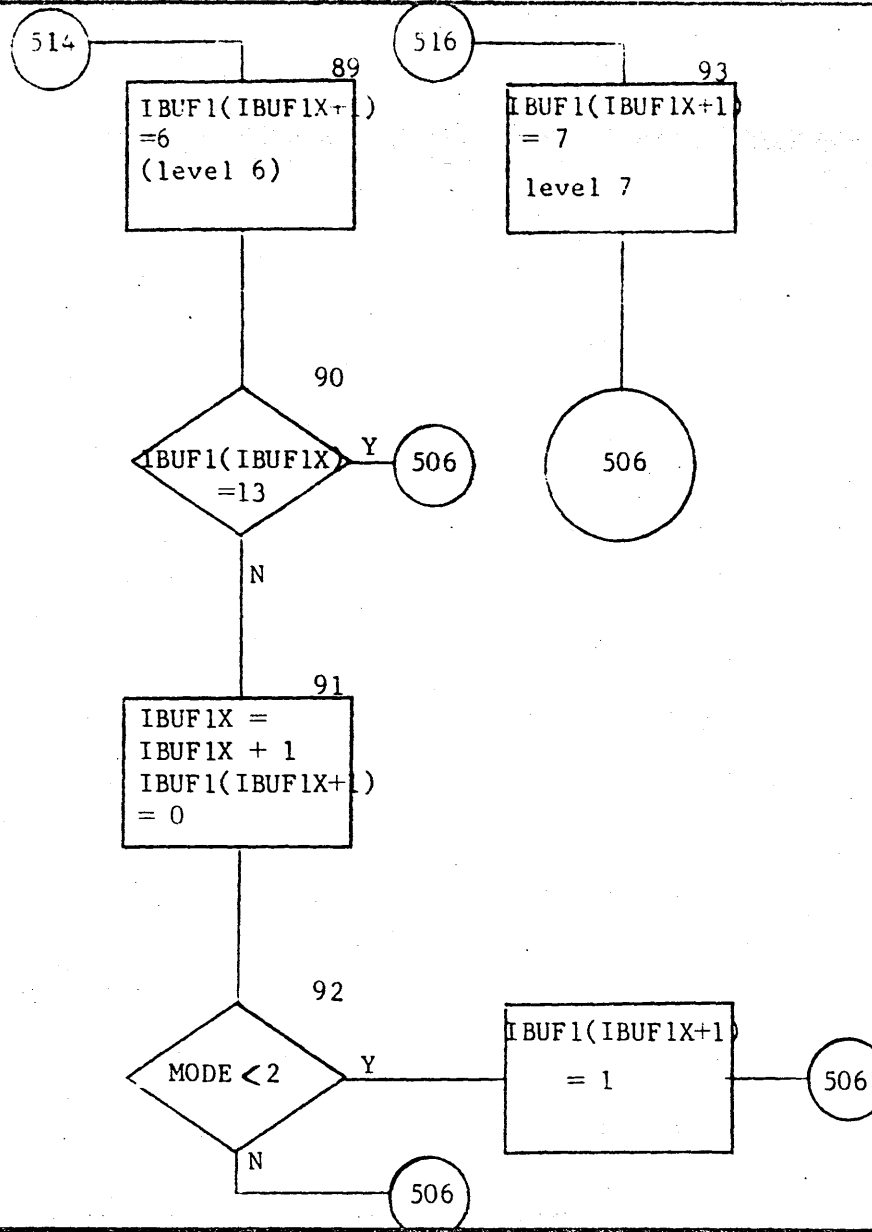


CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE   
FLOWCHART   
DECISION TABLE   
OTHER

DOCUMENT CLASS <i>TMS</i>	MACH. TYPE <i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE <i>ARITH</i>		PROJECT MGR			
	PAGE <i>17</i> OF <i>35</i>	PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.			
DRAWN BY	DATE	TASK NAME			

- 89. "\*/" processor.
- 90. Check for \*\*.
- 91. Clear symbol table pointer.
- 92. Check mode unassigned or integer.
- 93. "\*\*\*" processor.



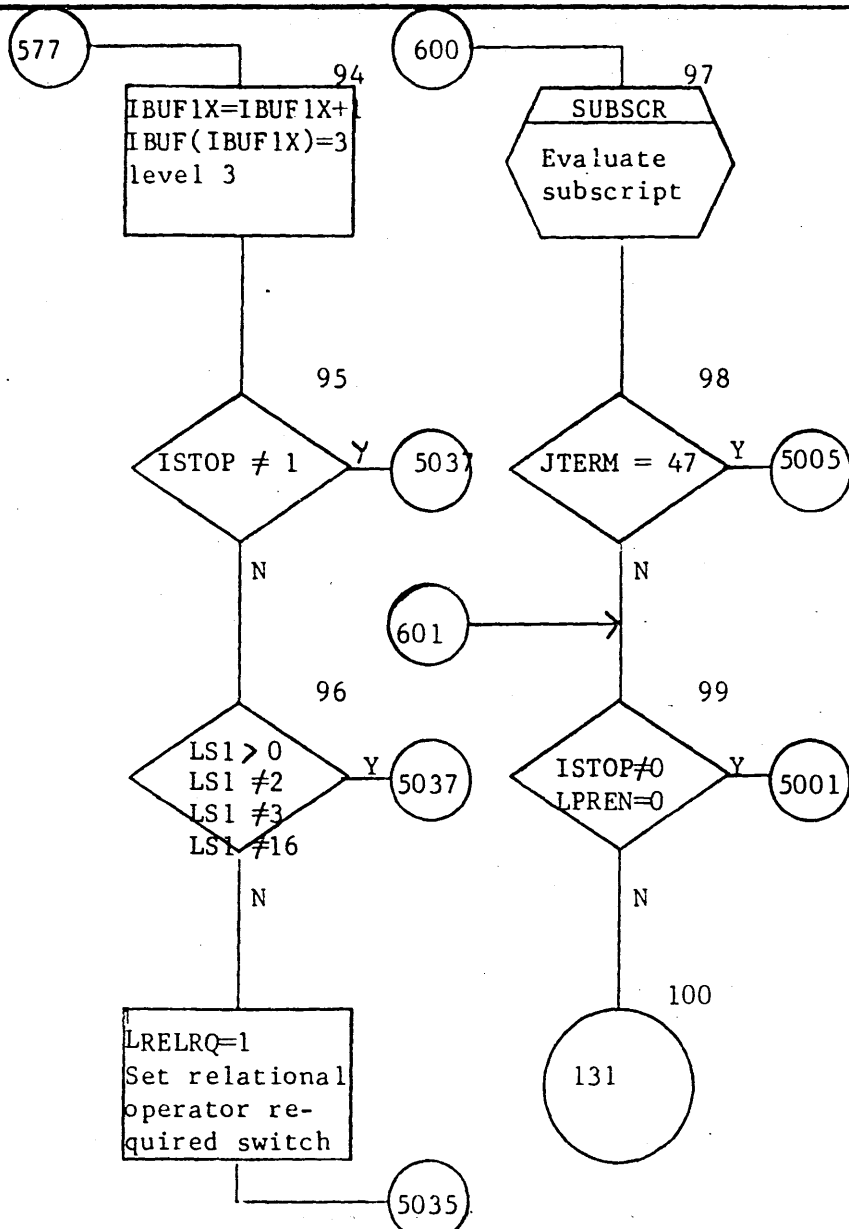
CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE  
FLOWCHART   
DECISION TABLE   
OTHER

DOCUMENT CLASS	<i>ZMS</i>	MACH. TYPE	<i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	<i>ARITH</i>			PROJECT MGR.			
		PAGE	<i>18 of 35</i>	PROJECT NAME			
NUMBER		ISSUE DATE		TASK NO.			
DRAWN BY		DATE	<i>5</i>	TASK NAME			



- 94. "NOT" processor.
- 95. Must be if statement.
- 96. Legal predecessors are AND, OR, left parenthesis.
- 97. Dimensioned variable.
- 98. Check for end of statement.
- 99. End of parenthesis means put statement in TREE.
- 100. Go process next field.

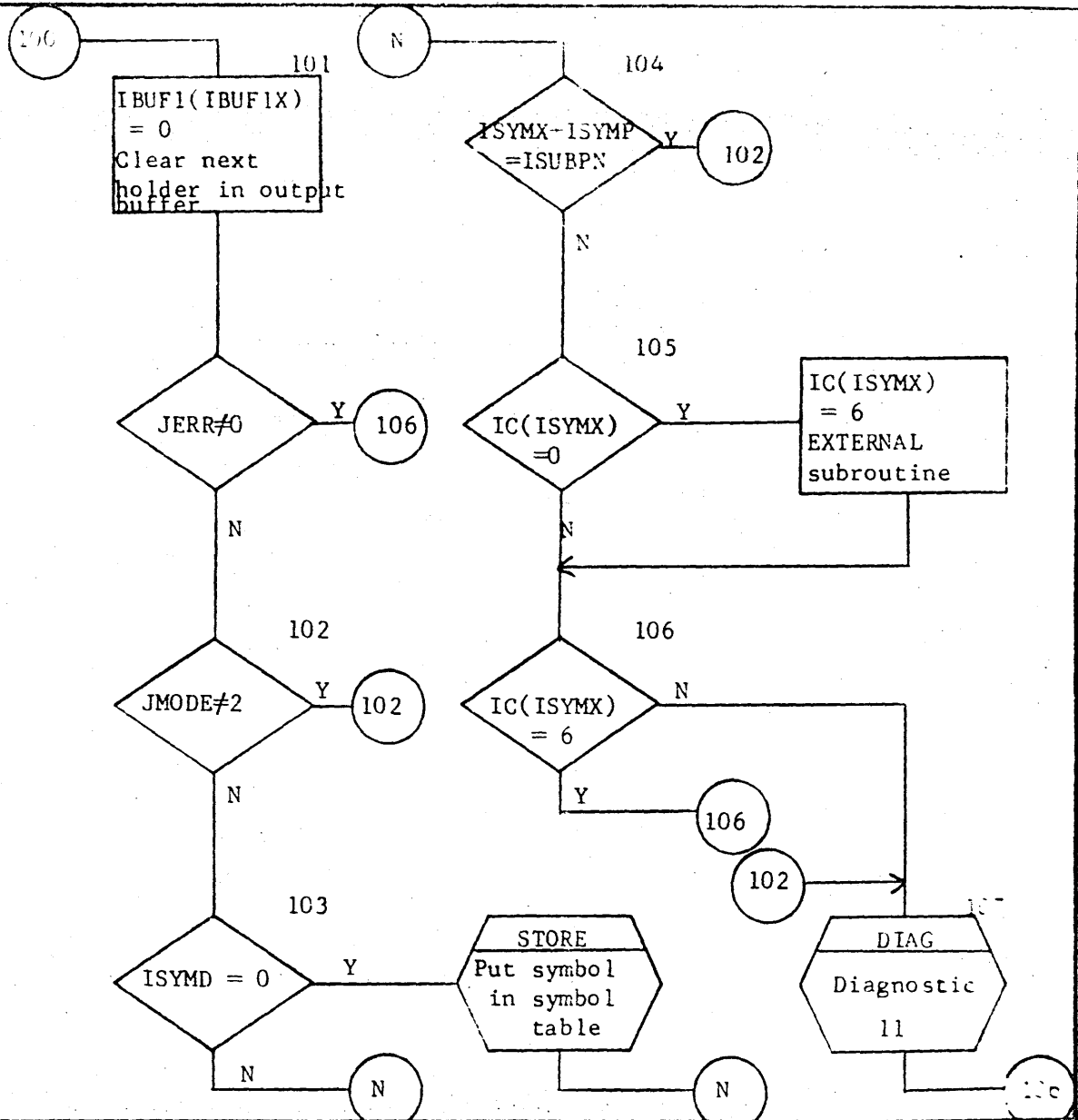


CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

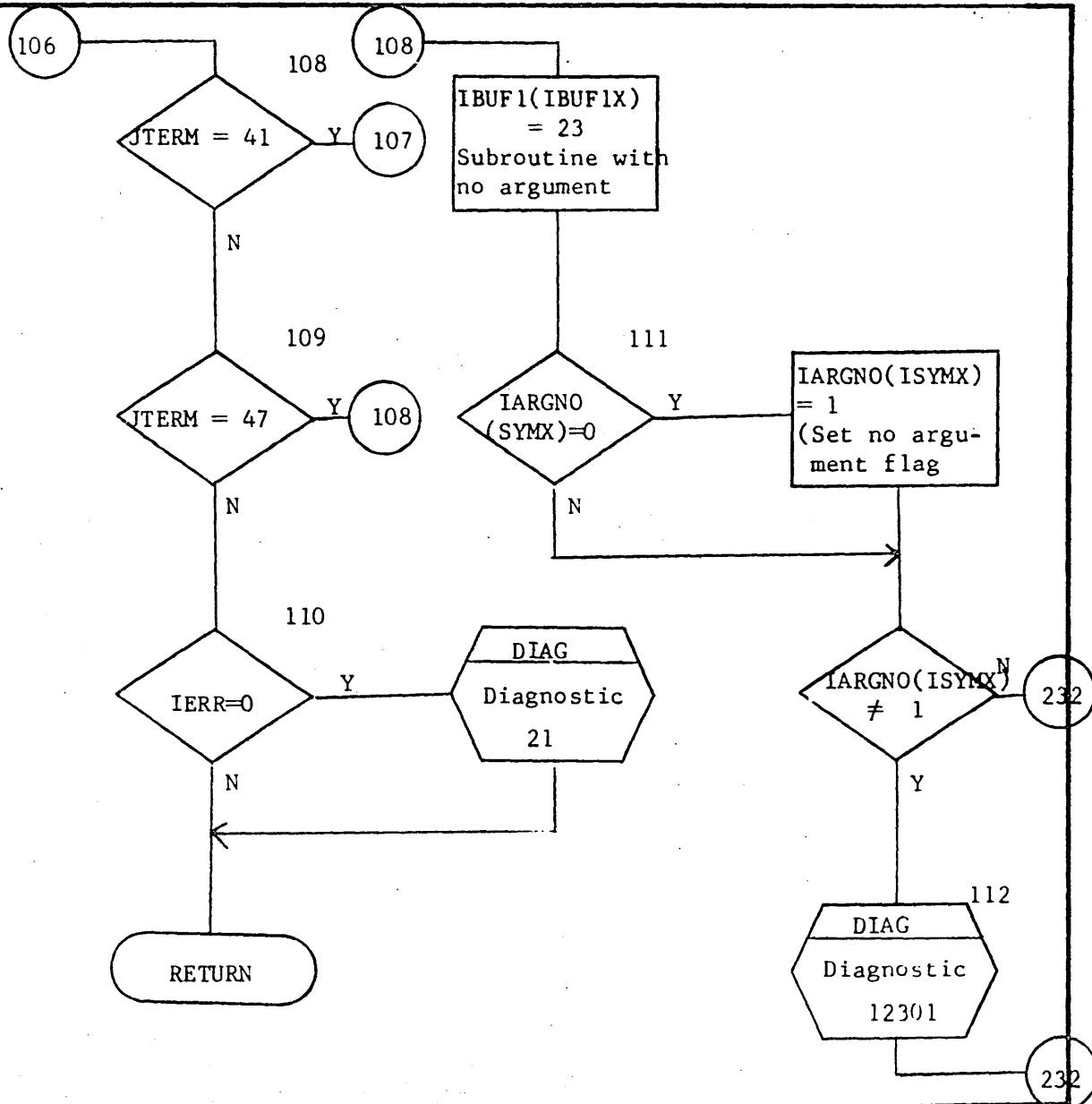
DOCUMENT CLASS	<i>IMS</i>	MACH. TYPE	<i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	<i>ARITH</i>	PAGE <i>10</i> OF <i>35</i>		PROJECT MGR.			
NUMBER	ISSUE DATE	DATE <i>1/1</i>		TASK NO			
DRAWN BY	DATE		TASK NAME				

101. CALL statement processor.
102. Check for symbol.
103. If SYMBOL not in symbol table, put it in.
104. Is CALL to same name as SUBROUTINE being compiled?
105. Check class unassigned.
106. Check class external.
107. Illegal subprogram reference.



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	MACH. TYPE	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE		PROJECT MGR.			
		PAGE 35 OF 35	PROJECT NAME			
	NUMBER	ISSUE DATE	TASK NO.			
	DRAWN BY	DATE	TASK NAME			

- 108. Check for next field starting with left parenthesis.
- 109. End of statement?
- 110. No previous error detected means a variable has been used as a subprogram name.
- 111. Check for argument flag set for no arguments.
- 112. Number of arguments differ in references to same subprogram.



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

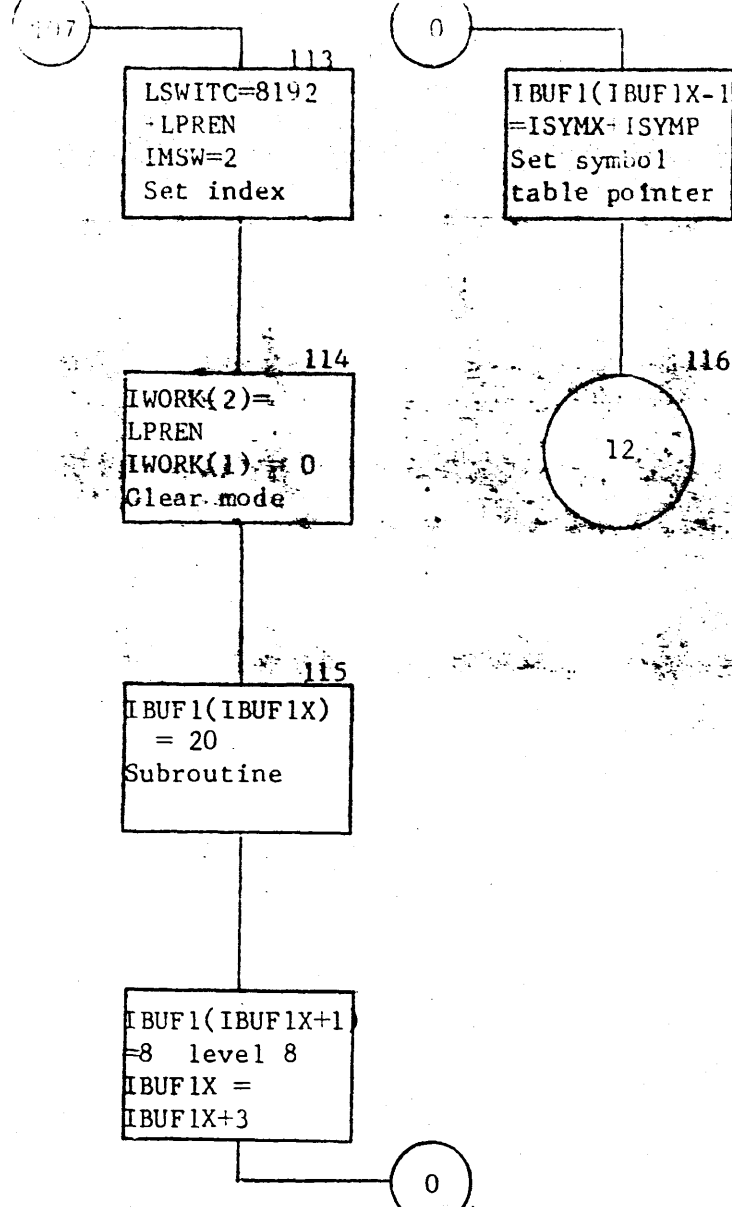
DOCUMENT CLASS <i>IMS</i>	MACH. TYPE <i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE <i>AP 174</i>		PROJECT MGR.			
	PAGE <i>2</i> OF <i>35</i>	PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.			
DRAWN BY	DATE	TASK NAME			

113. Make IWORK entry for this calling sequence parameter list. Set switch on and equal to 8192 + no. of left parentheses.

114. Set parenthesis level.

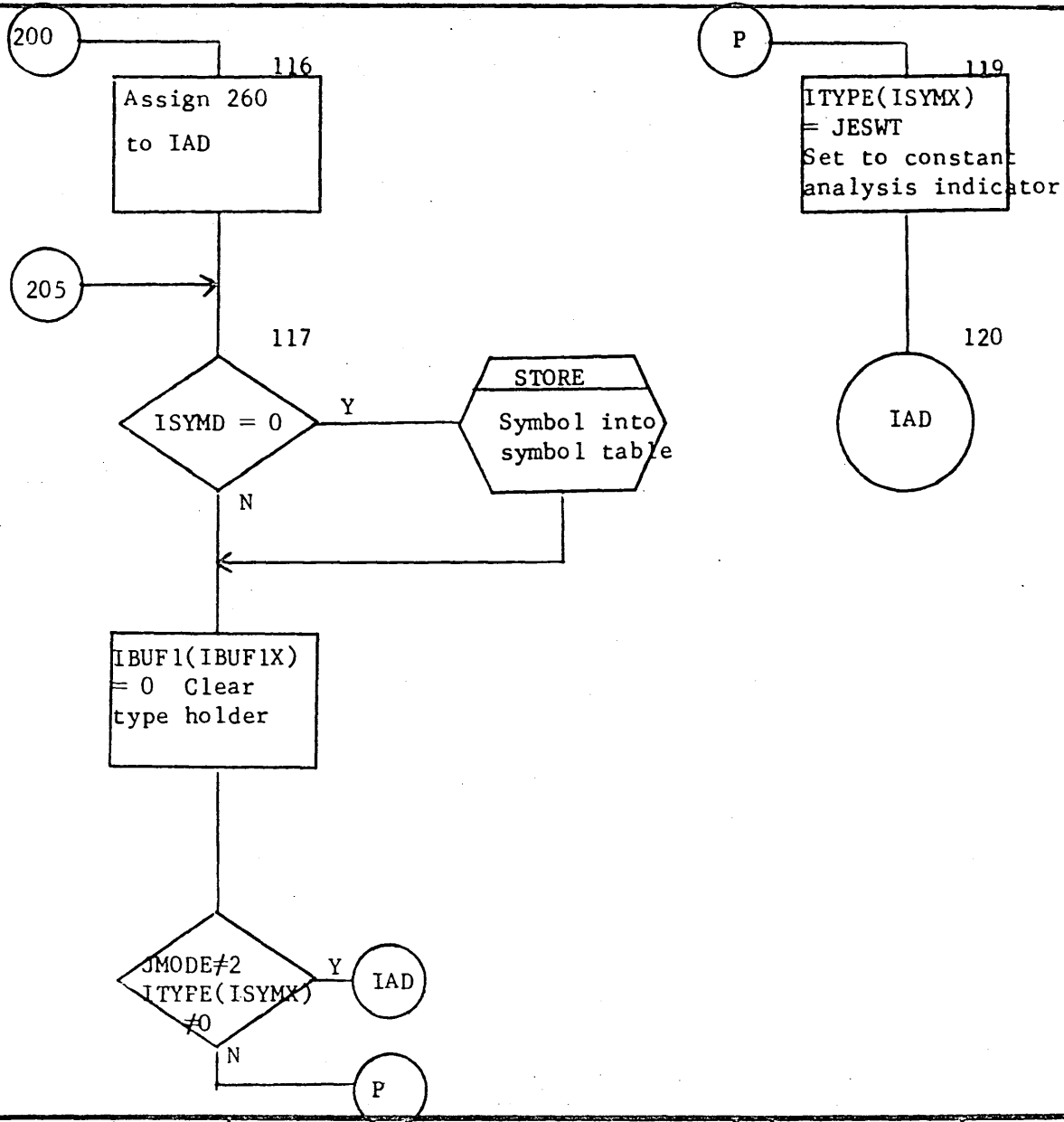
115. Subroutine opened

116. Go process "(".



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS <i>IMS</i>	MACH. TYPE <i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE <i>ARITH</i>		PROJECT MGR.			
		PAGE <i>22</i> OF <i>35</i>	PROJECT NAME			
	NUMBER	ISSUE DATE	TASK NO.			
	DRAWN BY	DATE <i>5-6-7</i>	TASK NAME			

- 116. Process alphanumeric or numeric portion of first field.
- 117. Put symbol in symbol table if not already there.
- 118. Check not symbol or if unassigned.
- 119. Must be constant so set to constant analysis indicator (JESWT).
- 120. Set to 260 if 1st field, otherwise 149 is set in normal loop.



**CONTROL DATA CORPORATION**  
 SOFTWARE DOCUMENT

SAMPLE CODE

FLOWCHART

DECISION TABLE

OTHER

DOCUMENT CLASS	<i>IMS</i>	MACH. TYPE	<i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	<i>APLTH</i>			PROJECT MGR.			
		PAGE	<i>33</i> OF <i>35</i>	PROJECT NAME			
NUMBER		ISSUE DATE		TASK NO			
DRAWN BY		DATE	<i>1/17</i>	TASK NAME			

- 121. Check for symbol.
- 122. Computed go to based on symbol in symbol table.
- 123. Numeric constant.
- 124. Check for buffer overflow, then recycle.

I - ISYMD+1  
(Set up computed GO TO)

MODE =  
ITYPE(ISYMX)  
Set mode

121  
JMODE = 2

122  
ISYMD ≠ 0

LS1 =  
IBUF1(IBUFX)  
Set last symbol

Assign 230  
to IAD

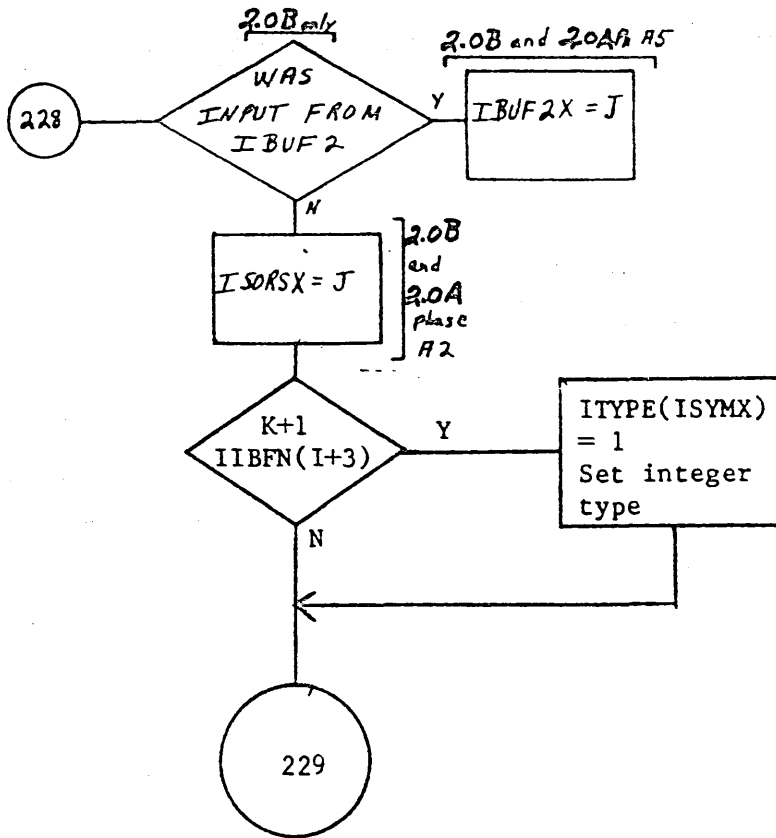
Assign 18  
to IAD

122  
ISYMD ≠ 0

144

<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	MACH TYPE	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE		PROJECT MGR.			
		PAGE OF	PROJECT NAME			
	NUMBER	ISSUE DATE	TASK NO.			
	DRAWN BY	DATE	TASK NAME			

132. Name is in table of intrinsic functions.
133. Check if number of parameters the same.

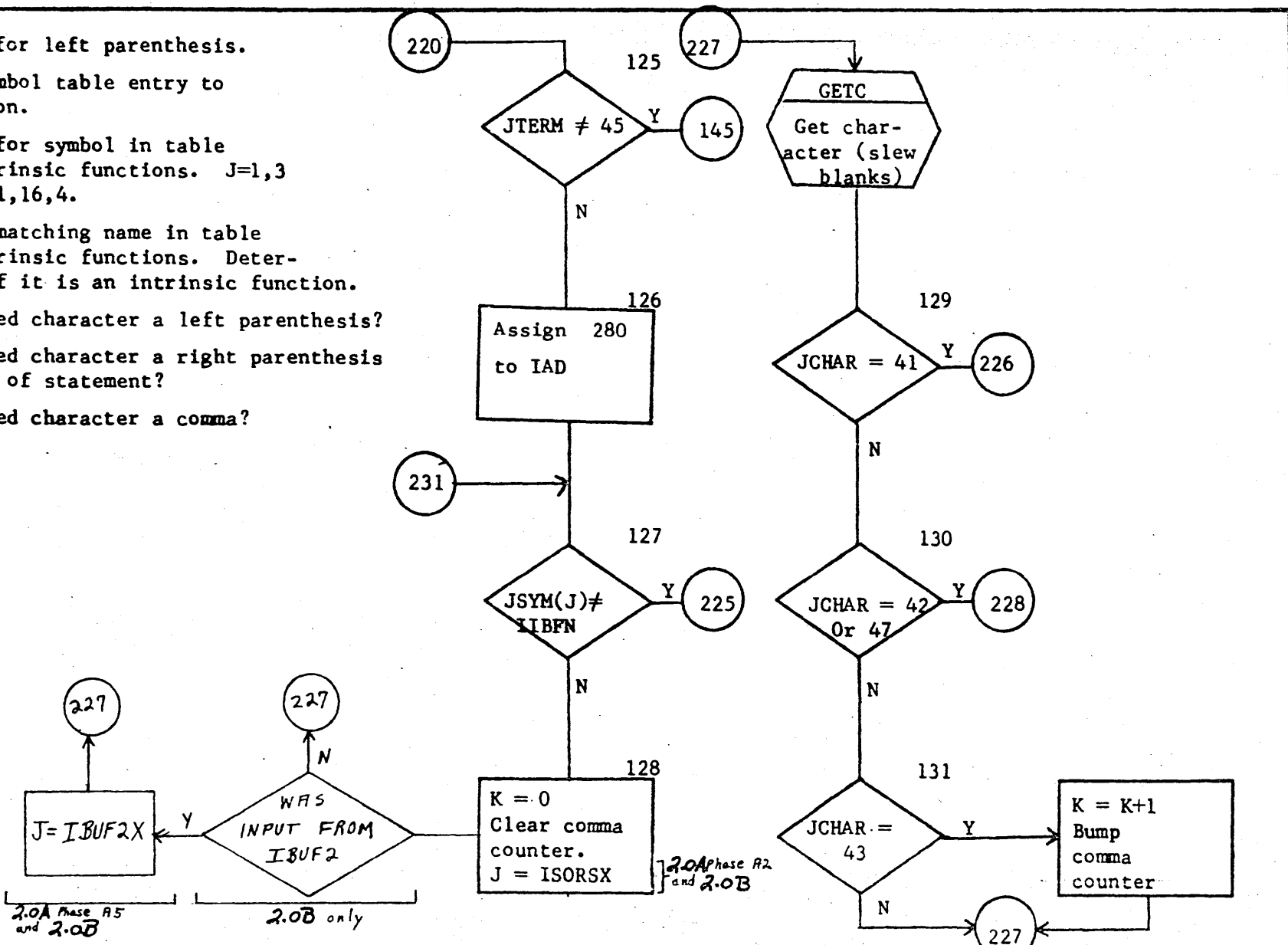


CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE  
FLOWCHART  
DECISION TABLE  
OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO	REV	APPROVED	DATE
DOCUMENT TITLE	ARITH			PROJECT MGR			
			PAGE 26 OF 35	PROJECT NAME			
NUMBER		ISSUE DATE		TASK NO			
DRAWN BY		DATE		TASK NAME			

- 125. Check for left parenthesis.
- 126. Set symbol table entry to function.
- 127. Check for symbol in table of intrinsic functions. J=1,3 and I=1,16,4.
- 128. Found matching name in table of intrinsic functions. Determine if it is an intrinsic function.
- 129. Imbedded character a left parenthesis?
- 130. Imbedded character a right parenthesis on end of statement?
- 131. Imbedded character a comma?



CONTROL DATA CORPORATION

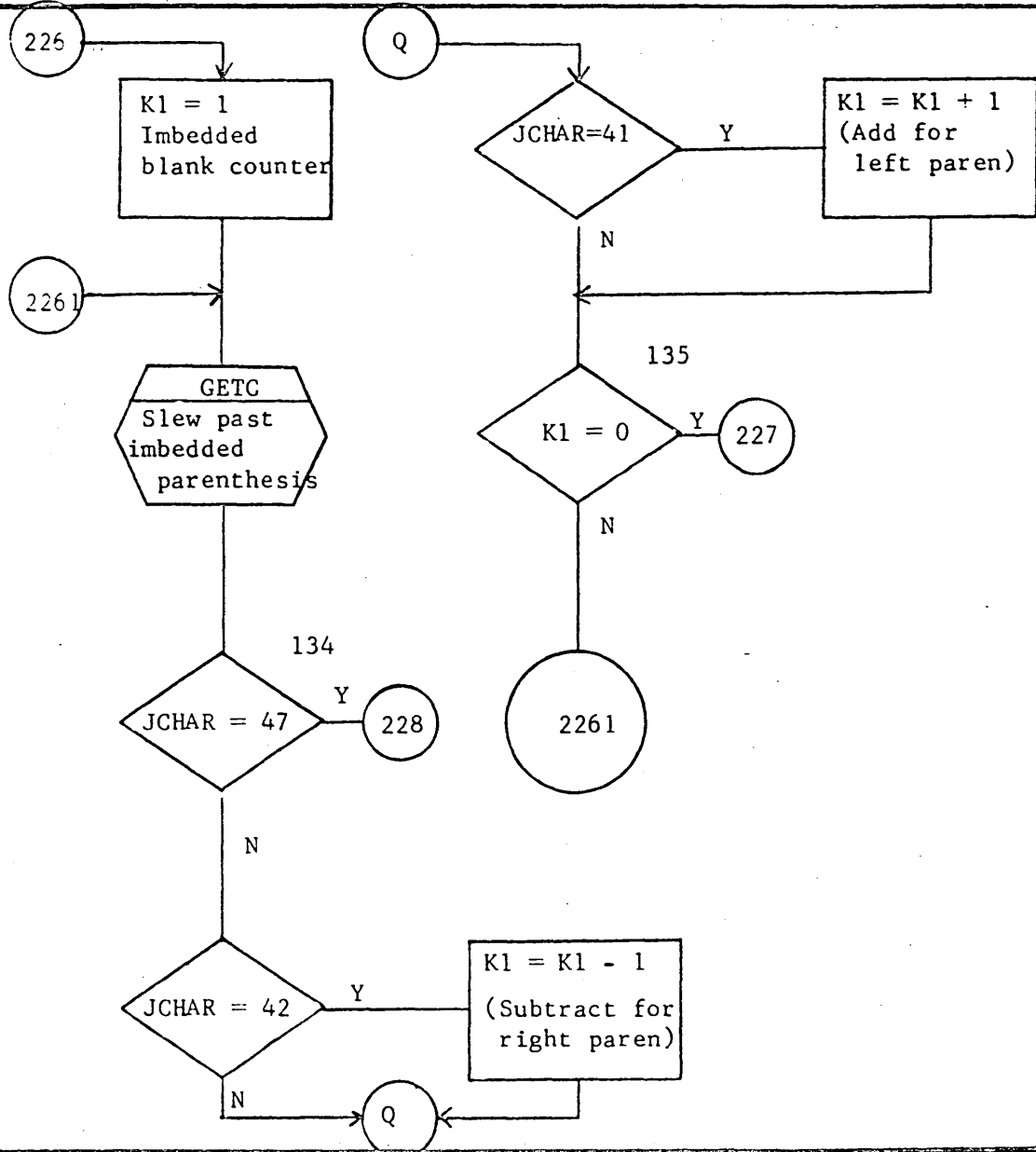
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.		REV		APPROVED		DATE	
DOCUMENT TITLE	ARITH			PROJECT MGR.							
		PAGE	25 OF 35	PROJECT NAME							
NUMBER		ISSUE DATE		TASK NO.							
DRAWN BY		DATE	3-67	TASK NAME							



134. Check for end of statement.  
 135. Check if past imbedded parenthesis.

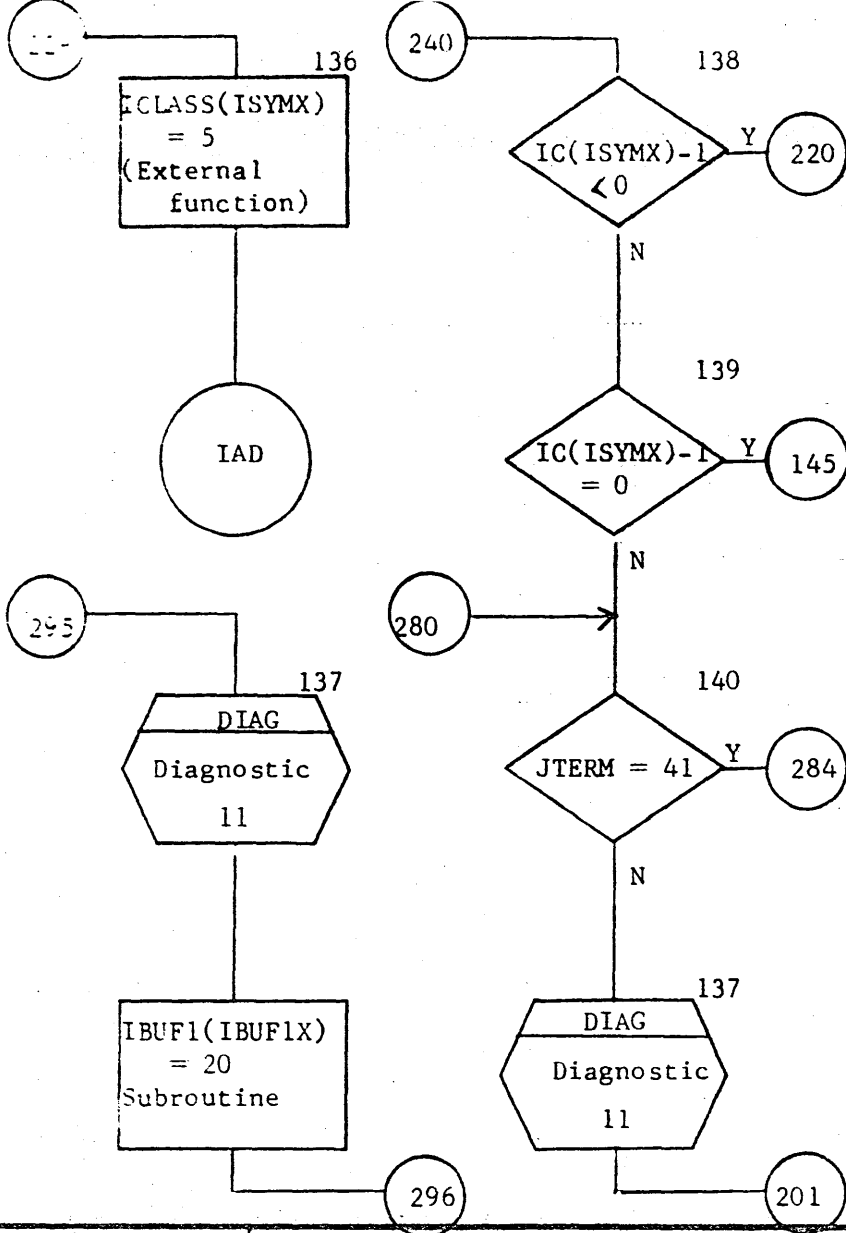


CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE  
FLOWCHART   
DECISION TABLE   
OTHER

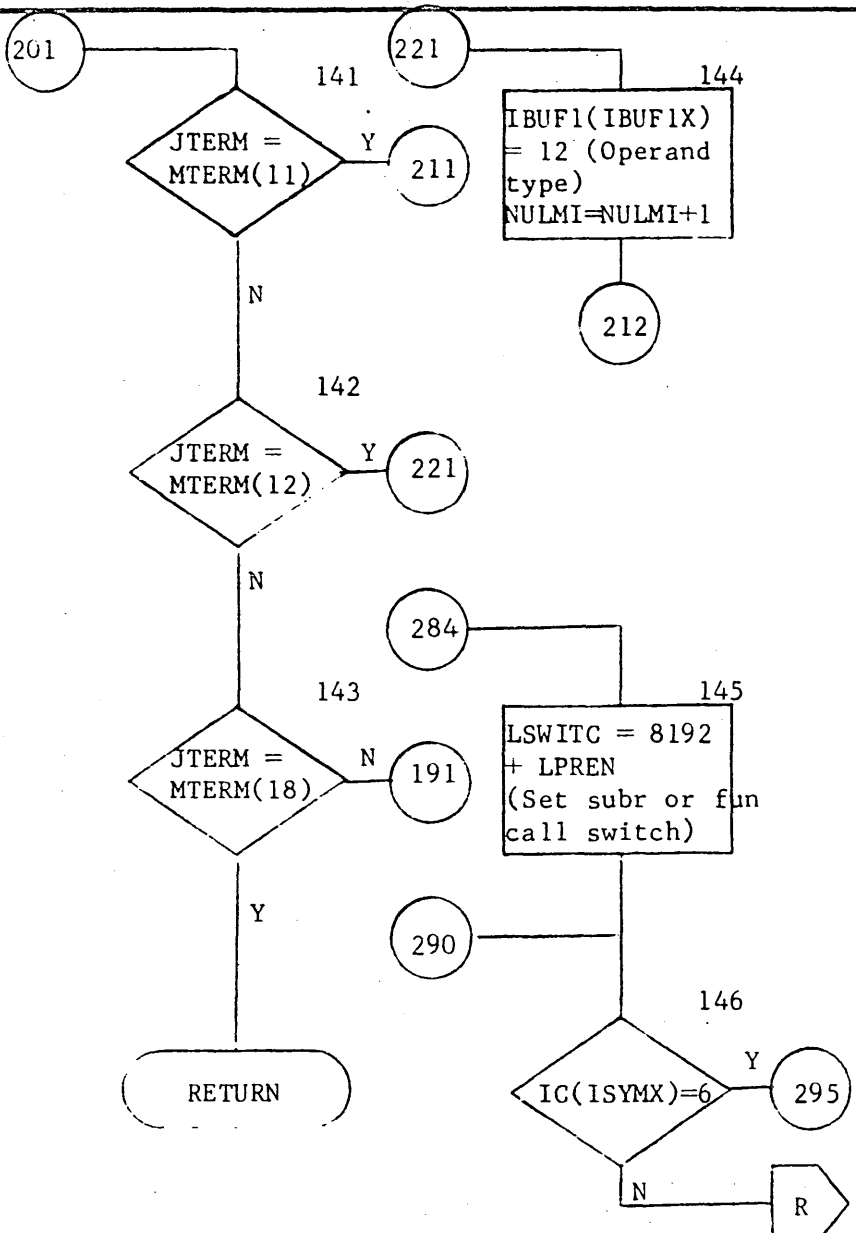
DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.		REV	APPROVED	DATE
DOCUMENT TITLE	ARITH			PROJECT MGR.				
			PAGE 27 OF 35	PROJECT NAME				
NUMBER		ISSUE DATE		TASK NO.				
DRAWN BY		DATE		TASK NAME				

- 136. Set class.
- 137. Illegal subprogram reference.
- 138. Check if class unassigned.
- 139. Check if class variable.
- 140. Other check for left parenthesis.



CONTROL DATA CORPORATION SOFTWARE DOCUMENT  SAMPLE CODE FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	<i>IMS</i>	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	<i>ARITH</i>	PROJECT MGR.			
		<i>PAGE 28 OF 35</i>	PROJECT NAME			
	NUMBER	ISSUE DATE	TASK NO.			
	DRAWN BY	DATE	TASK NAME			

- 141. Check if terminator of first field +.
- 142. Check if terminator of first field -.
- 143. Check if terminator of first field end of statement.
- 144. Terminator a -.
- 145. Left parenthesis encountered.
- 146. Check for external.

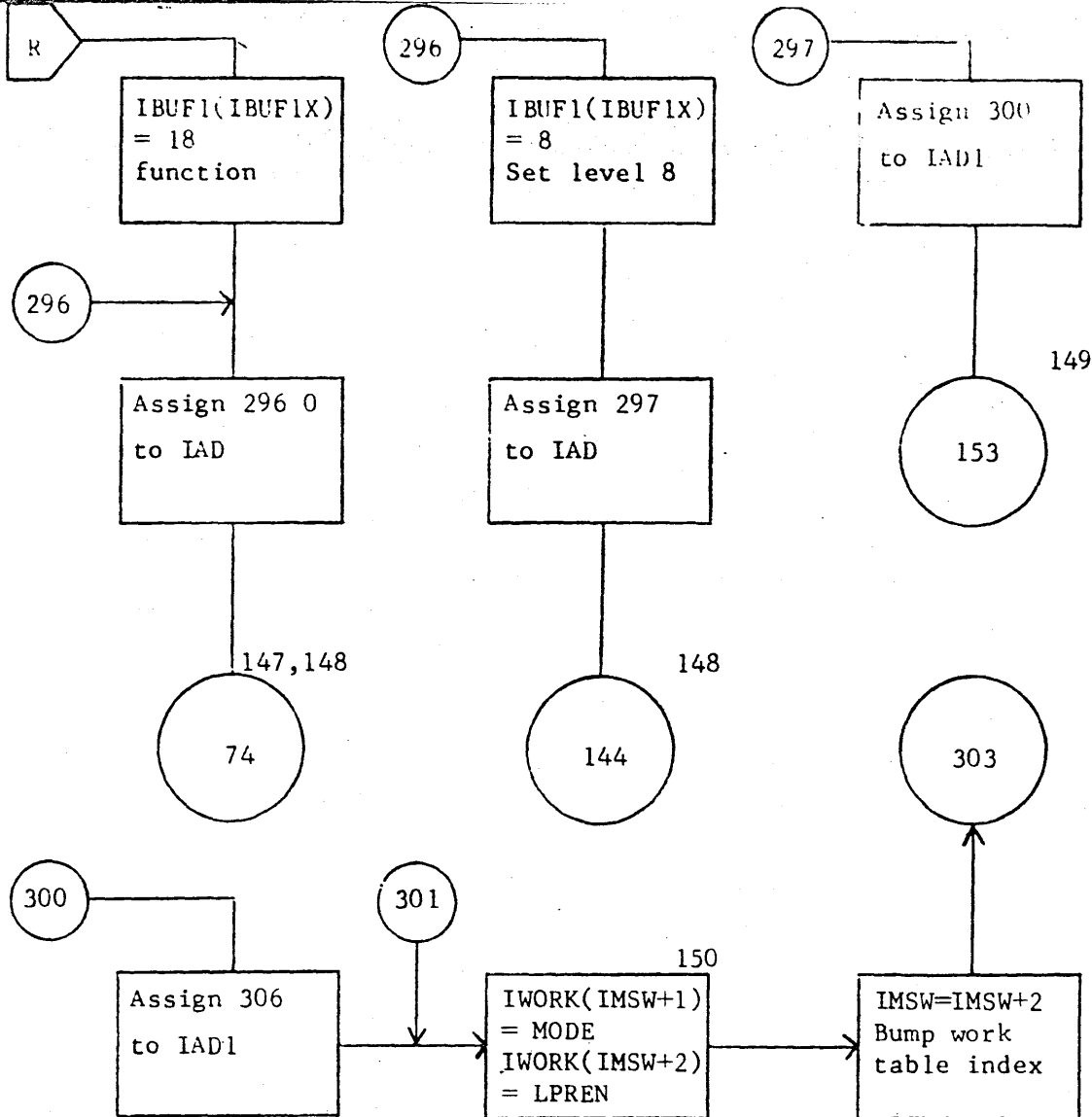


CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE  
FLOWCHART  
DECISION TABLE  
OTHER

DOCUMENT CLASS	<i>IMS</i>	MACH. TYPE	<i>1700</i>	PROJECT NO.		REV	APPROVED	DATE
DOCUMENT TITLE	<i>ARITH</i>			PROJECT MGR				
		PAGE	<i>28 of 35</i>	PROJECT NAME				
NUMBER		ISSUE DATE		TASK NO.				
DRAWN BY		DATE		TASK NAME				

- 147. Save type in ITEM.
- 148. Bump index, check for overflow and set SYMTAB entry number.
- 149. Check mode.
- 150. Set mode and level of parentheses in intrinsic function work table.



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

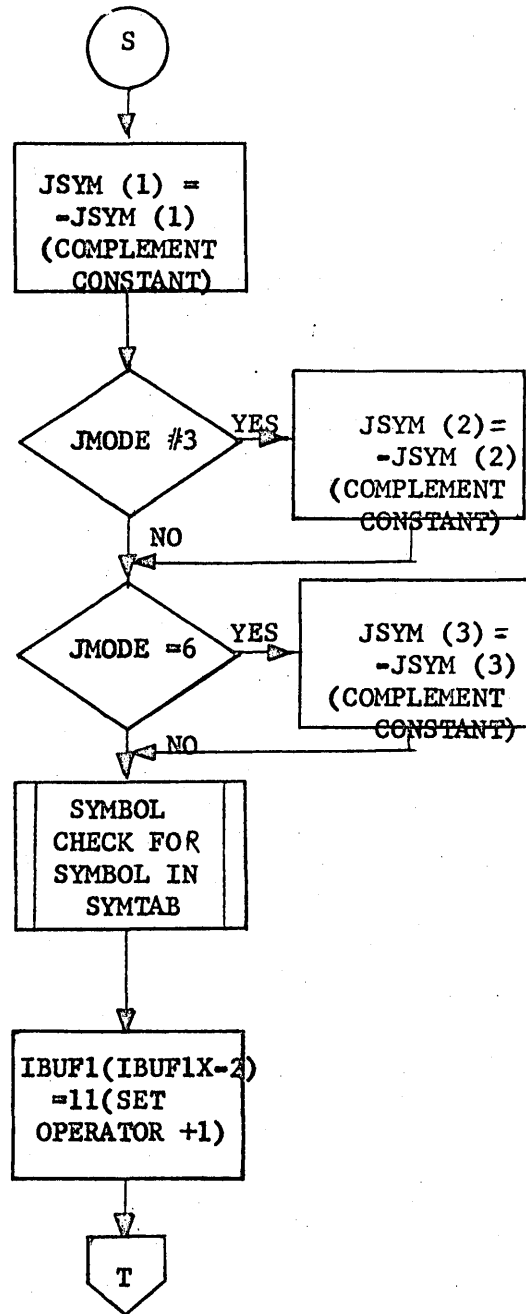
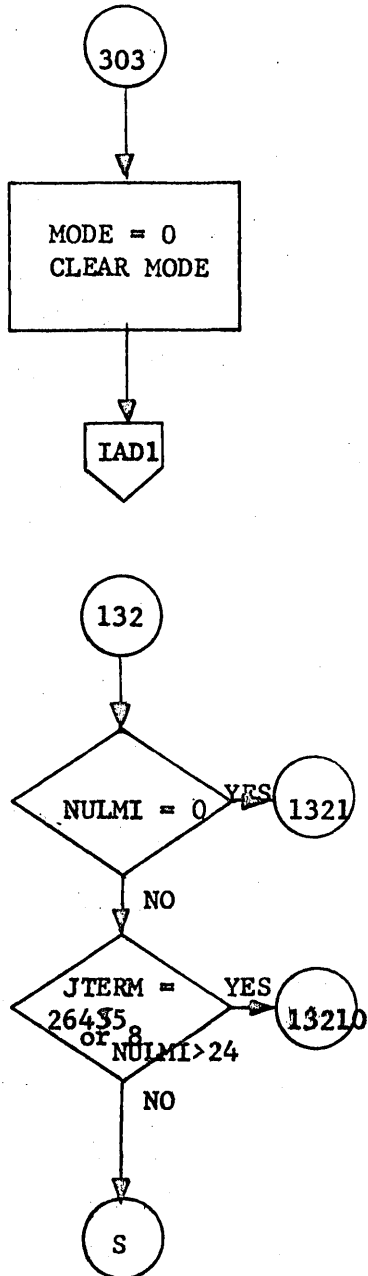
SAMPLE CODE  
FLOWCHART  
DECISION TABLE  
OTHER



DOCUMENT CLASS	<i>IMS</i>	MACH. TYPE	<i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	<i>ARITH</i>			PROJECT MGR.			
		PAGE	<i>30 OF 35</i>	PROJECT NAME			
NUMBER		ISSUE DATE		TASK NO.			
DRAWN BY		DATE	<i>5/67</i>	TASK NAME			

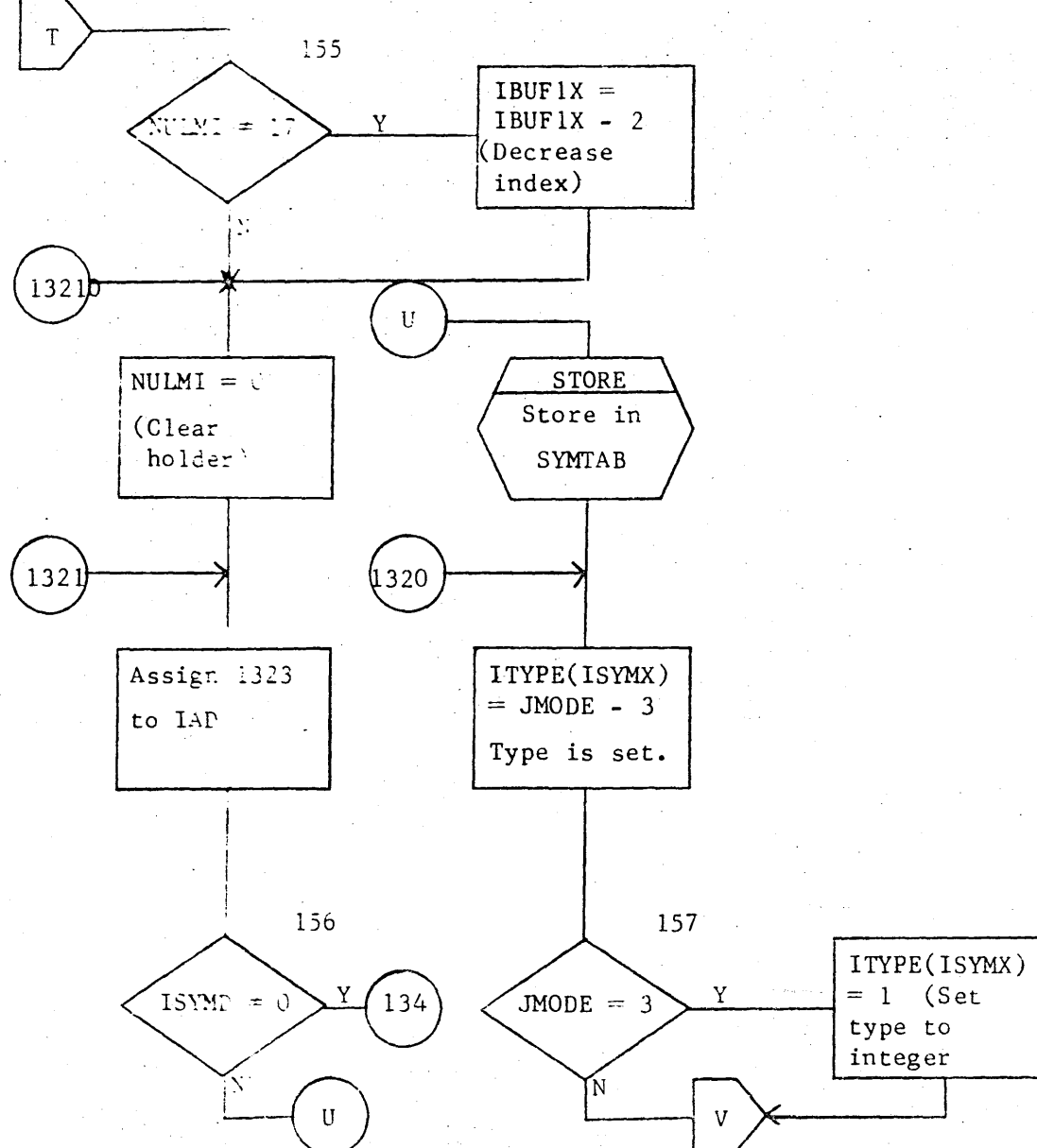


LA JOLLA FACILITY



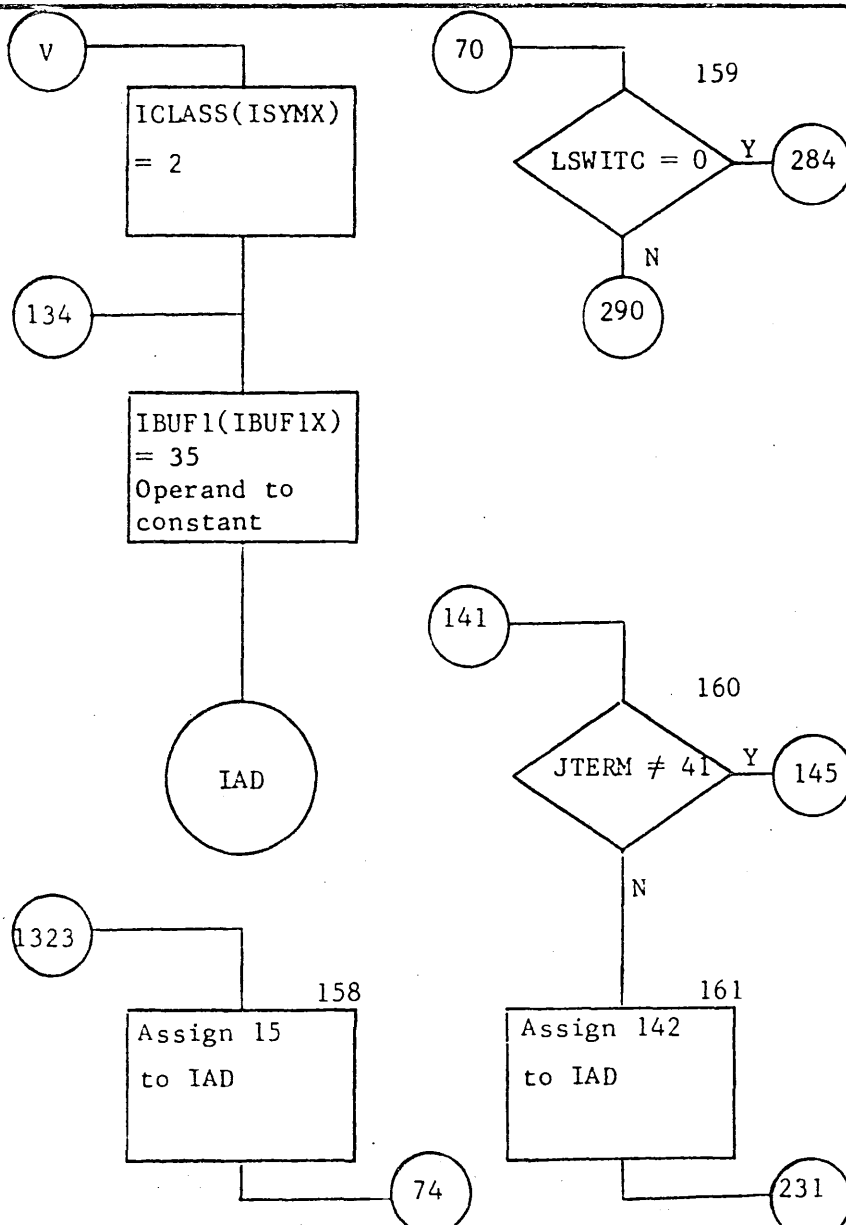
TITLE		ARITH		DRG. NO.	
DRAWN BY		PROJ.		REVISION	
CSD 203		DATE		SHEET 3 OF 3	

- 155. Last symbol not ")"?"
- 156. Symbol not in symbol table?
- 157. Is type integer?



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	<i>IMS</i>	MACH. TYPE	<i>1700</i>	PROJECT NO.	REV.	APPROVED	DATE
	DOCUMENT TITLE	<i>ARITH</i>			PROJECT MGR.			
		PAGE <i>32</i> OF <i>35</i>			PROJECT NAME			
	NUMBER	ISSUE DATE			TASK NO.			
	DRAWN BY	DATE <i>1/7</i>		TASK NAME				

- 158. Go check predecessor.
- 159. Class is function. Check for function indicator set.
- 160. Not left parenthesis? Yes, go to variable processor.
- 161. Process arguments as in subroutine call.

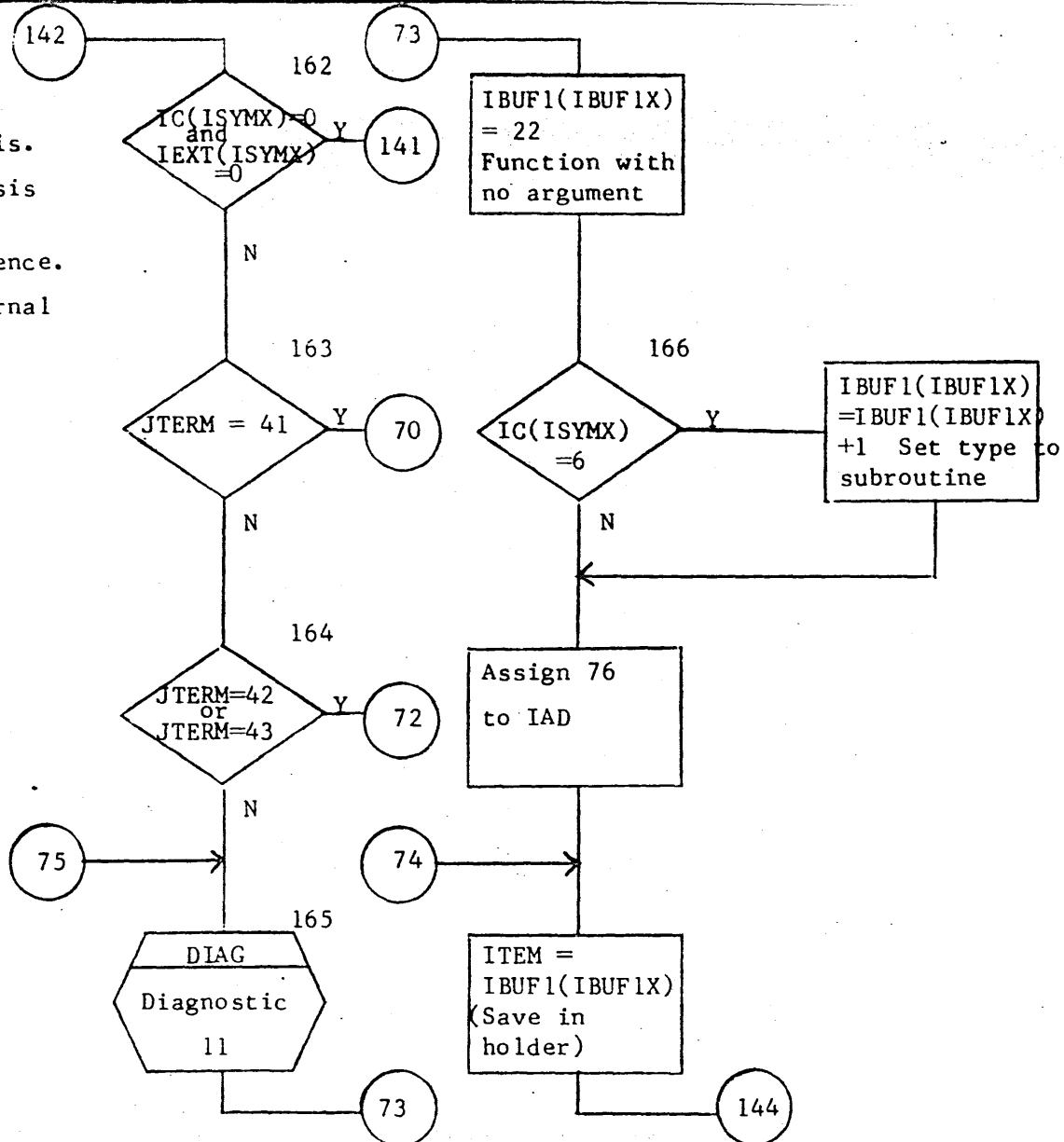


CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE  
FLOWCHART  
DECISION TABLE  
OTHER

DOCUMENT CLASS <i>JMS</i>	MACH. TYPE <i>M700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE <i>ARITH</i>		PROJECT MGR			
	PAGE <i>33</i> OF <i>35</i>	PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO			

- 162. Determine if class is set and external is set.
- 163. Check for left parenthesis.
- 164. Check for right parenthesis or comma.
- 165. Illegal subprogram reference.
- 166. Check if class says external subroutine.

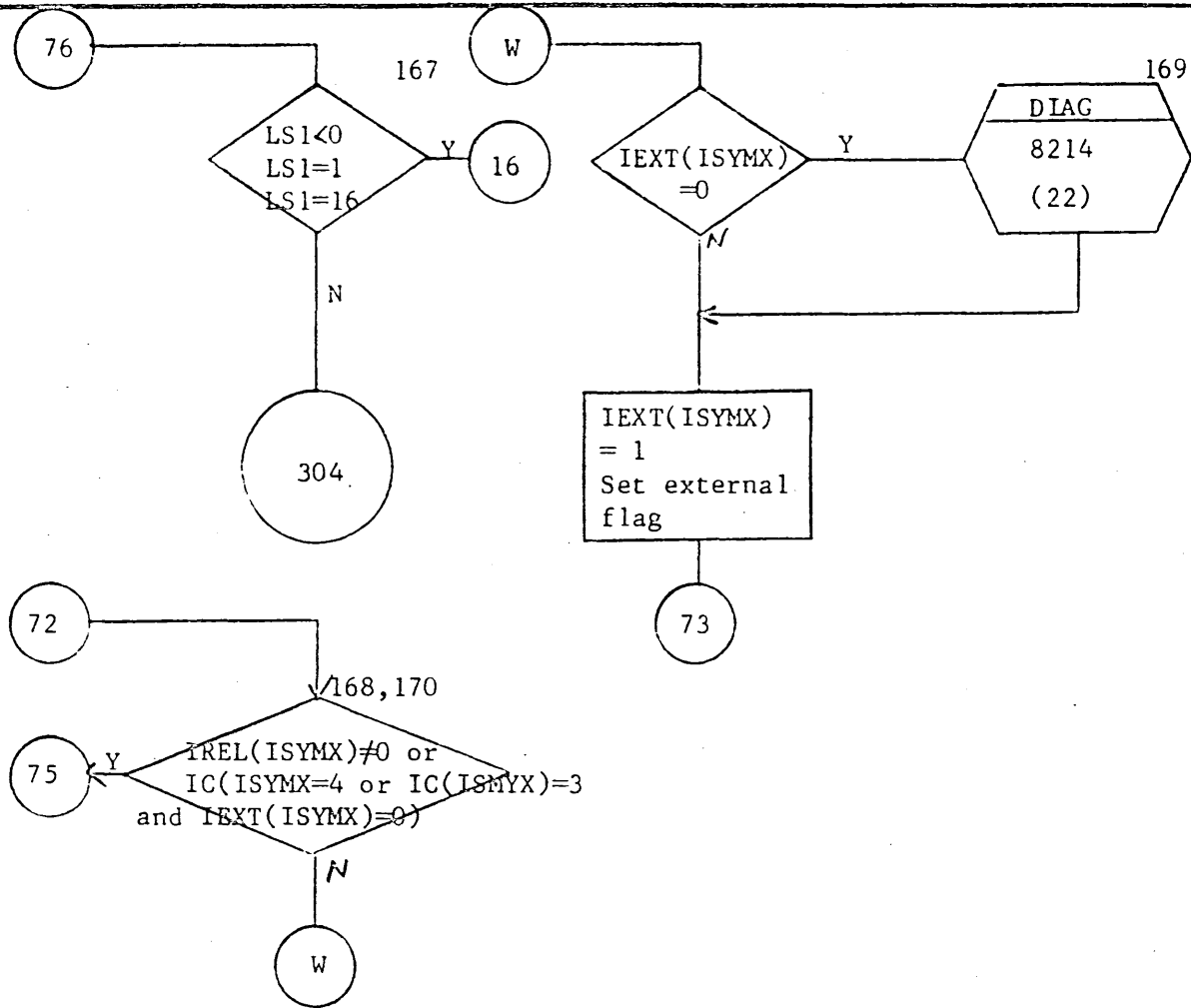


CONTROL DATA CORPORATION SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	<i>IMS</i>	MACH TYPE	<i>1700</i>	PROJECT NO.	REV	APPROVED	DATE	
	DOCUMENT TITLE	<i>ARITH</i>			PROJECT MGR.				
			PAGE	<i>34</i>	OF	<i>35</i>			
	NUMBER		ISSUE DATE		TASK NO.				
	DRAWN BY		DATE	<i>6/4</i>	TASK NAME				

34-E



- 167. Check for predecessor a comma or right parenthesis.
- 168. Check for RELATIVE, statement function or Intrinsic function and external. Relative external is illegal as an argument.
- 169. Subprogram name does not appear in external statement.
- 170. Run-anywhere option produces relative externals.



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE  
FLOWCHART  
DECISION TABLE  
OTHER

DOCUMENT CLASS	CLASS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	ARITH			PROJECT MGR			
		PAGE	35 OF 35	PROJECT NAME			
NUMBER		ISSUE DATE		TASK NO.			
		DATE		TASK NAME			

A

B

C

D

SUB  
SCR

```

IV=ISYMX
ISFLAG,IX1{1}
IX1{2},IX1{B}
IDL{1},IDL{P}
IDL{3},ISS{1}
ISS{2},ISS{3}=0

```

```

IVM=TYPE
IVM=TYPE

```

Two  
word  
integer

Yes

IVM=2

No

```

ISS{1} →
ISS{3} =
DIMENSIONS

```

```

ICL{1} →
ICL{3} =
IVM
IV=IBUFLX-2

```

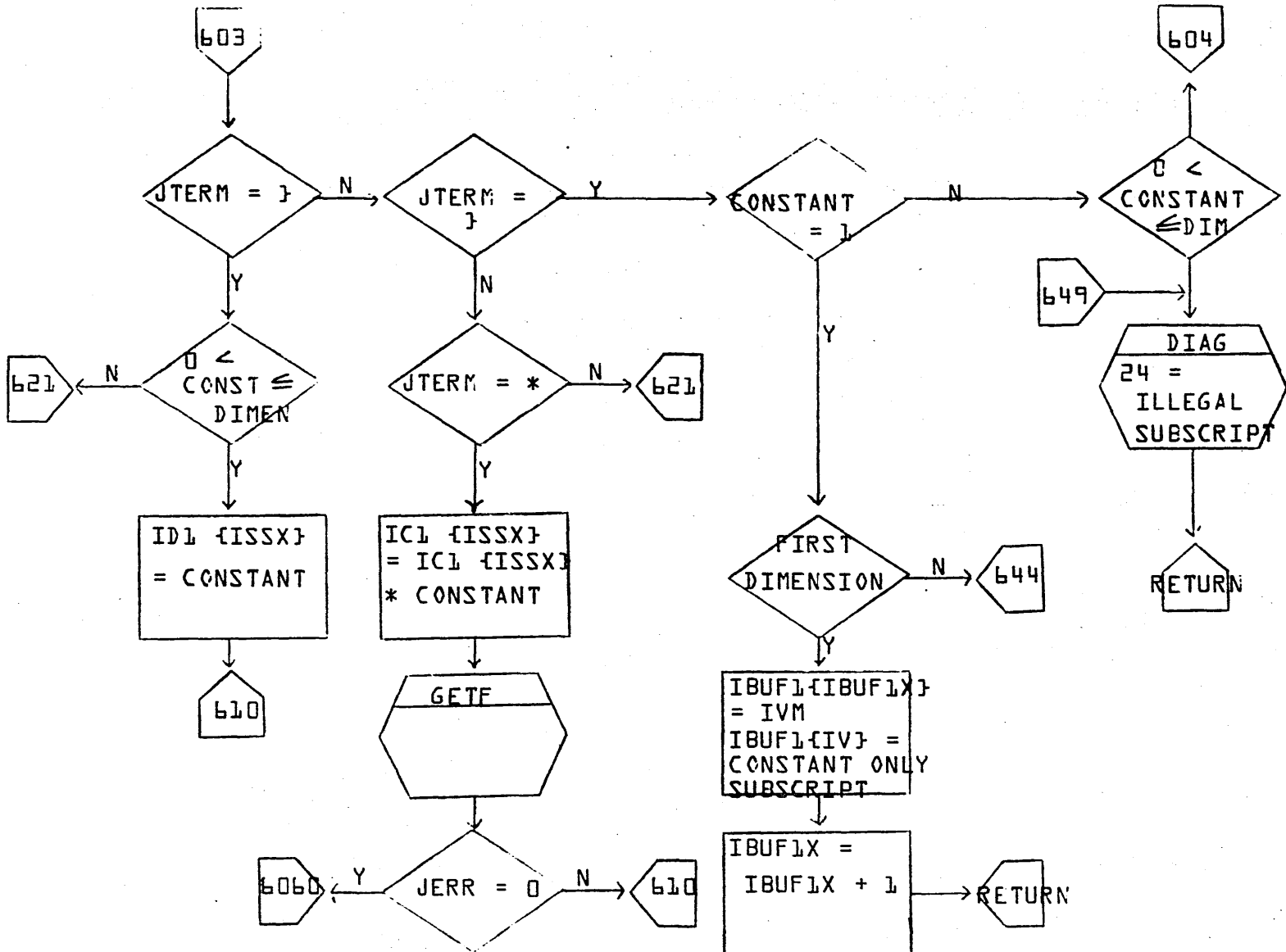
606

**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

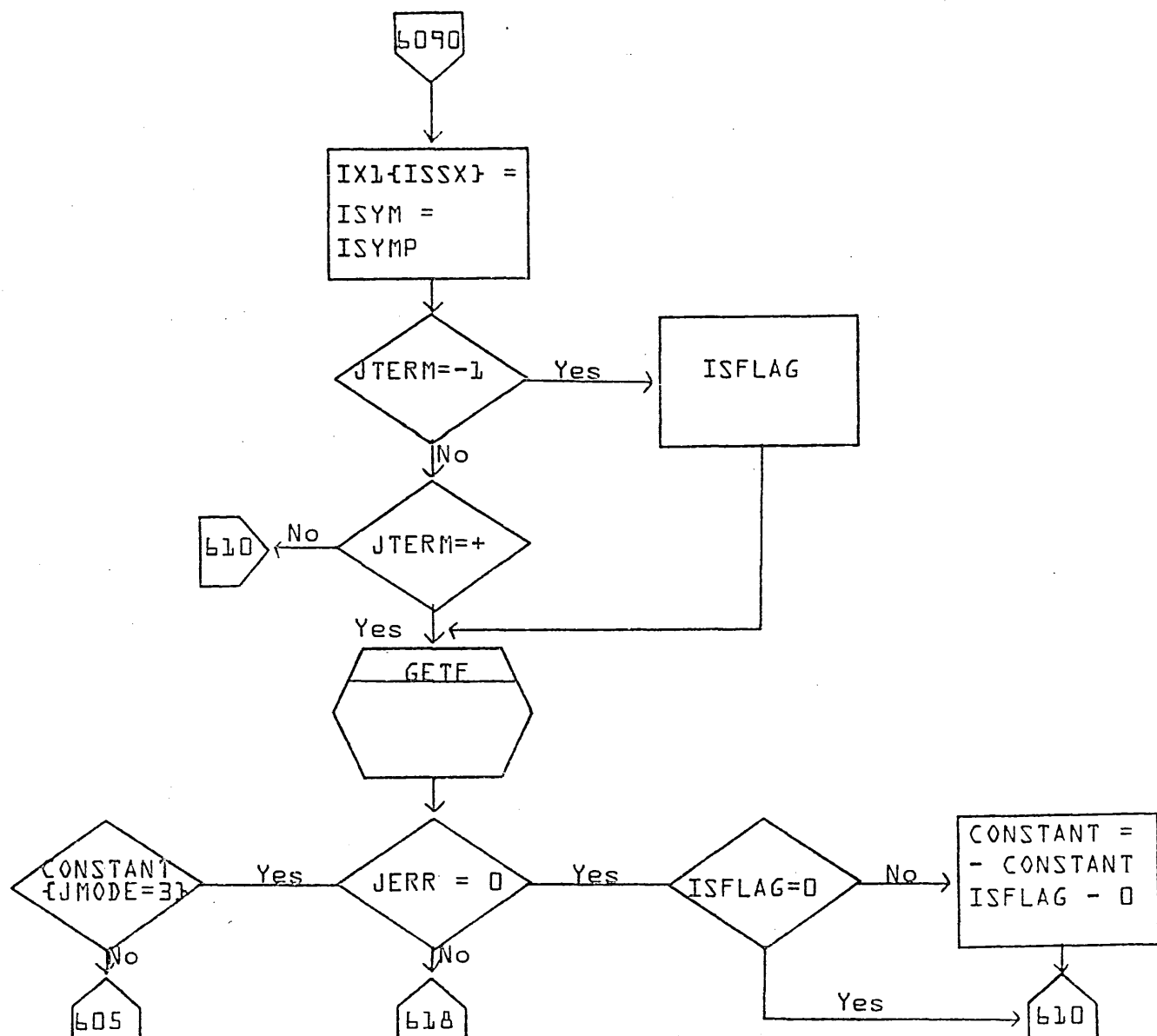
DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	SUBSCR			PROJECT MGR.			
		PAGE	1 OF 6	PROJECT NAME			
NUMBER		ISSUE DATE		TASK NO.			
DRAWN BY		DATE		TASK NAME			





<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	SUBSCR	PAGE 3 OF 6		PROJECT MGR.			
	NUMBER	ISSUE DATE			PROJECT NAME			
	DRAWN BY	DATE			TASK NO.			
					TASK NAME			

A  
B  
C  
D



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE

FLOWCHART

DECISION TABLE

OTHER

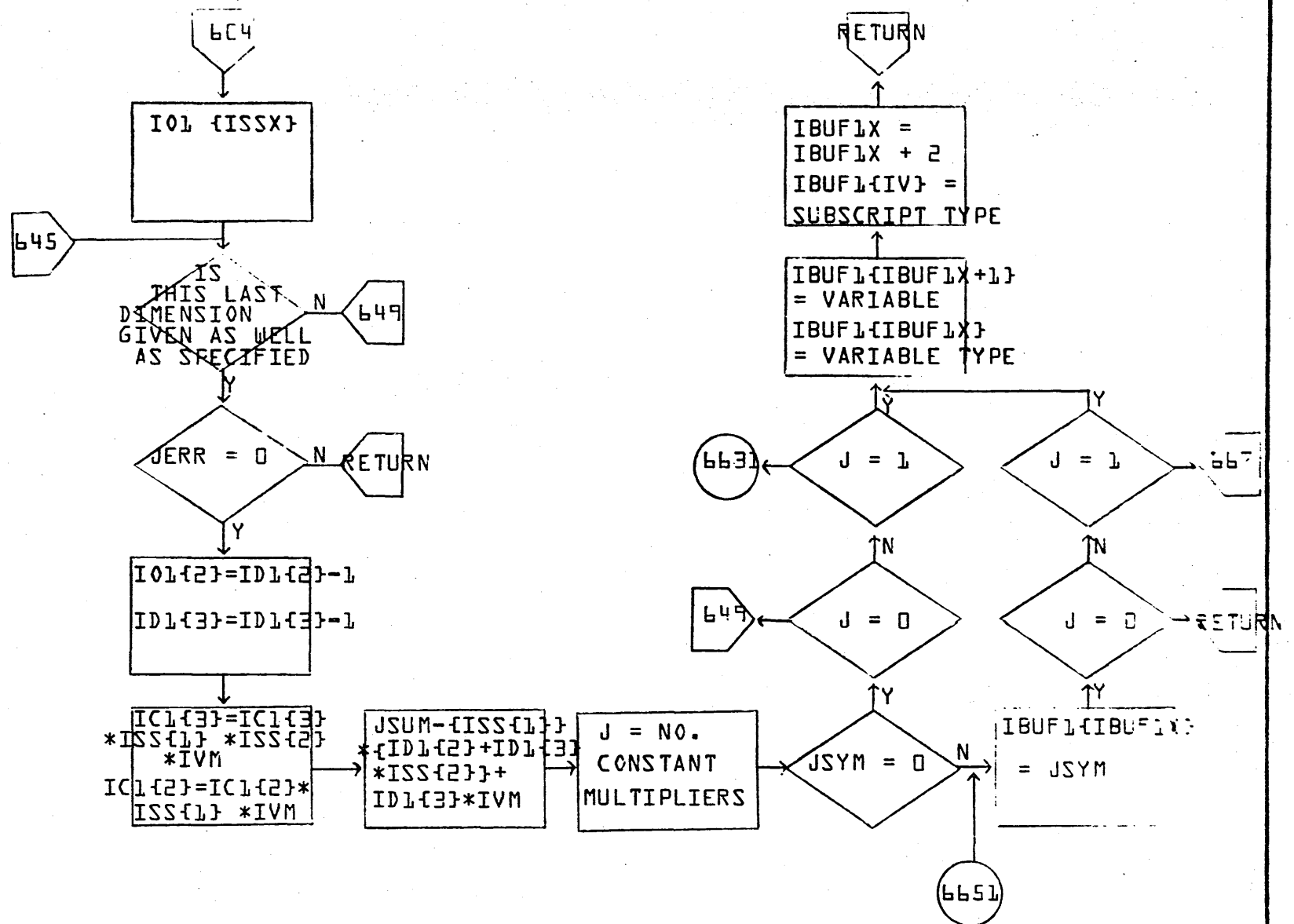
DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	SUBSCR			PROJECT MGR.			
PAGE 4 OF 6				PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.					
DRAWN BY	DATE	TASK NAME					

A

B

C

D



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE   
FLOWCHART   
DECISION TABLE   
OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	SUBSCR	PAGE 5 OF 6		PROJECT MGR.			
NUMBER	ISSUE DATE			PROJECT NAME			
DRAWN BY		DATE		TASK NO.			
				TASK NAME			

667

```

IBUFL {IBUFLX}
IBUFLX
IBUFL {IBUFLX+1}
= 1
IBUFL {IBUFLX+1}
LEVEL

```

```

IBUFLX =
IBUFLX + 1

```

```

IBUFL {IBUFLX}
+FF = C1+1+
C2+2*C3+3

```

```

IBUFL {IV=3}
JMODE IN
COMPLEX PART
OF SUBSCRIPT

```

```

IBUFL {IV}
= SUBSCRIPT
TYPE

```

RETURN

CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	SUBSCR			PROJECT MGR.			
PAGE 6 OF 6				PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.					
DRAWN BY	DATE	TASK NAME					

DOCUMENT CLASS IMS PAGE NO. 3-54  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. COO5\*3.1 A/B MACHINE SERIES 1700

3.4.3 Subroutine TREE

This routine picks up from IBUF1 an arithmetic expression that has been initially processed by ARITH and converts it into a tree form in IBUF2 for output. In order to make a tree, multiple passes are made over the statement. Each pass finds the primary operator and its related operands. If the operand is a subexpression, it, in turn, is broken up in a similar fashion and represents a lower level or branch of the tree. Each level of the tree is ordered by operator followed by its operands.

For example:

x a \* a + b \* c

is scanned and broken into the subexpressions:

x a \* a

and b \* c

The operator + is output

x a \* a

is scanned. The operator \* and the variables x, a and a are output then

b \* c

is scanned. The operator \* and the variables b and c are output.

The form of the Arithmetic tree set up in IBUF2 is as follows:

example = C+D

WORD 1	-1	Tree indicator
WORD 2	10	n = words in expression {from word 4 to end}
WORD 3	2	Mode of expression {1=integer, 2=real, 3=double}
WORD 4	11	Operator {see list}
WORD 5	2	n = number of operands
WORD 6	4	Pointer to 1st operand
...		Pointer to next operand, etc.
WORD {N}		N = {5 + number of operands}
...		
WORD {Z}		Operands follow operand pointers {see list}



DOCUMENT CLASS IMS PAGE NO. 3-54A  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3-1 A/B MACHINE SERIES 1700

example {cont'd}

WORD 7	7	Pointer to second operand
WORD 8	0	Normal/inverse switch
WORD 9	24	Operand type {see list}
WORD 10	→	SYMTAB Pointer to D
WORD 11	0	Normal/inverse switch
WORD 12	24	Operand type
WORD 13	→	SYMTAB Pointer to C

Operand pointer = first word number of operand - word  
number of operator

that is,  $4 = 8 - 4$   
that is,  $7 = 11 - 4$

Note: An operator on a lower level is treated as an operand by a higher level operator, therefore, an operator may be both an operator and operand depending on which level is being processed.



DOCUMENT CLASS \_\_\_\_\_ IMS \_\_\_\_\_ PAGE NO. 3-55  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT NO. C005 \_\_\_\_\_ VERSION 2.0 \_\_\_\_\_ MACHINE SERIES 1700 \_\_\_\_\_

TREE is called by the subroutine ARITH. In the 2.0A version, if ARITH detected mixed mode arithmetic in the expression, TREE calls MODMXR just before returning to ARITH.

Subroutines used: DIAG, GETSYM, PUNT, MODMXR.

Switches and Tables during tree building

IPERAT	Current operator
IBUF2	Table where output entry is built
IBUF2X	IBUF2 indexer
IBUF1	Table containing input expression
IBUF1X	IBUF1 indexer
LS1	.NOT. operator in effect switch
LPREN	IBUF1X value on which to stop current subexpression analysis
NOOP	Number of operators this level
MINUS	Leading variable preceded by an operator switch
NLP	First non-left parenthesis operator has been encountered switch - used in determining existence of leading sign
ISAV	IBUF2 index of operator this level
LSWITC	Last operator was a function switch
ISSX	Level of current operator
IVM	Start of expression in IBUF2
ISFLAG	First operator switch - used in counting leading left parens
IWORK	History of expression being built in IBUF2 Four words per entry Word 1 IBUF1 index on which to stop subexpr analysis (Dummy entry indicator for .NOT if 0) Word 2 IBUF1 index for operator, this subexpression or: Assumed operator indicator (-operator in upper six bits and IBUF1 index in lower 10) Word 3 IBUF2 index of operand pointer, this operand Word 4 IBUF2 index of operator this level
ISTWK	IWORK index where entries, this level, start
LO	Class of prior operator
IMSW	Integer divide operator switch 0 = no prior operator 1 = logical 2 = relational 3 = arithmetic
LRELQ	IWORK index
BRELEN	IBUF1 index at which to start current subexpression analysis

NOTE: Each operand (operator) except the first is preceded by a normal/inverse indicator.

DOCUMENT CLASS IMS PAGE NO. 3-5b  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT NO. 005 VERSION 2.0 MACHINE SERIES 1700

Operand Types within Trees (Operators are Also Operands)

1. COMMA - does not appear in File 2
2. AND
3. OR
4. NOT - does not appear in file 2
5. LT
6. GT
7. LE
8. GE
9. EQ
10. NE
11. +
12. - does not appear in File 2 - represented by inverse +
13. \* (  $\div$  inverse \*)
14. non-reorderable  $\div$  (integer)
15. \*\*
16. ( does not appear in File 2
17. ) does not appear in File 2
18. Function
19. unused
20. subroutine
21. unused
22. function with no argument
23. subroutine with no argument
24. non-subscripted variable
25. variable only sbuscripted variable
26. increment only subscripted variable
27. variable and increment subscripted variable
28. complex subscripted variable
29. non-subscripted partial variable
30. variable only subscripted partial variable
31. increment only subscripted partial variable
32. variable and increment subscripted variable
33. complex subscripted partial variable
35. numeric constant
36. calling sequence label - pass 4 generaged
37. material constant

Types 22 - 24, 29, 34, 35 are two word entries

WORD 1 operand type  
 WORD 2 symbol table pointer

Types 18 - 21 take two words tp express type

WORD 1 operand type  
 WORD 2 symbol table pointer  
 WORD 3 no. operands etc.

DOCUMENT CLASS TMS PAGE NO. 3-  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

Types 25 - 30 are of the following form

WORD 1 25,30  
WORD 2 symbol table pointer  
WORD 3 24,29 for subscript variable  
WORD 4 symbol table pointer

Types 26 and 31 are of the following form

WORD 1 26,31  
WORD 2 symbol table pointer  
WORD 3 constant subscript

Types 27 and 32 are of the following form

WORD 1 27,32  
WORD 2 symbol table pointer  
WORD 3 constant portion  
WORD 4 24,29 for subscript variable  
WORD 5 symbol table pointer for subscript variable

Types 28 and 33 are of the following form

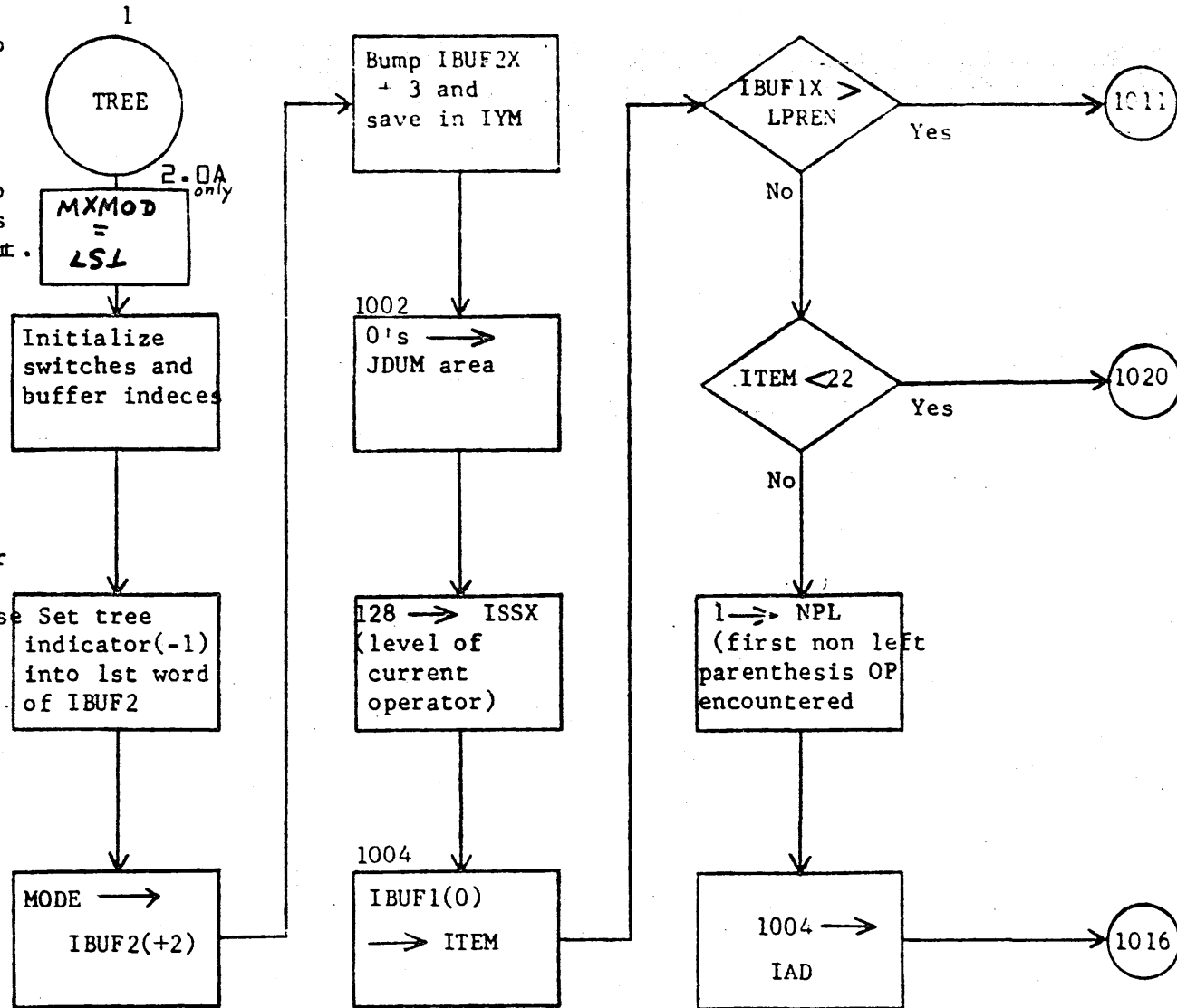
WORD 1 28,33  
WORD 2 symbol table pointer  
WORD 3 constant portion  
WORD 4 operator (+ or \*)  
WORD 5 no. operands pointers operands

1. This routine picks up an arithmetic expression that has been initially processed by ARITH from IBUF1, converts it to tree form, and places it in IBUF2 for output.

Tree Form

Word

- 1 Tree indicator(-1)
  - 2 # of words in expression
  - 3 Mode of expression
  - 4 Operator
  - 5 # operands
  - 6 pointer base operator this level
  - 7 Operand normal/inverse
  - 8 Operand
- Etc.



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

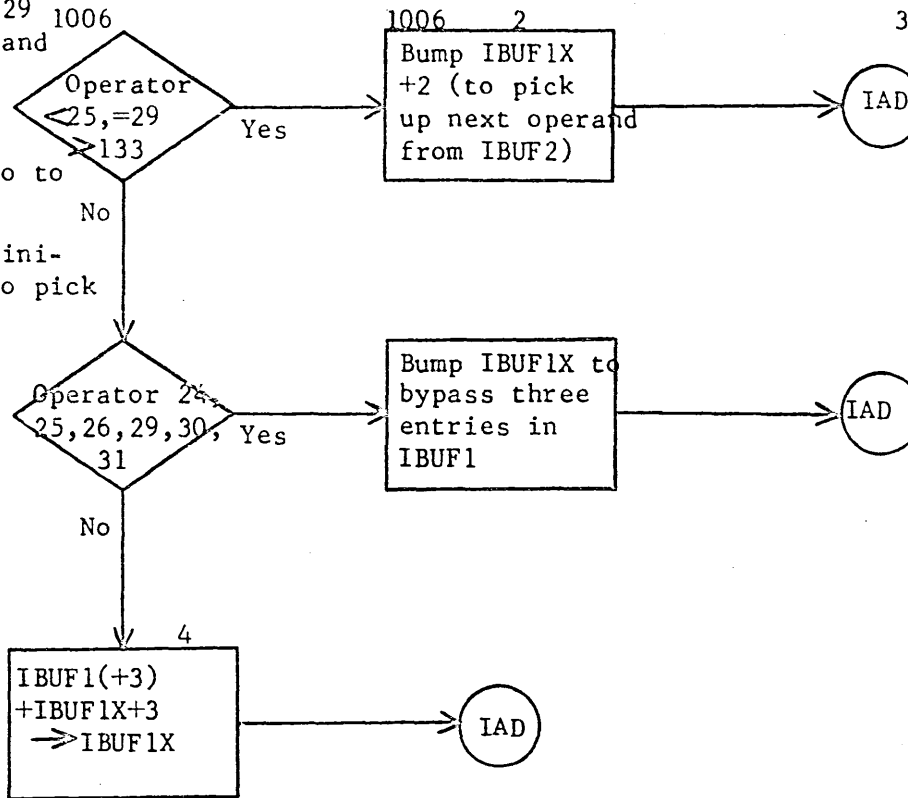
SAMPLE CODE   
FLOWCHART   
DECISION TABLE   
OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	TREE			PROJECT MGR			
		PAGE	1 OF 24	PROJECT NAME			
NUMBER		ISSUE DATE		TASK NO.			
DRAWN BY		DATE	5-6-69	TASK NAME			

2. If operator is 24 or 29 go to statement 1007 and bump +4. If operator is 25, 30; go to 1008 and bump +3. If operator is 26, 31; go to 1009 and bump +1.

3. This return (IAD) is initially made to 1004 to pick up next operand from IBUF1.

4. Add fixed increment at IBUF1 (IBUF1X+3) to IBUF1X setting +3 and return to IBUF1X.



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

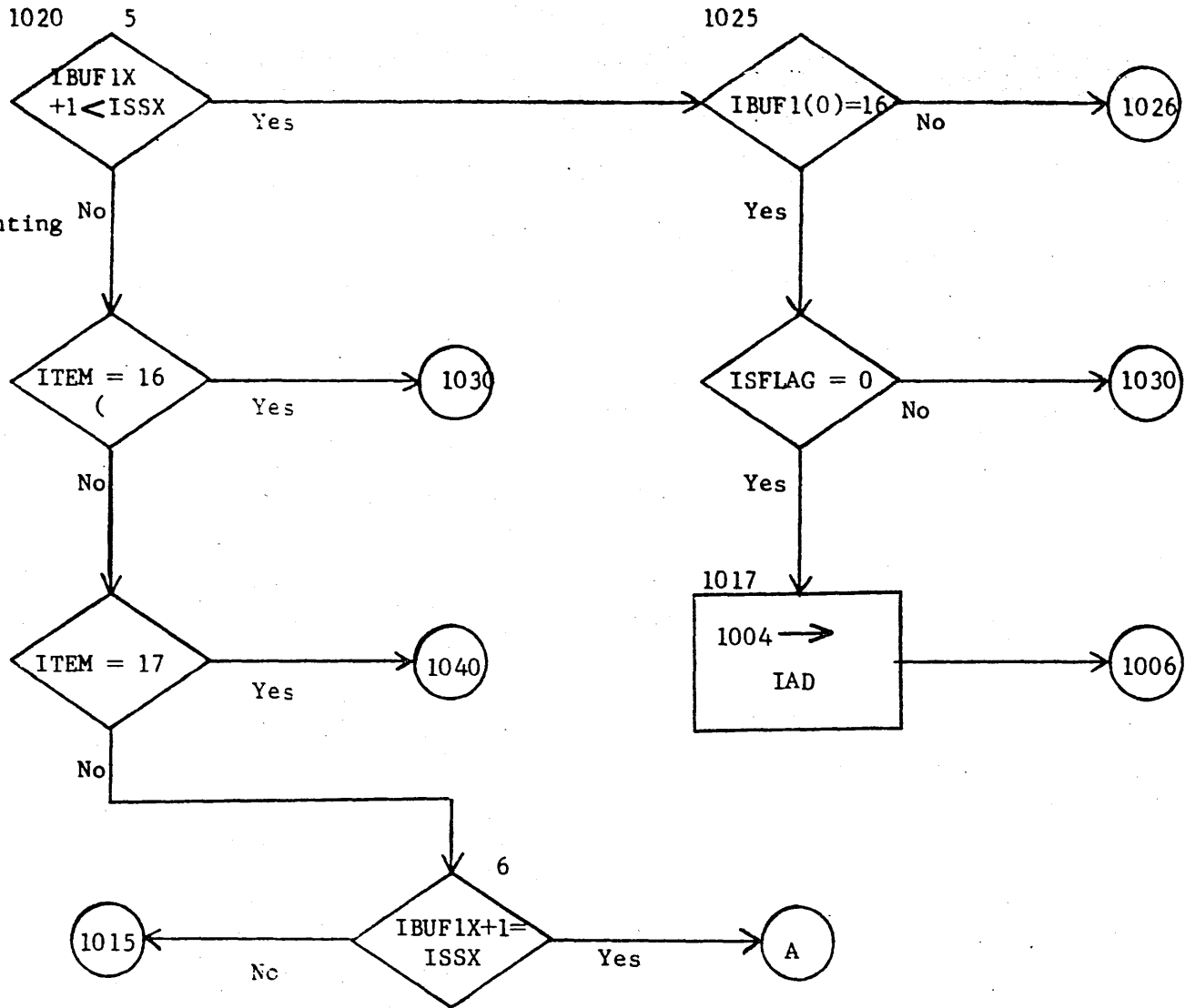
- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS <i>IMS</i>	MACH. TYPE <i>1400</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE <i>TREE</i>		PROJECT MGR.			
	PAGE <i>2</i> OF <i>24</i>	PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.			
DRAWN BY	DATE <i>3-67</i>	TASK NAME			

5. Operator is less than 22 to come here.

6. ISSX = level of current operator, initially 128.

7. ISFLAG = first operator switch used in counting leading left (.).



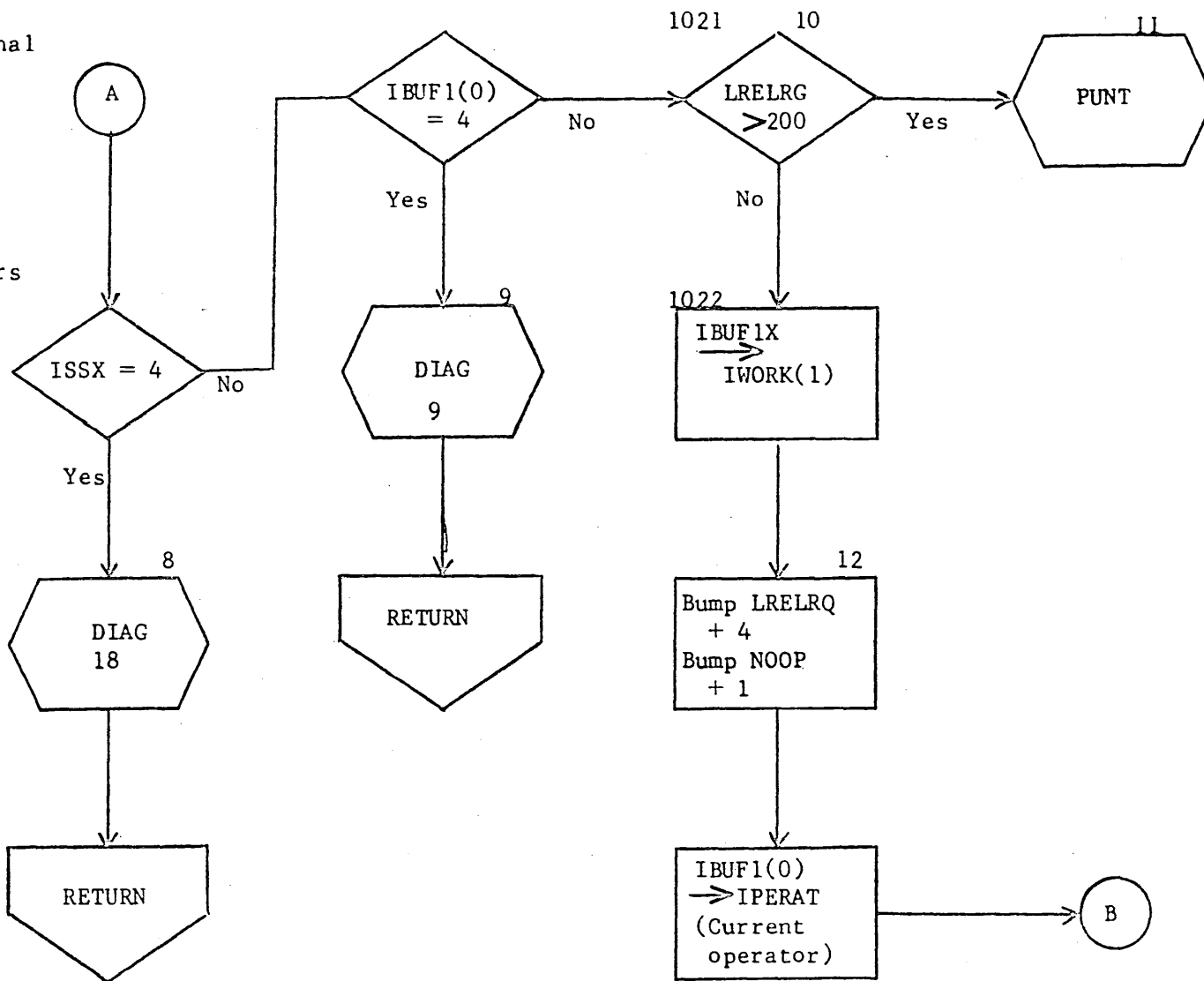
**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	<i>IMS</i>	MACH. TYPE	<i>1700</i>	PROJECT NO.		REV	APPROVED	DATE
DOCUMENT TITLE	<i>TREE</i>			PROJECT MGR.				
			<i>PAGE 3 OF 24</i>	PROJECT NAME				
NUMBER		ISSUE DATE		TASK NO.				
DRAWN BY		DATE	<i>3-67</i>	TASK NAME				



- 8. Illegal use of relational operator.
- 9. Illegal use of NOT.
- 10. LRELQ = IWORK index.
- 11. Error - IWORK table exceeded.
- 12. NOOP=number of operators this level.



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

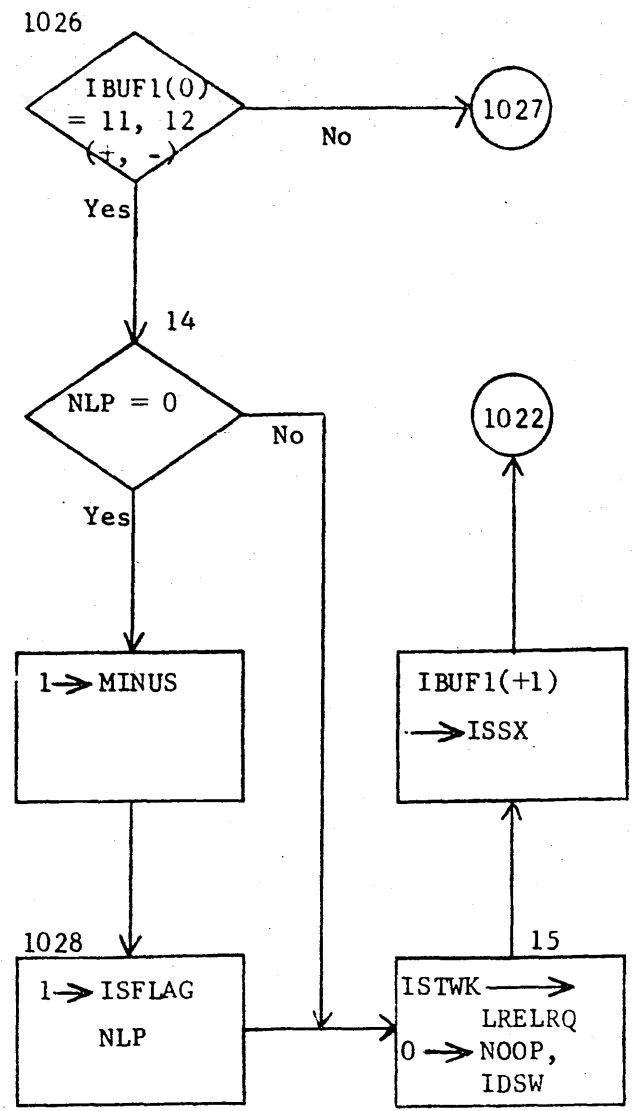
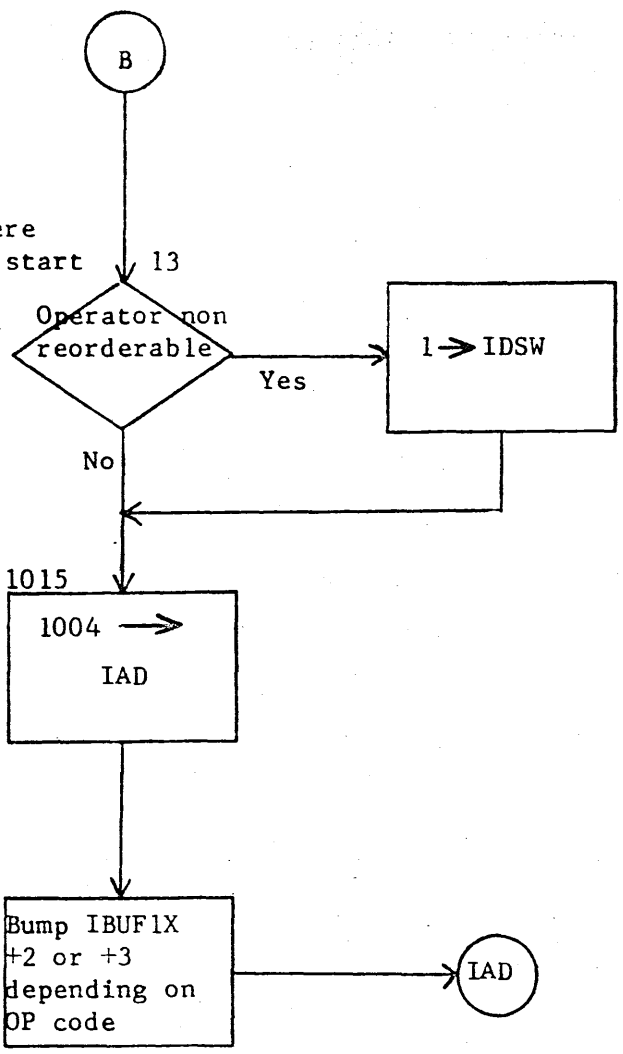
DOCUMENT CLASS <i>TMS</i>	MACH. TYPE <i>1900</i>	PROJECT NO.	REV.	APPROVED	DATE
DOCUMENT TITLE <i>TREE</i>		PROJECT MGR.			
	PAGE <i>4</i> OF <i>24</i>	PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.			
DRAWN BY	DATE <i>3-67</i>	TASK NAME			

13. IPERAT = 14  
 .AND.  
 IBUF1(+2) ≠ 0.

14. NPL = non left parenthesis operator for determining leading sign.

15. ISTWK = IWORK index where entries for this level start

NOOP = number of operators this level.



CONTROL DATA CORPORATION  
 SOFTWARE DOCUMENT

SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

DOCUMENT CLASS	IAS	MACH TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	TREE			PROJECT MGR.			
		PAGE	5 OF 24	PROJECT NAME			
NUMBER		ISSUE DATE		TASK NO.			
DRAWN BY		DATE	3-67	TASK NAME			

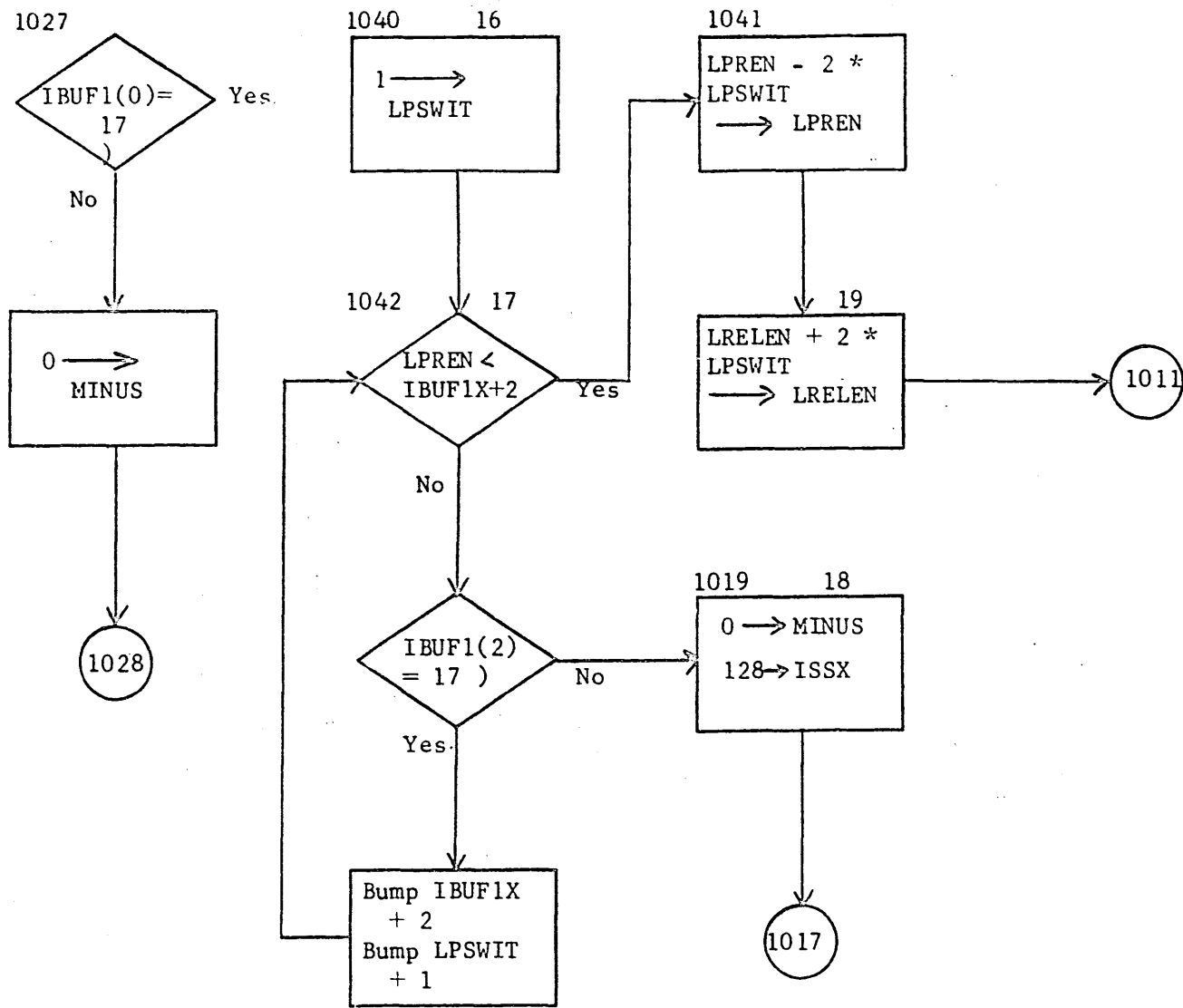
16. LPSWIT is count of non- 1027 imbedded ).

17. Is this end of expression?

18. Reset level back to maximum.

19. LRELEN = IBUF1 index at which to start current subexpression analysis.

LPREN = IBUF1X value at which to stop current subexpression analysis.



CONTROL DATA CORPORATION.

SOFTWARE DOCUMENT

SAMPLE CODE

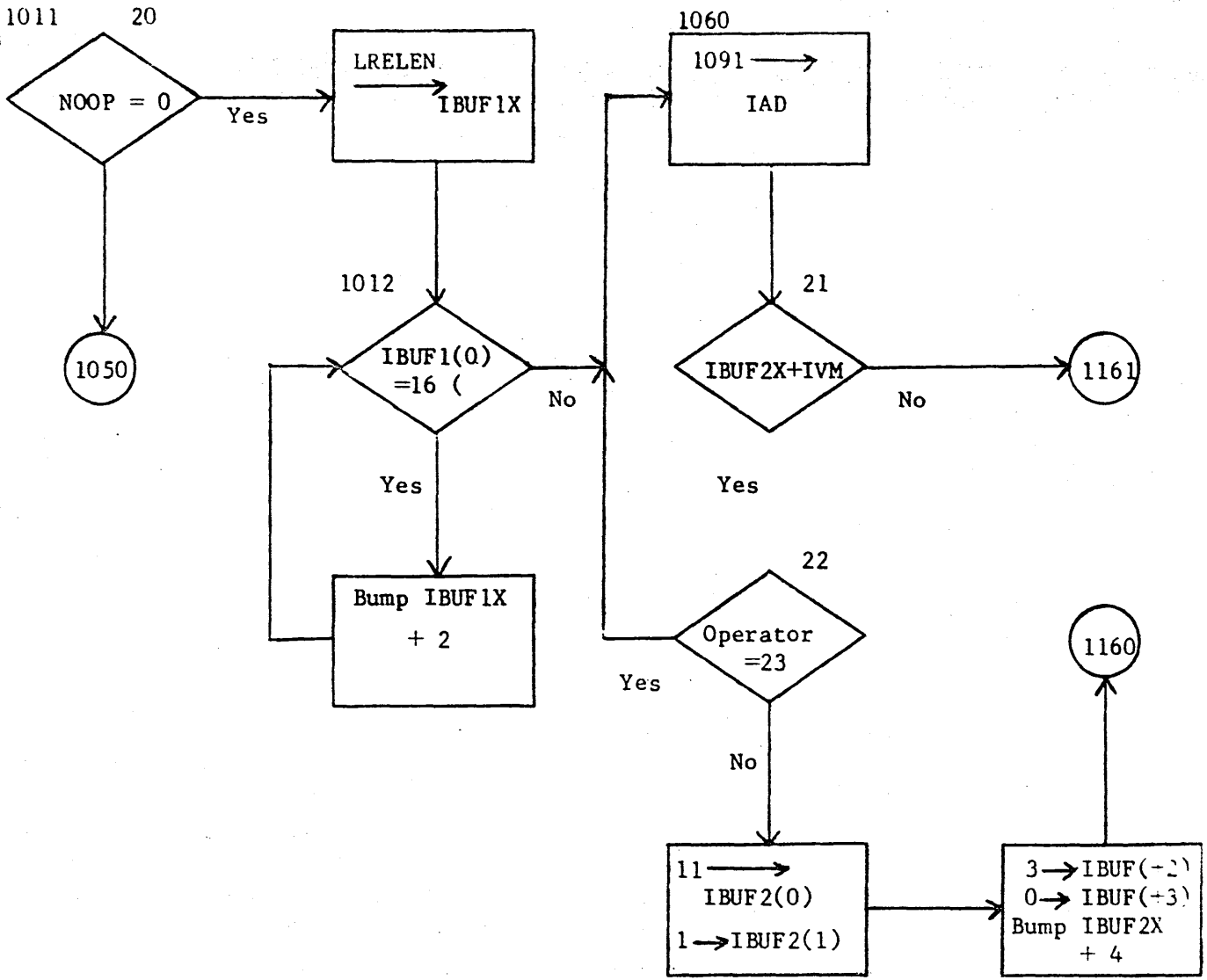
FLOWCHART

DECISION TABLE

OTHER

DOCUMENT CLASS <i>ZMS</i>	MACH. TYPE <i>M400</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE <i>TREE</i>		PROJECT MGR.			
	PAGE <i>6</i> OF <i>26</i>	PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.			
DRAWN BY	DATE <i>3-67</i>	TASK NAME			

20. End of expression comes here.
21. IVM = start of expression in IBUF2.
22. Subroutine with no argument.

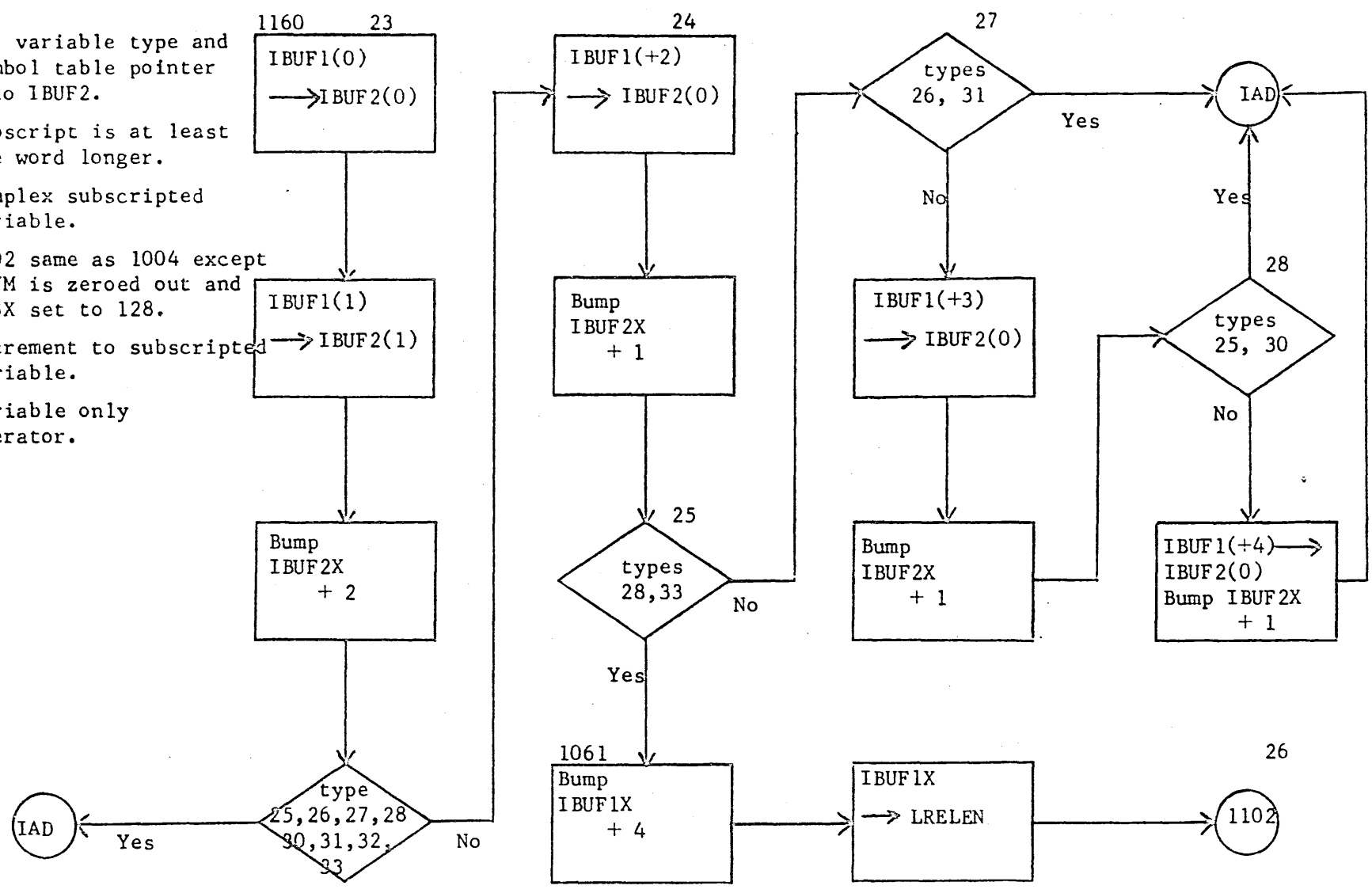


**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	<i>IMS</i>	MACH. TYPE	<i>1706</i>	PROJECT NO.		REV		APPROVED		DATE
DOCUMENT TITLE	<i>TREE</i>			PROJECT MGR.						
			PAGE <i>7</i> OF <i>24</i>	PROJECT NAME						
NUMBER		ISSUE DATE		TASK NO.						
DRAWN BY		DATE	<i>3-67</i>	TASK NAME						

- 23. Put variable type and symbol table pointer into IBUF2.
- 24. Subscript is at least one word longer.
- 25. Complex subscripted variable.
- 26. 1002 same as 1004 except JDUM is zeroed out and ISSX set to 128.
- 27. Increment to subscripted variable.
- 28. Variable only operator.



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE

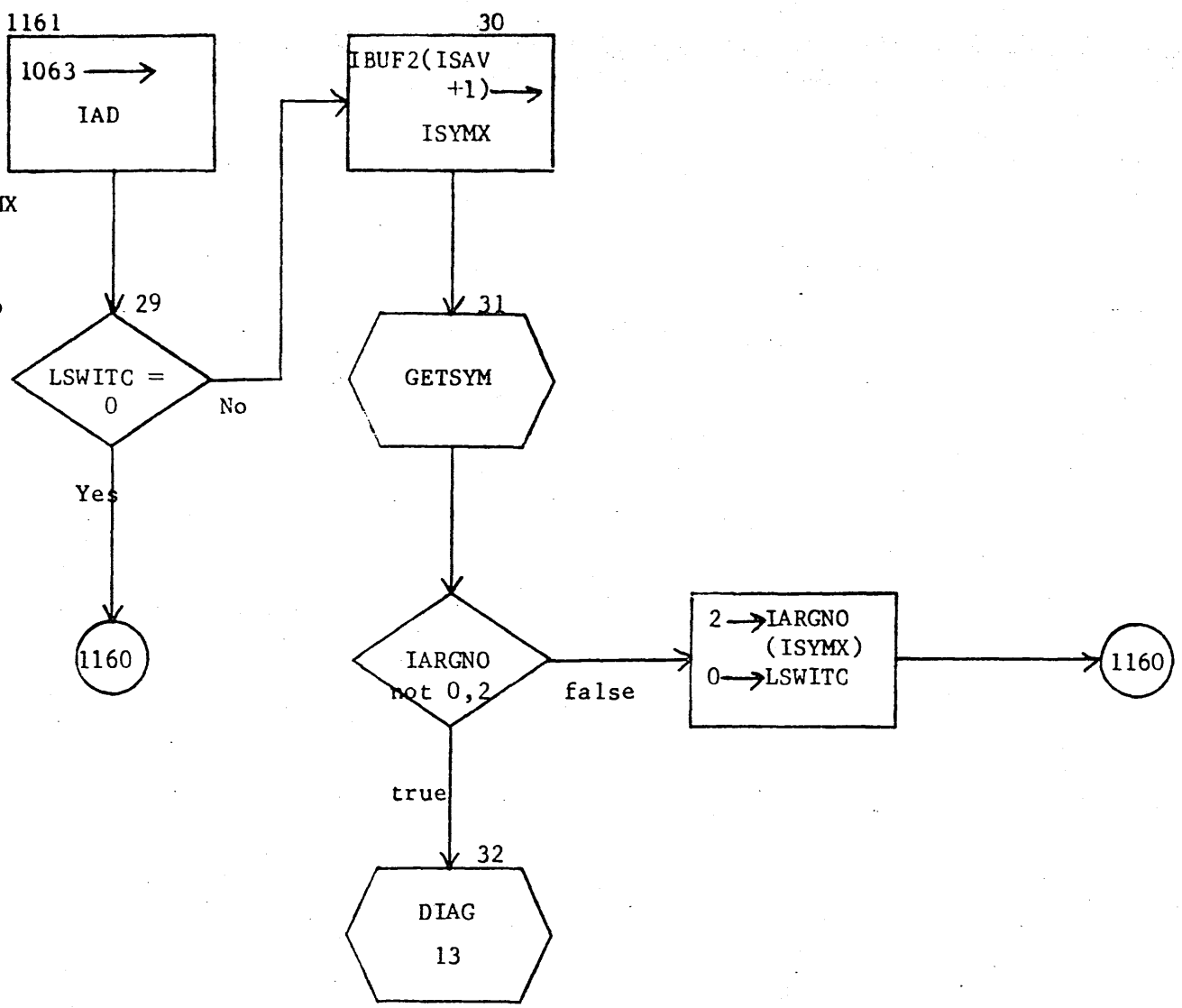
FLOWCHART

DECISION TABLE

OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1900	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	TREE			PROJECT MGR.			
		PAGE	8 OF 24	PROJECT NAME			
NUMBER		ISSUE DATE		TASK NO.			
DRAWN BY		DATE	3-67	TASK NAME			

- 29. Last operator was a function switch.
- 30. ISQV = index to operator this level.
- 31. Get page and symbol table pointer that ISYMX points to.
- 32. Number of arguments differ in references to same subprogram.



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE

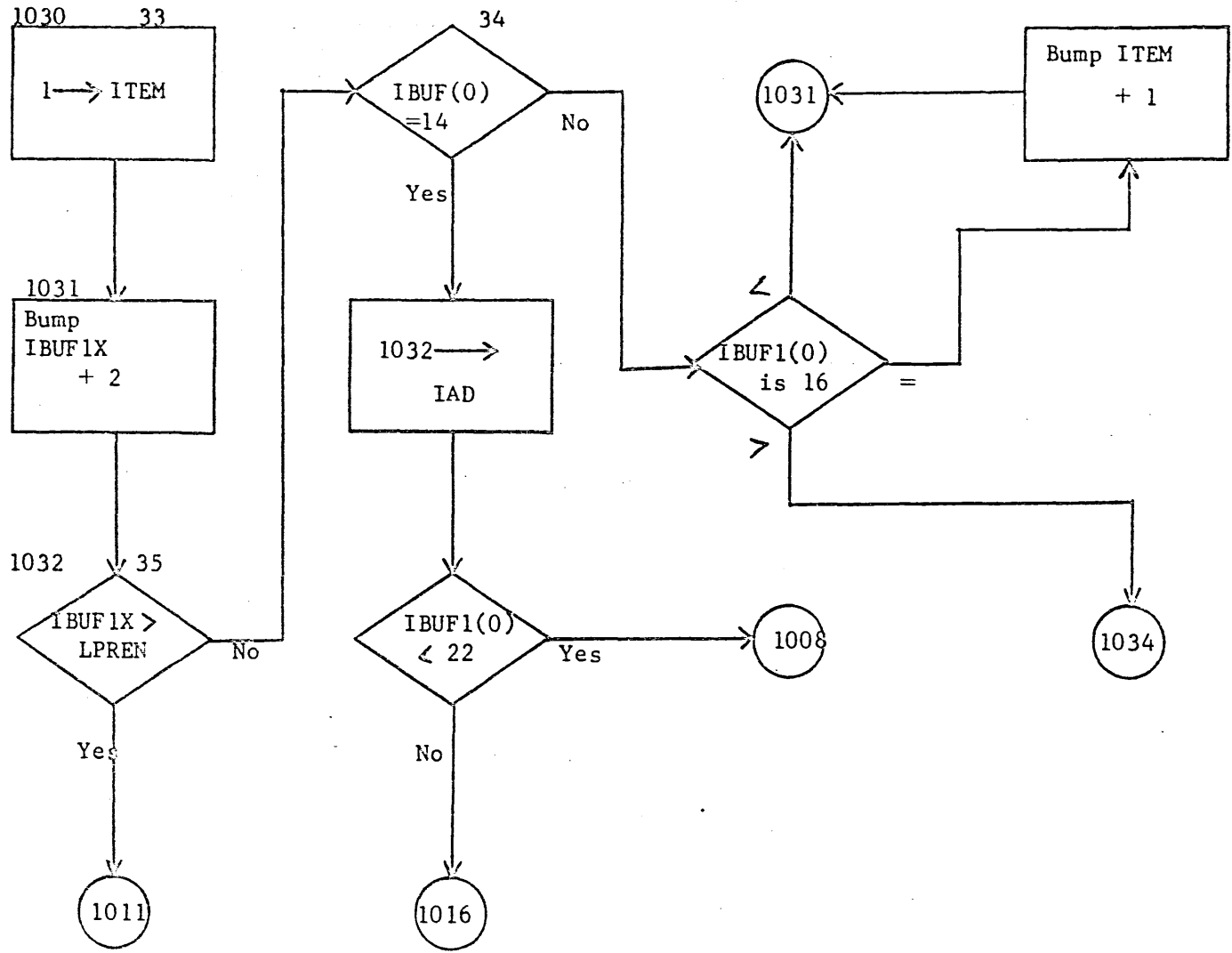
FLOWCHART

DECISION TABLE

OTHER

DOCUMENT CLASS <i>IMS</i>	MACH. TYPE <i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE <i>TREE</i>		PROJECT MGR.			
	PAGE <i>9</i> OF <i>24</i>	PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.			
DRAWN BY	DATE <i>3-67</i>	TASK NAME			

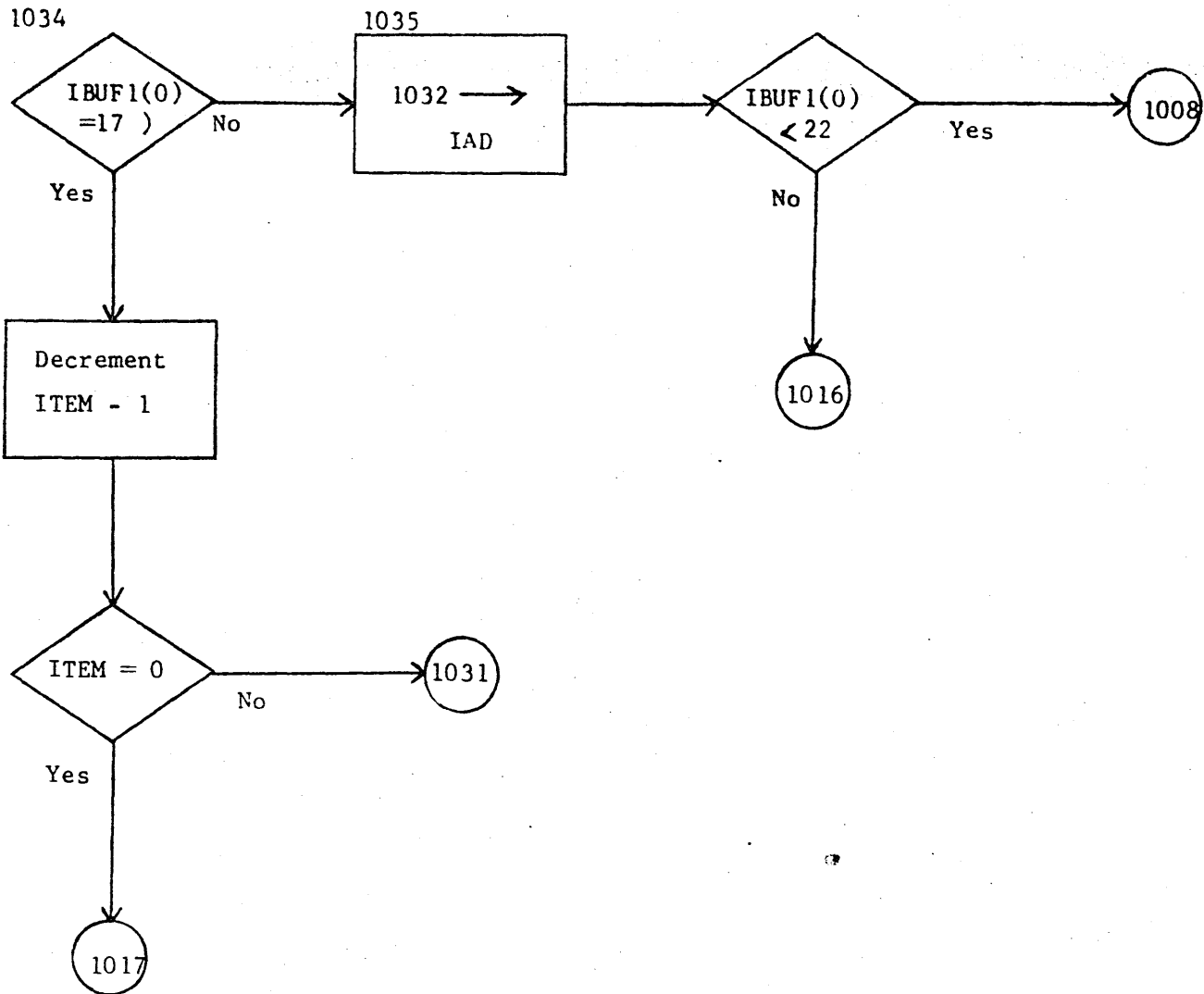
- 33. We found an imbedded (, slew to ).
- 34. Non-reorderable.
- 35. End of expression.



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS <i>ZUS</i>	MACH. TYPE <i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE <i>TREE</i>		PROJECT MGR.			
	PAGE <i>130</i> OF <i>286</i>	PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.			
DRAWN BY	DATE <i>3-1-69</i>	TASK NAME			



**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

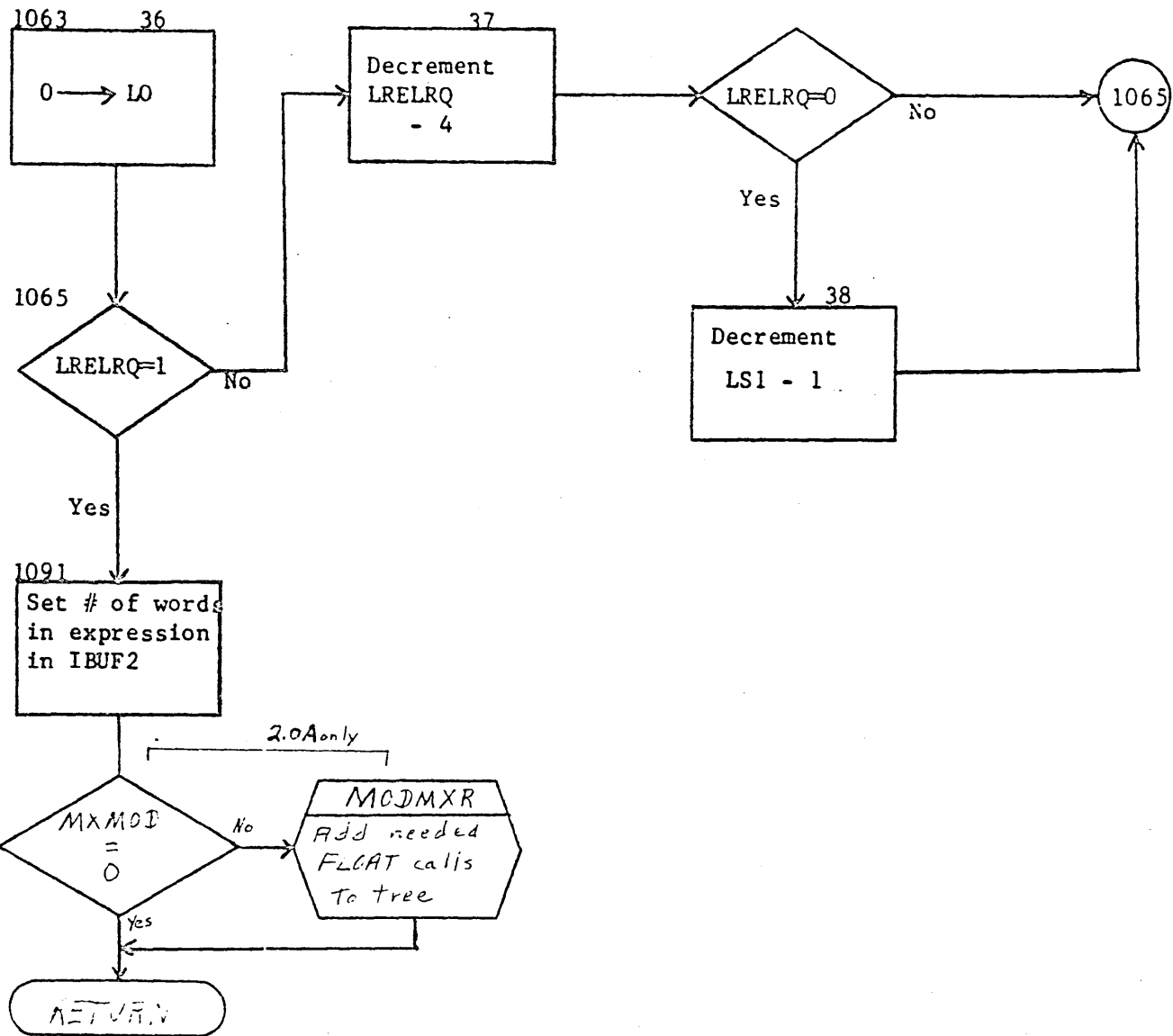
DOCUMENT CLASS	<i>JMS</i>	MACH. TYPE	<i>1760</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	<i>TREE</i>			PROJECT MGR.			
			<i>PAGE 11 OF 24</i>	PROJECT NAME			
NUMBER		ISSUE DATE		TASK NO.			
DRAWN BY		DATE	<i>3-67</i>	TASK NAME			



36. Gotten to by assign IAD.

37. IWORK index.

38. NOT operator in effect switch.



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

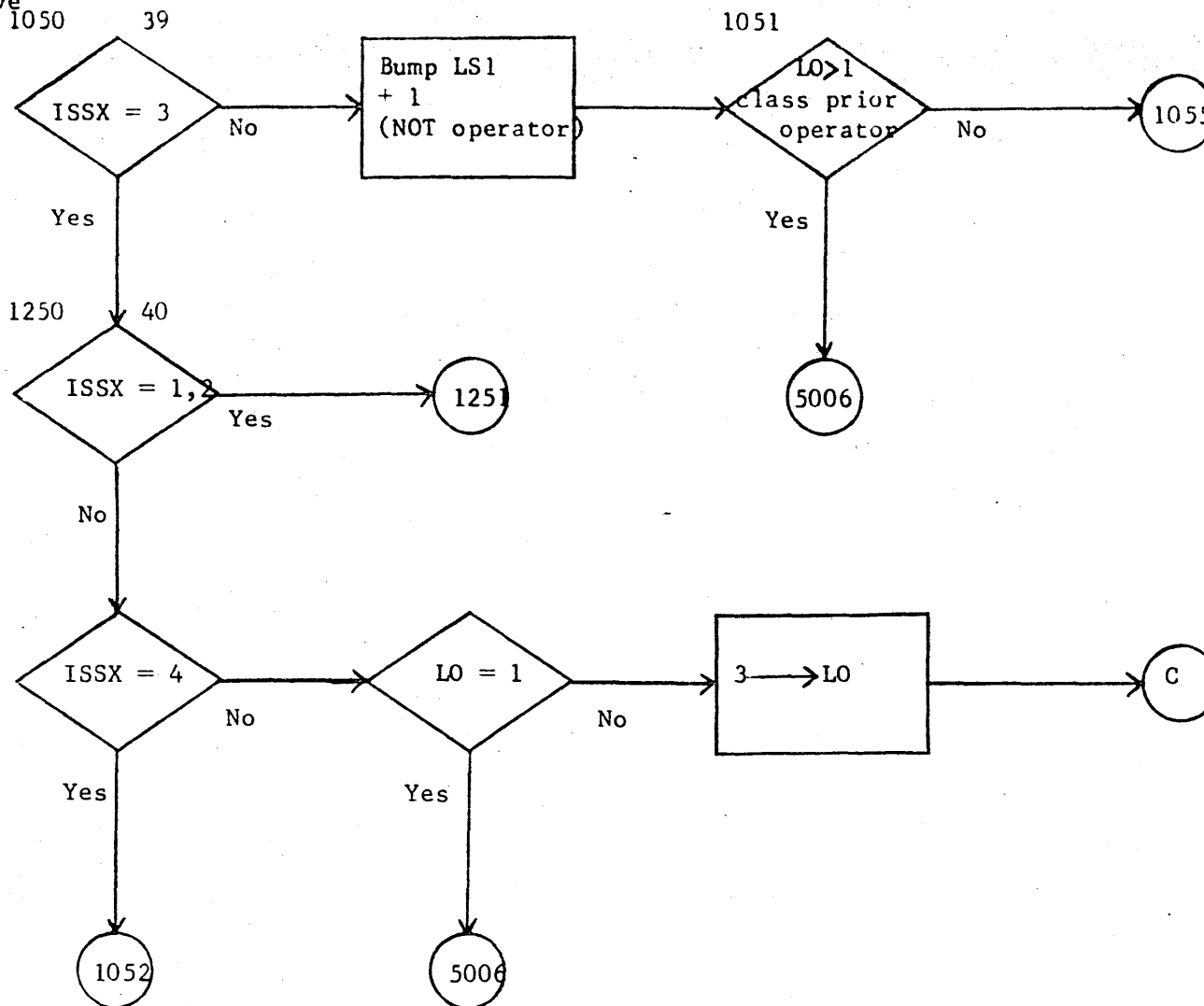
SAMPLE CODE  
FLOWCHART  
DECISION TABLE  
OTHER

DOCUMENT CLASS	<i>INS</i>	MACH. TYPE	<i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	<i>TREE</i>			PROJECT MGR			
		PAGE	<i>12 of 24</i>	PROJECT NAME			
NUMBER		ISSUE DATE		TASK NO.			
DRAWN BY		DATE	<i>3-67</i>	TASK NAME			

39. Number of operators have just been found to be NZ. (NOOP).

40. ISSX = level of current operator.

Level	Operator
0	,
1	OR
2	AND
3	NOT
4	relational
5	+ -
6	* /
7	**
8	( ) function

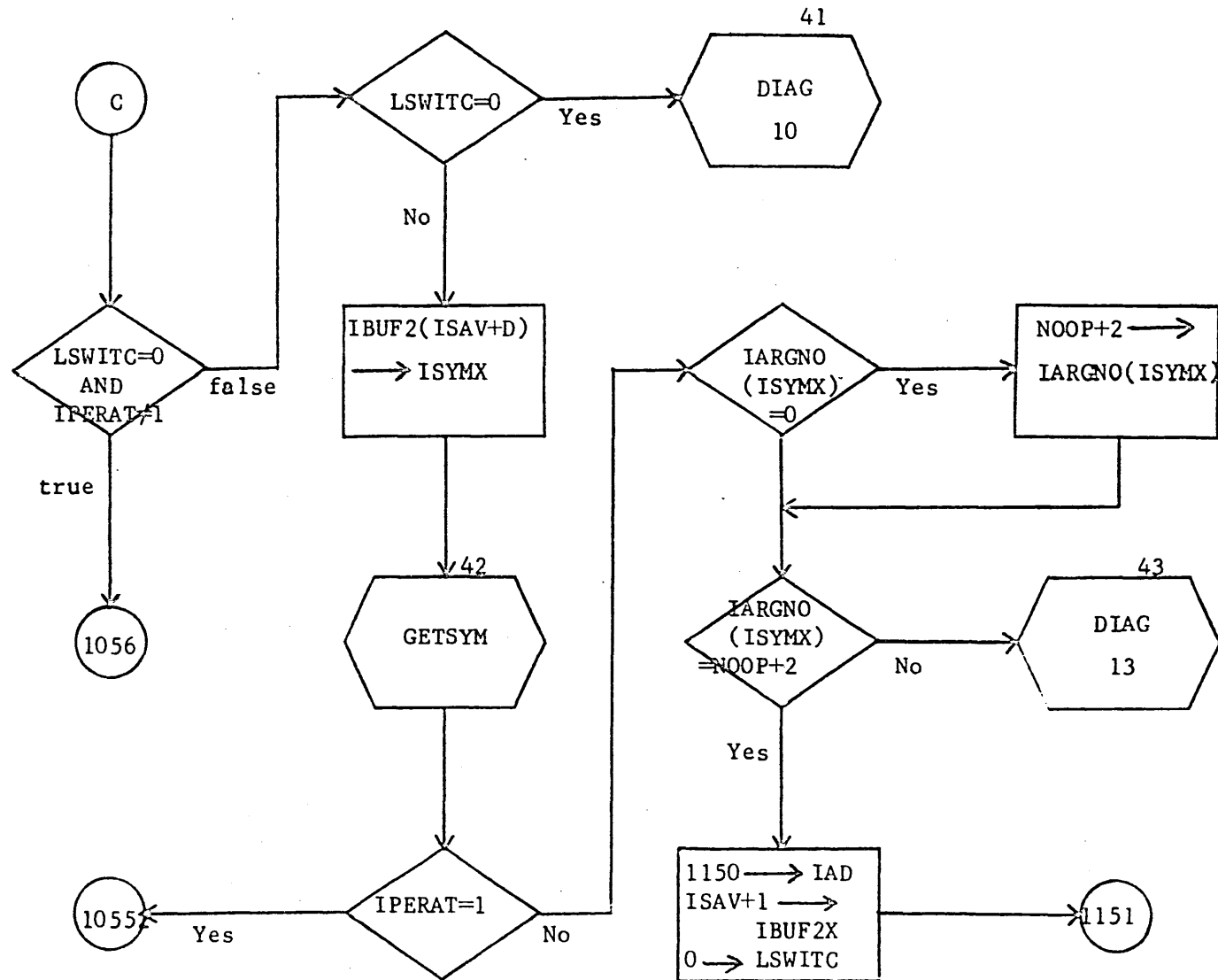


CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE   
FLOWCHART   
DECISION TABLE   
OTHER

DOCUMENT CLASS	<i>EHS</i>	MACH. TYPE	<i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	<i>TREE</i>			PROJECT MGR.			
		PAGE	<i>13 of 24</i>	PROJECT NAME			
NUMBER		ISSUE DATE		TASK NO.			
DRAWN BY		DATE	<i>3-67</i>	TASK NAME			

- 41. Illegal operator or operand.
- 42. Get page # and symbol position.
- 43. # of arguments differ in references to same subprogram.

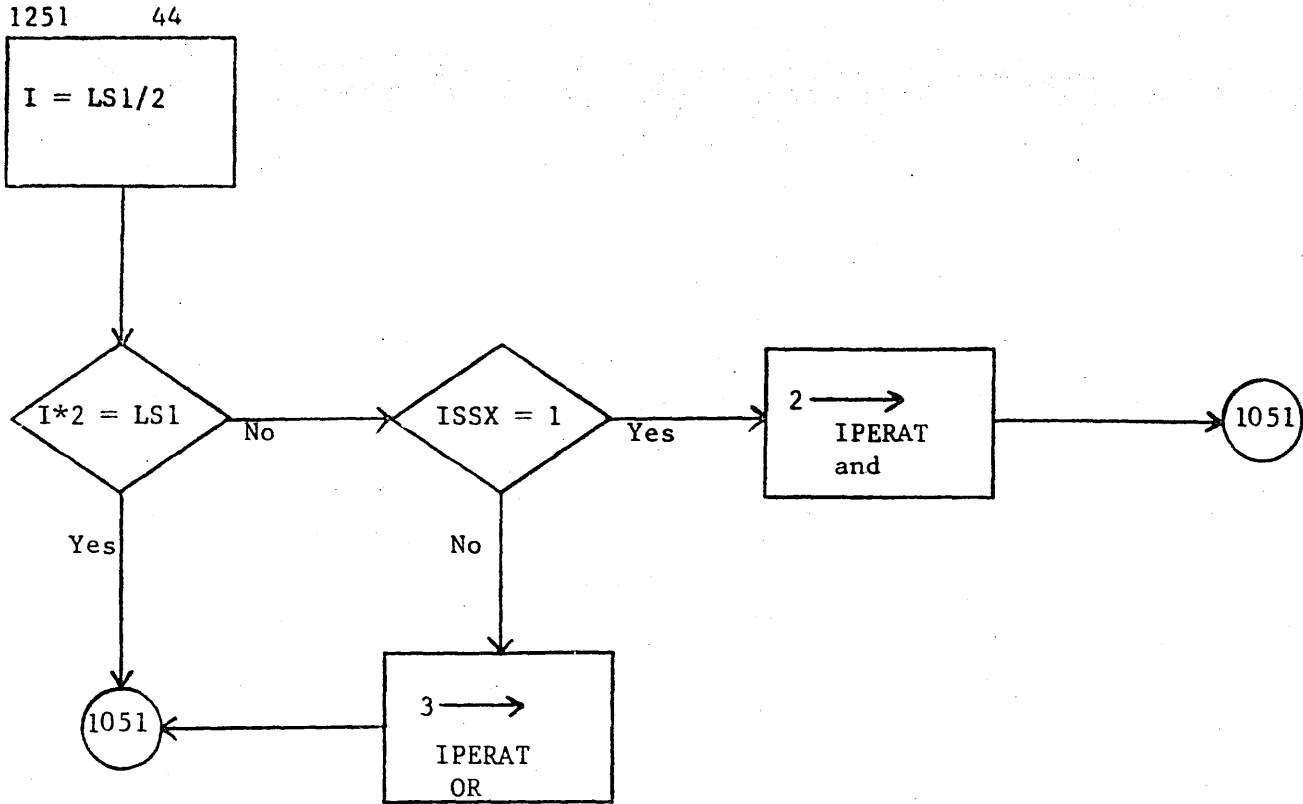


CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	<i>IMS</i>	MACH. TYPE	<i>1700</i>	PROJECT NO.		REV	APPROVED	DATE
DOCUMENT TITLE	<i>TREE</i>			PROJECT MGR.				
		PAGE	<i>14 of 24</i>	PROJECT NAME				
NUMBER		ISSUE DATE		TASK NO.				
DRAWN BY		DATE	<i>3-1-7</i>	TASK NAME				

44. Look for odd numbers  
(nested nots).



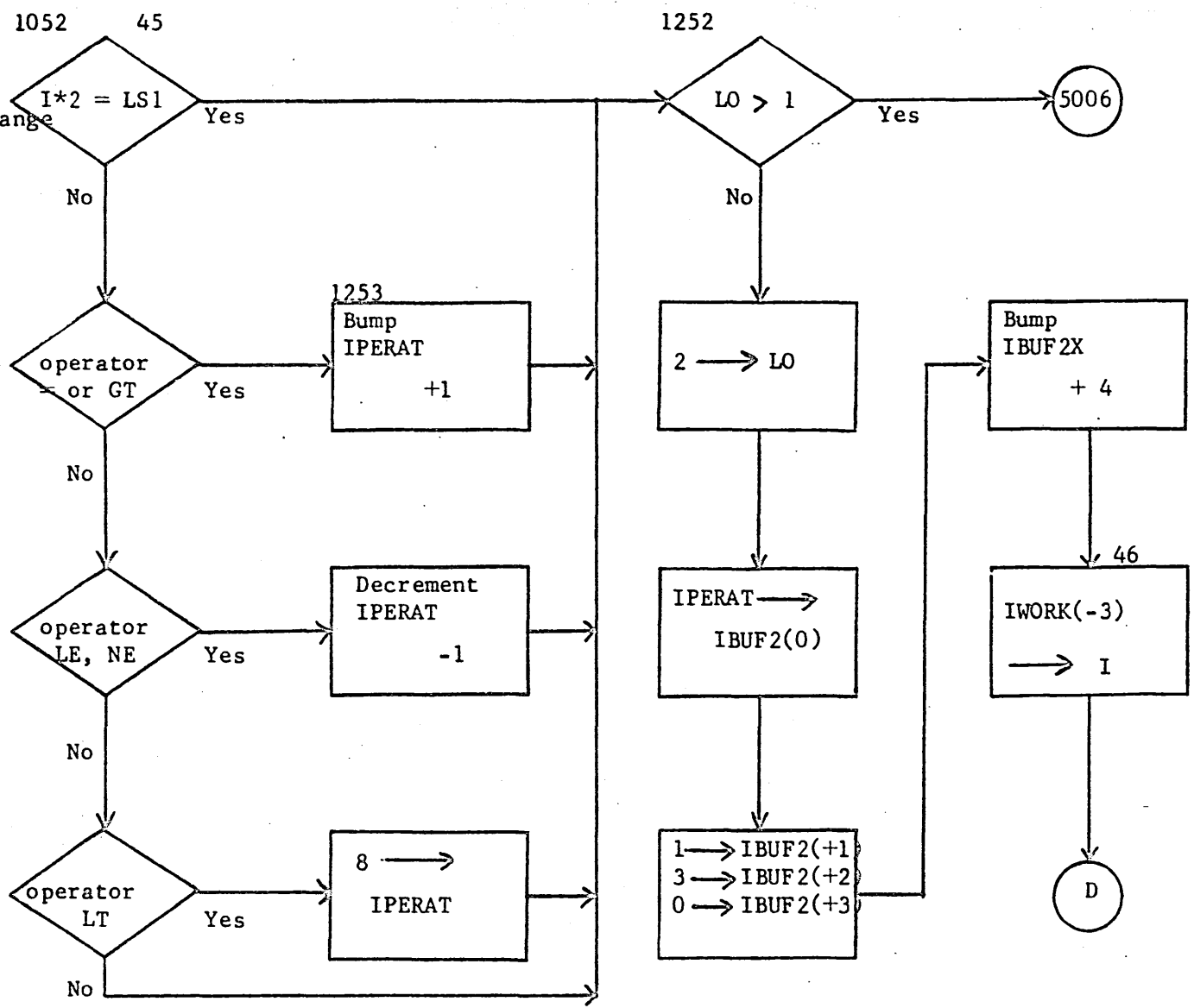
**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

DOCUMENT CLASS	<i>IMS</i>	MACH. TYPE	<i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	<i>TREE</i>	PAGE <i>15</i> OF <i>24</i>		PROJECT MGR.			
NUMBER		ISSUE DATE		TASK NO.			
DRAWN BY		DATE	<i>3-67</i>	TASK NAME			

45. ISSX contains a relational operator (page 13).

46. If written, A<sub>i</sub>RL, B, change to .RL. A + B, else make it .RL. A - B.



**CONTROL DATA CORPORATION**  
 SOFTWARE DOCUMENT

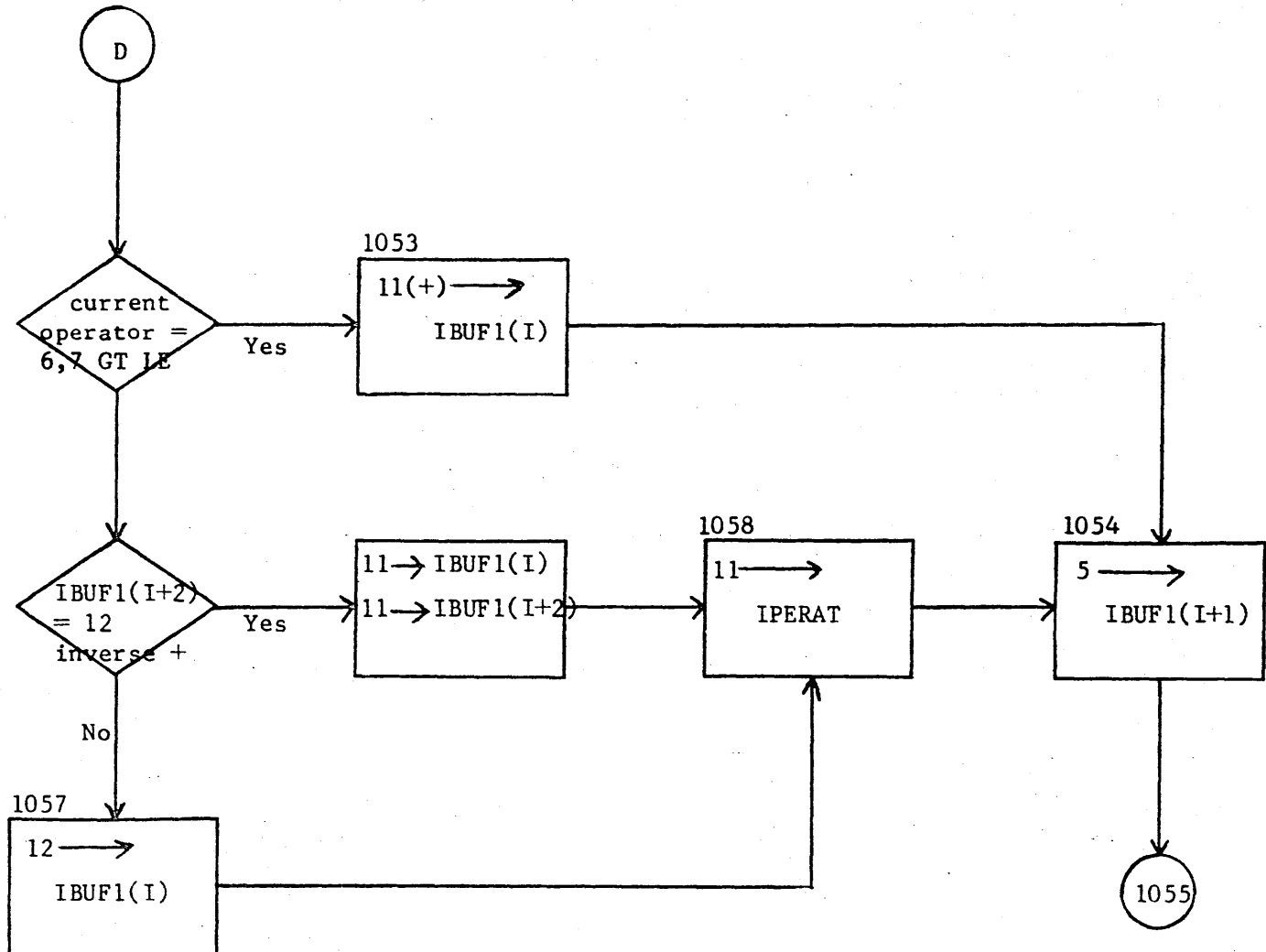
SAMPLE CODE

FLOWCHART

DECISION TABLE

OTHER

DOCUMENT CLASS	<i>IMS</i>	MACH. TYPE	<i>1900</i>	PROJECT NO.		APPROVED	DATE
DOCUMENT TITLE	<i>TREE</i>			PROJECT MGR.			
			<i>PAGE 16 OF 26</i>	PROJECT NAME			
NUMBER		ISSUE DATE		TASK NO.			
DRAWN BY		DATE	<i>3-67</i>	TASK NAME			



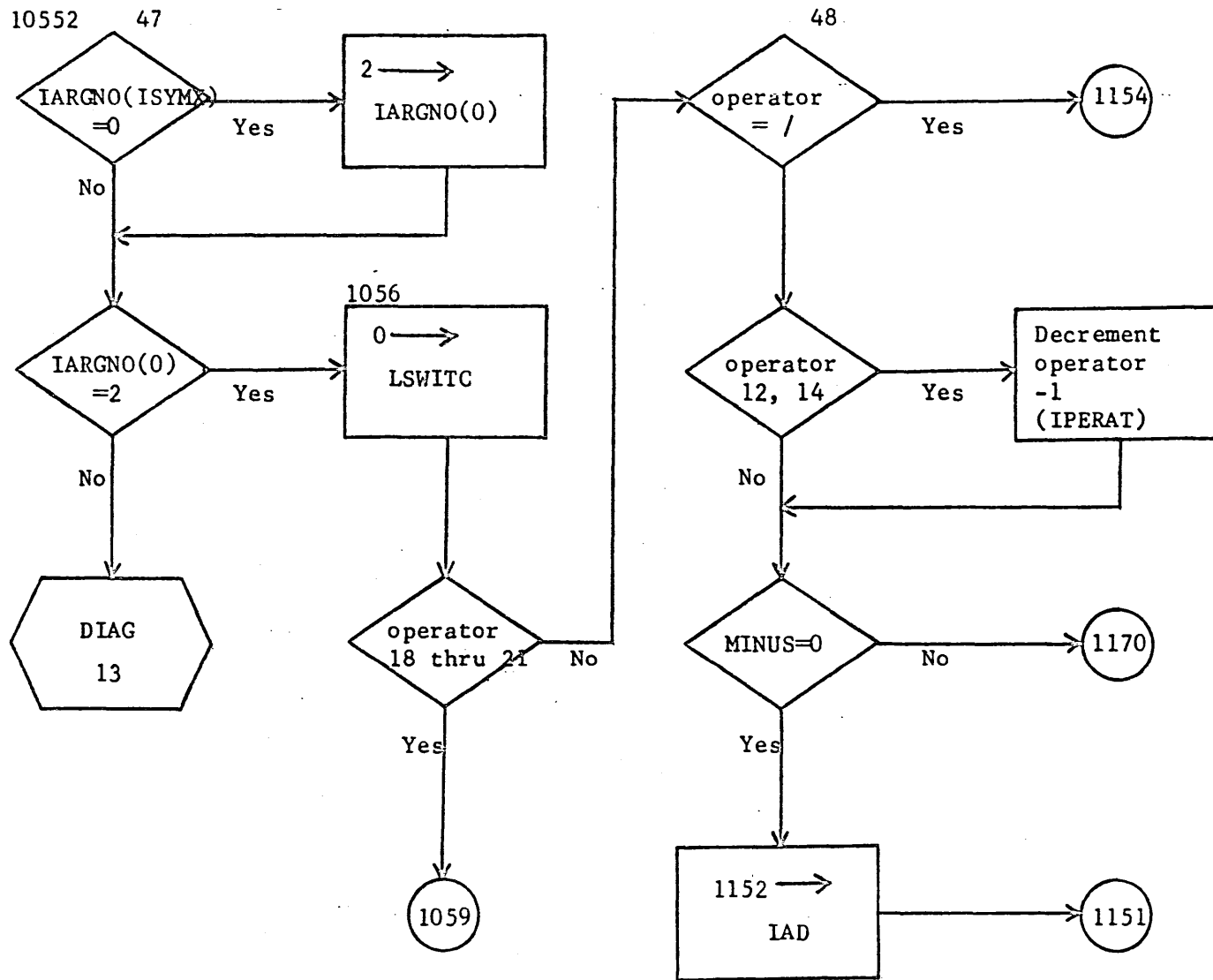
**CONTROL DATA CORPORATION**

SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	<i>IMS</i>	MACH. TYPE	<i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	<i>TREE</i>	PAGE <i>17</i> OF <i>24</i>		PROJECT MGR.			
NUMBER		ISSUE DATE		PROJECT NAME			
DRAWN BY		DATE	<i>3-67</i>	TASK NO.			
				TASK NAME			

47. Check number of parameters this call vs. number of parameters for any prior call.
48. Operator = 1, MINUS set NZ, NOOP = 1.



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE

FLOWCHART

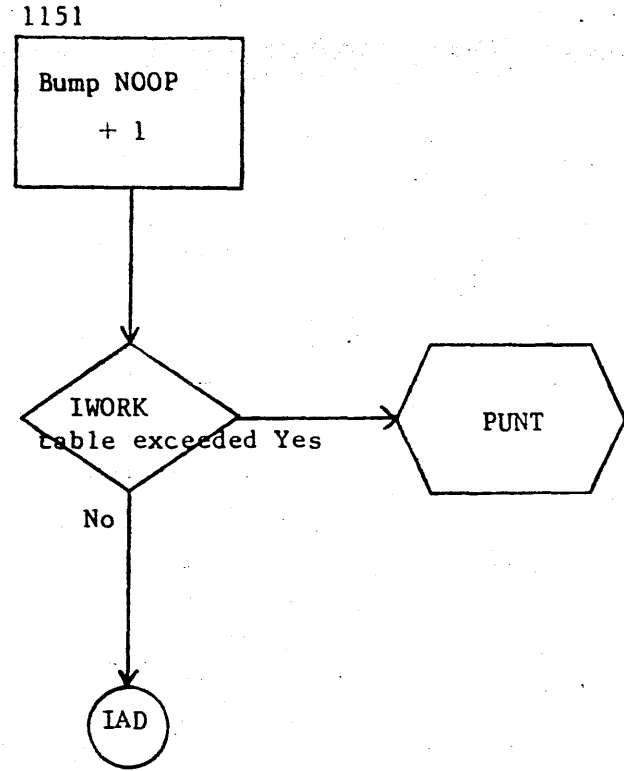
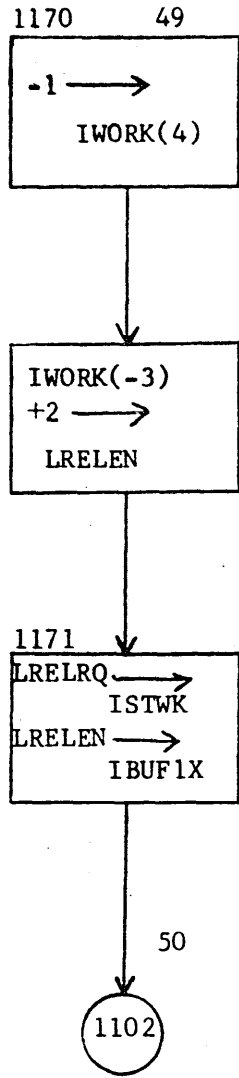
DECISION TABLE

OTHER

DOCUMENT CLASS <i>ZMS</i>	MACH. TYPE <i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE <i>TREE</i>		PROJECT MGR.			
	PAGE <i>18</i> OF <i>24</i>	PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.			
DRAWN BY	DATE <i>3-6-7</i>	TASK NAME			

49. NOT operator set up in IWORK but not in IBUF2.

50. Back to start.



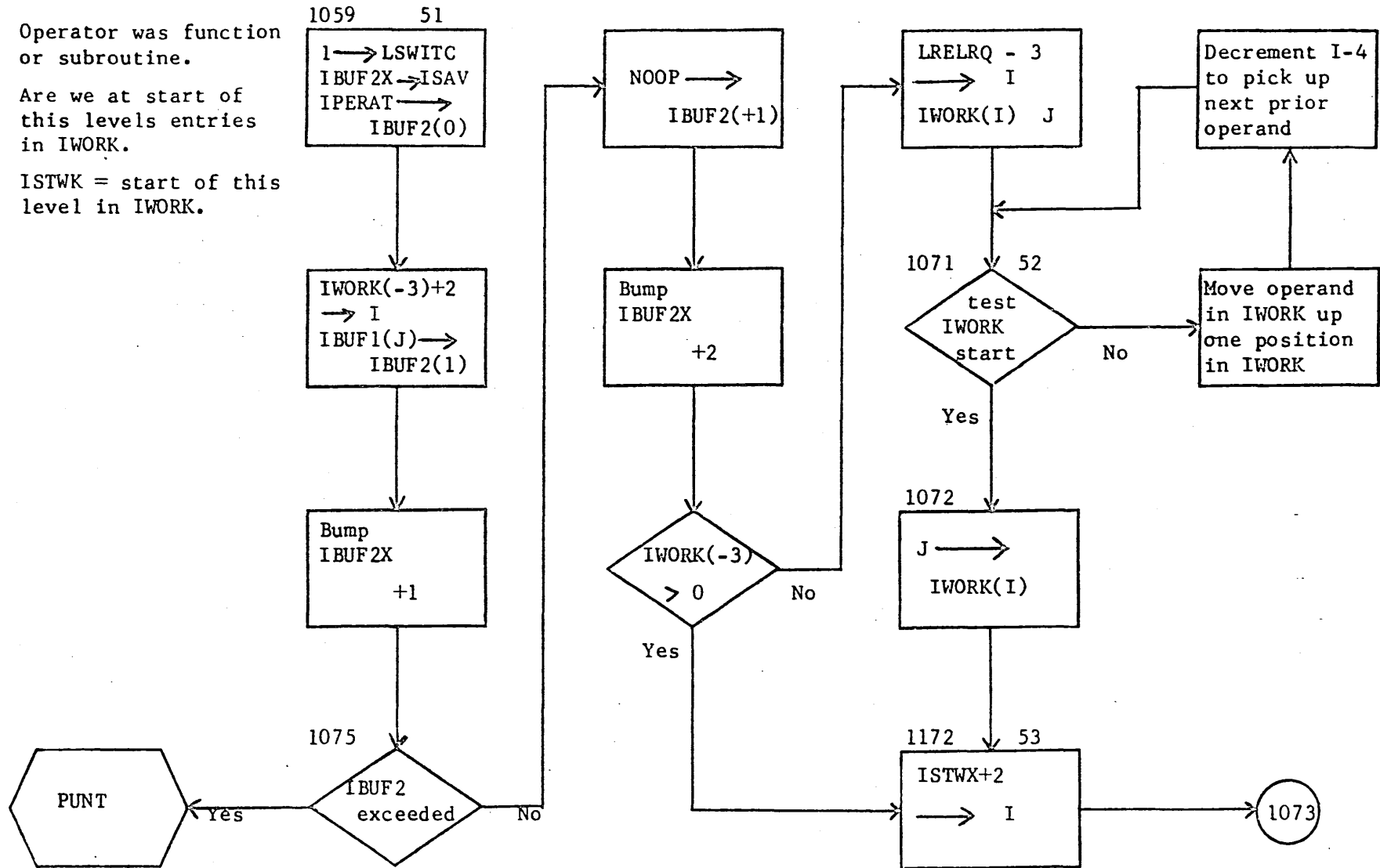
CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

DOCUMENT CLASS	<i>INS</i>	MACH. TYPE	<i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	<i>TREE</i>	PAGE <i>19</i> OF <i>24</i>		PROJECT MGR.			
NUMBER	ISSUE DATE			PROJECT NAME			
DRAWN BY	DATE <i>5-67</i>			TASK NO.			
				TASK NAME			



51. Operator was function or subroutine.  
 52. Are we at start of this levels entries in IWORK.  
 53. ISTWK = start of this level in IWORK.



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

- SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

DOCUMENT CLASS <i>IMS</i>	MACH. TYPE <i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE <i>TRAC</i>		PROJECT MGR.			
	PAGE <i>20</i> OF <i>24</i>	PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.			
DRAWN BY	DATE <i>3-67</i>	TASK NAME			

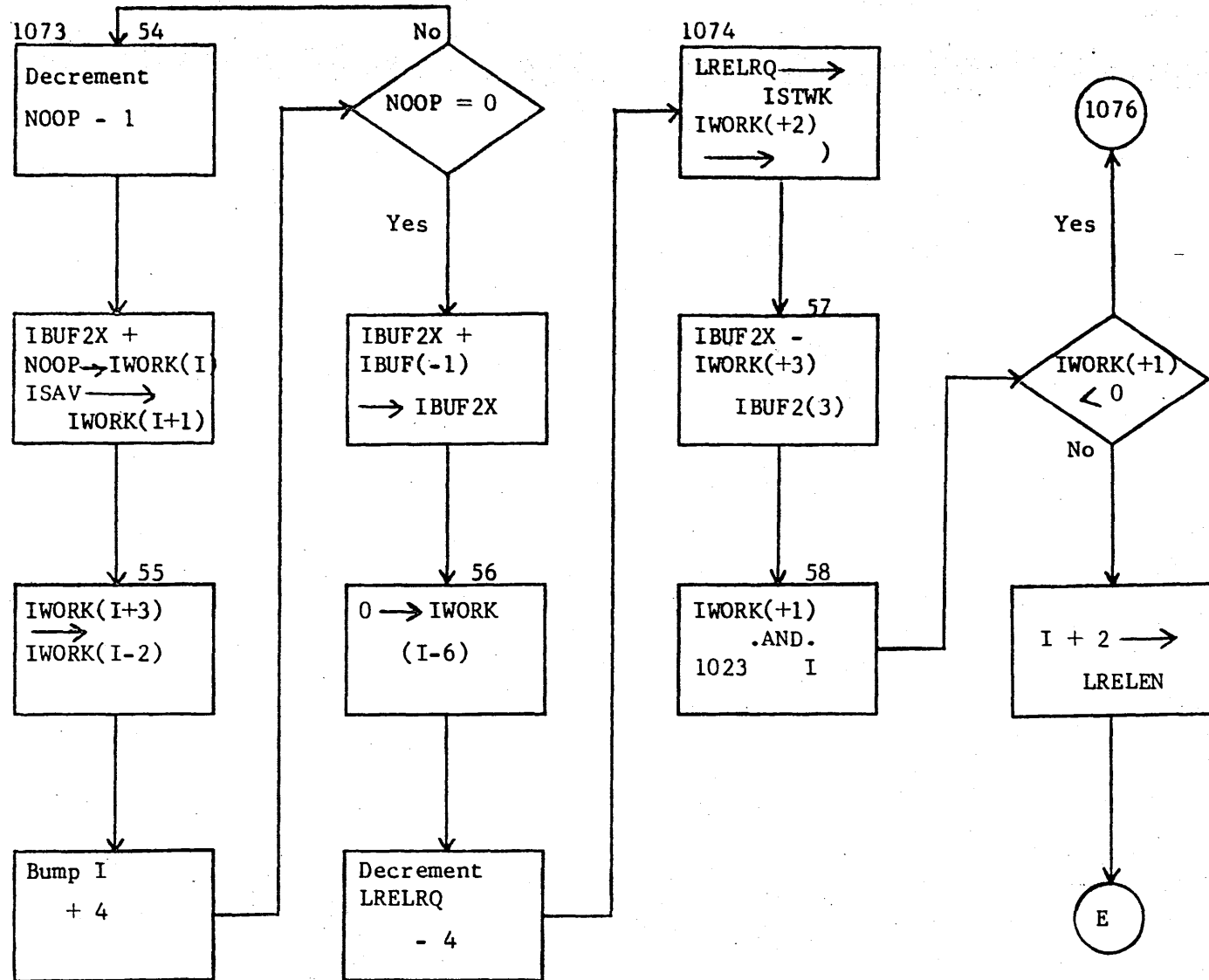
54. Put in IBUF2, settings for operand + operator in IWORK.

55. Put in stop pointer.

56. Clear out last stop pointer.

57. J = location of PTR in IBUF2.

58. Set new starting item.



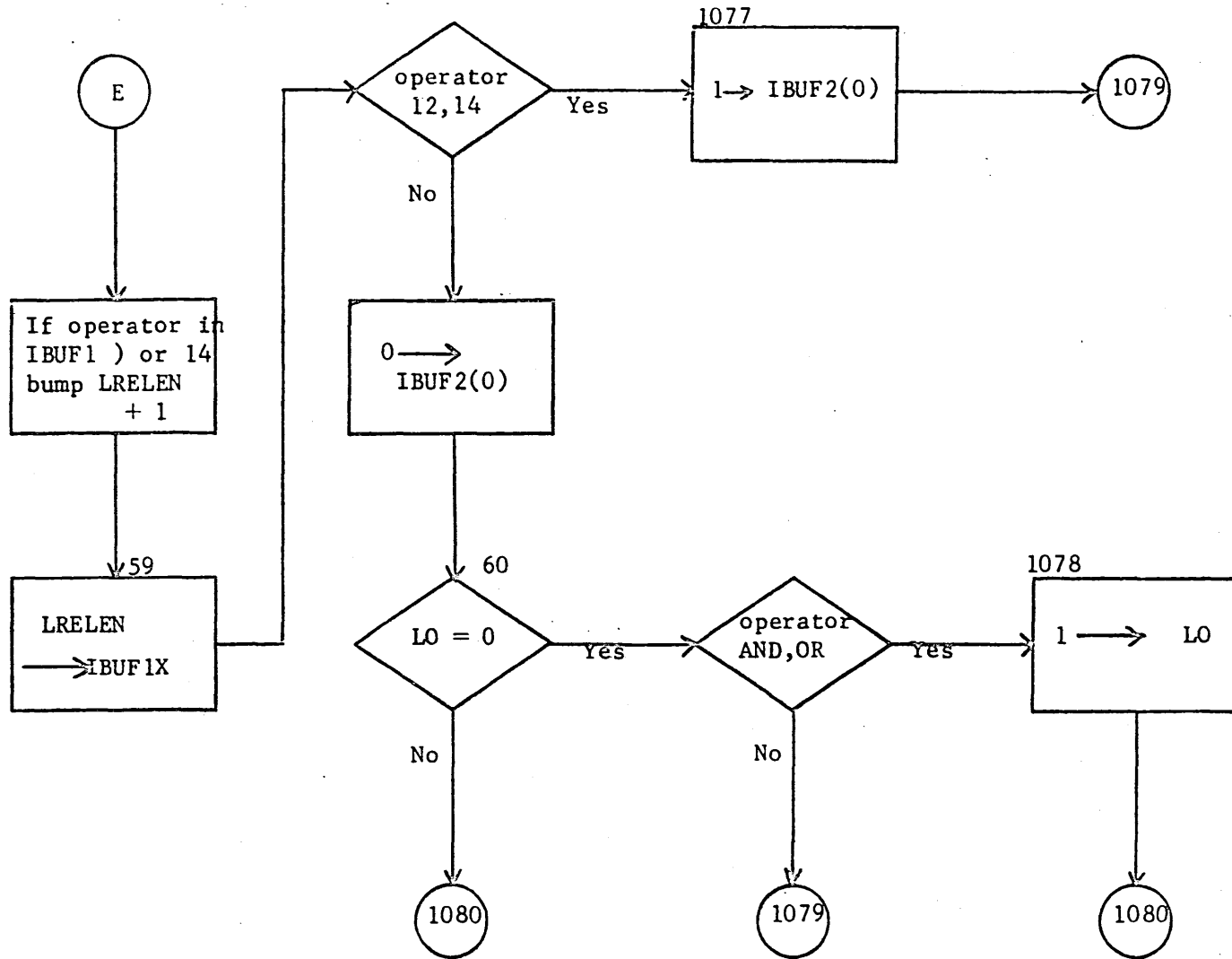
CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE   
FLOWCHART   
DECISION TABLE   
OTHER

DOCUMENT CLASS	<i>ZMS</i>	MACH. TYPE	<i>M700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	<i>TR3E</i>	PAGE <i>21</i> OF <i>24</i>		PROJECT MGR.			
NUMBER	ISSUE DATE	DATE <i>3-67</i>		TASK NO.			
DRAWN BY	TASK NAME						

59. Set new IBUF1 index.

60. LO = class of prior operand.

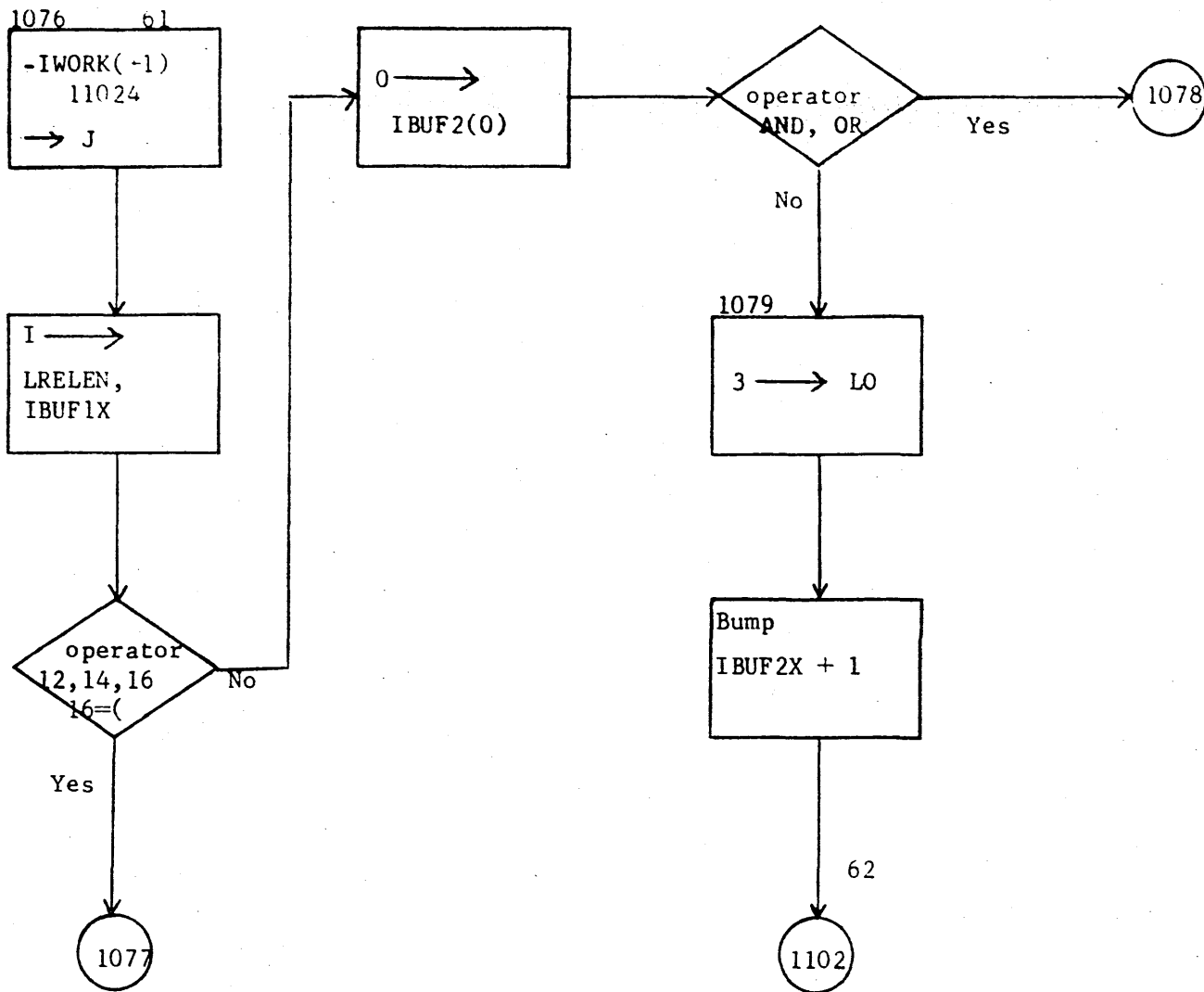


**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

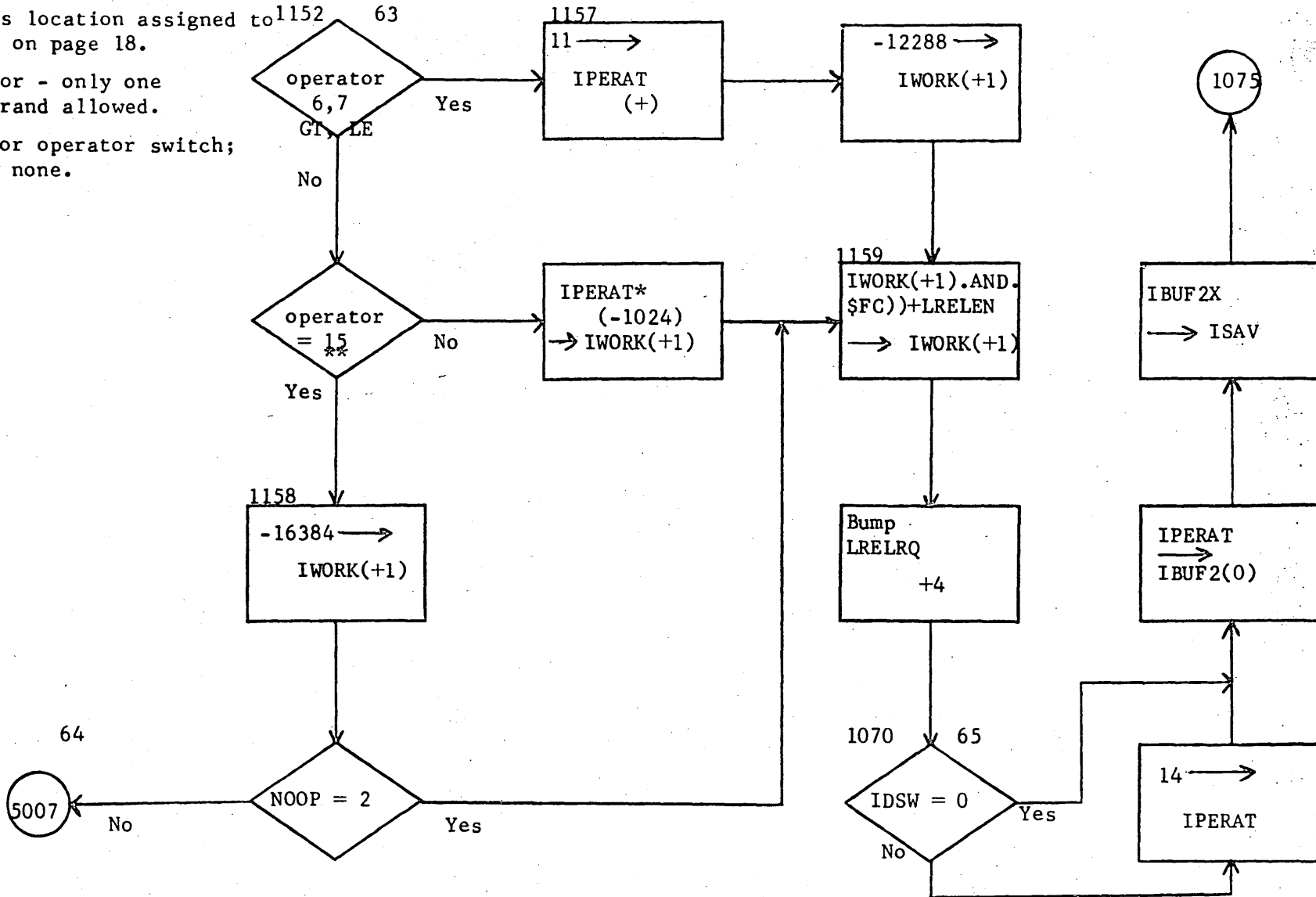
DOCUMENT CLASS <i>ZMS</i>	MACH. TYPE <i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE <i>TREE</i>		PROJECT MGR.			
	PAGE <i>22</i> OF <i>34</i>	PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.			
DRAWN BY	DATE <i>3-67</i>	TASK NAME			

61. Shift right 10 positions.  
 62. Back to beginning.



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	MACH. TYPE	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE		PROJECT MGR.			
		PAGE	PROJECT NAME			
	NUMBER	ISSUE DATE	TASK NO.			
	DRAWN BY	DATE	TASK NAME			

- 63. This location assigned to 1152 63 IAD on page 18.
- 64. Error - only one operand allowed.
- 65. Prior operator switch; 0 = none.



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

DOCUMENT CLASS	<i>JMS</i>	MACH. TYPE	<i>1700</i>
DOCUMENT TITLE	<i>TREE</i>	PAGE	<i>28 of 24</i>
NUMBER		ISSUE DATE	
DRAWN BY		DATE	<i>7-69</i>

PROJECT NO.	
PROJECT MGR.	
PROJECT NAME	
TASK NO.	
TASK NAME	

REV	APPROVED	DATE

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS IMS PAGE NO. 3-82  
PRODUCT NAME L700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. COO5\*3.1 A/B MACHINE SERIES L700

3.4.4 Subroutine MODMXR

This routine is called from TREE only if a mixed mode expression is being processed. It transforms the tree form generated in IBUF2 to indicate calls to function operators as necessary. Mixed mode expressions and their related calls are as follows:

INTEGER - REAL	call	FLOAT
INTEGER - DOUBLE PRECISION	call	DFLT
REAL - DOUBLE PRECISION	call	DBLE

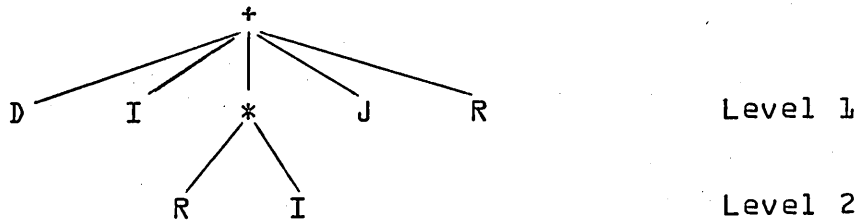
Please refer to section 3.4.3 for a description of the tree form generated by TREE.

One or two lower levels may be inserted in the tree at each mixed mode level. If the mixed mode expression is REAL and there is more than one integer branch on the same level, these branches are grouped so that the integer operation is performed before the call to FLOAT. If the mixed mode expression is DOUBLE PRECISION and there is more than one integer branch or more than one real branch on the same level, these branches are grouped respectively so that the integer operation is performed before the call to DFLT and the real operation is performed before the call to DBLE. Integer divides are not reorderable. The call to a function becomes the operator of a single branch on the previously mixed mode level.

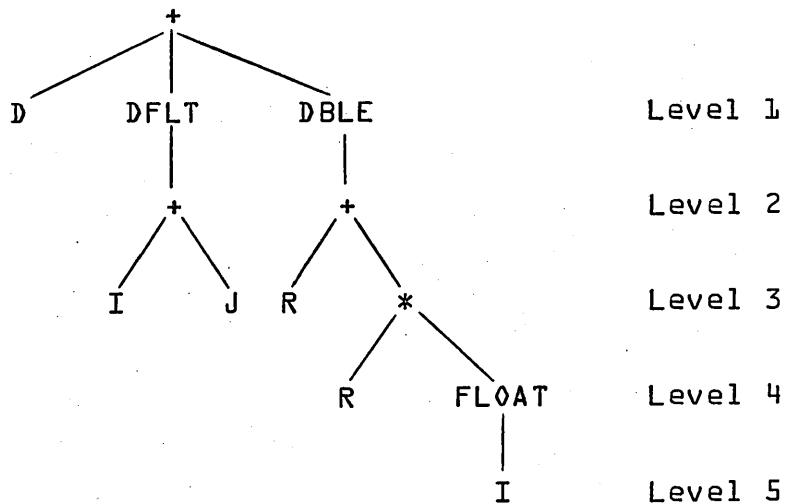
For example: D is a double precision variable  
R is a real variable  
I is an integer variable

DOCUMENT CLASS IMS PAGE NO. 3-83  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

The expression:  $D + I + R * I + J + R$   
 generates the mixed mode tree.

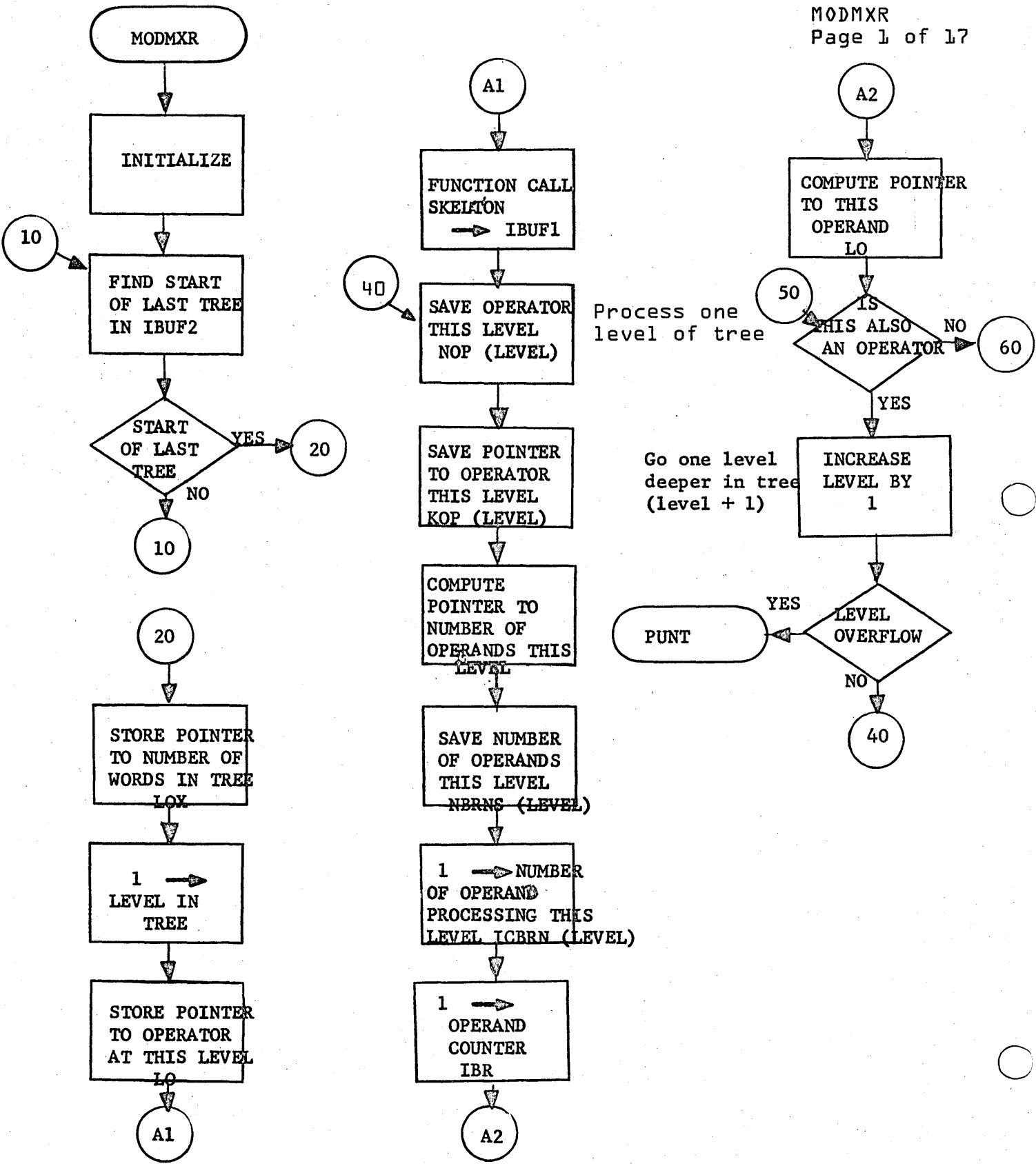


MODMXR transforms this tree into the following form:



DOCUMENT CLASS IMS PAGE NO. 3-84  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. CO05\*3.1 A/B MACHINE SERIES 1700

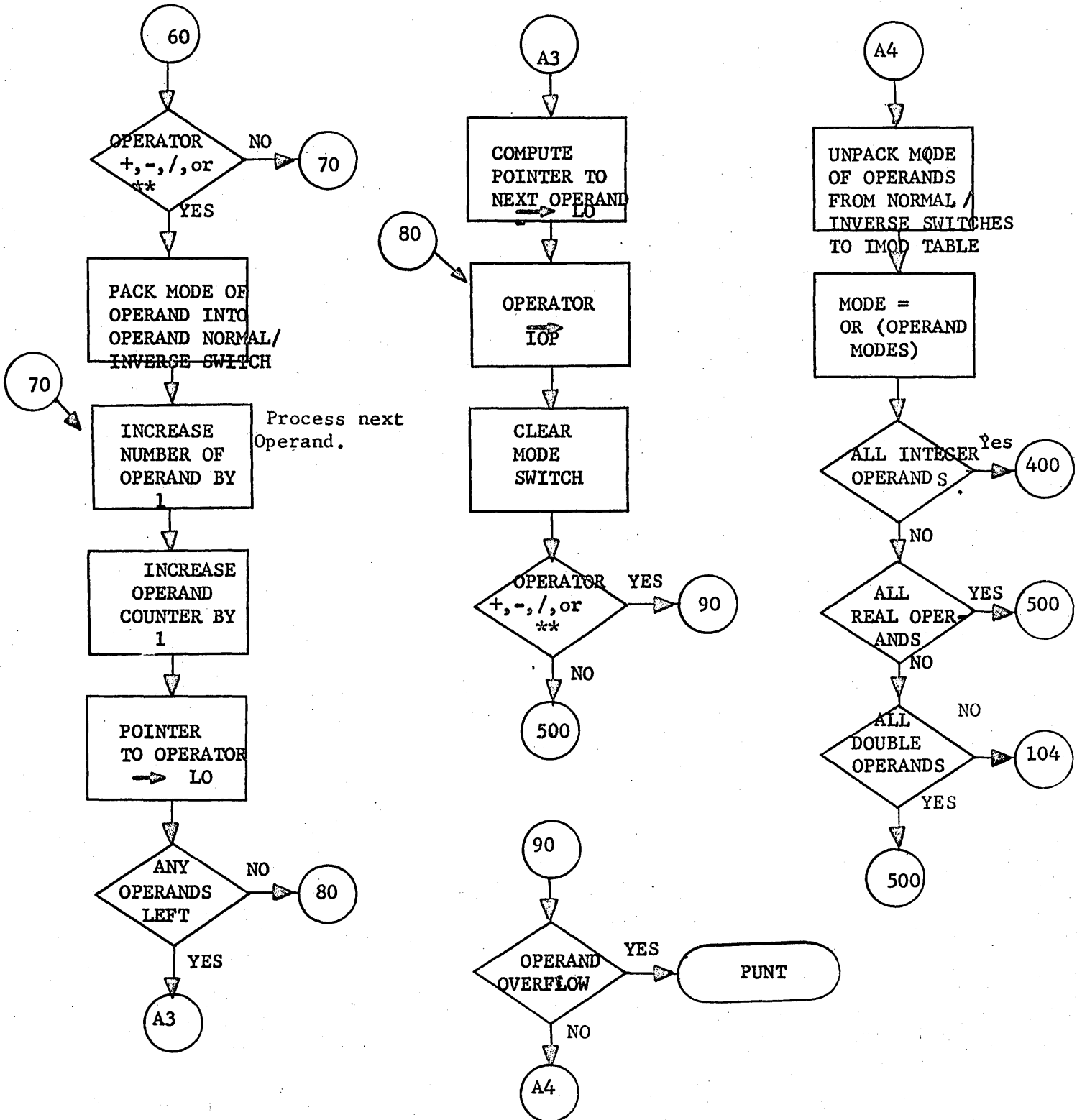
MODMXR  
 Page 1 of 17





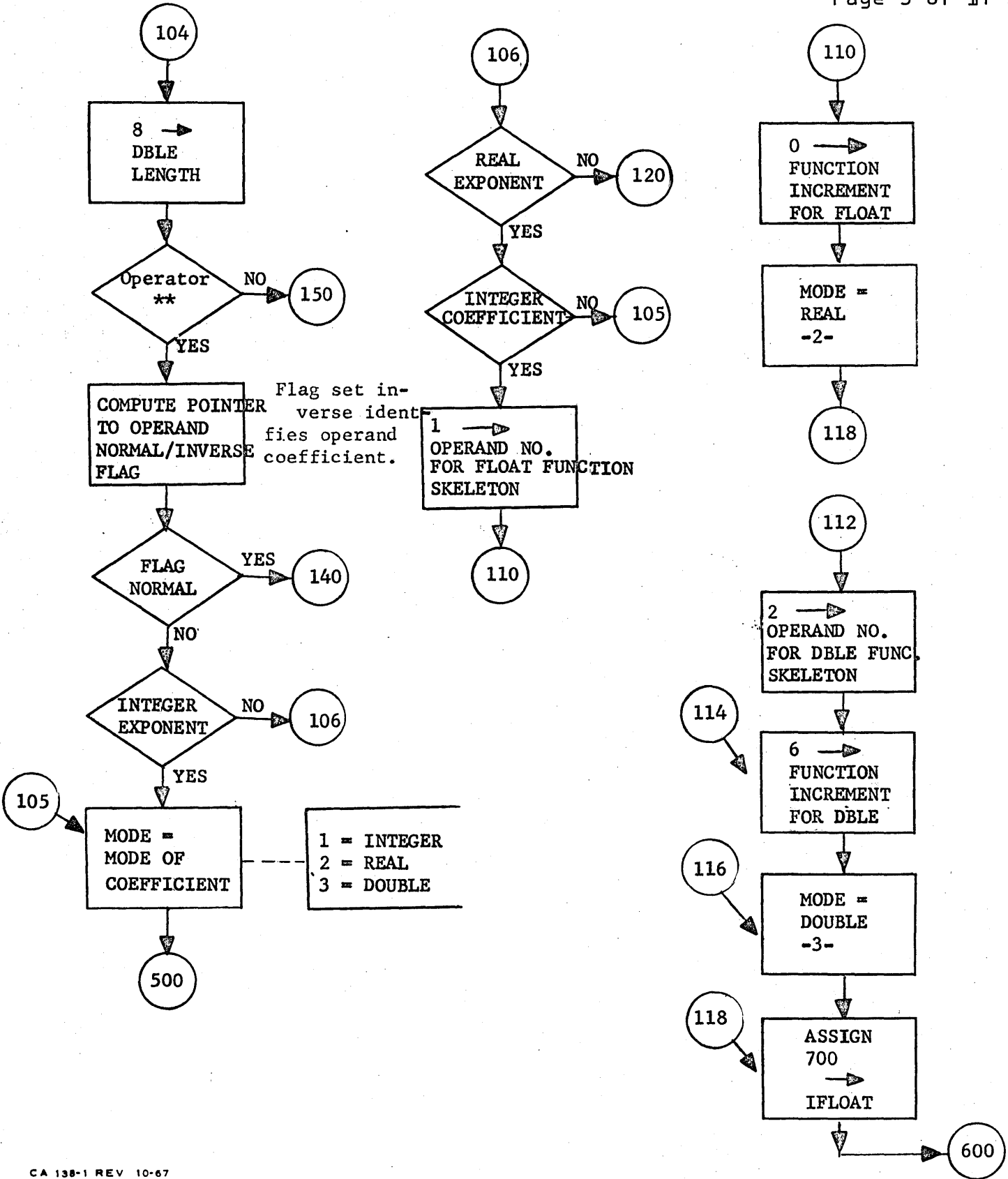
DOCUMENT CLASS IMS PAGE NO. 3-85  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

MODMXR  
 Page 2 of 17



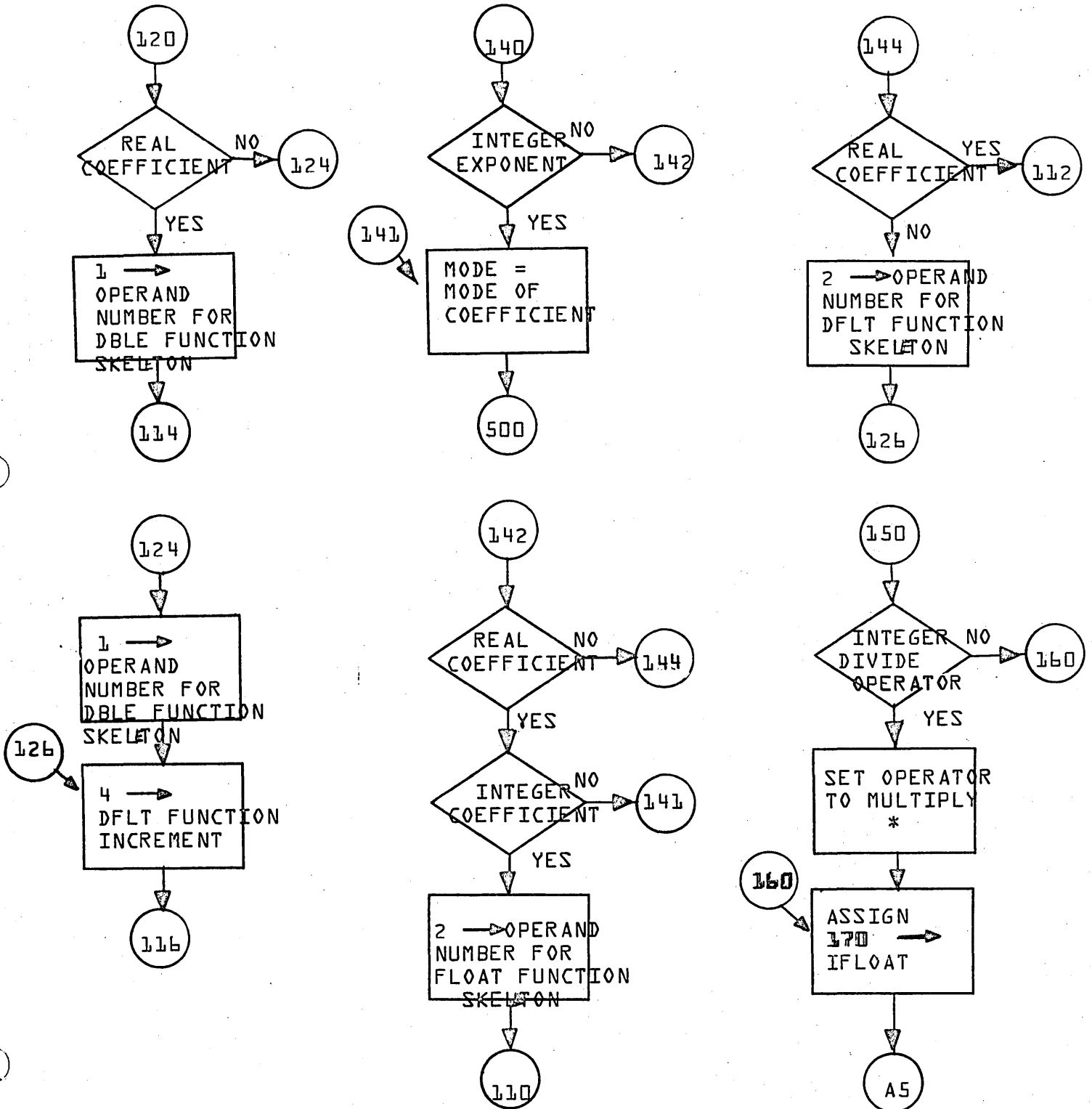
DOCUMENT CLASS IMS PAGE NO. 3-86  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

MODMXR  
 Page 3 of 17



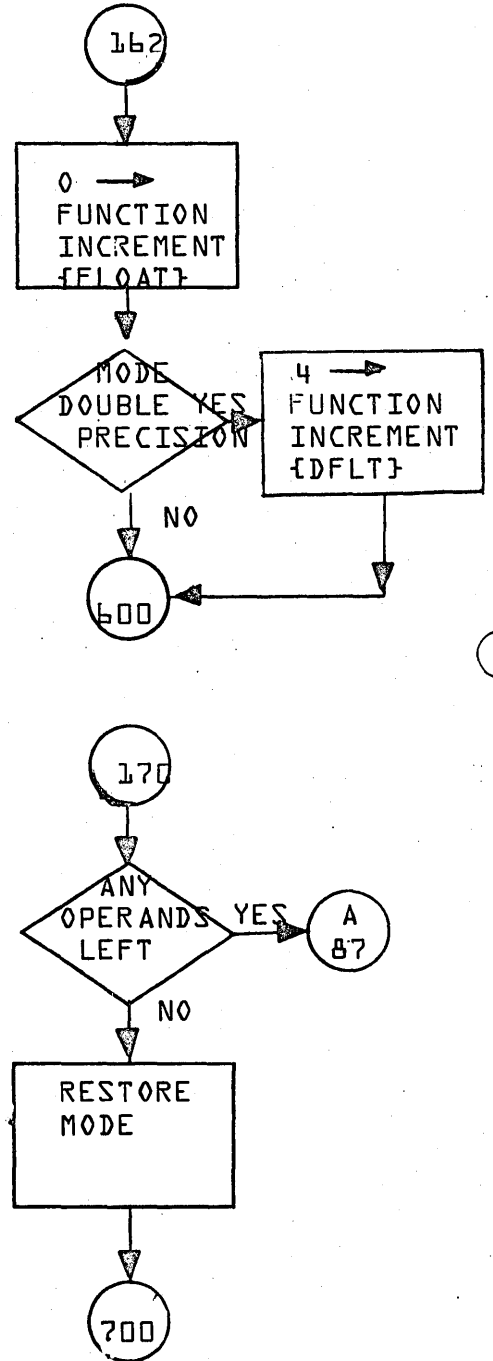
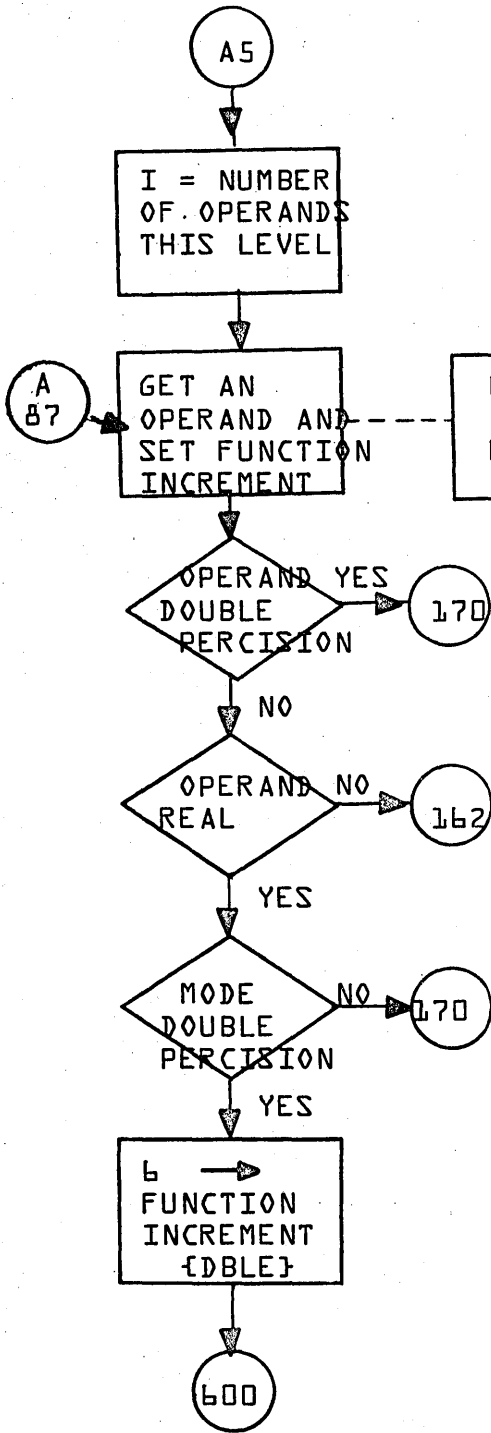
DOCUMENT CLASS IMS PAGE NO. 3-87  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

MODMXR  
Page 4 of 17



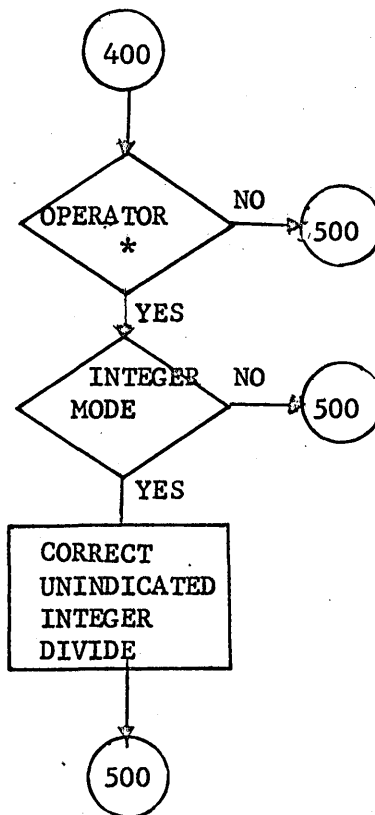
DOCUMENT CLASS IMS PAGE NO. 3-88  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

MODMXR  
 Page 5 of 17



DOCUMENT CLASS IMS PAGE NO. 3-89  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3-1 A/B MACHINE SERIES 1700

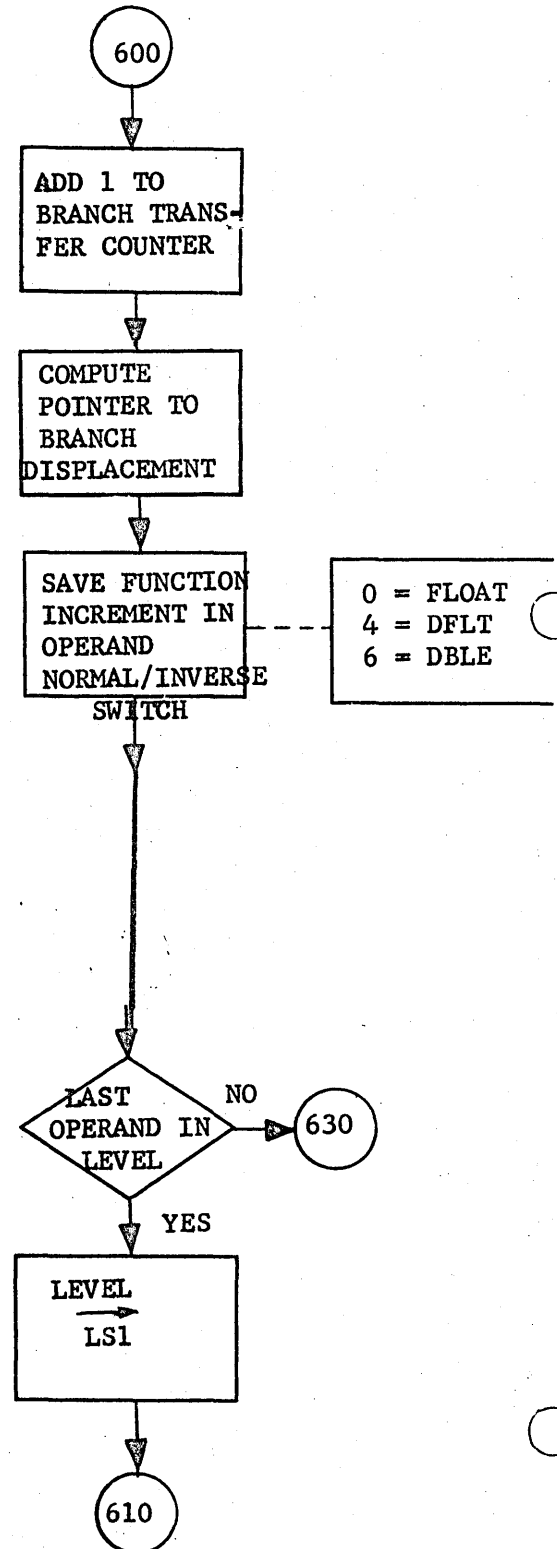
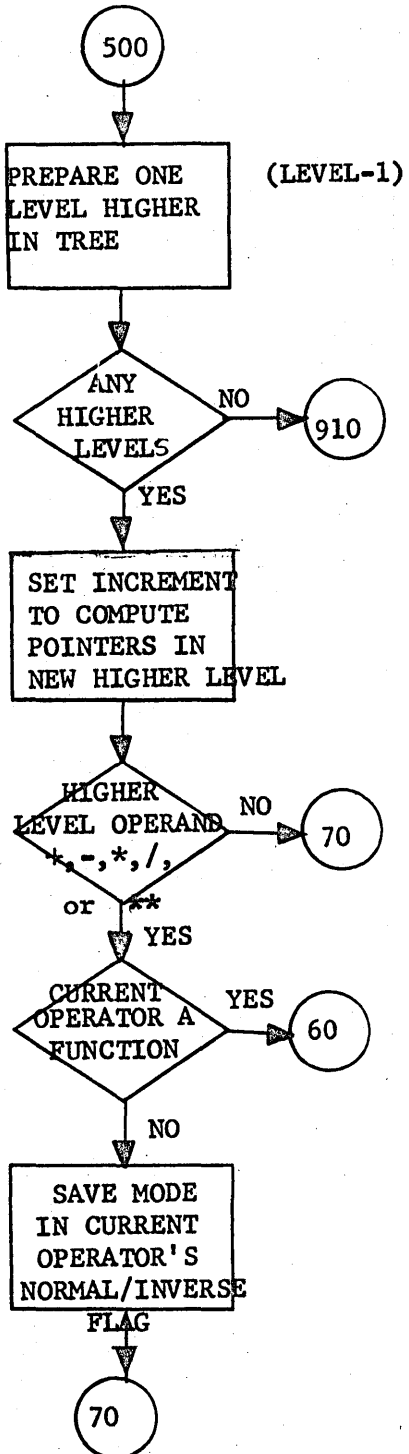
MODMXR  
Page 6 of 17



CONTROL DATA CORPORATION  
 LA JOLLA RESOURCE CENTER DIVISION

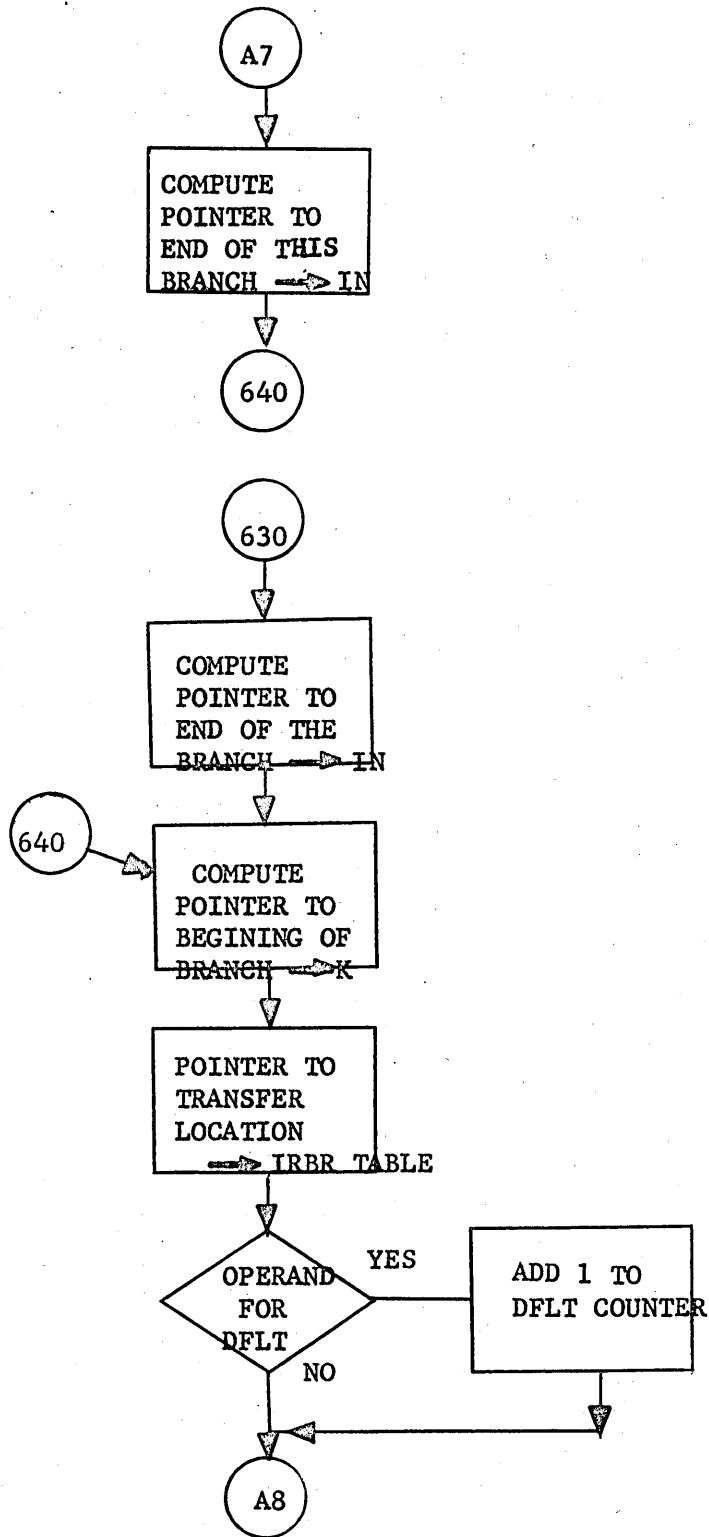
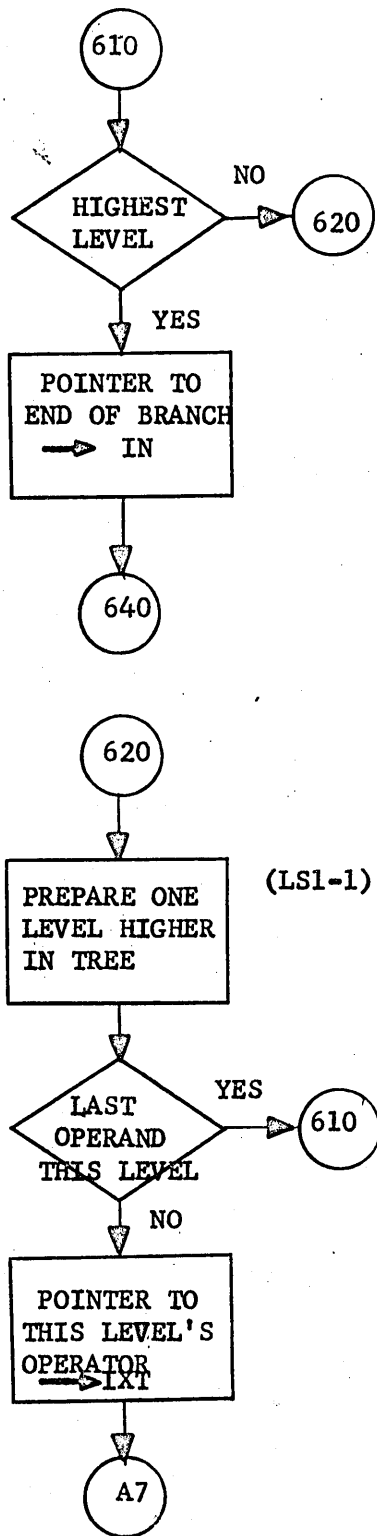
DOCUMENT CLASS IMS PAGE NO. 3-90  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

MODMXR  
 Page 7 of 17



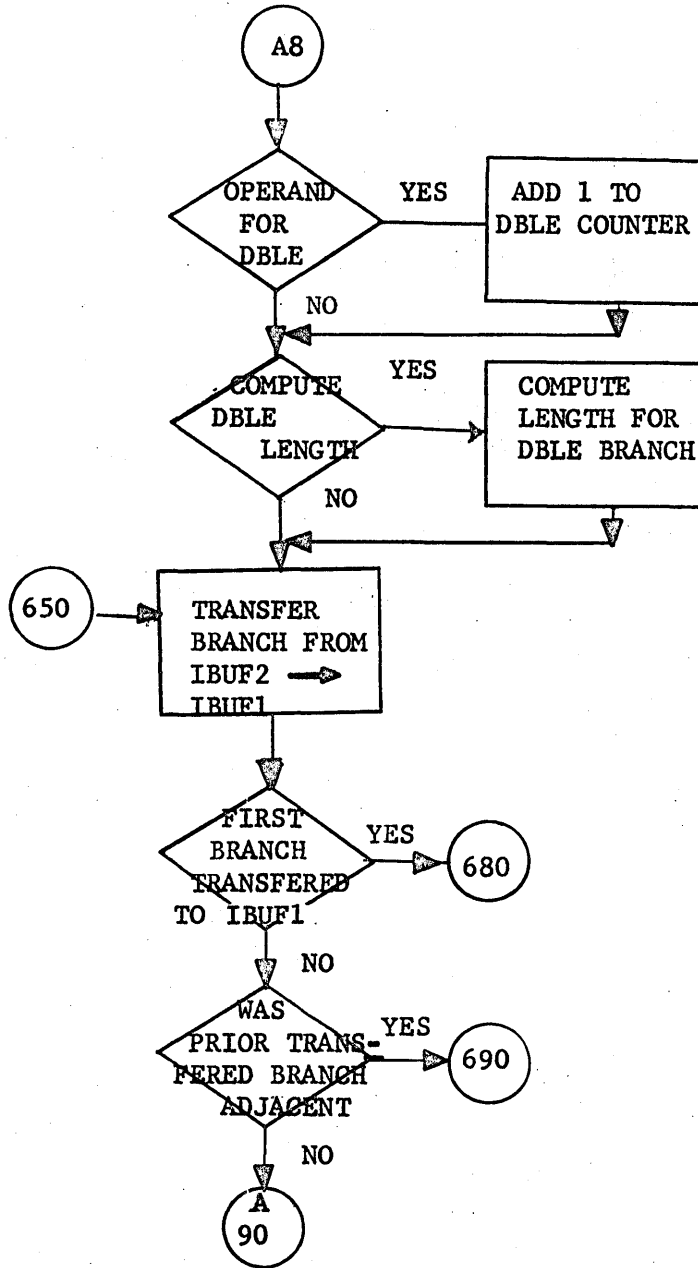
DOCUMENT CLASS IMS PAGE NO. 3-91  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

MODMXR  
 Page 8 of 17



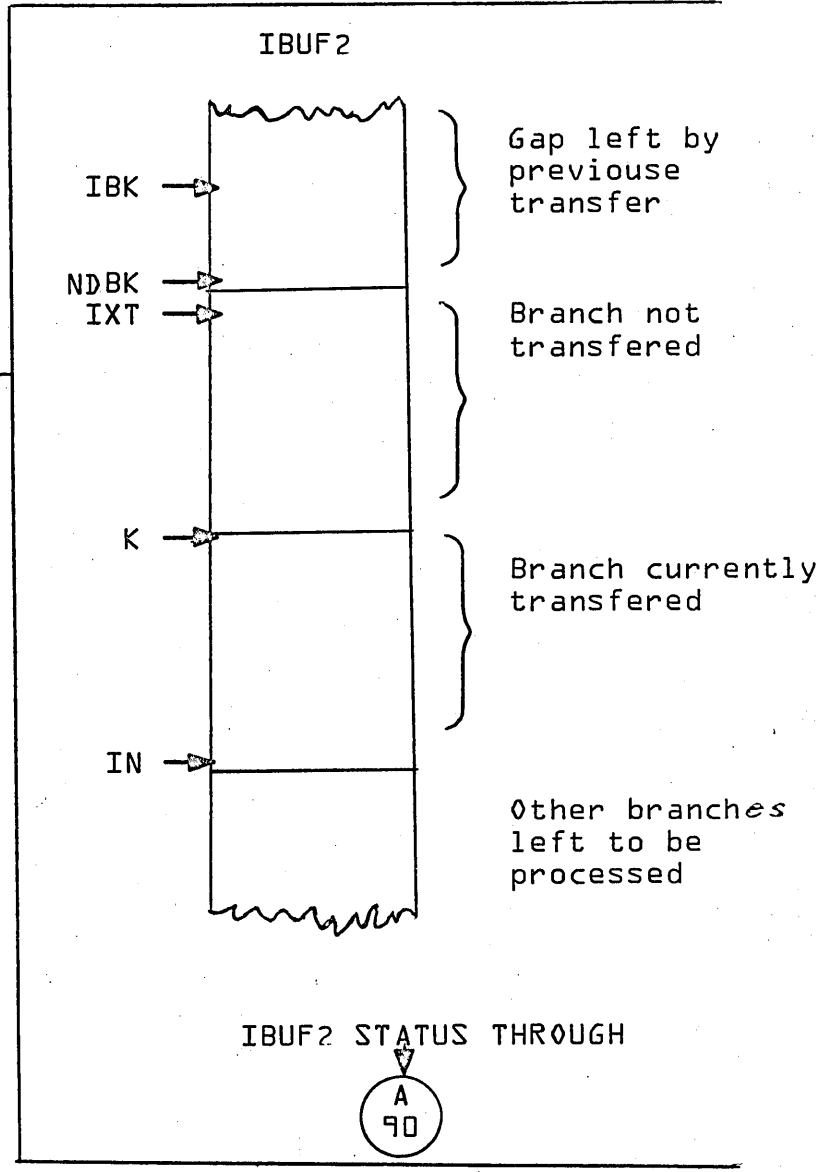
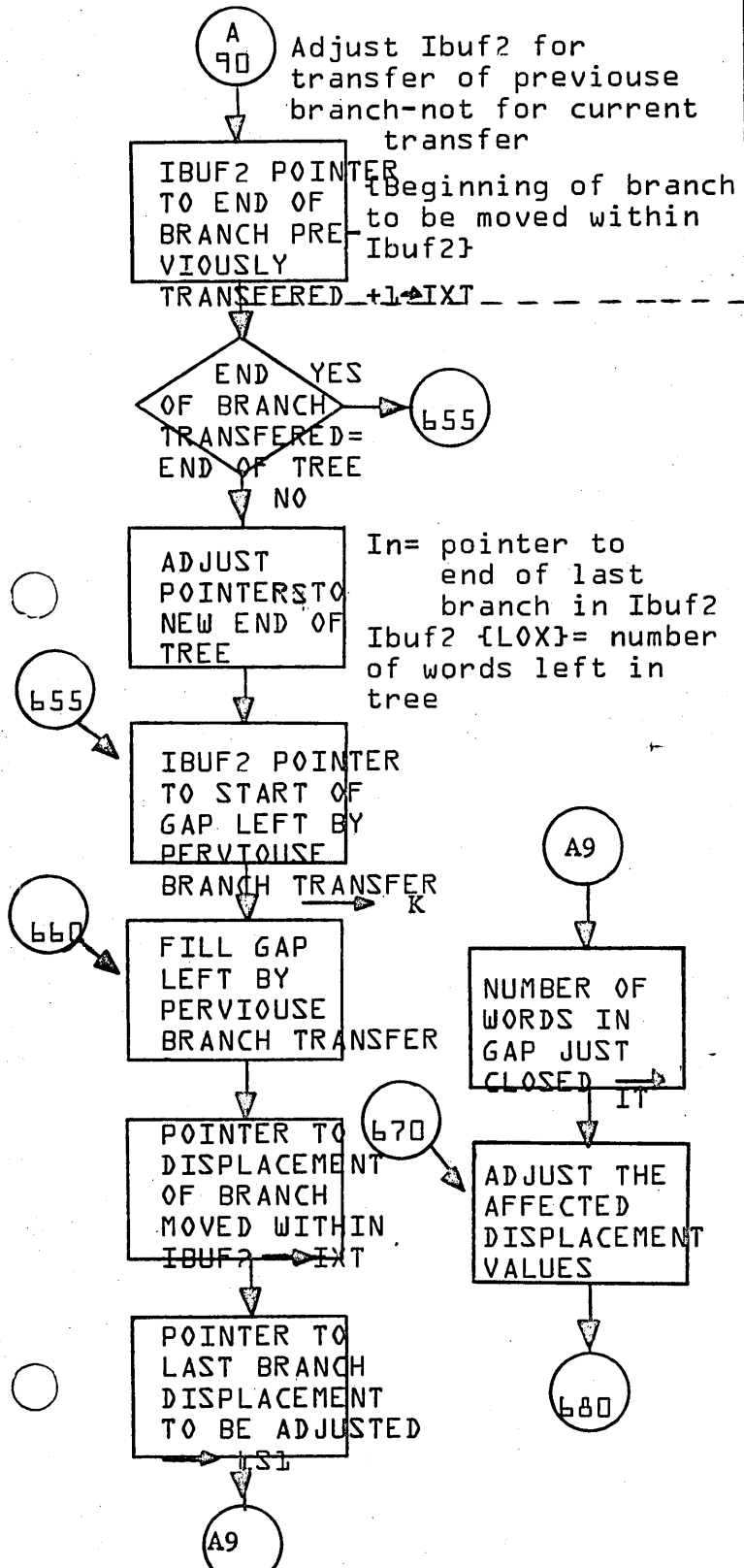
DOCUMENT CLASS IMS PAGE NO. 3-92  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3-1 A/B MACHINE SERIES 1700

MODMXR  
Page 9 of 17



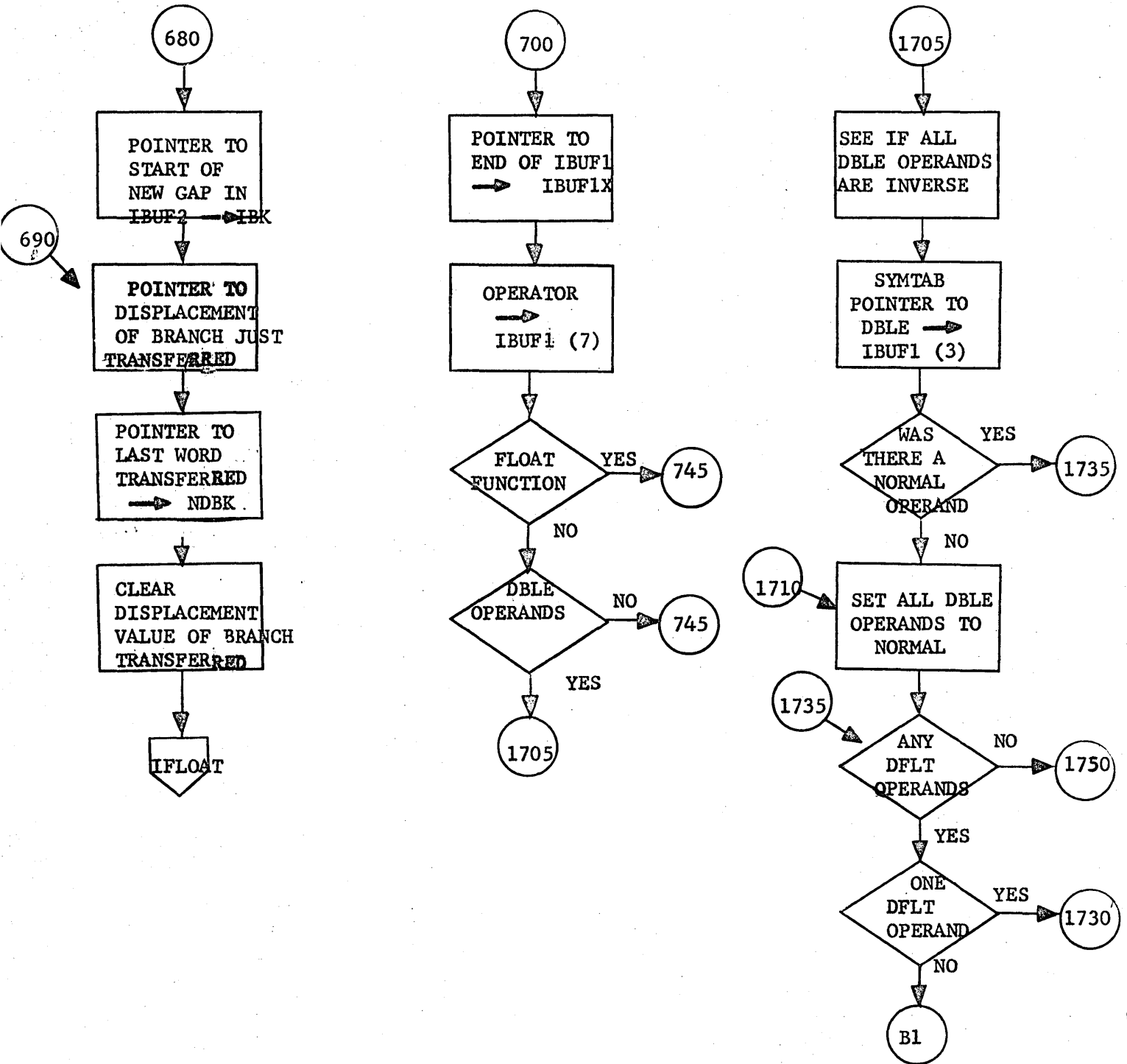


DOCUMENT CLASS IMS PAGE NO. 3-93  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700  
MODMXR  
Page 10 of 17



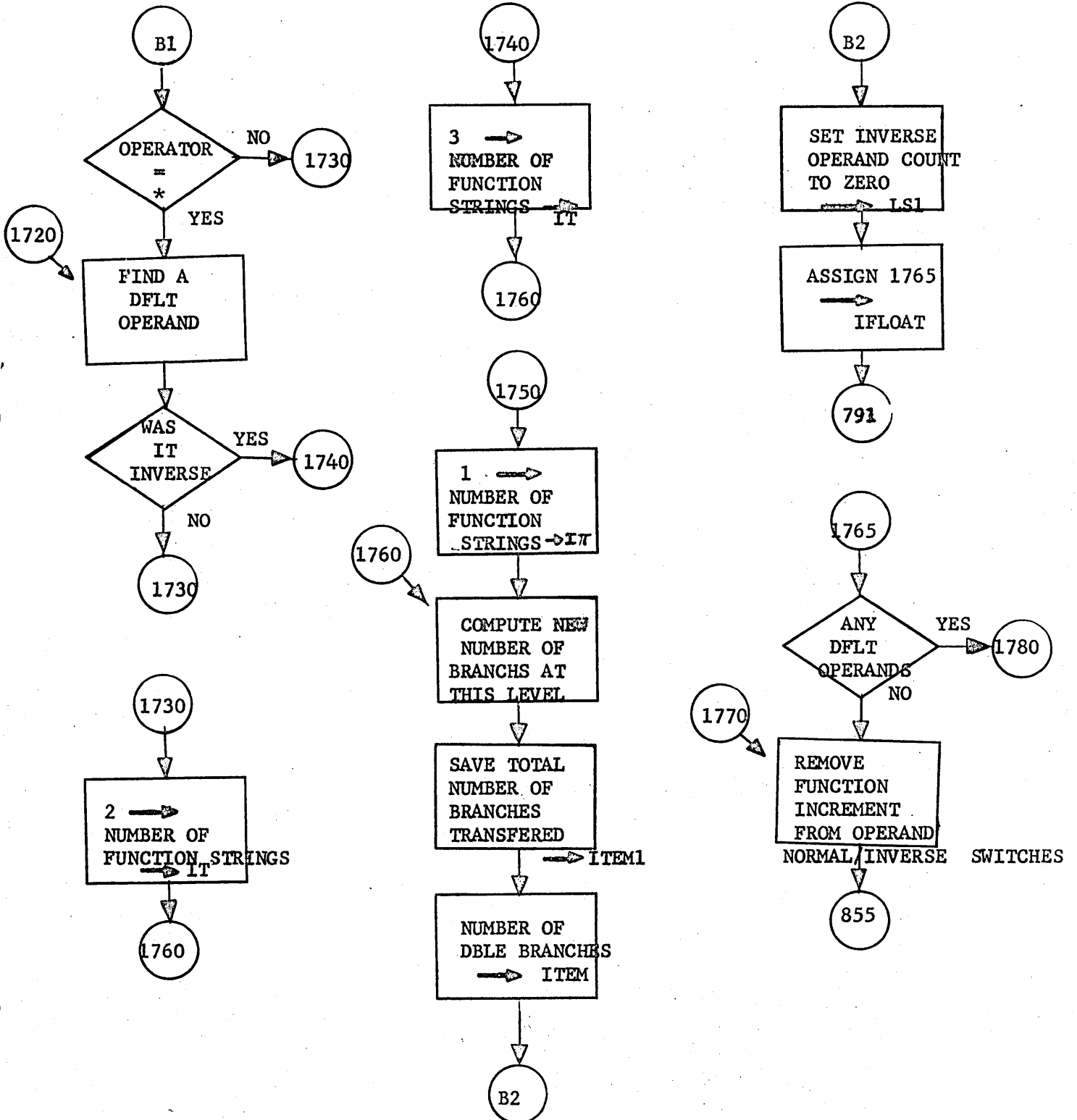
DOCUMENT CLASS IMS PAGE NO. 3-94  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

MODMXR  
 Page 11 of 17



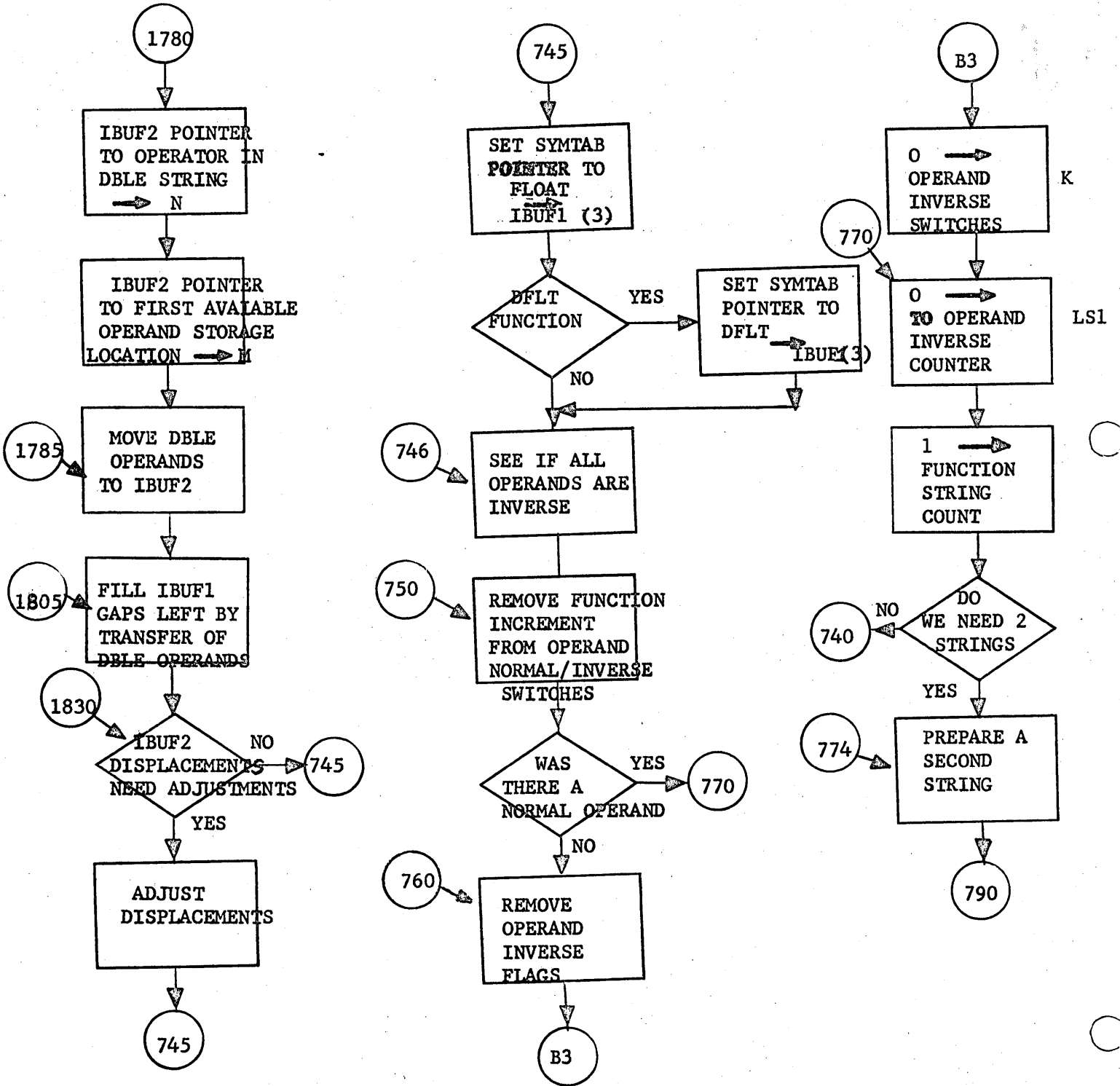
DOCUMENT CLASS IMS PAGE NO. 3-95  
 PRODUCT NAME L700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. COO5\*3-1 A/B MACHINE SERIES L700

MODMXR  
 Page 12 of 17



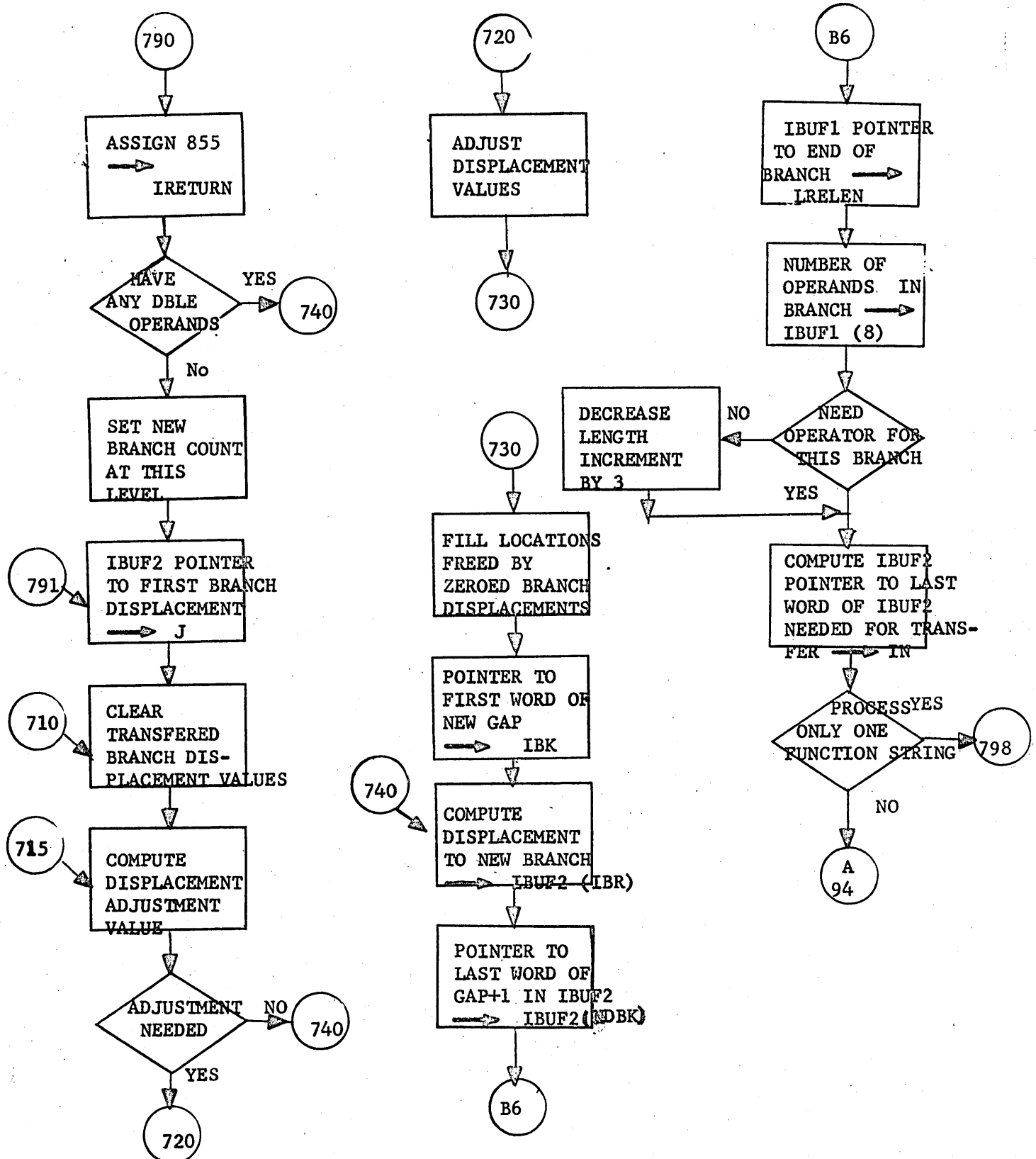
DOCUMENT CLASS IMS PAGE NO. 3-96  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. CO05\*3.1 A/B MACHINE SERIES 1700

MODMXR  
 Page 13 of 17



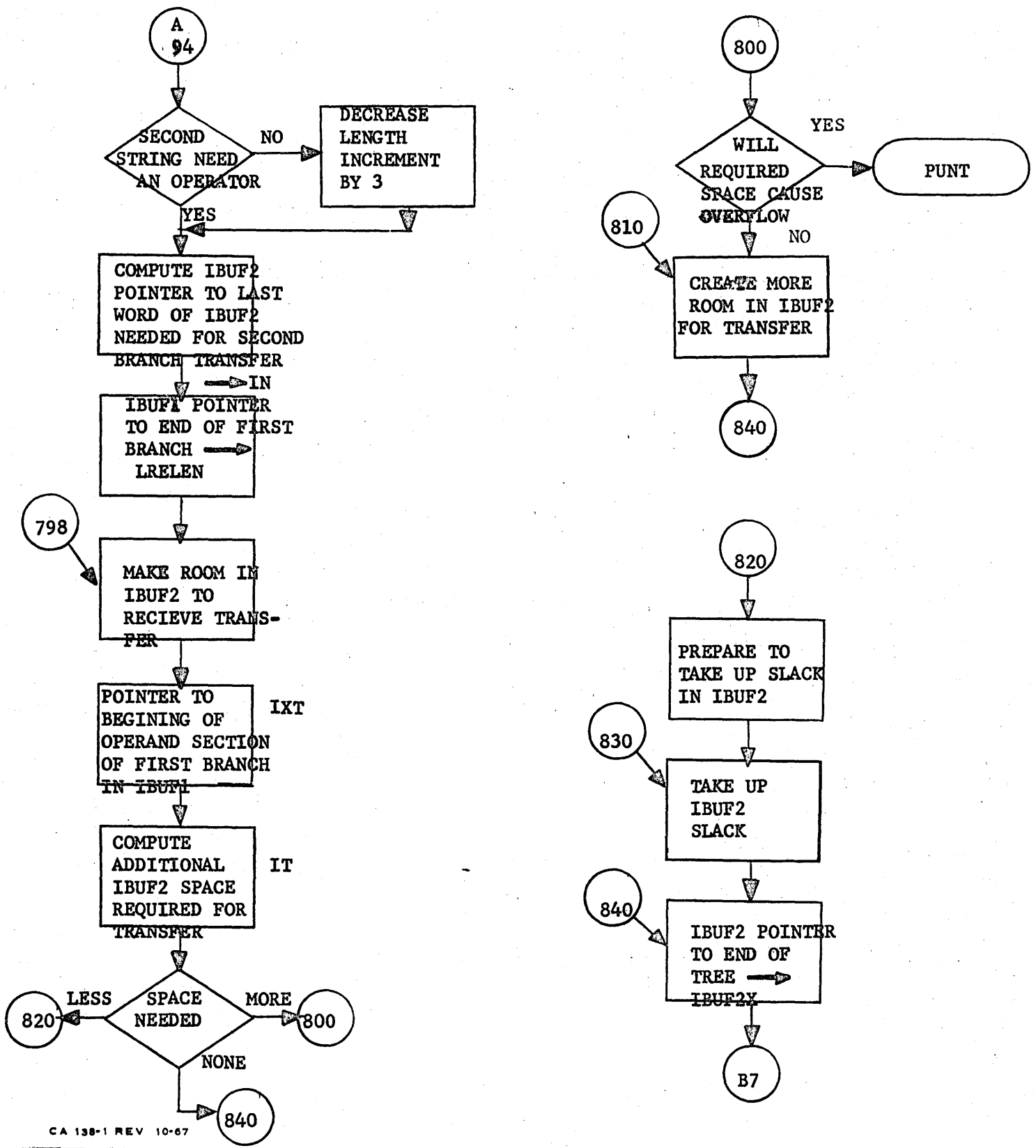
DOCUMENT CLASS IMS PAGE NO. 3-97  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. CD05\*3.1 A/B MACHINE SERIES 1700

MODMXR  
 Page 14 of 17



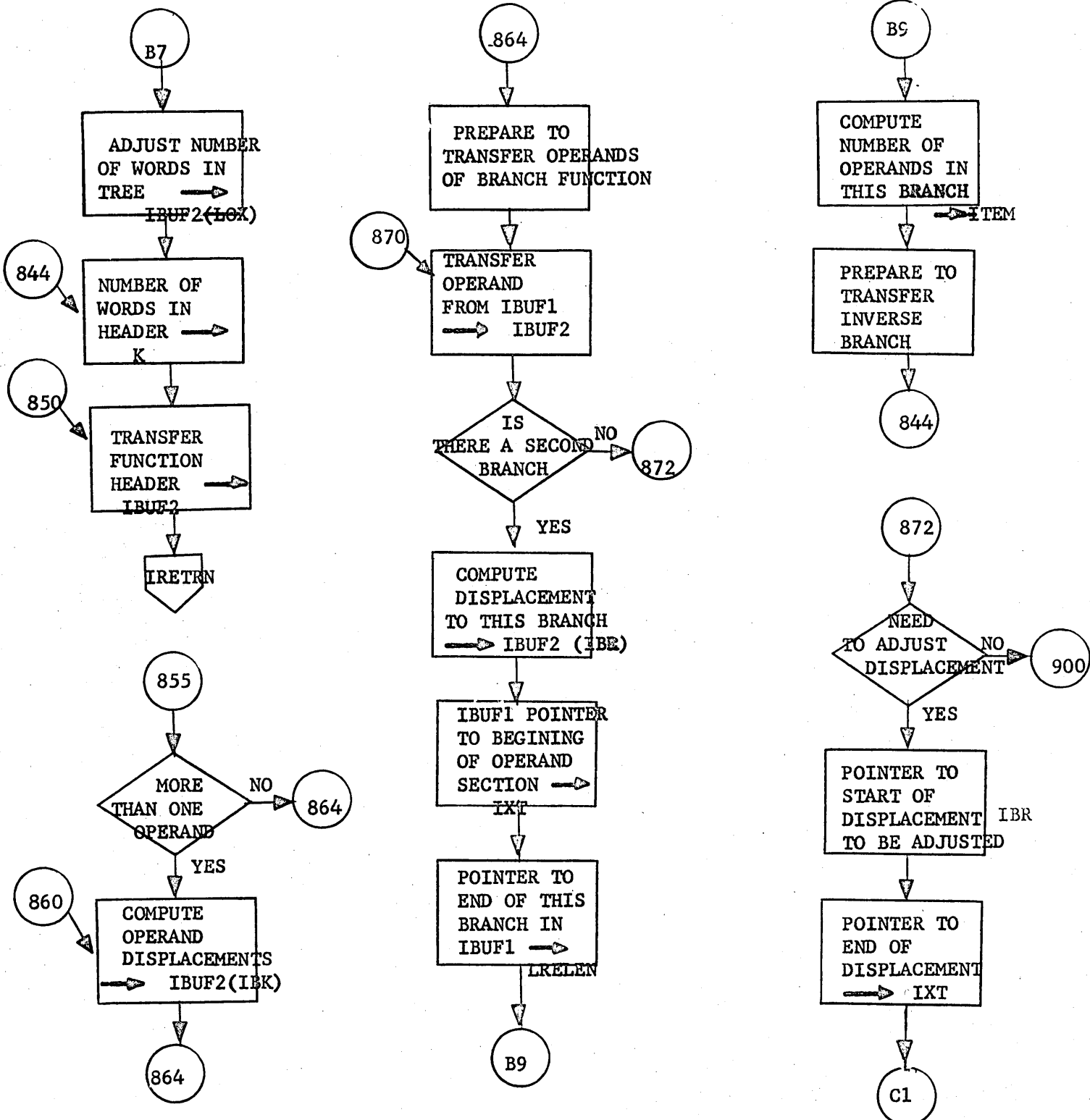
DOCUMENT CLASS IMS PAGE NO. 3-98  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

MODMXR  
 Page 15 of 17



DOCUMENT CLASS IMS PAGE NO. 3-99  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. COO5\*3.1 A/B MACHINE SERIES 1700

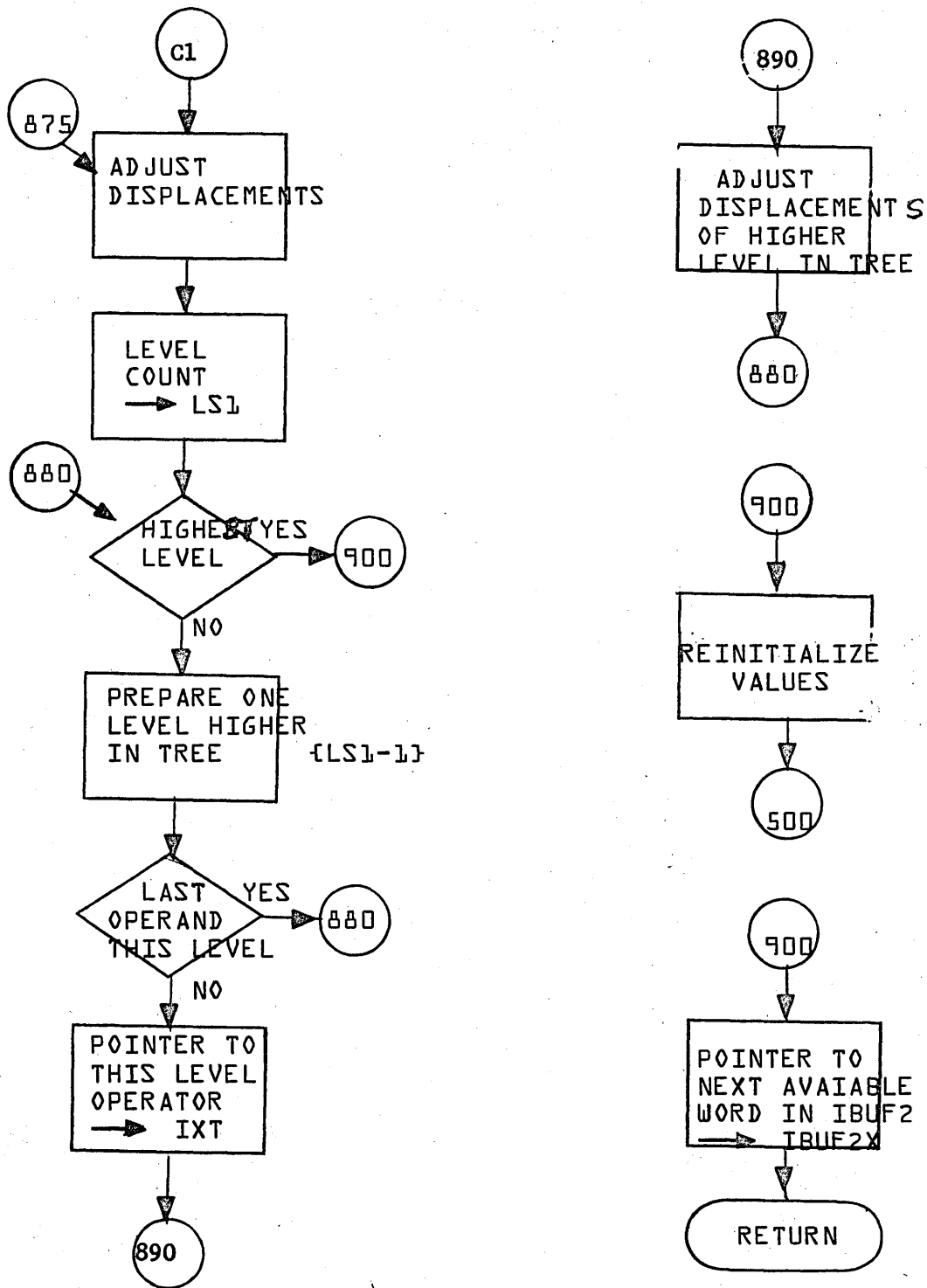
MODMXR  
 Page 16 of 17



CONTROL DATA CORPORATION  
LA JOLLA RESOURCE CENTER DIVISION

DOCUMENT CLASS IMS PAGE NO. 3-100  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

MODMXR  
 Page 17 of 17





DOCUMENT CLASS \_\_\_\_\_ IMS \_\_\_\_\_ PAGE NO. 4-1  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005 V.2.0 \_\_\_\_\_ MACHINE SERIES 1700

#### 4.0 PHASEB

##### 4.1 Phase B General Description

- 4.1.1 Phase B Function Within Compiler
- 4.1.2 Input Description
- 4.1.3 Output Description

##### 4.1.3.1 Instructions

##### 4.2 OPERATION OF THE PHASE B DRIVER (PHASE B)

- 4.2.1 Program Structure
- 4.2.2 Statement Function Processor
- 4.2.3 Arithmetic IF Statement Processor
- 4.2.4 Logical IF Statement Processor

##### 4.3 PHASE B STATEMENT PROCESSORS

- 4.3.1 SUBFUN
- 4.3.2 NOPROC
- 4.3.3 ARITHR
- 4.3.5 CGOTO
- 4.3.6 END
- 4.3.7 BGINDO and BANANA
- 4.3.10 ASSEM

##### 4.4 PHASE B SUBROUTINES

- 4.4.1 AFIDL
- 4.4.2 ASUPER
- 4.4.4 ENTCOD
- 4.4.5 FCMSTK
- 4.4.6 FINK
- 4.4.7 INTRAM
- 4.4.8 INXRST
- 4.4.9 KCPART
- 4.4.10 KOUTPT
- 4.4.11 KPC3PR and LABKPC
- 4.4.12 LABLER
- 4.4.13 ACP
- 4.4.14 PARTSB
- 4.4.15 READIR
- 4.4.16 SUBPR1 and SUBPR2 Subscript Processors
  - 4.4.16.1 SUBPR1
  - 4.4.16.2 SUBPR2
- 4.4.17 SUBPR3

DOCUMENT CLASS IMS PAGE NO. 4-2  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

T A B L E O F C O N T E N T S

4.4.18 TSALOC  
4.4.19 HELEN  
4.4.20 DUMMY  
4.4.20.A KPCSTK  
4.4.21 KSYMGN

DOCUMENT CLASS IMS PAGE NO. 4-3  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

#### 4.0 PHASE B

#### 4.1 GENERAL DESCRIPTION

##### 4.1.1 Phase B Function Within Compiler

Phase *B* is the instruction generation pass of the 1700 Tape FORTRAN compiler, or, what is termed the pre-assembly phase. Its responsibility is translating FORTRAN statements into instructions. Other peripheral tasks relevant to the pre-assembly and assembly phase (Phase *C*) are carried out by Phase *B*.

##### 4.1.2 Input Description

Input to Phase *B* is of two types:

1. Pertinent information residing in that portion of the compiler's common blocks which remains intact from one pass to the next pass. (LABELLED COMMON) The information contained therein is discussed in detail in 4.5.
2. An input file on *disk* corresponding to the output file of Phase *A*. It consists of a consecutive string of logical records the last of which represents the END line. The logical records in the Phase *B* input file correspond one to one, and in the same physical order as the 1700 Tape FORTRAN source statements, with the following set of exceptions.
  - a. Only executable statements reach the Phase *B* input file with the following additions and exceptions: Statements which are not classified as executable but which are passed:

- (1) SUBROUTINE
- (2) FUNCTION
- (3) DATA
- (4) FORMAT
- (5) END
- (6) Statement Function

Statements which are classified as executable but which are not passed comprise the set of Input/Output and Auxiliary Input/Output statements:

- (1) READ (formatted)
- (2) WRITE (formatted)
- (3) READ (unformatted)
- (4) WRITE (unformatted), and ...
- (5) REWIND

DOCUMENT CLASS IMS PAGE NO 4-4  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT NO. 005 VERSION 2.0 MACHINE SERIES 1700

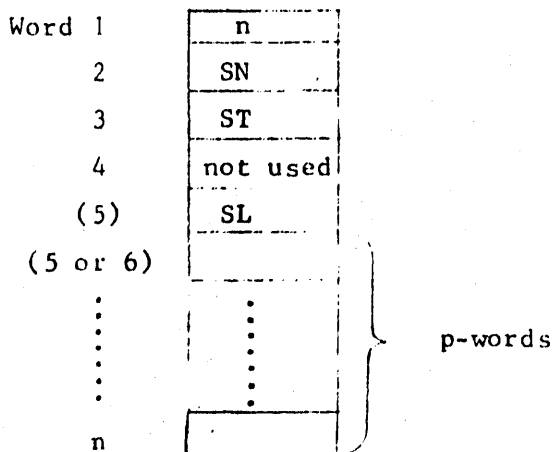
- (6) BACKSPACE
- (7) ENDFILE

b. Certain conditions in compilation cause the creation of additional executable statements in conjunction with those which appeared in the original source input.

1. Input/Output and Auxiliary Input/Output statements as noted in paragraph a, are replaced in the same relative position within the source input, with an expanded set of statements consisting of one or more statements of type CALL, BEGIN DO, END DO. (See discussion of END DO below.)
2. When two conditions are satisfied, a STOP statement is created and inserted immediately prior to the END line. First, the program being compiled is a main program, and second, the last executable statement of the program is neither a STOP, STOP n, or any other unconditional transfer statement. (Unconditional transfer statements which comprise the latter (truth) portion of a logical IF do not qualify.)
3. For each DO statement in the source input the Phase  $\beta$  input file contains a BEGIN DO statement record. But additionally, a special statement, END DO, internal to the compiler, is generated uniquely for each BEGIN DO and located logically in the input file so as to aid Phase  $\beta$  in DO-loop instruction generation.

c. The Logical IF statement appears in the Phase  $\beta$  input file as two separate statements.

The figure below illustrates the format of a record of the Phase  $\beta$  input file.



DOCUMENT CLASS IMS PAGE NO. 4-5  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

Where:

$n$  = the number of words in the record ( $n \geq 4$ )  
SN = s.n. or -s.n., where s.n. is the statement number  
ST = the statement type  
SL = the symbol table pointer of the statement label  $p \geq 0$

The statement label is optional and its existence is indicated by SN being negative. When SN is negative, the remaining  $p$  words of the record begin at word 6 and end at word  $n$ . When SN is positive, the remaining  $p$  words of the record begin at word 5 and end at word  $n$ .

The interpretation of the last  $p$  words of any record by the Phase  $B$  processors is primarily determined by ST, the statement type. Each statement type designates what information, if any, follows in the remaining  $p$  words.

Table 4-1 lists all statements processed in the 1700 Tape FORTRAN compiler. Each statement is cross-referenced with its statement type code and its disposition by Phase  $B$ . Refer to the following key for further explanation:

(n.u.) - this ST code is not used in 1700 Tape FORTRAN

(n.r.) - this statement type is not received by Phase  $B$  in its input file.

"processed by XXXXXX" - Phase  $B$  driver calls upon subprogram (named XXXXXX) which is designed to perform the appropriate processing (usually generation of instructions) for that particular type of statement.

"processed by Phase  $B$  driver" - indicates that the appropriate processing for that particular type of statement is performed within the Phase driver and no subprogram is called which is specifically designed to process that statement type.

DOCUMENT CLASS IMS PAGE NO. 4-6  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT NO. 0005 VERSION 2.0 MACHINE SERIES 1700

TABLE 4 - 1

## Statement Types and Phase B Disposition

<u>Type Code</u>	<u>Statement Name</u>	<u>Disposition by Phase B</u>
0	DIMENSION	(n.r.)
1	COMMON	(n.r.)
2	INTEGER	(n.r.)
3	REAL	(n.r.)
4	(n.u.)	
5	(n.u.)	
6	(n.u.)	
7	SINGLE	(n.r.)
8	BYTE, SIGNED BYTE	(n.r.)
9	(n.u.)	
10	EXTERNAL	(n.r.)
11	RELATIVE	(n.r.)
12	EQUIVALENCE	(n.r.)
13	BLOCK DATA	(n.r.)
14	SUBROUTINE	processed by SUBFUN
15	FUNCTION	processed by SUBFUN
16	DATA	processed by NOPROC
17	FORMAT	processed by NOPROC
18	Replacement Statement	processed by ARITHR
19	Statement Function	processed by <i>PHASE B DRIVER</i>
20	ASSIGN	processed by Phase B driver
21	CALL	processed by Phase B driver
22	RETURN	processed by Phase B driver
23	(n.u.)	
24	GO TO (unconditional)	processed by Phase B driver
25	GO TO (computed)	processed by CGOTO
26	GO TO (assigned)	processed by Phase B driver
27	CONTINUE	processed by Phase B driver
28	STOP	processed by Phase B driver
29	STOP n	processed by Phase B driver

DOCUMENT CLASS IMS PAGE NO. 4-7  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

<u>Type Code</u>	<u>Statement Name</u>	<u>Disposition by Phase B</u>
30	PAUSE	processed by Phase <i>B</i> driver
31	PAUSE n	processed by Phase <i>B</i> driver
32	END	processed by END
33	ENDFILE	(n.r.)
34	REWIND	(n.r.)
35	BACKSPACE	(n.r.)
36	READ (unformatted)	(n.r.)
37	READ (formatted)	(n.r.)
38	WRITE (unformatted)	(n.r.)
39	WRITE (formatted)	(n.r.)
40	BEGIN DO	processed by BGINDO
41	END DO	processed by BANANA
42	Arithmetic IF	processed by <i>PHASE B DRIVER</i>
43	Logical IF	processed by <i>PHASE B DRIVER</i>
44	ASSEM	processed by ASSEM

DOCUMENT CLASS IMS PAGE NO. 4-8  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

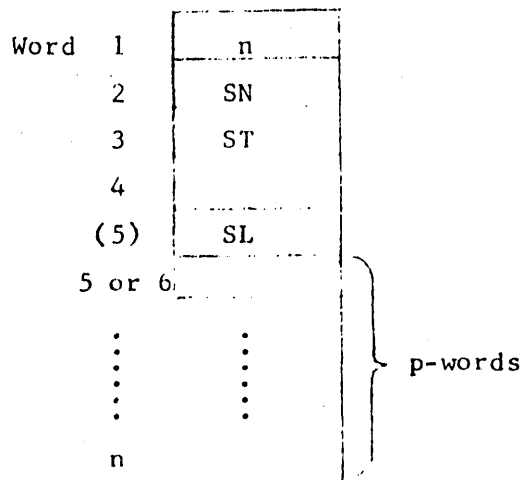
Phase *B* specifications will not give a general summary of the detailed formats of the input records for each statement type. Refer to 7.5 for a detailed description. References will be made to such formats as may be appropriate in clarifying explanations.

4.1.3 Output Description

Output from Phase *B* is of two types

1. Pertinent information (either retained from previous passes or set by Phase *B*) residing in that portion of the compiler's common blocks which remains intact from Phase *B* to Phase *C*. (The information contained therein is discussed in detail in 4.5.) Use and definition of the information is discussed throughout Phase *B* description.
2. An output file on disk corresponding to what will become the Phase *C* input file. It consists of a consecutive string of logical records, the last of which is indicative of the end of the file.

The figure below illustrates the general format of a record of the Phase *B* output file:





DOCUMENT CLASS IMS PAGE NO. 4-9  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

Three cases of the general format occur:

Case 1:

Where:

$n =$  the number of words in the record ( $6 \leq n \leq 9$ )

SN = (not used)

ST = 49

SL = (does not exist)

the next  $p$  words begin with word 5

$2 \leq p \leq 5$

Case 1 is the instruction record. The  $p$  words describe a single instruction generated by Phase  $B$ .

Case 2:

Where:

All definitions are the same for the Phase  $B$  input records (see 4.1.2 P 2c). Case 2 occurs for the case in which the processing of an input record by Phase  $B$  primarily is the transfer of that record to the output file exactly as it was read without modification. Records which fall into this category are DATA and FORMAT (ST = 16 or ST = 17)

Case 3:

Where:

$n = 1$

Case 3 is a flag to Phase  $C$  to indicate that the next record is a case 2 type record.

#### 4.1.3.1 Instructions

The 2 to 5 words in the case 1 record format describe a single instruction generated by Phase  $B$ . This format is as follows:



3. S specifies whether or not the operand address is subscripted.

S = 0 not subscripted

S = 1 subscripted

4. D tells whether instruction is to be made indirect or not.

D = 0 not indirect

D = indirect

5. R Decision whether instruction is to be made 1 or 2 words.

R = 0 decision is left to next phase which will make any such command 1 word whenever possible, dependent upon the size of constant operands and distances of relative addresses, etc.

R = 1 force instruction two words, regardless of operand size. This is usually done where operand has to be plugged.

6. Q&F These two bits are set aside for the next phase as working space in its index register assignment, one for index Q and one for FF. In special instances, Phase 4 may set Q and generate its own instructions to load Q.

7.  $\emptyset$  operand type

A. For instruction code  $\neq 2$

$\emptyset = 0$  normal case, operand is a symbol table pointer. Addressing mode is assigned by assembly pass (Phase 5)

$\emptyset = 1$  Operand is an absolute value. For the instruction code being a storage reference instruction, the operand is less than  $256_{10}$

DOCUMENT CLASS IMS PAGE NO 4-12  
PRODUCT NAME 170 Mass Storage FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

$\emptyset = 2$  Operand is an absolute value. For the instruction code being a storage reference instruction, Phase 5 assigns two-word constant address mode.

$\emptyset = 3$  Not used

$\emptyset = 4$  Operand is relocatable address constant (biased)

$\emptyset = 5$  Operand is absolute address constant (unbiased)

B. For instruction code = 2

$\emptyset = 0$  not used

$\emptyset = 1$  This address constant is in a table of addresses used in a Computed GO TO statement. The address is self relative and is calculated using 16-bit arithmetic.

$\emptyset = 2$  This address constant is a parameter in a calling sequence to the floating point subroutine.

$\emptyset = 3, 4, 5$  refers to parameters in a calling sequence to a subroutine other than the floating point subroutine.

$\emptyset = 3$  Address self relative computed in 15-bit arithmetic with 15 bit set to 1.

$\emptyset = 4$  Relocatable, address. Address is location of referenced operand.

$\emptyset = 5$  Same as type 4 except address is unbiased.

Word 2 is always the instruction code except when the instruction is a label (bit T=1). Instruction codes are shown in table below:

CONTROL DATA CORPORATION • DEVELOPMENT DIV • SOFTWARE DOCUMENT

DOCUMENT CLASS IMS PAGE NO 4-13  
 PRODUCT NAME 1700 Mass Storage FORTRAN  
 PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

OCTAL CODE	DECIMAL CODE	INSTRUCTION	OCTAL CODE	DECIMAL CODE	INSTRUCTION
1	1	BSS	41	33	INQ
2	2	ADC	42	34	LRS
3	3	CON	43	35	LLS
4	4	END	44	36	QRS
5	5	PST	45	37	QLS
6	6	STN	46	38	ARS
7	7		47	39	ALS
10	8		50	40	AJLGZ
11	9		51	41	AJEZ
12	10	JMP	52	42	AJLZ
13	11	RTJ	53	43	AJGEZ
14	12	LDA	54	44	AJLEZ
15	13	AND	55	45	AJGZ
16	14	STA	56	46	
17	15	STQ	57	47	QJEZ
20	16	RAO	60	48	
21	17	ADD	61	49	QJGEZ
22	18	SUB	62	50	
23	19	LDQ	63	51	
24	20	DVI	64	52	FCM
25	21	ADQ	65	53	FSB
26	22	MUI	66	54	FMU

CONTROL DATA CORPORATION  
LA JOLLA RESOURCE CENTER DIVISION

DOCUMENT CLASS IMS PAGE NO. 4-14  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

OCTAL CODE	DECIMAL CODE	INSTRUCTION	OCTAL CODE	DECIMAL CODE	INSTRUCTION
27	23	EOR	67	55	FDV
30	24		70	56	FLD
31	25		71	57	
32	26		72	58	FST
33	27		73	59	FAD
34	28	KLDQ	74	60	CAQA
35	29	KSTQ	75	61	TCQA
36	30	ENA	76	62	TRAQ
37	31	INA	77	63	TCAA
30	32	ENQ	100	64	TCQQ
			110	72	DCM
			111	73	DSB
			112	74	DMN
			113	75	DDV
			114	76	DLD
			115	77	
			116	78	DST
			117	79	DAD

CONTROL DATA CORPORATION  
LA JOLLA RESOURCE CENTER DIVISION

DOCUMENT CLASS IMS PAGE NO. 4-15  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

Each instruction corresponds uniquely to one actual 1700 instruction with the noted exceptions:

1. END, PST, STN are discussed where appropriate in other sections.
2. FCM, FSB, FMU, FDV, FLD, FST, FAD and DCM, DSB, DMU, DDV, DLD, DST, DAD are single precision and double precision floating point instructions used only within Phase B. Phase C doesn't encounter floating commands as such, but rather the floating calling sequences:

```
RTJ  FL0T
CON  ...
ADC
ADC
:
etc.
```

or

```
RTJ  DFLOT
CON  ...
ADC
ADC
:
etc.
```

3. "Place holding"--pseudo instructions BSS, ADC and CON.
4. Conditional Jumps

Phase B has occasion to generate code to test the accumulator or Q-register for 6 possible conditions:

```
Acc or Q < 0 {less than zero}
"         = 0 {equal to zero}
"         > 0 {greater than zero}
"         ≤ 0 {less than or equal to zero}
"         ≥ 0 {greater than or equal to zero}
"         ≠ 0 {less than or greater than zero}
```

The 1700 Instructions Test--The condition in such a manner that if the condition is true, control is transferred to the operand address and if the condition is false, the program continues in its normal sequence.

The conditional jump instructions {for the A-register} in the 1700 operate in the following manner:

- A. SAZ, condition is satisfied if all 16 bits of A are 0.
- B. SAN, condition is satisfied if any or all of the 16 bits of A are ≠ 0.

DOCUMENT CLASS IMS PAGE NO. 4-16  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

- C. SAP, condition is satisfied if bit 15 (sign bit) of A is 0.
- D. SAM, condition is satisfied if bit 15 of A is  $\neq$  0.

Problems in testing arise from the manner in which these instructions operate:

1. -0, will not be detected by SAZ.
2. -0, will falsely satisfy SAN.
3. SAP, does not exactly test for greater than 0 but for  $A = +0$  or  $A > 0$ .
4. SAM, does not exactly test for less than 0 but for  $A = -0$  or  $A < 0$ .

(The same holds true for the analogous Q-register tests, SQZ, SQN, ...etc.)

To aid in effective code generation, eight special conditional jumps have been created. Each corresponds to a specific test needed by Phase B to test the accumulator. Each results in the generation of some group of real instructions. The resulting group of real instructions is left to Phase C due to information which is unavailable during Phase B operations. (Specifically, the distance of the operand address from the test instructions affects the type of code generation.)

The table below cross-references the special codes with the code that will be generated by Phase C depending upon the noted conditions:



if: Phase C will generate  
the following code:

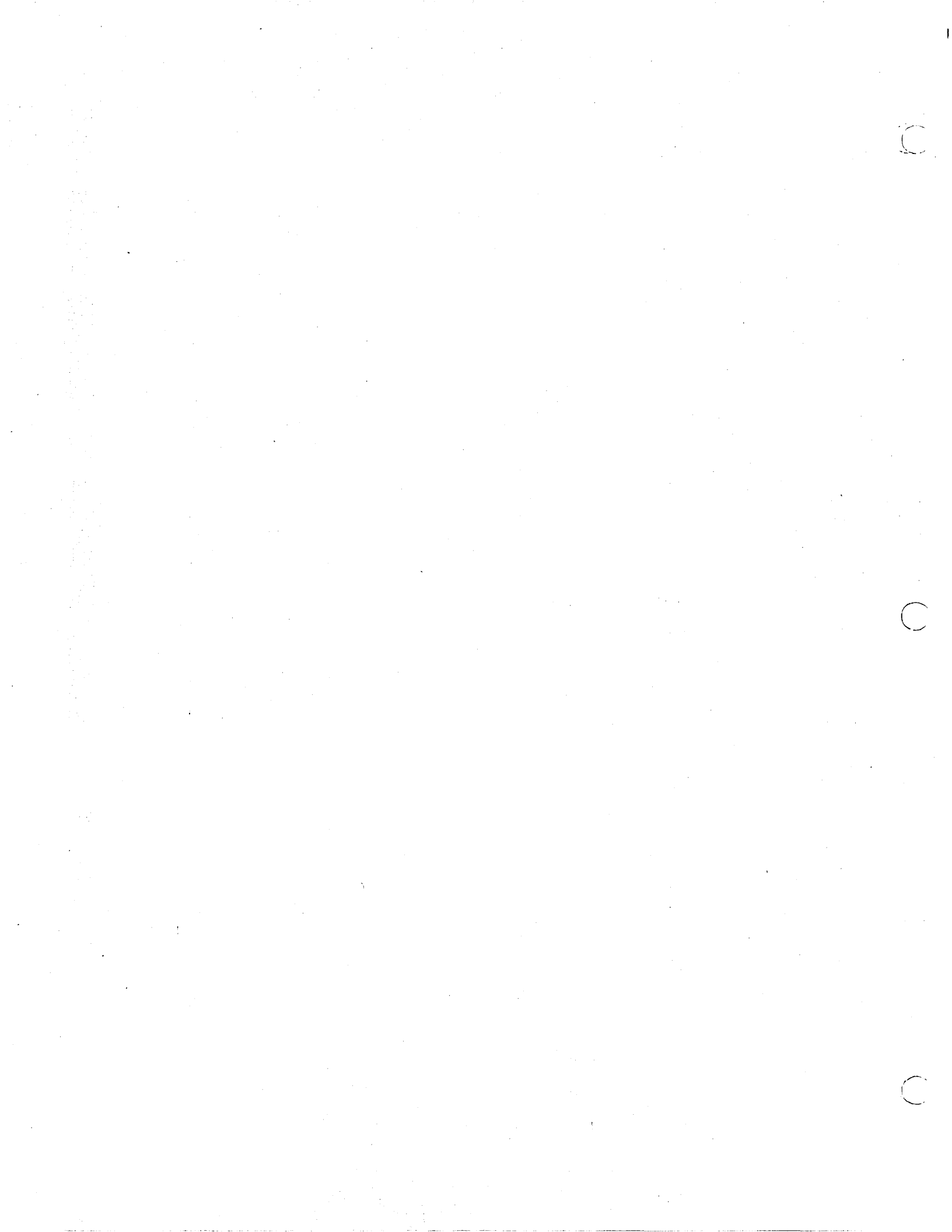
PHASE B pseudo code	Meaning	If operand address ADD is not more than 15 cells distant in a forward direction from its references:	If address Add is more than 15 cells distant in a forward direction, or is in a backward direction:
AJD,LZ	ADD $A < \underline{+0}$	SAM ADD-**-1	SAP (1 or 2) JMP* ADD
AJP,EZ	ADD $A = \underline{+0}$	SAZ ADD-**-1	SAN (1 or 2) JMP* ADD
AJP,GZ	ADD $A > \underline{+0}$	SAZ 1 SAP ADD-**-1	SAZ (2 or 3) SAM (1 or 2) JMP ADD
AJP,LEZ	ADD $A \leq \underline{+0}$	SAM ADD-**-1 SAZ ADD-**-1	SAZ 1 SAP (1 or 2) JMP* ADD
AJP,GEZ	ADD $A \geq \underline{+0}$	SAP ADD-**-1	SAM (1 or 2) JMP* ADD
AJP,LGZ	ADD $A \neq \underline{+0}$	SAN ADD-**-1	SAZ (1 or 2) JMP* ADD

(NOTE: Operands in parenthesis are two alternatives depending upon whether JMP\* ADD is a one or two-word command, respectively.)

There is limited use of Q-register testing. Only the two instructions QJP,EZ and QJP,GEZ (analogous to AJP, EZ and AJP,GEZ) are needed, thus the total number of conditional jumps is eight.

#### Word 3:

- Whenever the instruction code is an instruction which has an operand, the operand is in word 3. The meaning of the operand, whether its a constant or a symbol table pointer is dependent upon field  $\emptyset$  in the indicator word.



DOCUMENT CLASS IMS PAGE NO. 4-18  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

2. Whenever the instruction code is an instruction which has no operand, word 3 will contain no significant information.

Word 4:

Whenever field A in the indicator word is set  $\neq 0$  word 4 will contain the additive value to the operand contained in Word 3.

Word 5:

Whenever field S in the indicator is set  $\neq 0$  word 5 will contain the symtab pointer to the variable which subscripts the operand.

#### 4.2 OPERATION OF THE PHASE B DRIVER (PHASE B)

PHASE B is the driver which controls the overall operation of Phase B. Its functions are:

1. Initialize all flags and switches necessary to begin the phase.
2. Generate special program-wide labels.
3. Generate all internal arrays, constants and variables by calling HELEN.
4. Where program is run-anywhere, generate special instructions which compute relocation address at object-time.
5. Call READIR to read input records. Make decision on which processor to call based upon the statement type. Call that processor. Statement functions, arithmetic IF statements, and logical IF statements are processed in PHASEB as described in 4.2.2-4.
6. In case of logical IF reset input buffer pointers and output special labels as is appropriate.
7. Monitor the processing of statement functions and reinitialize flags and generate certain labels as appropriate.

##### 4.2.1 Program Structure

Program structure is a function of the program type and the run-anywhere switch. The illustration below will clarify how these conditions interrelate to produce the program structure. The ordering of the various elements of the program is equivalent to the ordering in the table.

DOCUMENT CLASS IMS PAGE NO. 4-19  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

run-anywhere switch	main-program	block data subprogram subroutine	function subprogram	code generated
on				LABEL <u>TOP</u> (label of top of program. Relo address 0000)
	x			LABEL Q8QNAM (generated name of main program) JMP* MBEGIN (jump from first cell of program to first executable instruction)
	x	x	x	generation of all programmer-defined arrays, variables, and constants in this FORTRAN program. (structure of this code is elucidated in section on program HELEN)
				code generation for 1st statement function,  code generation for 2nd statement function,  • • •,  code generation for n-th statement function  (structure of this code is elucidated in section dealing with statement function processing, PHASE B)
	x	x	x	

DOCUMENT CLASS IMS PAGE NO. 4-20  
 PROJECT NAME 1700 Mass Storage FORTRAN  
 PRODUCT NO. CD05 VERSION 2.0 MACHINE SERIES 1700

run-anywhere switch	main-program	block data subprogram	subroutine	function subprogram	code generated
	x		x	x	LABEL MBEGIN (label of first executable instruction in program)
on					RTJ*      BIAS ADC <del>TOP</del> BIAS      BSS      1 LDA      BIAS ADD      BIAS      -1 STA      BIAS  (this code computes a value representing the value of the relocation address of the program and places that value in BIAS. Run-anywhere considerations require that the bias must be computed at run time.)
	x		x	x	main body of program includes code generation of all executable statements, and generation of jumps around format statements.
			x	x	LABEL RETURN (label referenced by jumps which are the result of RETURN statements)
			x	x	code to restore indexes  (structure of this code is elucidated in section on program INXRST)

DOCUMENT CLASS IMS PAGE NO. 4-21  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

run-anywhere switch	main-program	block data subprogram	subroutine	function subprogram	code generated
				x	LDA or FLD of value of function into accumulator or floating point accumulator respectively.
			x	x	JMP* (KPRNAM) (indirect jump to entry point) KABEL KPRNAM
			x	x	entry code. code which is responsible for picking up parameters and plugging them into the program with the proper increments. The code also includes the entry point BSS 1 and the jump to the first executable statement (structure of this code is elucidated in section on program ENTCOD)
	x	x	x	x	NEND (terminating pseudo instruction)

4.2.2 STATEMENT FUNCTION PROCESSOR

PHASEB processes the statement function in the following steps:

1. Generates and outputs a label to tag the first executable instruction of the statement function.
2. Places the symtab pointer of the statement function name into holder.
3. Initializes table KSFAT with the statement function arguments. Upon completion of this operation the table appears in its initialized state as shown

KSFAT (12,2)

KSFAT (1,1)	ptr 1	0	
KSFAT (2,1)	ptr 2	0	KSFAT (1,2)
	•		
	•		
	•		
KSFAT (n,1)	ptr n	0	
	0		
	•		
	•		
	•		
	•		
	•		
	•		
KSFAT (12,1)			

where ptr 1-n are the symtab pointers of the statement function arguments. The 0 at KSFAT (M+1,1) signifies the logical end of the parameter list. The second word is used later to contain pointers to assigned temporary storage (See KPCSTK)

4. Update NDPRS (number of argument-increment pairs) by the number of arguments.

CONTROL DATA CORPORATION  
LA JOLLA RESOURCE CENTER \_\_\_\_\_ DIVISION

DOCUMENT CLASS IMS PAGE NO. 4-23  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

5. Determine if the mode of the right side of the statement function expression is different from the mode of the function and generate instructions to call the appropriate conversion routines {integer to real, real to integer, integer to double precision, etc.} where necessary.
6. Generate instructions to restore indexes, the exit command {indirect jump through the entry point}, the entry point, and sets up and calls ENTCOD to generate instructions for parameter pickup.
7. Decrement KSFCNT {# of statement functions} by 1.



DOCUMENT CLASS IMS PAGE NO 4-24  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT NO. 0005 VERSION 2.0 MACHINE SERIES 1700

## 4.2.3 ARITHMETIC IF STATEMENT PROCESSOR

Processes the Arithmetic IF statement. For purposes of discussion, the Arithmetic If can be described as follows:

IF (E) K1, K2, K3  
 L (statement)

where, E is the arithmetic expression

K1 is the address for E = 0

K2 is the address for E = 0

K3 is the address for E = 0

and L is the label of the next statement following the Arithmetic IF.

Processing takes place in two basic steps:

1. The expression (E) is given to subroutine ASUPER for processing. PHASEB does not directly call ASUPER however, but goes through routine AFIDL which in turn calls ASUPER and is responsible for insuring that after the expression has been evaluated that -0 is eliminated thus simplifying the conditional testing.
2. One or more conditional and/or unconditional jumps are generated such that testing is performed in the most efficient manner. Two indexes are used IDTI and IDTJ whose values are dependent upon various considerations about the values of K1, K2, etc. The values of these indexes then determine the most efficient code to be generated. The settings and their meanings are shown:

IDTI = 1, K1=K2  
 IDTI = 2, K1=K3  
 IDTI = 3, K2=K3  
 IDTI = 4, K1≠K2≠K3≠K1

IDTJ = 1, K1=L  
 IDTJ = 2, K2=L (K1≠L)  
 IDTJ = 3, K3=L (K1≠L, K2≠L)  
 IDTJ = 4, K1≠L, K2≠L, K3≠L

DOCUMENT CLASS TMS PAGE NO. 4-25  
 PRODUCT NAME 17 Mass Storage FORTRAN  
 PRODUCT NO. C005 2.0 MACHINE SERIES 1700

After values for IDTI and IDTJ have been computed, then instructions are generated according to the following table:

		IDTI=			
		1	2	3	4
IDTJ=	1	AJP,GZ K3	AJP,EZ K2	AJP,GEZ K3	AJP,LZ K1 AJP,ZR K2 JMP K3
	2	/	AJP,LGZ K1	AJP,LZ K1	AJP,LZ K1 AJP,ZR K2 JMP K3
	3	AJP,LEZ K1	/	/	AJP,LZ K1 AJP,ZR K2
	4	AJP,LEZ K1 JMP K3	AJP,LGZ K1 JMP K2	AJP,LZ K1 JMP K2	AJP,LZ K1 AJP,ZR K2 JMP K3

4.2.4 LOGICAL IF STATEMENT PROCESSOR

PHASEB is the processor of the Logical IF statement.  
 For reference, the Logical IF is defined:

IF (e) S

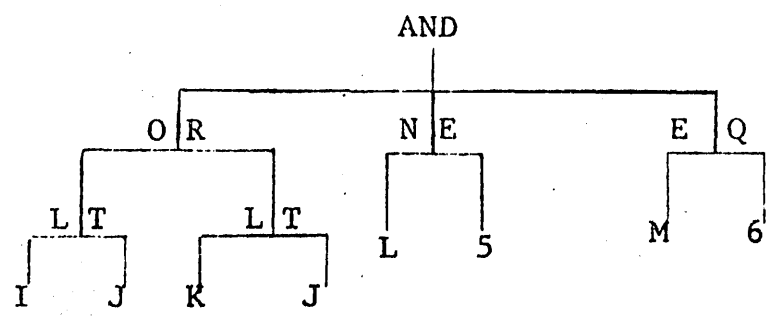
where:

- e is a logical expression
- S is any executable statement except a DØ statement or another Logical If.

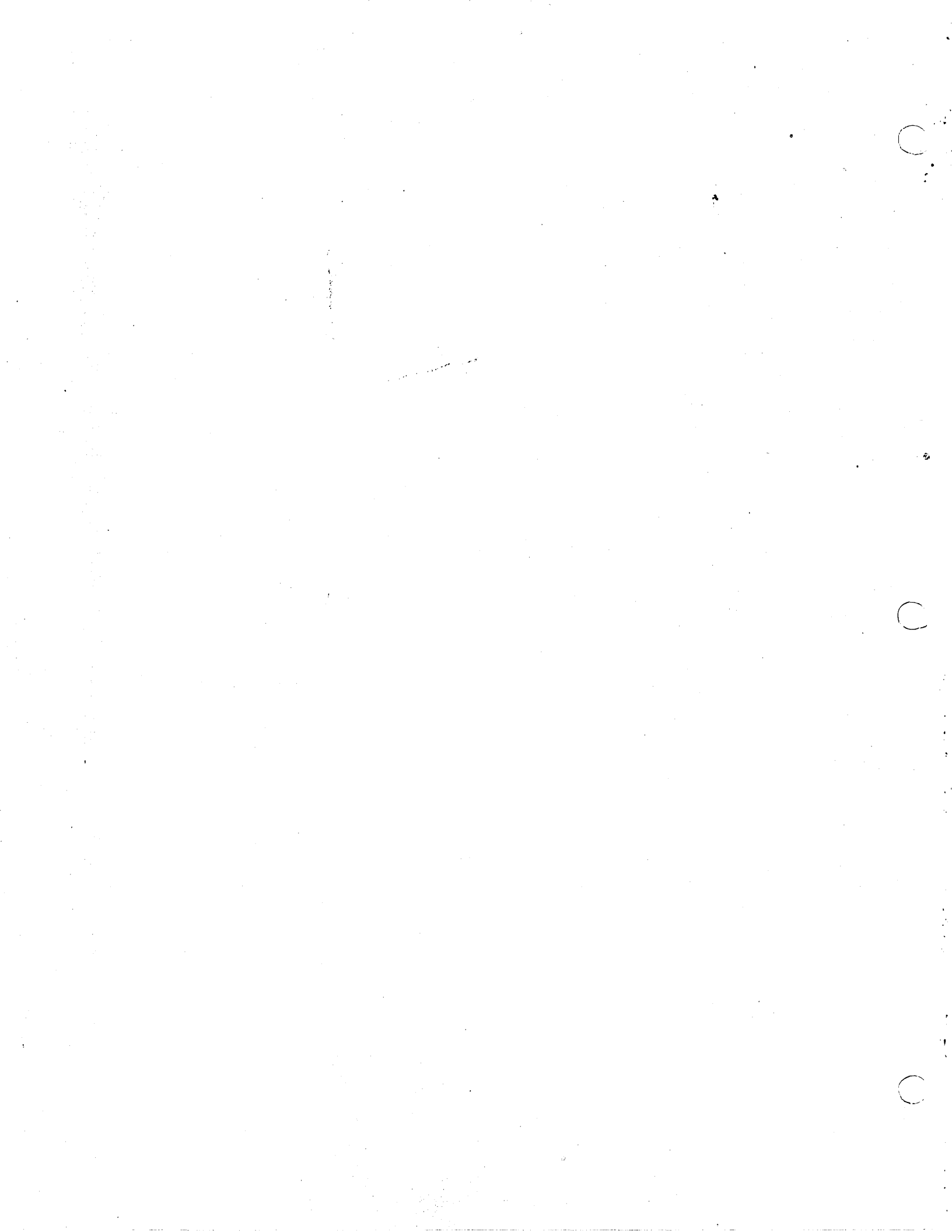
The logical expression becomes a "tree structure" in Phase A and is processed as such in Phase B. "Level" tables are employed in the processing of the tree similar to those used in the processing of arithmetic expressions. For illustration the following logical expression will be used:

((I.LT.J.OR.K.LT.J).AND.(L.NE.5).AND.(M.EQ.6))

The tree structure becomes:



Relational expressions (I.LT.J, L.NE.5...etc.) are treated simply as arithmetic expressions where the operator is + and one of the two variables is made an inverse. The table below indicates which variable is made the inverse:



DOCUMENT CLASS IMS PAGE NO. 4-27  
 PRODUCT NAME 1700 Mass Storage FORTRAN  
 PRODUCT NO. CU05 VER. NO. 2.0 MACHINE SERIES 1700

relational expression is interpreted:

I.LT.J	I-J
I.GT.J	J-I
I.LE.J	J-I
I.GE.J	I-J
I.EQ.J	I-J
I.NE.J	I-J

The order is immaterial in the EQ and NE cases. But in the other 4 cases the order of subtraction is determined such that the fewest number of test instructions are required to determine whether or not the condition has been satisfied.

Every logical and relational operator is treated as a binary value; that is either true or false. According to the true or false condition of the operator there is a corresponding branch to either of two possible points.

All logical and relational operations are assigned "truth" and "false" addresses which are based upon:

1. The type of logical operator of which they are a branch.
2. Whether or not the branch is the last branch of the logical operator.

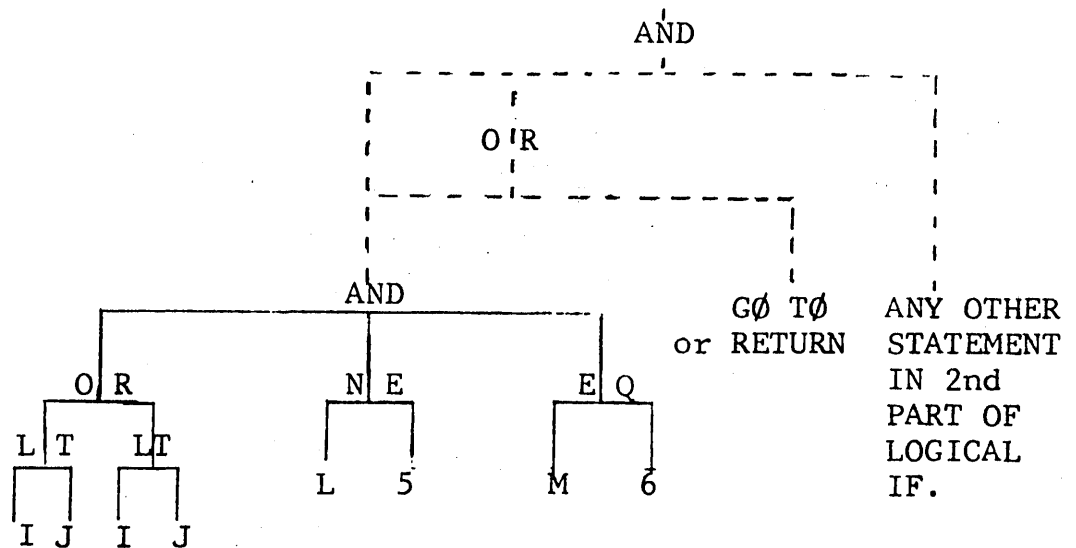
The following table shows how assignments are made:

The type of logical operator of which this operator is a branch	This operator <u>is not</u> the <u>last</u> branch		This operator <u>is</u> the <u>last</u> branch	
	truth address	false address	truth address	false address
AND	the next branch of the AND	the same as the false address of the AND operator	the same as the truth address of the AND operator	the same as the false address of the AND operator
ØR	the same as the truth address of the OR operator	the next branch of the OR	the same as the truth address of the OR operator	the same as the false address of the OR operator

Processing proceeds as follows.

1. The 2nd statement of the Logical IF (S) is also read into the input buffer and it is determined whether or not that statement is an unconditional GØ TØ or RETURN statement.
2. If the 2nd statement was a GØ TØ or RETURN the truth address of the highest level operator is set to that address. (LOGIF is set to 2)
3. If the 2nd statement was not a GO TO or RETURN, the truth address is a generated label. (LOGIF is set to 1)
4. The false address is a generated label.
5. In the actual level tables the highest (1st) level operator is actually an artificial one based upon the following criterion:
  - a. If the 2nd statement was a GO TO or RETURN, the operator is set to ØR.
  - b. If not, the operator is set to AND.

The actual logical expression then becomes the first branch of an artificially created AND or OR operator:



6. The processing then proceeds to cycle through all the branches of the logical operator (from left to right) when a branch is itself a logical operator, the level tables are updated to the next level down and the processing proceeds at that level with the first branch. When a branch is encountered which is a relational operator (and therefore an arithmetic expression) the code is generated via subroutine ASUPER to compute the value of the expression.

Next, a conditional jump command is generated. Each relational expression has associated with it a "truth" condition and a "false" condition and likewise a corresponding conditional jump command:

relational	arithmetic	conditional jump	
		true	false
I.LT.J	I-J	AJP,LZ	AJP,GEZ
I.GT.J	J-I	AJP,LZ	AJP,GEZ
I.LE.J	J-I	AJP,GEZ	AJP,LZ
I.GE.J	I-J	AJP,GEZ	AJP,LZ
I.EQ.J	I-J	AJP,EZ	AJP,LGZ
I.NE.J	I-J	AJP,LGZ	AJP,EZ

Only one of the two conditional jumps is actually generated. It is always true that either the truth address or the false address is the next expression to be computed (graphically, left to right across the tree, the "next" expression is the one to the immediate right; henceforth called the "fall-through address.") Therefore, if the truth case is actually the "fall-through," the conditional jump is the jump to the "false" address. If the false case is actually the "fall-through" the conditional jump is the jump to the "truth" address.

Returning to the example code is generated in the following order:

1. Test if I is less than J.  
    compute I-J
2. If true, go to the next branch of the AND  
    AJP,LZ LABEL 1
3. If false, fall through.  
    test if K is less than J  
    compute K-J
4. If false, go to the address of the statement  
    after the Logical If.  
    AJP,GEZ LABEL 2
5. If true, fall through. Output label LABEL 1  
    test if L is not equal to 5  
    compute L-5
6. If false, go to the address of the statement  
    after the Logical IF  
    AJP,EZ LABEL 2



DOCUMENT CLASS TMS PAGE NO. 4-31  
 PRODUCT NAME 1700 Mass Storage FORTRAN  
 PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

7. If true, fall through.  
 Test if M is equal to 6.  
     compute M-6
8. (If 2nd half of Logical If is neither a GØ TØ or RETURN)  
 If false, go to the address of the statement after the Logical IF.  
     AJP,EZ LABEL 2  
 If true, fall through to the statement which is the 2nd half of the Logical IF
9. (If 2nd half of Logical If is a GØ TØ or RETURN)  
 If true, go to the address specified in the GØ TØ or implicit in the RETURN.  
     AJP,NEZ X  
 If false, fall through to the statement after the Logical If. (The GØ TØ or RETURN is then ignored.)

Level Tables

Index	NDTYP	NBRNS	ICBRN	KOP	KTRUT	KFALS	KFATH	
LIFTX →								level 1
								level 2
								level 3
								.
								.
								.
								.
								level n

where:

- NDTYP = type of operator (AND,ØR) (node type).
- NBRNS = number of branches
- ICBRN = current branch
- KØP = (working pointer)
- KTRUT = holder of "truth" address symtab ptr.
- KFALS = holder of "false" address symtab ptr.
- KFATH = holder of "fall-through" address symtab ptr.

DOCUMENT CLASS IMS PAGE NO 4-32  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT NO. 005 VERSION 2.0 MACHINE SERIES 1700

PHASE $\beta$  checks the LOGIF switch whose settings are:

- LOGIF = 0, no Logical IF statement has occurred
- = 1, 2nd statement of Logical IF was not a G $\emptyset$  T $\emptyset$  or RETURN
- = 2, 2nd statement of Logical IF was a G $\emptyset$  T $\emptyset$  or RETURN

IF LOGIF = 1 PHASE $\beta$  sets up its buffer and buffer index for processing of the 2nd half statement of the

Logical IF and outputs the level 1 "fall-through" label if LOGIF = 2 PHASE $\beta$  ignores the 2nd half statement and outputs the level 1 "false" label. It then returns to proceed with normal processing and read in the next statement. LOGIF is set to 0.

After the completion of any other statement processing, LOGIF is tested for =1. If so PHASE outputs the level 1 "false" label, then sets LOGIF = 0.

PHASE B

INITIALIZATION

KTØFLG =  
KRTNS =  
KFCSW =  
  
NDPRS =  
KSUBAT(1) =  
KSFAT(1,1) =  
LØGIF =  
INTRAS(1) =  
LIFTX = 0

LOOPTX =  
KLLTBX =  
INFLAG =  
KSEBUX = 1

A

A  
OUTBUF(3) = 49

LABLER (KRETRN)

IS PROGRAM RUN ANYWHERE?  
NO  
YES

B

B  
LABLER (KTOP)

LABKPC (LABEL KTOP)

LABLER (MBEGIN)

C

C  
KPRNAM = LABX

WAS THERE A PROGRAM CARD?

IS THIS A MAIN OR BLOCK DATA PROG.?

2B

LABLER (KPRNAM)

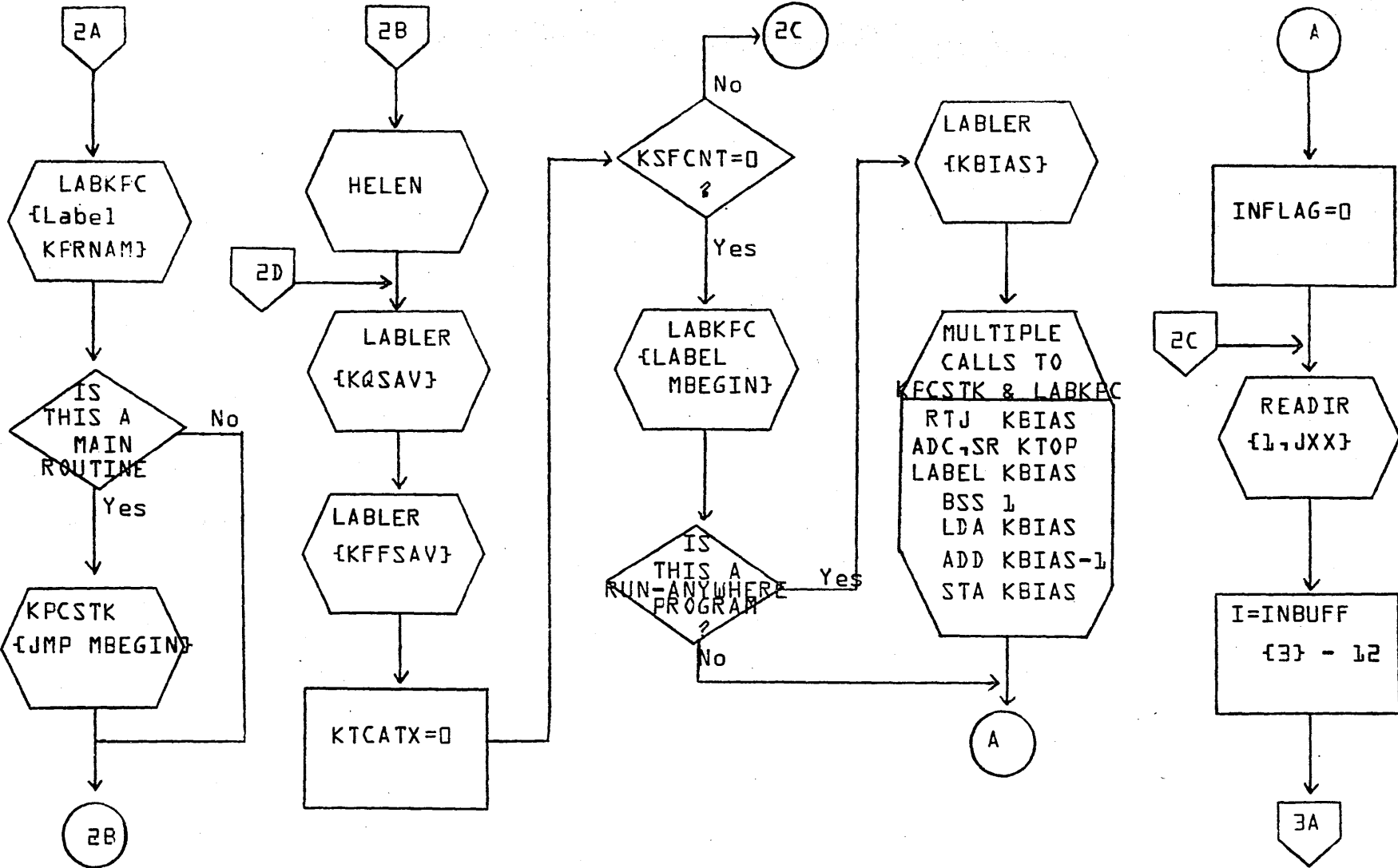
KPRNAM SYMBOL SET TO  
QBQNAM for main  
OR  
QBQBDs for block data

2A

CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE   
FLOWCHART   
DECISION TABLE   
OTHER

DOCUMENT CLASS <i>IAS</i>	MACH. TYPE <i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE <i>PHASE B</i>		PROJECT MGR.			
	PAGE <i>1</i> OF <i>20</i>	PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.			
DRAWN BY	DATE <i>1/1</i>	TASK NAME			



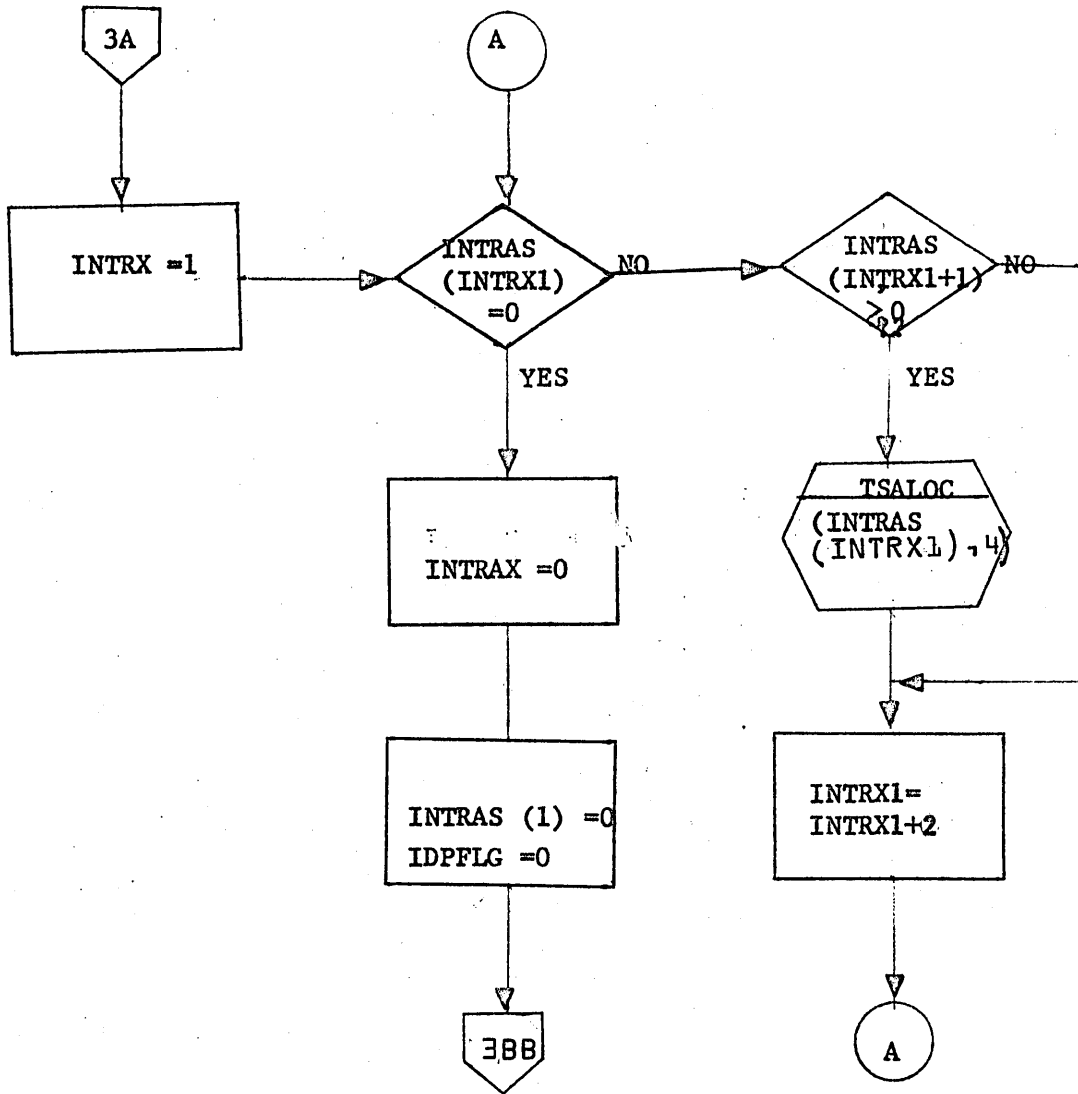
**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	PHASE 8	PAGE 2 OF 208		PROJECT MGR.			
NUMBER	ISSUE DATE			PROJECT NAME			
DRAWN BY	DATE			TASK NO.			
				TASK NAME			

**CONTROL DATA**

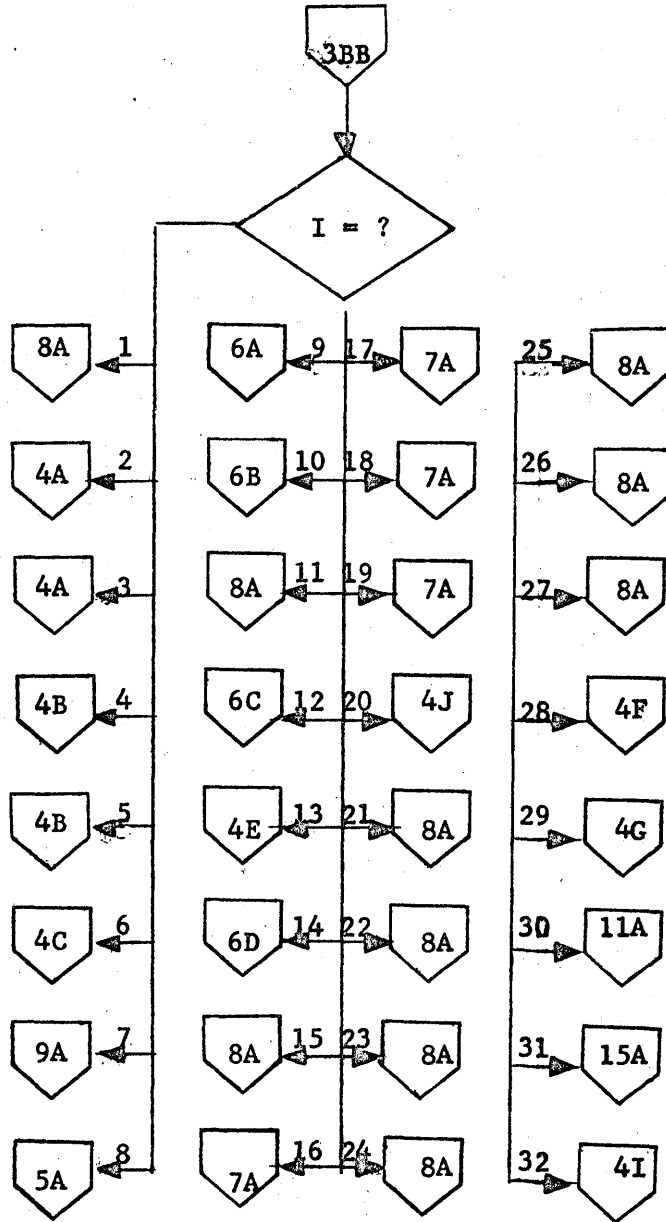
LA JOLLA RESOURCE CENTER



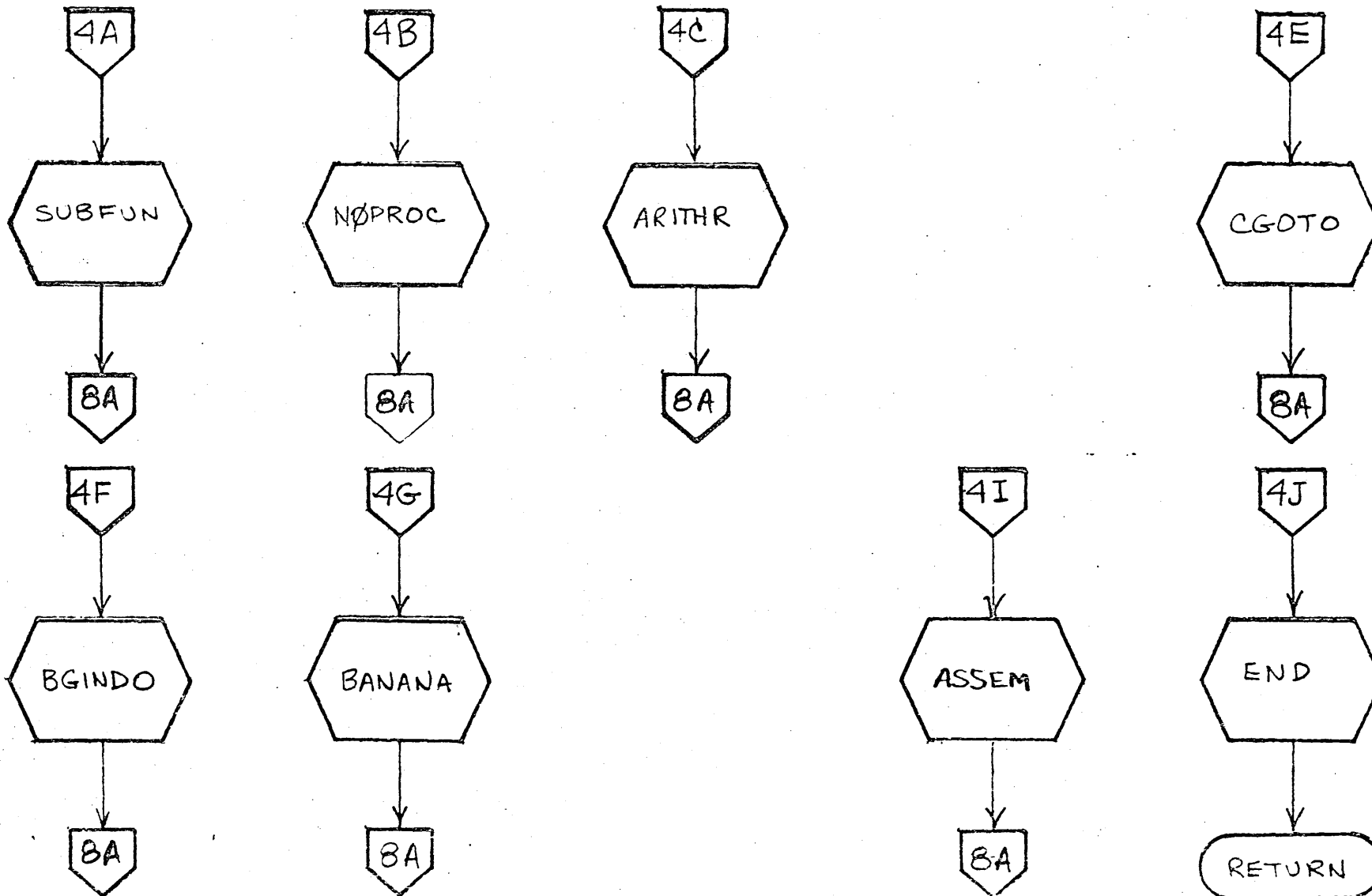
TITLE		PHASE B		DRG. NO.	
				REVISION	
DRAWN BY	PROJ.	DATE	SHEET	3	OF 20



LA JOLLA RESOURCE CENTER



TITLE PHASEB			DRG. NO.	
			REVISION	
DRAWN BY	PROJ.	DATE	SHEET 3A OF 20	
CSD 203 ANALYSIS				



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

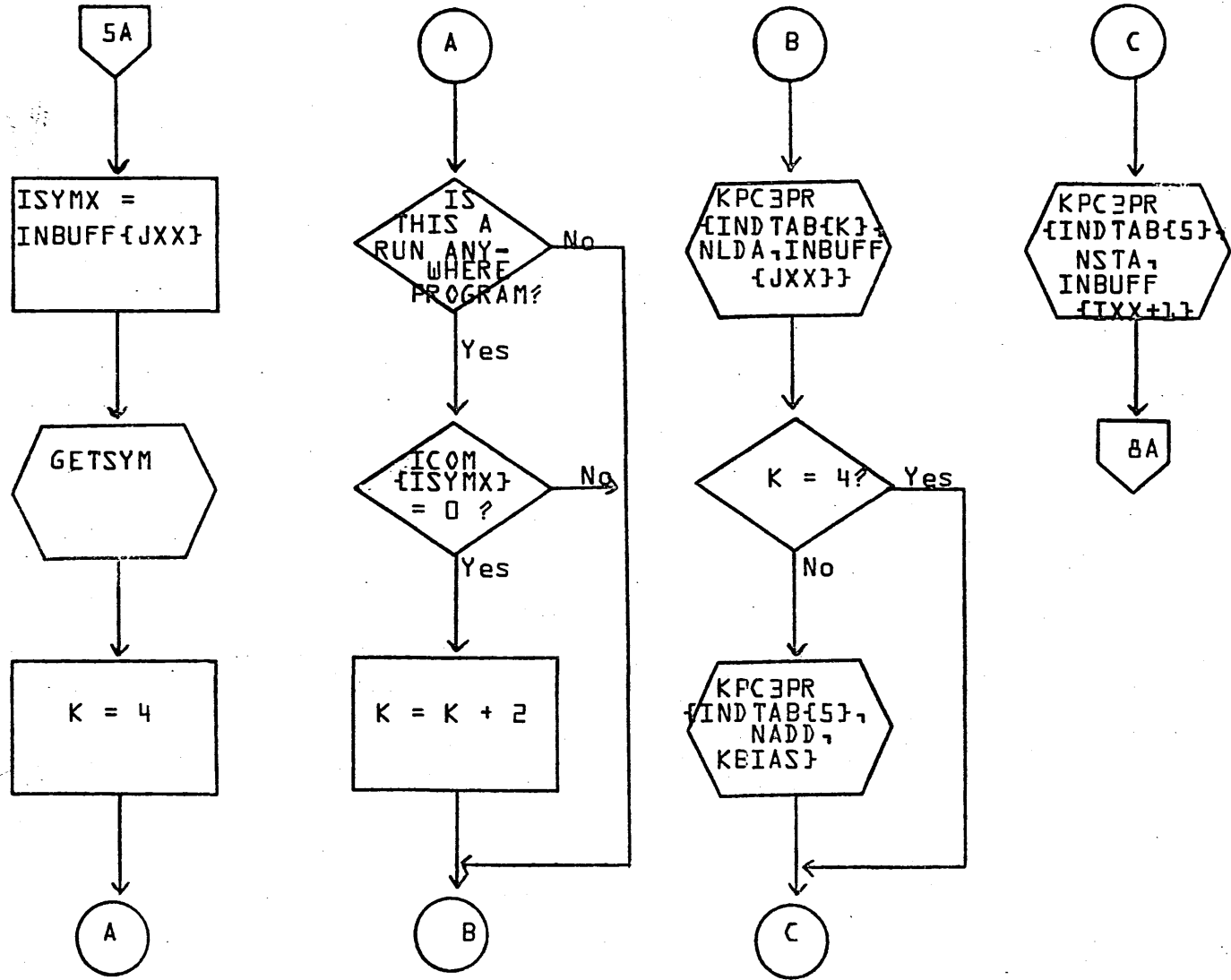
SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

DOCUMENT CLASS	<i>IIMS</i>	MACH. TYPE	<i>1700</i>	PROJECT NO.		APPROVED	DATE
DOCUMENT TITLE	<i>PHASE B</i>			PROJECT MGR.			
			PAGE <i>4</i> OF <i>20</i>	PROJECT NAME			
NUMBER		ISSUE DATE		TASK NO.			
DRAWN BY		DATE	<i>3 67</i>	TASK NAME			





A  
B  
C  
D



**CONTROL DATA CORPORATION**  
**SOFTWARE DOCUMENT**

SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

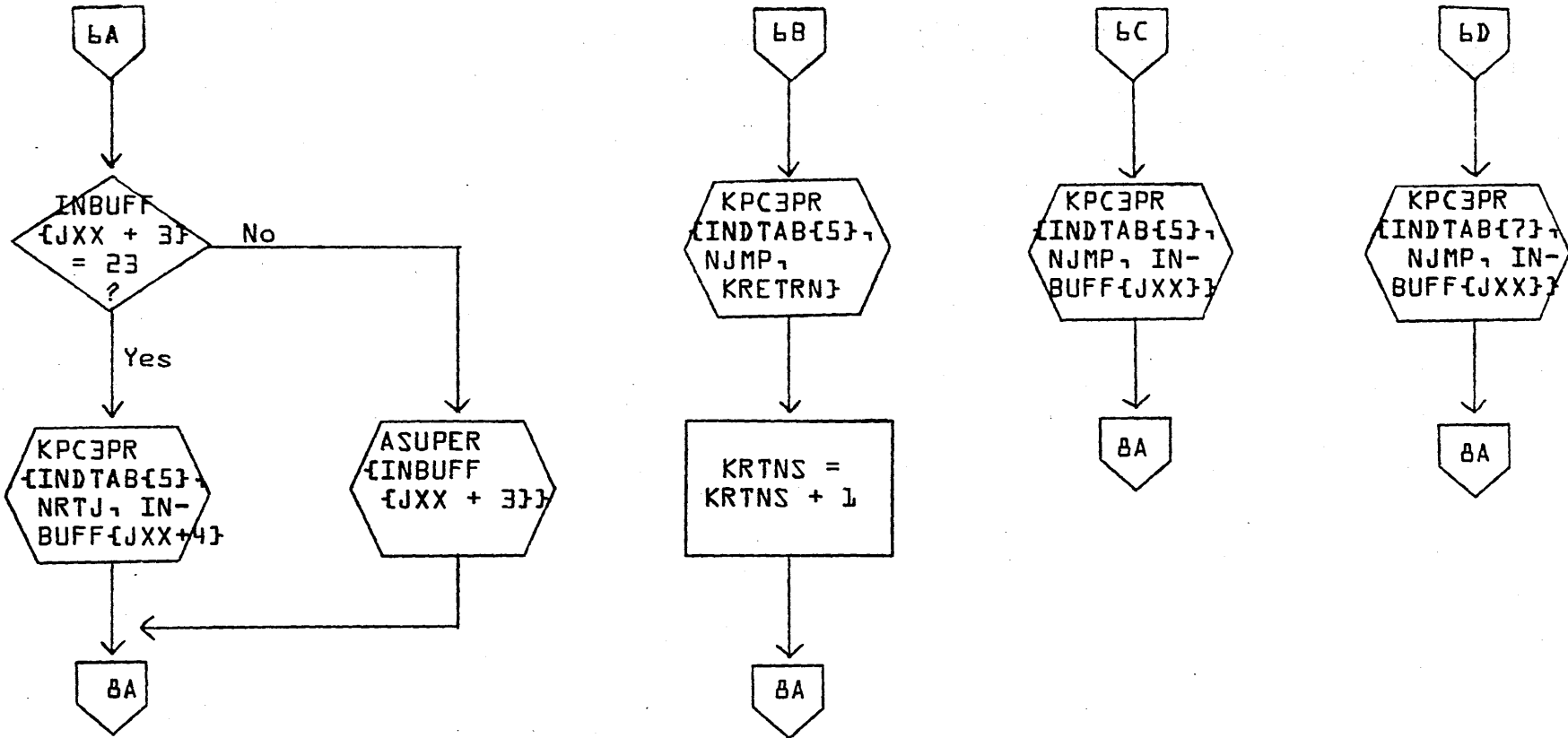
DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	PHASE B			PROJECT MGR.			
			PAGE 5 OF 20	PROJECT NAME			
NUMBER		ISSUE DATE		TASK NO.			
DRAWN BY		DATE		TASK NAME			

A

B

C

D



**CONTROL DATA CORPORATION**  
 SOFTWARE DOCUMENT

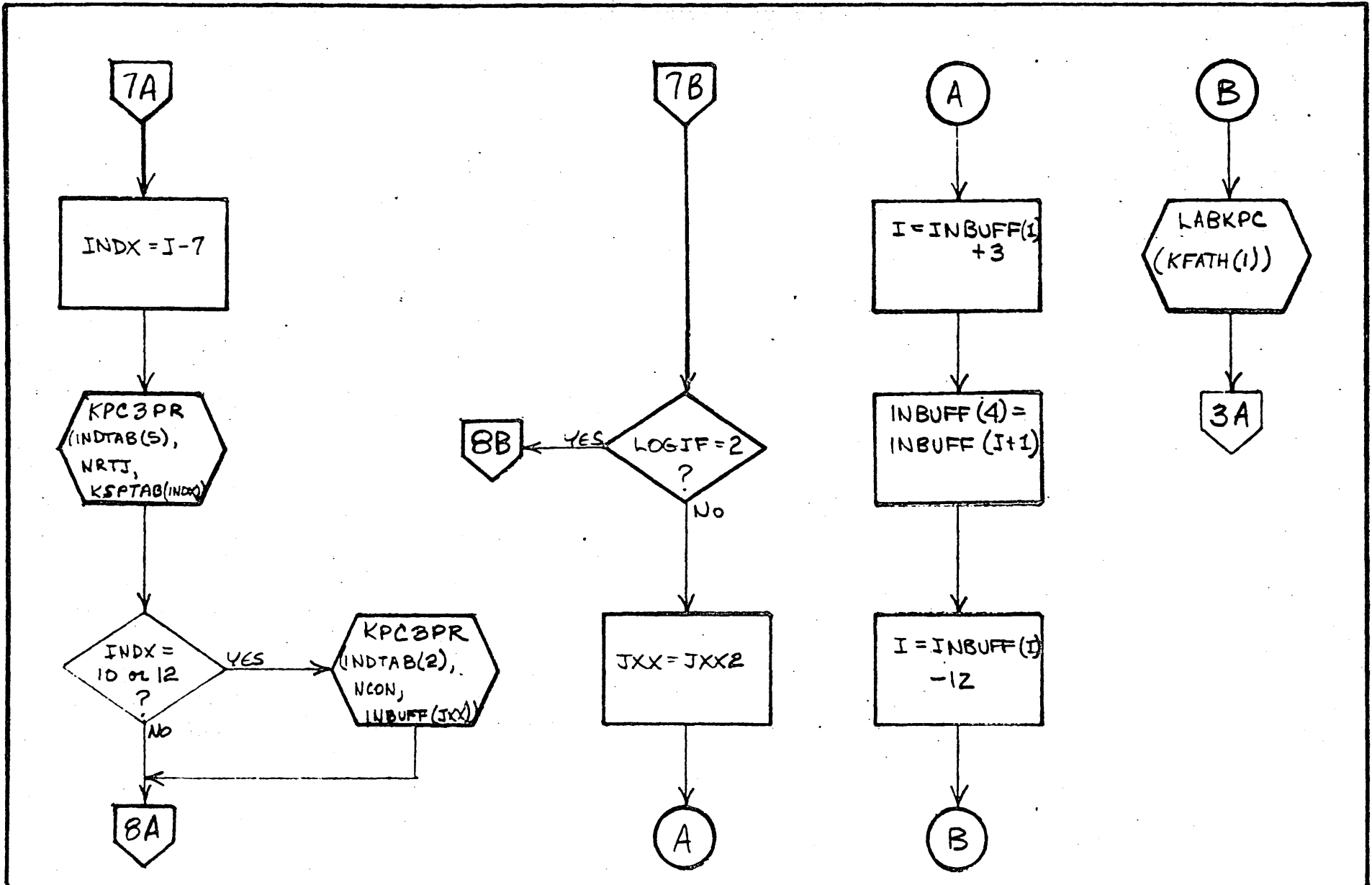
SAMPLE CODE

FLOWCHART

DECISION TABLE

OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	PHASEB	PAGE 6 OF 20		PROJECT MGR.			
NUMBER	ISSUE DATE			PROJECT NAME			
DRAWN BY	DATE			TASK NO.			
				TASK NAME			



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE

FLOWCHART

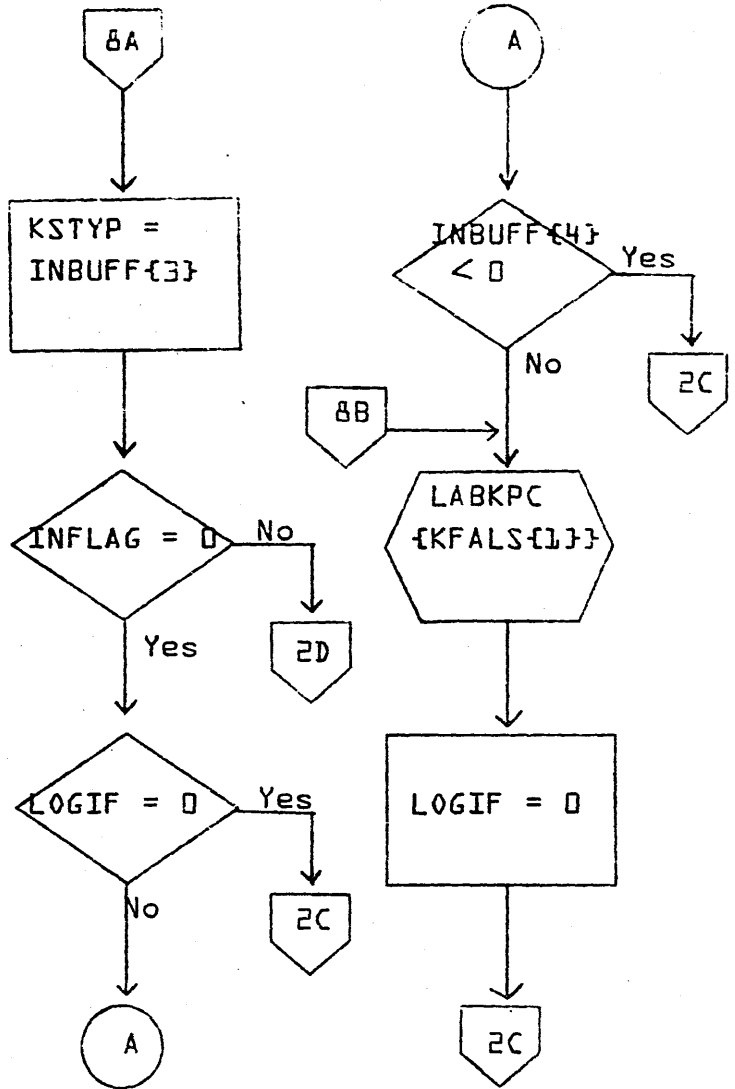
DECISION TABLE

OTHER

DOCUMENT CLASS	<i>IMS</i>	MACH. TYPE	<i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	<i>PHASE B</i>	PAGE <i>7</i> OF <i>20</i>		PROJECT MGR.			
NUMBER	ISSUE DATE			PROJECT NAME			
DRAWN BY	DATE <i>3-67</i>			TASK NO			
				TASK NAME			

bE-h

A  
B  
C  
D

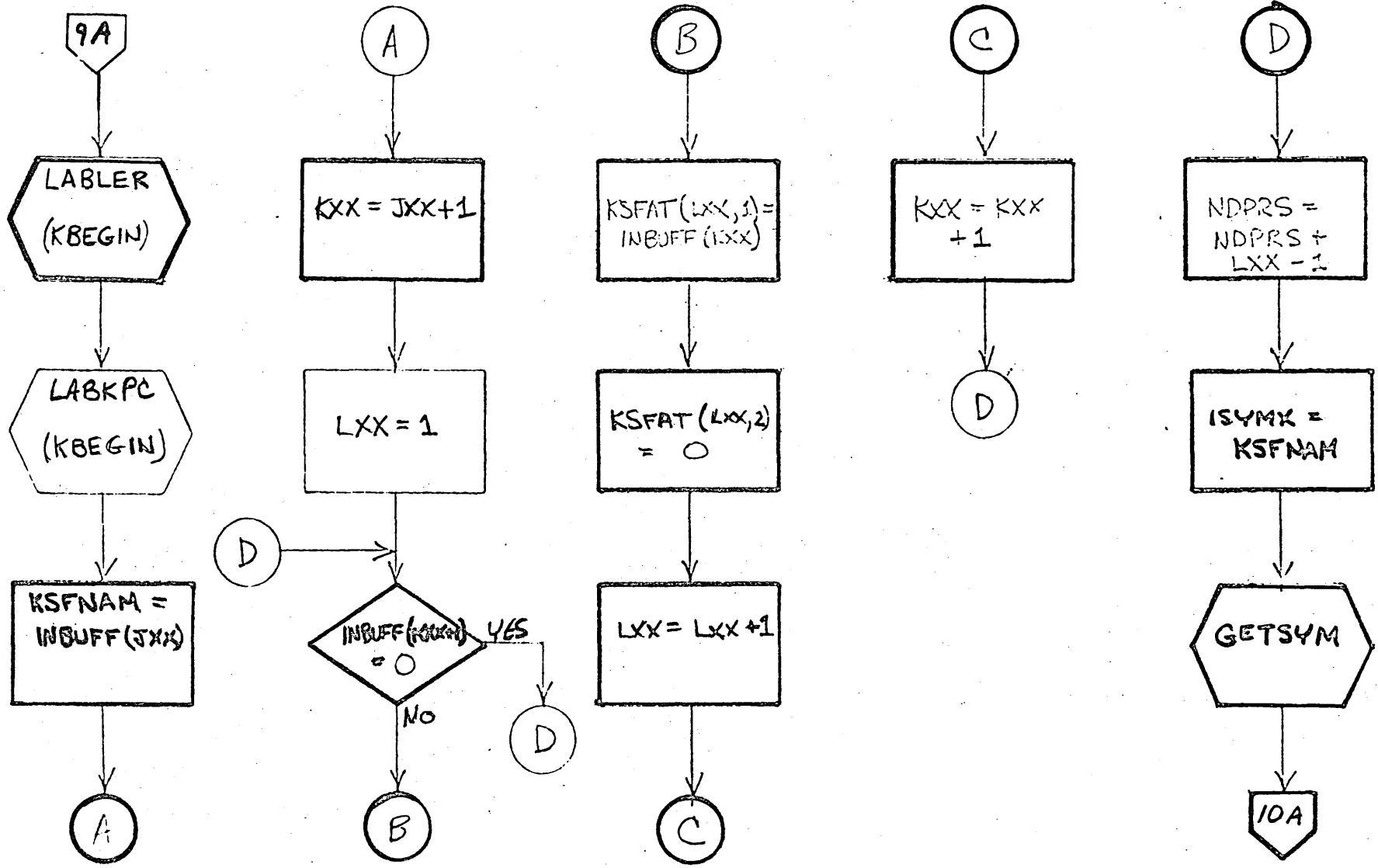


CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	PHASE 4			PROJECT MGR.			
		PAGE	8 OF 8	PROJECT NAME			
NUMBER		ISSUE DATE		TASK NO.			
DRAWN BY		DATE		TASK NAME			

Oh-h



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE

FLOWCHART

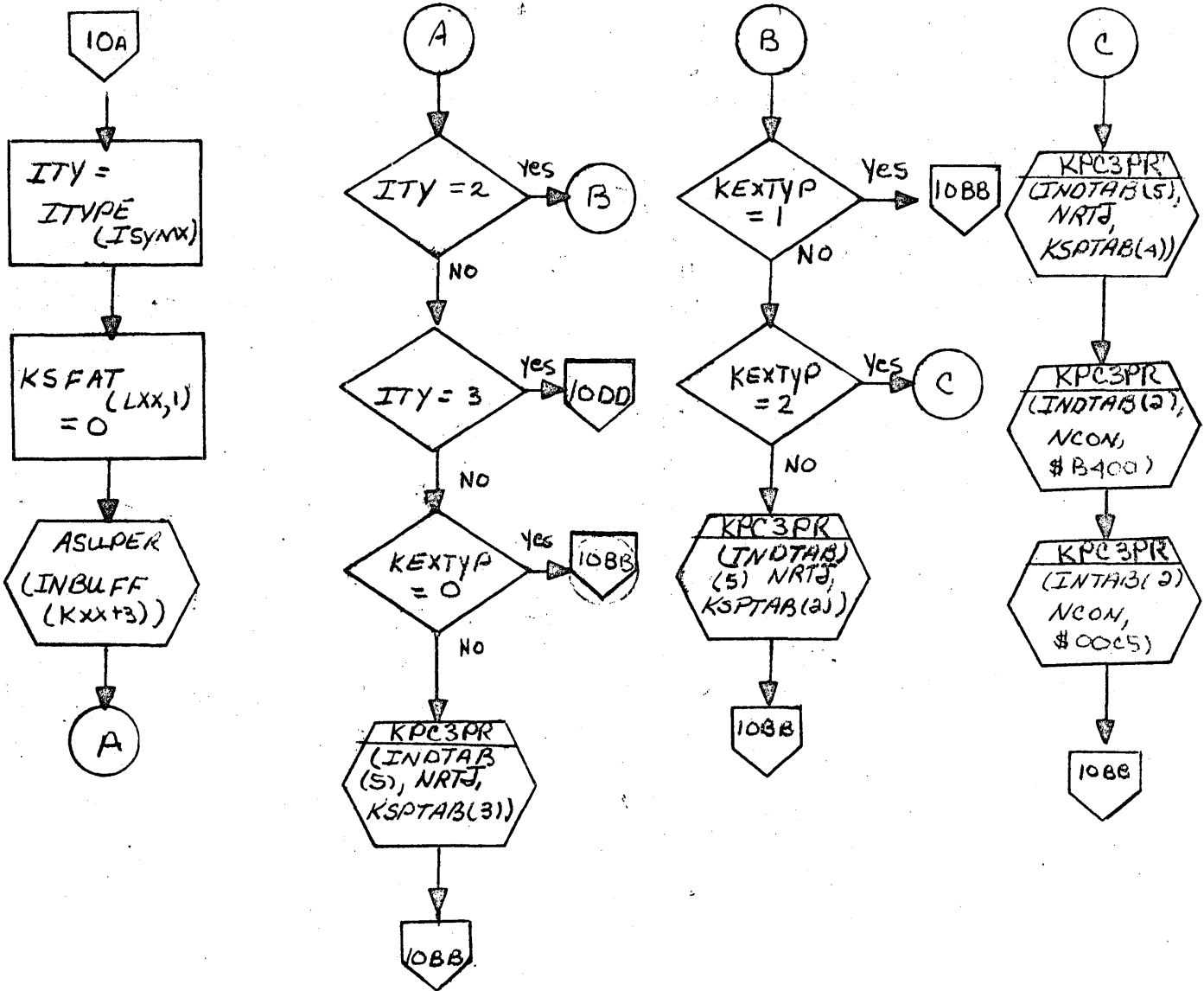
DECISION TABLE

OTHER

DOCUMENT CLASS	<b>IMS</b>	MACH. TYPE	<b>1700</b>	PROJECT NO.		APPROVED	DATE
DOCUMENT TITLE	<b>PHASE B</b>			PROJECT MGR.			
		PAGE	<b>9</b> of <b>20</b>	PROJECT NAME			
		ISSUE DATE		TASK NO.			
DRAWN BY		DATE		TASK NAME			



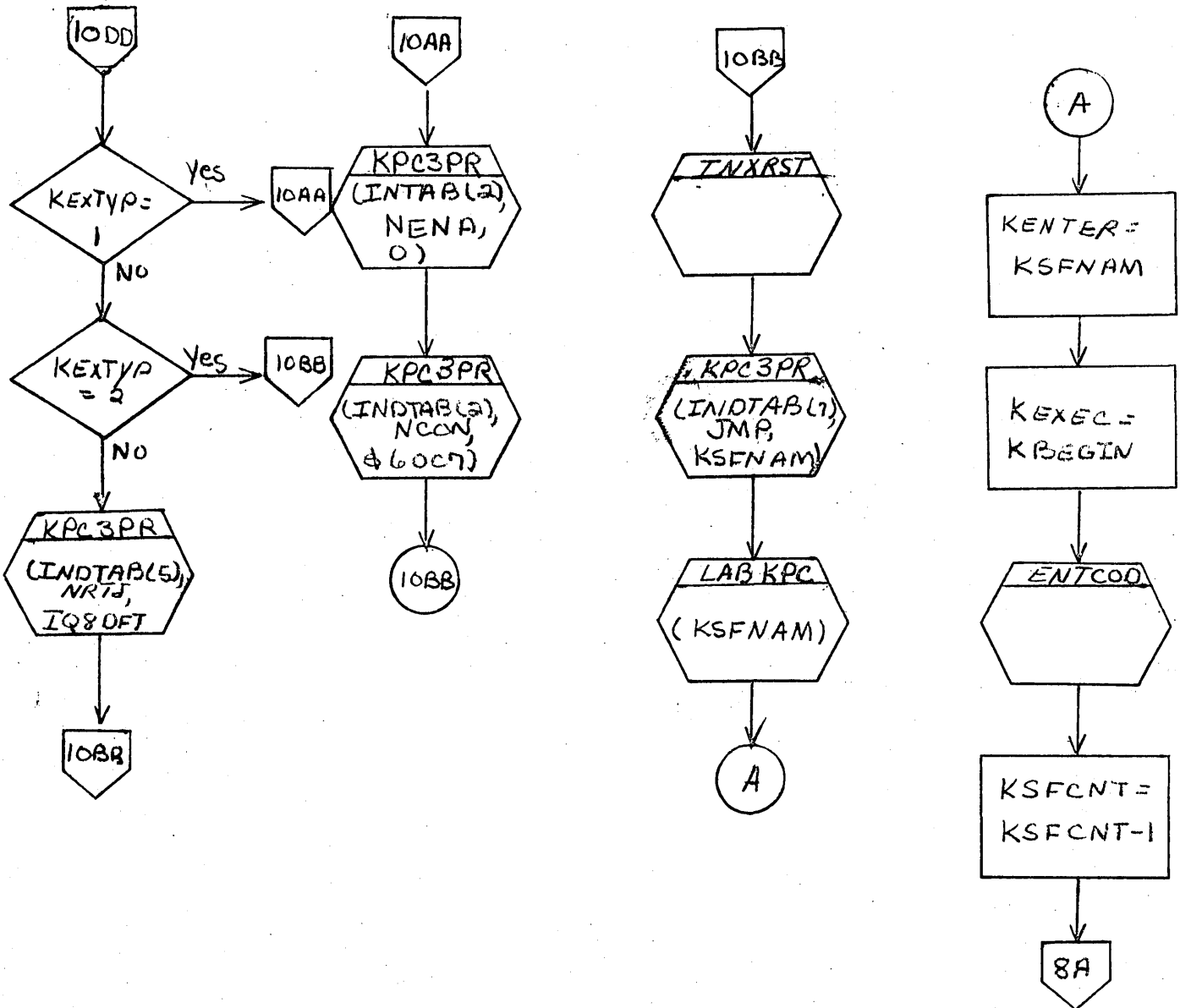
LA JOLLA FACILITY



TITLE			PHASEB		DRG. NO.	
DRAWN BY			PROJ.		REVISION	
CSD 203			DATE		SHEET 10 OF 20	



LA JOLLA FACILITY



TITLE		PHASE B		DRG. NO.	
				REVISION	
DRAWN BY	PROJ.	DATE	SHEET 10A OF 20		
CSD 203					





1

2

3

4

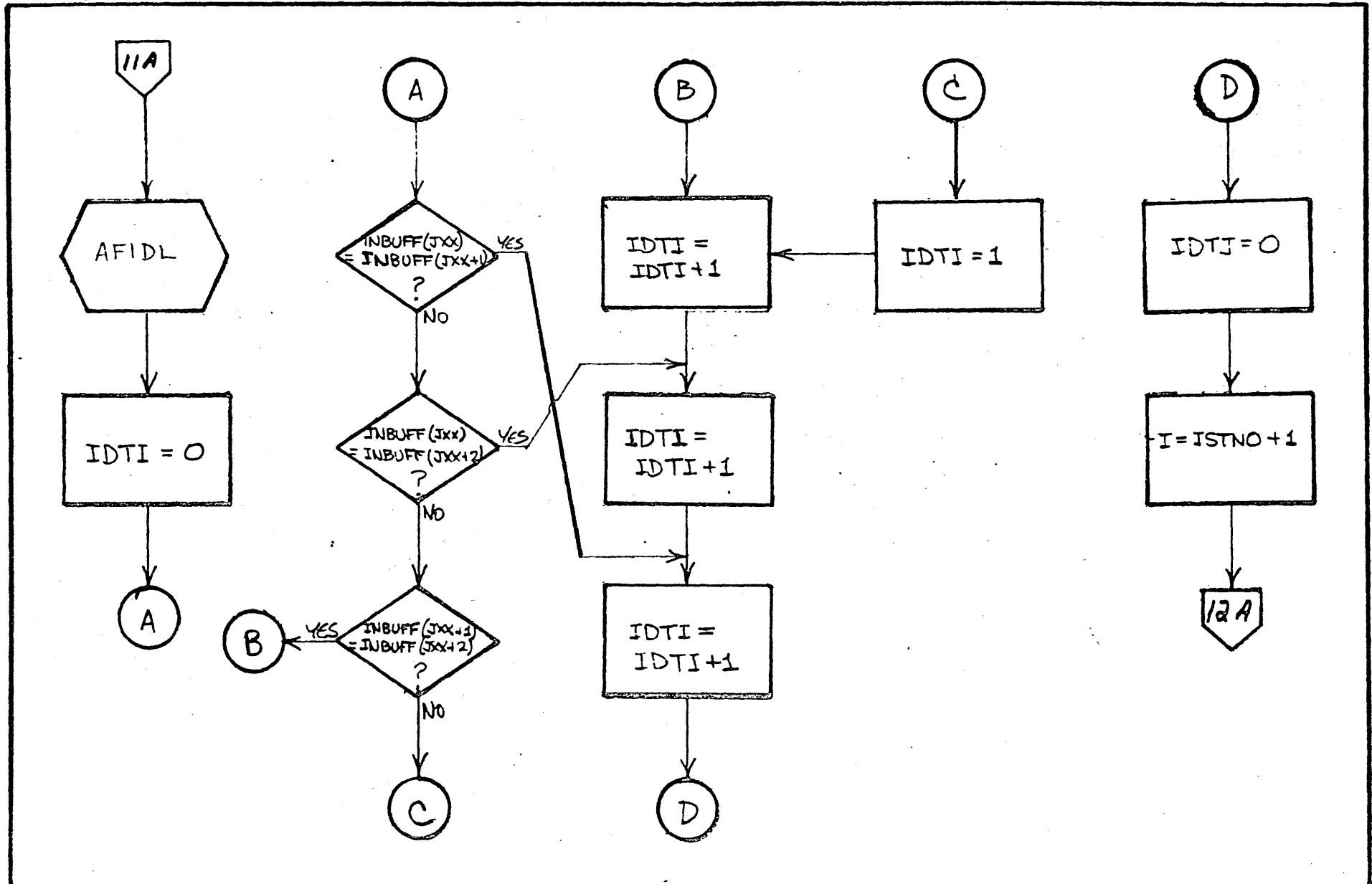
5

A

B

C

D



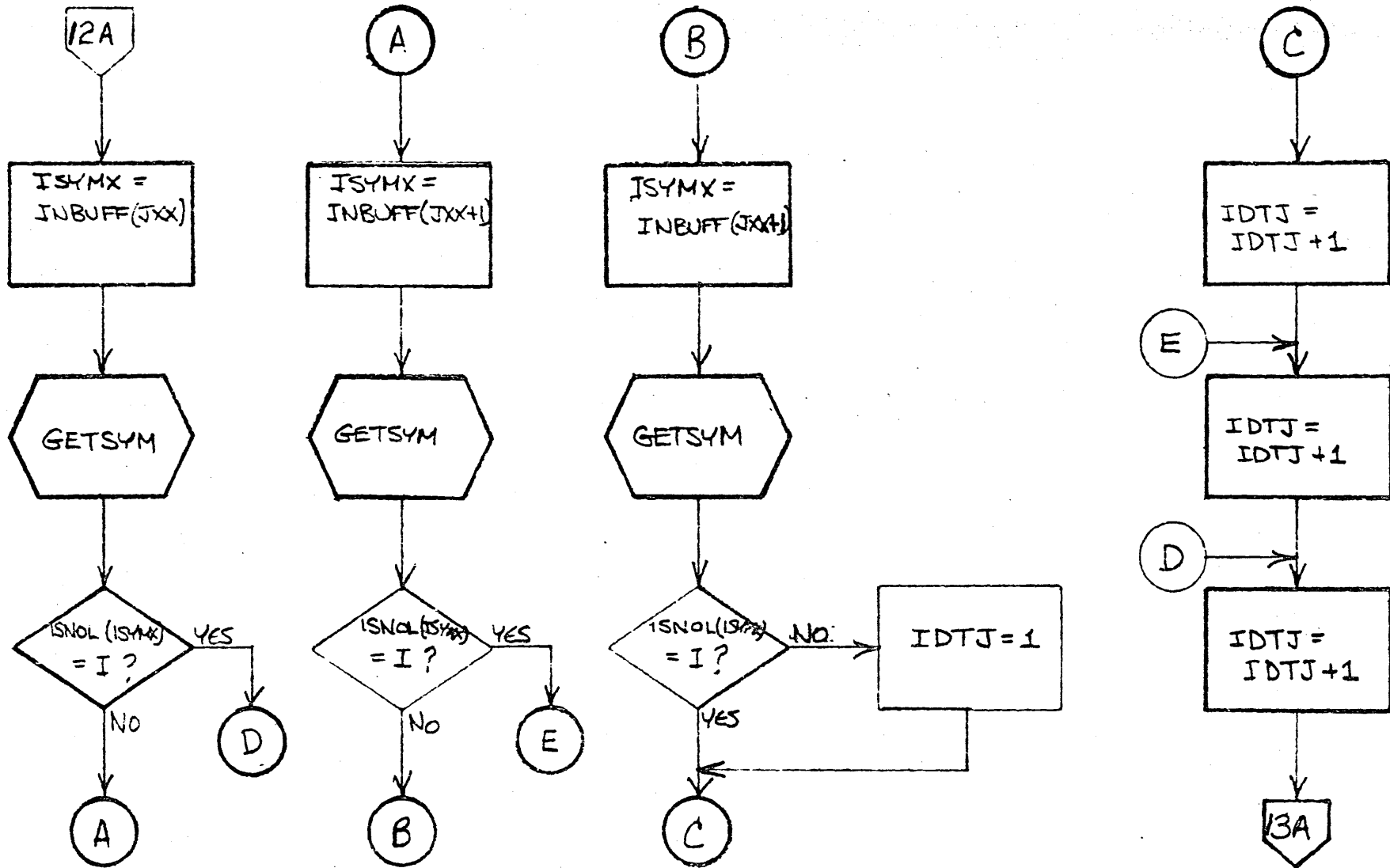
CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	<i>INS</i>	MACH. TYPE	<i>1700</i>
DOCUMENT TITLE	<i>PHASE B</i>		
	PAGE <i>11</i> OF <i>20</i>		
NUMBER	ISSUE DATE	TASK NO.	
DRAWN BY	DATE <i>3-6-7</i>	TASK NAME	

REV.	APPROVED	DATE

EN-F

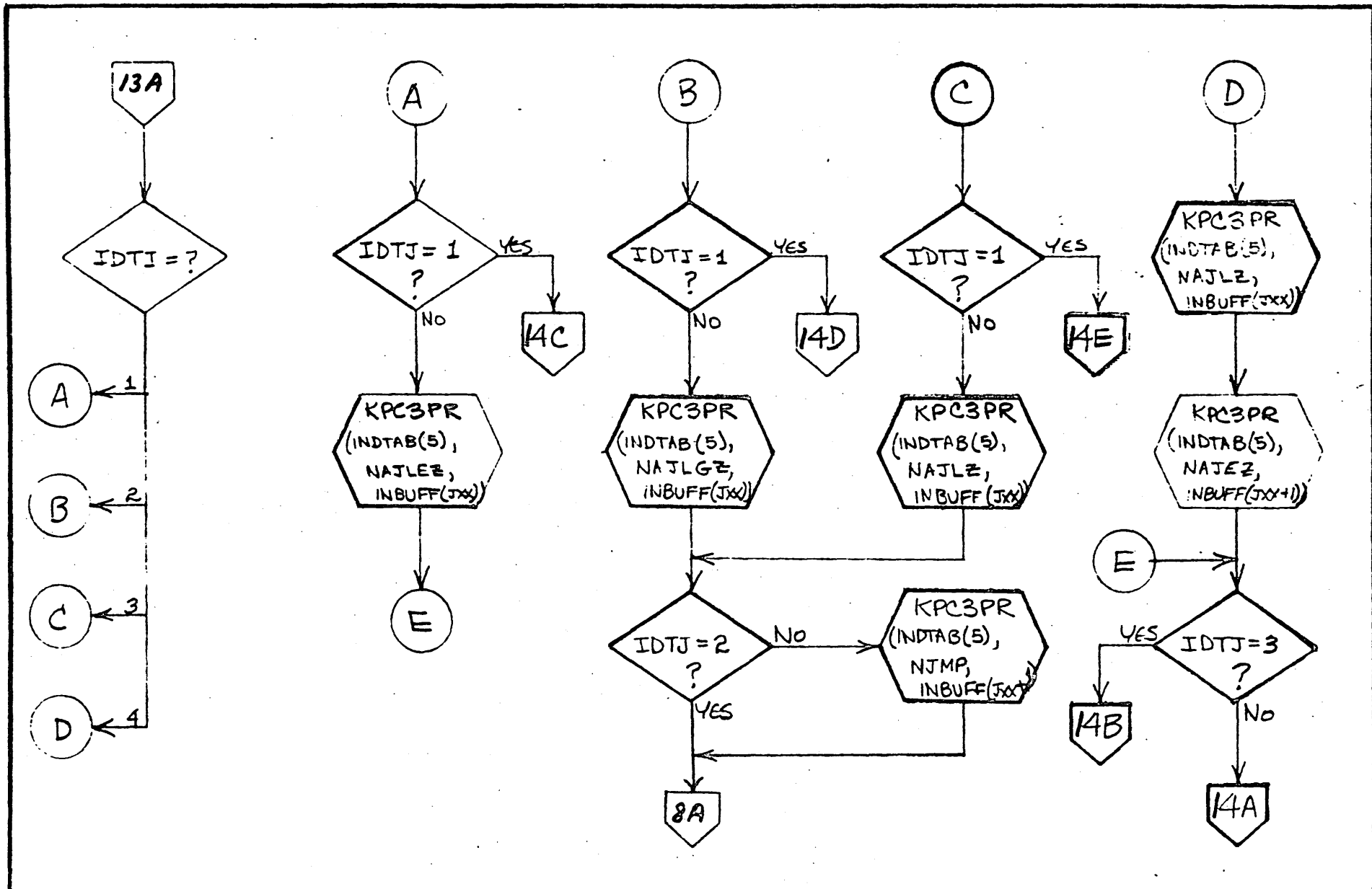


CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE   
FLOWCHART   
DECISION TABLE   
OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700
DOCUMENT TITLE	PHASE B		
NUMBER	PAGE 12 OF 20		
ISSUE DATE			
DRAWN BY	DATE 3 67		

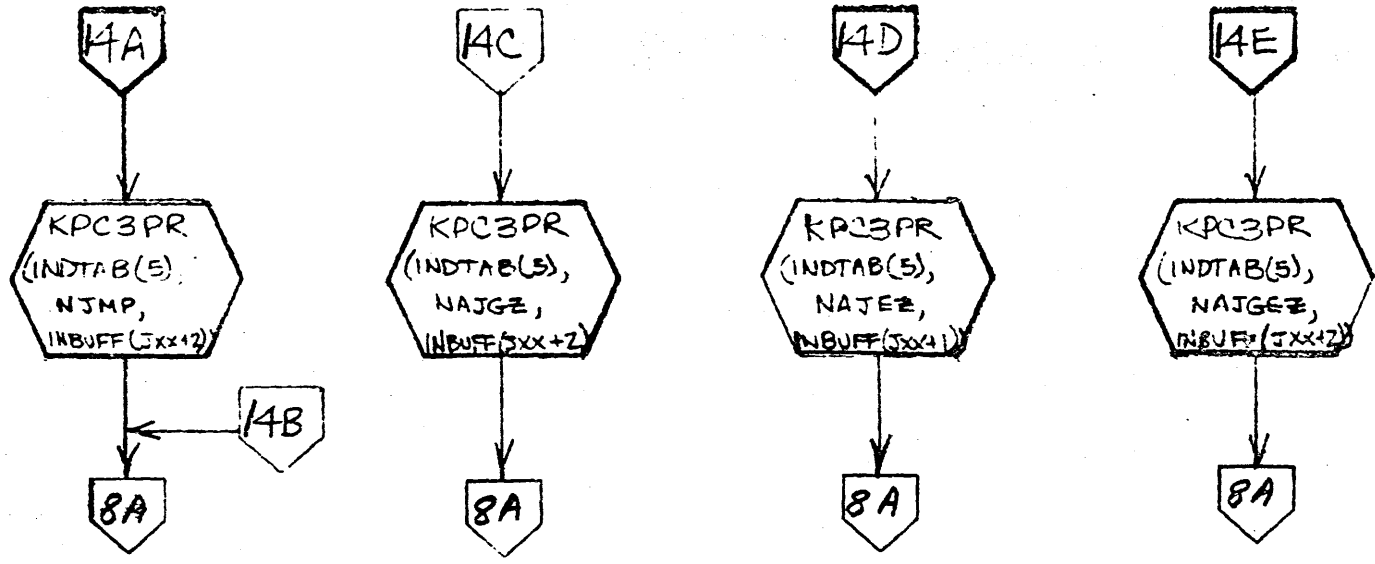
PROJECT NO.	REV	APPROVED	DATE
PROJECT MGR.			
PROJECT NAME			
TASK NO.			
TASK NAME			



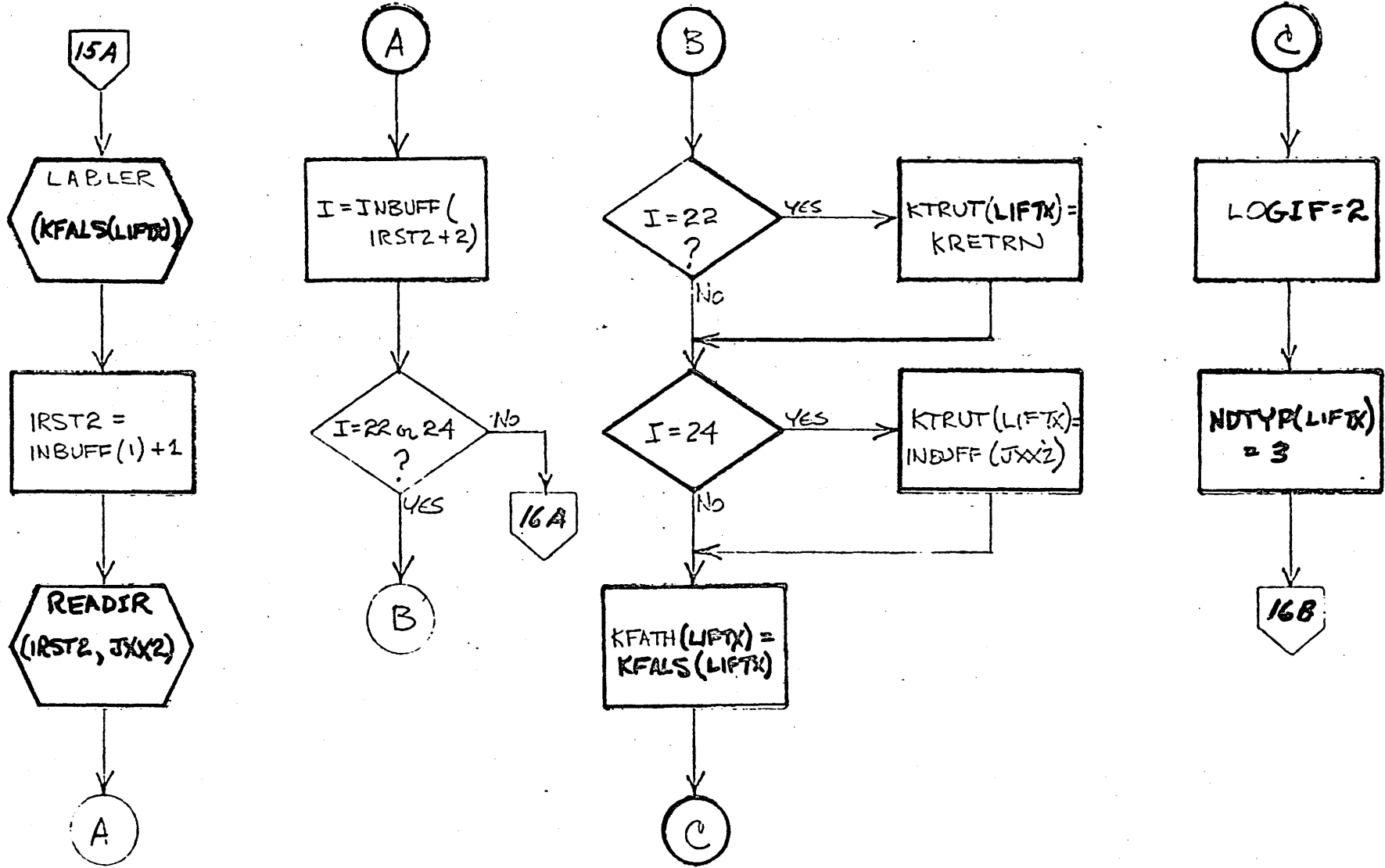
**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS <i>IMS</i>	MACH. TYPE <i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE <i>PHASE B</i>	PAGE <i>13 OF 20</i>	PROJECT MGR.			
NUMBER	ISSUE DATE	PROJECT NAME			
DRAWN BY	DATE <i>3-67</i>	TASK NO.			
		TASK NAME			



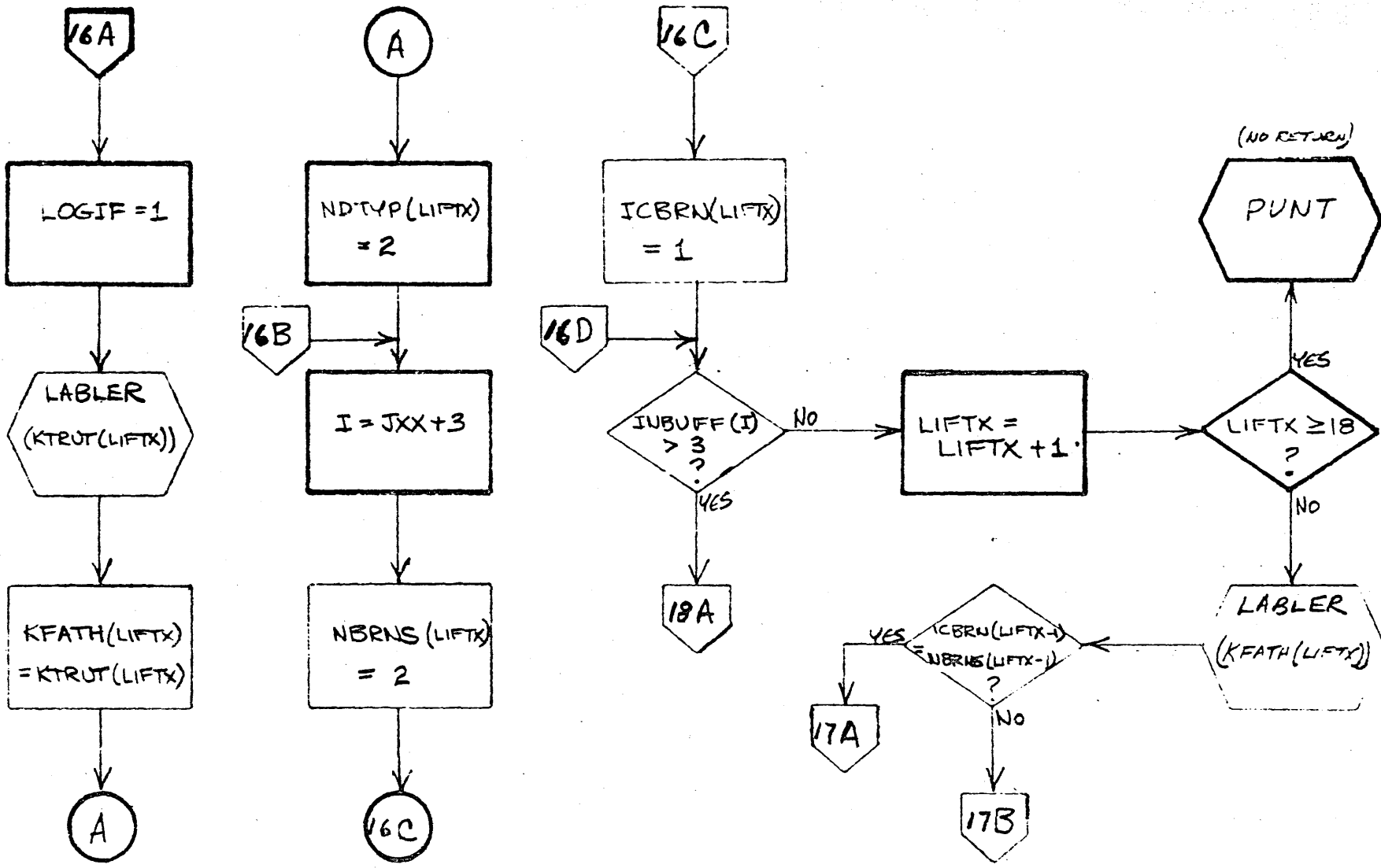
<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS <i>IMS</i>	MACH. TYPE <i>1700</i>	PROJECT NO.	RTV	APPROVED	DATE
	DOCUMENT TITLE <i>PHASE B</i>		PROJECT MGR.			
		PAGE <i>14</i> OF <i>20</i>	PROJECT NAME			
	NUMBER	ISSUE DATE	TASK NO.			
	DRAWN BY	DATE <i>3-67</i>	TASK NAME			



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	<i>IMS</i>	MACH. TYPE	<i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	<i>PHASE B</i>	PAGE <i>15</i> OF <i>20</i>		PROJECT MGR.			
NUMBER		ISSUE DATE		PROJECT NAME			
DRAWN BY		DATE	<i>2/17</i>	TASK NO.			
				TASK NAME			

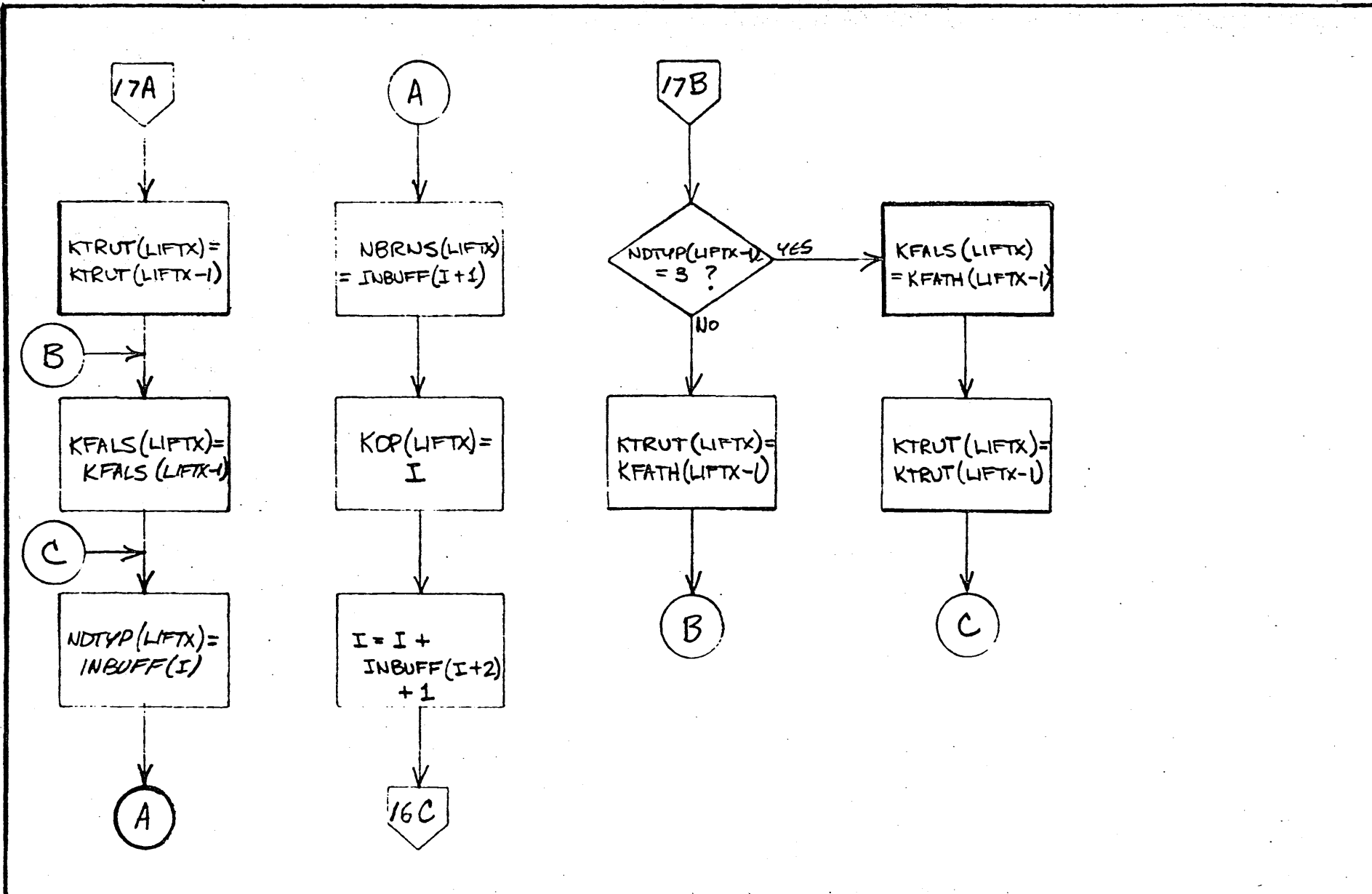


CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

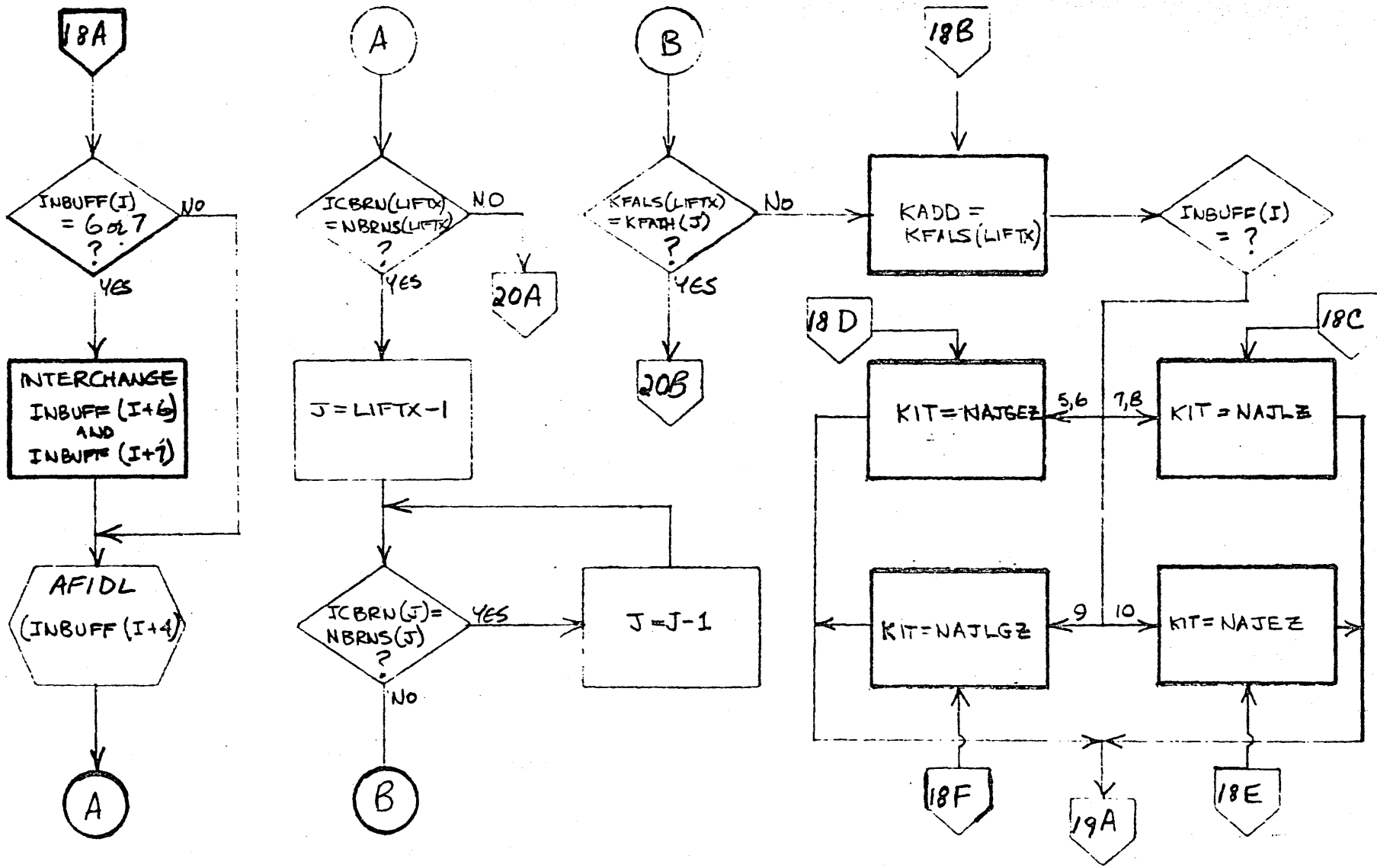
SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

DOCUMENT CLASS	<i>INS</i>	MACH. TYPE	<i>1700</i>	PROJECT NO.		REV.	APPROVED	DATE
DOCUMENT TITLE	<i>PHASE B</i>			PROJECT MGR.				
			<i>PAGE 16 OF 20</i>	PROJECT NAME				
NUMBER		ISSUE DATE		TASK NO.				
DRAWN BY		DATE	<i>3-67</i>	TASK NAME				

44-B



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS <i>DIS</i>	MACH. TYPE <i>1700</i>	PROJECT NO.	REV.	APPROVED	DATE
	DOCUMENT TITLE <i>PHASE B</i>		PROJECT MGR.			
		PAGE <i>17</i> OF <i>20</i>	PROJECT NAME			
	NUMBER	ISSUE DATE	TASK NO.			
	DRAWN BY	DATE <i>5-67</i>	TASK NAME			



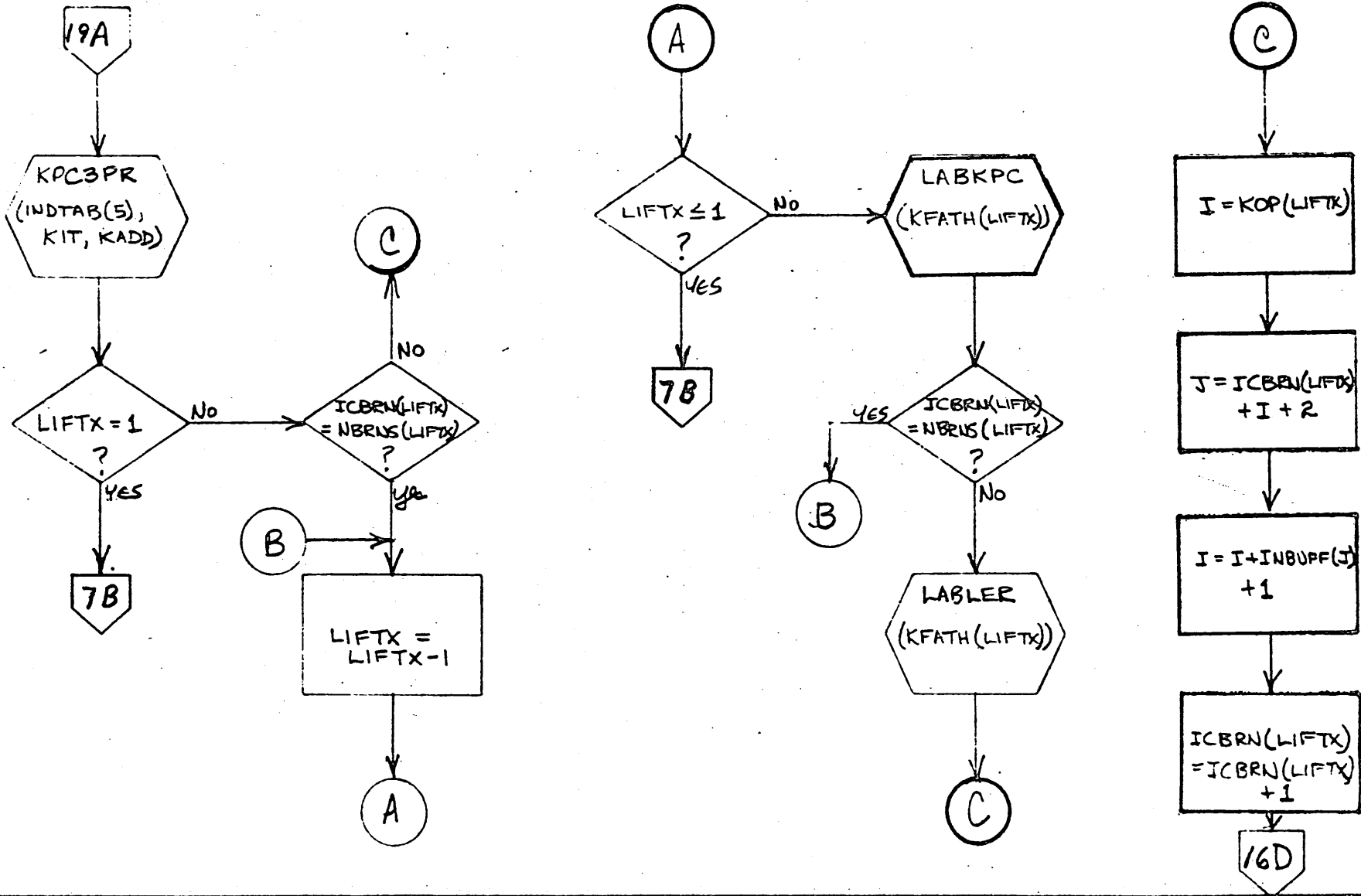
CONTROL DATA CORPORATION  
 SOFTWARE DOCUMENT

SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

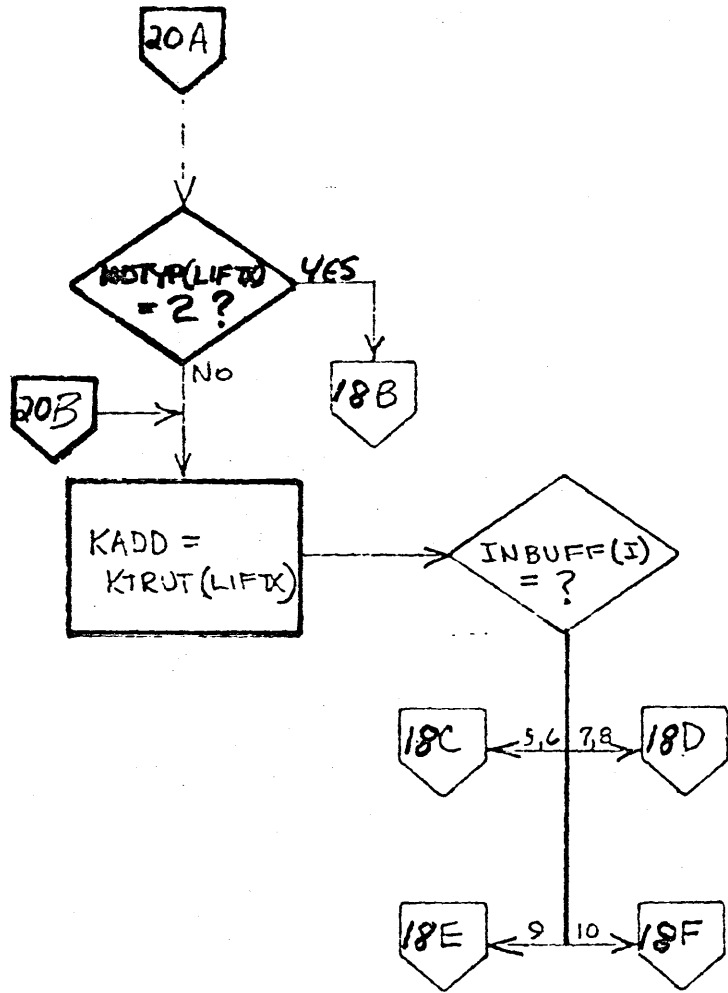
DOCUMENT CLASS	TMS	MACH. TYPE	1700
DOCUMENT TITLE	PHASE B	PAGE	18 OF 20
NUMBER	ISSUE DATE	DRAWN BY	DATE 3-67

PROJECT NO.	REV	APPROVED	DATE
PROJECT MGR.			
PROJECT NAME			
TASK NO.			
TASK NAME			





<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS <i>IAS</i>	MACH. TYPE <i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE <i>PHASE B</i>		PROJECT MGR.			
		PAGE <i>19</i> OF <i>20</i>	PROJECT NAME			
	NUMBER	ISSUE DATE	TASK NO.			
	DRAWN BY	DATE <i>3-67</i>	TASK NAME			



CONTROL DATA CONFIGURATION  
 SOFTWARE DOCUMENT  
 SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

DOCUMENT CLASS **IMS** MACH. TYPE **1700**  
 DOCUMENT TITLE **PHASE B**  
 PAGE **20** OF **20**  
 NUMBER \_\_\_\_\_ DATE \_\_\_\_\_  
 DRAWN BY \_\_\_\_\_ DATE **3-1-71**

PROJECT NO.	APPROVED	DATE
PROJECT NAME		
TASK NO.		
TASK NAME		

4.3 PHASE B STATEMENT PROCESSORS  
 4.3.1 SUBFUN

This routine processes both the SUBROUTINE and FUNCTION statements. Its primary task is to transfer the arguments of the subroutine or function subprogram (in the form of symbol table pointers) to KSUBAT, the Subroutine Argument Table. (A detailed outline of the statement formats will be found in 4.4.20) Completion of this process leaves KSUBAT in its initialized form as shown:

KSUBAT	(1)	ptr 1	symtab pointer of 1st argument
	(2)	1	pointer to next argument
	(3)	ptr 2	symtab pointer of 2nd argument
	(4)	1	pointer to next argument
		.	
		.	
		.	
		.	
	(2n-1)	ptr n	symtab pointer of nth argument
	(2n)	1	pointer to next argument
	(2n+1)	0	indicates no more arguments
		.	
		.	
		.	
		.	
		.	

The generalized form of this table is found under the discussion of subroutine interface.

In addition to transfer of arguments SUBFUN also performs the following additional operations:

DOCUMENT CLASS TMS PAGE NO 4-54  
PRODUCT NAME 1700 Mass Storage FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

1. transfers the subroutine or function name into the standard holder KPRNAM.
2. In the case of a function subprogram, creates a new holder to contain the results of the function.

Flowcharts:

SUBFUN

KPRNAM =  
INBUFF(JXX)

IS THIS A  
FUNCTION  
SUBPRG.  
?

NO

YES

RB

ISYMX =  
KPRNAM

A

A

KFUNVL =  
KPRNAM

GETSYM

I = 1

C

C

KPRNAM =  
ISYMX + I - 1

NBRNS (I) =  
ISYM(KPRNAM)

I = 2 ?

No

D

D

I = I + 1

LABLER  
(KPRNAM)

I = 1

2A

CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

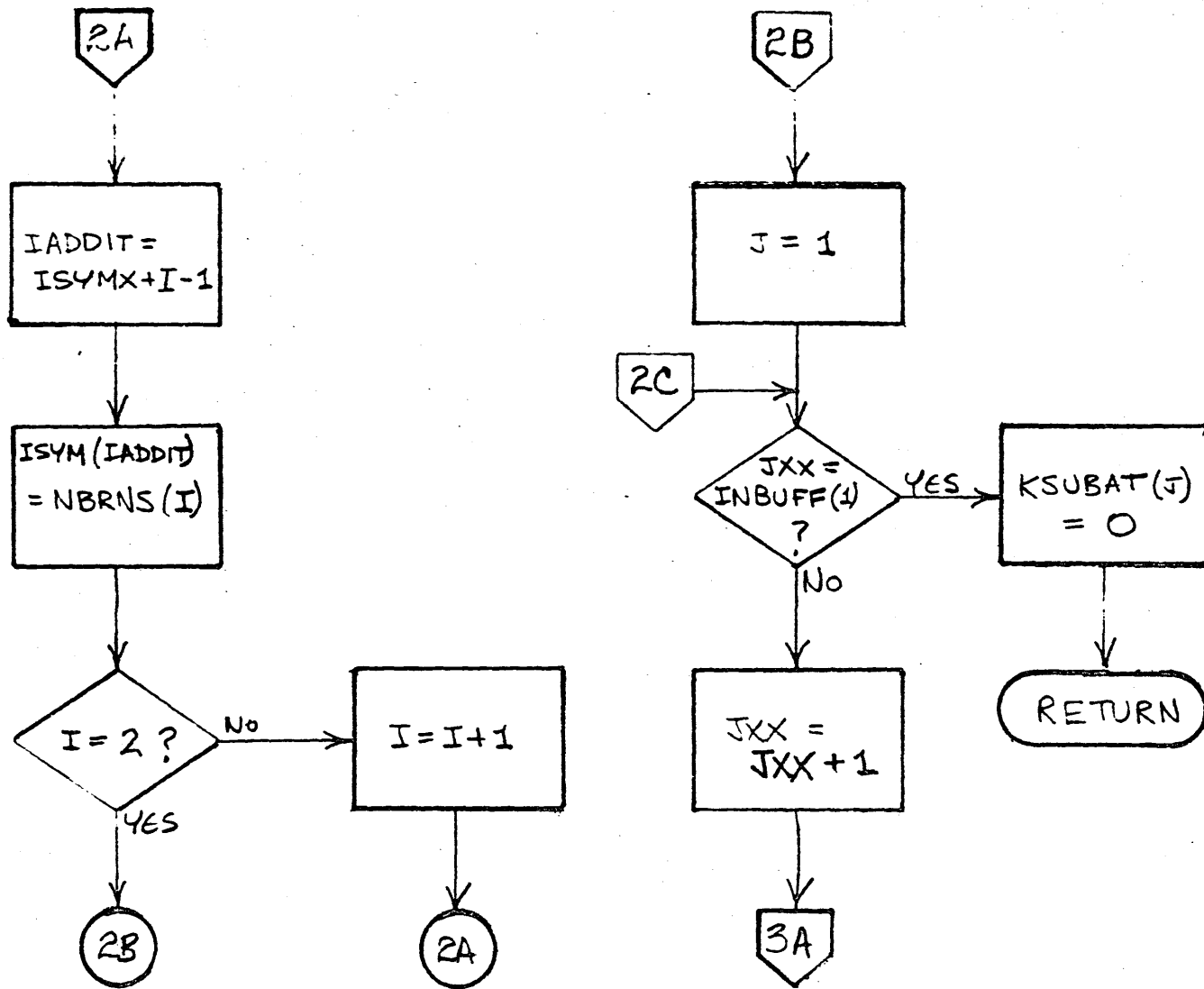
SAMPLE CODE

FLOWCHART

DECISION TABLE

OTHER

DOCUMENT CLASS <b>TMS</b>	MACH. TYPE <b>1700</b>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE <b>SUBFUN</b>		PROJECT MGR.			
PAGE <b>1</b> OF <b>3</b>		PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.			
DRAWN BY	DATE <b>3-67</b>	TASK NAME			



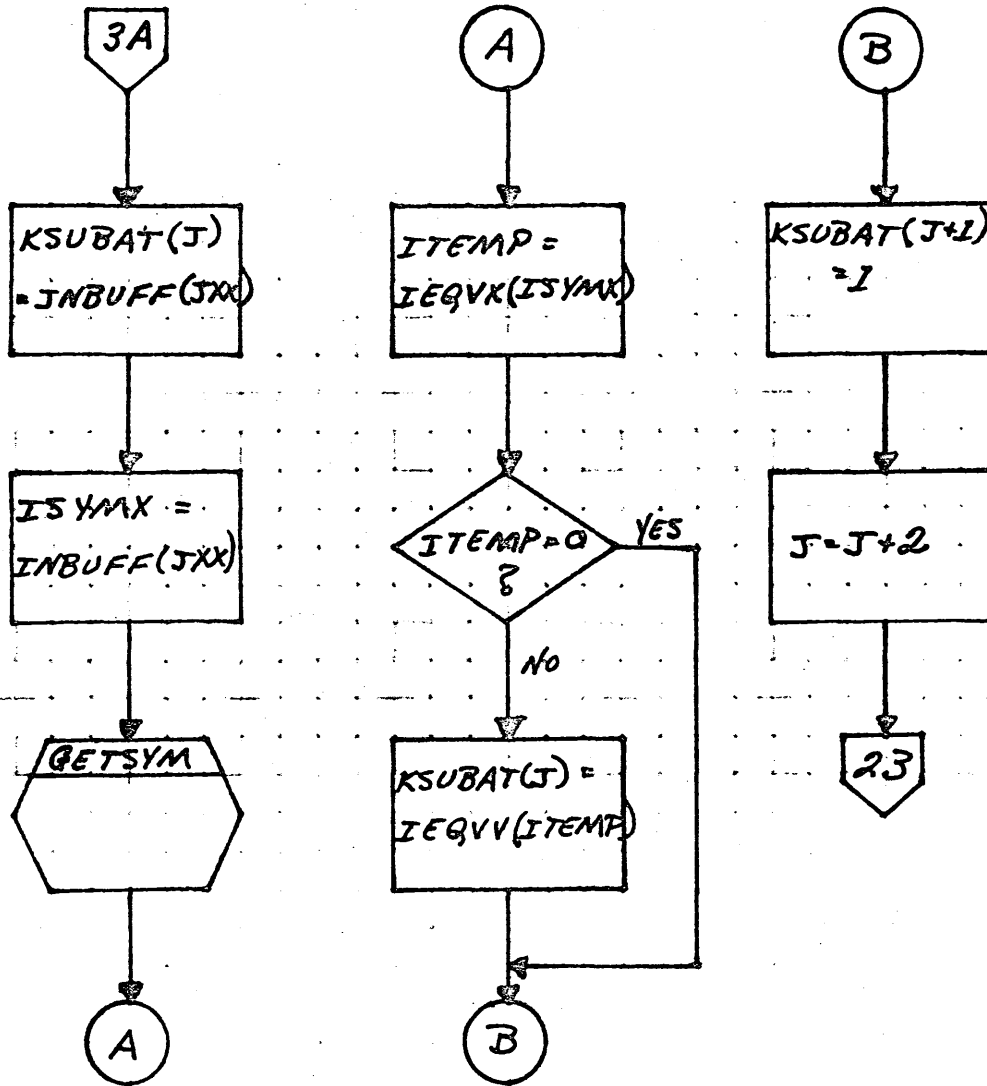
<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	MACH. TYPE	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE		PROJECT MGR.			
		PAGE 2 OF 3	PROJECT NAME			
	NUMBER	ISSUE DATE	TASK NO.			
	DRAWN BY	DATE	TASK NAME			

A

B

C

D



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	<i>IMS</i>	MACH. TYPE	<i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	<i>SUBFUN</i>			PROJECT MGR.			
PAGE <i>3</i> OF <i>3</i>				PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.					
DRAWN BY		DATE		TASK NAME			

25-4

DOCUMENT CLASS \_\_\_\_\_<sup>TM</sup> PAGE NO. 4-5B  
PRODUCT NAME 1700 Mass Storage FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

4.3.2 NOPROC

NOPROC's basic function is to transfer from the input buffer to the output file of PHASE B records of those statements which require no direct processing.

Operating is as follows:

1. If statement is a format and if the last instruction generated by PHASE B was not an unconditional jump: generate a label and output a jump to that label.
2. Output special flag record which indicates that the record following is a DATA or FORMAT (see output description).
3. Output the DATA or FORMAT record exactly as it was input.
4. If a jump around a FORMAT was generated, then output the label that was generated to jump to.

Flowcharts:

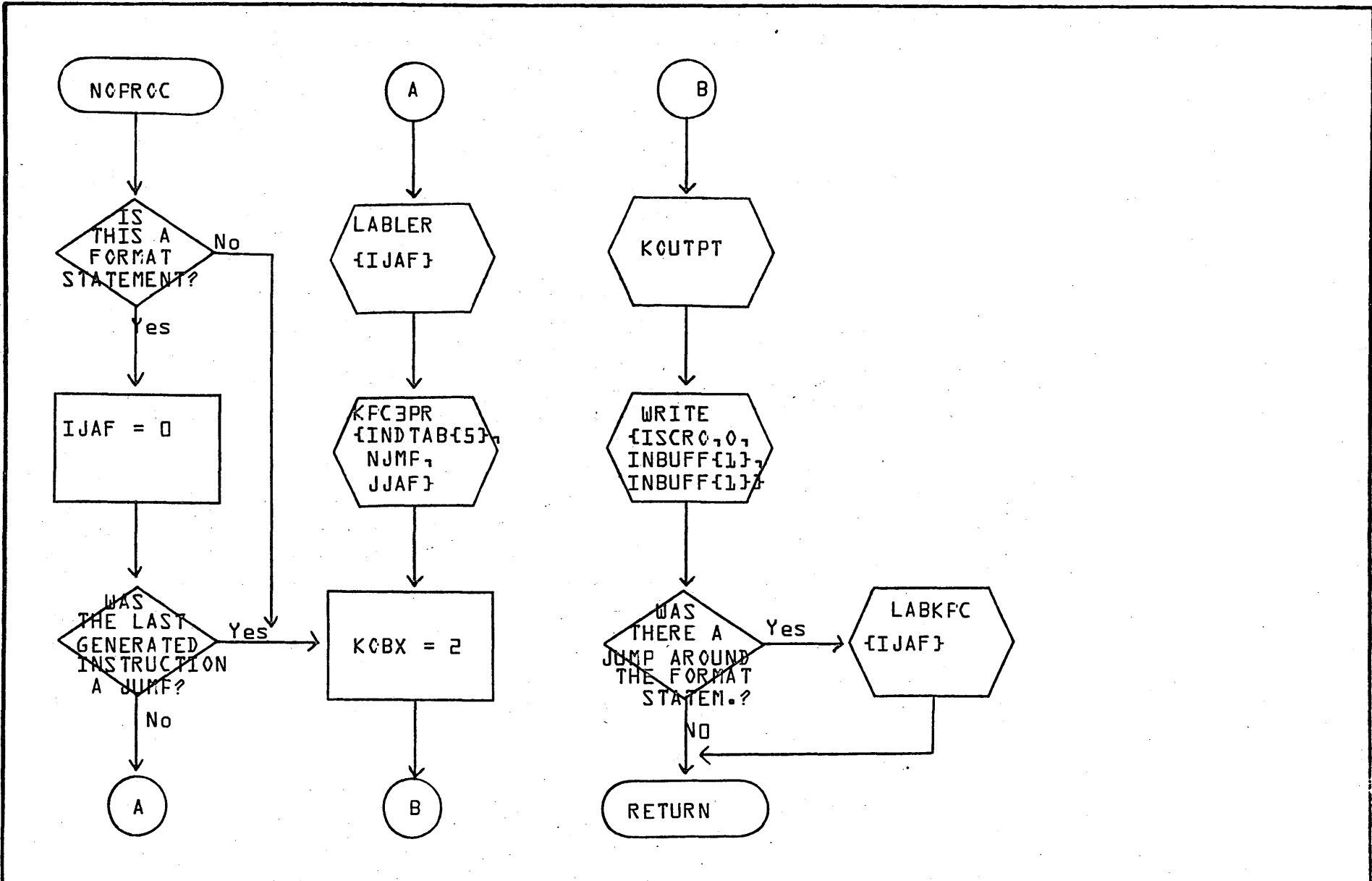


A

B

C

D



**CONTROL DATA CORPORATION**  
 SOFTWARE DOCUMENT

SAMPLE CODE

FLOWCHART

DECISION TABLE

OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	NCPROCC			PROJECT MGR.			
		PAGE	1 OF 1	PROJECT NAME			
NUMBER		ISSUE DATE		TASK NO.			
DRAWN BY		DATE		TASK NAME			

DOCUMENT CLASS IMS PAGE NO. 4-60  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

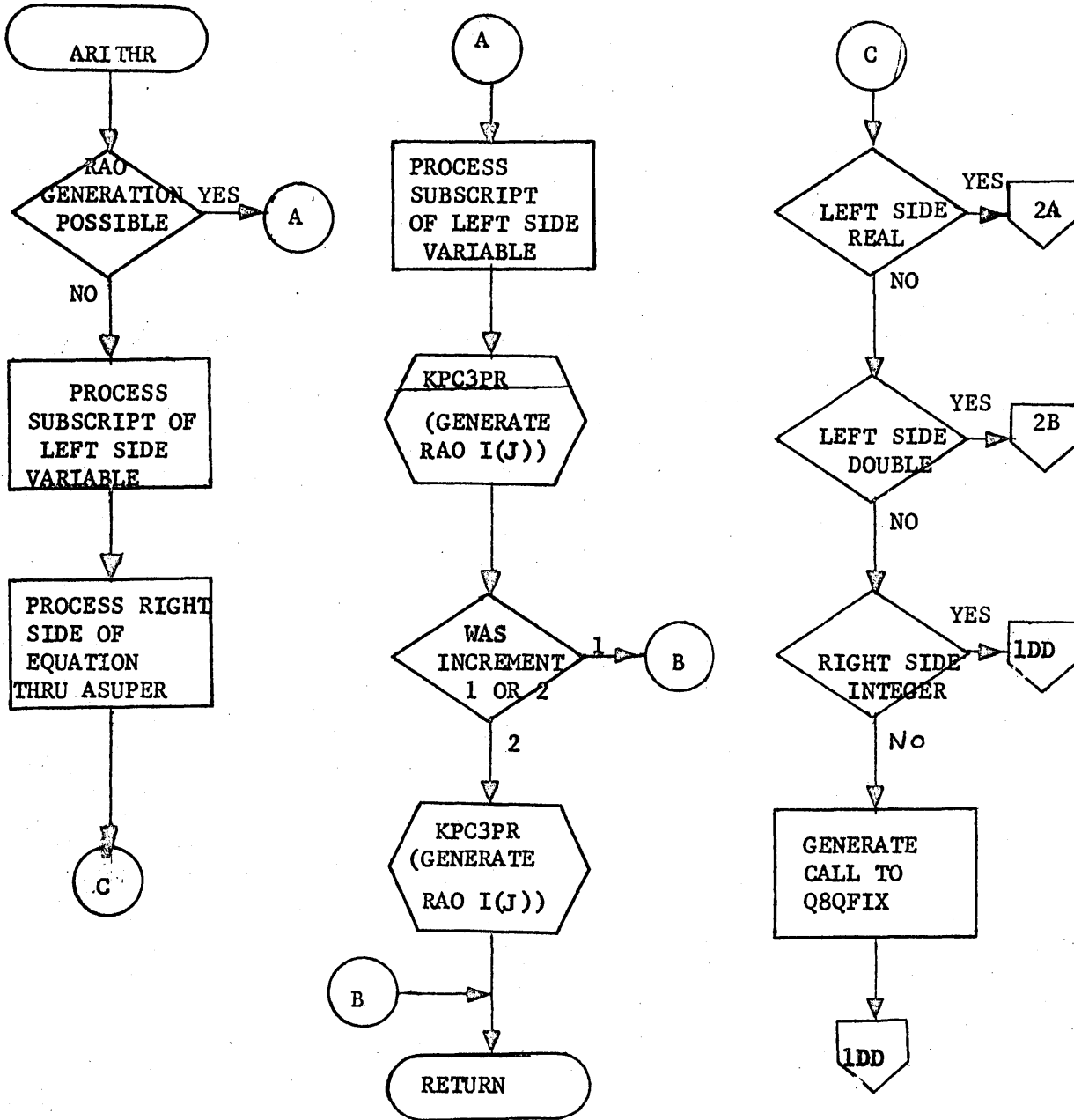
4.3.3 ARITHR

ARITHR is the routine to process the arithmetic replacement statement. It first checks for the special case  $I\{J\}=I\{J\}+1$  or 2, meaning that a non-partial variable is increased by either one or two. This allows for the use of one or two RAO commands in place of the normal sequence of commands. Otherwise the processing proceeds as follows: The subscript of the variable on the left of the equal sign is processed through the subscript expression processor SUBPR1. The variable or expression on the right side of the equal sign  $\{=\}$  is processed through the arithmetic expression processor, ASUPER. Finally, the STA, FST, or DST is generated into the left side variable preceded, if necessary, by a necessary call to a conversion routine {real to integer, integer to real, integer to double precision, etc.} whenever there is a difference between the mode of the right side expression and the left side variable. When the left side variable is a partial, code is generated to store the value into that partial.

Flowcharts:

**CONTROL DATA**

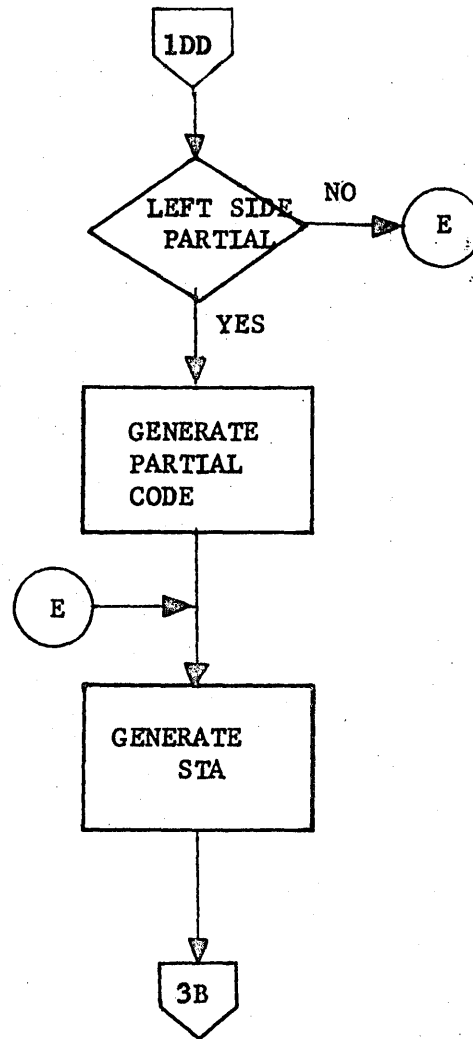
LA JOLLA RESOURCE CENTER



TITLE			ARITHR		DRG. NO.	
DRAWN BY			PROJ.		REVISION	
CSD 203 ANALYSIS			DATE		SHEET 1 OF 3	



LA JOLLA RESOURCE CENTER



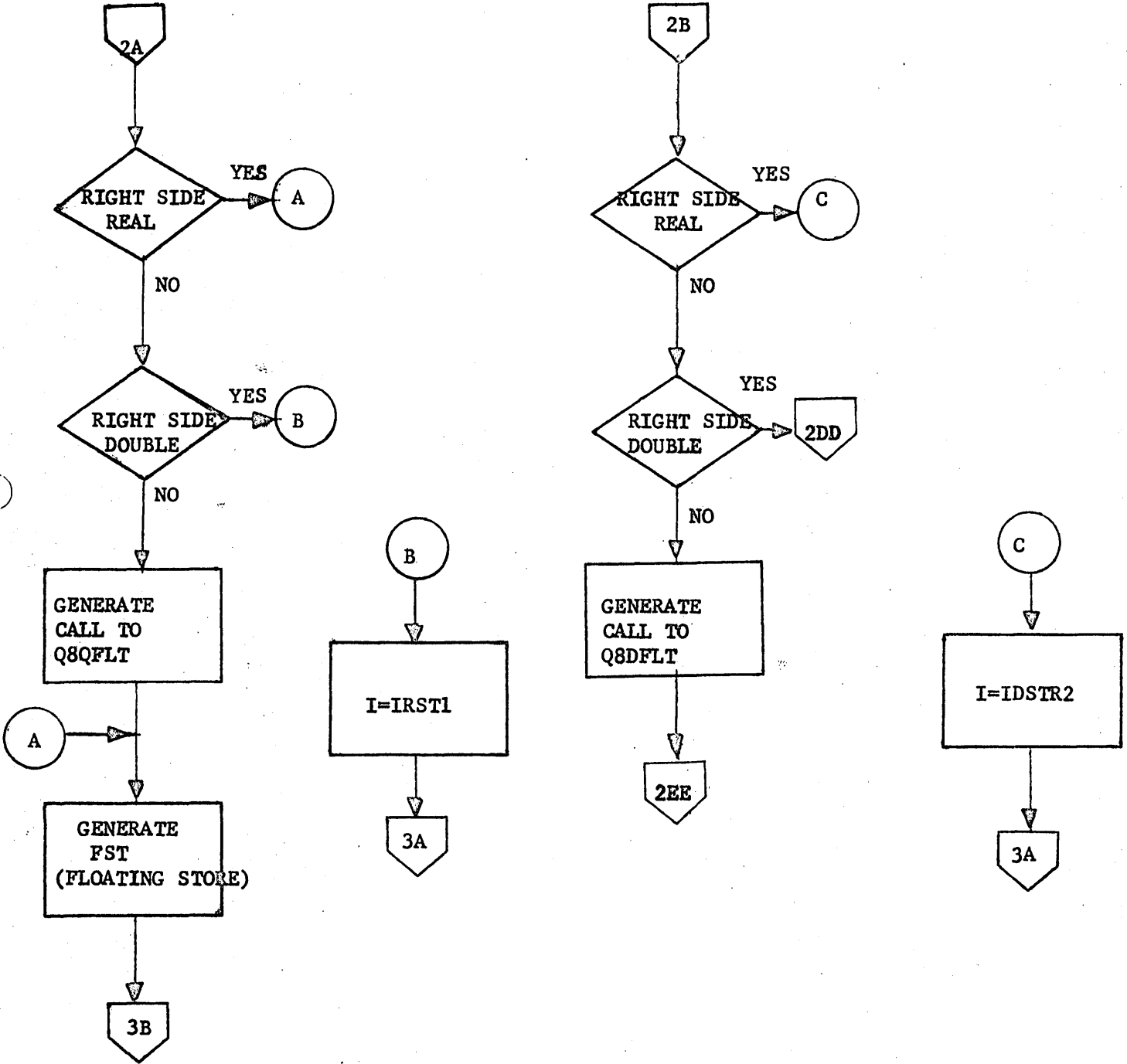
TITLE		ARITHR		DRG. NO.	
				REVISION	
DRAWN BY	PROJ.	DATE	SHEET	1A	OF 3



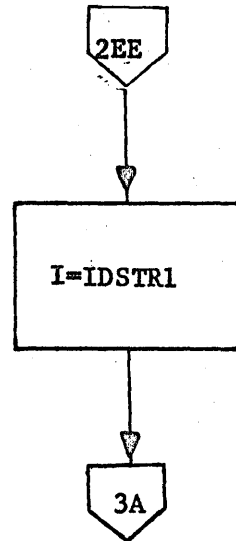
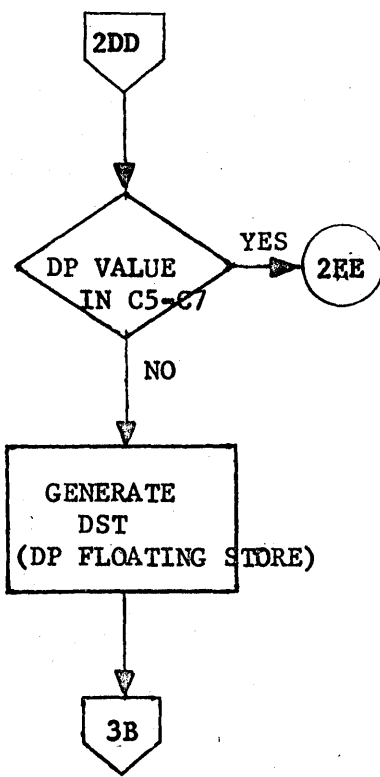
LA JOLLA RESOURCE CENTER

LEFT SIDE IS REAL

LEFT SIDE IS DOUBLE



TITLE			DRG. NO.	
ARITHR			REVISION	
DRAWN BY	PROJ.	DATE	SHEET	OF
CSD 203 ANALYSIS			2	3

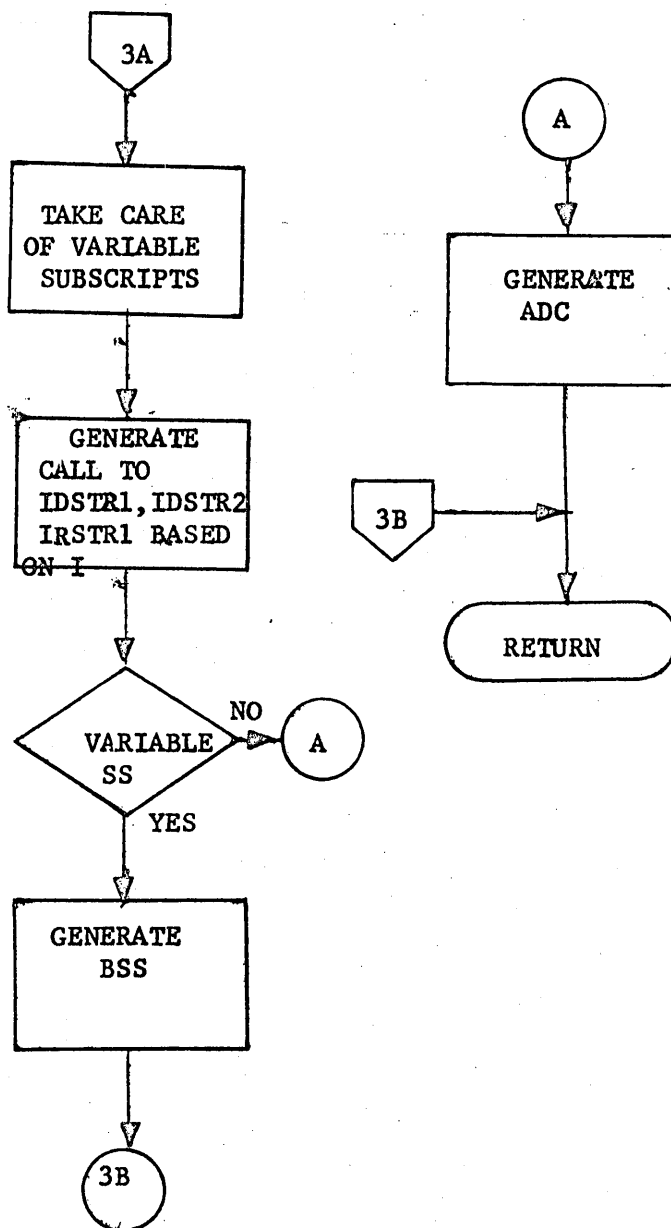


TITLE		ARITHR		DRG. NO.	
DRAWN BY		PROJ.		REVISION	
DATE		SHEET 2A		OF 3	

**CONTROL DATA**

LA JOLLA RESOURCE CENTER  
IMS Page 4-62  
1700 MASS STORAGE FORTRAN  
C005\*3.1 A/B

LA JOLLA RESOURCE CENTER



TITLE			DRG. NO.	
ARITHR			REVISION	
DRAWN BY	PROJ.	DATE	SHEET 3	OF 3





DOCUMENT CLASS IMS PAGE NO 4-63  
PRODUCT NAME 1700 Mass Storage FORTRAN  
PRODUCT NO. C005 Version 2.0 MACHINE SERIES 1700

#### 4.3.5 ~~CGTO~~

~~CGTO~~ processes the computed GO TO statement in the following manner:

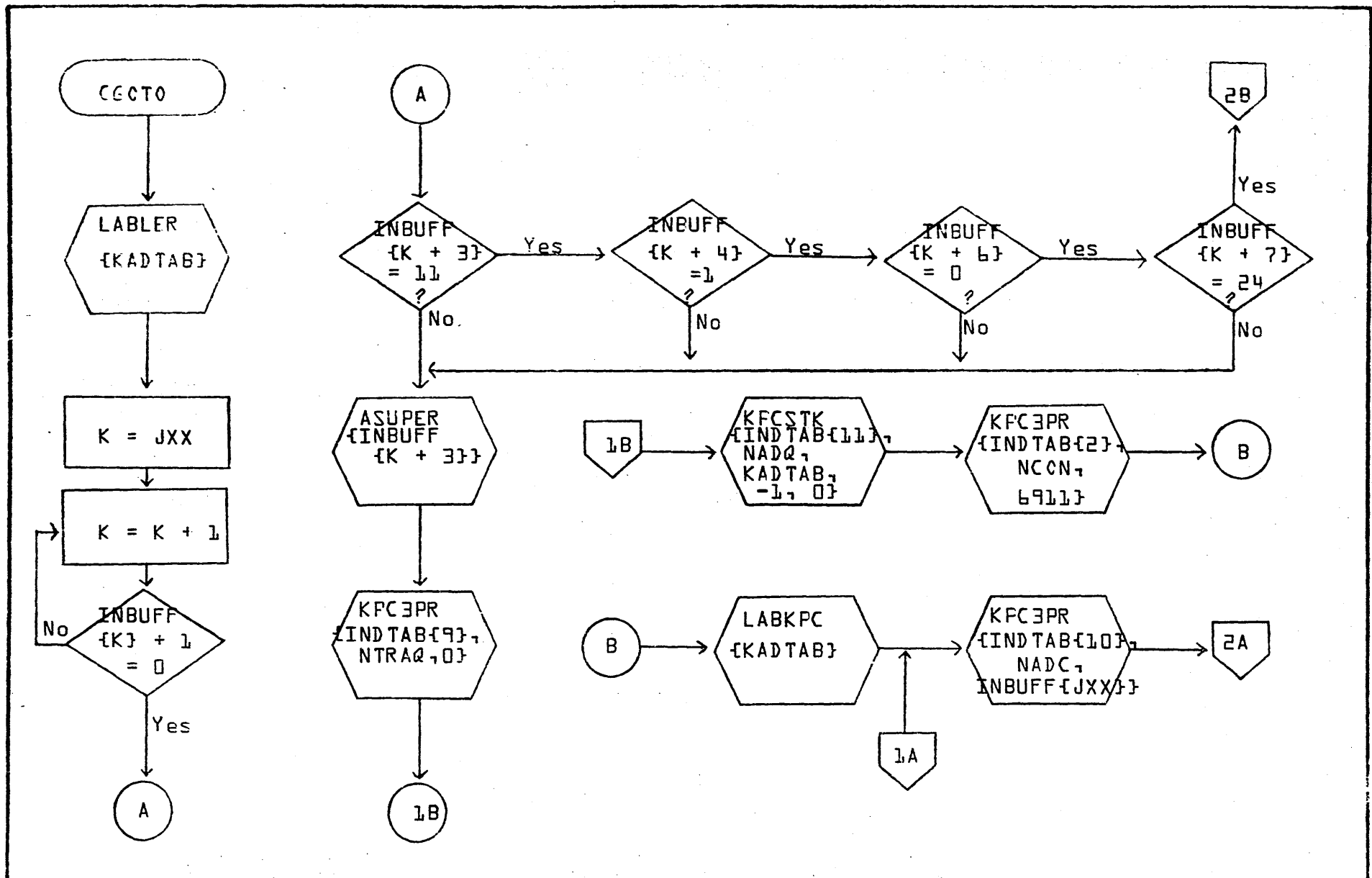
1. Creates a label to label the table of addresses in the instruction generation.
2. Tests if the GO TO expression is only a variable. If so, a LDQ command is generated. If not, the instructions are generated to compute the value of the expression followed by a TRAQ command to place the value in Q.
3. Instructions are generated to index into and jump through the proper table entry.
4. The label for the address table is generated followed by the table of addresses corresponding to the label list in the computed GO TO.

A

B

C

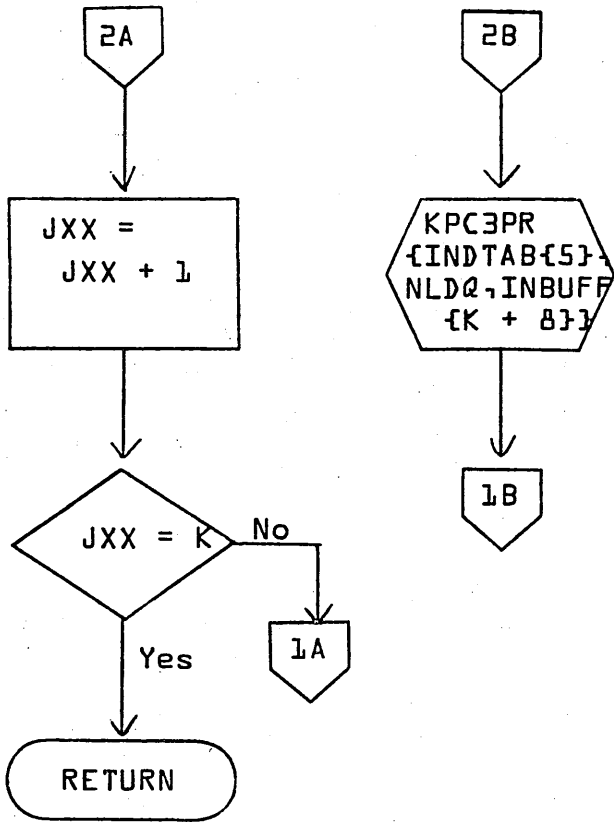
D



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE   
FLOWCHART   
DECISION TABLE   
OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	CGCTO	PAGE 1 OF 2		PROJECT MGR.			
NUMBER		ISSUE DATE		PROJECT NAME			
DRAWN BY		DATE		TASK NO.			
				TASK NAME			



**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	CGOTO	PAGE 2 OF 2		PROJECT MGR.			
NUMBER	ISSUE DATE			PROJECT NAME			
DRAWN BY	DATE			TASK NO.			
				TASK NAME			

CONTROL DATA CORPORATION  
LA JOLLA RESOURCE CENTER \_\_\_\_\_ DIVISION

DOCUMENT CLASS IMS PAGE NO. 4-66  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. COO5\*3.1 A/B MACHINE SERIES 1700

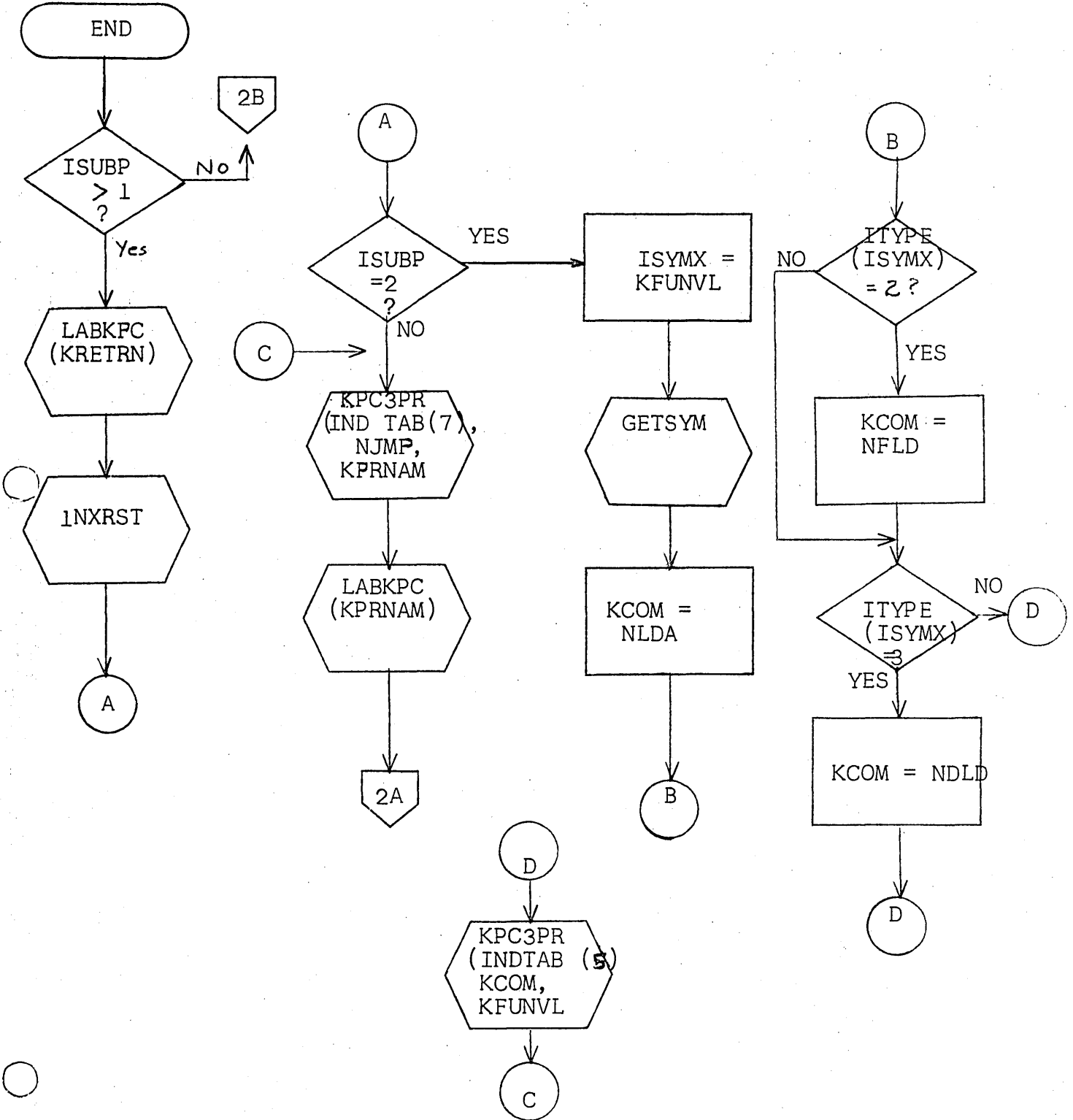
4.3.6 END

END processes the END statement in the following manner:

1. For block data and main programs, the "end" pseudo-instruction is generated signifying the end of the Phase B output and Phase C input.
2. For subroutines and function subprograms, the label of reference for all return statements is output followed by instruction generation of instructions to restore index registers.
3. For function subprograms, another instruction is then generated to load the value of the function {either LDA, FLD, or DLD}.
4. For subroutines and function subprograms, then a jump command through the entry point is generated followed by output of the entry point label. Then the parameters are set up and ENTCOD is executed to generate the parameter pickup instruction sequence. Then the "end" is generated as in step 1.
5. Finally the SYMTAB points to the program name is transferred to holder for use by Phase C.

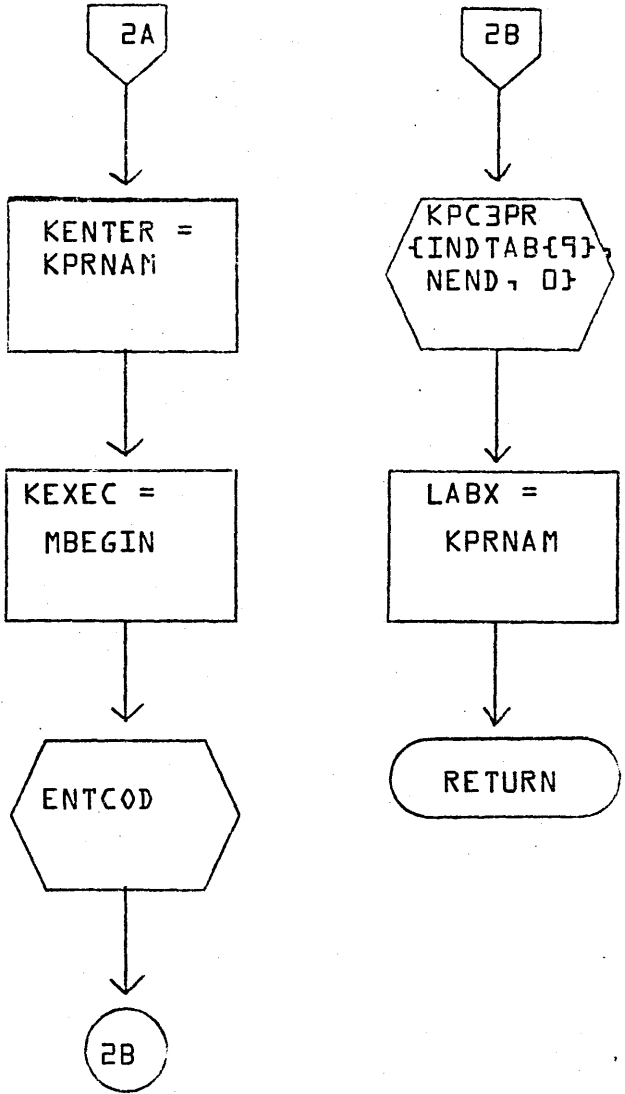


LA JOLLA RESOURCE CENTER



TITLE			END		DRG. NO.	
DRAWN BY			PROJ.		REVISION	
DATE			SHEET		1 OF 2	
CSD 203 ANALYSIS						

A  
B  
C  
D



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	END	PAGE 2 OF 2		PROJECT MGR.			
	NUMBER		ISSUE DATE		TASK NO.			
	DRAWN BY		DATE		TASK NAME			

8-5-4

DOCUMENT CLASS \_\_\_\_\_ IMS \_\_\_\_\_ PAGE NO. 4-69  
PRODUCT NAME 1700 Mass Storage FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

#### 4.3.7 BGINDØ and BANANA

DØ statements are reorganized by Phase 1 into two separate statements Begin Do and End Do processed in Phase 4 by BGINDO and BANANA respectively. Detailed descriptions of the philosophy and processing of the DO statement can be found in Phase 1.

The Begin Do occurs physically in the same position in the Phase 4 input file as the DO statement does in the FORTRAN source input. The END DØ corresponding to a BEGIN DØ follows the terminal statement of the associated DØ and precedes the next statement. When more than one DO statement has the same terminal statement the End Do's will occur consecutively in the output file, the first one corresponding to the innermost Begin Do, etc.

Both BGINDO and BANANA generate executable instructions which work together to perform the loop. In generating executable instructions both core space and execution time was taken into consideration. Several factors enter into the analysis:

1. Class of the initial parameter (i.e. constant or variable).
2. Class of terminal parameter.
3. Class of the incrementation parameter.
4. Whether the loop is an incrementing or decrementing loop.
5. In certain cases, whether or not the initial parameter is a constant of 1.
6. In certain cases, whether or not the terminal parameter is a constant of 1.
7. In certain cases, whether or not the incrementation parameter is a constant of 1.

In the vast majority of cases the number of times a loop is executed is a function of the loop parameters (initial, terminal, and incrementation values). If  $n$  represents this number then the number of times the terminal value must be compared with the control variable is:





DOCUMENT CLASS IMS PAGE NO. 4-71  
PRODUCT NAME 1700 Mass Storage FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

Both tables are indexed by KLLTBX. BGINDO adds an entry and updates index KLLTBX for each Begin Do encountered. BANANA uses the information in an entry and decrements KLLTBX for each End Do encountered. The first word of the KLLTB entry is used to contain the symtab pointer to a generated label. This label is created by BGINDO to reference the end of a loop (i.e. the instruction to be executed when the loop has been satisfied). This label is only needed (and thus only created) when testing is done at the beginning of the loop. The second word of the KLLTB entry is used to contain the symtab pointed to the top-of-loop label, i.e. the label of the first instruction to be operated whenever the loop range is executed again. LPTYP contains a code that BGINDO uses to communicate with BANANA and designate what instructions should be generated at the end of the loop.

Flowcharts:

A

B

C

D

BGINDO

KSWIT = 0

J =  
INBUFF{JXX}

KENDL1 =  
LFE{J}

A

A

KENDL1 =  
LLABL2  
{KENDL1}

KLLTB  
{KLLTBX,2}  
= KENDL1

KLLTBX =  
KLLTBX = 1

B

B

ISYMX =  
LBEG{J}

GETSYM

ICLASS  
{ISYMX}  
= 1?

2A

KSWIT =  
KSWIT + 1

NOTE: Closed subroutines internal to the program are signified by numeric names and their flowcharts are found within the program flowcharts

CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

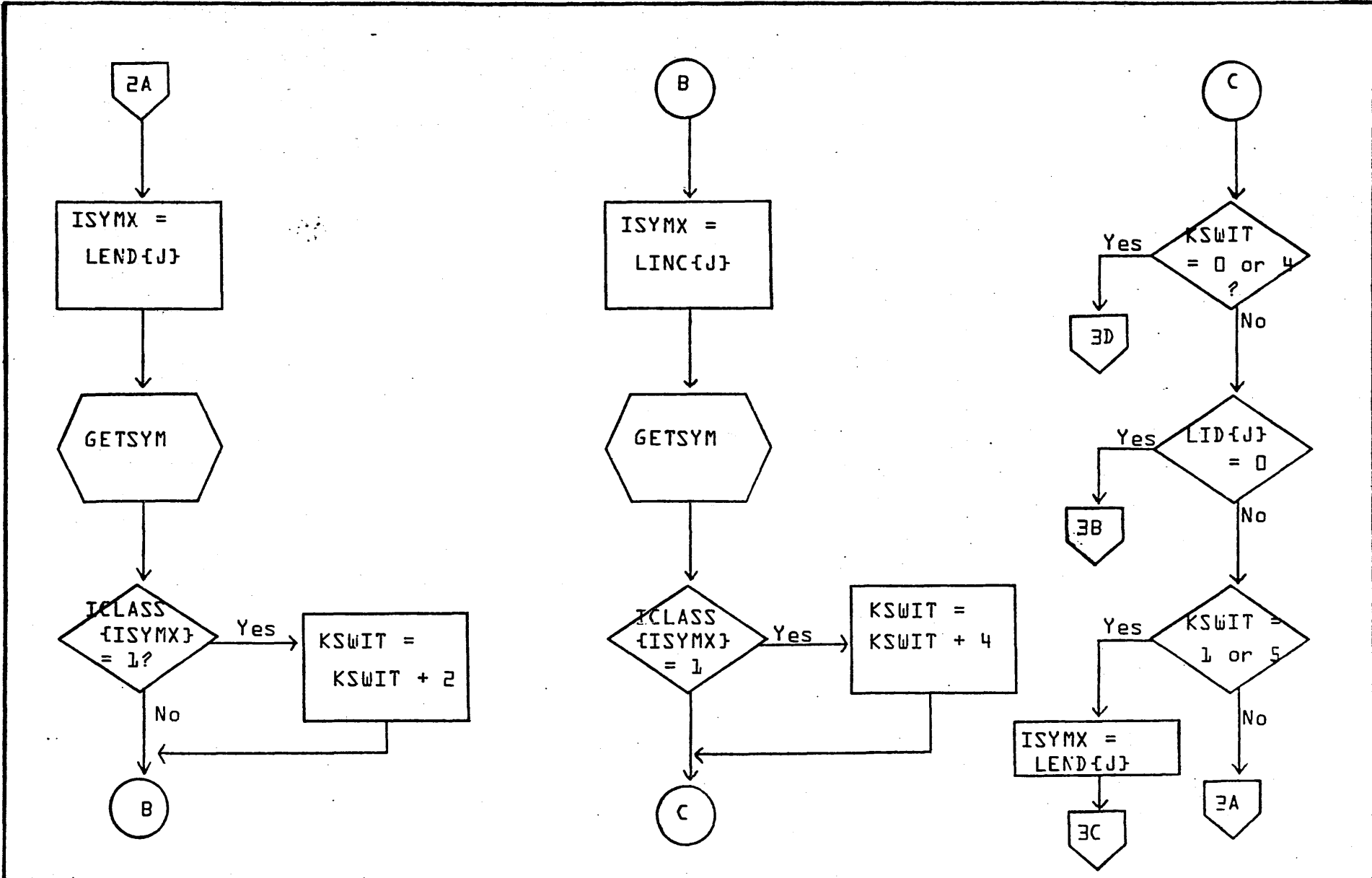
DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	BGINDO	PAGE 1 OF 5		PROJECT MGR.			
NUMBER	ISSUE DATE	DATE		PROJECT NAME			
DRAWN BY	DATE	DATE		TASK NO.			
		DATE		TASK NAME			

A

B

C

D

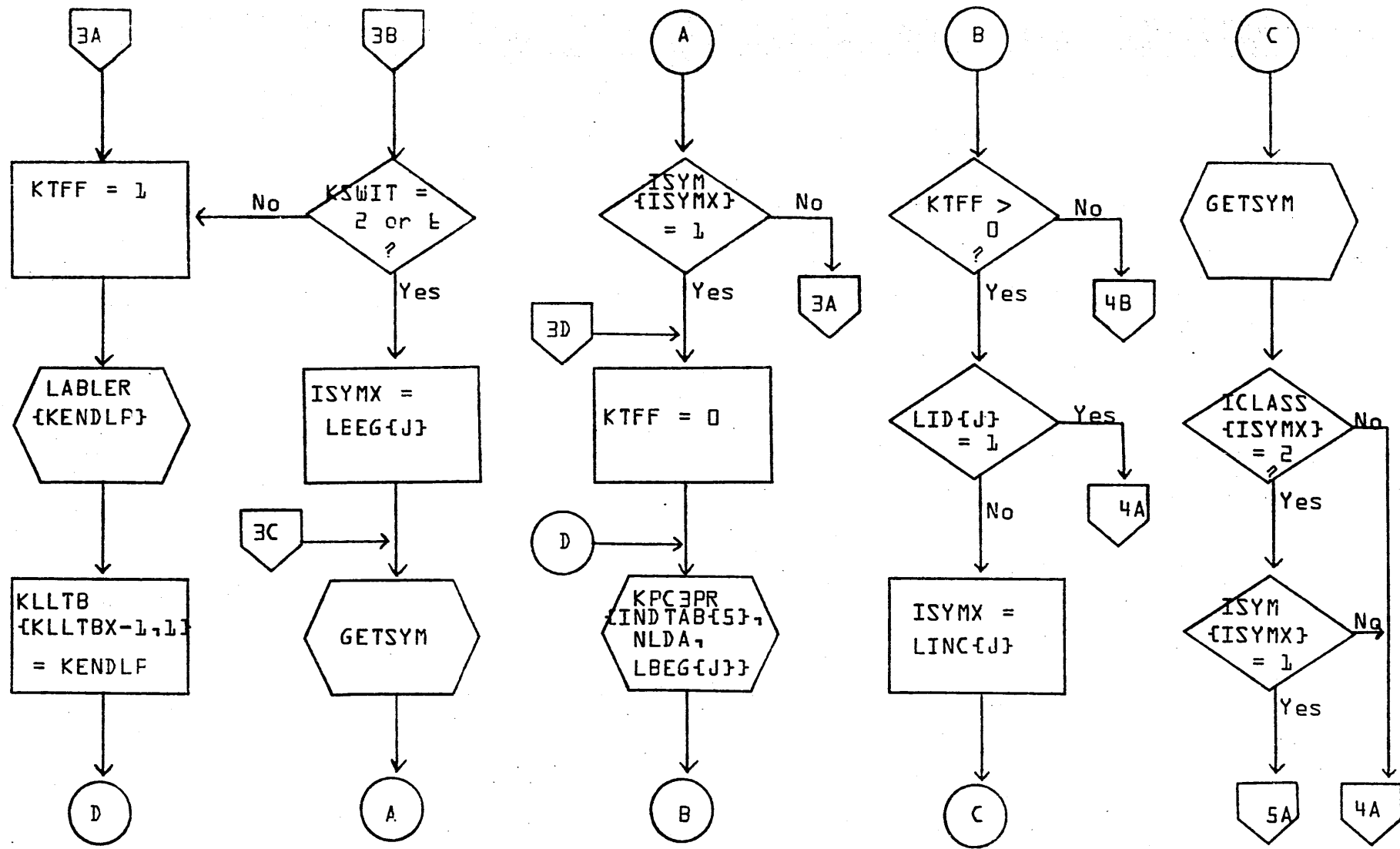


**CONTROL DATA CORPORATION**  
 SOFTWARE DOCUMENT

SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	BGINDO	PAGE 2 OF 5		PROJECT MGR.			
NUMBER	ISSUE DATE			PROJECT NAME			
DRAWN BY	DATE			TASK NO.			
				TASK NAME			

A  
B  
C  
D



**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	BGINDO	PAGE 3 OF 5		PROJECT MGR.			
NUMBER	ISSUE DATE	TASK NO.		TASK NAME			
DRAWN BY	DATE						

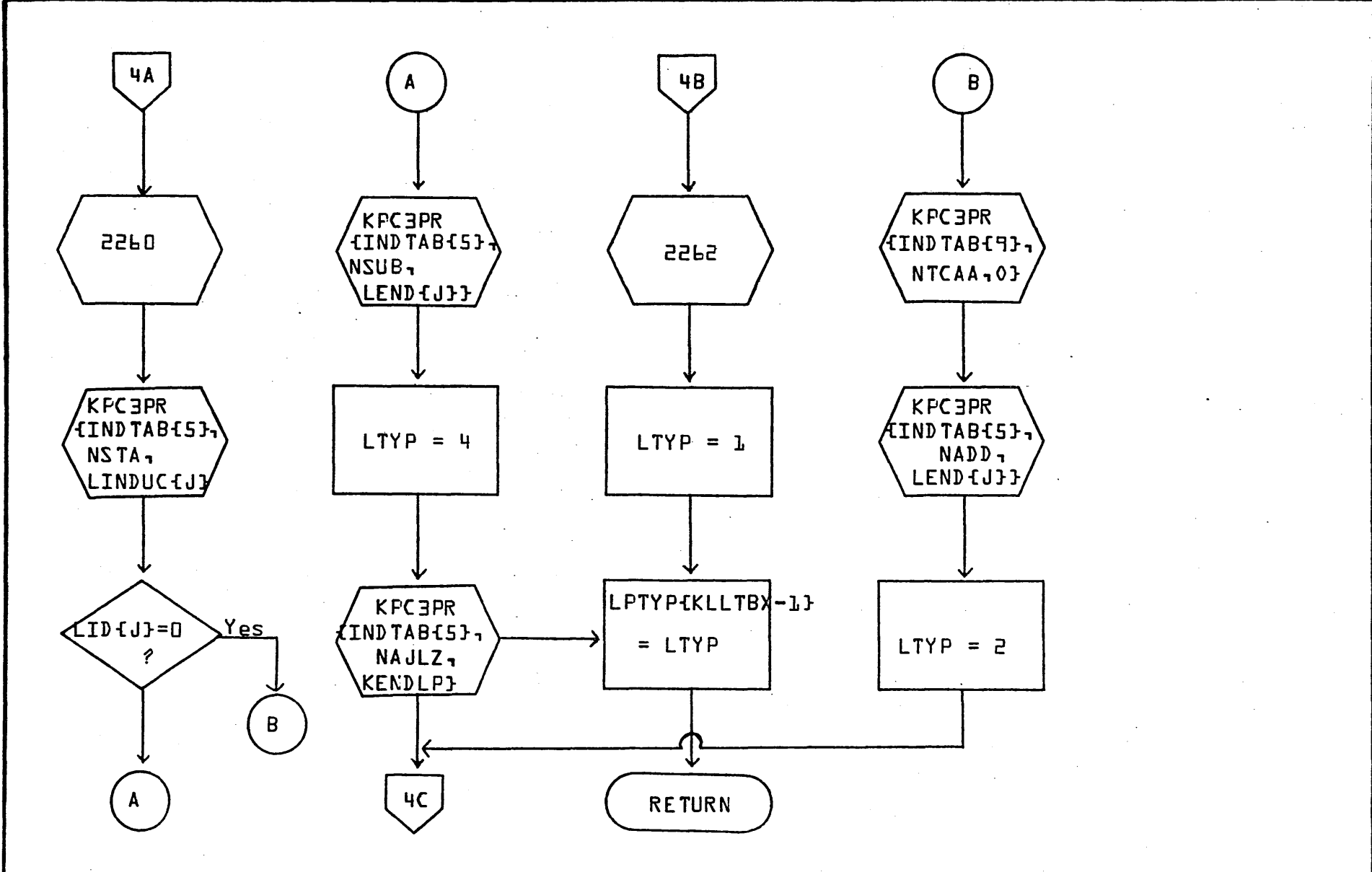
4-74

A

B

C

D



**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

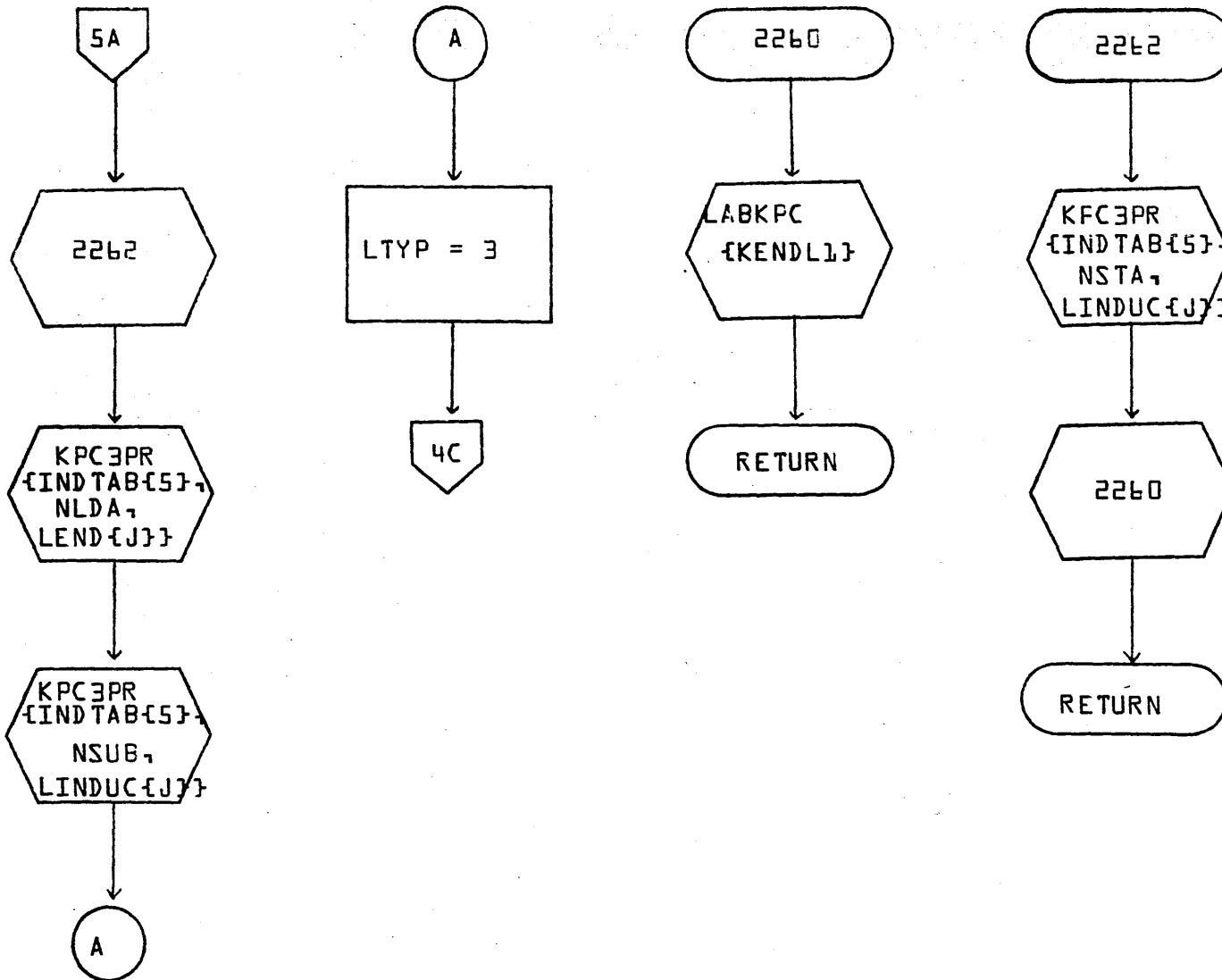
DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	BGINDO			PROJECT MGR.			
PAGE 4 OF 5				PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.					
DRAWN BY		DATE		TASK NAME			

A

B

C

D



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	TMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	BGINDO			PROJECT MGR.			
	PAGE 5 OF 5				PROJECT NAME			
	NUMBER	ISSUE DATE			TASK NO.			
	DRAWN BY		DATE		TASK NAME			

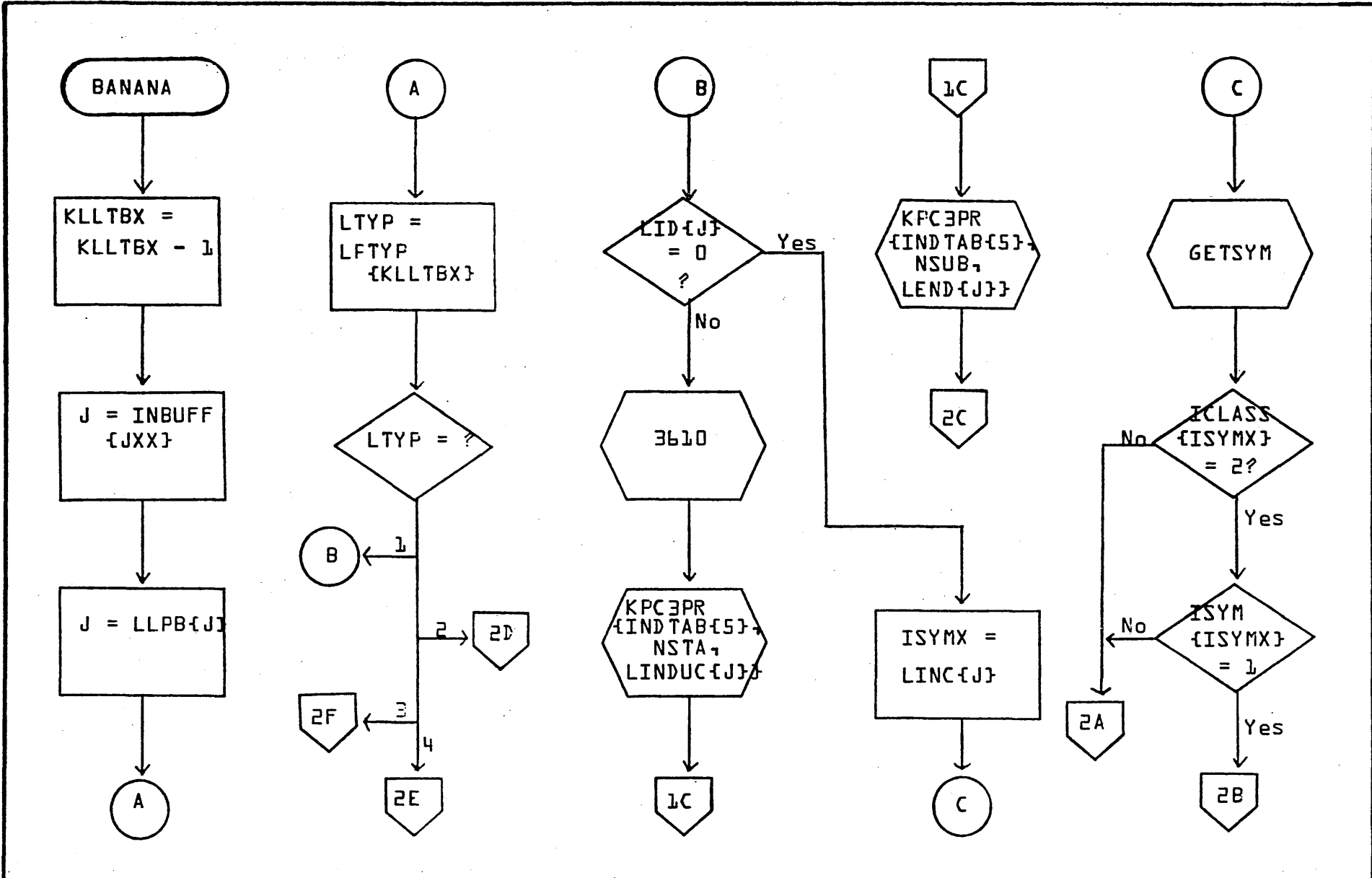
4-75

A

B

C

D



**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

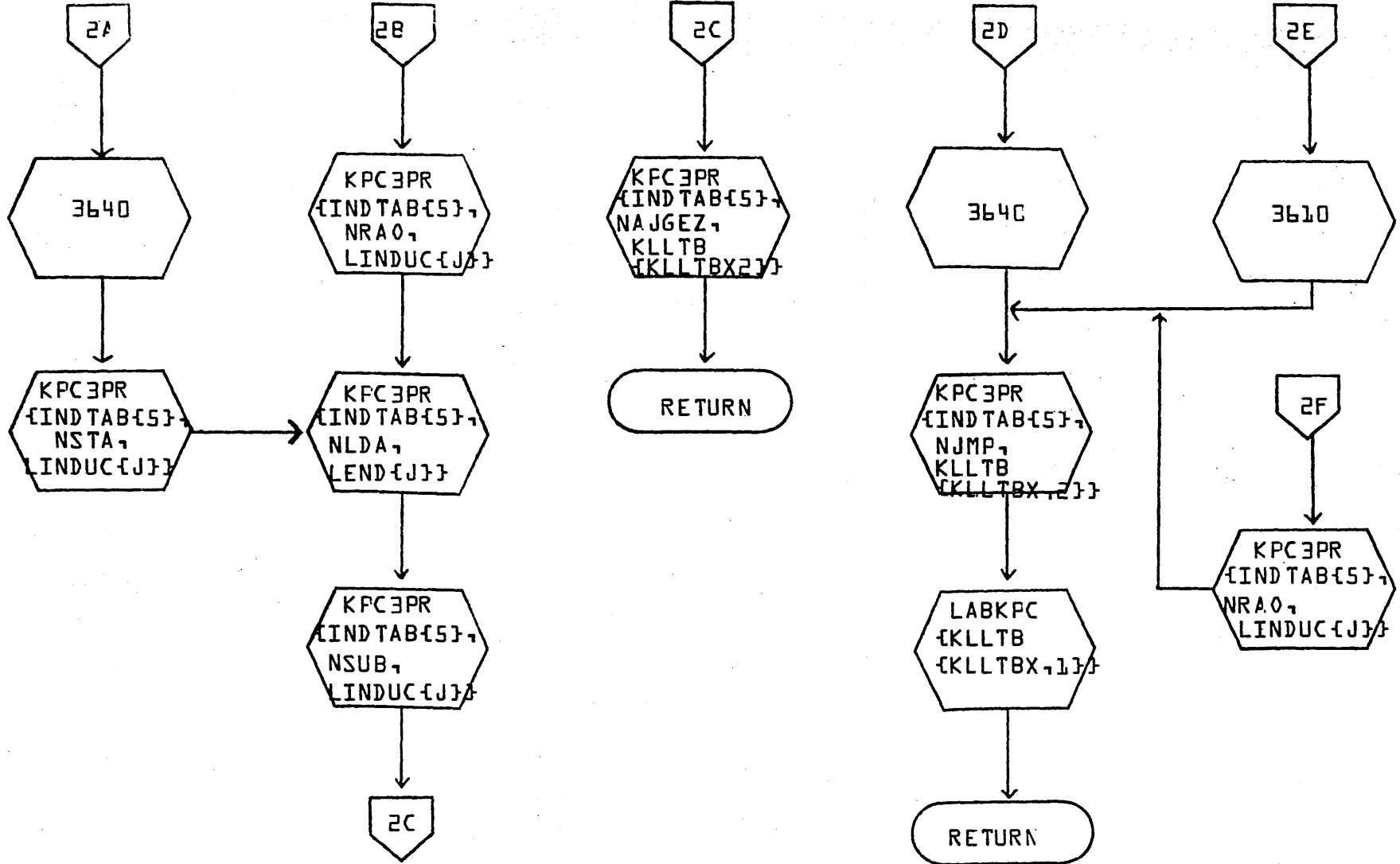
DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	BANANA			PROJECT MGR.			
PAGE 1 OF 3				PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.					
DRAWN BY		DATE		TASK NAME			

A

B

C

D



**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

SAMPLE CODE

FLOWCHART

DECISION TABLE

OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	BANANA			PROJECT MGR.			
PAGE 2 OF 3				PROJECT NAME			
NUMBER	ISSUE DATE		TASK NO.				
DRAWN BY	DATE		TASK NAME				



A

B

C

D

3610

3640

KPC3PR  
{INDTAB{5},  
NLDA,  
LINDUC{J}}

KPC3PR  
{INDTAB{5},  
NLDA,  
LINDUC{J}}

KPC3PR  
{INDTAB{5},  
NSUB,  
LINC{J}}

KPC3PR  
{INDTAB{5},  
NADD,  
LINC{J}}

RETURN

RETURN

CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

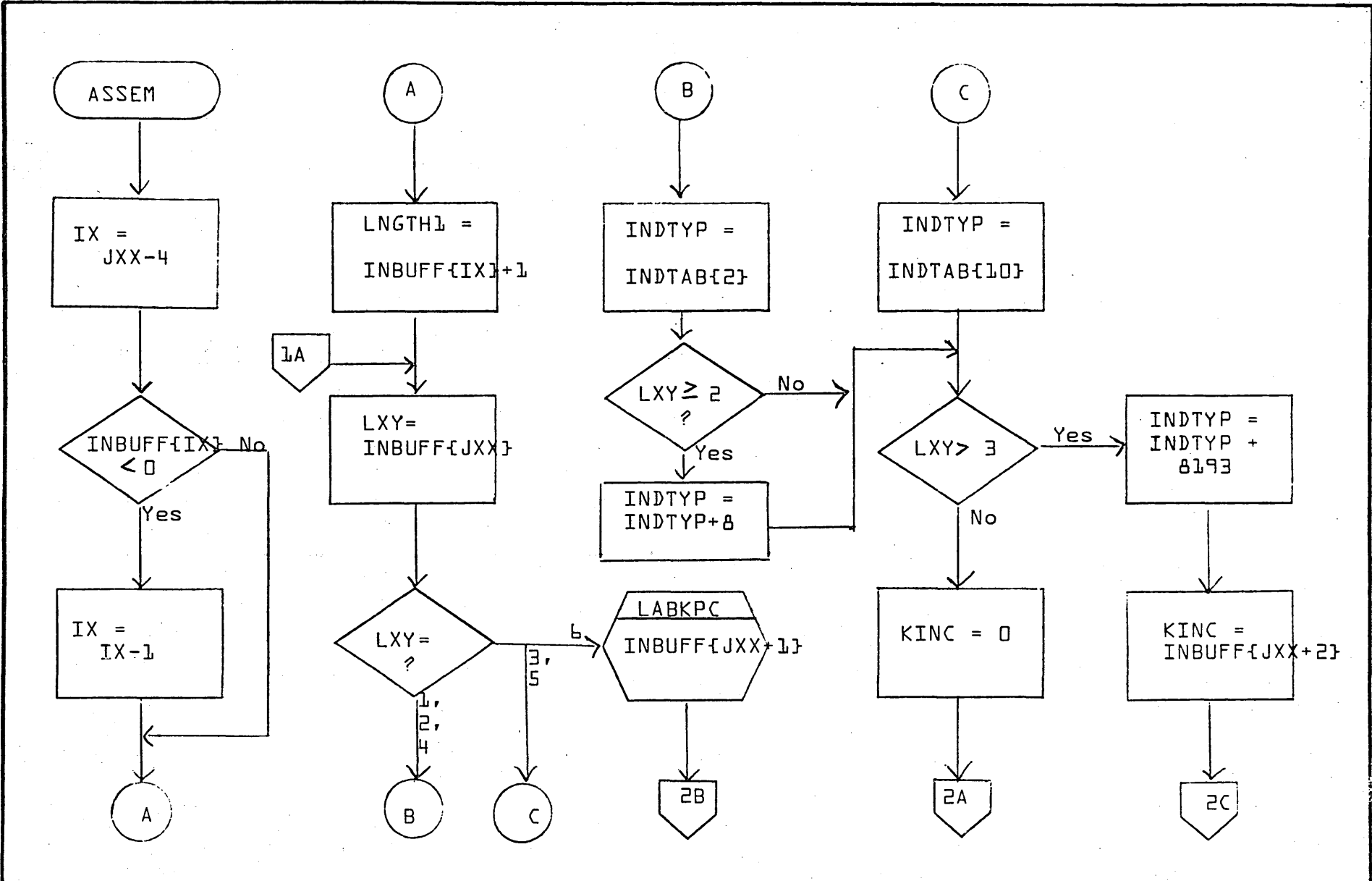
DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	BANANA	PAGE 3 OF 3		PROJECT MGR.			
NUMBER	ISSUE DATE			PROJECT NAME			
DRAWN BY		DATE		TASK NO.			
				TASK NAME			

4-29

DOCUMENT CLASS IMS PAGE NO. 4-80  
PRODUCT NAME 1700 Mass Storage FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

4.3.10 ASSEM

ASSEM is the processor of the ASSEM statement. The ASSEM statement provides the facility for the FORTRAN programmer to generate hexadecimal constants, address constants, and statement labels. These are received by Phase B as a consecutive list and processing is simply a matter of generating CON and ADC commands as appropriate.



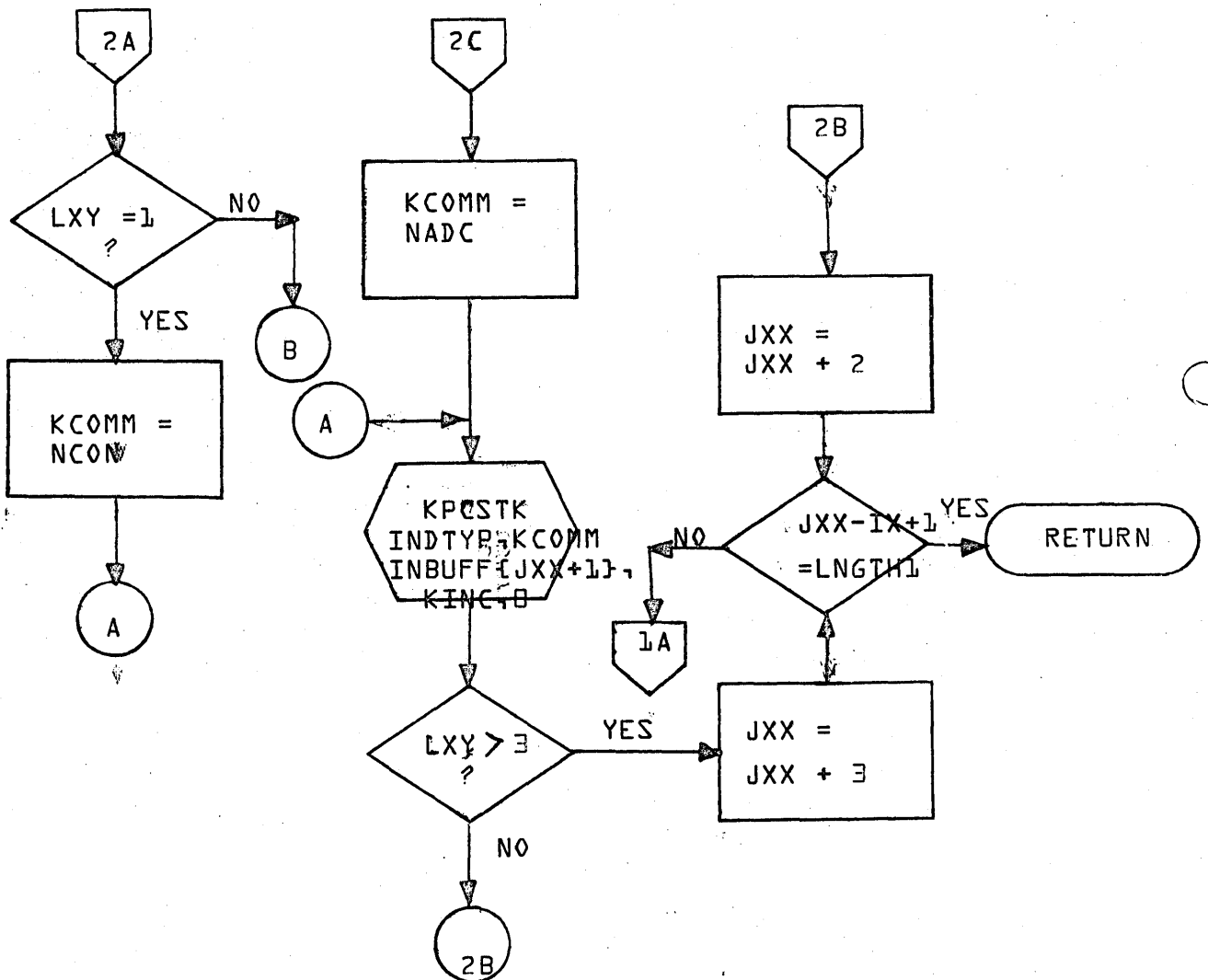
**CONTROL DATA CORPORATION**  
 SOFTWARE DOCUMENT

SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	ASSEM PROCESSOR			PROJECT MGR.			
	PAGE 1 OF 2			PROJECT NAME			
NUMBER	ISSUE DATE			TASK NO.			
DRAWN BY	DATE			TASK NAME			



LA JOLLA FACILITY



TITLE		ASSEM PRØCESSØR		DRG. NO.	
DRAWN BY		PROJ.		REVISION	
CSD 203		DATE		SHEET 2 OF 2	

DOCUMENT CLASS IMS PAGE NO. 4-83  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

4.4 PHASEB Subroutines

4.4.1 AFIDL

Subroutine AFIDL is an intermediate link between the PHASEB in-line processor of arithmetic IF statements and logical IF statements.

AFIDL calls ASUPER and in addition performs certain checking and code generation. {ASUPER is responsible for generating code to evaluate arithmetic expressions.}

1. AFIDL looks at the mode of the expression which was processed by ASUPER. If it is either a single precision or double precision floating point expression AFIDL generates an instruction to load the upper portion of the pseudo-accumulator {address C5} into the hardware accumulator so that test instructions may operate upon that value.
2. If the expression type was integer, AFIDL determines whether or not it is necessary to generate an

INA 0

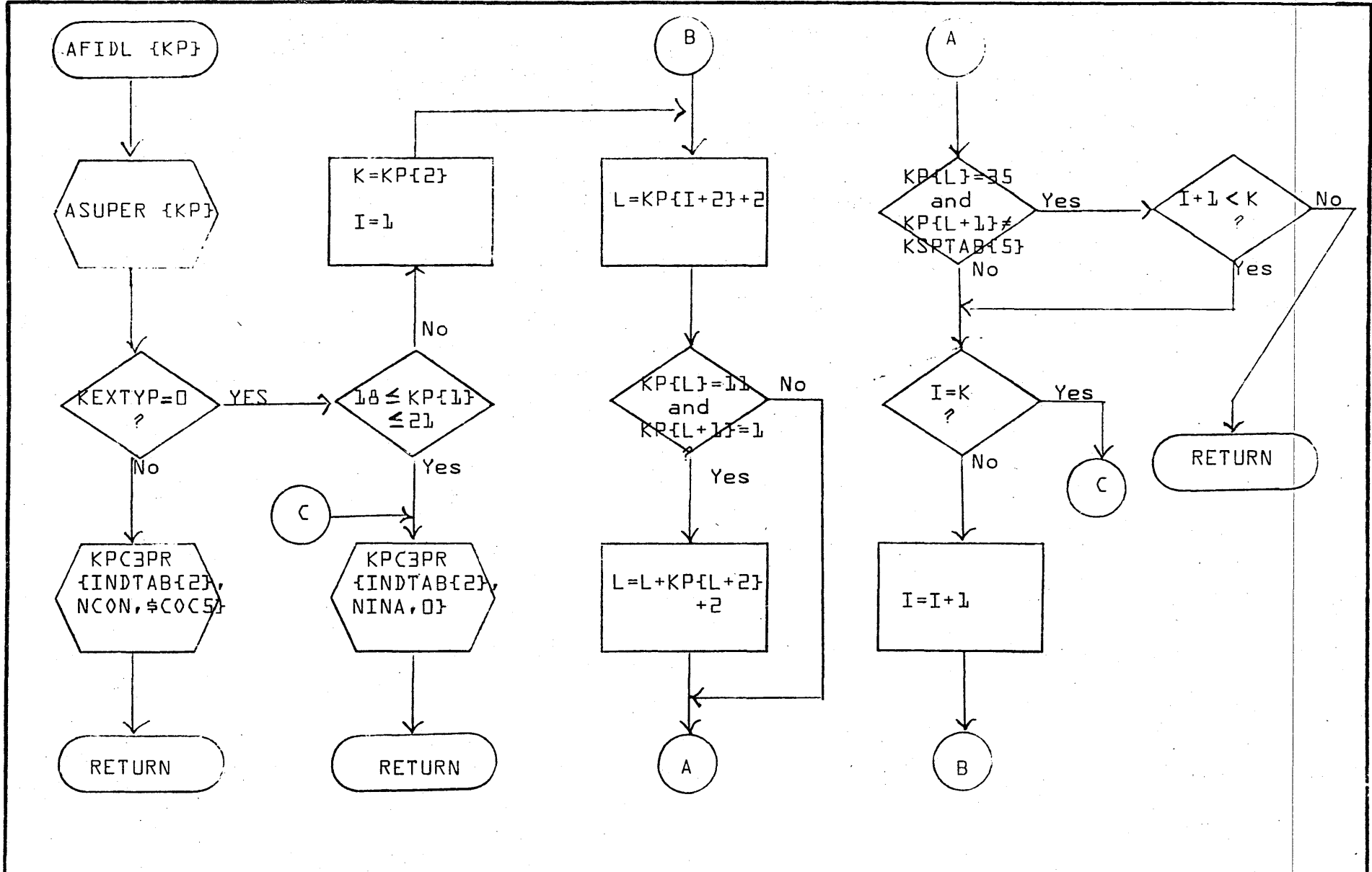
command to insure that the value -0 {FFFF<sub>16</sub>} is not left in the accumulator. {Test instructions do not allow for -0.} If necessary, AFIDL generates such an instruction.

A

B

C

D



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE

FLOWCHART

DECISION TABLE

OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	AFIDL	PAGE 1 OF 1		PROJECT MGR.			
NUMBER	ISSUE DATE	TASK NO.		PROJECT NAME			
DRAWN BY	DATE	TASK NAME					

h9-h

DOCUMENT CLASS IMS PAGE NO. 4-85  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

## 4.4.2

## ASUPER

ASUPER is the arithmetic "tree" processor. It is called once by each statement processor where the statement being processed contains an arithmetic expression. The following statements contain arithmetic expressions.

## 1. Arithmetic Replacement Statement

Processed by ARITHR which calls ASUPER for the expression on the right of the equal (=) sign.

## 2. Statement Function

Processed by *PHASEB* which calls ASUPER for the expression on the right of the equal (=) sign.

## 3. CALL Statement

Processed in-line by Phase *B* which calls ASUPER for the CALL expression. (A CALL statement is put into the "tree" format where the CALL is the highest level operator and its parameters are the operands.

## 4. GO TO (Computed)

Processed by CGOTO which calls ASUPER for the expression which follows the GO TO parenthesized label list.

## 5. Arithmetic IF

Processed by *PHASEB* which calls AFIDL and AFIDL calls ASUPER in turn for the expression within the parenthesis of the IF.

## 6. Logical IF

Processed by *PHASEB* which calls AFIDL and AFIDL in turn calls ASUPER for each arithmetic expression within the parenthesized logical expression.

The job of ASUPER is to generate executable instructions which perform the evaluation of an expression. This is performed in two steps.

I. The arithmetic tree is stepped through, operand by operand, in a search for subscripts. The disposition of them is based upon their type and whether or not their operator is a CALL, non-inline function call, or exponentiation (\*\*). If the operand is a non subscripted variable or non-subscripted partial variable, it is passed over.

A. Variable only subscripted variable (partial or non-partial).

DOCUMENT CLASS IMS PAGE NO. 4-86  
PRODUCT NAME 1700 Mass Storage FORTRAN  
PRODUCT NO. C005 Version 2.0 MACHINE SERIES 1700

- B. Increment only subscripted variable (partial or non-partial).
- C. Variable and increment only subscripted variable (partial or non-partial).
- D. Complex subscripted variable (partial or non-partial).

Then one of two subscript processors (SUBPR1 or SUBPR2) is called upon to further process the subscripted variable.

A. If the variable is an operand of an operator which is:

- 1. a CALL
- 2. a non-inline function, or
- 3. an exponentiation (\*\*)

then SUBPR2 is called.

B. If the variable is an operand of any other operator then SUBPR1 is called.

II. Having completed the processing of all subscripts then ASUPER calls ACP which generates the instructions which evaluate the expression.



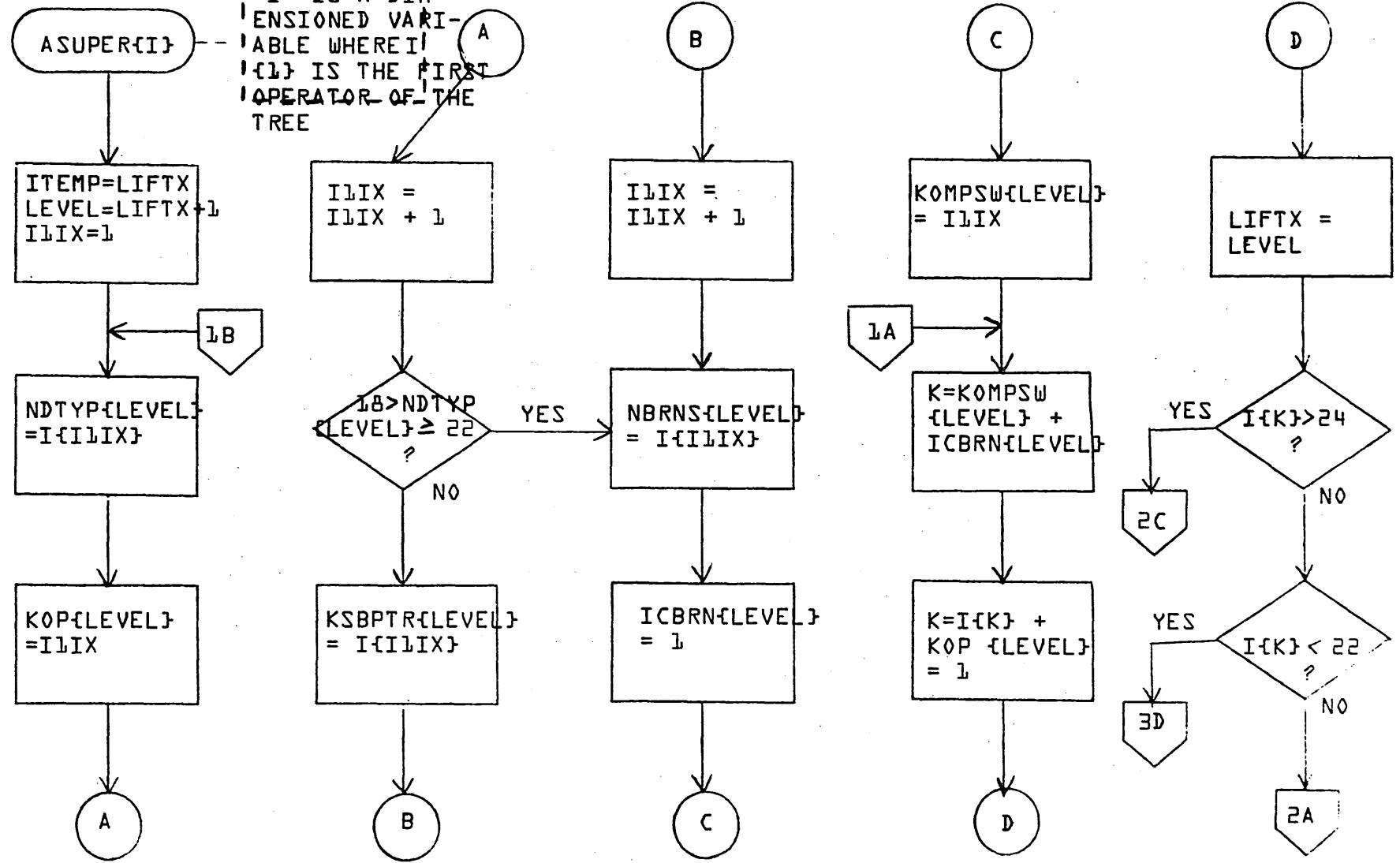
A

B

C

D

$I\{I\}$  IS A DIM-  
 ENSIONED VARI-  
 ABLE WHERE  $I\{I\}$   
 IS THE FIRST  
 OPERATOR OF THE  
 TREE



**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

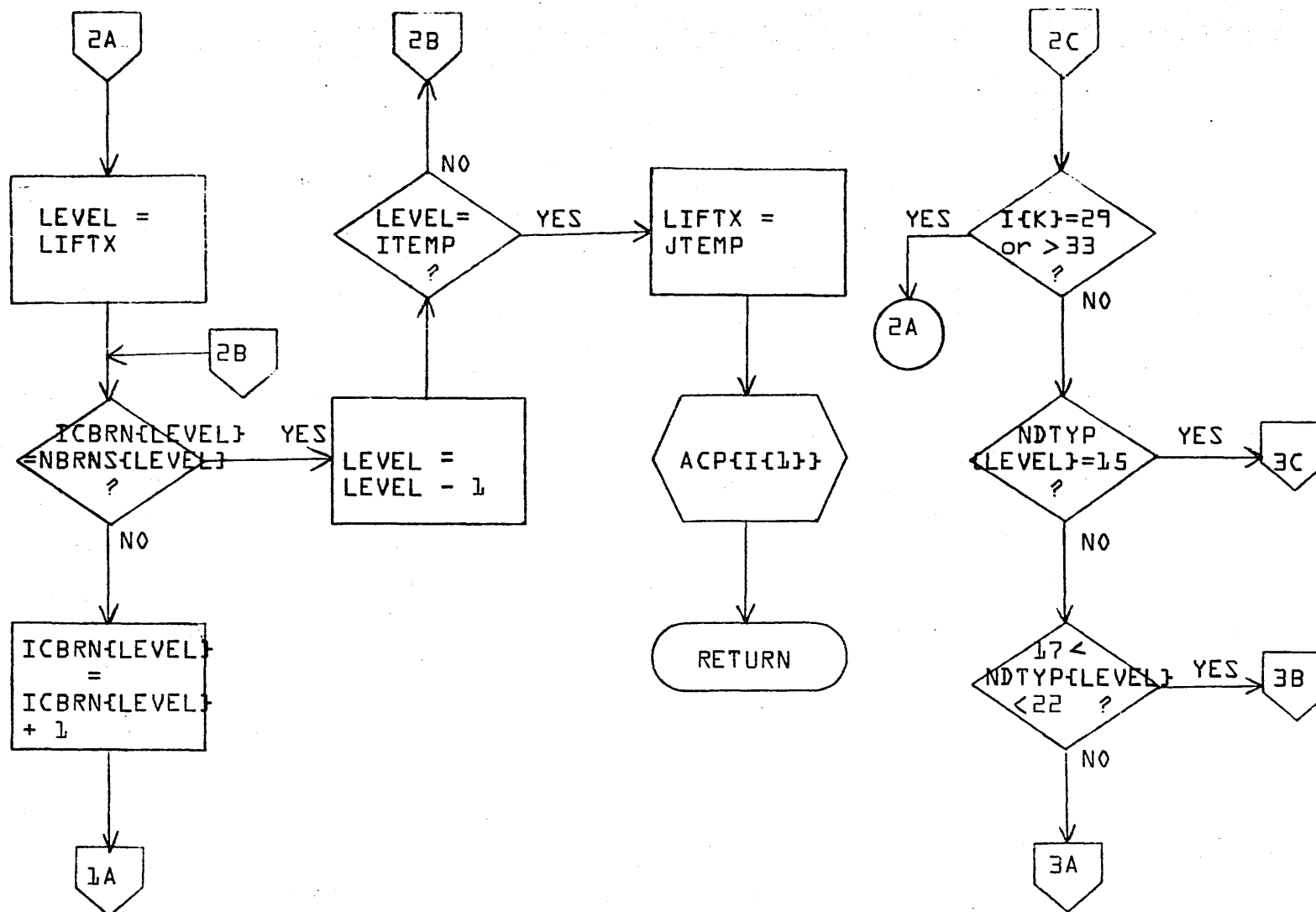
DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	ASUPER	PAGE 1 OF 3		PROJECT MGR.			
NUMBER	ISSUE DATE			PROJECT NAME			
DRAWN BY	DATE			TASK NO.			
				TASK NAME			

A

B

C

D


**CONTROL DATA CORPORATION**  
 SOFTWARE DOCUMENT

 SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER 

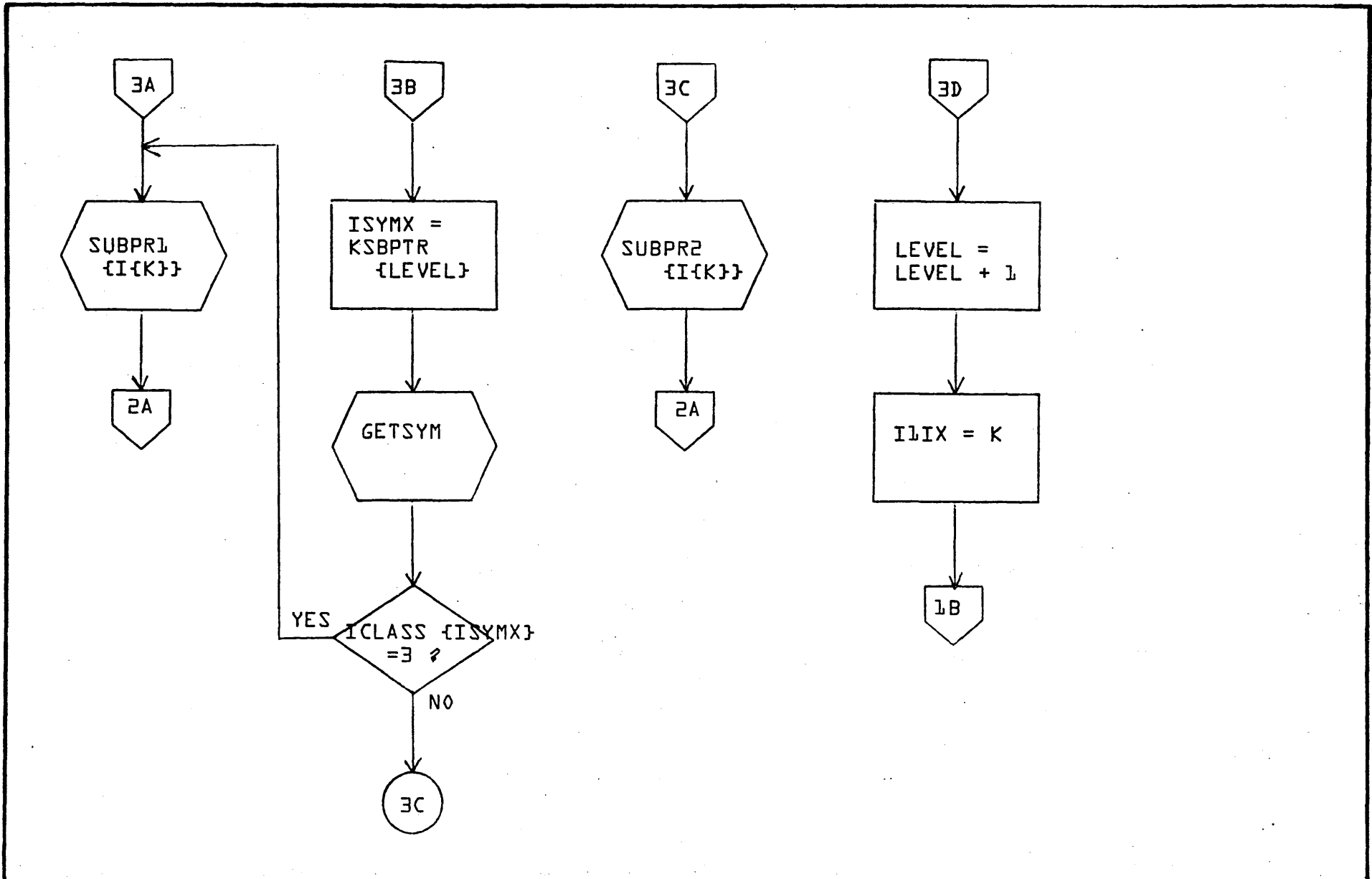
DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	ASUPER	PAGE 2 OF 3		PROJECT MGR.			
NUMBER	ISSUE DATE			PROJECT NAME			
DRAWN BY	DATE			TASK NO.			
				TASK NAME			

A

B

C

D



**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	ASUPER			PROJECT MGR.			
PAGE 3 OF 3				PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.					
DRAWN BY		DATE		TASK NAME			

4-1-60

4.4.4 ENTCOD

Refer to section 4.2.1 (Program Structure)

ENTCOD is responsible:

1. For Subroutines and Function Subprograms, generation of all instructions from the entry point to the transfer to the start of the routine (label MBEGIN).
2. For Statement Functions (with Main Subroutine, or Function Subprograms) generation of all instructions from the statement function entry point to the transfer to the start of the Statement Function.

The Structure of the entry code (thus the instruction generated by ENTCOD) is as shown:

CON	0	constant of 0 marks the entry point to the subroutine or function subprogram or statement function.
STQ	QSAV	instructions which save the value of Q and I indexes by storing in created storage locations QSAV and FFSAV.
LDQ	FF	
STQ	FFSAV	
RTJ	Q8PREP	return jump to the parameter pickup routine (PARAMS) at the initialization entry Q8PREP with an address constant of the entry point as a parameter. This code is generated only if there is at least one parameter to the subroutine or function that the entry code is being generated for.
ADC	name	
RTJ	Q8PKUP	return jump to the parameter pickup routine (PARAMS) at the pickup entry Q8PKUP which returns the address of the <u>first</u> parameter in the A register.
	(series of INA and PST commands to plug this parameter)	
RTJ	Q8PKUP	pickup of address of <u>second</u> parameter.
	(series of INA and PST commands to plug this parameter)	
		⋮
RTJ	Q8PKUP	pickup of address of <u>nth</u> parameter.
	(series of INA and PST commands to plug this parameter)	

DOCUMENT CLASS IMS PAGE NO. 4-91  
 PRODUCT NAME 1700 Mass Storage FORTRAN  
 PRODUCT NO. C005 Version 2.0 MACHINE SERIES 1700

JMP	KEXEC		jump to beginning of program. This jump actually initiates the execution of the subroutine or function, having completed the necessary index saving and parameter pickup and plugging
FFSAV	BSS	1	storage cells to save Q and I index registers
QSAV	BSS	1	

The INA and PST commands which follow each RTJ Q8PKUP are based upon the final settings of KSUBAT, the subroutine argument table (for subroutines and function subprograms) or the settings of KSFAT, the statement function argument table (for entry code of statement functions).

1. Subroutines and Function Subprograms. At the point of operation of program ENCODE, KSUBAT has been completely built and it reflects each argument and each increment that has occurred with that argument throughout the compilation. The increments are sorted from lowest to highest and to each argument/increment pair is assigned a unique temporary storage cell.

Having called Q8PKUP which returns with the address of the argument in the accumulator ENTCOD then generates instructions which compute each resulting address (address of argument and increment) and a PST (Parameter store) where the operand of the PST is the temporary storage assignment.

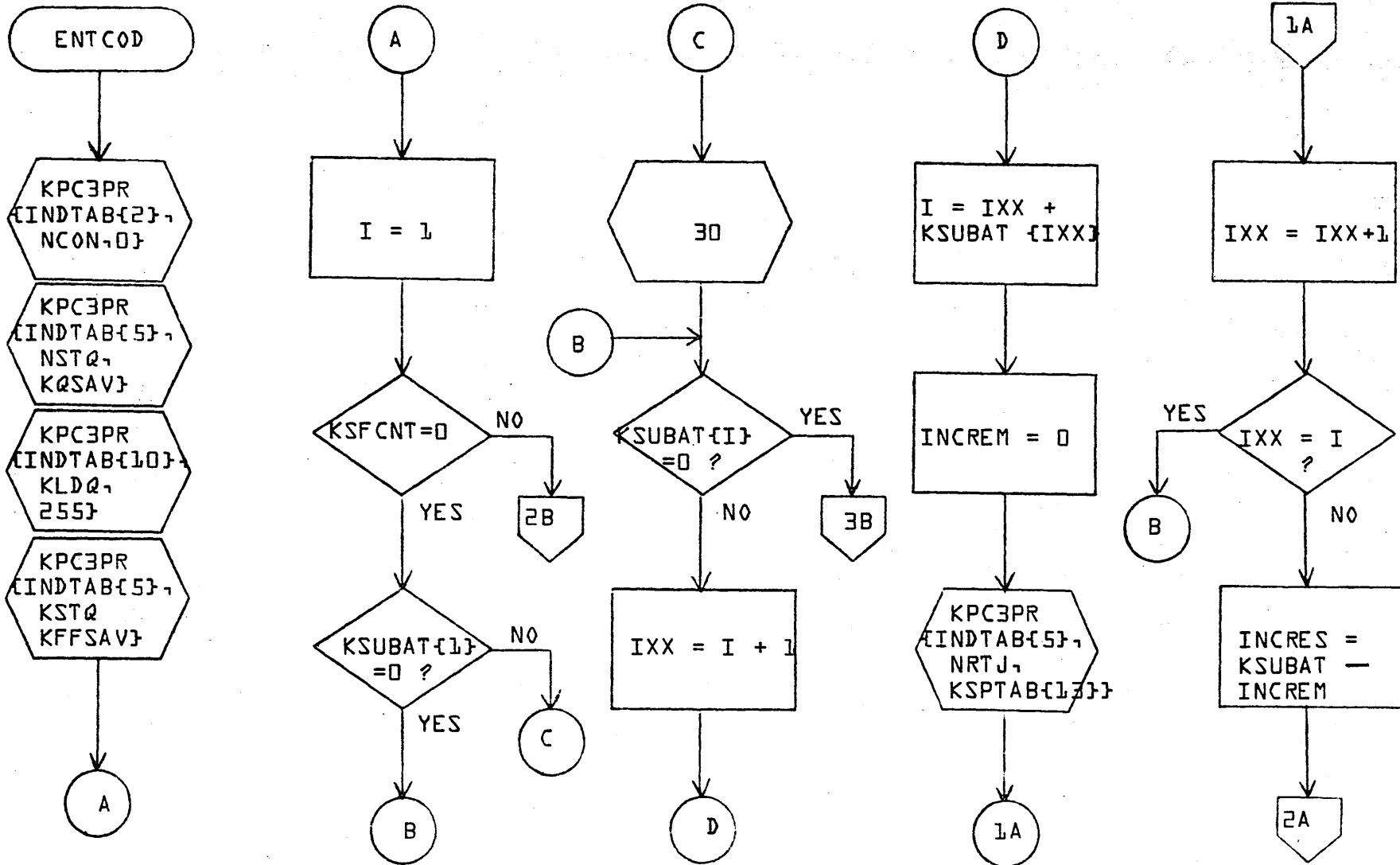
2. Statement Functions. At the point of operation of program ENTCOD, KSFAT has been completely built and it reflects each argument and a unique temporary storage cell assigned to that argument. (Statement function argument can occur with only the nominal increment of 0 since they may never be subscripted).

Having called Q8PKUP which returns with the address of the argument in the accumulator ENTCOD then generates a PST where the operand of the PST is the temporary storage assignment.

Section 4.4.20 on DUMMY contains a more detailed description of the structure and dynamics of KSUBAT and KSFAT and a more detailed description of subprogram interface design.

Flowcharts:

A  
B  
C  
D



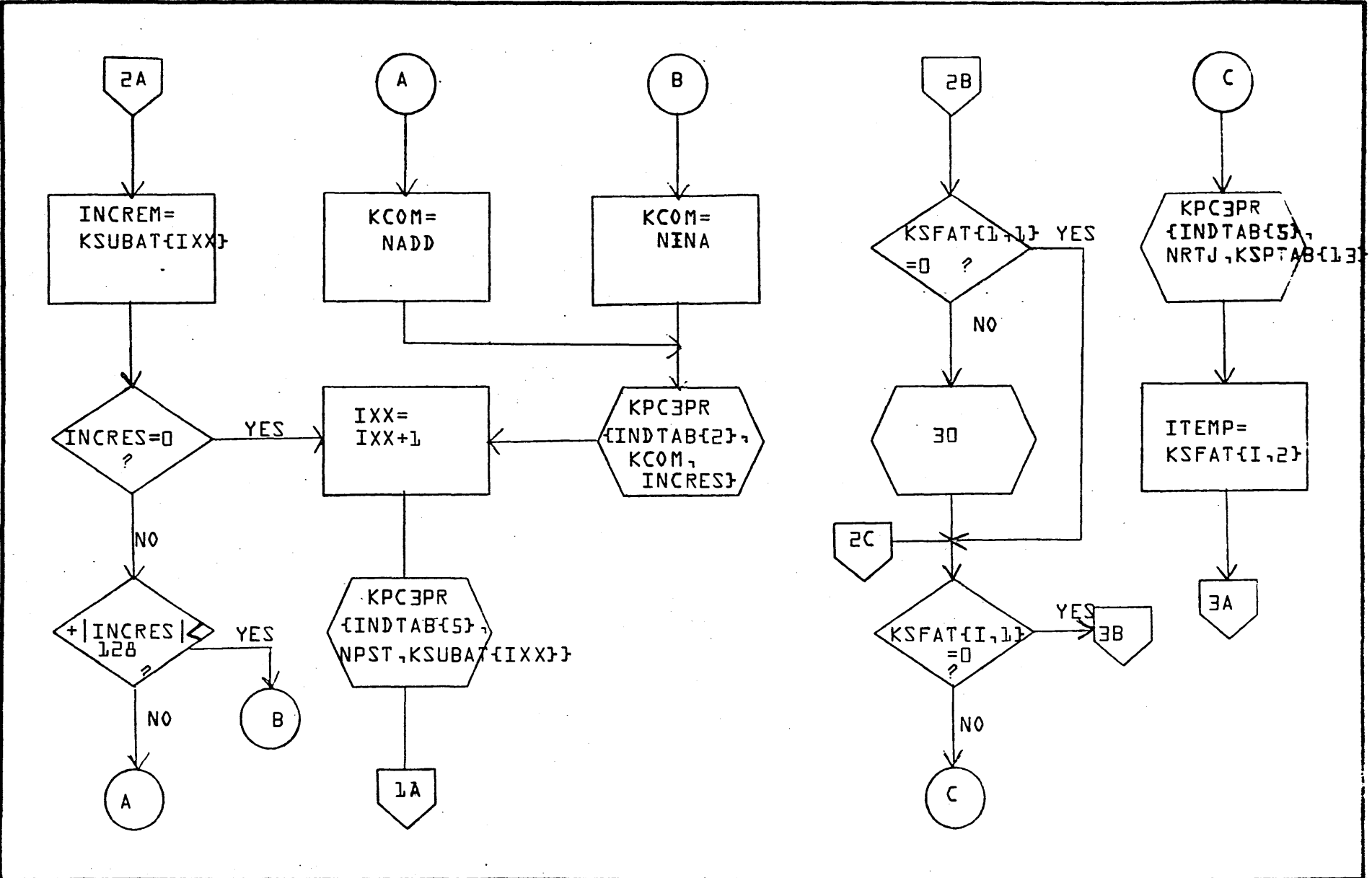
<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	ENTCOD		PAGE 1 OF 3		PROJECT MGR.		
	NUMBER	ISSUE DATE	TASK NO.		PROJECT NAME			
	DRAWN BY	DATE	TASK NAME					

A

B

C

D



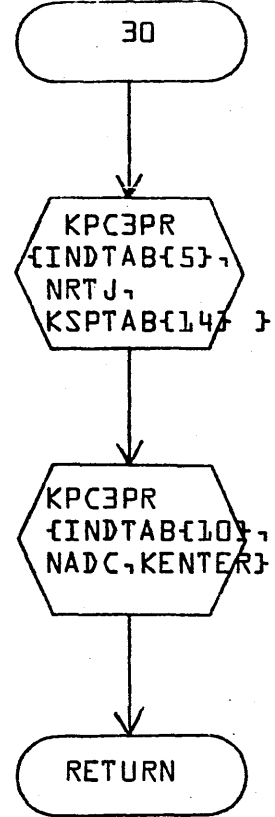
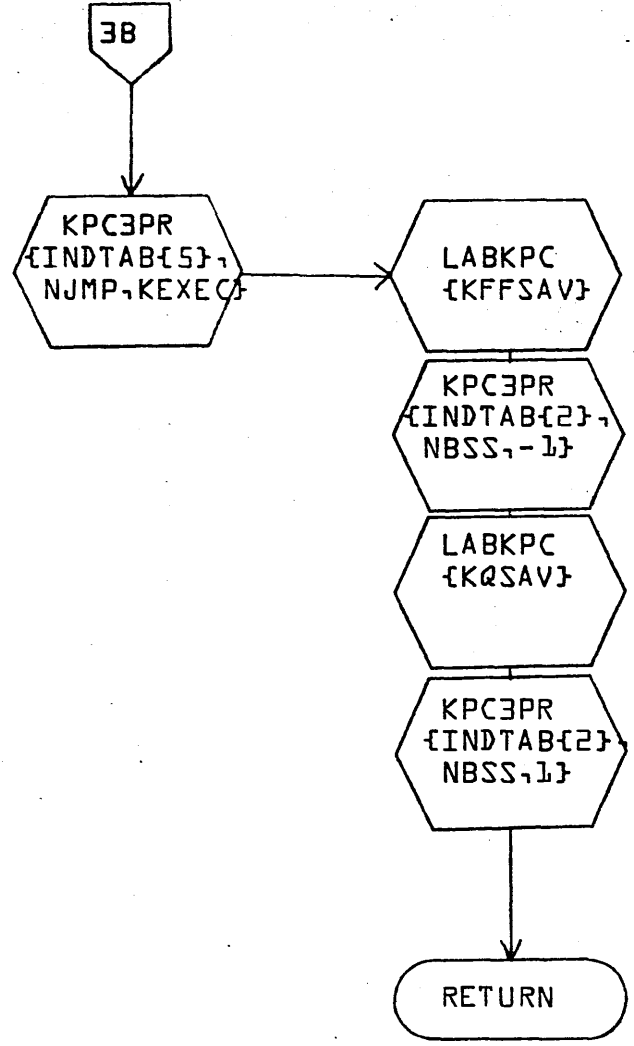
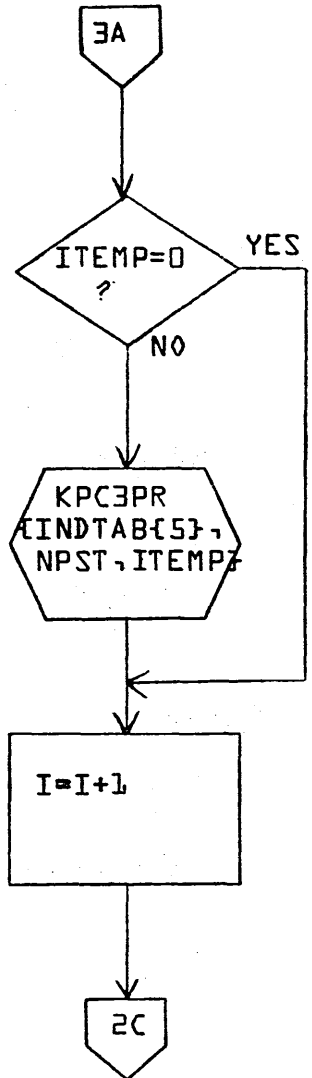
<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	ENTCOD	PAGE 2 OF 3		PROJECT MGR.			
	NUMBER		ISSUE DATE		PROJECT NAME			
	DRAWN BY		DATE		TASK NO.			
					TASK NAME			

A

B

C

D



**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

SAMPLE CODE

FLOWCHART

DECISION TABLE

OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	ENTCOD			PROJECT MGR.			
PAGE 3 OF 3				PROJECT NAME			
NUMBER	ISSUE DATE			TASK NO.			
DRAWN BY	DATE			TASK NAME			

56-4



DOCUMENT CLASS IMS PAGE NO. 4-95  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005x3.1 A/B MACHINE SERIES 1700

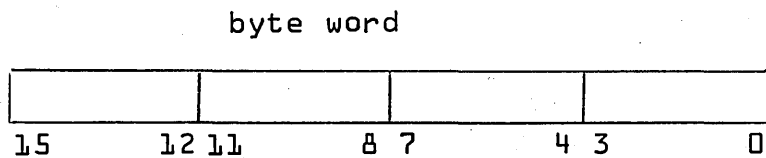
4.4.5 FCMSTK

FCMSTK is a subroutine used exclusively by KPCSTK for building floating-point calling sequences. The type of the floating-point calling sequence {single precision or double precision} is transparent to FCMSTK. If the symbol table pointer is minus one, it indicates the special case of a double precision load from C5. FCMSTK specifically is responsible for monitoring the KFCET table. The structure of KFCET is shown:

KFCET

	n <sub>1</sub>	n <sub>2</sub>	n <sub>3</sub>
1 <sub>m</sub>	BYTE WORD	NOT USED	
2 <sub>m</sub>	INDICATOR WORD	OPERAND	ADDITIVE
3 <sub>m</sub>	⋮	⋮	⋮
4 <sub>m</sub>	⋮	⋮	⋮
5 <sub>m</sub>	⋮	⋮	⋮

The byte word {KFCET{1,1}} is divided into four 4-bit bytes in accordance with the floating point calling sequence format. KBYTX indexes the byte position as shown:



and:

- KBYTX = 1, bits 15-12
- = 2, bits 11-8
- = 3, bits 7-4
- = 4, bits 3-0

KFCETX indexes the rows of KFCET. There is space for from 0 to 4 entries in KFCET where each entry is 3 words and comprises a row of KFCET {Rows 2 through 5}.

CONTROL DATA CORPORATION  
LA JOLLA RESOURCE CENTER DIVISION

DOCUMENT CLASS IMS PAGE NO. 4-95A  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

Parameters: There are two parameters to FCMSTK:

KF1 - floating instruction {see floating point package for a detailed list of floating commands}. The floating instruction is a 4 - bit value.

KF2 - operand for those floating commands such as FLD, DLD, FST, and DST where there is an operand; KF2 = 0 for those commands which have no operands {FCOM, etc.}.

DOCUMENT CLASS IMS PAGE NO 4-96  
PRODUCT NAME 1700 Mass Storage FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

FCMSTK receives a call from KFCSTK with the parameter set for a floating command. (Hence, a request to add an entry to KFCET). FCMSTK adds the instruction byte to the byte word and updates index KBYTX. If there is an operand, FCMSTK updates KFCETX to the next entry and adds the operand to the table. (The indicator word and the additive are picked up from the original calling sequence to KPCSTK, KQ(1) and KQ(4) respectively. The exception is an indexing command,  $KF1 = F_{16}$ , where the indicator word is created as a function of the current addressing mode and there is not an additive.)

If:

1. All four bytes of the byte word have been filled with floating commands, or,
2. The last floating command entered was a stop (4)

then FCMSTK "dumps" the KFCET table in the following manner:

1. Generates the byte word as an integer constant.
2. Generates each entry in the table as an address constant.
3. Reinitializes all indexes to the table.



A

B

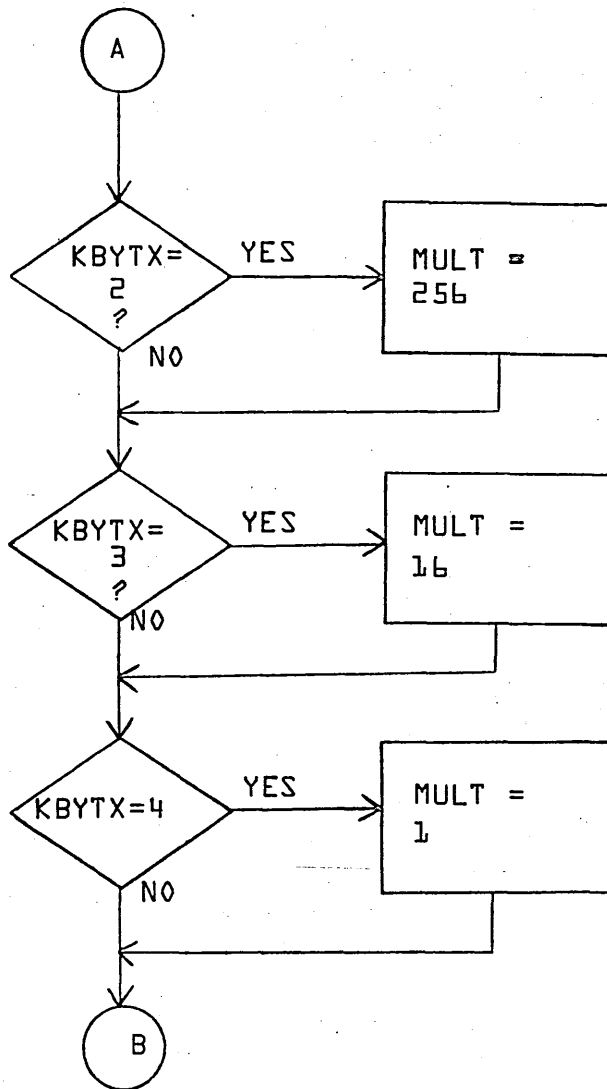
C

D

FCMSTK  
{KF1, KF2}

MULT =  
4096

A



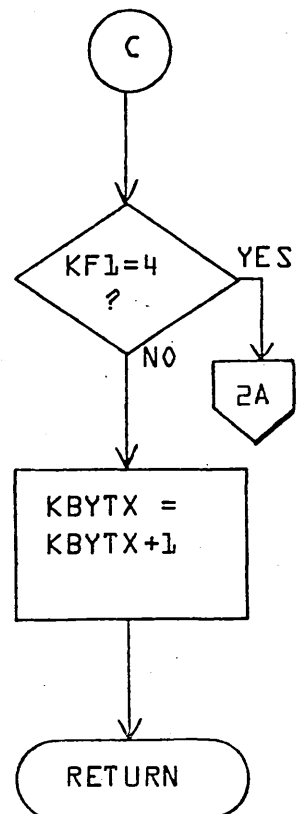
B

KFCET{1,1}=  
KFCET{1,1}+  
KF1 \* MULT

1A

KBYTX=4

C



C

KF1=4?

2A

KBYTX =  
KBYTX+1

RETURN

CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

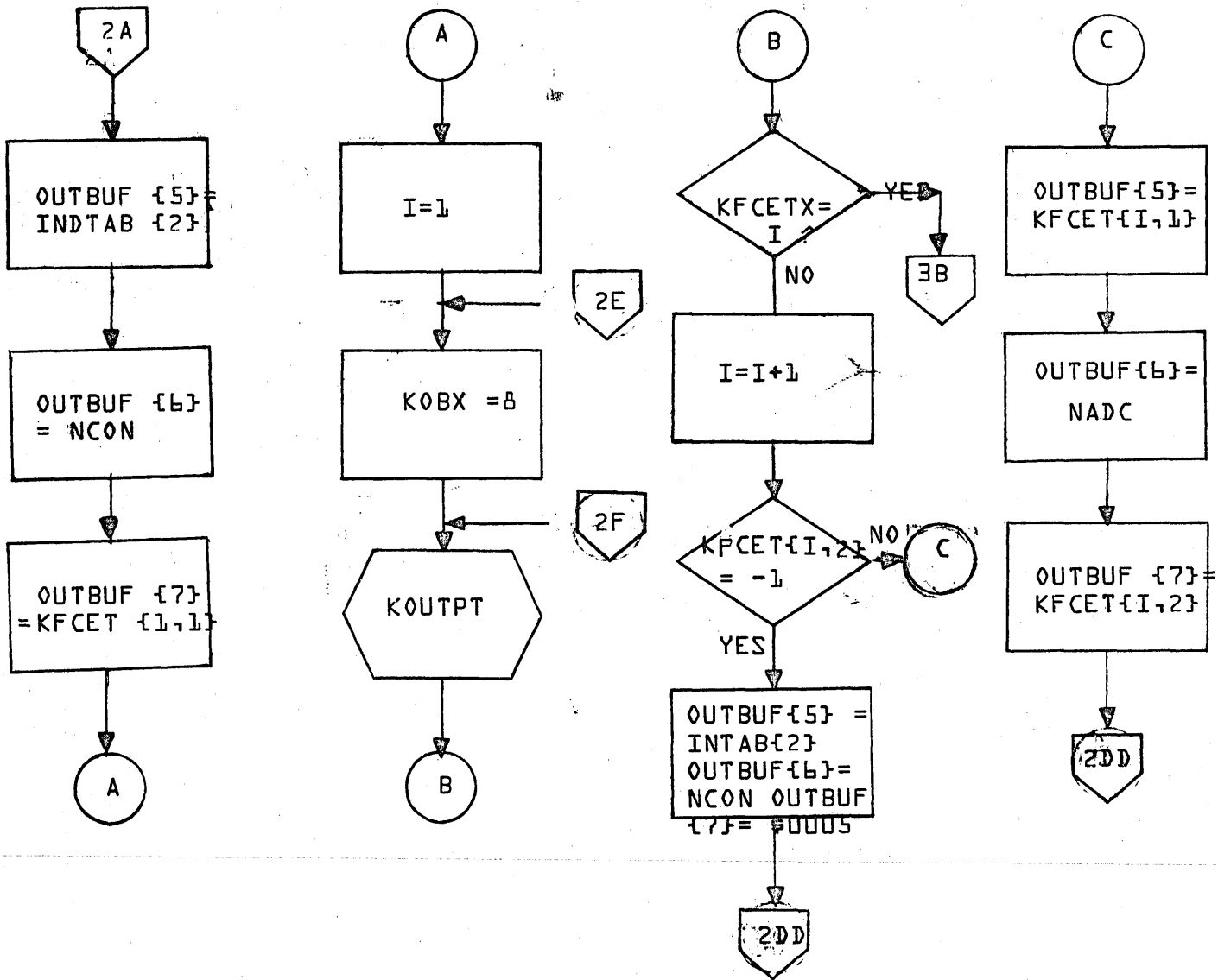
DOCUMENT CLASS	IMS	MACH. TYPE	1700
DOCUMENT TITLE	FCMSTK		
PAGE 1 OF 3			
NUMBER	ISSUE DATE		
DRAWN BY	DATE		

PROJECT NO.	REV	APPROVED	DATE
PROJECT MGR.			
PROJECT NAME			
TASK NO.			
TASK NAME			

4-97



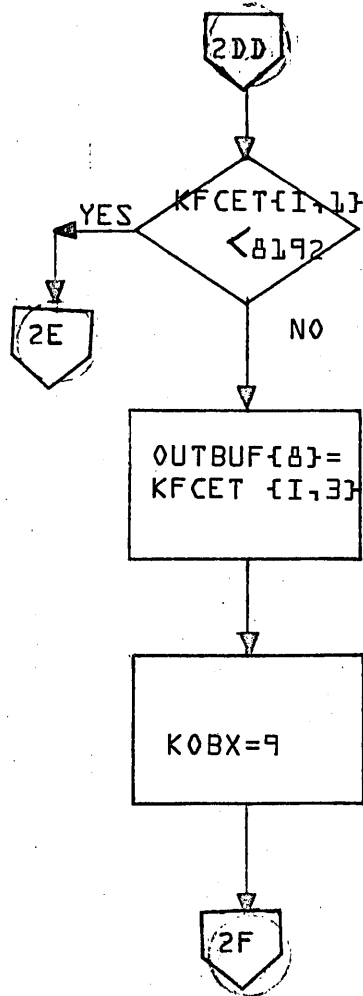
LA JOLLA FACILITY



TITLE		FCMSTK		DRG. NO.	
DRAWN BY		PROJ.		REVISION	
CSD 203		DATE		SHEET 2 OF 3	



LA JOLLA FACILITY

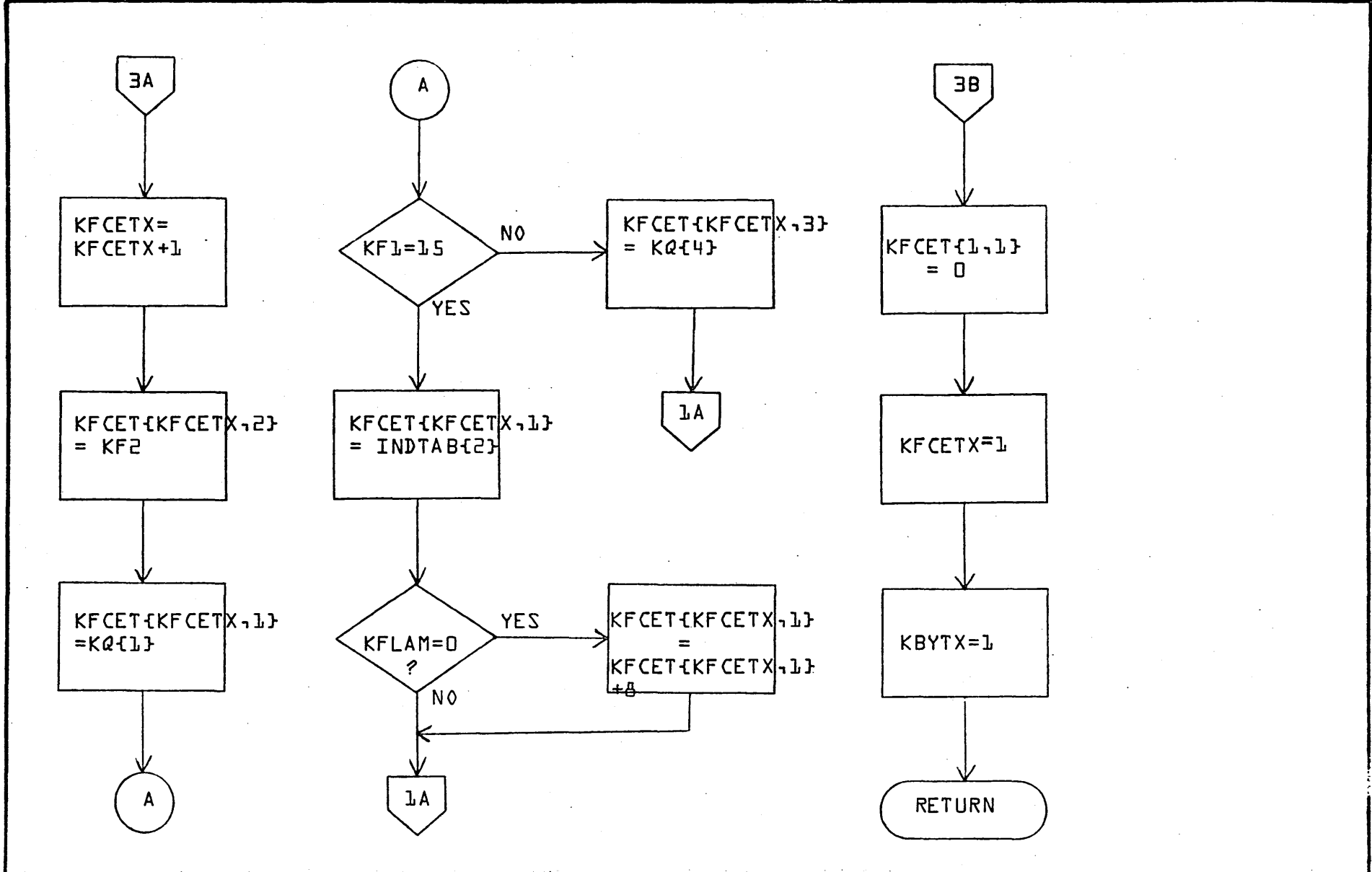


TITLE			DRG. NO.
FCMSTK			REVISION
DRAWN BY	PROJ.	DATE	SHEET 2a OF 3
CSD 203			





A  
B  
C  
D



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	FCMSTK	PAGE 3 OF 3		PROJECT MGR.			
NUMBER	ISSUE DATE			PROJECT NAME			
DRAWN BY	DATE			TASK NO.			
				TASK NAME			

U-99

DOCUMENT CLASS IMS PAGE NO 4-100  
 PRODUCT NAME 1700 Mass Storage FORTRAN  
 PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

## 4.4.6

## FINK

FINK is a subroutine responsible for generating instructions of in-line functions. The choice to make a function in line or not in-line was made on an individual basis for each function based upon:

1. The number of locations which would be required to perform the function in line. And in contrast,
2. The number of locations which would be required to execute a calling sequence.
3. The overhead that must be paid to make a function a separate subroutine.

The following functions were made in-line functions. They are listed with the instructions which are generated by FINK to execute them.

1. IABS(I)
 

LDA	I
AJP,GEZ	SKIP
TCA	A
SKIP	:
	:
2. AND (I,J)
 

LDA	I
AND	J
3. OR (I,J)
 

LDA	I
LDQ	J
TCA	A
TCQ	Q
CAQ	A
4. EOR (I,J)
 

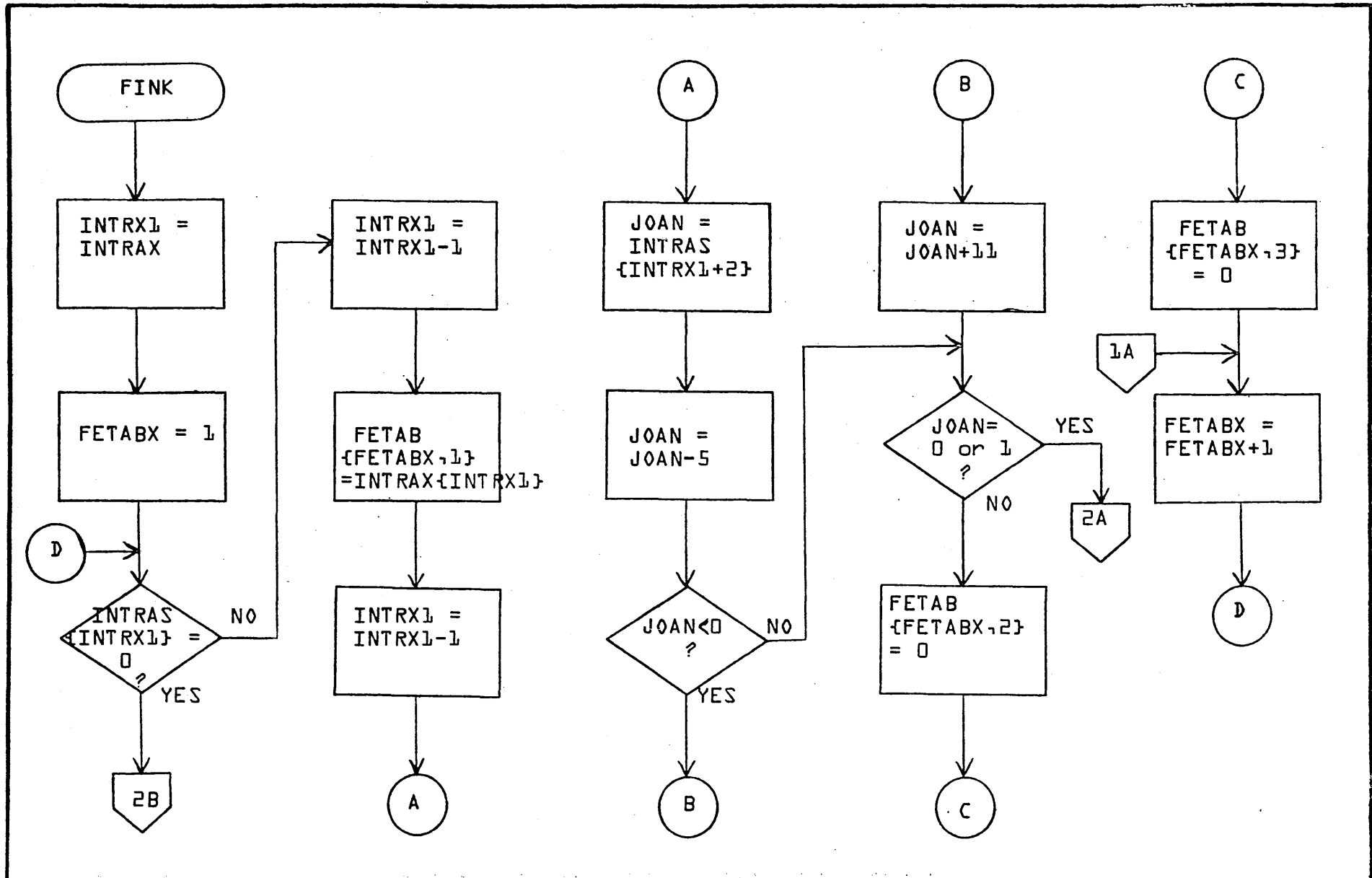
LDA	I
EOR	J
5. NOT (I)
 

LDA	I
TCA	A
6. ISIGN (I,J)
 

LDA	I
MUI	J
LRS	31
EOR	I

All other functions are library subroutines.

A  
B  
C  
D



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	FINK	PAGE 1 OF 4		PROJECT MGR.			
	NUMBER		ISSUE DATE		PROJECT NAME			
	DRAWN BY		DATE		TASK NO.			
					TASK NAME			

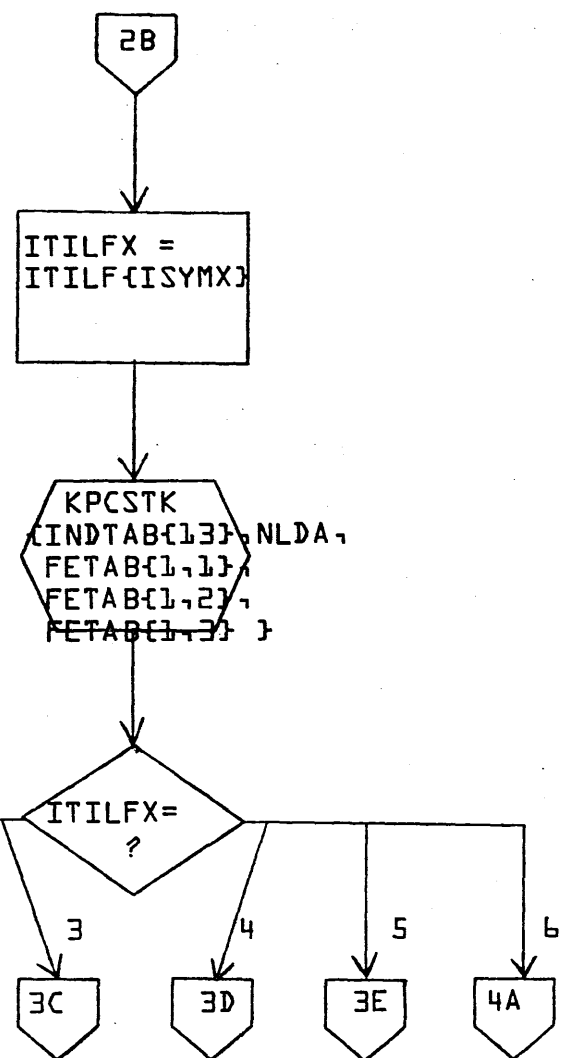
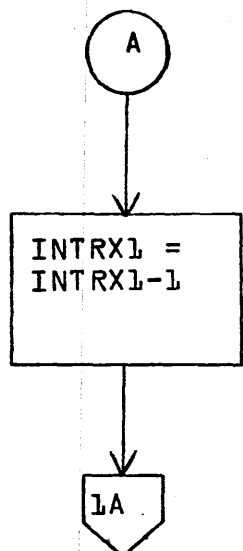
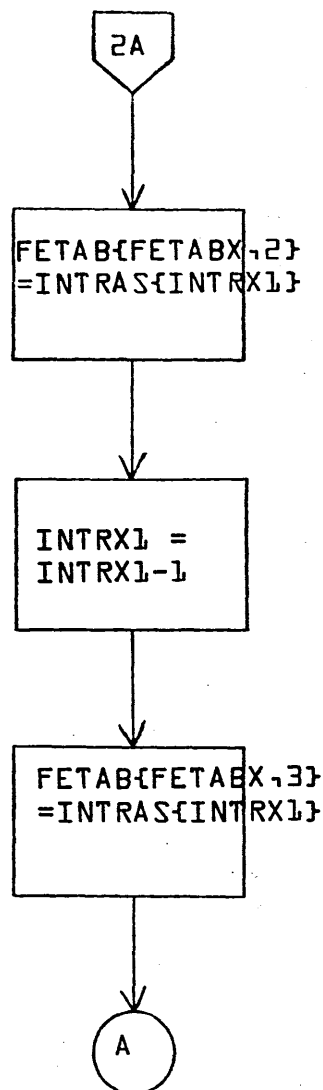
4-101

A

B

C

D



**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

SAMPLE CODE

FLOWCHART

DECISION TABLE

OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	FINK	PAGE 2 OF 4		PROJECT MGR.			
NUMBER	ISSUE DATE	PROJECT NAME		TASK NO.			
DRAWN BY	DATE	TASK NAME					

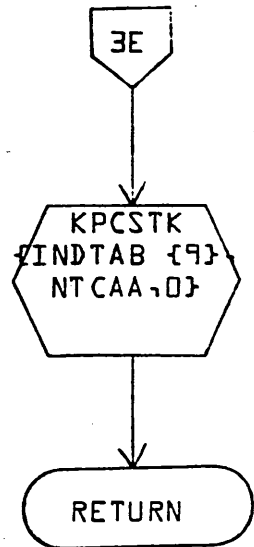
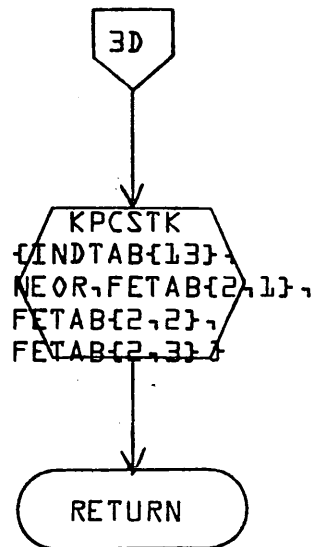
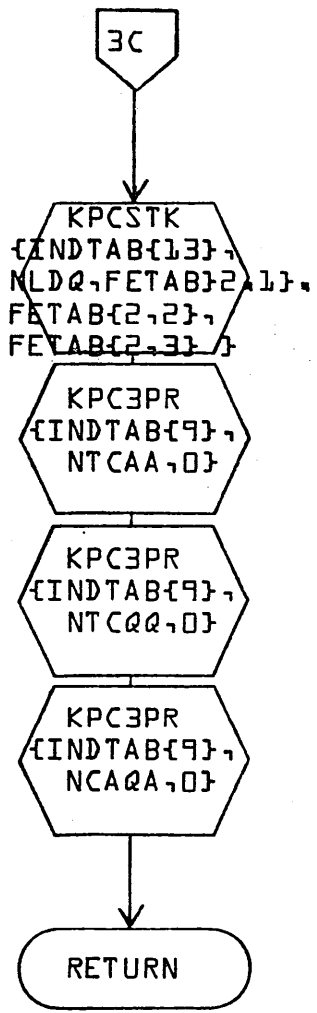
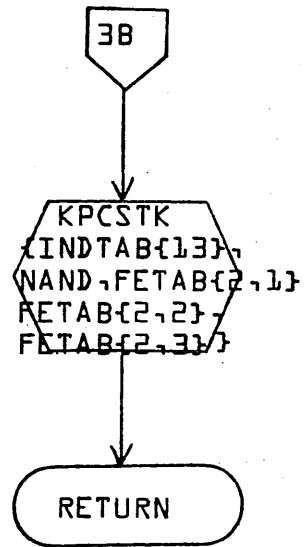
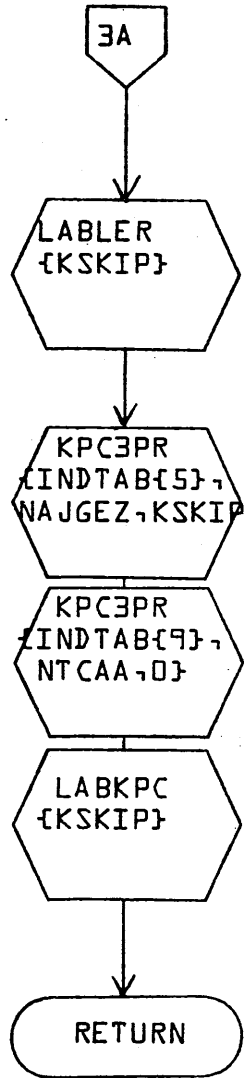
4-102

A

B

C

D



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	FINK			PROJECT MGR.			
PAGE 3 OF 4				PROJECT NAME			
NUMBER	ISSUE DATE			TASK NO.			
DRAWN BY	DATE			TASK NAME			

4A

KPCSTK  
{INDTAB{1,3},  
NMUI,FETAB{2,1},  
FETAB{2,2},  
FETAB{2,3}} }

KPC3PR  
{INDTAB{2},  
NLRS,3}}

KPCSTK  
{INDTAB{1,3},  
NEOR,FETAB{1,1},  
FETAB{1,2},  
FETAB{1,3}} }

RETURN

CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT  
SAMPLE CODE   
FLOWCHART   
DECISION TABLE   
OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	FINK	PAGE 4 OF 4		PROJECT MGR.			
NUMBER	ISSUE DATE	TASK NO.					
DRAWN BY	DATE	TASK NAME					

NOTES

DOCUMENT CLASS \_\_\_\_\_ IMS \_\_\_\_\_ PAGE NO. 4-105  
 PRODUCT NAME 1700 Mass Storage FORTRAN  
 PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

4.4.7

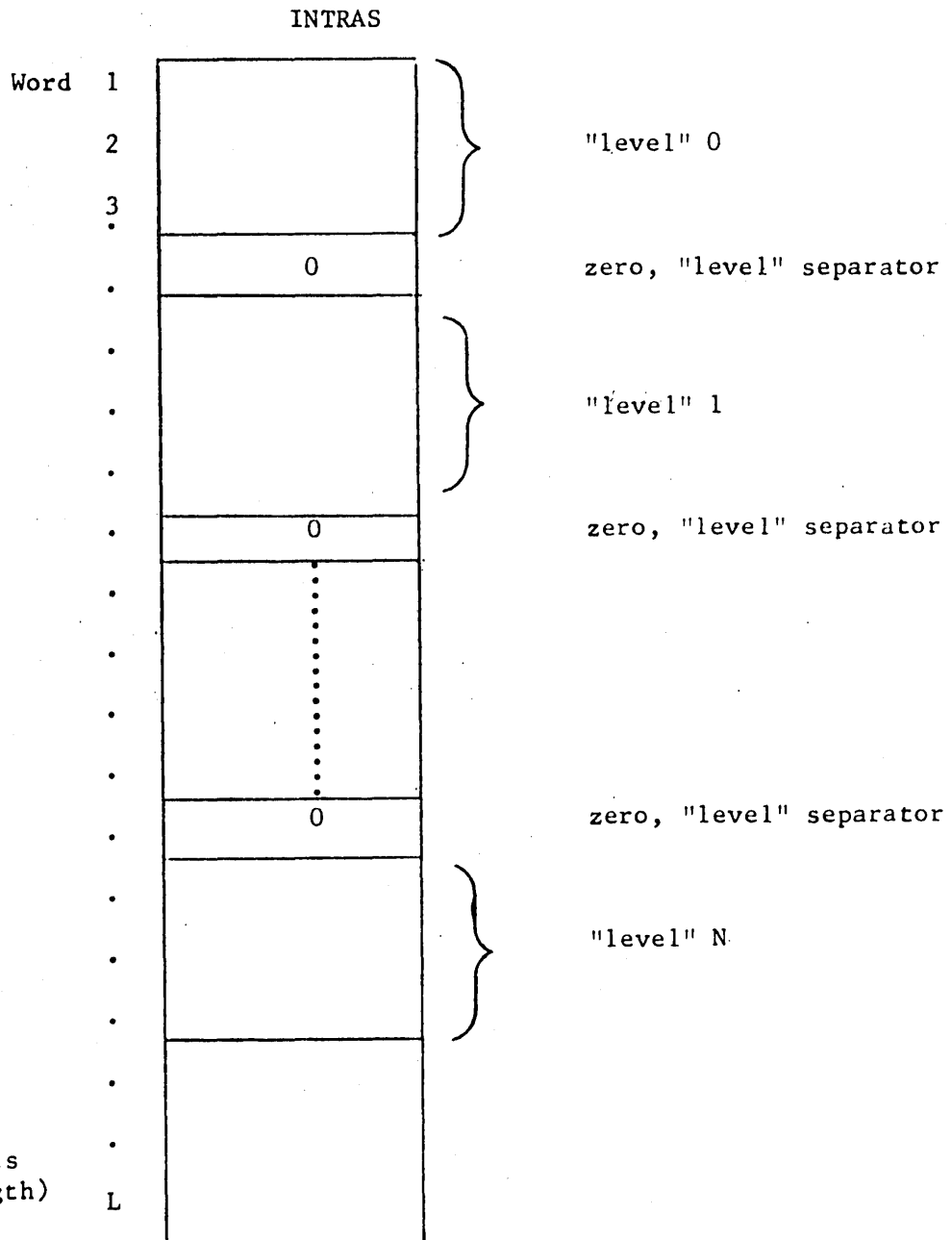
INTRAM

INTRAM is a routine used in conjunction with ACP to generate instructions.

INTRAM, basically, updates the one-dimensional table INTRAX. The major index to the table, INTRAX, is initialized to 0 by PHASE4 prior to entry of each statement processor.

Format of the table is shown:

INTRA-STATEMENT TEMPORARY STORAGE TABLE



(Where L is table length)

Elements are of three types depending upon the type of storage they identify. Information within the elements is generally organized from bottom to top since updating of the table proceeds downward but information is scanned and introduced in a backward direction.

A. Temporary Storage Element

Word 1	P
Word 2	N

where: P is the symbol table pointer of the assigned temporary storage.  
 N is the element designator.

If N = 3 the temporary storage was assigned to contain the results of a computation whose operator was not inverse.  
 N = 4 the temporary storage was assigned to contain the results of a computation whose operator was inverse.

B. Operand Element

Word 1	Ps
Word 2	A
Word 3	P
Word 4	N

where: P is the symbol table pointer of the variable.  
 A is the additive of the base address where the operand is an array element. A = 0 if additive is 0 or if there is no additive.  
 Ps is the symbol table pointer of the operand subscript where the operand is an array element. Ps = 0 if there is no subscript.  
 N is the element designator.

If N = 5 the operand was not an inverse.  
 N = 6 the operand was an inverse.

C. Calling Sequence Label Element

Word 1	P
Word 2	N



DOCUMENT CLASS IMS PAGE NO 4-107  
 PRODUCT NAME 170 Mass Storage FORTRAN  
 PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

where: P is a symbol table pointer assigned previously by the subscript processor to label the cell containing the results of an address computation which must be stuffed into a calling sequence.  
 N = 7 or 8.

D. Constant (non-symbol table pointer) Calling Sequence Element

Word 1	C	where C is a constant
Word 2	N	N = -7

Parameters

Parameters are common locations KNTROL, IPTR, IADDIT, ISBSCP, and KINTYP.

INTRAM performs 10 different functions depending upon the value of the controlling parameter KNTROL which can assume values -7 and +1 through +11. If KNTROL = ...

1. INTRAS table is updated to next level:
  - a. INTRAX is incremented by 1
  - b. INTRAS (INTRAX) set to 0
2. INTRAS table is updated back to next lower level.
  - a. INTRAX is decremented by 1
3. Temporary storage element of type  $N = \pm 3$  is added to current level of table.
  - a. INTRAX is incremented by 1 and INTRAS (INTRAX) set to IPTR.
  - b. INTRAX is incremented by 1 and INTRAS (INTRAX) set to KNTROL.
4. Temporary storage element of type  $N = \pm 4$  is added to current level of table (as per  $N = 3$ ).
5. Operand element of type  $N = 5$  is added to current level of table.

CONTROL DATA CORPORATION  
LA JOLLA RESOURCE CENTER DIVISION

DOCUMENT CLASS IMS PAGE NO. 4-108  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. COO5\*3.1 A/B MACHINE SERIES 1700

- a. INTRAX is incremented by 1 and INTRAS {INTRAX} set to ISBSCP.
  - b. INTRAX is incremented by 1 and INTRAS {INTRAX} set to IADDIT.
  - c. INTRAX is incremented by 1 and INTRAS {INTRAX} set to IPTR.
  - d. INTRAX is incremented by 1 and INTRAS {INTRAX} set to KNTR0L.
6. Operand element of type N = 6 is added to current level of table {as per N = 5}.
- 7.8. Calling sequence label element is added to current level of table {as per N = 3}.
- 7. Calling Sequence constant is added to current level of table {as per N = 3}.
9. Dump instruction string, add/subtract mode. The INTRAS table is scanned in a backward direction beginning at the current position of the index INTRAX at the last word of the latest entry added to the current level.
- a. As each element is encountered, an add or subtract instruction is generated which references the operand designated in that element as per the element designator specification of non-inverse respectively. ADD, FAD, or DAD; SUB, FSB, or DSB is decided upon by interrogation of parameter KINTYP. KINTYP is set to 1, 2 or 3 as computation is integer, real or double precision respectively.
  - b. If the element was temporary storage, and element designator was positive valued, that temporary storage is released by call to the temporary storage allocation routine, TSALOC. {Negative valued element designator indicates that the temporary storage was assigned for a subexpression and therefore should not be released until the use count has been decremented to 0.}
  - c. If the element was an operand element containing a non-zero subscript pointer, level 0 is searched for a temporary storage element containing that pointer. If found, that temporary storage is also released through TSALOC. {A subscript pointer may be a temporary storage yet not in level 0 because it contains a subexpression result and must not be released.} Upon completion of control function 8, INTRAX points to the 0 level break between the level just processed and the preceding level of INTRAS.

CONTROL DATA CORPORATION  
LA JOLLA RESOURCE CENTER DIVISION

DOCUMENT CLASS IMS PAGE NO. 4-109  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

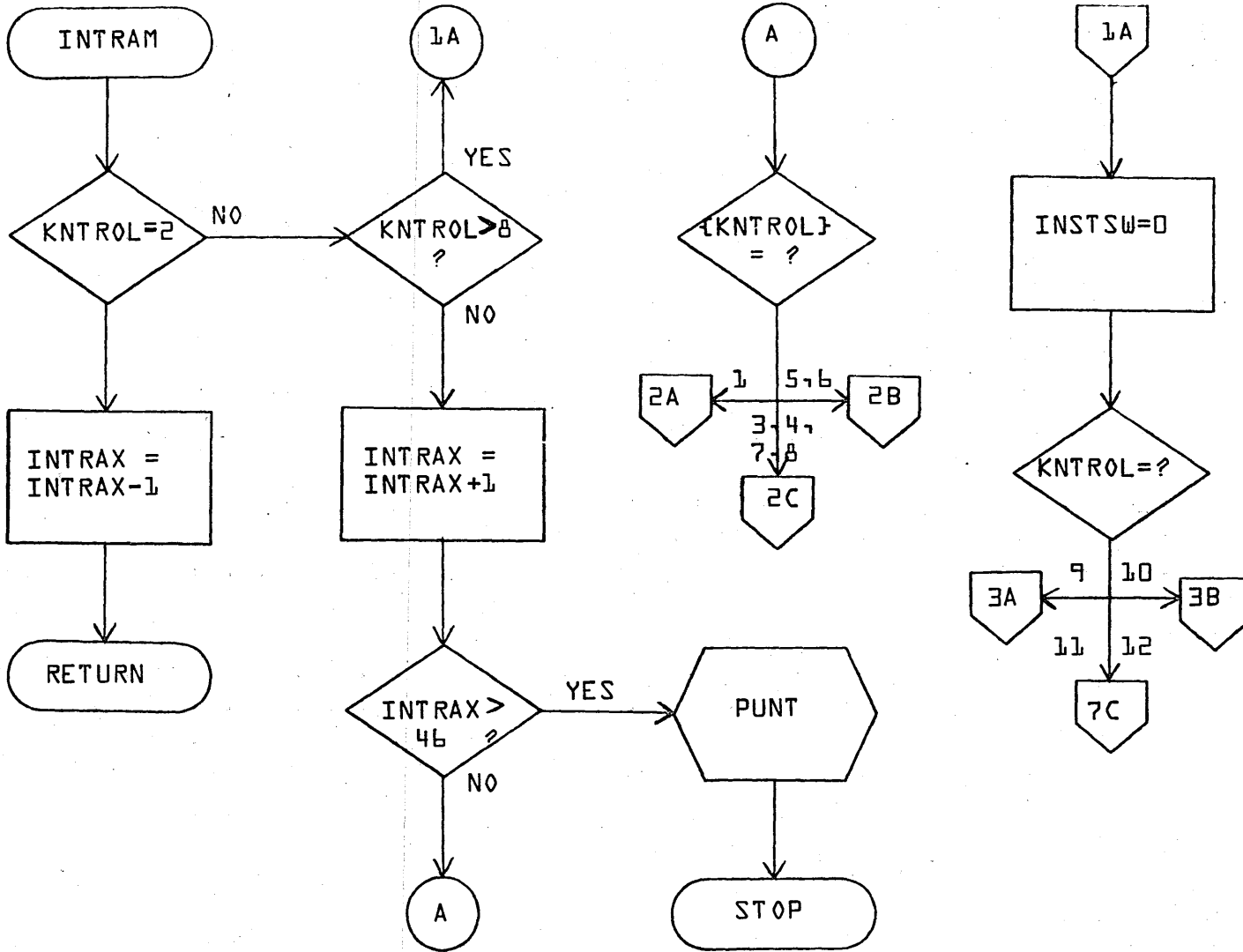
10. Dump instruction string, multiply/divide mode. Exactly the same procedure as 8 with MUI/FMU/DMU, DVI/FDV/DDV, with following exceptions:
- a. Constant, integer operands are checked to determine whether they should be replaced by shifts, and shifts substituted where appropriate.
  - b. Instructions for A-Q sign extension are appended preceding integer divides.
  - c. Check is made to see if code is being generated for pass 1 of a floating multiply. If so, inverses are ignored and all FDV {or DDV} instructions become FMU's {or DMU's}.
11. Dump parameter string. The INTRAS table is scanned in a backward direction beginning at the current position of the index INTRAX at the last word of latest entry added to the current level. As each element is encountered a calling sequence entry is created.
- a. Temporary Storage Elements. An address constant is generated for the temporary storage as appropriate to run-anywhere switch. The temporary storage is then released by call to TSALOC.
  - b. Operand Elements. An address constant is generated for the temporary storage as appropriate to run-anywhere switch.
  - c. Calling Sequence Label Element. A label is generated as per the pointer in the element. This is followed by a generated BSS 1.
  - d. Calling Sequence Constant. A constant is generated.
  - e. IREF bit is set in symbol table to show that operands have been referenced in a calling sequence.
12. Backup over Parameter String. Function is exactly as in 11 except no actual code is output. This is performed whenever a function is done in-line and therefore, there is no need to generate address constants for the parameter list.

A

B

C

D



**CONTROL DATA CORPORATION**  
 SOFTWARE DOCUMENT

SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	INTRAM	PAGE 1 OF 9		PROJECT MGR.			
NUMBER	ISSUE DATE			PROJECT NAME			
DRAWN BY	DATE			TASK NO.			
				TASK NAME			

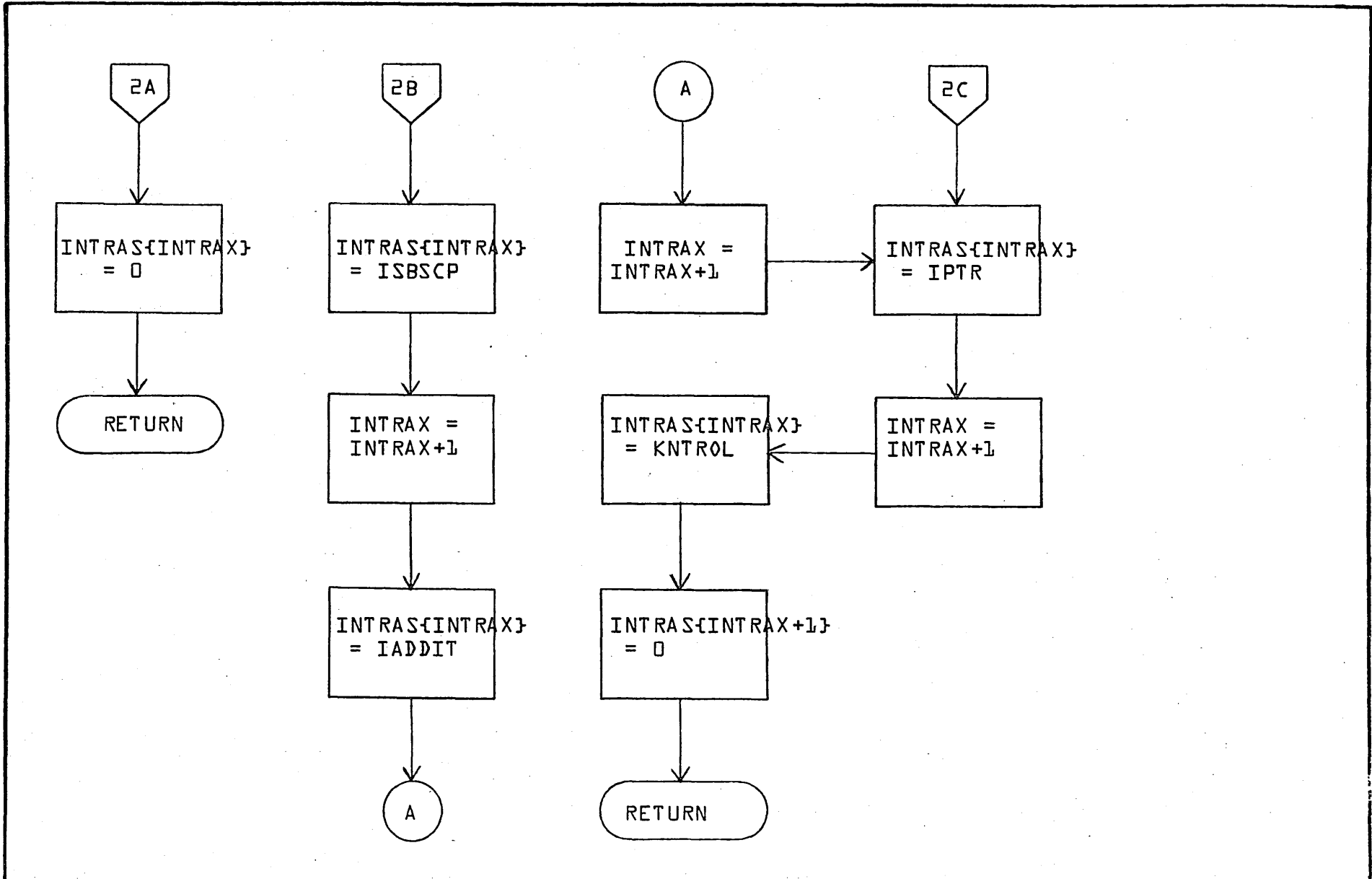
4-1770

A

B

C

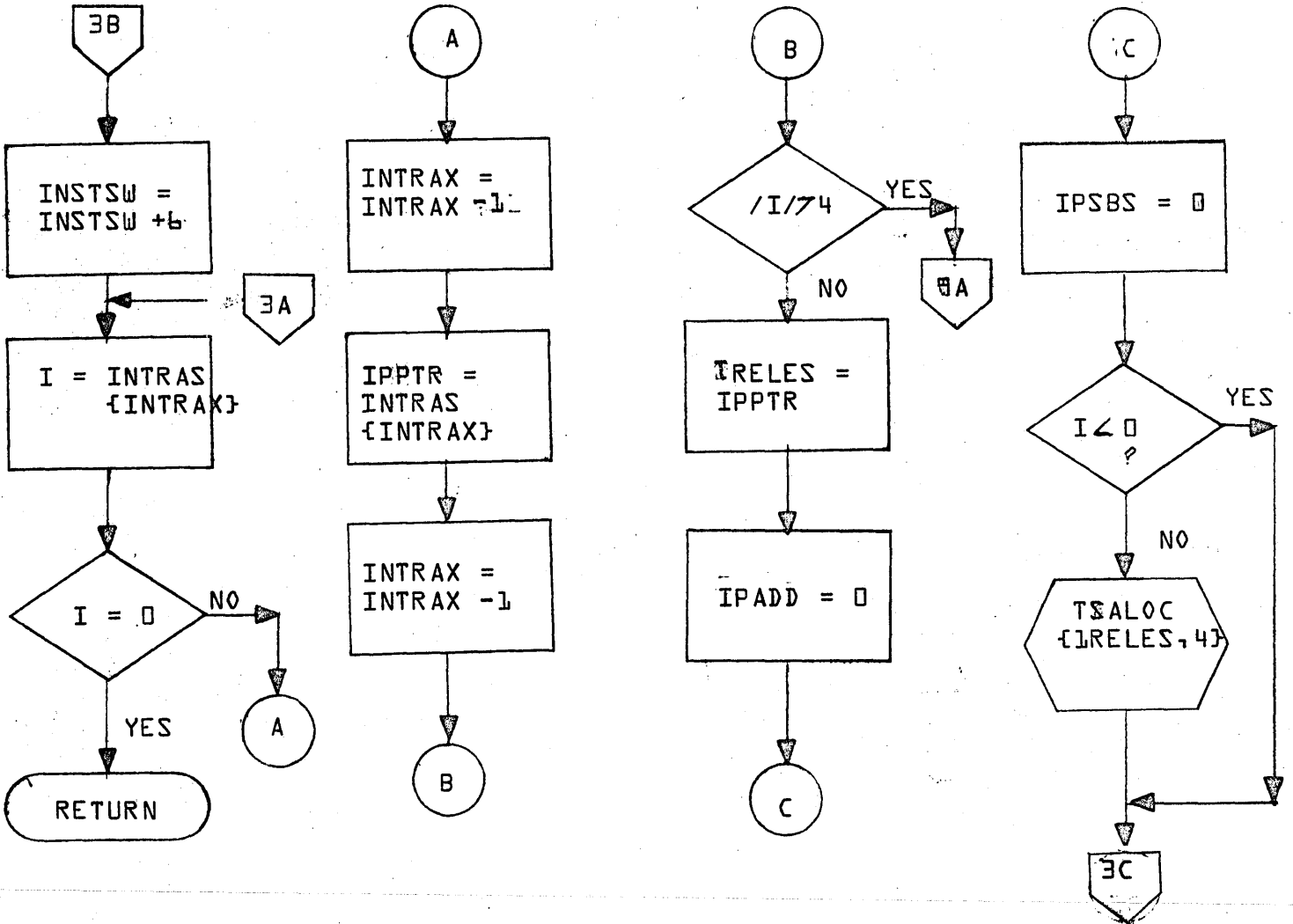
D



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	INTRAM	PAGE 2 OF 9		PROJECT MGR.			
	NUMBER	ISSUE DATE			PROJECT NAME			
	DRAWN BY	DATE			TASK NO.			
					TASK NAME			



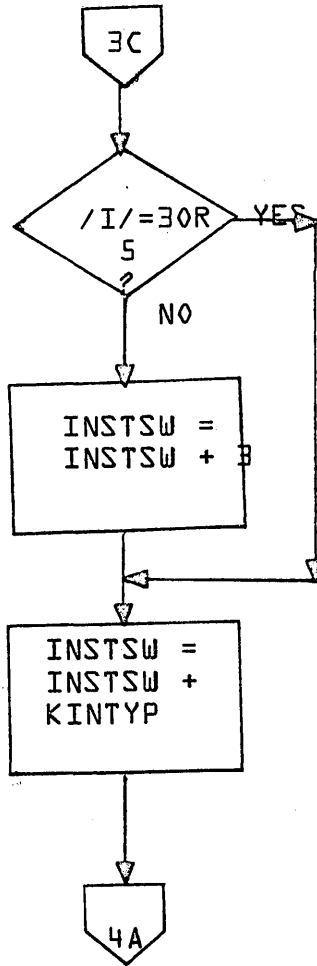
LA JOLLA FACILITY



TITLE			DRG. NO.	
INTRAM			REVISION	
DRAWN BY	PROJ.	DATE	SHEET	OF
CSD 203			3	9



LA JOLLA FACILITY



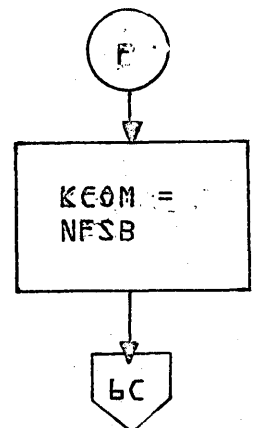
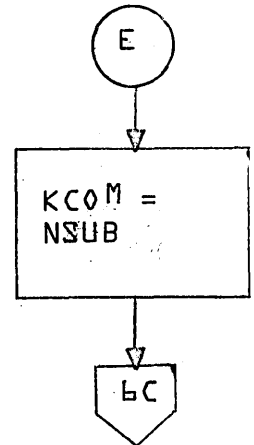
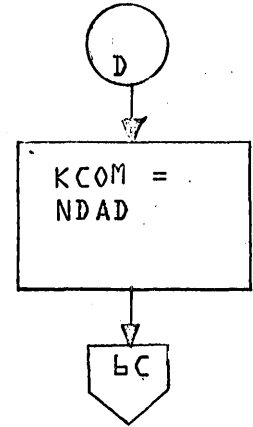
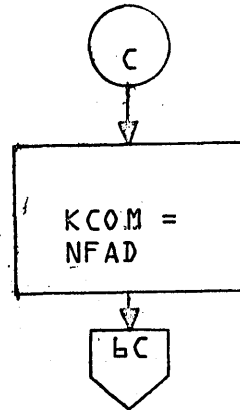
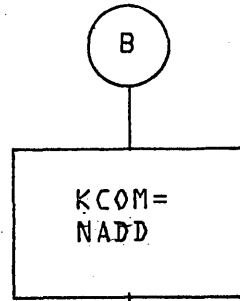
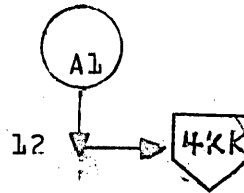
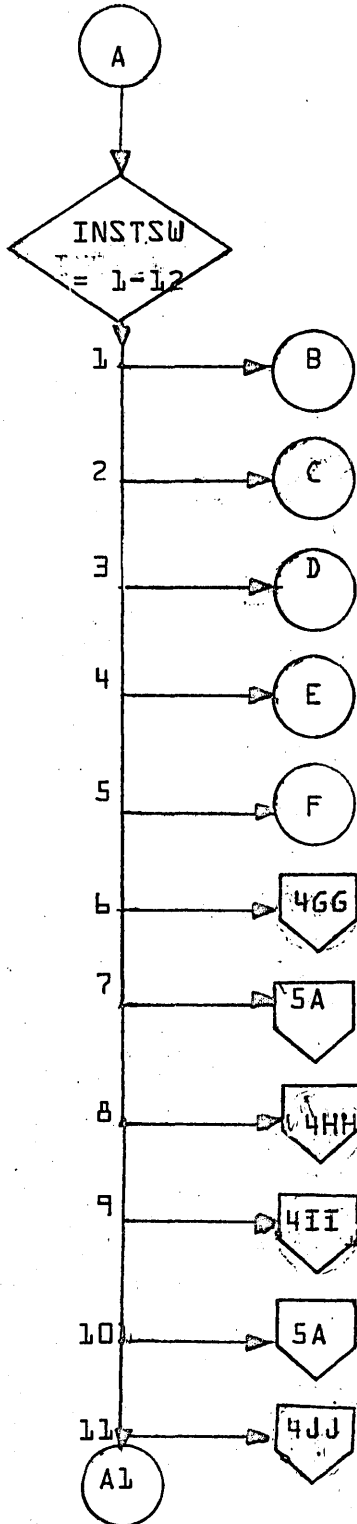
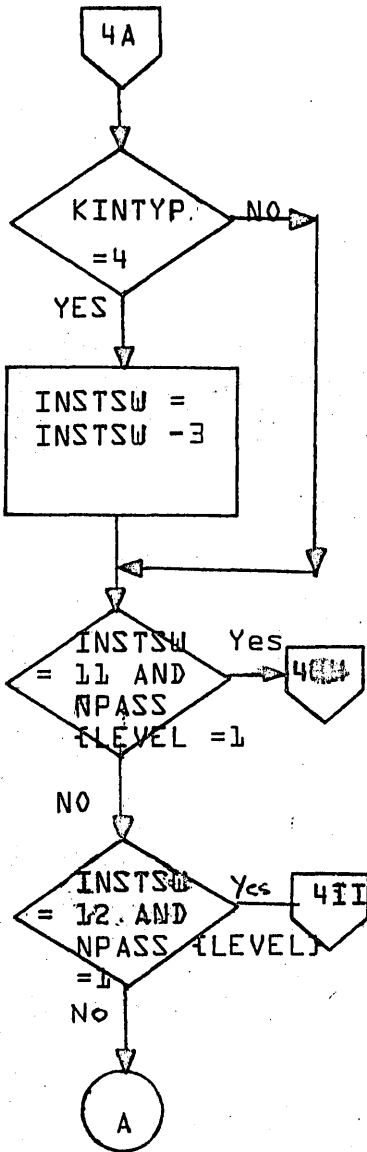
TITLE			INTRAM		DRG. NO.	
DRAWN BY			PROJ.		REVISION	
CSD 203			DATE		SHEET 3A OF 9	







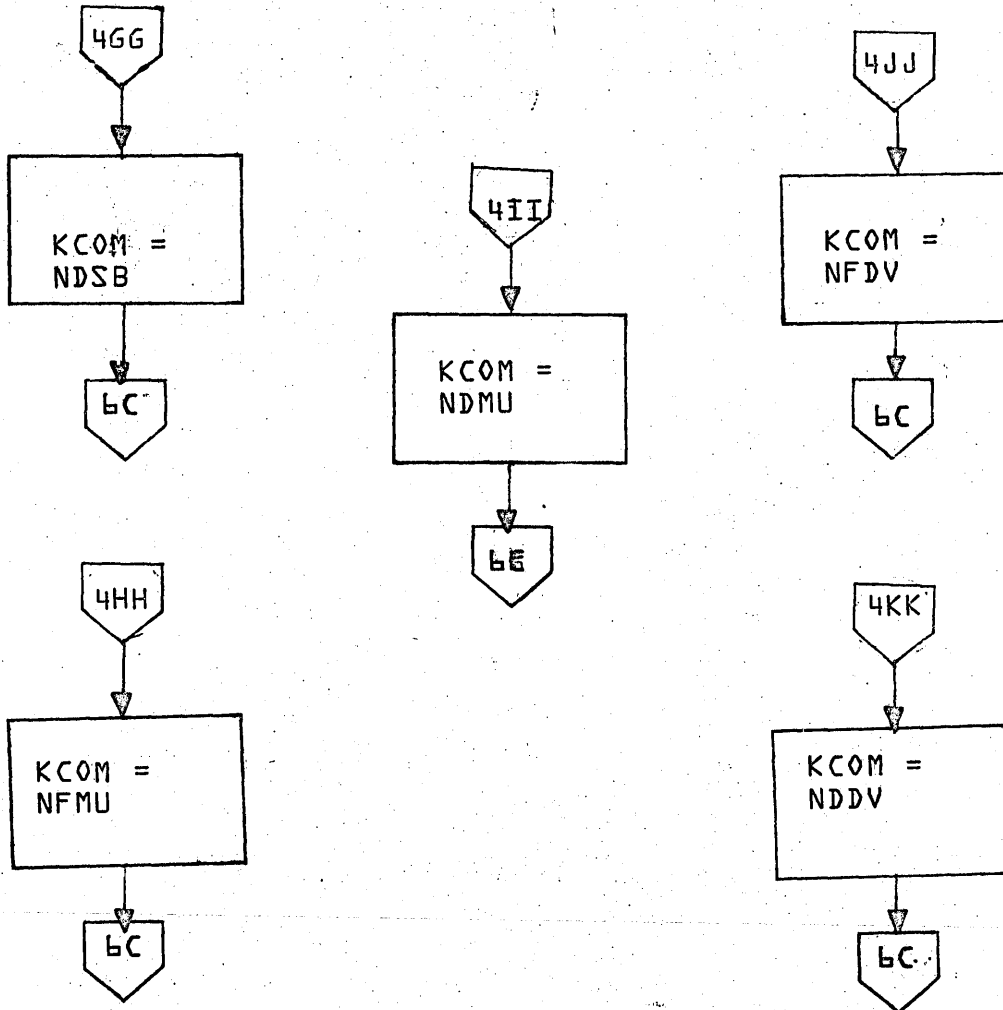
LA JOLLA FACILITY



TITLE			DRG. NO.		
INTRAM			REVISION		
DRAWN BY	PROJ.	DATE	SHEET	4	OF 9
CSD 203					



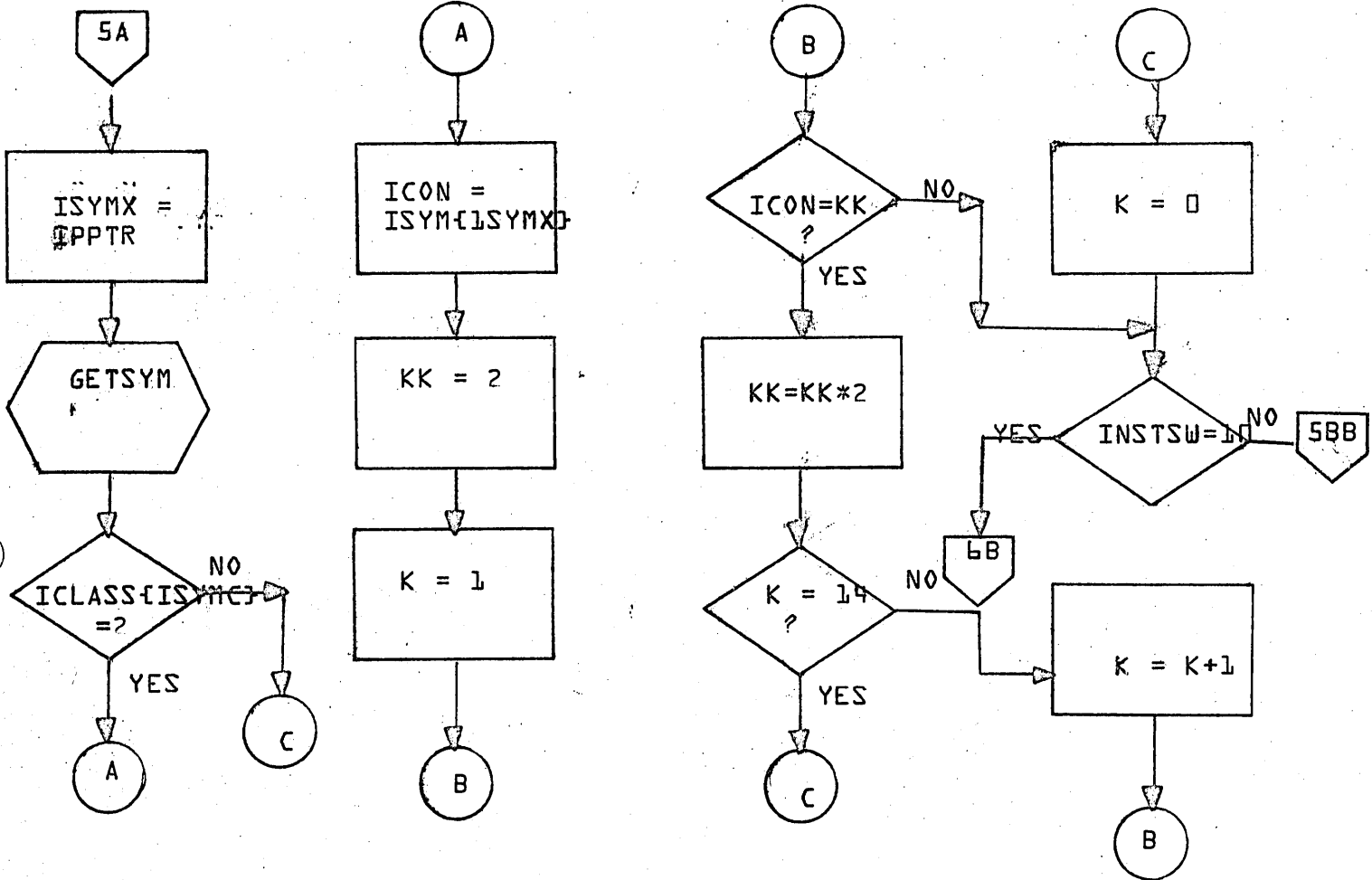
LA JOLLA FACILITY



TITLE		INTRAM		DRG. NO.	
				REVISION	
DRAWN BY	PROJ.	DATE	SHEET 4A OF 9		



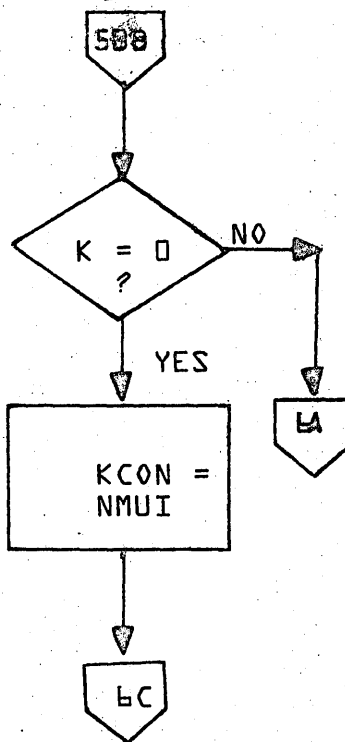
LA JOLLA FACILITY



TITLE		INTRAM		DRG. NO.	
DRAWN BY		PROJ.		REVISION	
CSD 203		DATE		SHEET 5 OF 9	



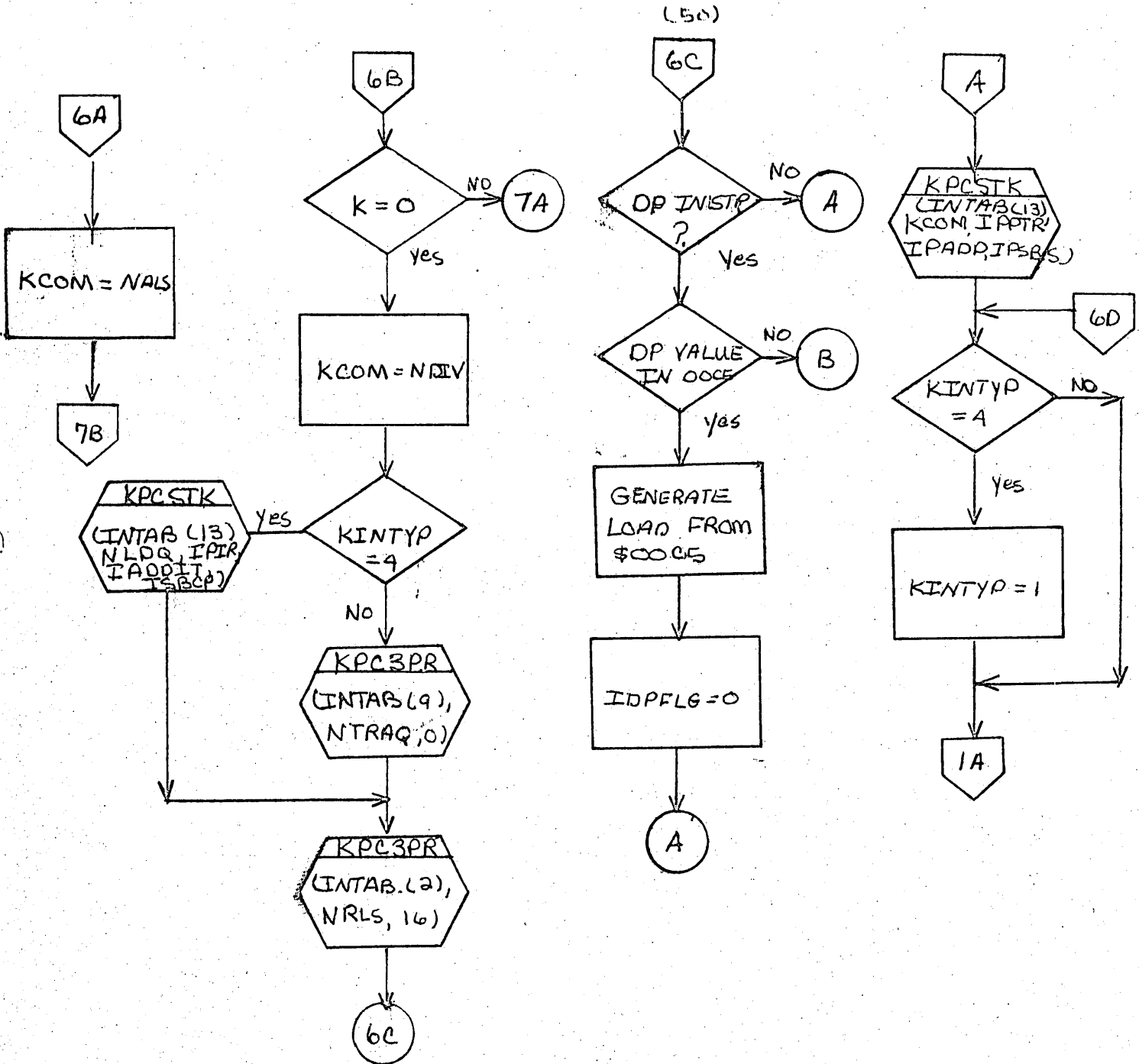
LA JOLLA FACILITY



TITLE		INTRAM		DRG. NO.	
				REVISION	
DRAWN BY	PROJ.	DATE	SHEET 5a OF 9		



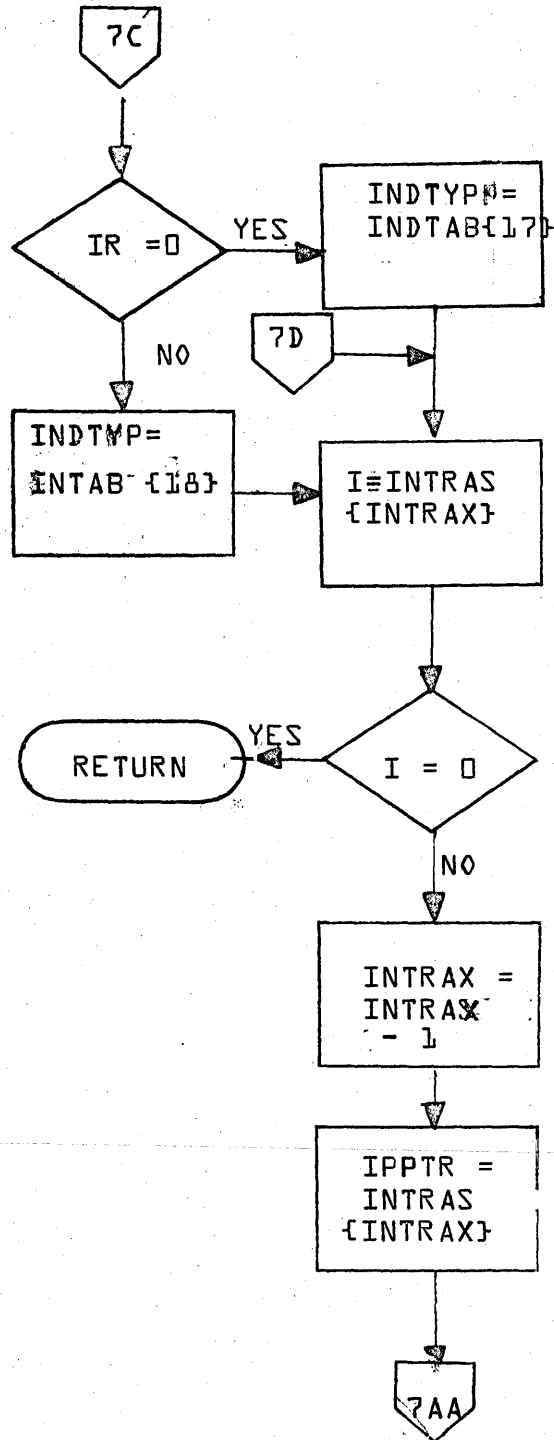
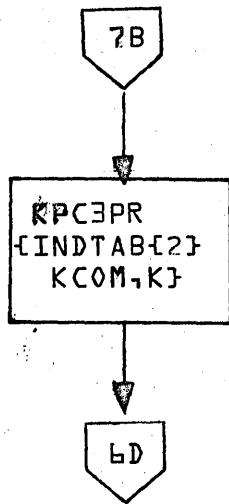
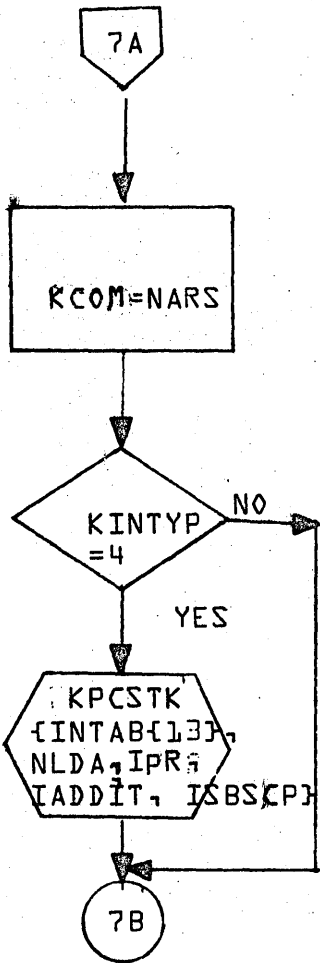
LA JOLLA FACILITY



TITLE		INTRAM		DRG. NO.	
				REVISION	
DRAWN BY	PROJ.	DATE	SHEET	6	OF 9
CSD 203					



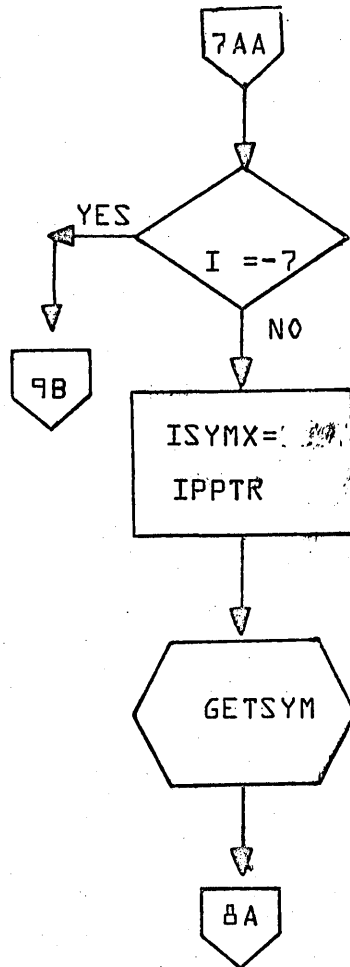
LA JOLLA FACILITY



TITLE		INTRAM		DRG. NO.	
				REVISION	
DRAWN BY	PROJ.	DATE	SHEET	7	OF 9
CSD 203					



LA JOLLA FACILITY



TITLE INTRAM			DRG. NO.
			REVISION
DRAWN BY	PROJ.	DATE	SHEET 7A OF 9

C

C

C

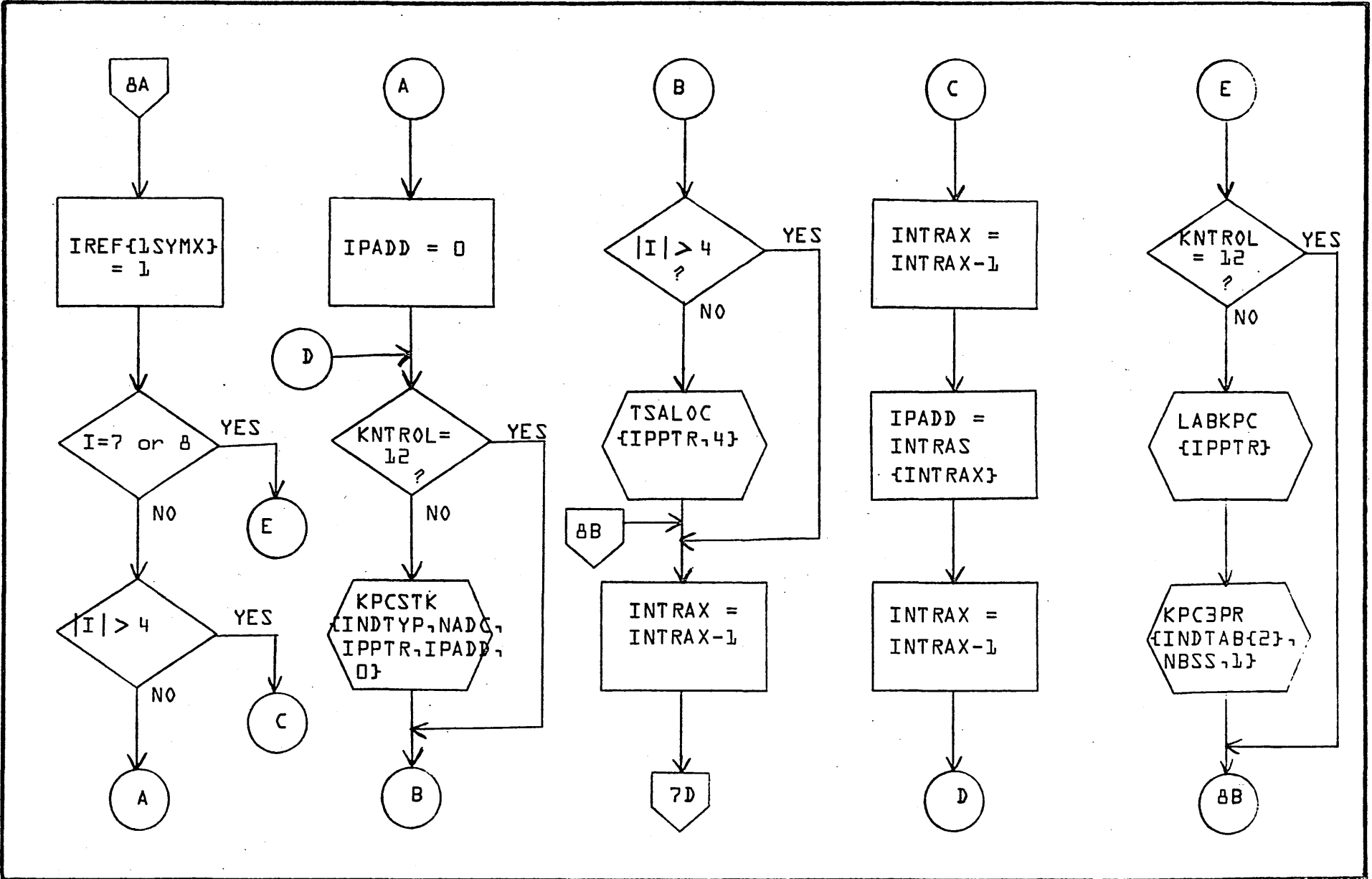


A

B

C

D



**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

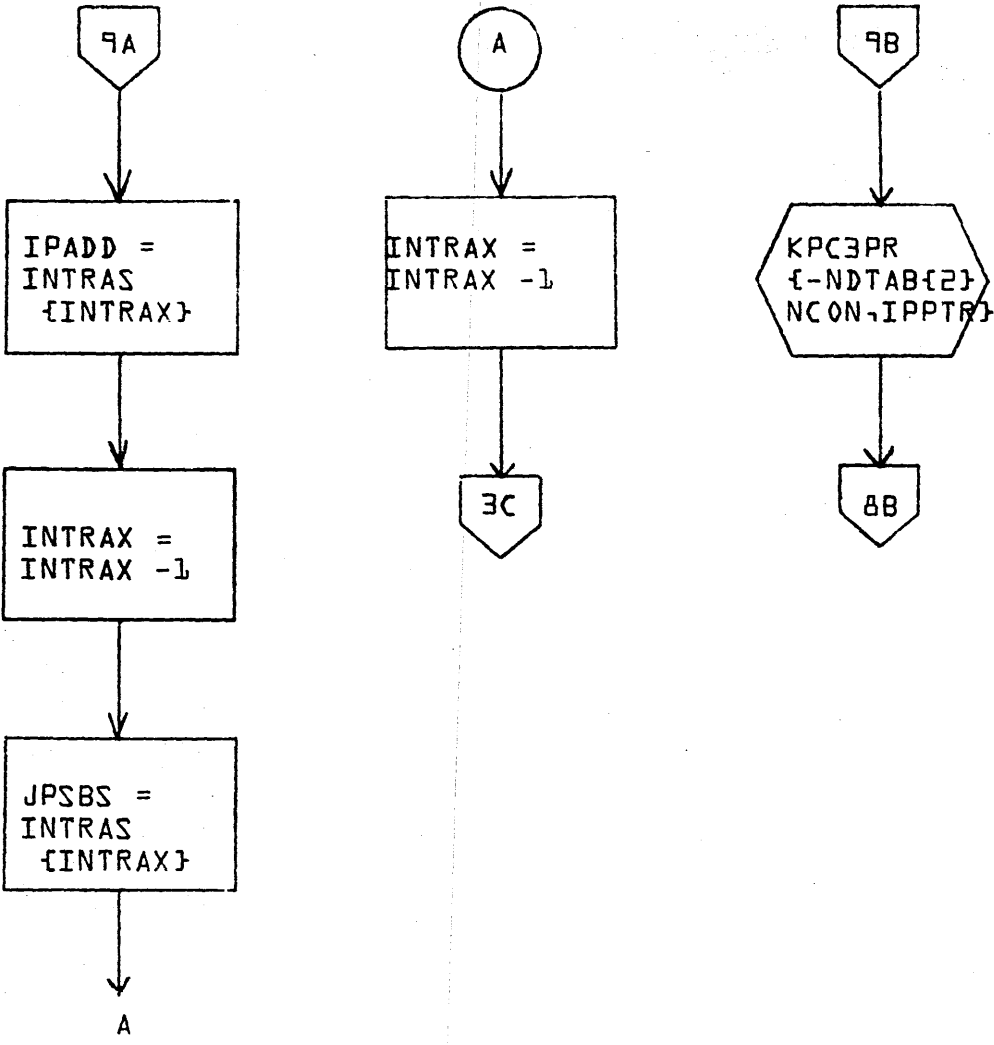
DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	INTRAM			PROJECT MGR.			
PAGE 8 OF 9				PROJECT NAME			
NUMBER	ISSUE DATE			TASK NO.			
DRAWN BY	DATE			TASK NAME			

A

B

C

D



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	Intram			PROJECT MGR.			
	PAGE 9 OF 9				PROJECT NAME			
	NUMBER	ISSUE DATE			TASK NO.			
	DRAWN BY		DATE		TASK NAME			

4-118

DOCUMENT CLASS IMS PAGE NO. 4-119  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

4.4.8 INXRST

INXRST is a short subroutine to generate the instruction sequence which restores the Q and I index registers. INXRST is called by END and *PHASEB*

INXRST

KPC3PR  
(INTAB(S),  
KLDQ, KFFSAV)

KPC3PR  
(INDTAB(10),  
KSTQ, 255)

KPC3PR  
(INDTAB(S),  
NLDQ, KQSAV)

RETURN

CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE   
FLOWCHART   
DECISION TABLE   
OTHER

DOCUMENT CLASS *IMS* MACH. TYPE *1700*  
DOCUMENT TITLE *INXRST*  
PAGE *1* OF *1*  
NUMBER ISSUE DATE  
DRAWN BY DATE

PROJECT NO.	REV	APPROVED	DATE
PROJECT MGR.			
PROJECT NAME			
TASK NO.			
TASK NAME			

DOCUMENT CLASS IMS PAGE NO. 4-121  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

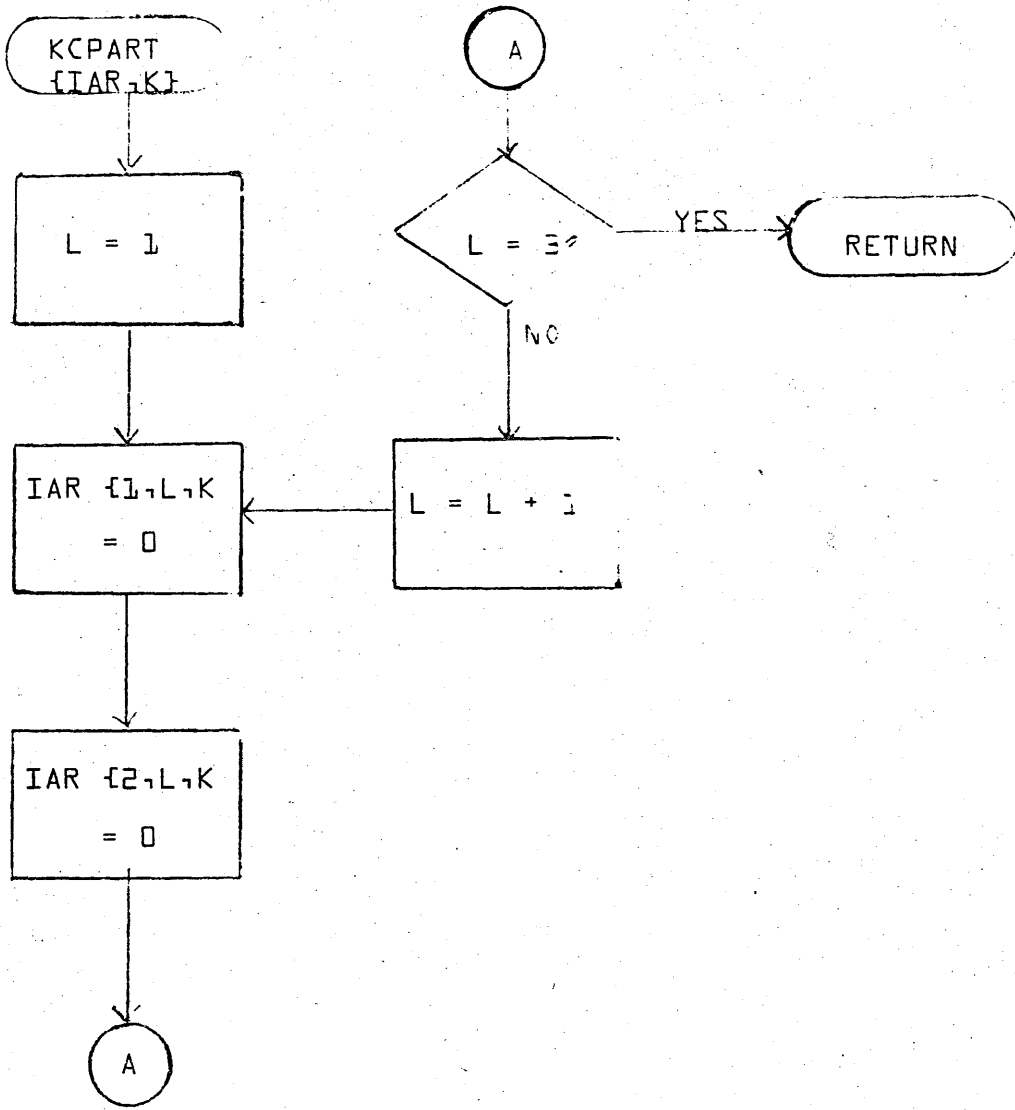
4.4.9 KCPART

KCPART is used in conjunction with KPCSTK in keeping a memory of the 'current' contents of the accumulator, single precision floating point pseudo accumulator, and double precision floating point pseudo accumulator. KCPART is called to clear either the accumulator portion, sp pseudo accumulator portion, or dp pseudo accumulator portion of the accumulator record table KART.

PARAMETERS:

IAR address of table KART  
K =1 clear accumulator record  
=2 clear sp pseudo accumulator record  
=3 clear dp pseudo accumulator record

A  
B  
C  
D



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	KCPL="			PROJECT MGR.			
	PAGE 1 OF 1				PROJECT NAME			
	NUMBER	ISSUE DATE		TASK NO.				
	DRAWN BY	DATE		TASK NAME				

DOCUMENT CLASS \_\_\_\_\_ IMS \_\_\_\_\_ PAGE NO 4-123  
PRODUCT NAME 1.0 Mass Storage FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

## 4.4.10 KOUTPT

KOUTPT is the routine called to write a PHASE4 pseudo instruction output record. It is called upon for all pseudo instructions but not for the special records such as DATA and FORMAT.

## Steps:

1. Transfer the number of words in this record to OUTBUF(1) (the word in any output record which normally designates the number of words in the record). This value is obtained by subtracting 1 from KOBX the output buffer index.
2. Call WRITE.

A

B

C

D

KOUTPT



OUTBUF{1} =  
KOBX - 1



WRITE  
{ISCR0, 0,  
OUTBUF {1},  
OUTBUF{1}}



RETURN

<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	KOUTPT			PROJECT MGR.			
	PAGE 1 OF 1				PROJECT NAME			
	NUMBER	ISSUE DATE			TASK NO.			
	DRAWN BY	DATE			TASK NAME			

421-4



DOCUMENT CLASS \_\_\_\_\_ TMS \_\_\_\_\_ PAGE NO. 4-12  
PRODUCT NAME 1700 Mass Storage FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

4.4.11 KPC3PR and LABKPC

Both of these routines are simply a single call to KPCSTK. Most calls to KPCSTK are made through KPC3PR or LABKPC to reduce the number of parameters, hence the number of words in the call.

KPC3PR handles all calls where the last two of the five KPCSTK parameters are always 0. LABKPC handles all calls to KPCSTK which are requests to output a label. (Only the second parameter varies from one call to the next).

A

B

C

D

KPC3PR  
{ILX,ILY,ILZ}

LABKPC  
{ILX}

KPCSTK  
{ILX,ILY,ILZ,  
0,0}

KPCSTK  
{INDTAB{1},  
ILX,0,0,0}

RETURN

RETURN

CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE

FLOWCHART

DECISION TABLE

OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	KPC3PR & LABKPC			PROJECT MGR.			
PAGE 1 OF 1				PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.					
DRAWN BY	DATE	TASK NAME					

4-126

DOCUMENT CLASS IMS PAGE NO 4-127  
PRODUCT NAME 1700 Mass Storage FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

4.4.12 LABLER

LABLER is called by various PHASEB subroutines to create a symbol table entry which represents a label, i.e., a FORTRAN generated label.

Parameters:

I - returns the symbol table pointer of the entry created by LABLER.

Flowchart:

A

LABLER  
{I}



B

KSYMGN{I}



C

ICLASS  
{ISYMX}  
= ?



D

RETURN

<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	LABLER		PROJECT MGR.				
	PAGE 1 OF 1				PROJECT NAME			
	NUMBER	ISSUE DATE		TASK NO.				
	DRAWN BY		DATE		TASK NAME			

4-128

DOCUMENT CLASS IMS PAGE NO. 4-12  
PRODUCT NAME 1700 Mass Storage FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

#### 4.4.13 ACP

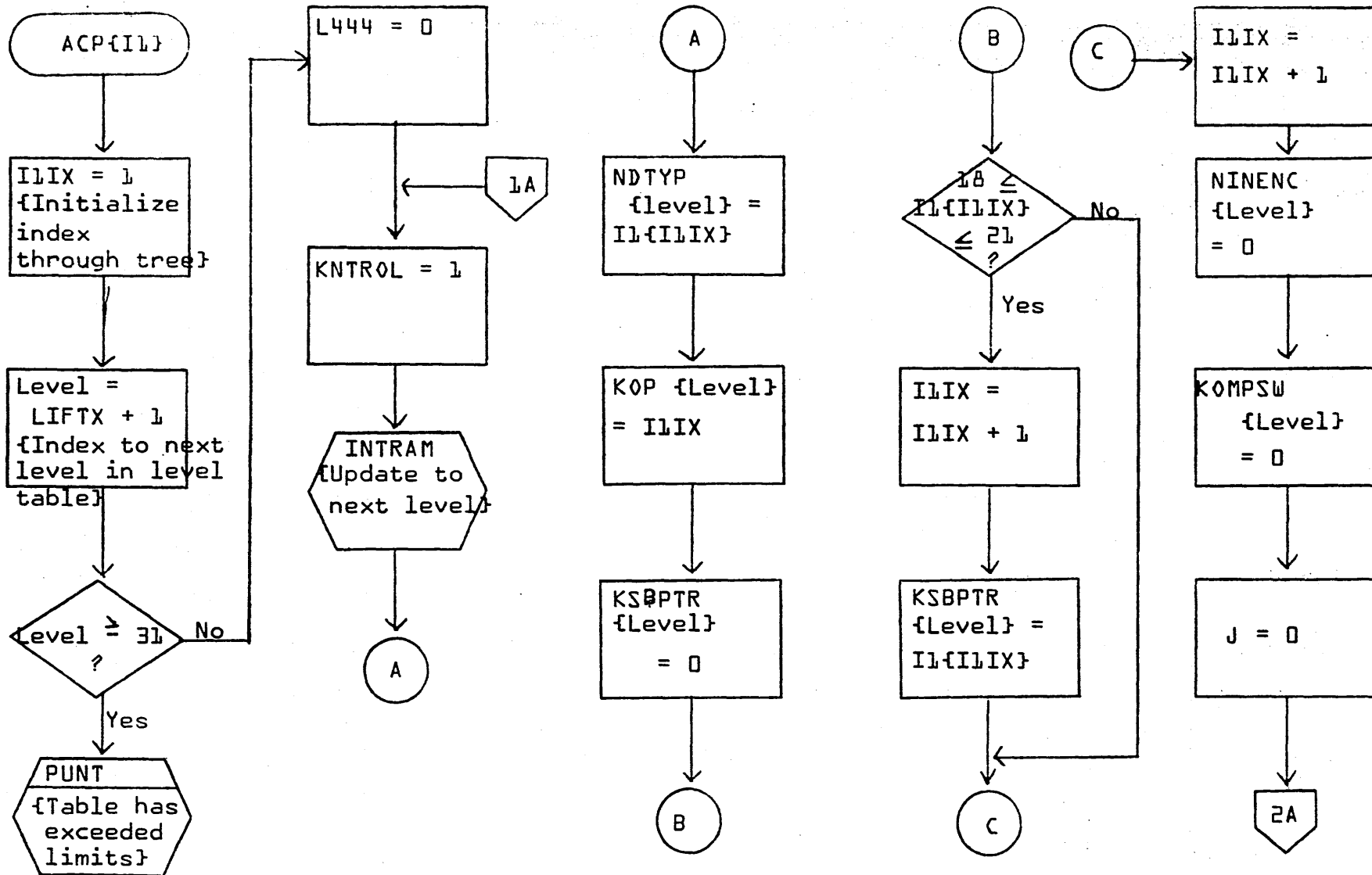
ACP is the routine which actually processes arithmetic trees. ACP scans each tree and generates commands. ACP is called by ASUPER and SUBPR3. ASUPER and SUBPR3 are in turn called by individual statement processes. ACP calls INTRAN or KPCSTK directly to output commands.

A

B

C

D



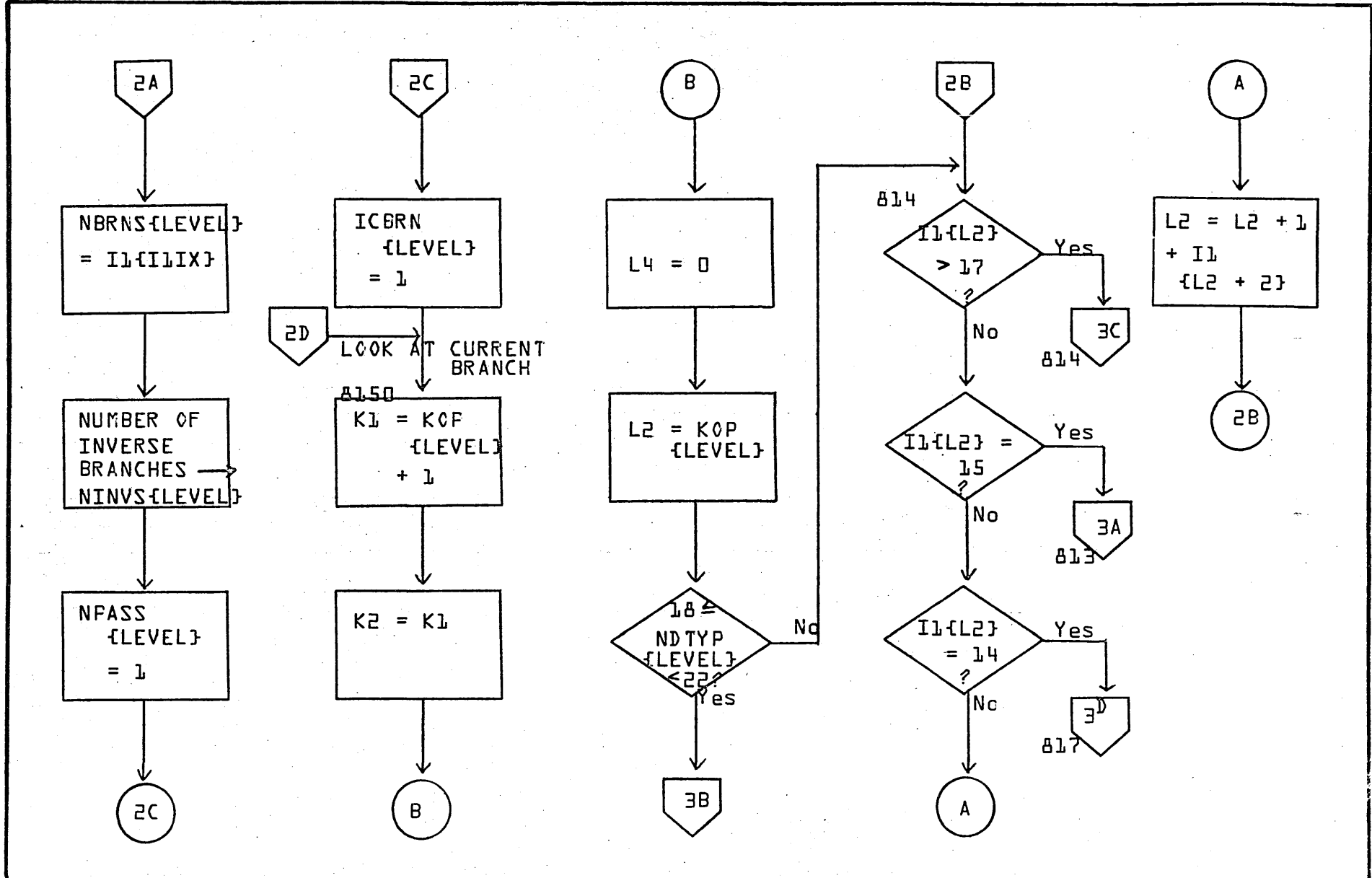
CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE   
FLOWCHART   
DECISION TABLE   
OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700
DOCUMENT TITLE	ACP	PAGE 1 of 17	
NUMBER	ISSUE DATE	TASK NO.	
DRAWN BY	DATE	TASK NAME	

PROJECT NO.	REV	APPROVED	DATE
PROJECT MGR.			
PROJECT NAME			
TASK NO.			
TASK NAME			

DET-130



**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

SAMPLE CODE

FLOWCHART

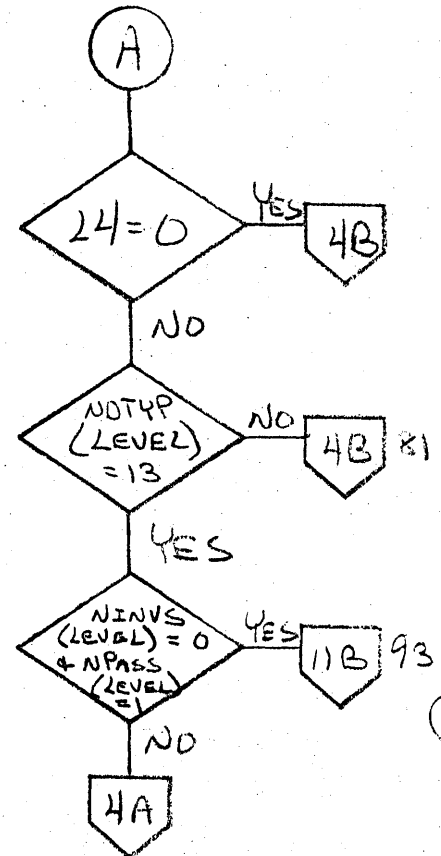
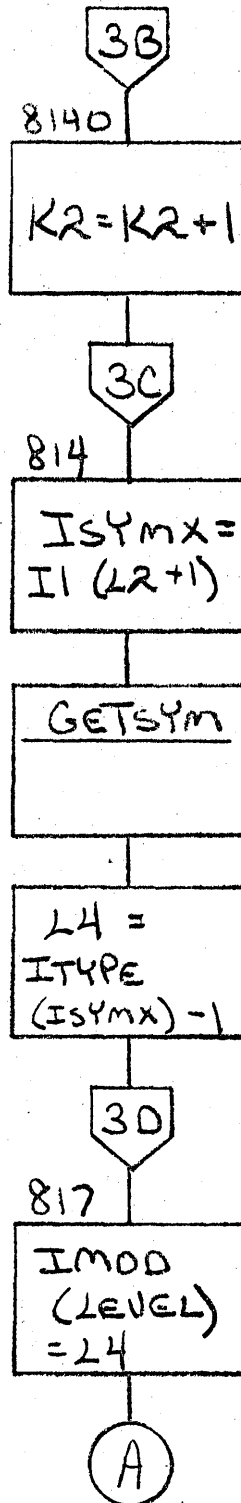
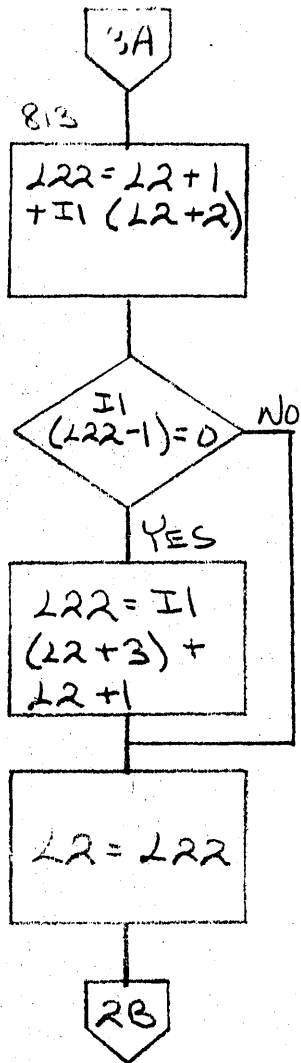
DECISION TABLE

OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	ACF	PAGE 2 OF 17		PROJECT MGR.			
NUMBER	ISSUE DATE			PROJECT NAME			
DRAWN BY	DATE			TASK NO.			
				TASK NAME			



LA JOLLA FACILITY



TITLE		ACF		DRG. NO.	
DRAWN BY		PROJ.		REVISION	
DATE		SHEET 3		OF 17	

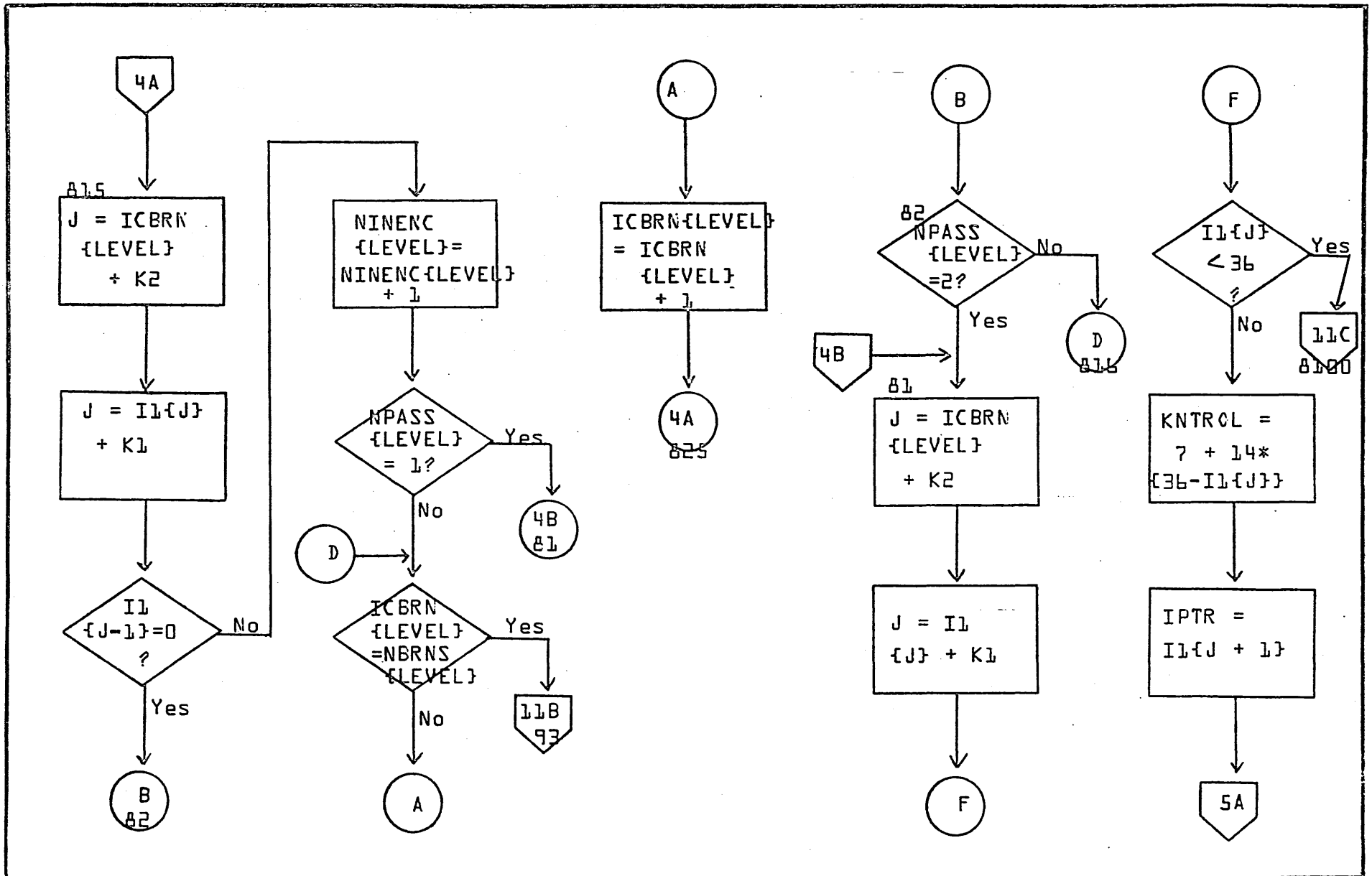


A

B

C

D



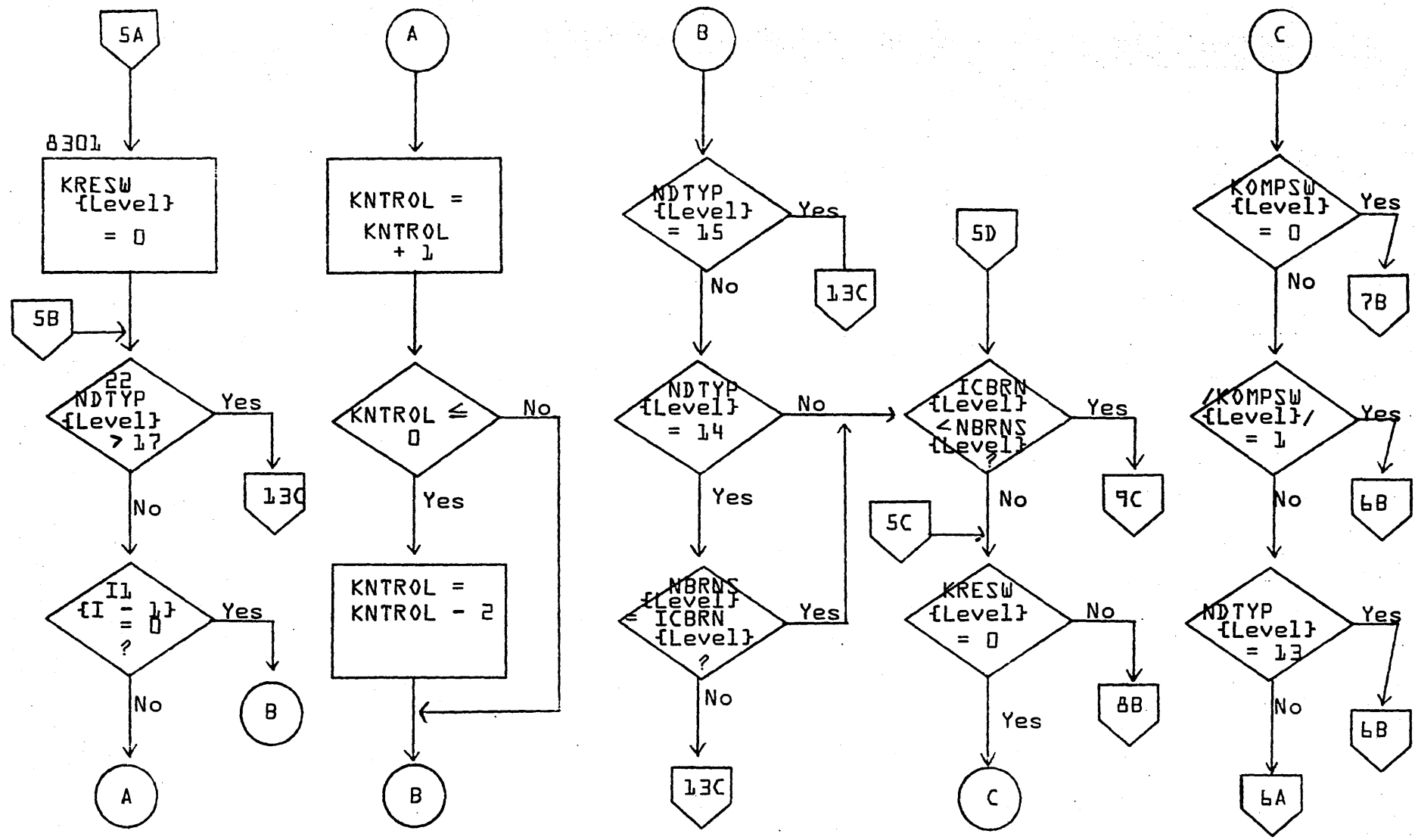
CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	ACP	PAGE 4 OF 17		PROJECT MGR.			
NUMBER	ISSUE DATE			PROJECT NAME			
DRAWN BY	DATE			TASK NO.			
				TASK NAME			

EEI-h

A  
B  
C  
D



**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

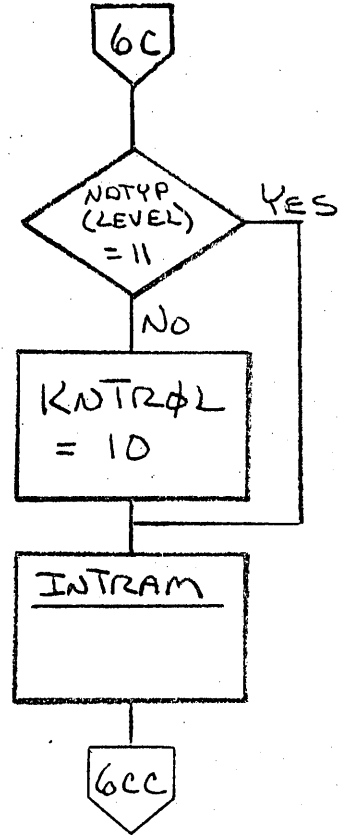
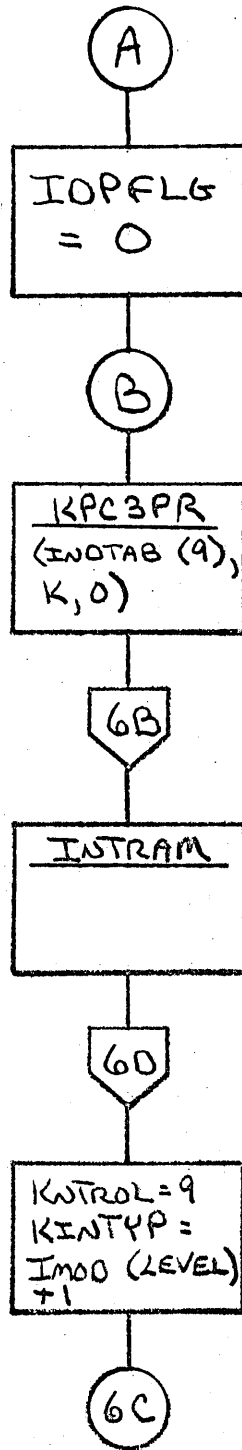
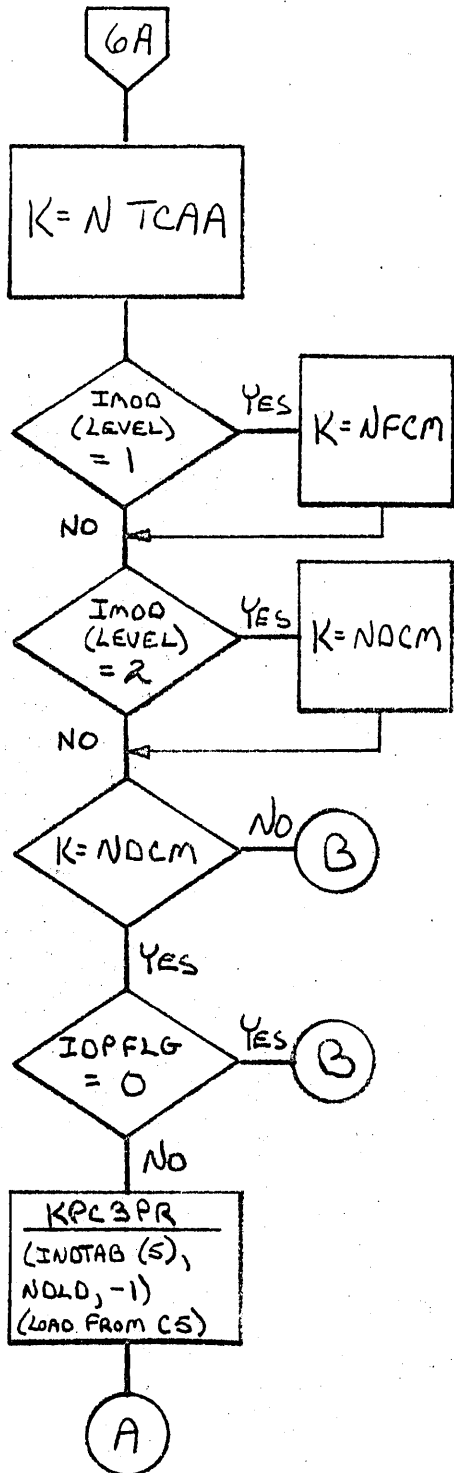
- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	ACP	PAGE 5 OF 17		PROJECT MGR.			
NUMBER	ISSUE DATE			TASK NO.			
DRAWN BY	DATE			TASK NAME			

hET-h



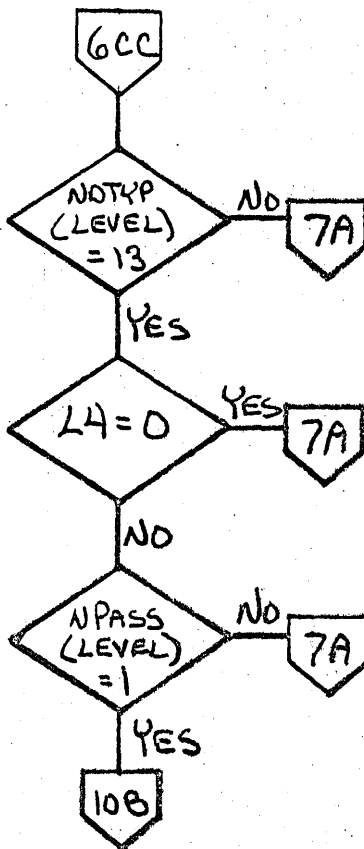
LA JOLLA FACILITY



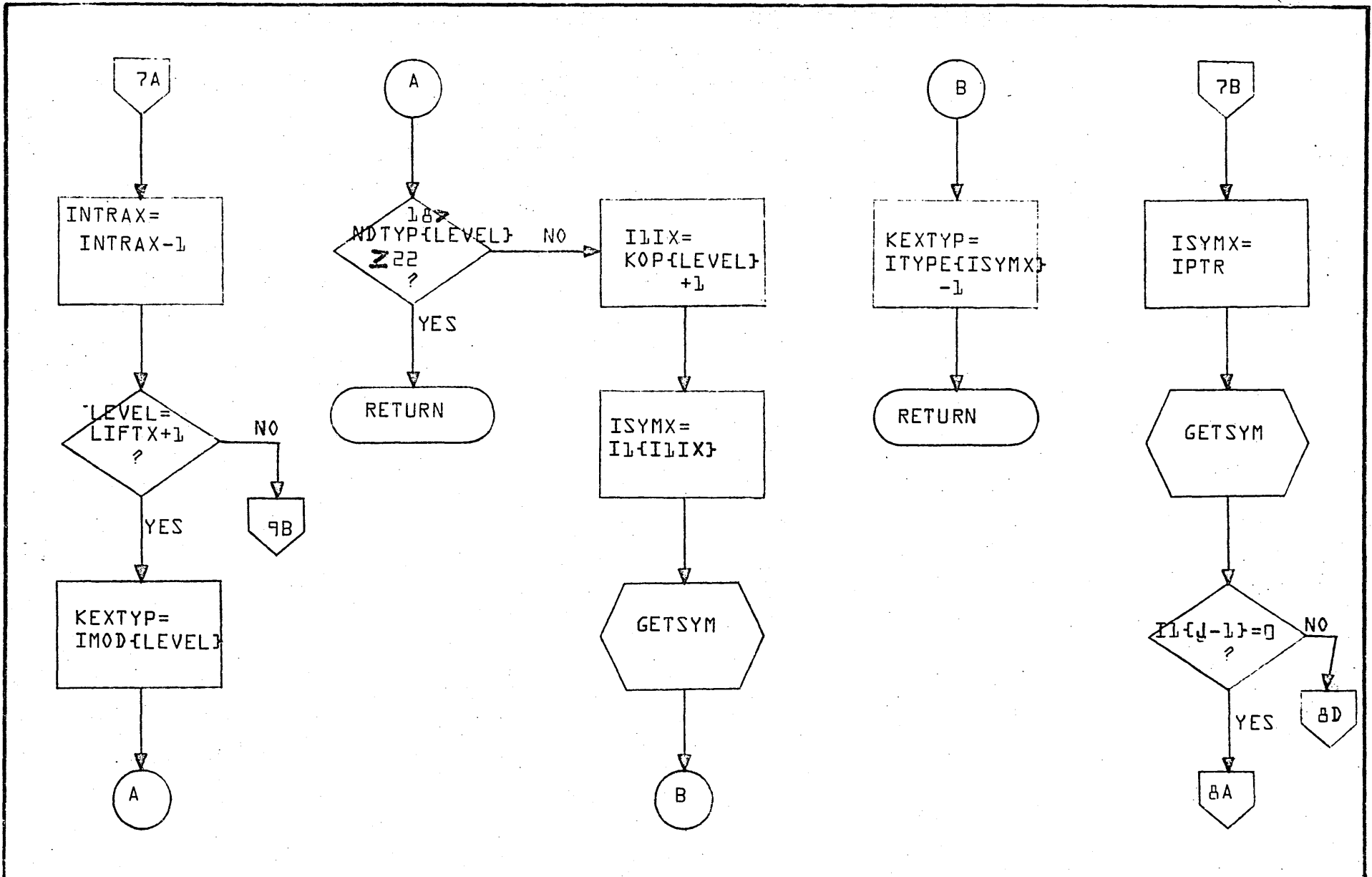
TITLE		ACF		DRG. NO.	
DRAWN BY		PROJ.		REVISION	
DATE		SHEET 6		OF 17	



LA JOLLA FACILITY



TITLE		ACP		DRG. NO.	
DRAWN BY		PROJ.		REVISION	
DATE		SHEET 6A		OF 17	



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

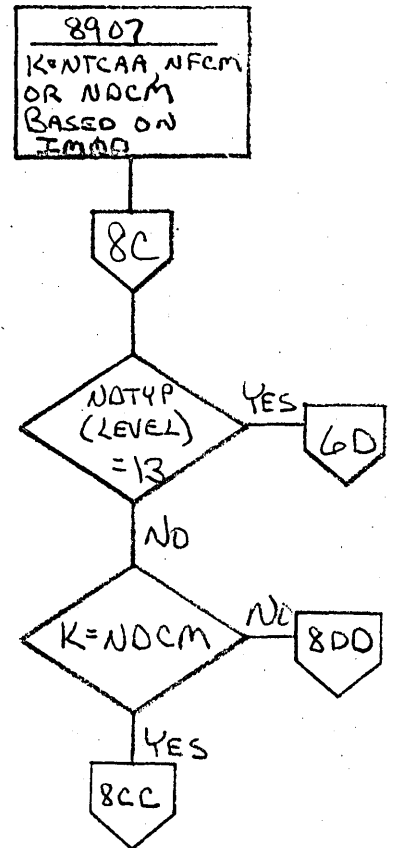
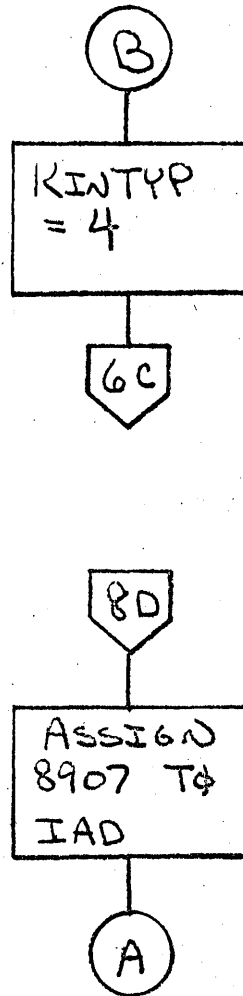
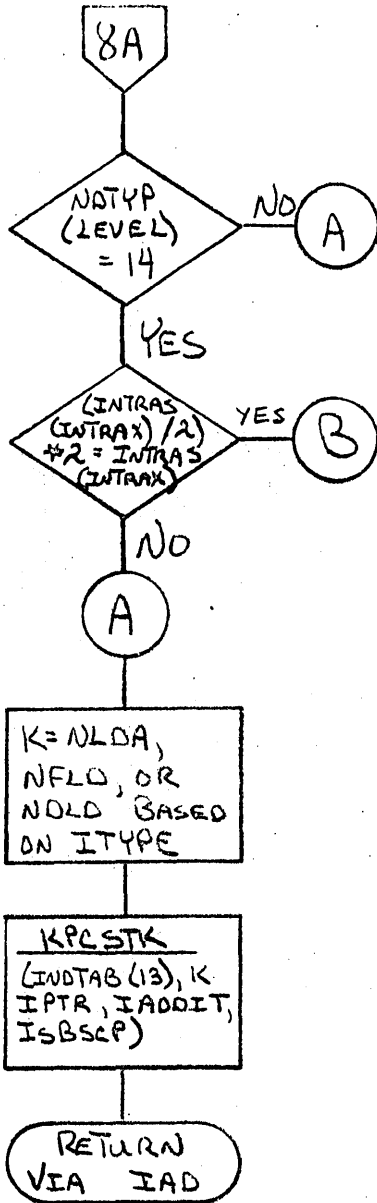
SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	ACP	PAGE 7 OF 17		PROJECT MGR.			
NUMBER	ISSUE DATE			PROJECT NAME			
DRAWN BY	DATE			TASK NO.			
				TASK NAME			





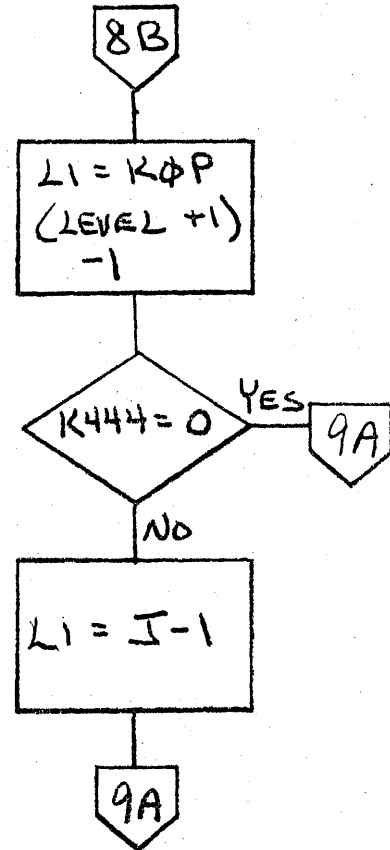
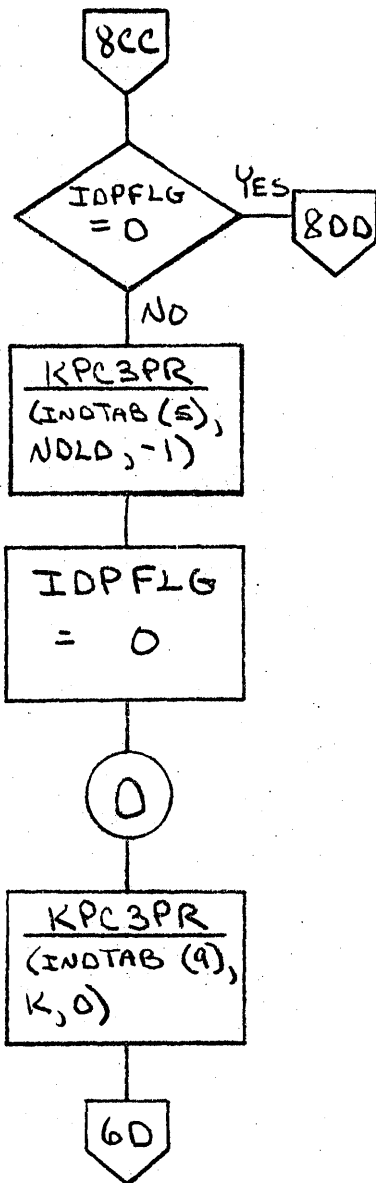
LA JOLLA FACILITY



TITLE		ACP		DRG. NO.	
				REVISION	
DRAWN BY	PROJ.	DATE	SHEET 8	OF	17
CSD 203					



LA JOLLA FACILITY

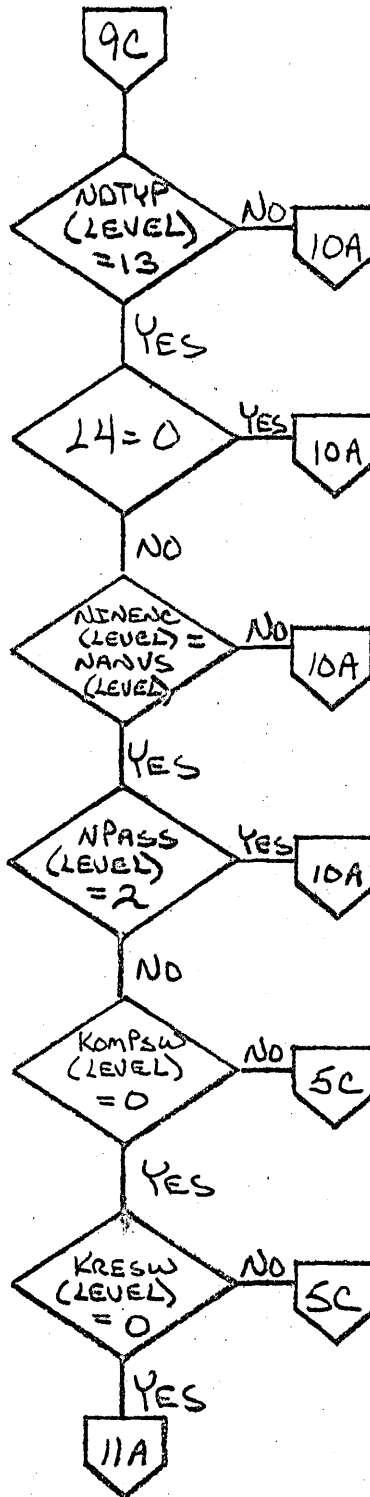
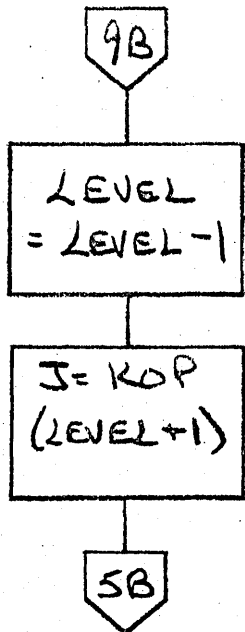
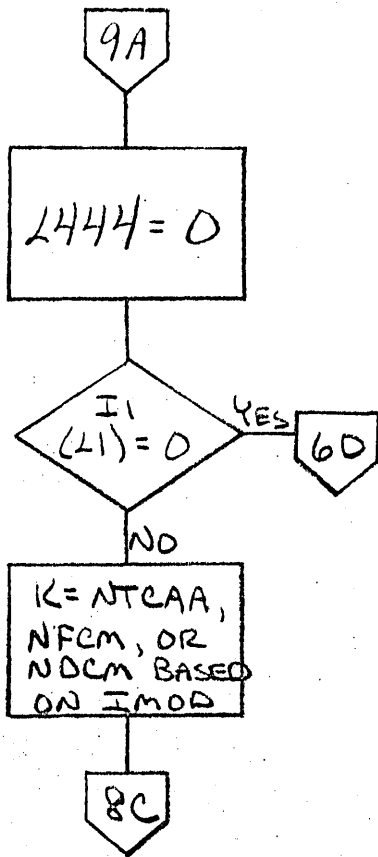


TITLE		ACP		DRG. NO.	
DRAWN BY		PROJ.		REVISION	
CSD 203		DATE		SHEET 8A OF 17	





LA JOLLA FACILITY

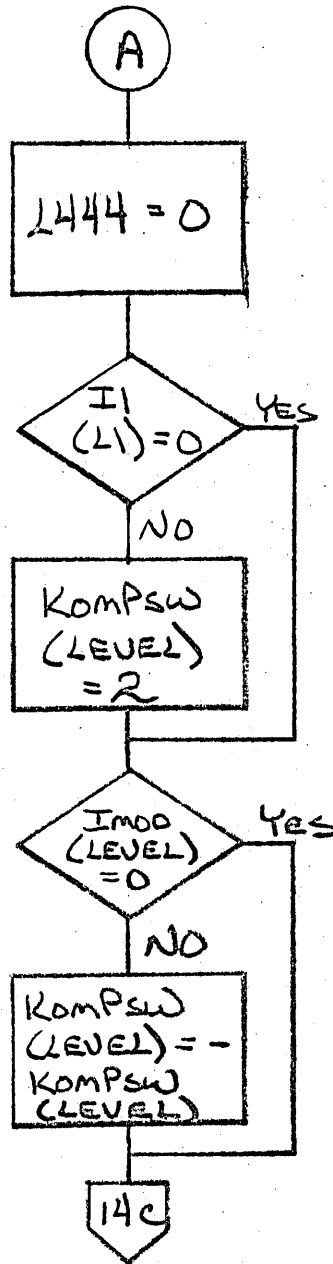
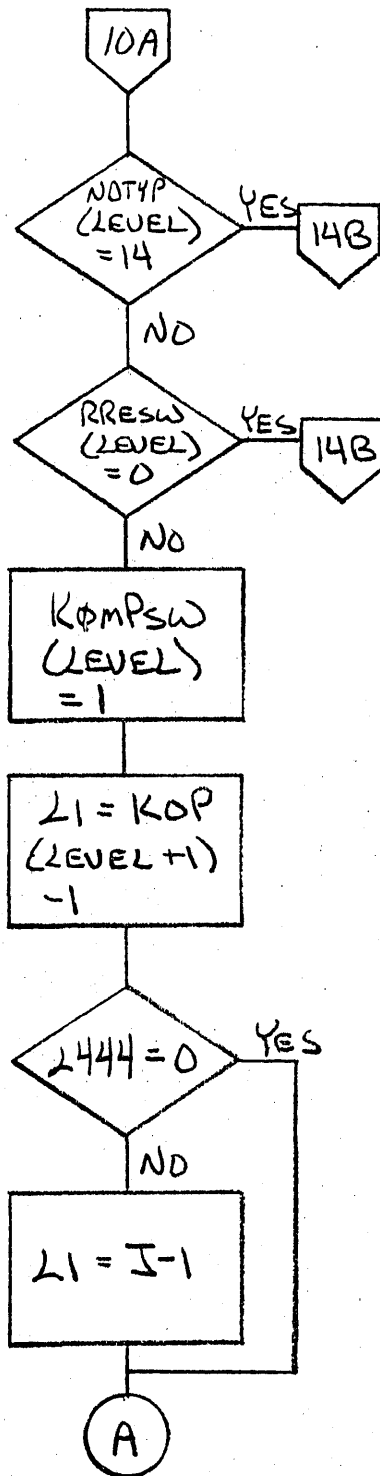


TITLE ACP			DRG. NO.	
			REVISION	
DRAWN BY	PROJ.	DATE	SHEET 9	OF 17
CSD 203				





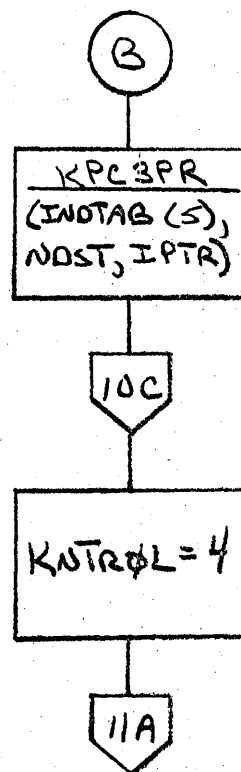
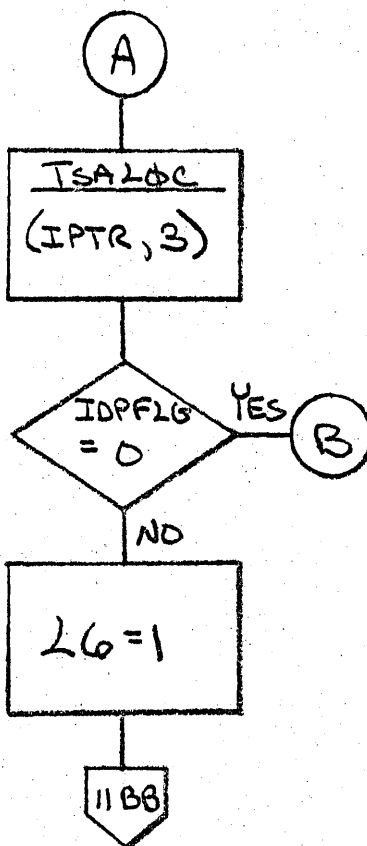
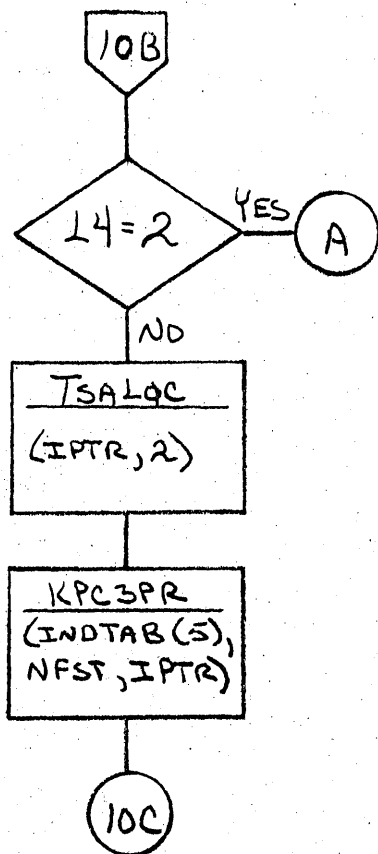
LA JOLLA FACILITY



TITLE		ACP		DRG. NO.	
				REVISION	
DRAWN BY	PROJ.	DATE	SHEET 10	OF	17
CSD 203					



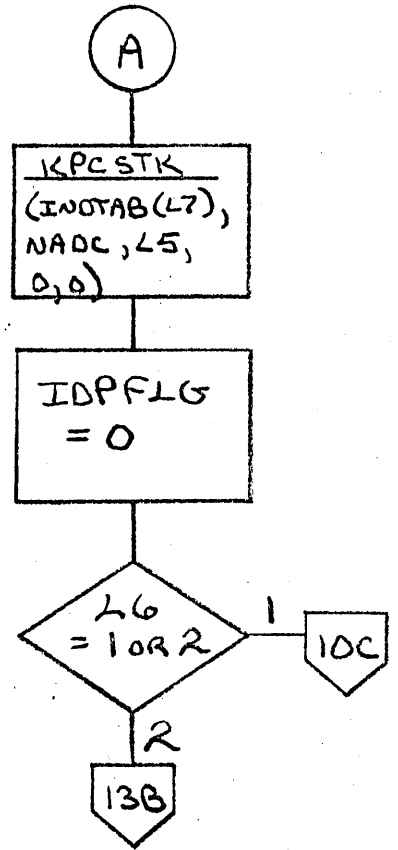
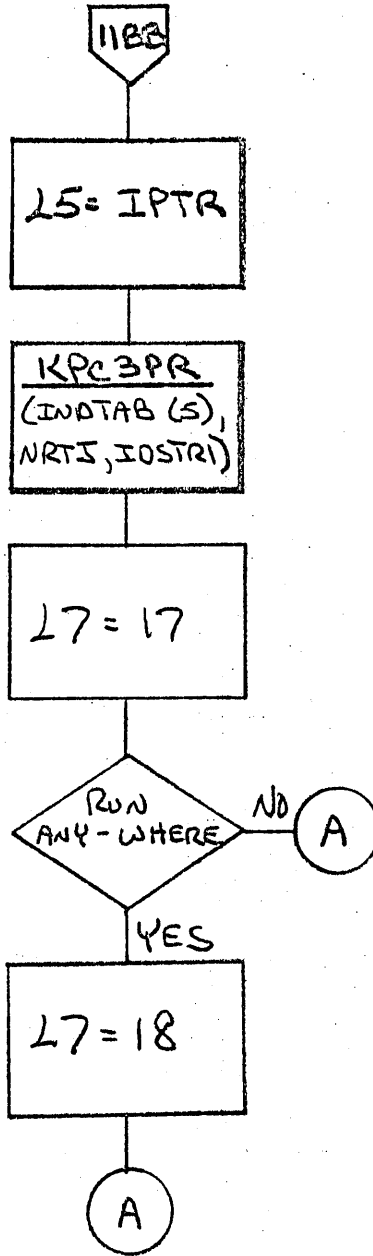
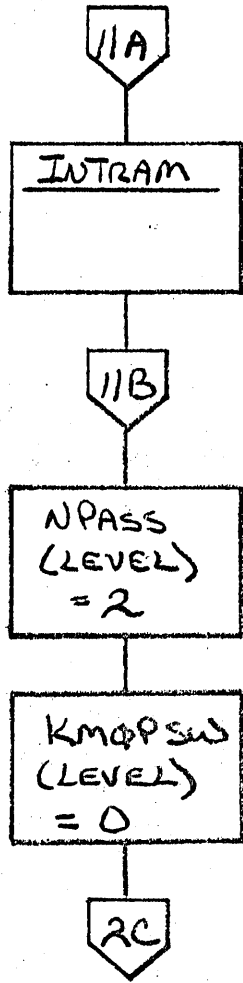
LA JOLLA FACILITY



TITLE		ACP		DRG. NO.	
DRAWN BY		PROJ.		REVISION	
DATE		SHEET 10A		OF 17	



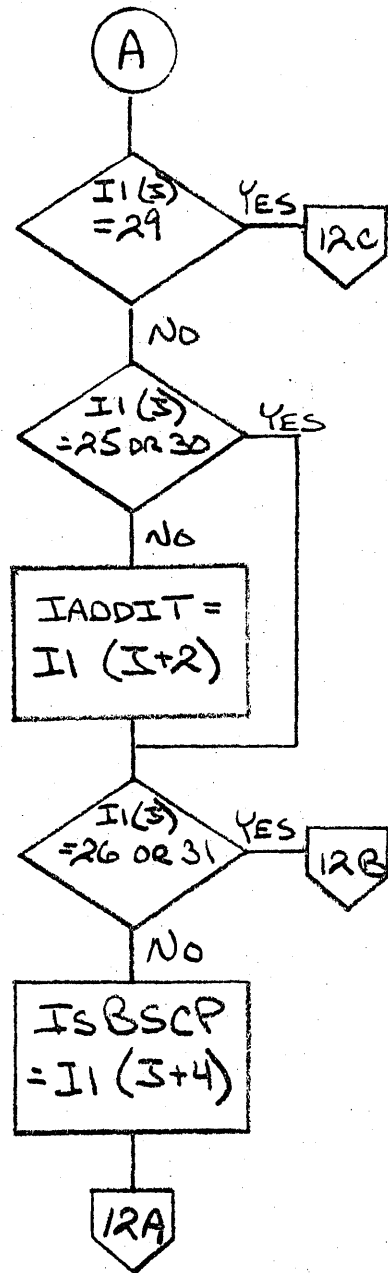
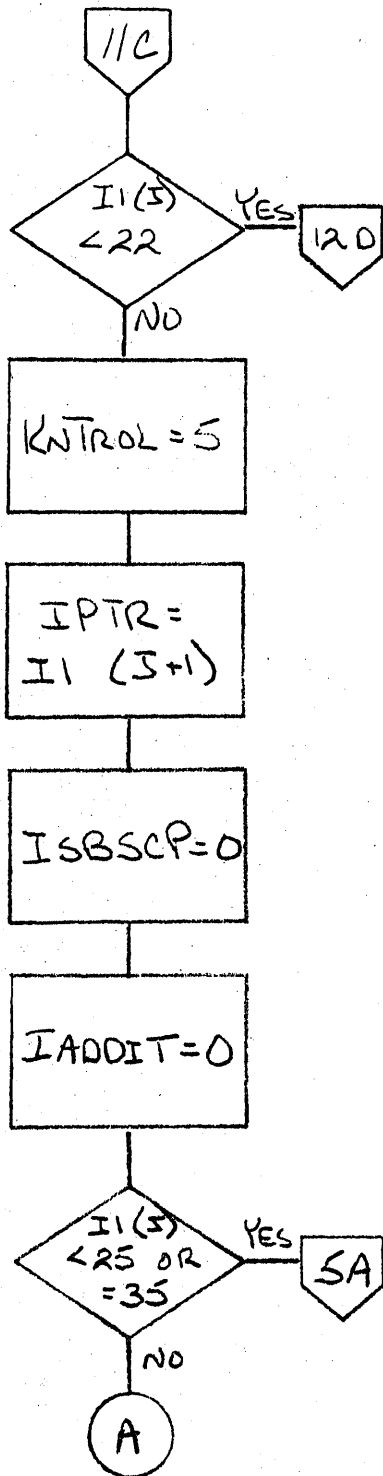
LA JOLLA FACILITY



TITLE			ACF		DRG. NO.	
DRAWN BY			PROJ.		REVISION	
DATE			SHEET		OF	
CSD 203			11		17	



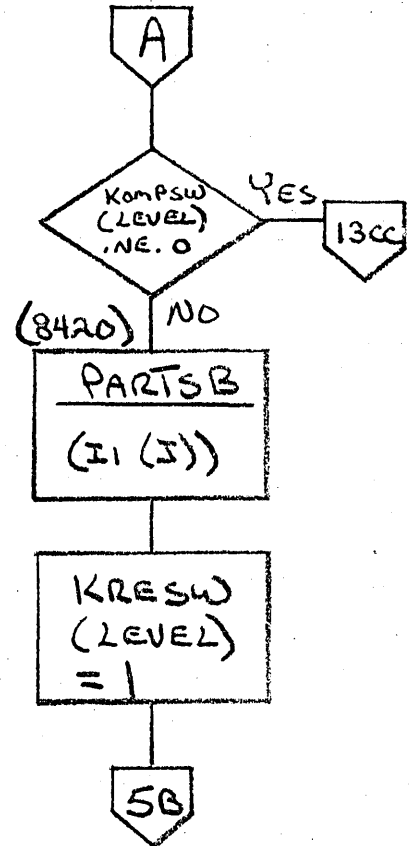
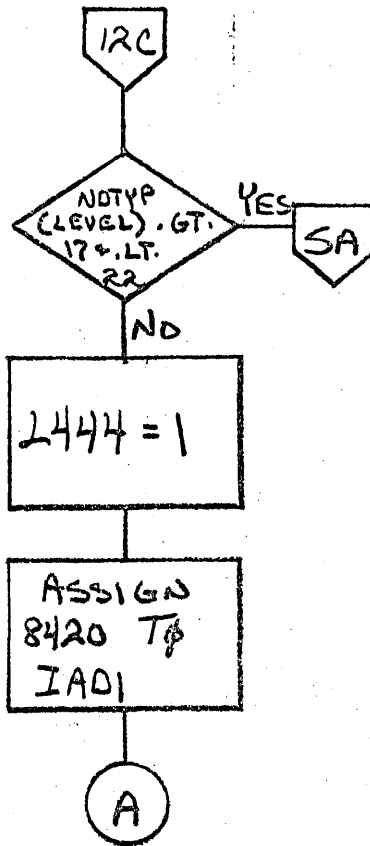
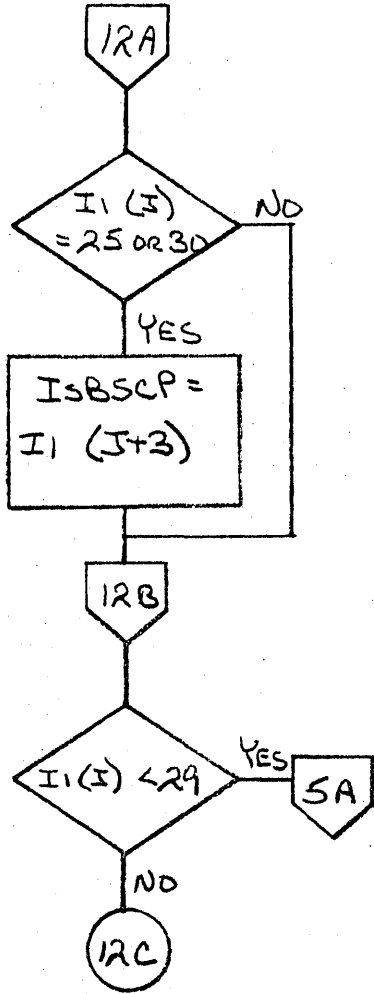
LA JOLLA FACILITY



TITLE		ACF		DRG. NO.	
DRAWN BY		PROJ.		REVISION	
DATE		SHEET 1/A		OF 17	



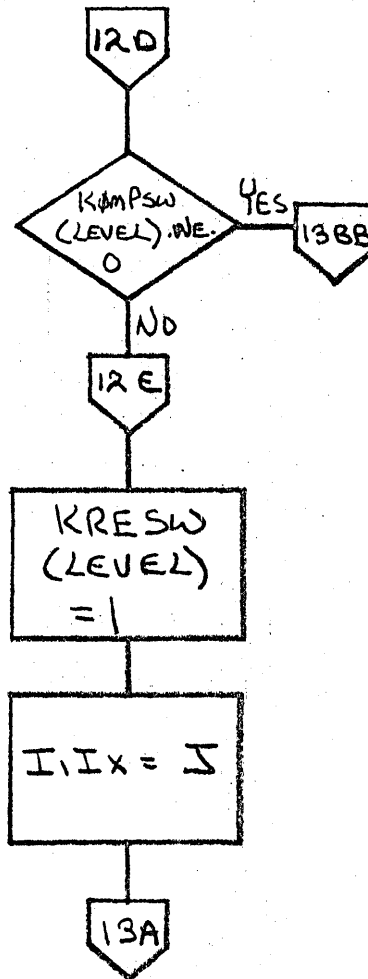
LA JOLLA FACILITY



TITLE		ACP		DRG. NO.	
DRAWN BY		PROJ.		REVISION	
CSD 203		DATE		SHEET 12 OF 17	



LA JOLLA FACILITY

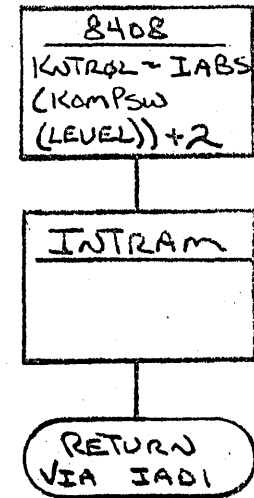
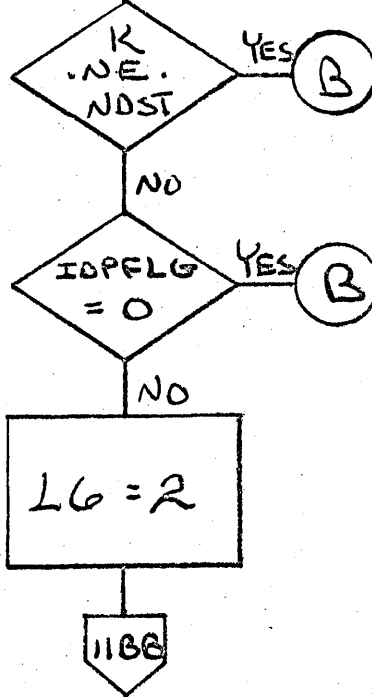
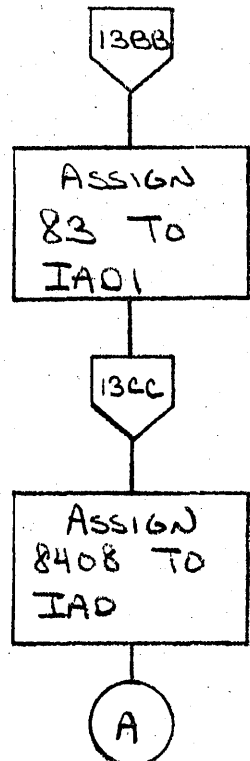
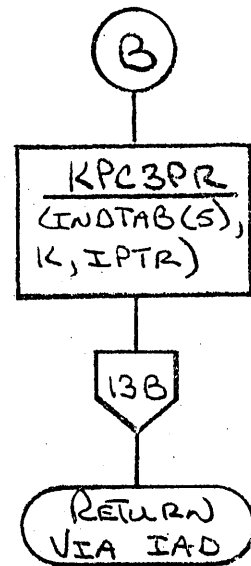
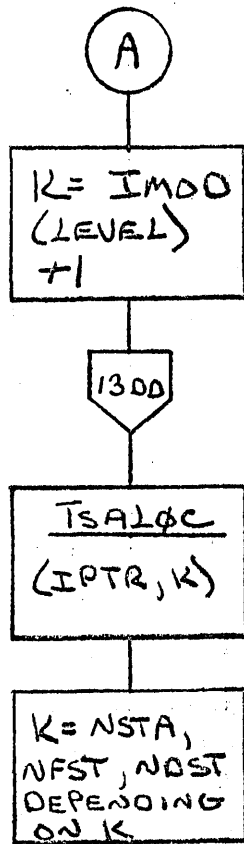
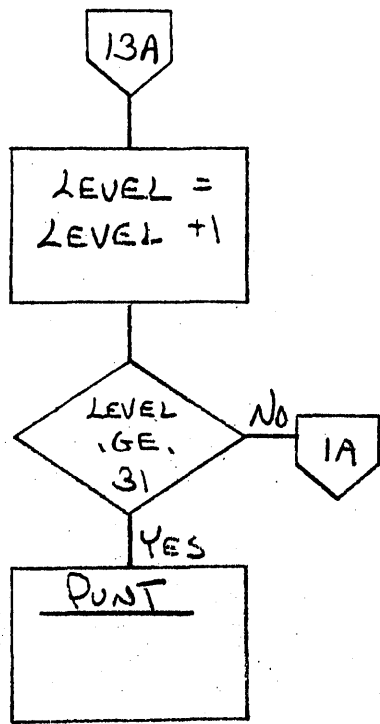


TITLE		ACP		DRG. NO.	
				REVISION	
DRAWN BY	PROJ.	DATE	SHEET 12A OF 17		





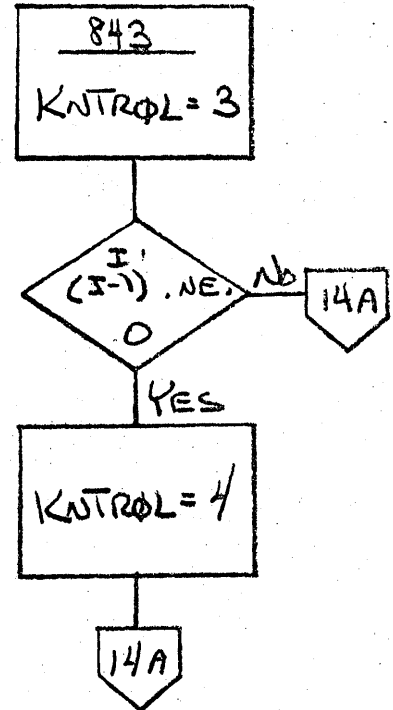
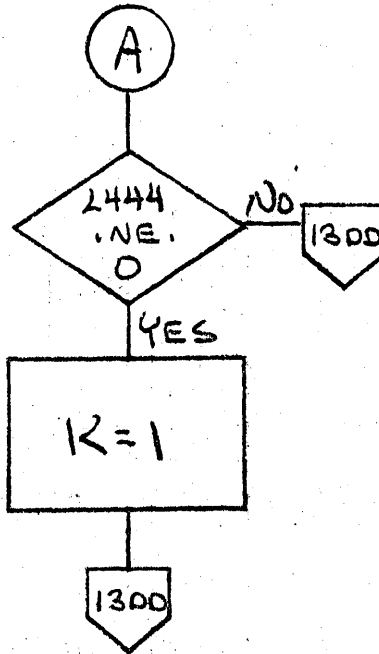
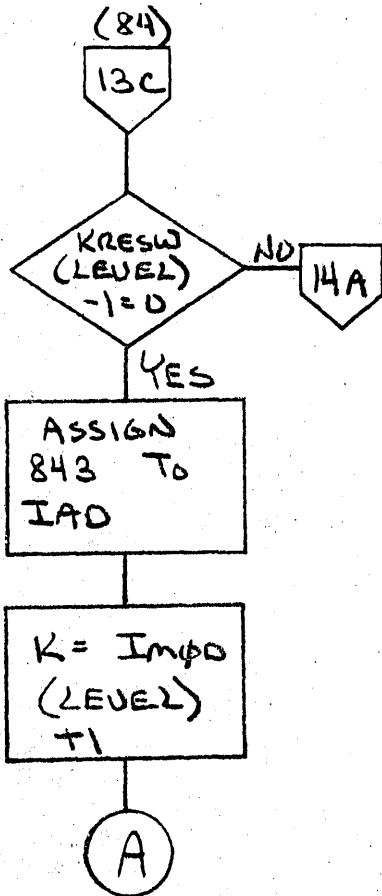
LA JOLLA FACILITY



TITLE		ACF		DRG. NO.	
				REVISION	
DRAWN BY	PROJ.	DATE	SHEET 13	OF	17
CSD 203					



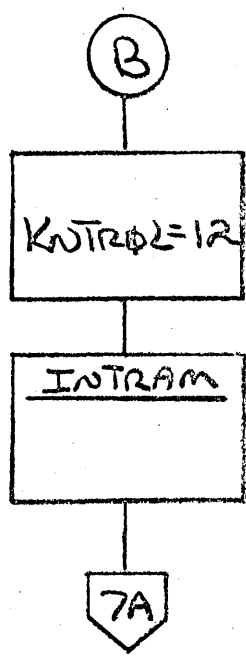
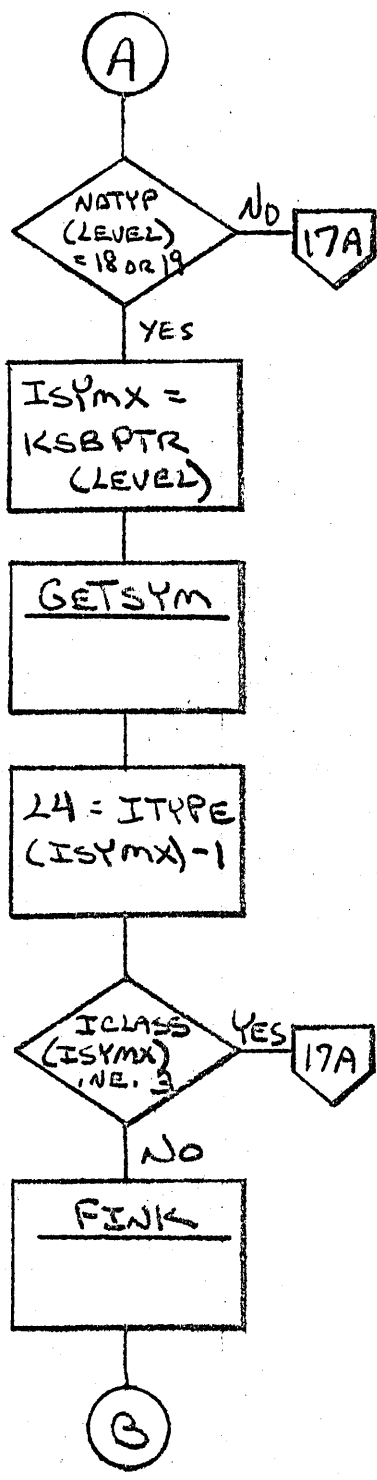
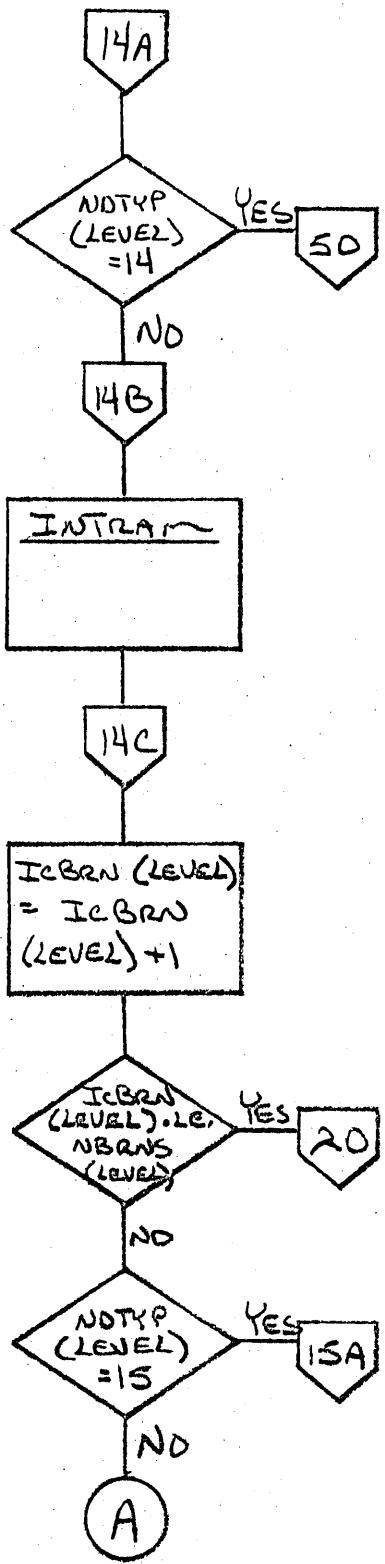
LA JOLLA FACILITY



TITLE	ACP	DRG. NO.	
		REVISION	
DRAWN BY	PROJ.	DATE	SHEET 13A OF 17



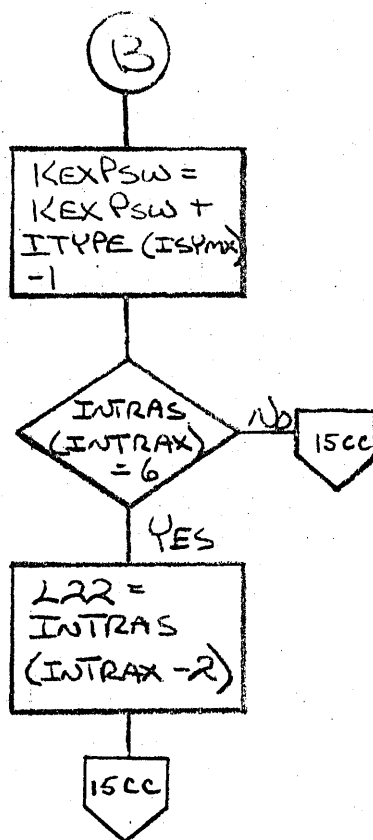
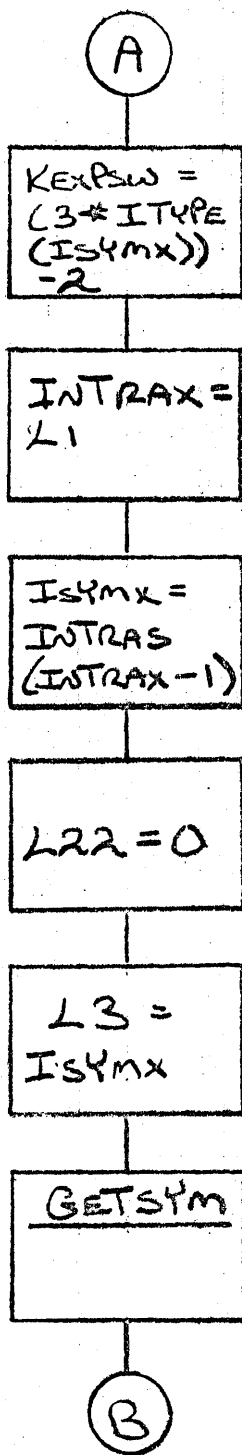
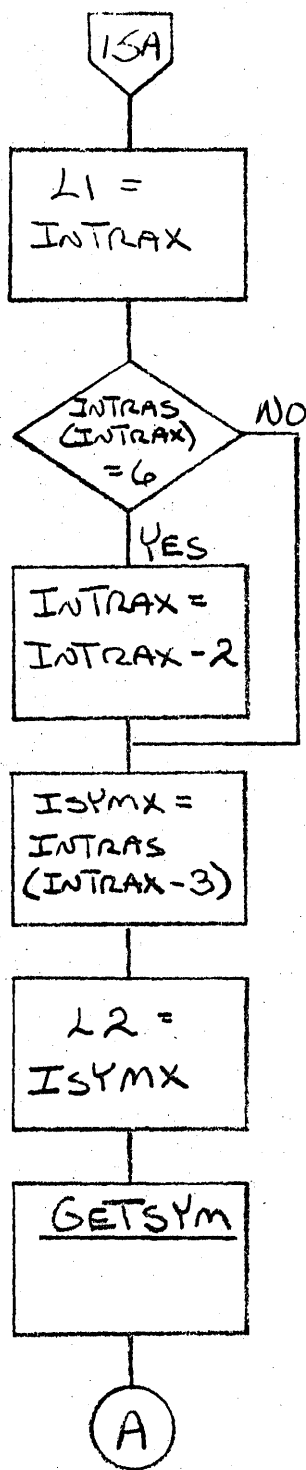
LA JOLLA FACILITY



TITLE ACP			DRG. NO.
			REVISION
DRAWN BY	PROJ.	DATE	SHEET 14 OF 17
CSD 203			



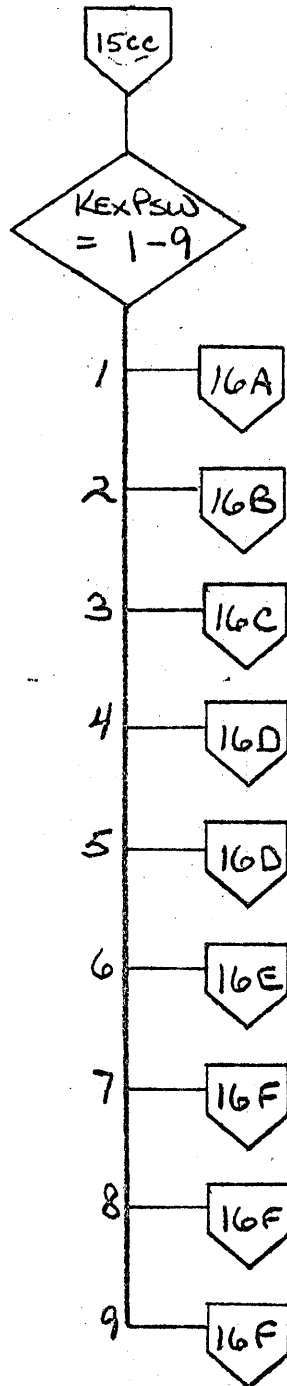
LA JOLLA FACILITY



TITLE		ACP		DRG. NO.	
				REVISION	
DRAWN BY	PROJ.	DATE	SHEET 15 OF 17		



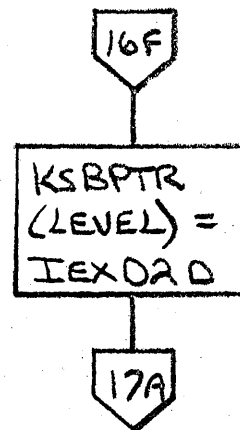
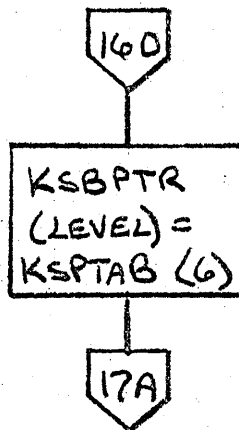
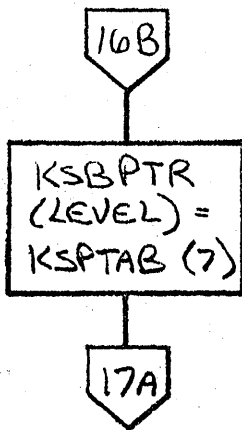
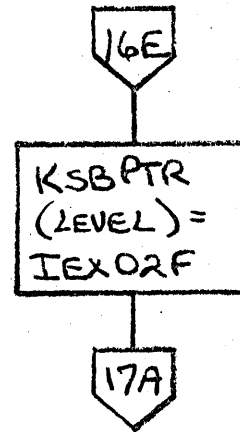
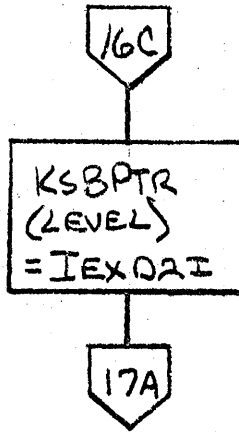
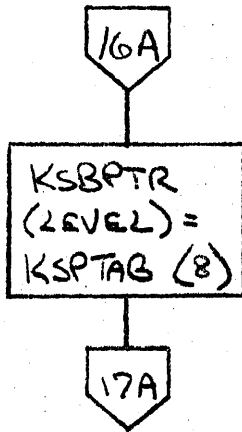
LA JOLLA FACILITY



TITLE		ACF		DRG. NO.	
				REVISION	
DRAWN BY	PROJ.	DATE	SHEET 15A OF 17		
CSD 203					



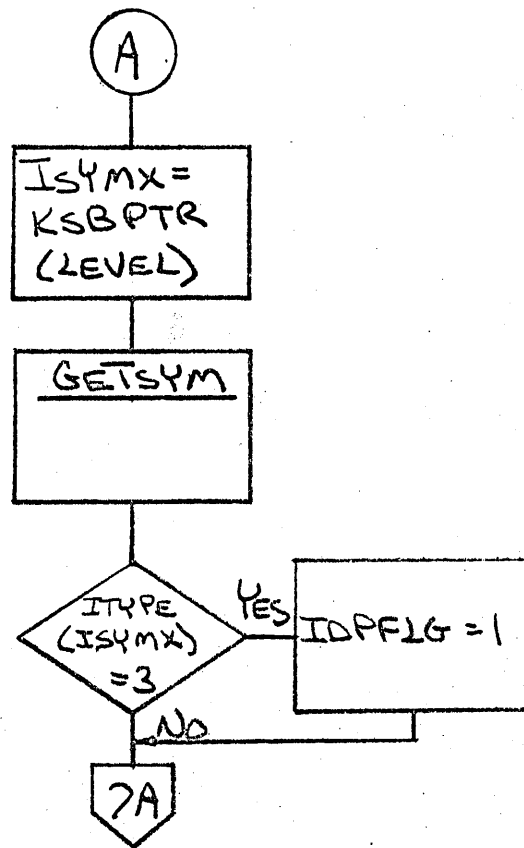
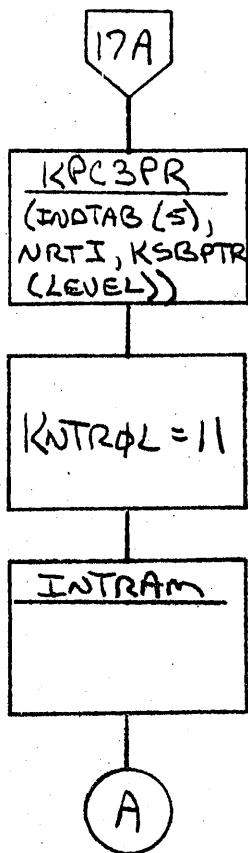
LA JOLLA FACILITY



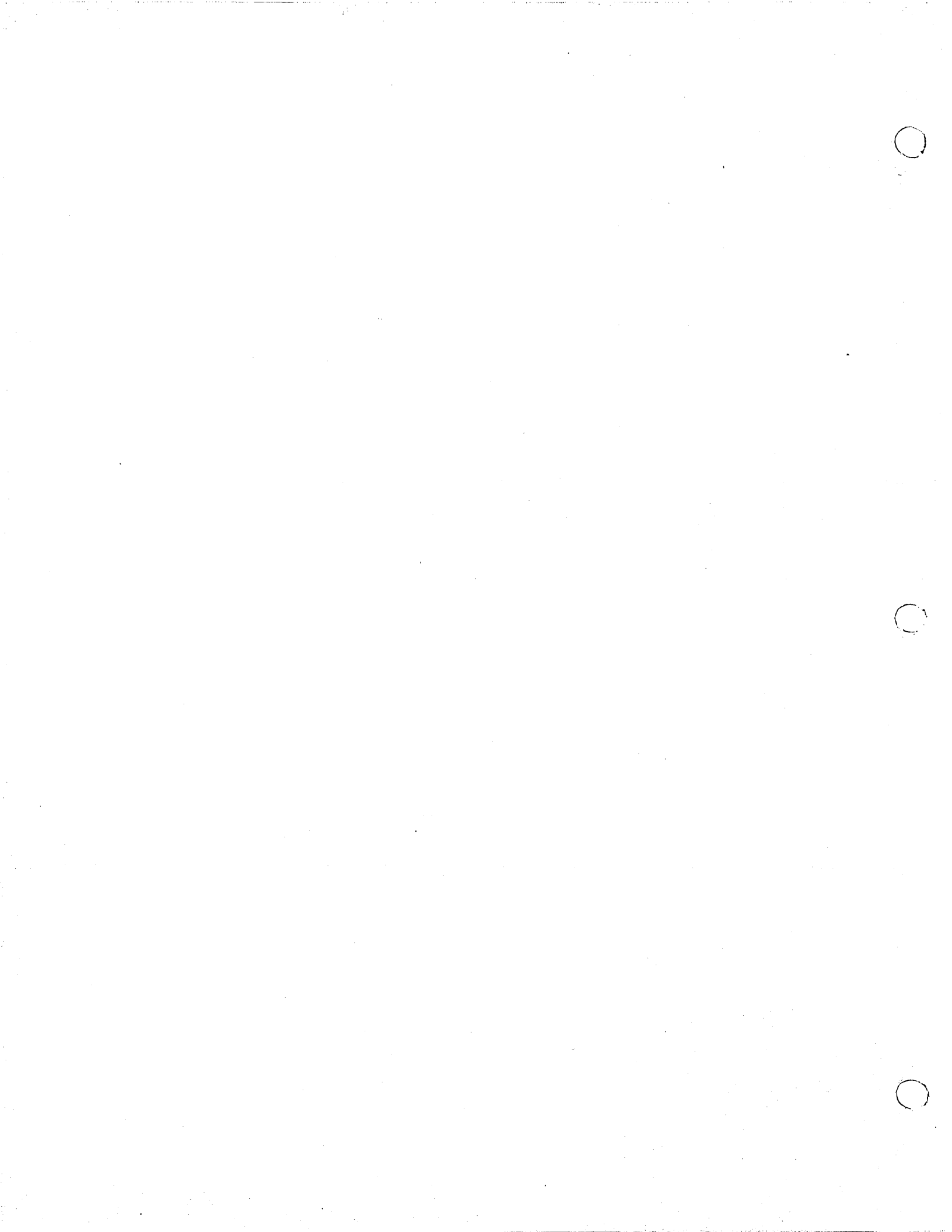
TITLE		ACF		DRG. NO.	
				REVISION	
DRAWN BY	PROJ.	DATE	SHEET	16	OF 17



LA JOLLA FACILITY



TITLE			ACF		DRG. NO.	
					REVISION	
DRAWN BY		PROJ.	DATE		SHEET	7 OF 17
CSD 203						

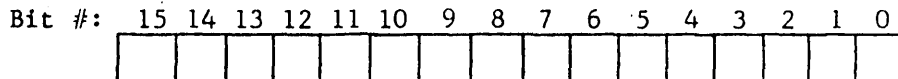




DOCUMENT CLASS TMS PAGE NO 4-147  
 PRODUCT NAME 1700 Mass Storage FORTRAN  
 PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

4.4.4 PARTSB

PARTSB is called upon to generate instructions which load a byte or signed byte into the accumulator. Recalling the bit designations of the 1700 16-bit word:



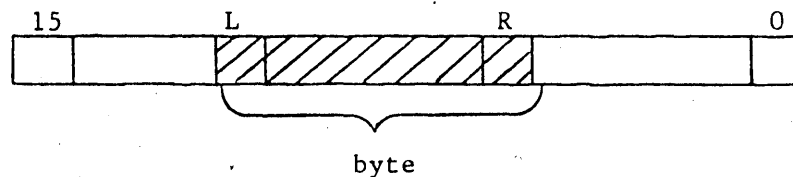
Bytes are delineated in the symbol table by

IPART(ISYMX) = 1, byte  
 = 2, signed byte

IPARTR(ISYMX) = the bit number of the right-most (least significant bit of the byte)

IPARTL(ISYMX) = the bit number of the leftmost (most significant bit of the byte)

In a given byte:



two basic instruction sequences are used:

- Byte (unsigned - except for the nominal case, L = 15, R = 0)
 

LDA	X	where X is the word containing the byte
ARS	m	where m = IPARTR(ISYMX) (omitted when m = 0)
AND	MASK	where the mask is a generated constant such that bits 15 through bit (L-R+1) are 0 and bits (L-R) through 0 are 1. The generated constants for the masks are entered in the symbol as they are required.
- Signed Byte
 

LDA	X	where X is the word containing the byte
ALS	m	where m = 15-IPARTL(ISYMX) (omitted when m = 0)
ARS	n	where n = 15-IPARTL(ISYMX)+IPARTR(ISYMX)

Flowcharts:

PARTSB  
{N}

ISYMX=N{2}

KSYMX=N{2}

KPAD=0

A

A

KPSB=0

N{1}=31  
?

KPAD=N{3}

N{1}=30  
?

KPSB=N{4}

B

C

N{1}=32  
?

KPAD=N{3}

KPSB=N{5}

D

D

KPCSTK  
{INDTAB{13},  
NLDA,N{2},  
KPAD,KPSB}

ISYMX=  
KSYMX

GETSYM

2A

**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	PARTSB			PROJECT MGR.			
	PAGE 1 OF 3			PROJECT NAME			
NUMBER		ISSUE DATE		TASK NO.			
DRAWN BY		DATE		TASK NAME			

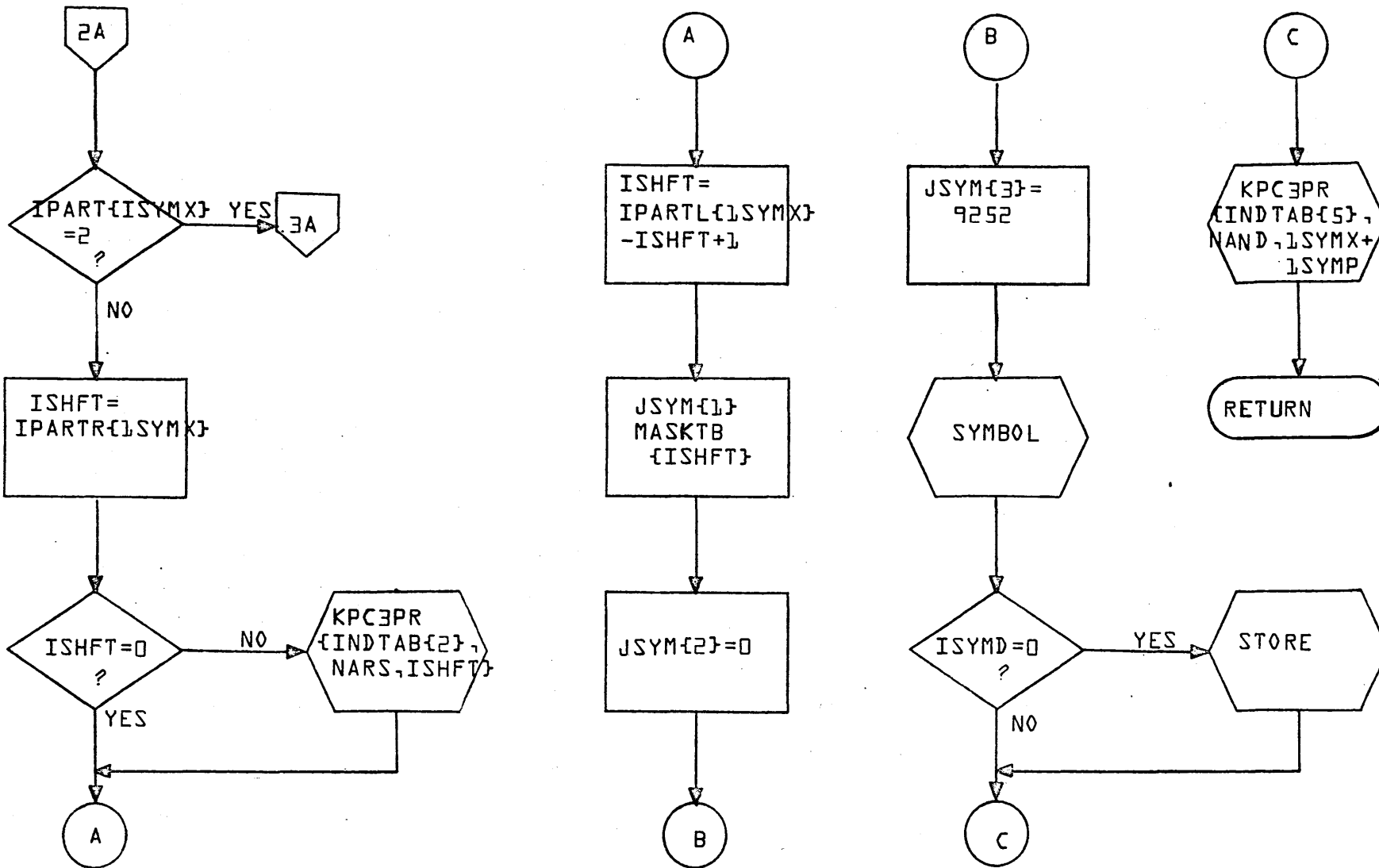
4-118

A

B

C

D



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700
DOCUMENT TITLE	PARTSB		
PAGE 2 OF 3			
NUMBER	ISSUE DATE		
DRAWN BY	DATE		

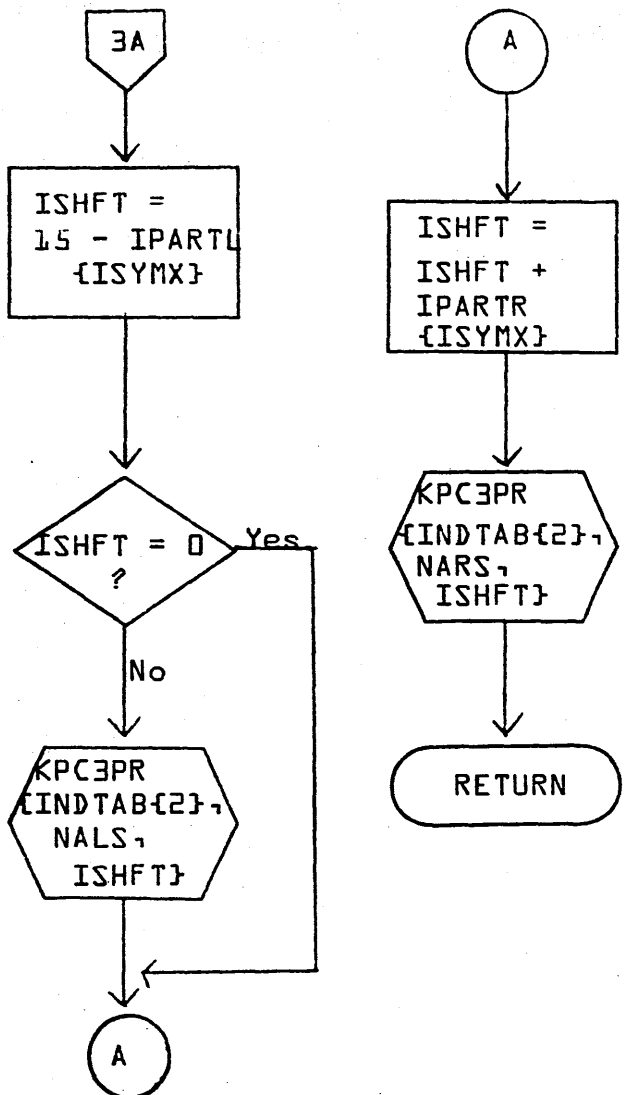
PROJECT NO.	REV	APPROVED	DATE
PROJECT MGR.			
PROJECT NAME			
TASK NO.			
TASK NAME			

A

B

C

D



**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	PARTSB			PROJECT MGR.			
PAGE 3 OF 3				PROJECT NAME			
NUMBER	ISSUE DATE			TASK NO.			
DRAWN BY	DATE			TASK NAME			

4-150

DOCUMENT CLASS \_\_\_\_\_ IMS \_\_\_\_\_ PAGE NO. 4-151  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT NO. C005 VERSION 1.0 MACHINE SERIES 1700

## 4.4.15 READIR

READIR is the PHASE B subroutine for reading an input record. It is called by PHASE B.

## Parameters:

I = the index to the input buffer (INBUFF(I)) designating the starting location to begin input.

J = output parameter contains index to the input buffer (INBUFF(J)) designating the beginning of non-header information of the record just read in.

## Operation:

1. Call READ and input a record beginning at INBUFF(I) specifying 203-I words. Since the buffer is 202 words long all inputs are through the last word of the input buffer even though most records do not require that much space.
2. Compute J
  1. If there is no statement label,  $J = I + 5$ .
  2. If there is a statement label
    - a.  $J = I + 6$
    - b. output the label.
  3. Update ISTNO to new statement number.
  4. Output STN pseudo instruction (gives PHASE C indication that a new statement is beginning and what the statement number is.
  5. Check to insure that the statement just read was not too large for the available input buffer. If so call PUNT.

## Flowcharts:

READIR  
(I,J)

I > 202 ?

YES

PUNT

No

READ  
(ISCR1, 0,  
203-I,  
INBUFF(I))

J = J + 4

A

A

INBUFF(I+1) < 0 ?

No

J = J + 1

No

ISTNO =  
IABS (INBUFF  
(I+1))

YES

INBUFF(I+2) = 17

No

KPC3PR  
(INDTAB(2),  
NSTN,  
ISTNO)

YES

INBUFF(I+2) = 17

No

LABKPC  
(INBUFF(I+1))

I + INBUFF(I) - 1 < 202 ?

No

PUNT

YES

RETURN

CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE   
FLOWCHART   
DECISION TABLE   
OTHER

DOCUMENT CLASS *IMS* MACH. TYPE *1700*

DOCUMENT TITLE *READIR*

PAGE 1 OF 1

NUMBER ISSUE DATE

DRAWN BY DATE

PROJECT NO.

PROJECT MGR.

PROJECT NAME

TASK NO.

TASK NAME

REV

APPROVED

DATE

DOCUMENT CLASS \_\_\_\_\_ IMS \_\_\_\_\_ PAGE NO. 4-153  
PROJECT NAME 1700 Mass Storage FORTRAN  
PROJECT NO. C005 VERSION 3.0 MACHINE SERIES 1700

#### 4.4.16 SUBPR1 and SUBPR2 subscript processors.

SUBPR1 and SUBPR2 are called upon by ASUPER to process subscripts. (See discussion of subscript processing in 4.4.2, ASUPER).

Both programs check for operand types 26 (increment only subscripted variable) and 31 (increment only subscripted partial variable). No processing is performed on these two types. What then remains are types 25, 27, 28, 30, 32, and 33, variable only, variable and increment, and complex subscripted partial/non partial variables.

##### Parameters:

I - input parameter to both SUBPR1 and SUBPR2 represents address of array which begins at the operand "type" code of a "tree" operand. I(1) = type code.

#### 4.4.16.1 SUBPR1

SUBPR1 processes subscripts which subscript variables that are not part of calling sequences.

The subscript expression is then passed on to SUBPR3 which generates the instruction, which evaluate the subscript and leave the result in the accumulator.

SUBPR3 sets its parameter

I1 = 0 if no actual evaluation was required. (i.e. operand was type 25, 27, 30, or 32 and the variable in the subscript expression was not a partial.

I1 = 1, or 2 if instructions for evaluation were actually required. (i.e., operand type was 28 or 33 (complex) or the variable in the subscript expression was a partial).

SUBPR1 then references the returned parameter I1. If I1 = 0, processing is complete. If I1 ≠ 0, then:

1. A temporary storage location is allocated.
2. A store instruction is generated to store the result of the subscript evaluation.
3. The variable type in the subscript expression is set to 24 (non-subscripted, non-partial variable).
4. The temporary storage cell is added to INTRAS through a call to INTRAM (at this point INTRAS is at level 0, the level which contains subscript assignments. See discussion in 4.4.7, INTRAM).

DOCUMENT CLASS \_\_\_\_\_ 'MG' \_\_\_\_\_ PAGE NO. 4-154  
PRODUCT NAME Mass Storage FORTRAN  
PRODUCT NO. CP05 VERSION 2.0 MACHINE SERIES 1700

## 4.4.16.2 SUBPR2

SUBPR2 is the analogous processor to SUBPR1 but processes subscripts which subscript variables that are part of calling sequences.

As in SUBPR1, the subscript expression is passed on to SUBPR3 which generates the instruction which evaluates the subscript and leaves the result in the accumulator.

Then I1 is tested as in SUBPR2.

If I1 = 0 no actual evaluation was required but the value of the subscript must still be in the A register so an LDA command is generated to load the subscript variable.

Then an ADD instruction is generated to add the address of the subscripted variable to the value of the subscript. (In the run-anywhere case the relocation constant is also added).

Finally, a label is created and a store instruction referencing that label is generated. The subscripted variable operand is then modified to become a "calling sequence label element" (type 36).

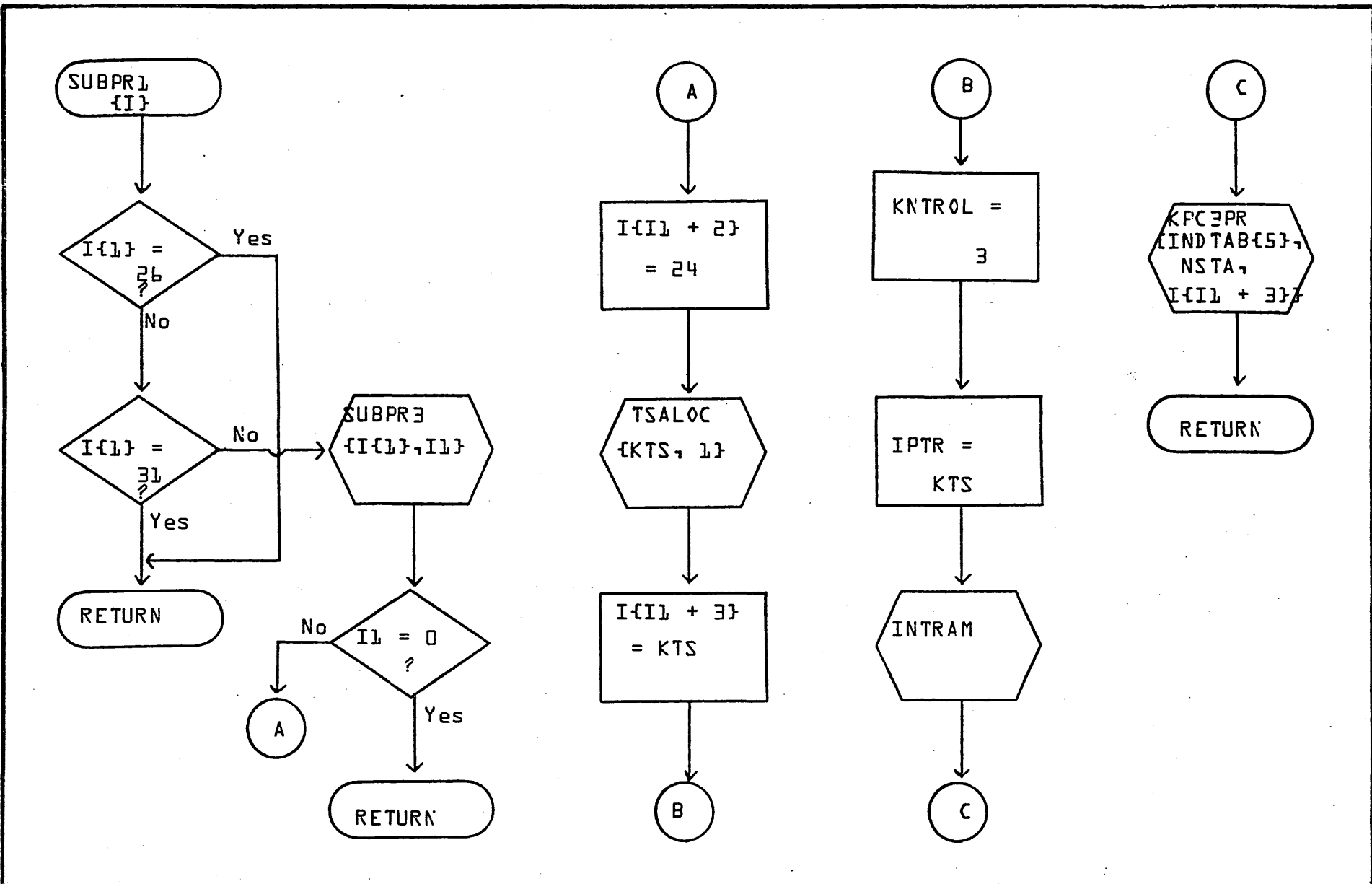


A

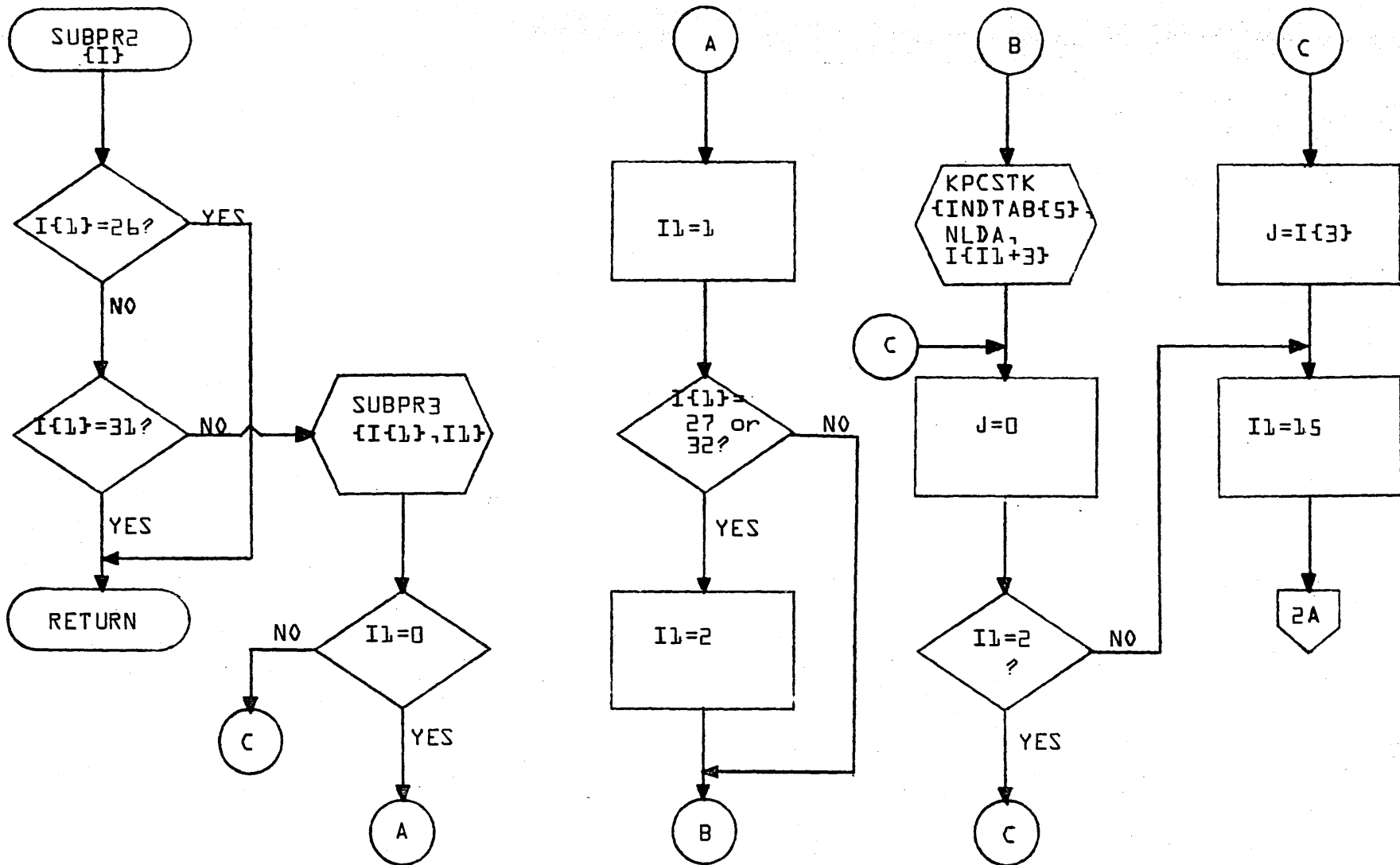
B

C

D



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	SUBPR1			PROJECT MGR.			
		PAGE 1 OF 1			PROJECT NAME			
	NUMBER	ISSUE DATE			TASK NO.			
	DRAWN BY	DATE			TASK NAME			



**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	SUBPR2			PROJECT MGR.			
NUMBER		ISSUE DATE	PAGE 1 OF 2	PROJECT NAME			
DRAWN BY		DATE		TASK NO.			
				TASK NAME			

1

2

3

4

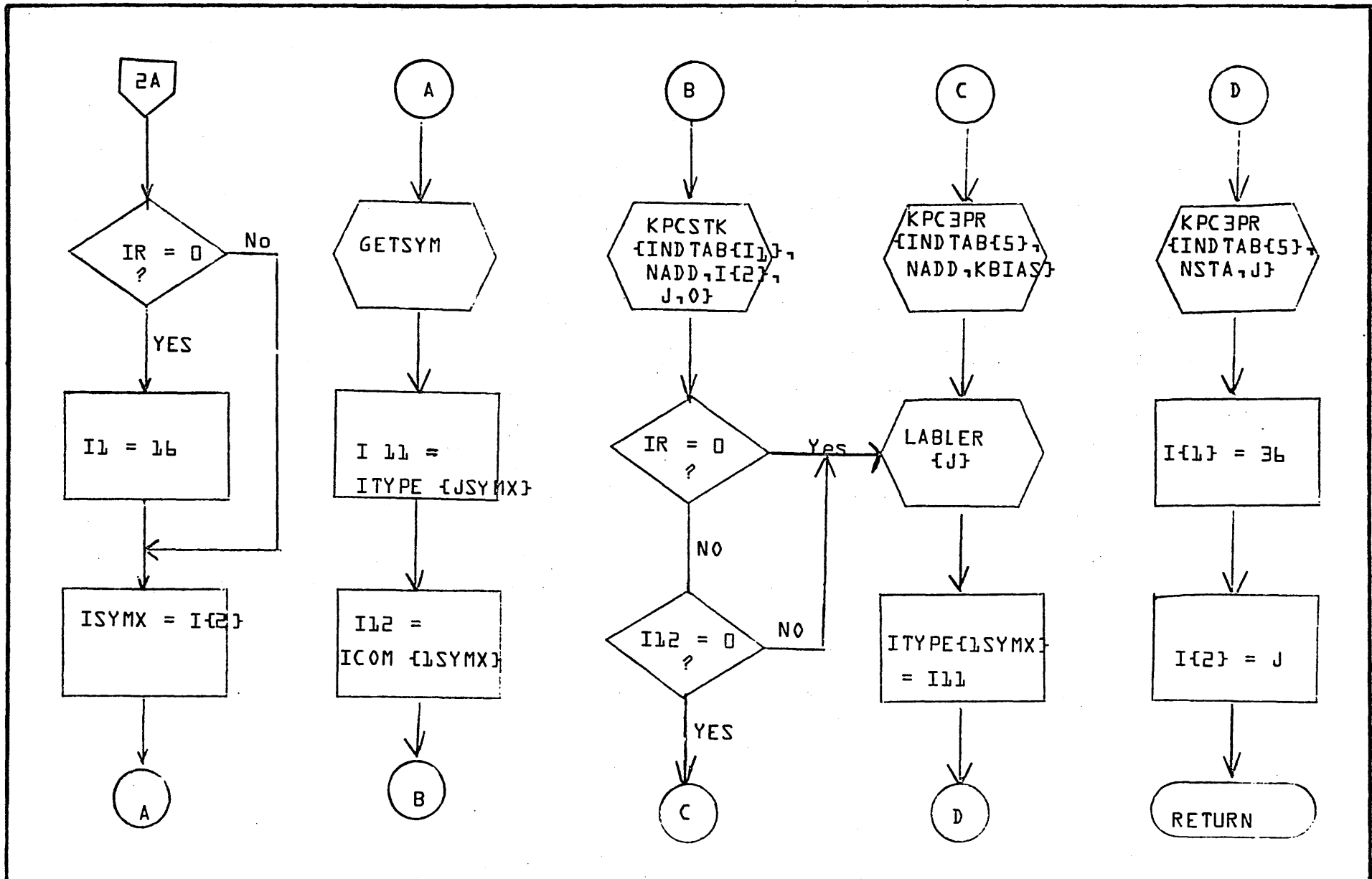
5

A

B

C

D



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS <b>TMS</b>	MACH. TYPE <b>1200</b>	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE <b>SUBPR2</b>		PROJECT MGR.			
		PAGE <b>2</b> OF <b>2</b>	PROJECT NAME			
	NUMBER	ISSUE DATE	TASK NO.			
	DRAWN BY	DATE <b>3 - 67</b>	TASK NAME			

4-157

DOCUMENT CLASS \_\_\_\_\_ IMS \_\_\_\_\_ PAGE NO. 4-158  
PRODUCT NAME 1700 Mass Storage FORTRAN  
PRODUCT NO. C005 \_\_\_\_\_ VERSIO 2.0 \_\_\_\_\_ MACHINE SERIES 1700

4.4.17 SUBPR3

SUBPR3 is called by SUBPR1 and SUBPR2 to generate the instructions for evaluating a subscript. Code generation is performed in three cases and the output parameter I1 is set accordingly:

1. Partial variable only subscript, I1 = 1.
2. Partial variable with increment subscript, I1 = 2.
3. Complex subscript I1 = 2.

When no instructions are generated, I1 = 0.

SUBPR3  
{I,IL}

IL = 0

{I} = 25 or 30?

{I} = 29 ?

B

{I} = 27 or 32 ?

{I} = 29

A

ACP  
{I}

{I} = 28 or 33?

RETURN

{I} =  
{I} - 1

IL = 2

RETURN

A

PARTSB  
{I}

B

PARTSB  
{I}

IL = 1

RETURN

**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	SUBPR3	PAGE 1 OF 1		PROJECT MGR.			
NUMBER	ISSUE DATE			PROJECT NAME			
DRAWN BY	DATE			TASK NO.			
				TASK NAME			



CONTROL DATA CORPORATION  
LA JOLLA RESOURCE CENTER DIVISION

DOCUMENT CLASS IMS PAGE NO. 4-161  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

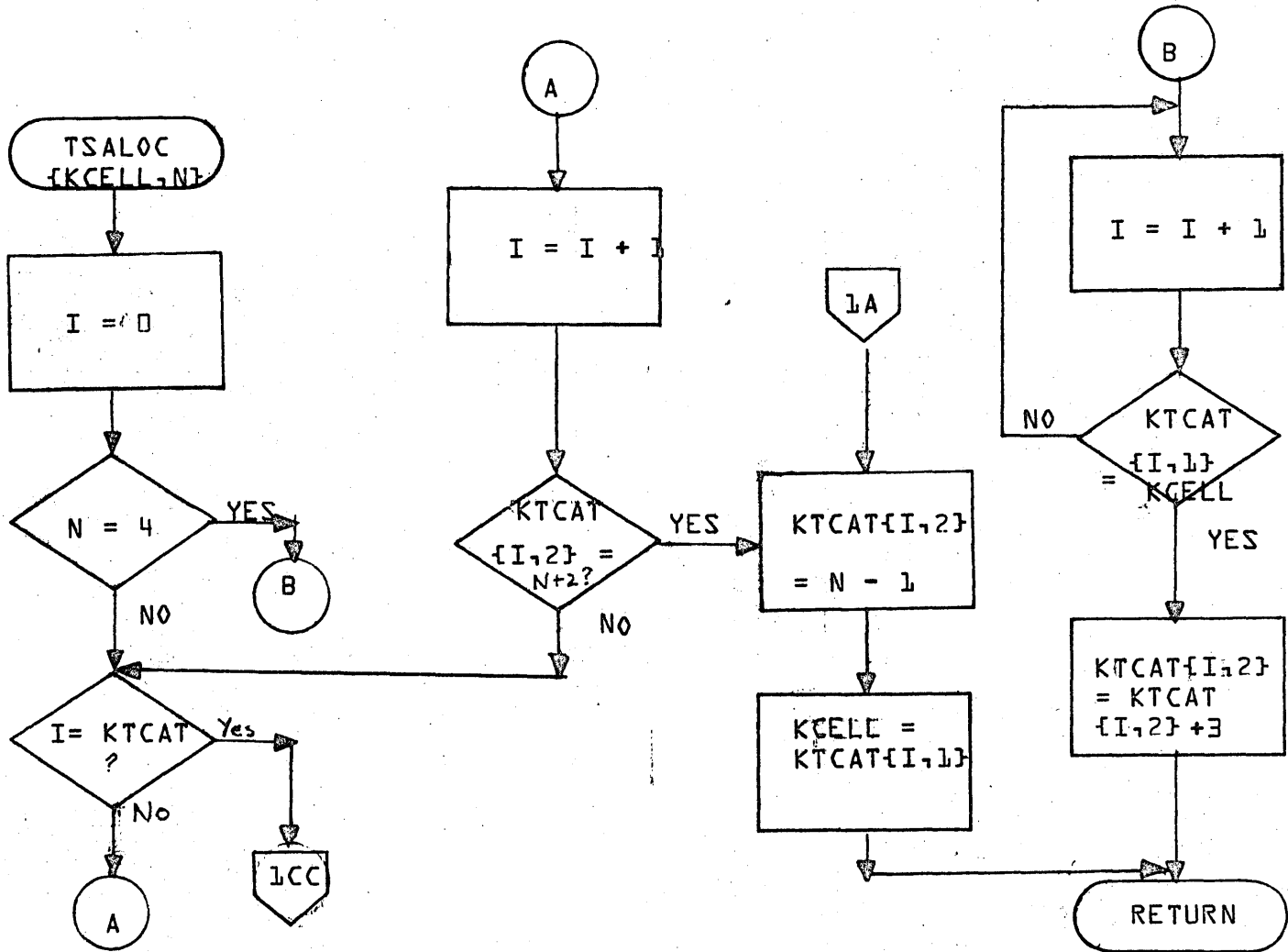
N request parameter =

- 1 integer temporary storage requested. TSALOC searches KTCAT for an available integer temporary storage {A=3}. If one is found, the pointer is passed back in KCELL and availability indicator is set to 0. If none is found a new one is created and KTCAT is updated.
- 2 real temporary storage requested. TSALOC searches KTCAT for an available real temporary storage {A=4}. If one is found, the pointer is passed back in KCELL and availability indicator is set to 1. If none is found, a new one is created and KTCAT is updated.
- 3 double precision temporary storage requested. TSALOC searches KTCAT for an available double precision temporary storage {A=5}. If one is found, the pointer is passed back in KCELL and availability indicator is set to 2. If none is found, a new one is created and KTCAT is updated.
- 4 release request. KCELL contains pointer of temporary storage cell to be released. It is looked up in KTCAT and availability indicator is set to available:

0 → 3  
1 → 4  
2 → 5



LA JOLLA FACILITY

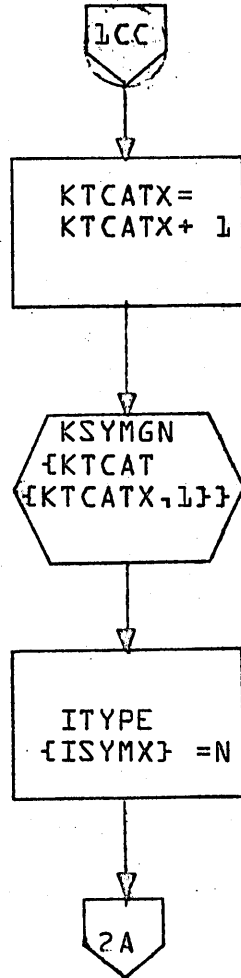


TITLE			DRG. NO.	
TSALOC			REVISION	
DRAWN BY	PROJ.	DATE	SHEET	1 OF 2
CSD 203				





LA JOLLA FACILITY

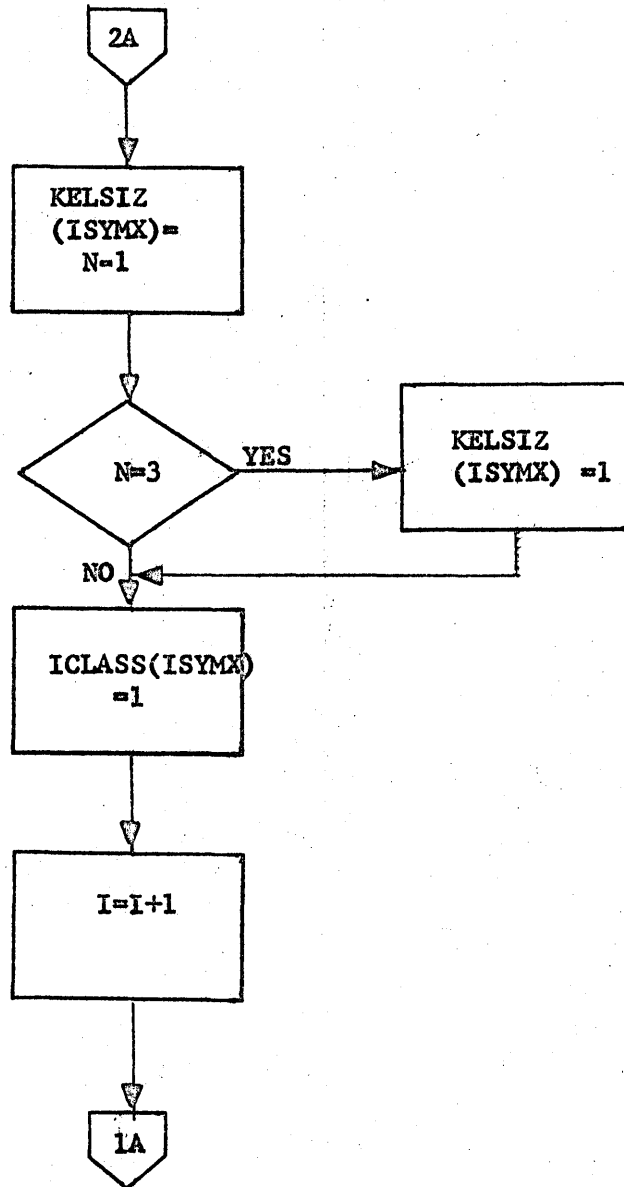


TITLE		TSALOC		DRG. NO.	
				REVISION	
DRAWN BY	PROJ.	DATE	SHEET 1A OF 2		

**CONTROL DATA**

LA JOLLA RESOURCE CENTER  
IMS Page 4-163  
1700 MASS STORAGE FORTRAN  
C005\*3.1 A/B

LA JOLLA RESOURCE CENTER



TITLE		TSALOC		DRG. NO.	
DRAWN BY		PROJ.		REVISION	
CSD 203 ANALYSIS		DATE		SHEET 2 OF 2	

DOCUMENT CLASS IMS PAGE NO. 4-164  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

4.4.19 HELEN

### I. General

Locations of several types of data and storage cells are not established implicitly in the source program. They include locations of:

1. Variable Arrays
2. Variables
3. Constants
4. Format information
5. Temporary cells assigned by Phase B to contain intermediate computations.

The emphasis of 1700 FORTRAN on conserving core space and optimizing relative addressing, necessitates location of this data in relation to the rest of the program such that space and speed are minimized.

### II. Implementation

All programmer defined arrays, variables, constants, etc., except format specifications will be output at the top of each program by Phase B. Under certain conditions Phase C will take them from their place at the top of the program and relocate them within the program body.

#### A. Format Information

See NOPROC

#### B. Arrays

1. Though placement of an array within the program body may shorten references to it, far more references may be destroyed by so doing.
2. Arrays are most often referred to in loops. Though there may be a great deal more references made to array elements during the logical sequence of a program's operation there will be far fewer physical references; that is, less space will be devoted to array references than to single variable references.
3. Considering the non-subscripted references to arrays, it is likely that because of their size at least some of the elements would not be accessible through 1-word commands regardless of the placement of the array.



DOCUMENT CLASS IMS PAGE NO. 4-165  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. CO05\*3.1 A/B MACHINE SERIES 1700

4. Equivalence considerations may make optimum assignment of array positions a highly difficult analysis. Therefore, arrays will be located at the beginning of the program.

C. Constants, Variables, Temporary Cells

1. Phase B

Phase B will set a flag for a reference to any location which is a candidate for becoming a 1-word command. The flag KRFCNT<sub>1</sub> will be kept in a portion of each symbol table entry. Before execution of Phase C, those counts will be set to 0 which refer to any types of data which have already been located by Phase B (only special cases are located by Phase B).

2. Phase C

Phase C will take all those with the flag set and relocate them in the body of the program such that they immediately follow the first unconditional jump after the first reference which defines the location (i.e., STA, etc.). Phase C will also revise references in the following manner:

- a. The first reference to a cell which falls outside of a 127 cell distance between the cell and its reference is made a two-word command with a relocatable address operand.
- b. Every subsequent reference which falls outside of the mentioned range but within range of the relocatable address operand will be made a relative-indirect reference to that operand.
- c. When a subsequent reference is found which is outside of both ranges, the process begins over again as in 1.
- d. Variables and constants used only in Calling Sequences or others whose references are never candidates for 1-word commands end up with KRFCNT = 0.

D. Temporary Cells

1. Both one-word, two-word and three-word temporary cells will be generated by Phase B to contain integer, real and double precision intermediate results respectively. Temporary storages are the result of three major functions of Phase B.

DOCUMENT CLASS IMS PAGE NO. 4-166  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3-1 A/B MACHINE SERIES 1700

- a. Storage of intermediate computations in arithmetic expressions. Such storage is reusable from one expression to the next and is never needed between expressions.
- b. Special temporary storages for such things as saving indexes, is in use from beginning to end and never usable for anything else. At any rate, the locations for these types of storage are decided upon by Phase B.

## 2. Reuse

Most temporary cells will be used for a brief time during computations and then become unused. It is therefore advantageous to reuse temporary cells as much as possible rather than create new ones.

Temporary storage which is in use or available for use will be kept in a table KTCAT, which will cross-reference their names {symbol table pointers} with information about their type and use. The table will be regularly updated in the following manner:

- a. As temporary cells are needed, the table will be searched for an unused temporary cell of the same type {i.e., integer, real or double precision}. If one is found, it will be noted in use. If one is not found, a new one will be generated and noted in use.
- b. As temporary cells terminate use, the table will be updated to show that that cell is no longer in use. {See TSALOC 4.4.18, for further discussion.}

## 3. Placement

### a. Types

#### 1. Two-word Temporary Storage

It is likely that in most cases all references to two-word temporary storage will occur only in single precision floating point calling sequences. If so, it will be advantageous {as in FORMAT and in most arrays} to place these cells outside of the program body.

DOCUMENT CLASS IMS PAGE NO. 4-166A  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

2. One-word Temporary Cells

It is likely that in most cases all references to one-word temporary storage will occur in instructions which are candidates for one word command and proper placement may improve efficiency to a very great extent.

3. Three-word Temporary Storage

{Same as 3.a.1 on page 4-166}

b. Overlapping

It is conceivable that when a real {two-word} temporary storage becomes free for reuse, that it could be reused for two one-word temporary cells.

Considerations:

1. The ranges of two one-word temporary cells would have to very closely coincide such that they could be made consecutive cells. Such an analysis as this is not practical in Utility FORTRAN;





DOCUMENT CLASS \_\_\_\_\_ IMS \_\_\_\_\_ PAGE NO. 4 LL  
PRODUCT NAME 1700 Mass Storage FORTRAN  
PRODUCT NO. C005 \_\_\_\_\_ VERSION 2.0 \_\_\_\_\_ MACHINE SERIES 1700

- at any rate such a situation is rare.
2. If it were done, the method of placement of one-word cells (as the second word of constant commands) would be precluded and placement would become less efficient.
  3. A typical program will not have 1 and 2-word temporary storages in equal quantities but rather will have mostly one or the other.

Conclusion:

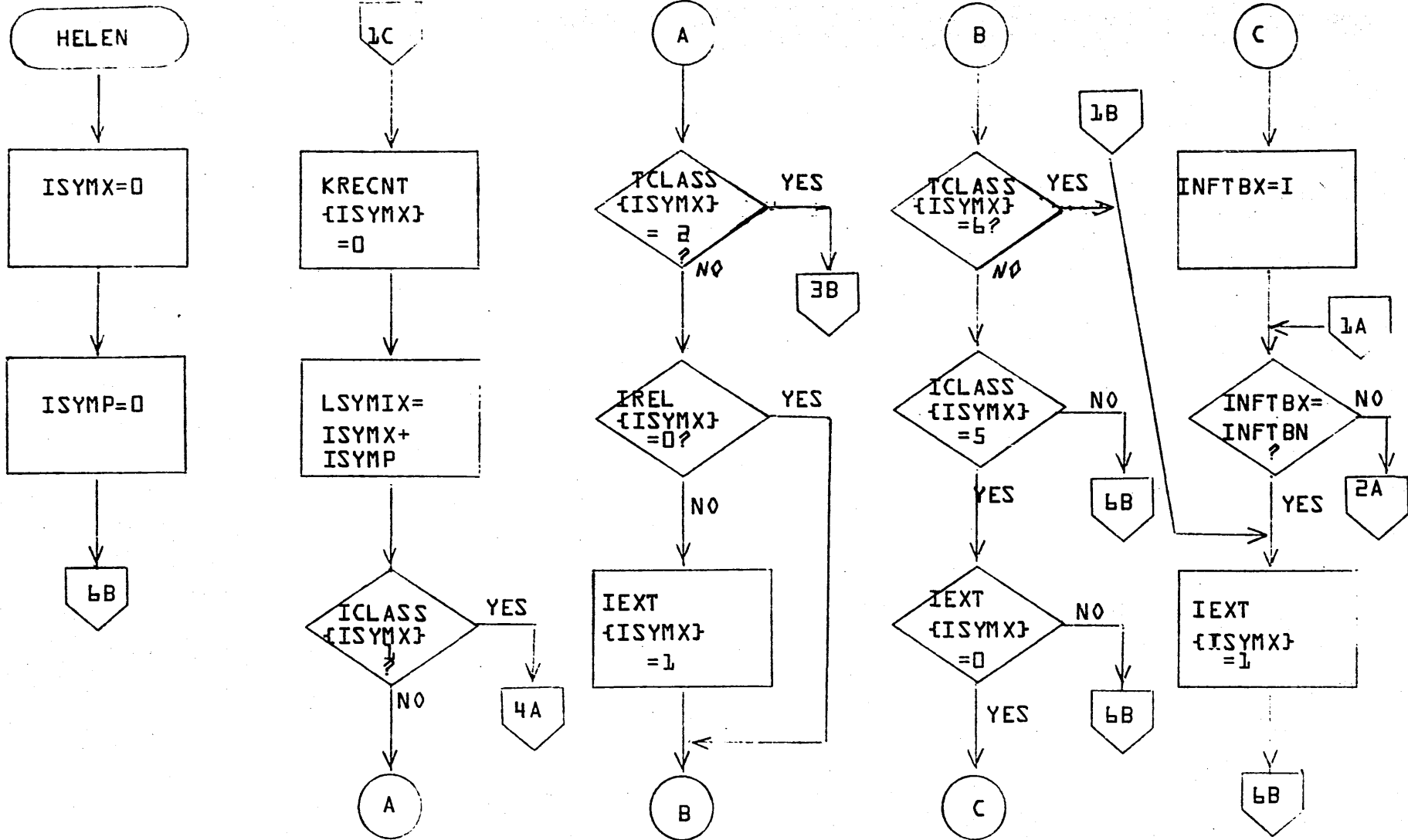
Very little is gained by overlapping. FORTRAN does not attempt to overlap re-use of one-word and two-word temporary storage.

III. Responsibilities of HELEN

The responsibilities of subprogram HELEN, then, are as follows:

- A. To output BSS commands of arrays, variables, including equivalence considerations, and CON commands of constants.
- B. To process functions.
  1. Replace coded names of standard library functions with their library name.
  2. Set up index for in-line functions for use by FINK (see 4.4.6) in generating in-line function instructions.

A  
B  
C  
D



**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

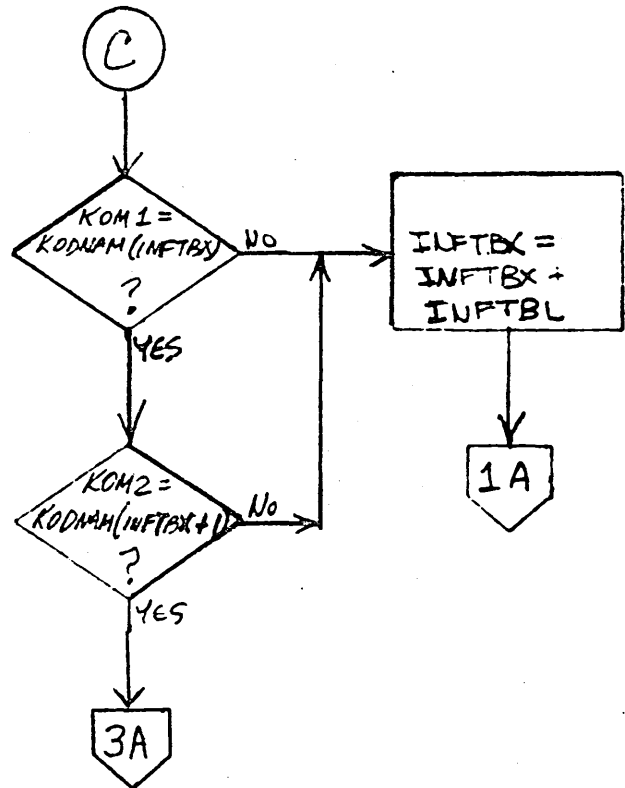
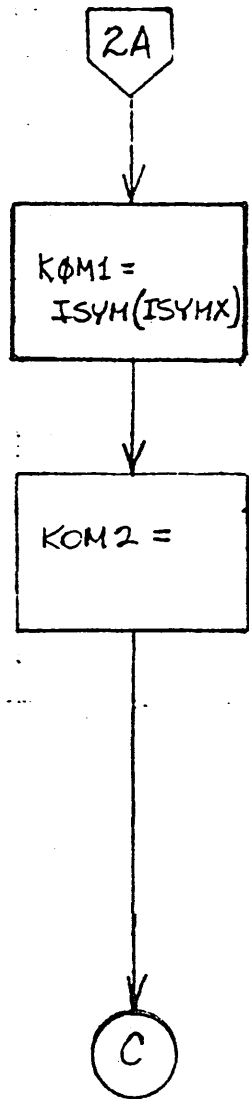
SAMPLE CODE

FLOWCHART

DECISION TABLE

OTHER

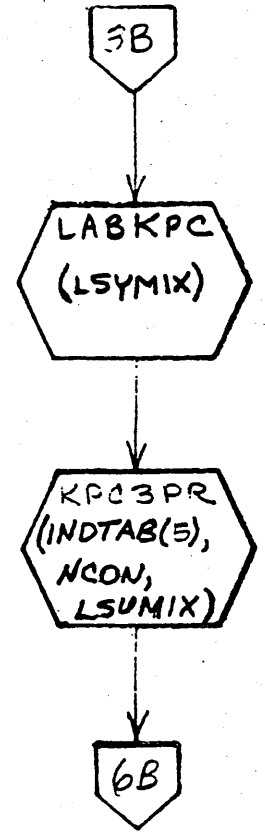
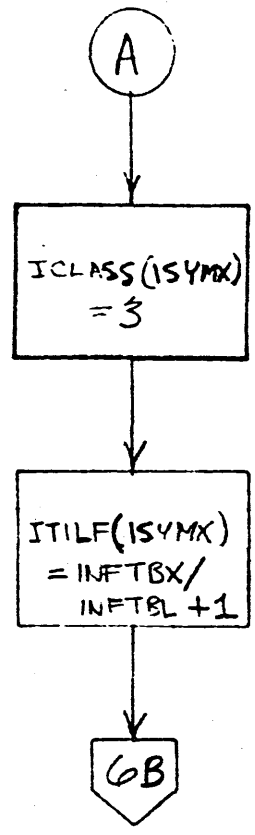
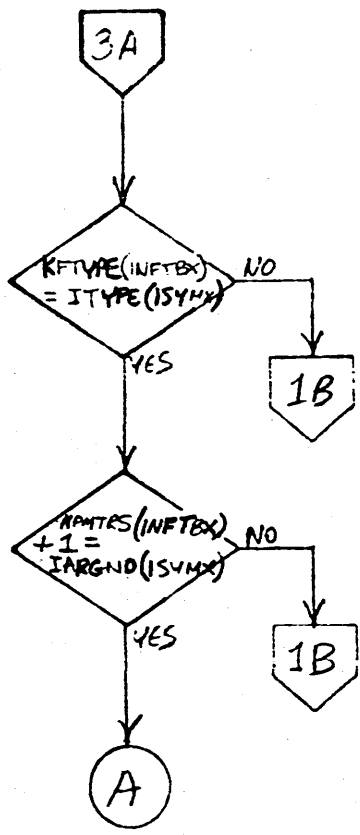
DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	HELEN	PAGE 1 OF 6		PROJECT MGR.			
NUMBER	ISSUE DATE	DATE 5/67		PROJECT NAME			
DRAWN BY				TASK NO.			
				TASK NAME			



**CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT**

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	<i>IAS</i>	MACH. TYPE	<i>1700</i>	PROJECT NO.		APPROVED	DATE
DOCUMENT TITLE	<i>HELEN</i>	PAGE <i>2</i> OF <i>6</i>		PROJECT MGR.			
NUMBER		ISSUE DATE		PROJECT NAME			
DRAWN BY		DATE	<i>3-67</i>	TASK NO.			
				TASK NAME			



**CONTROL DATA CORPORATION**  
 SOFTWARE DOCUMENT

SAMPLE CODE

FLOWCHART

DECISION TABLE

OTHER

DOCUMENT CLASS <i>ZMS</i>	MACH. TYPE <i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE <i>HELEN</i>		PROJECT MGR.			
	PAGE <i>3</i> OF <i>6</i>	PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.			
DRAWN BY	DATE <i>3-67</i>	TASK NAME			

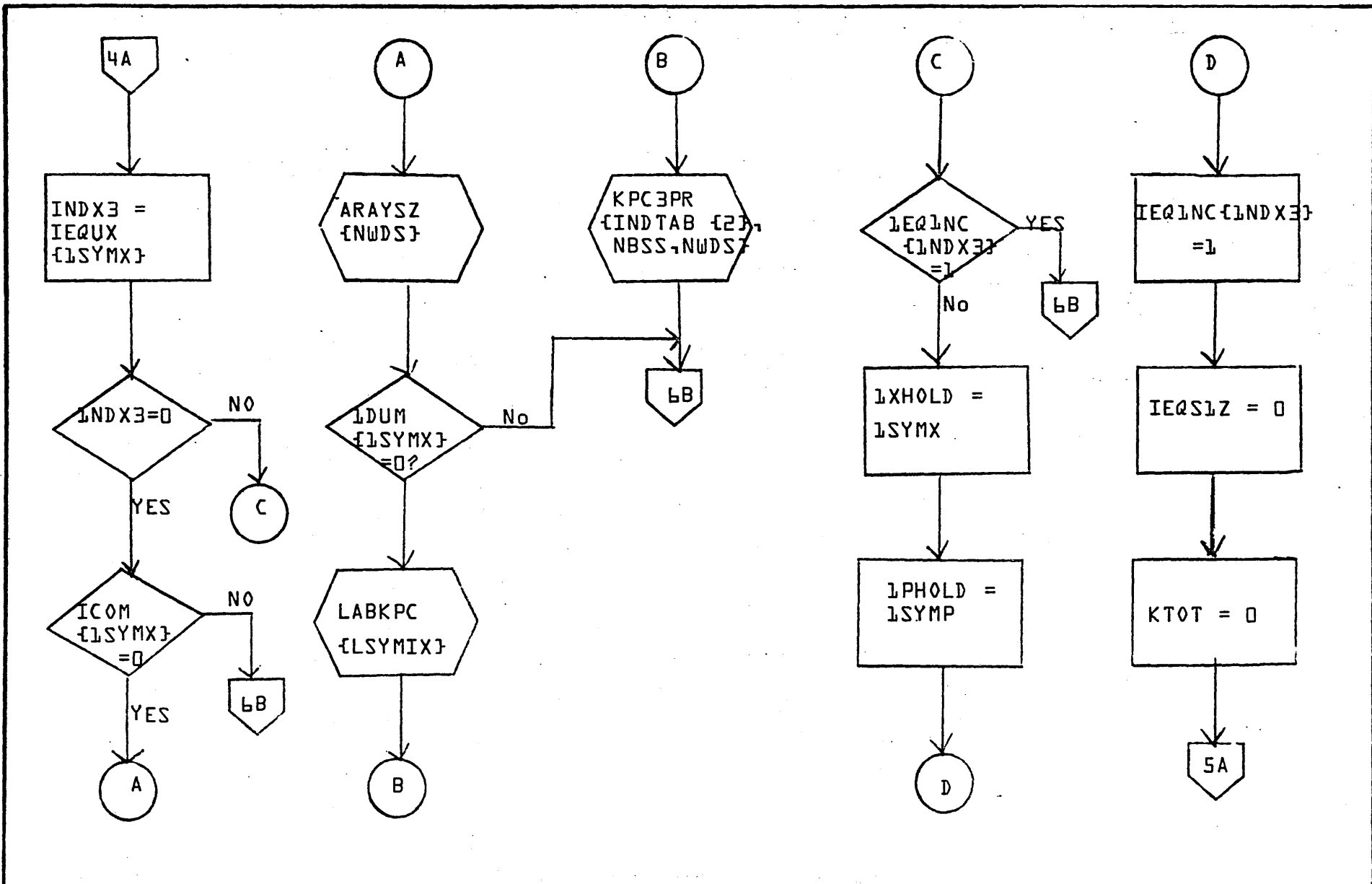
4-170

A

B

C

D



**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	HELEN	PAGE 4 of 6		PROJECT MGR.			
NUMBER	ISSUE DATE			PROJECT NAME			
DRAWN BY	DATE	3/67		TASK NO.			
				TASK NAME			

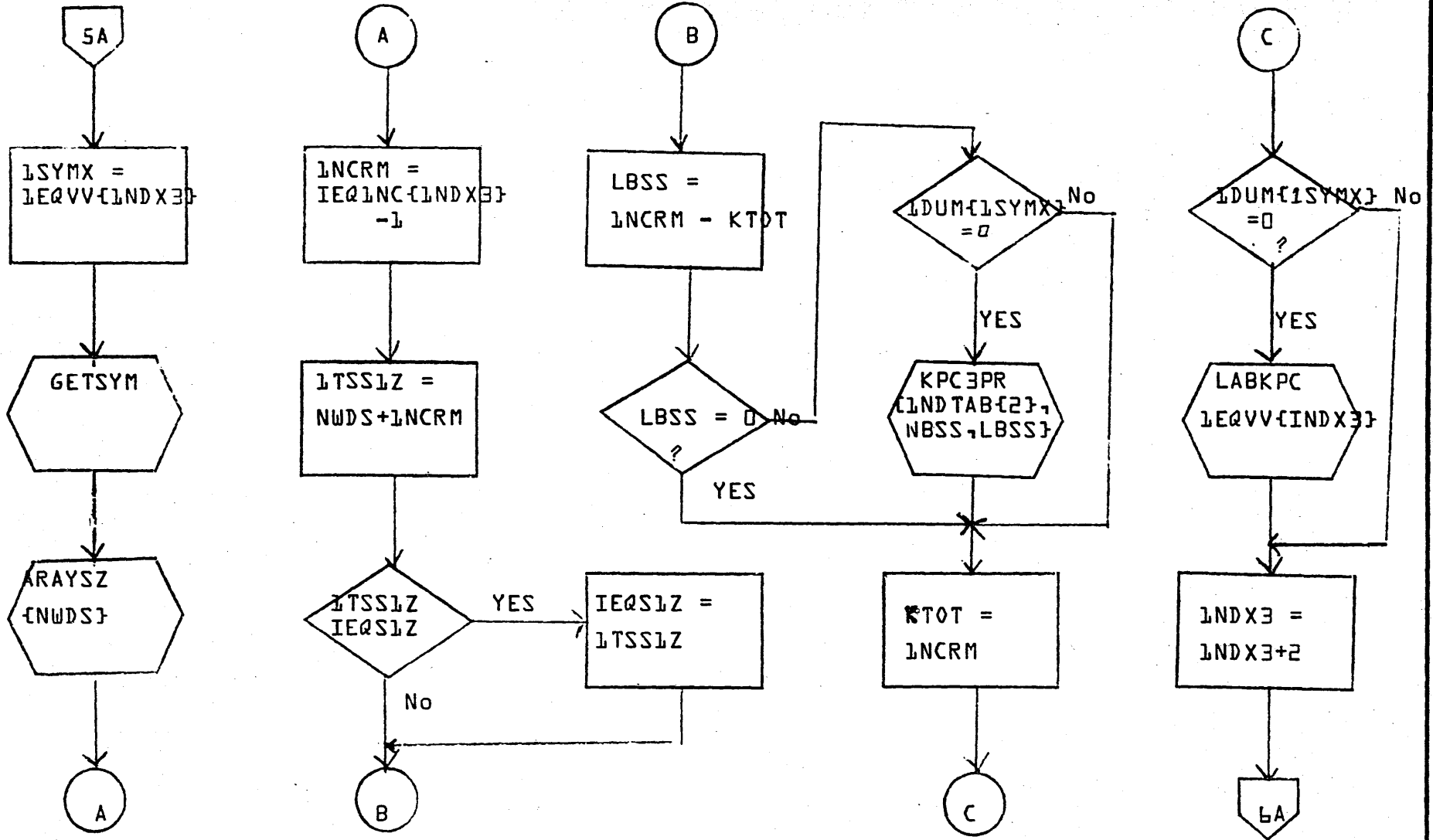
121-4

A

B

C

D



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE   
FLOWCHART   
DECISION TABLE   
OTHER

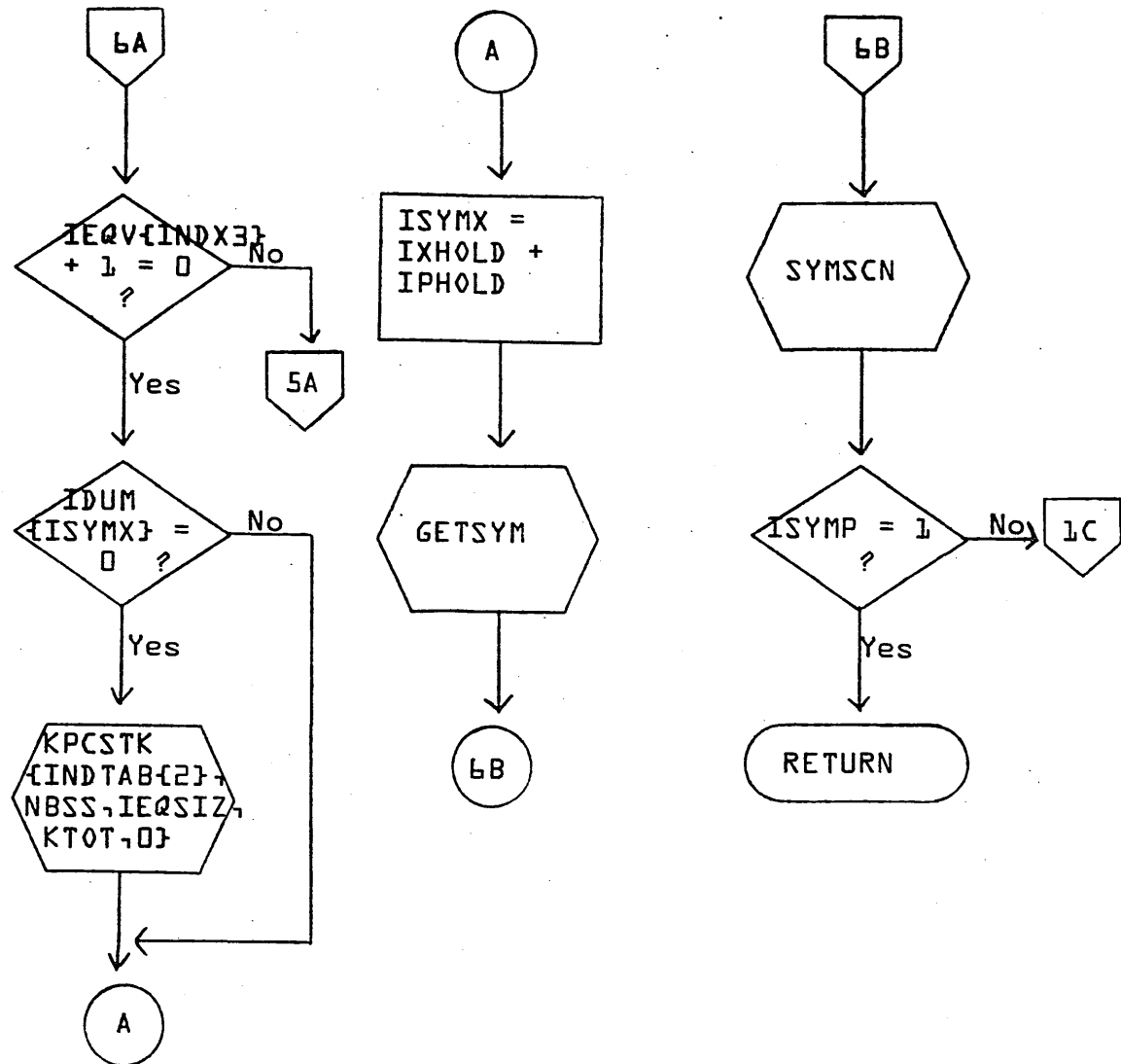
DOCUMENT CLASS	TMS	MACH. TYPE	1700	PROJECT NO.		REV	APPROVED	DATE
DOCUMENT TITLE	HELEN			PROJECT MGR.				
		PAGE	5 OF 6	PROJECT NAME				
NUMBER		ISSUE DATE		TASK NO.				
DRAWN BY		DATE	3/67	TASK NAME				

A

B

C

D



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	HELEN	PAGE 6 OF 6		PROJECT MGR.			
NUMBER	ISSUE DATE	TASK NO.		PROJECT NAME			
DRAWN BY		DATE		TASK NAME			

DOCUMENT CLASS IMC PAGE NO 4-174  
 PRODUCT NAME 1700 Mass Storage FORTRAN  
 PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

## 4.4.20 DUMMY

Subroutine interface for programs calling subroutines or functions generated by FORTRAN or calling sequences generated by FORTRAN.

## I. Calling Sequences

## A. General

Calling sequences written in assembly language and intended to communicate with FORTRAN-generated subroutines, and those generated by FORTRAN from statements or expressions such as: CALL SUB (P<sub>1</sub>, P<sub>2</sub>, P<sub>3</sub>, ..., P<sub>n</sub>) or FUNC(P<sub>1</sub>, P<sub>2</sub>, P<sub>3</sub>, ..., P<sub>n</sub>), will have the following code:

LOC	RTJ*	SUB
LOC+1		address of parameter 1
LOC+2		address of parameter 2
LOC+3		address of parameter 3
⋮		
LOC+N		address of parameter N
LOC+N+1		(return address)

(If the RTJ requires a 2-word relative command, then the first parameter will be at address LOC+2, etc., but this fact has no bearing on the logic of the interface.)

In all cases, parameters are addresses of:

1. variables
2. constants
3. storage locations containing results of computed parameters
4. in the case of I/O calls, actual constants rather than addresses

The CALL below contains examples of each of the first three types of parameters respectively.

```
CALL SUB (VAR, 3.5, A+B)
```

Addresses of parameters occur in consecutive locations following the RTJ command, one cell per address in the order that the parameters appear in the CALL or FUNCTION statement, left to right.



DOCUMENT CLASS IMS PAGE NO. 4-175  
 PRODUCT NAME 1700 Mass Storage FORTRAN  
 PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

### B. Floating Point

Floating point calling sequences include other special data interspersed in the parameter list. Consult floating point specifications for differences. They are, however, germane to the discussion which follows.

### C. Run Anywhere Case

Parameters may consist of two types of addresses:

#### 1. Absolute addresses

- a. All variables which are in COMMON will be generated as absolute addresses.
- b. All variables which are also parameters to the routine which contains the calling sequence will be computed at object time and placed into the calling sequence as absolute addresses. (Excludes floating point calling sequence parameters.)

#### 2. Relative Addresses

These will be generated by FORTRAN for all variables which do not fall into the categories for absolute addresses. This need arises from the necessity to compute effective addresses of parameters at object time. Shifting the location of blocks of code destroys the integrity of any absolute addressing which references variables internal to that block. What will be termed "self relative" addresses (which is what is meant by a relative address in a calling sequence) will be determined in the following manner:

The address of the parameter (for example parameter 3) minus the location of the "self relative" address in the calling sequence (LOC+3), or:

$$\text{self relative address of parameter 3} = (\text{address of } P_3) - (\text{address LOC}+3)$$

#### 3. Bit 15

Only the fifteen low order bits of a parameter (14-0) are absolutely necessary to express a parameter whether absolute or relative.

##### a. Non-floating point calls

In all calling sequences (except floating point) bit 15 then, is used as a flag to distinguish between the two addressing modes in that:

DOCUMENT CLASS TMS PAGE NO. 4-176  
PRODUCT NAME 1700 Mass Storage FORTRAN  
PRODUCT NO. CD05 REVISION 2.0 MACHINE SERIES 1700

bit 15 = 0, if this parameter is an absolute address.  
bit 15 = 1, if this parameter is a relative address.  
In cases of forward references, the bit would have to be set; in backward references, the bit would already be set. Thus part of the computation of a relative address (see entry routine) involves masking bit 15 off and extending bit 14 to the left.

b. Floating point calls

Floating point calls employ a different method of distinguishing between relative and absolute addresses. Bit 15, for relative addresses, is artificially set to 0. Absolute addresses already have bit 15 = 0.

D. Non-Run Anywhere Case

1. Non-floating point calling sequences

Parameters will consist only of absolute addresses as mentioned in 3a but will never require object-time computation, since programs will never enter memory except through the loader. Parameters still must be plugged by the entry routine but no computation of effective address is required.

2. Floating point calling sequences

The contrasting design of the floating point package now takes advantage of the indirect addressing ability of the 1700. Parameters which are indirect addresses are given a 1 in bit position 15. Its precise use will be outlined under paragraph B.

## II. Subroutine or Function Entry

A. Entry routines will have the following general responsibilities:

1. Picking up parameters and, if necessary, biasing the ones which are relative and plugging the addresses wherever necessary.
2. Computing and setting up the proper return address to the calling program.
3. Initializing the execution of the main part of the subroutine or function.

B. The task of properly linking up instructions and other references to external parameters begins at the start of instruction generation. The Subroutine Argument Table (KSUBAT) and the Statement Function Argument Table (KSFAT) are built and updated in the following manner.

DOCUMENT CLASS \_\_\_\_\_ IM: \_\_\_\_\_ PAGE NO. 4-177  
PRODUCT NAME 1700 Mass Storage FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

## 1. Initializing

### a. Function Subprograms

The arguments are all placed in KSUBAT and made empty. (I.e., the table is initialized to show no increments.)

Since the name of the function subprogram logically applies to the value of the function, but physically must apply to the entry point, the following procedure will be used to create the distinction:

Two separate symbol table entries will be created for that name by PASS1 and both will be passed on to PHASE 4. The symbol table entry which refers to the entry point label will be differentiated from the symbol table entry which refers to the label of the function value by having bit 15 of ISYM(1) set to 1. In the latter ISYM(1) will be 0. In fact, all symbol table entries which are entry points (i.e., to include names of subroutines as well) will have this bit set.

### b. Statement Functions

Statement Functions do not compute their values into labeled locations nor are they external to the calling programs. Thus, only a single symbol table entry is used and bit 15 is 0.

All arguments (including the additional one if function is floating point) are initialized into the Statement Function Argument Table (KSFAT) in precisely the same manner as described for function subprograms.

### c. Subroutines

Subroutine name entries in the symbol table will also have bit 15 set as noted above. Since subroutines pass results back through parameters they need no extra label nor do they need an extra parameter. Otherwise all arguments are initialized into KPCT precisely in the same manner as described for function subprograms.

## 2. Updating (This is the function of subroutine DUMMY)

### 2a. Updating KSUBAT

Each time an instruction operand or a calling sequence operand is encountered which is an argument to the subroutine being compiled, it is paired with the additive (or increment) of that reference and considered unique from any other argument (or the same argument but different additives).

DOCUMENT CLASS IMS PAGE NO. 4-17B  
PRODUCT NAME 1700 Mass Storage FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

The unique argument additive is looked up in KSUBAT to see if it exists. If not, it is added to the table such that the additives are sorted lowest to highest. Also, a temporary storage cell is assigned to it and the reference to it is modified to a reference to the temporary storage cell as already described.

When a temporary storage cell is assigned to replace argument references a special bit, KDUMMY, is set in the symbol table entry for that temporary storage.

#### 2b. Updating KSFAT

Updating KSFAT follows exactly the same procedure as outlined for KSUBAT, except that the instruction operands and calling sequence operands concerned are not those external to the program being compiled but rather external to the Statement Function being compiled (i.e., dummy arguments of the function).

#### 3. Phase C

Phase C must perform the following functions for both run-anywhere and non-run-anywhere:

- a. When the first reference is encountered to a temporary cell (of the type described above, i.e., KDUMMY = 1) it is made a two word command such that the operand is the contents of the address located in the second half of this instruction. The address of the second word of the instruction is kept and cross-referenced with the temporary cell.
- b. All subsequent references (less than 128 cells distant) are made indirect references to the address portion of that command.
- c. When a reference is encountered outside of the range the process begins over again as in step 1.
- d. Whenever a reference is discovered in a calling sequence, the address of that parameter's location in the calling sequence is noted as in 1. Subsequent references may then refer back to this location subject to the same restrictions.
- e. When Phase C encounters the pseudo-instruction PST, the operand portion will contain a pointer to a temporary cell (of the type described above). At this point, PHASE5 must generate a STA command into each one of the addresses cross-referenced with that temporary cell. Phase B will already have generated all the preliminary code to compute the address of the parameter and its increment.

DOCUMENT CLASS \_\_\_\_\_ IMS \_\_\_\_\_ PAGE NO. 4-17  
PRODUCT NAME 1700 Mass Storage FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

#### 4. Entry Routines

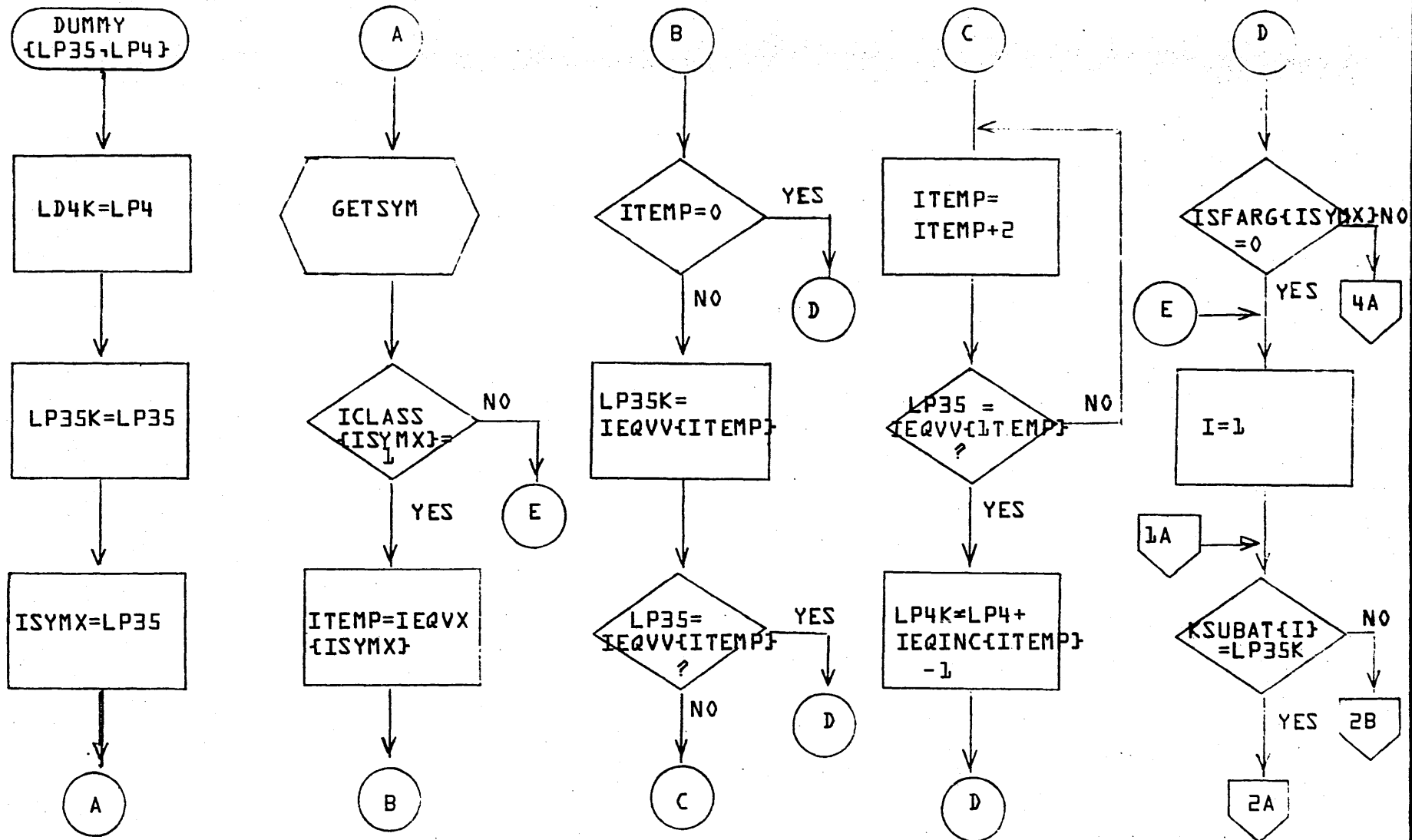
The entry routine is generated for each Statement Function and again for the whole Subroutine or Function Subprogram. (See ENTCOD 4.4.4.)

A

B

C

D


**CONTROL DATA CORPORATION**  
 SOFTWARE DOCUMENT

 SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER 

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	DUMMY	PAGE 1 OF 4		PROJECT MGR.			
NUMBER	ISSUE DATE	DATE 3-67		PROJECT NAME			
DRAWN BY				TASK NO.			
				TASK NAME			

1

2

3

4

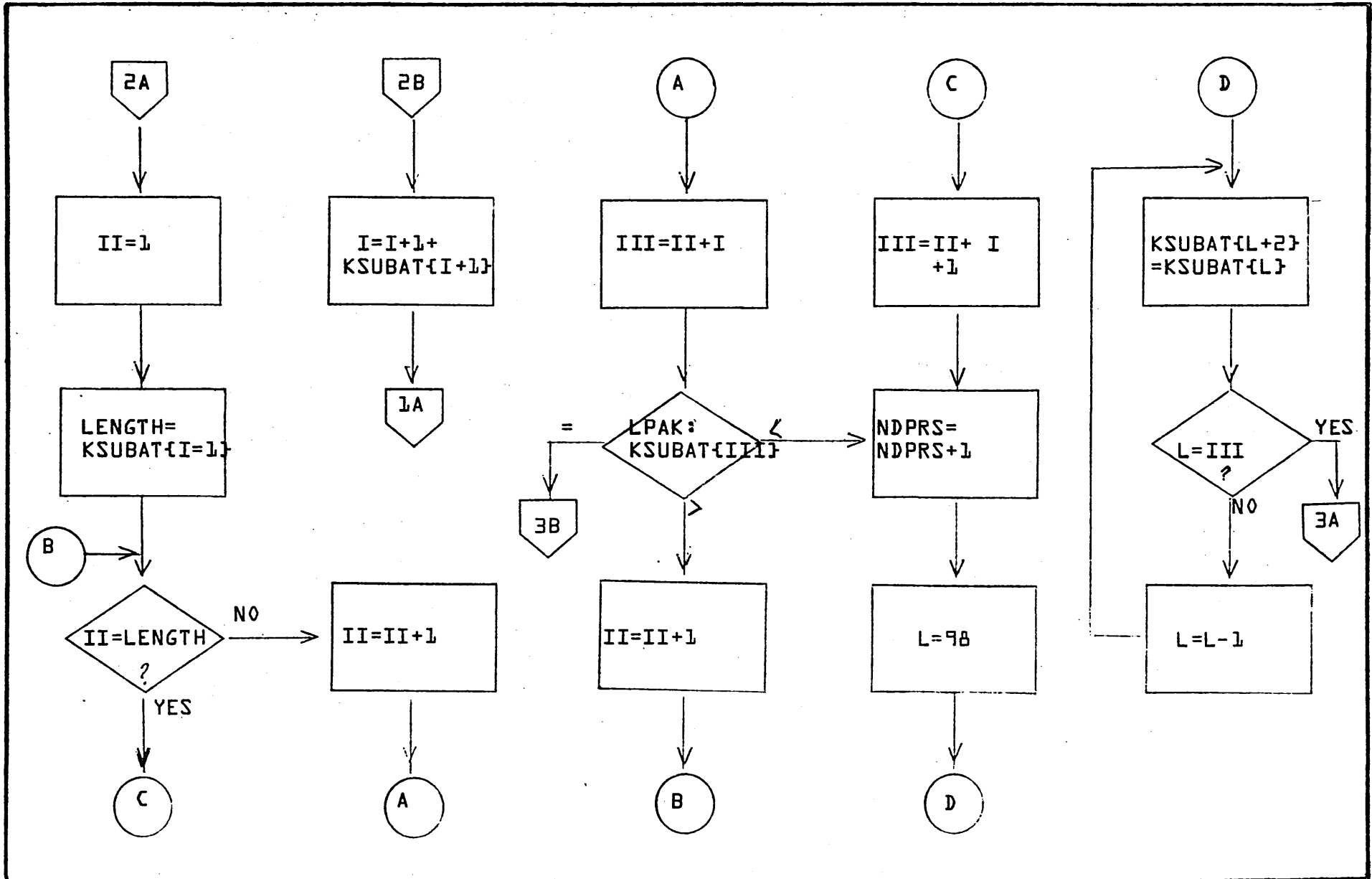
5

A

B

C

D



**CONTROL DATA CORPORATION**  
 SOFTWARE DOCUMENT

SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	DUMMY	PAGE 2 OF 4		PROJECT MGR.			
NUMBER	ISSUE DATE	3/67		PROJECT NAME			
DRAWN BY	DATE			TASK NO.			
				TASK NAME			

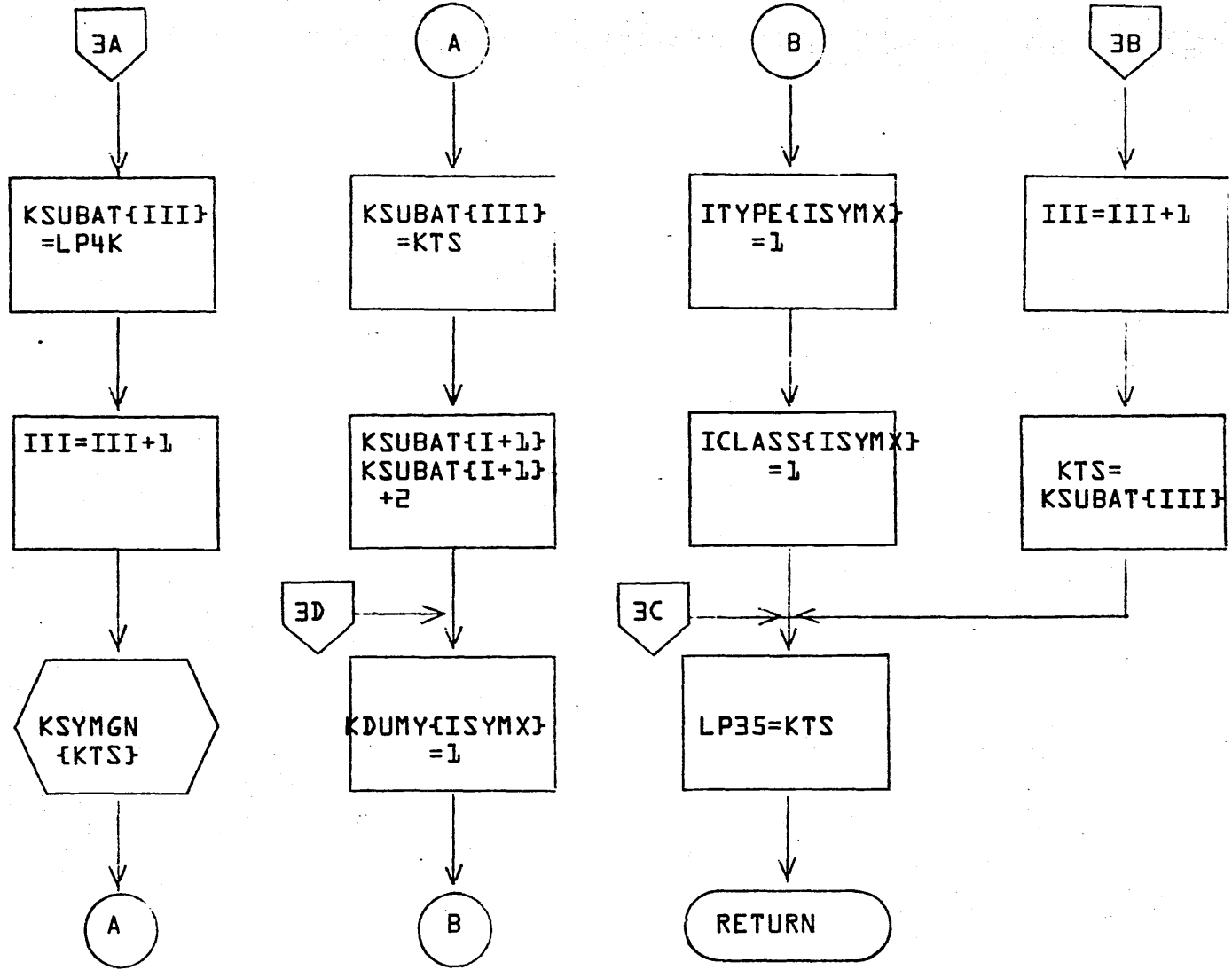
4-181

A

B

C

D



**CONTROL DATA CORPORATION**  
 SOFTWARE DOCUMENT

SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	DUMMY	PAGE 3 OF 4		PROJECT MGR.			
NUMBER	ISSUE DATE	DATE 3/67		PROJECT NAME			
DRAWN BY	DATE 3/67		TASK NO.	TASK NAME			

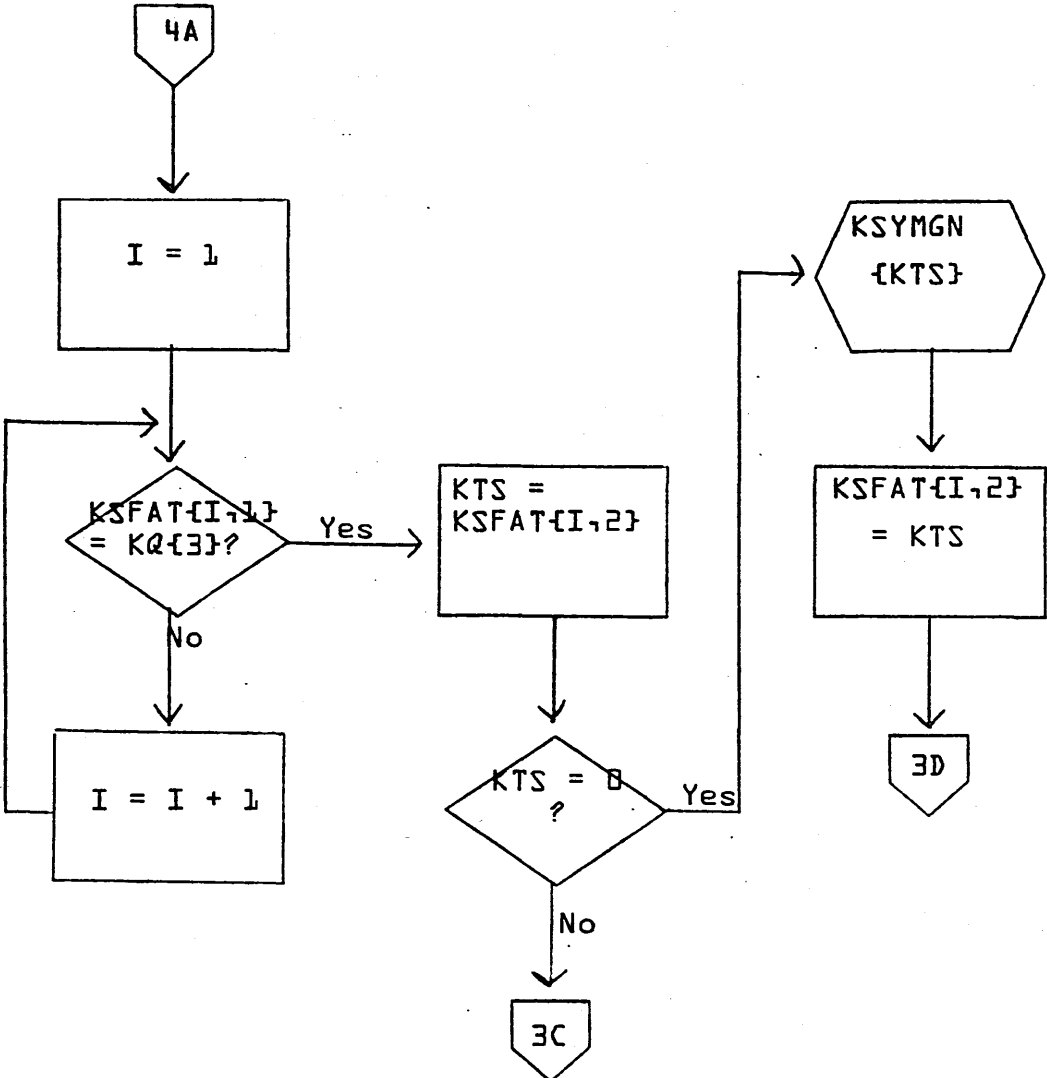


A

B

C

D



**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	DUMMY	PAGE 4 OF 4		PROJECT MGR.			
NUMBER	ISSUE DATE			PROJECT NAME			
DRAWN BY	DATE			TASK NO.			
				TASK NAME			

4-183

DOCUMENT CLASS IMS PAGE NO 4-184  
PRODUCT NAME 1700 Mass Storage FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

KPCSTK

KPCSTK outputs pseudo code. KPCSTK updates the accumulator memory table (KART). KPCSTK adjusts addresses to reflect additives and subscripts. Some optimization is performed by checking sizes of constants (conversion of LDA commands to ENA wherever possible).

Input parameters:

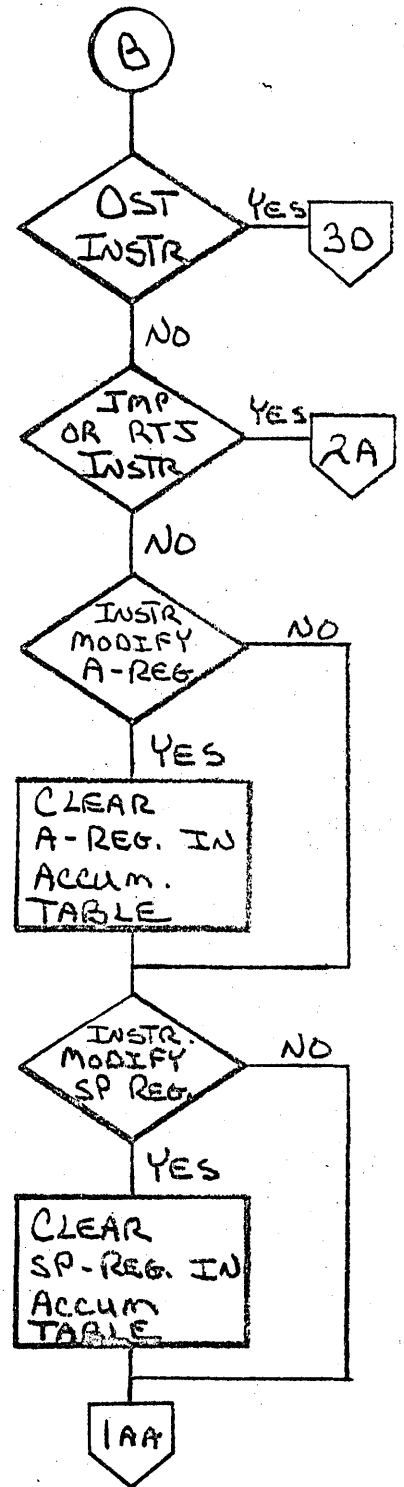
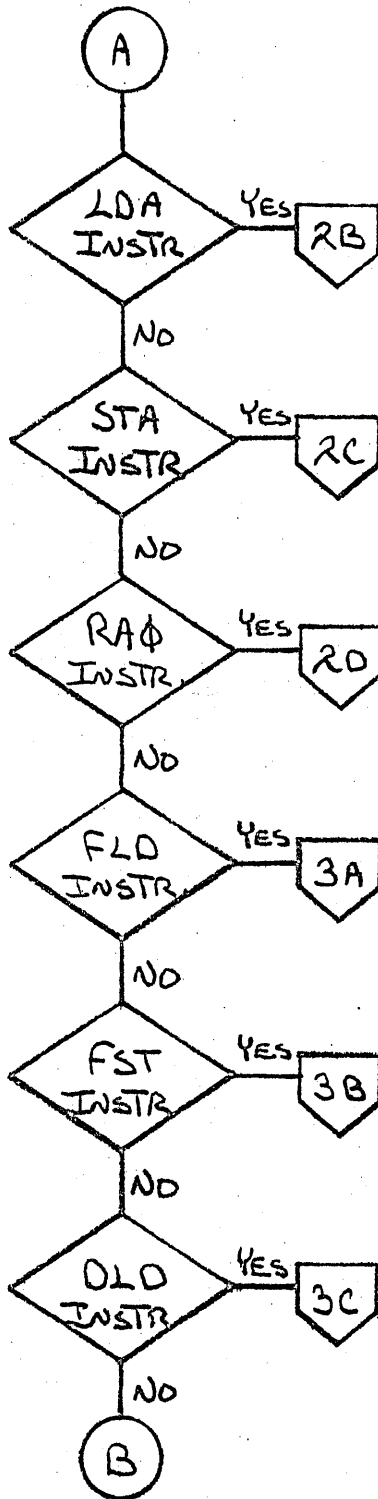
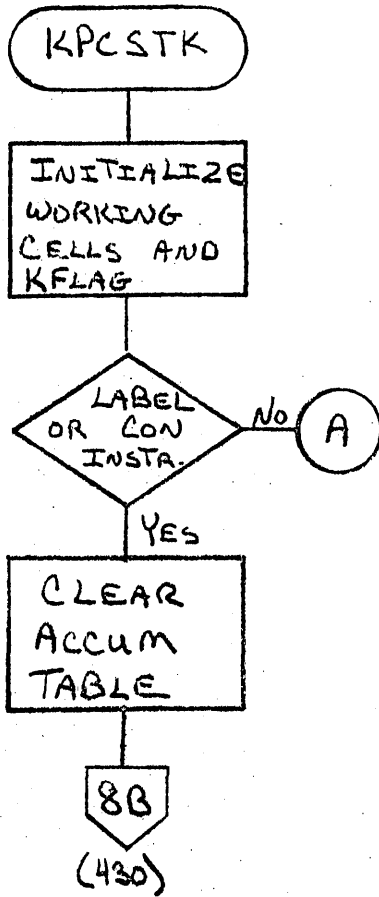
1. Indicator entry from INDTAB
2. Operation code
3. Operand (constant or symbol table pointer)
4. Additive value to operand
5. Symbol table pointer to variable subscript of operand

KPCSTK calls:

KPOUT  
FCMSTK  
DUMMY  
GETSYM  
KCPART



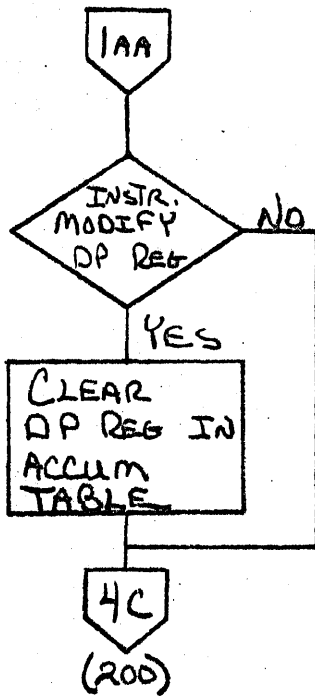
LA JOLLA FACILITY



TITLE		KPCSTK		DRG. NO.	
DRAWN BY		PROJ.		REVISION	
DATE		SHEET /		OF 9	



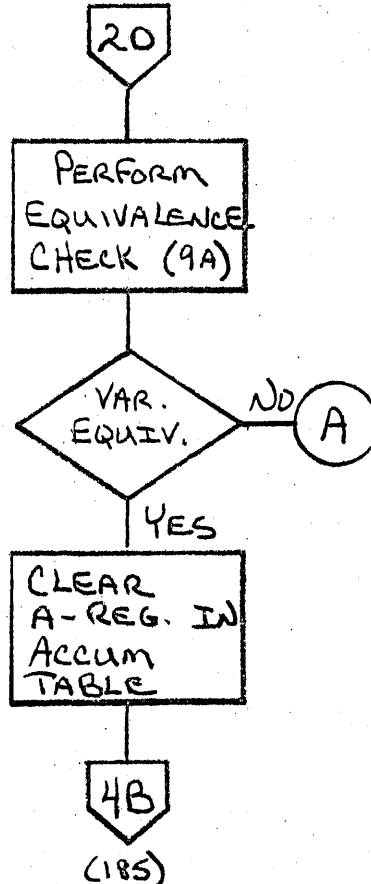
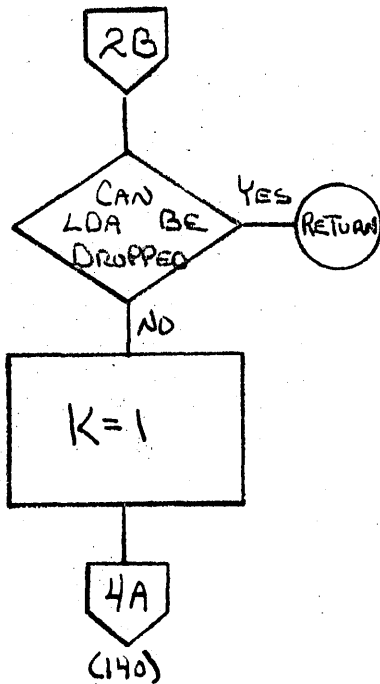
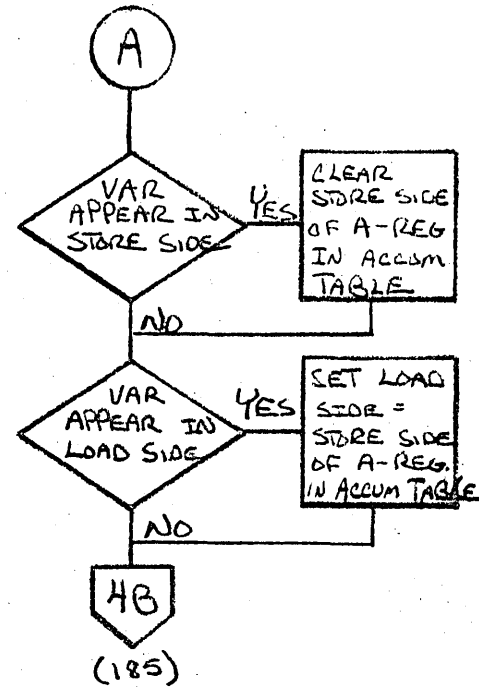
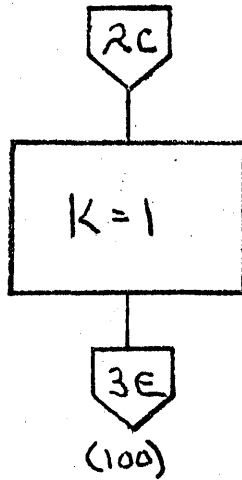
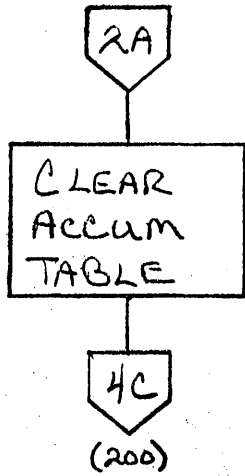
LA JOLLA FACILITY



TITLE		KPCSTK		DRG. NO.	
				REVISION	
DRAWN BY	PROJ.	DATE	SHEET	1A	OF 9



LA JOLLA FACILITY

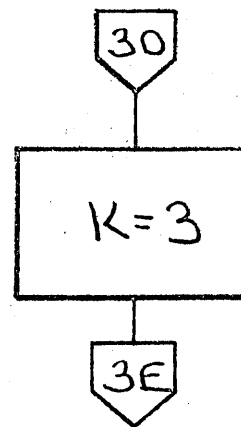
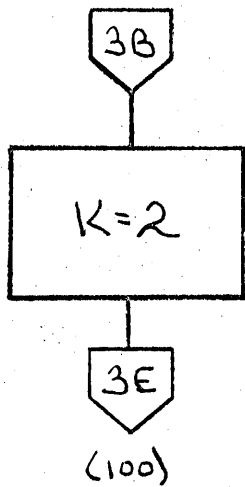
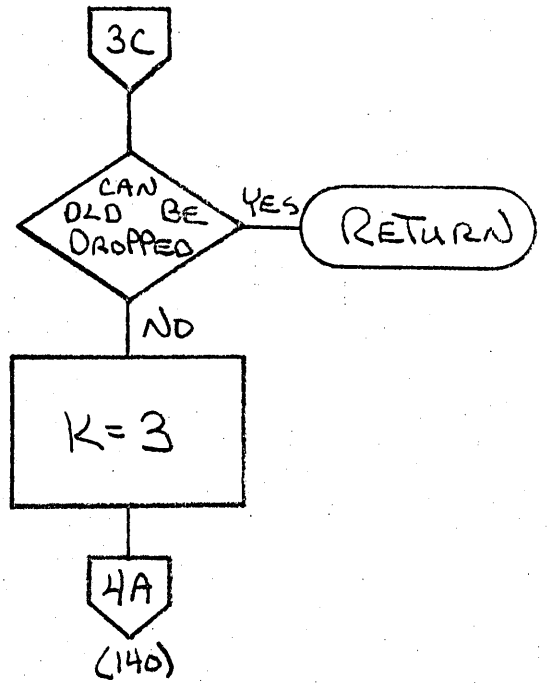
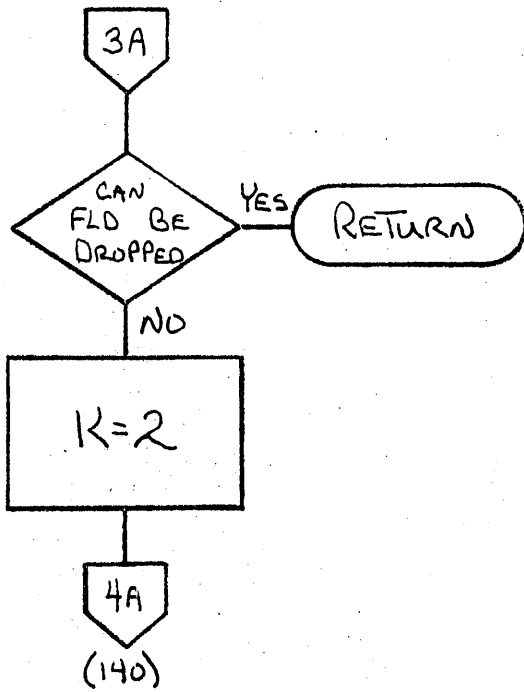


TITLE		KPCSTK		DRG. NO.	
DRAWN BY		PROJ.		REVISION	
CSD 203		DATE		SHEET 2 OF 9	





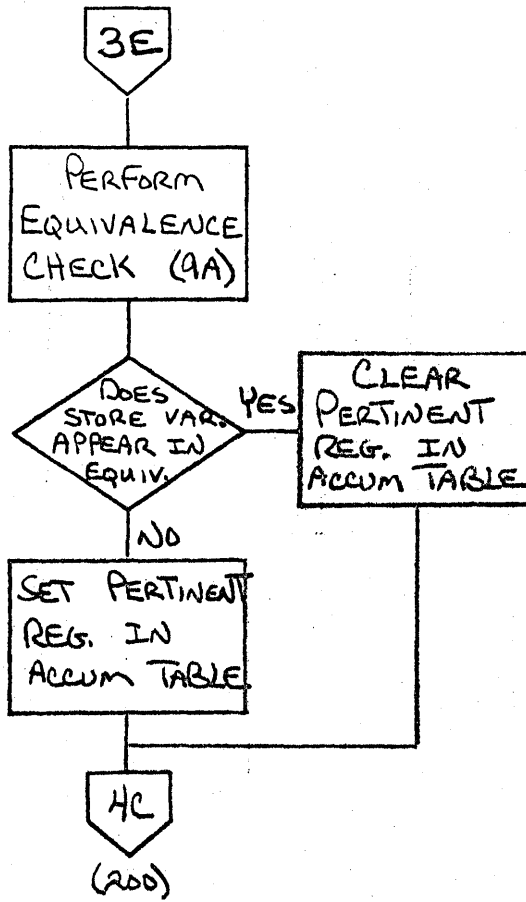
LA JOLLA FACILITY



TITLE		KPCSTK		DRG. NO.	
				REVISION	
DRAWN BY	PROJ.	DATE	SHEET 3	OF 9	
CSD 203					



LA JOLLA FACILITY

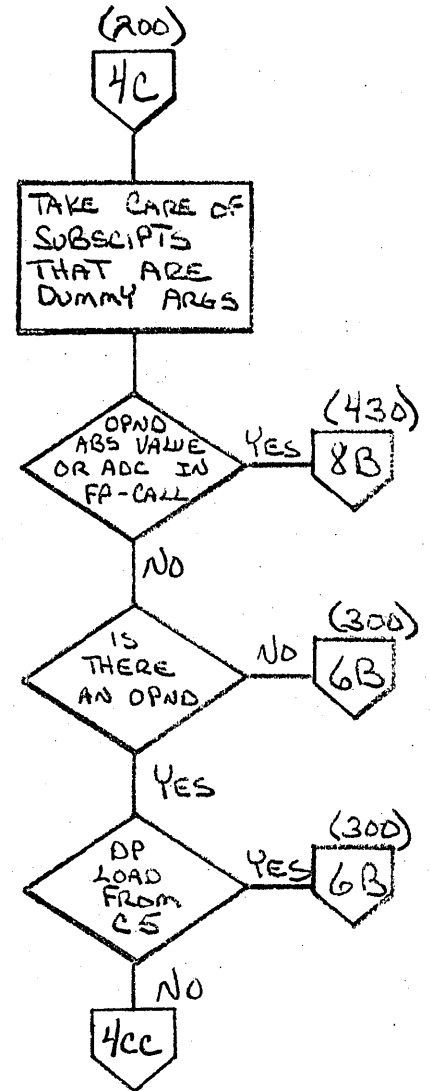
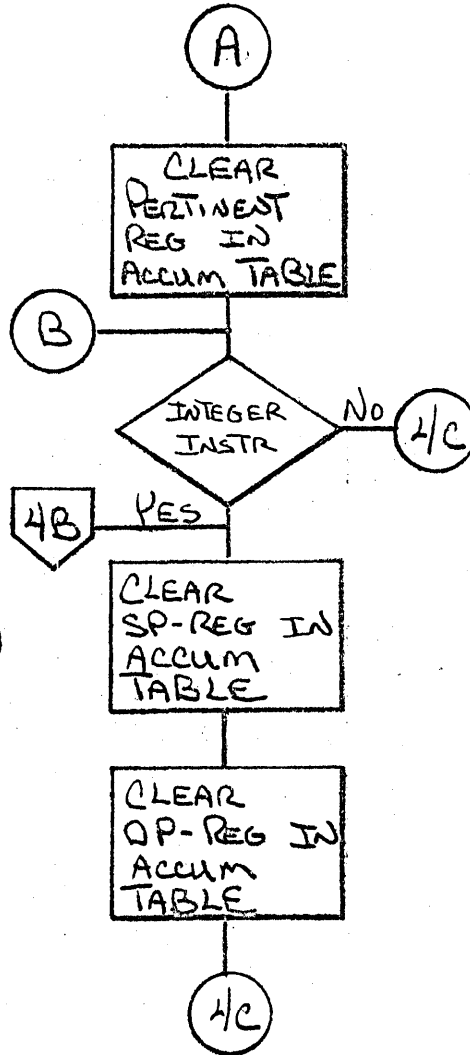
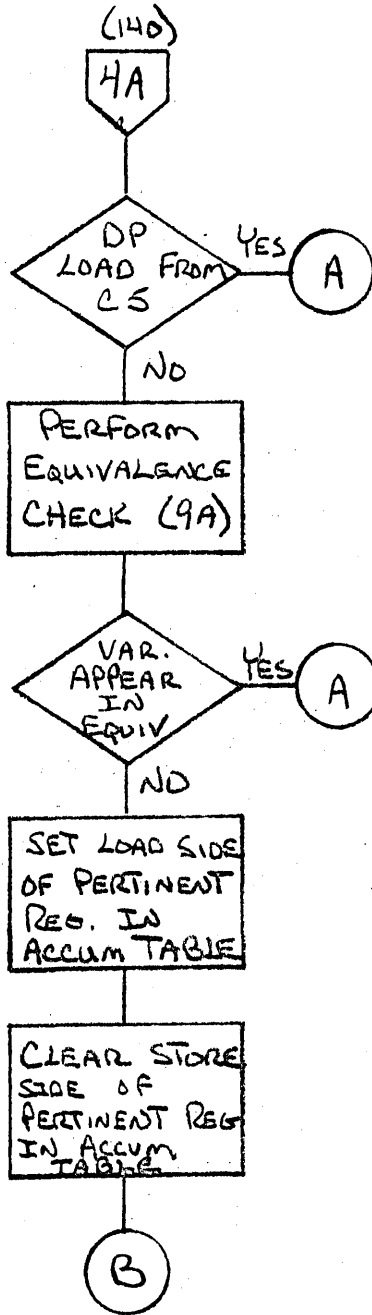


TITLE		KPCSTK		DRG. NO.	
				REVISION	
DRAWN BY	PROJ.	DATE	SHEET 3A	OF 9	





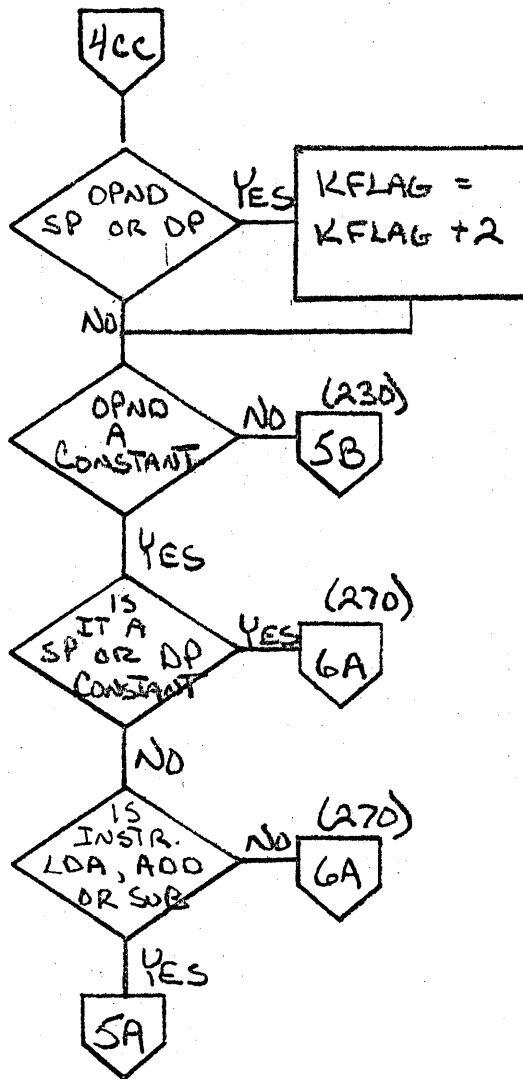
LA JOLLA FACILITY



TITLE		KPCSTK		DRG. NO.	
DRAWN BY		PROJ.		REVISION	
DATE		SHEET 4		OF 9	



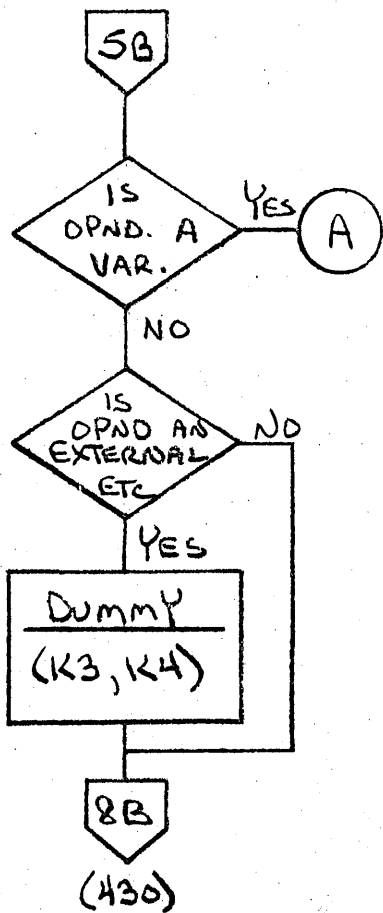
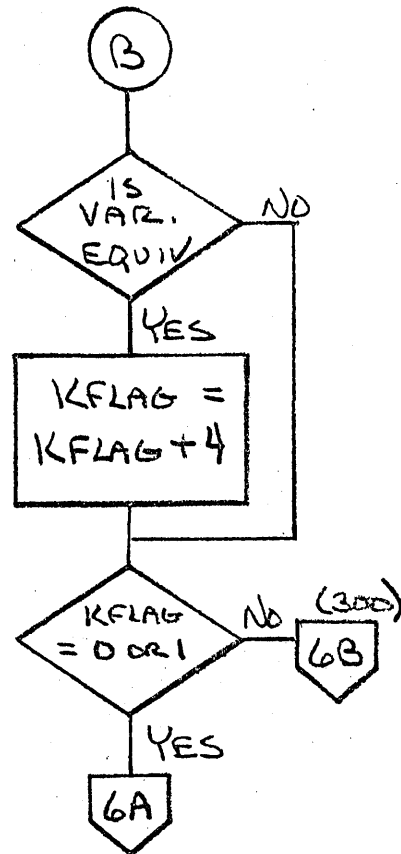
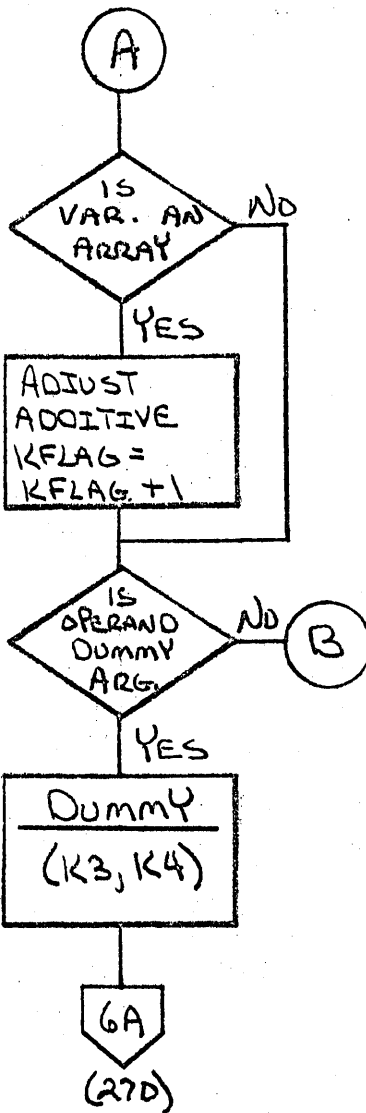
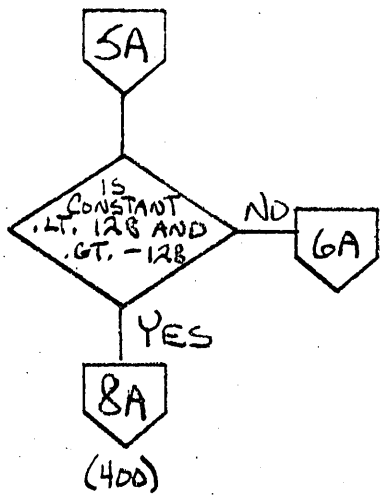
LA JOLLA FACILITY



TITLE		KPCSTK		DRG. NO.	
				REVISION	
DRAWN BY	PROJ.	DATE	SHEET 4A	OF 9	
CSD 203					



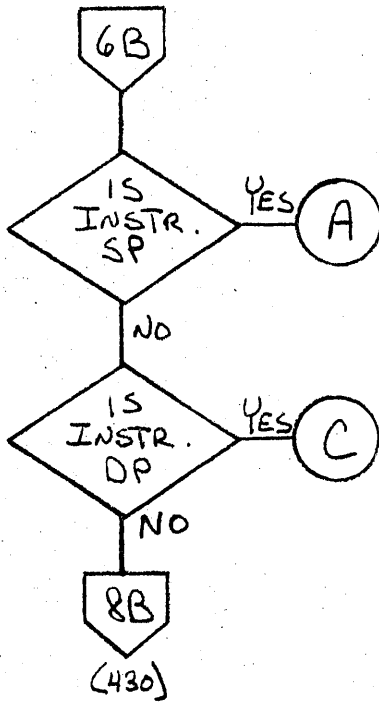
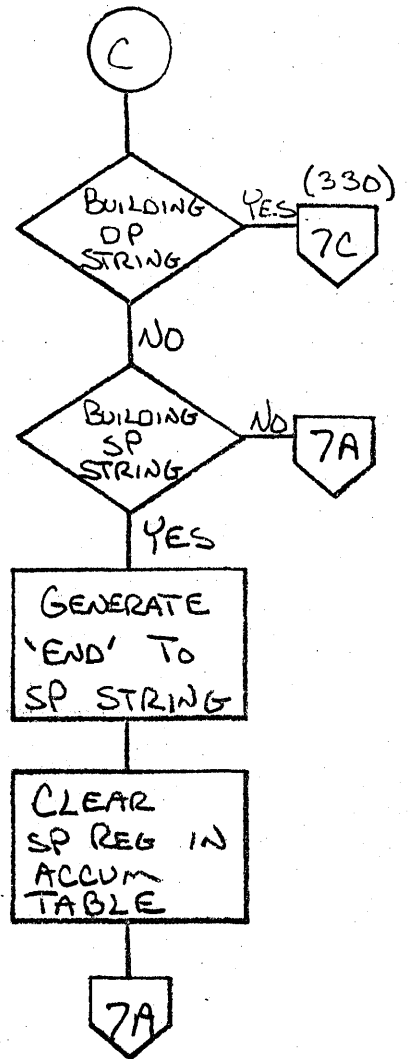
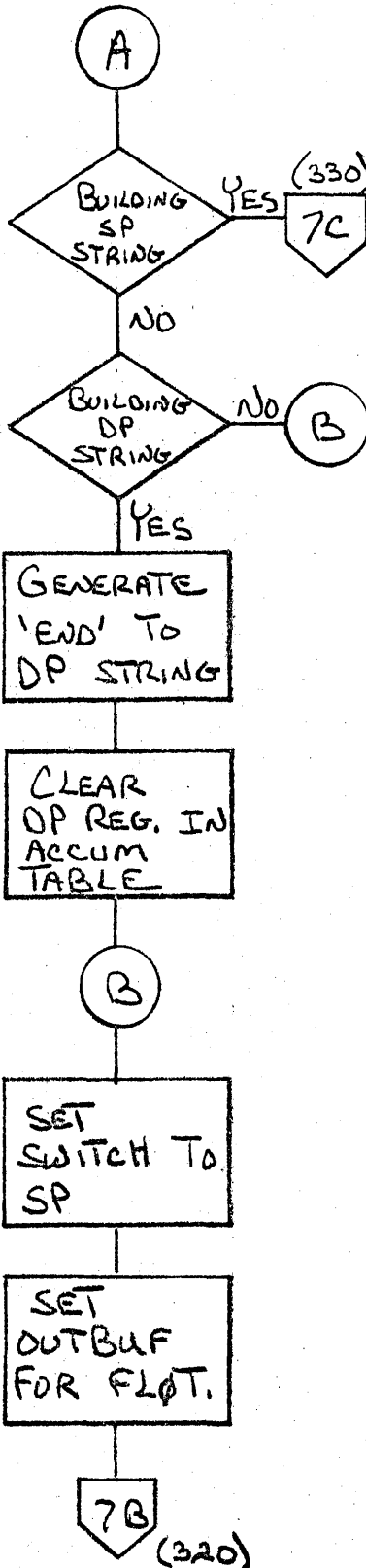
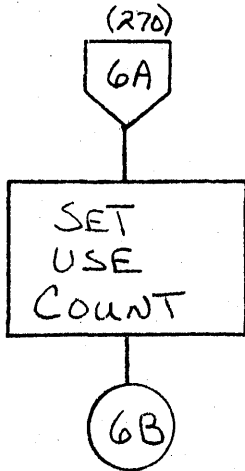
LA JOLLA FACILITY



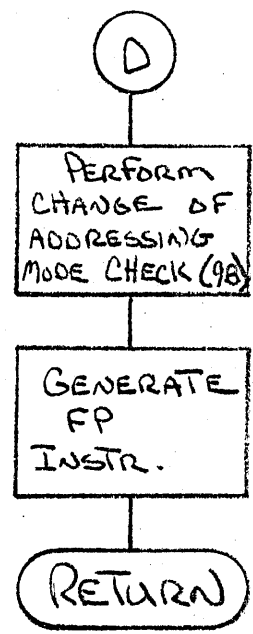
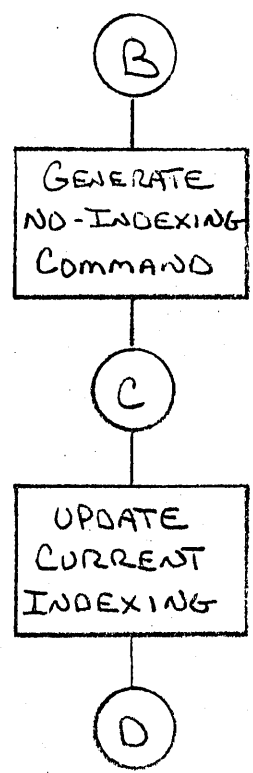
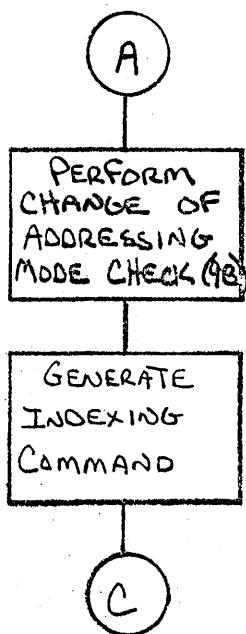
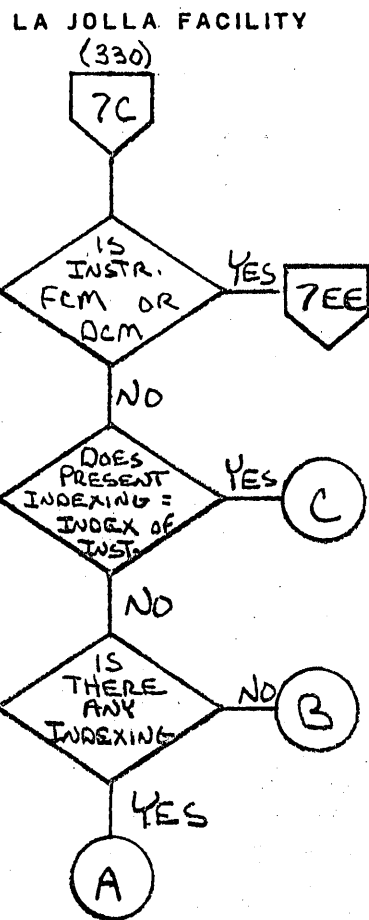
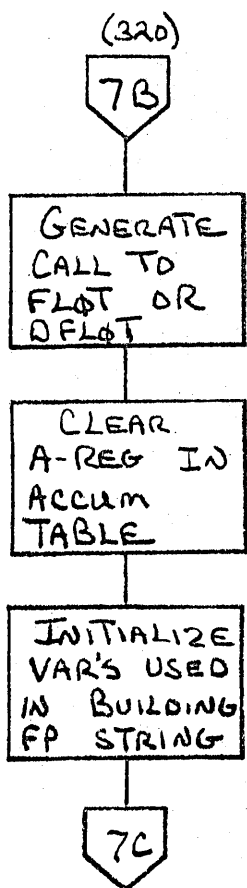
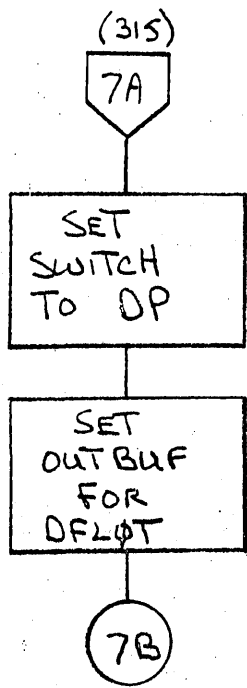
TITLE		KPCSTK		DRG. NO.	
				REVISION	
DRAWN BY	PROJ.	DATE	SHEET	5	OF 9



LA JOLLA FACILITY



TITLE			KPCSTK		DRG. NO.	
DRAWN BY			PROJ.		REVISION	
DATE			SHEET 6		OF 9	

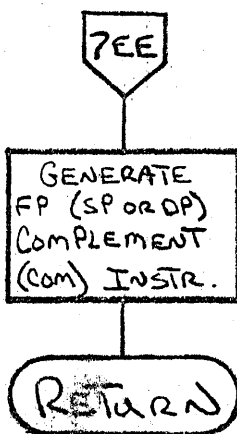


TITLE		KPCSTK		DRG. NO.	
DRAWN BY		PROJ.		REVISION	
CSD 203		DATE		SHEET 7 OF 9	



LA JOLLA RESOURCE CENTER  
IMS Page 4-192  
1700 MASS STORAGE FORTRAN  
C005\*3.1 A/B

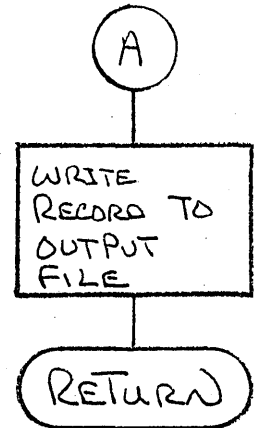
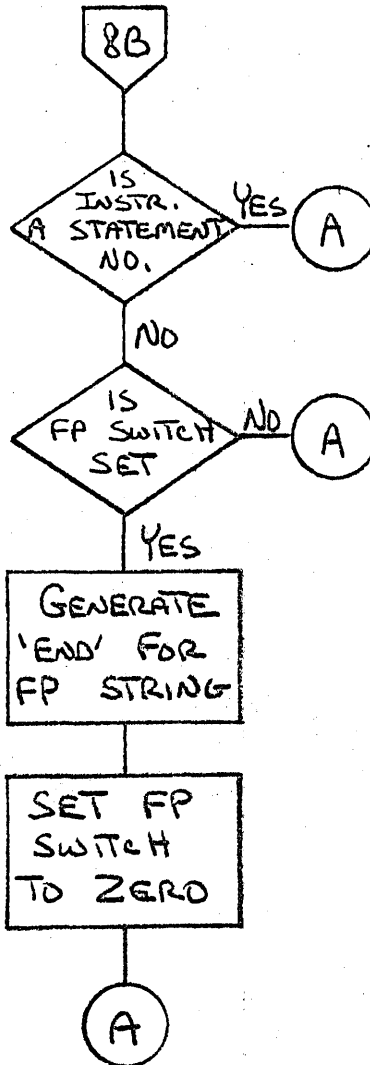
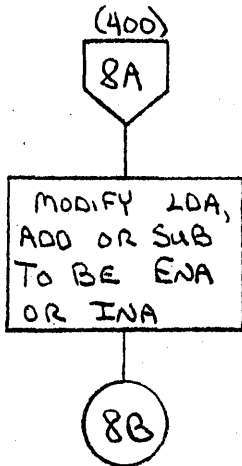
LA JOLLA FACILITY



TITLE		KPCSTK		DRG. NO.	
				REVISION	
DRAWN BY	PROJ.	DATE	SHEET	7A	OF 9



LA JOLLA FACILITY

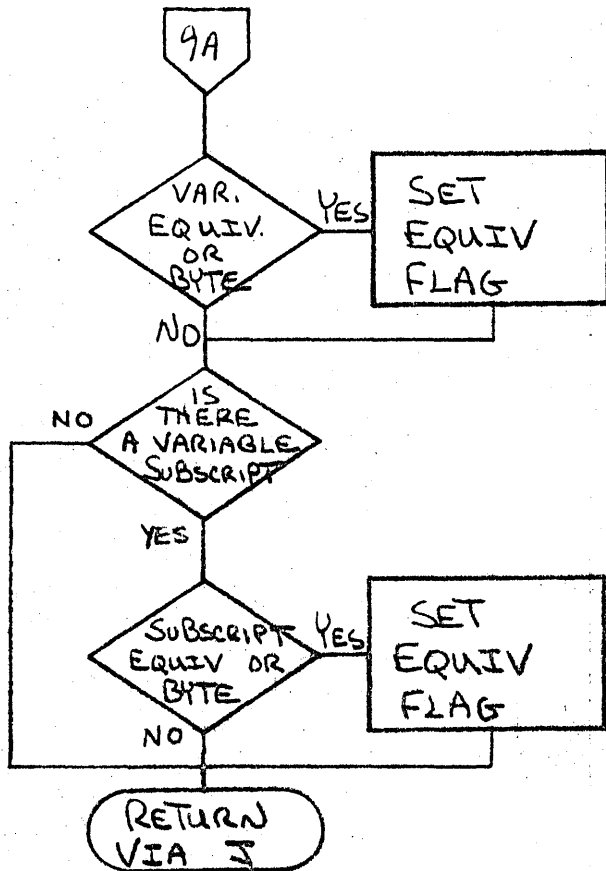


TITLE		KPCSTK		DRG. NO.	
DRAWN BY		PROJ.		REVISION	
CSD 203		DATE		SHEET 8 OF 9	

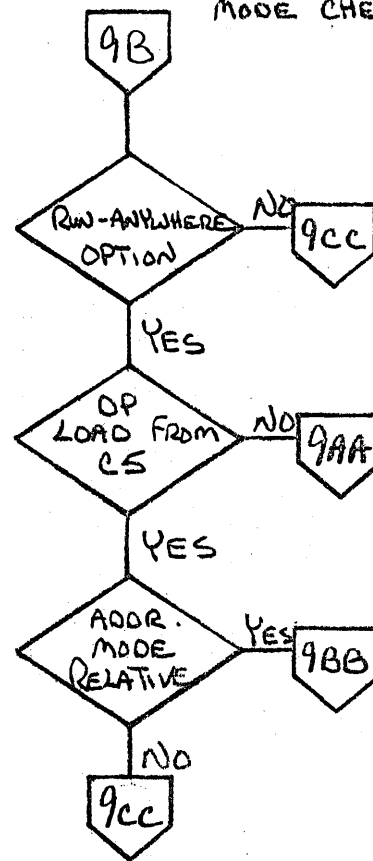


LA JOLLA FACILITY

(190 - EQUIVALENCE CHECK)



(380 - CHANGE OF ADDRESSING MODE CHECK)

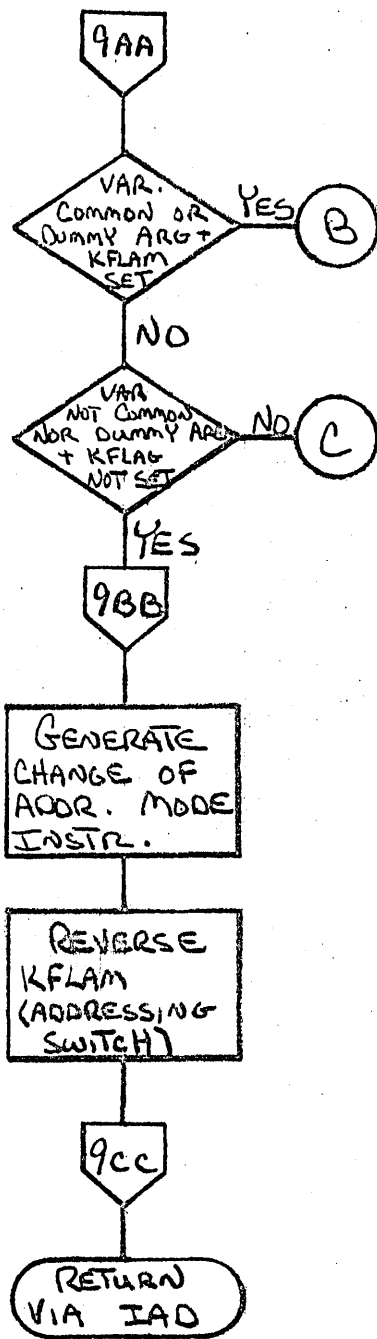


TITLE		KPCSTK		DRG. NO.	
				REVISION	
DRAWN BY	PROJ.	DATE	SHEET 9	OF 9	





LA JOLLA FACILITY



TITLE		KPCSTK		DRG. NO.	
				REVISION	
DRAWN BY		PROJ.	DATE	SHEET 9A	OF 9

CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 4-196  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. 005 V.2.0 MACHINE SERIES 1700

4.4.21 KSYMGN

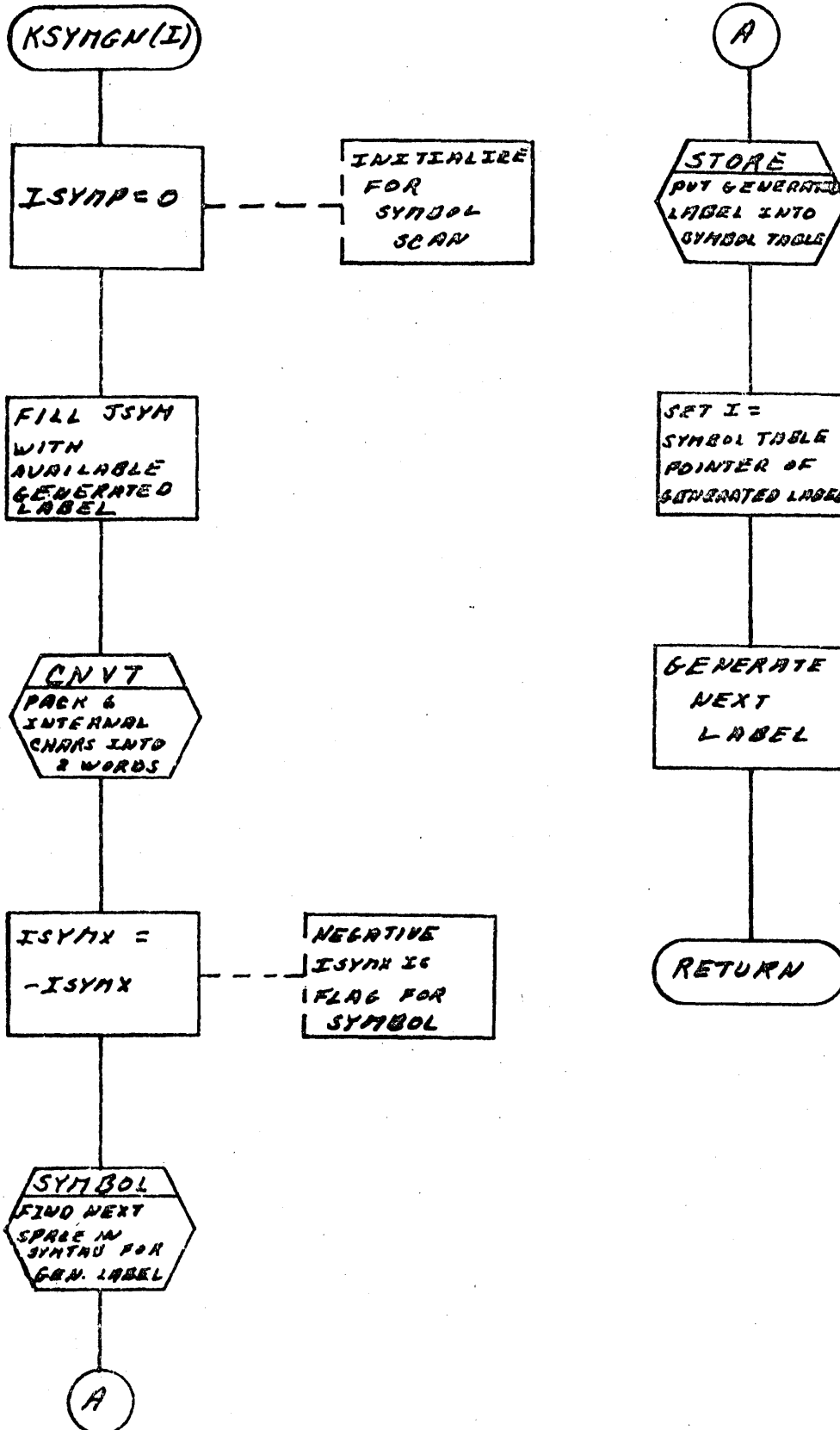
KSYMGN generates variable names used internally in the compilation of a routine as labels, temporary storage, or dummy parameters. The variables are stored in the symbol table and the pointer to the symbol table entry is passed to the calling routine in the parameter I.

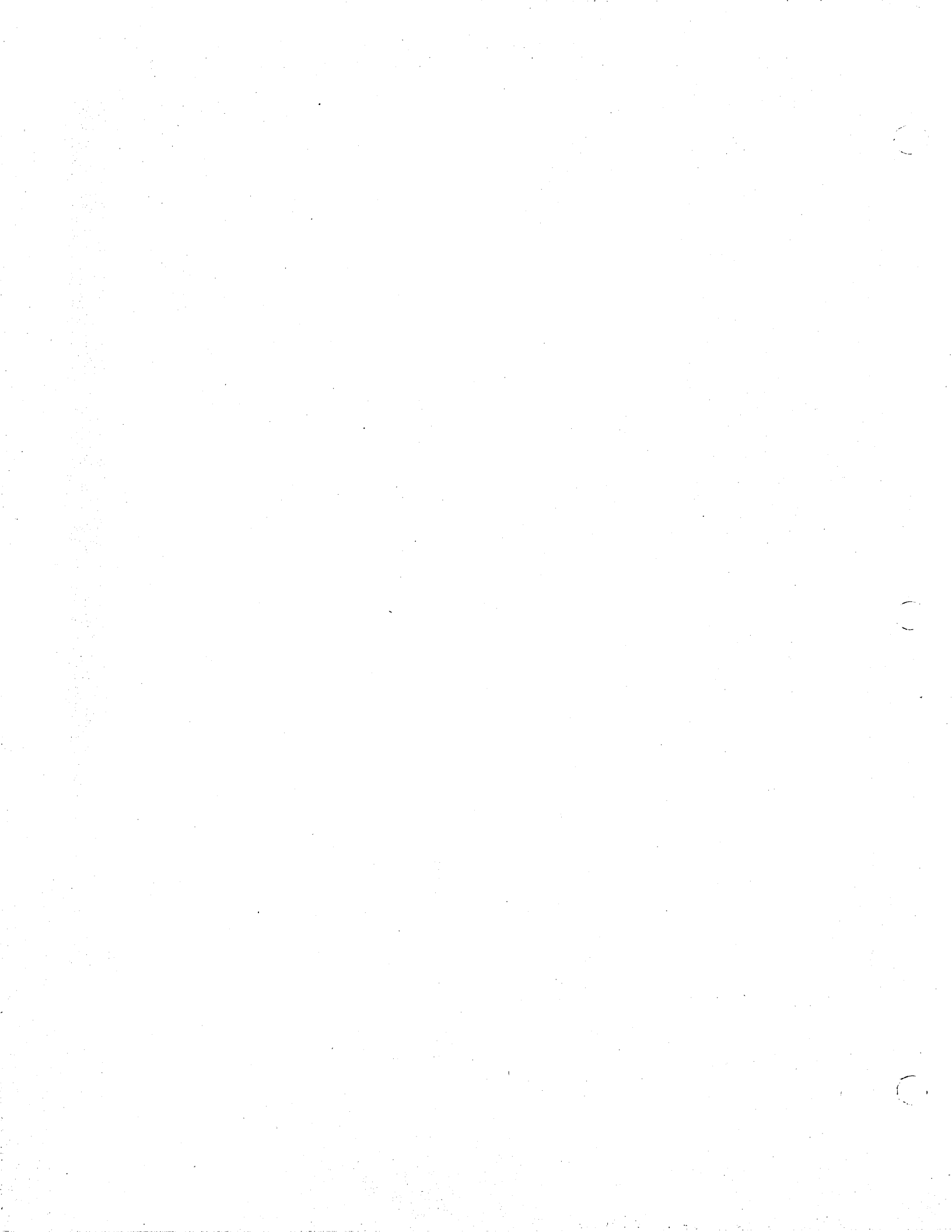
The generated names are of the form .hhhhh where h denotes a hexadecimal digit and the first generated name is .00000.

KSYMGN calls:

CNVT  
STORE  
SYMBOL

DOCUMENT CLASS IMS PAGE NO. 4-197  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. COO5 V.2.0 MACHINE SERIES 1700





DOCUMENT CLASS IMS PAGE NO. 5-1  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. COO5 V.2.0 MACHINE SERIES 1700

## Chapter 5

## TABLE OF CONTENTS

- 5.0 Assembler Phases C and D/E
- 5.1 Assembler Input Phase C
  - 5.1.1 Pseudo Instructions
  - 5.1.2 Data Strings
  - 5.1.3 FORMAT Records
  - 5.1.4 Record Formats
    - 5.1.4.1 Input Record Instruction Codes
    - 5.1.4.2 Breakdown of Input Record Indicator
    - 5.1.4.3 Input Pseudo Instruction Record Formats
    - 5.1.4.4 Output Record Formats
    - 5.1.4.5 Assembler Instruction Type Codes
    - 5.1.4.6 Breakdown of Output Record Indicator
    - 5.1.4.7 Phase C and D/E Common Blocks
  - 5.1.5 Subroutine PHASEC
  - 5.1.6 Subroutine BKDWN
  - 5.1.7 Subroutine BLDUP
  - 5.1.8 Subroutine BSS
  - 5.1.9 Subroutine CHKWD
  - 5.1.10 Subroutine CHOP
  - 5.1.11 Subroutine CL12
  - 5.1.12 Subroutine CON
  - 5.1.13 Subroutine DATAST
  - 5.1.14 Subroutine INOUT
  - 5.1.15 Subroutine IXOPT
  - 5.1.16 Subroutine LABEL
  - 5.1.17 Subroutine LABIN
  - 5.1.18 Subroutine QXLD
  - 5.1.19 Subroutine SKIP
- 5.2 Assembler Output Phase
  - 5.2.1 Subroutine PHASE6
  - 5.2.2 Subroutine BEGINO
  - 5.2.3 Subroutine AMOUT
  - 5.2.4 Subroutine ADMAX
  - 5.2.5 Subroutine CONV
  - 5.2.6 Subroutine COUNT
  - 5.2.7 Subroutine FINISH
  - 5.2.8 Subroutine IACON
  - 5.2.9 Subroutine IHCON
  - 5.2.10 Subroutine INDEX
  - 5.2.11 Subroutine LABOUT
  - 5.2.12 Subroutine NPUNCH
  - 5.2.13 Subroutine NP2OUT

DOCUMENT CLASS IMS PAGE NO. 5-2  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005 V.2.0 MACHINE SERIES 1700

## 5.2 Assembler Output Phase (continued)

5.2.14 Subroutine NWRITE  
5.2.15 Subroutine PACK  
5.2.16 Subroutine RBDX  
5.2.17 Subroutine RBPK  
5.2.18 Subroutine SETPRT  
5.2.19 Subroutine TABDEC  
5.2.20 Subroutine UNPUNC

DOCUMENT CLASS IMS PAGE NO. 5-3  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005 V.2.0 MACHINE SERIES 1700

## 5. Assembler Phases C and D/E

Phases C and D/E are the assembler phases of the FORTRAN compiler. Their function is to assign storage locations, assign index registers, optimize use of one-word addressing and produce relocatable binary and list output.

Phase C is the input phase of the assembly process, and Phase D/E is the output phase. Subroutine PHASEC is the main processor for the input phase. The intermediate pseudo-instruction output of phase C is written on a disk scratch file, from which it is read by PHASE6, the main processor for the output phase D/E.

### 5.1 Assembler Input Phase C

Three types of records are handed to the assembler by the previous pass; pseudo instructions, data strings and format records.

#### 5.1.1. Pseudo Instructions

Pseudo instruction records consist of 1700 commands, pseudo commands, labels and statement numbers. Subroutine PHASEC processes all pseudo instruction records except storage reference instructions, conditional skip instructions, BSS pseudo instructions, constants and labels. Those items not processed by subroutine PHASEC are processed by calls to the following subroutines:

	<u>Subroutine</u>
Storage Reference	CL12
Conditional Skip	SKIP
BSS	BSS
Constant	CON
Label	LABIN or LABEL

The initial pseudo instruction records read in the input pass contain variables, constants and arrays to which the assembler must assign storage locations. Subroutine LABIN performs this function.

When the first executable command is encountered, excluding the jump at the top of a main program, pseudo instructions are sent to the Index Optimization Subroutine, IXOPT. If an instruction is indexed, the subroutine decides which index to use. If the instruction is not indexed, it is checked for its possible effect on subsequent indexing.

DOCUMENT CLASS IMS PAGE NO. 5-4  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

An output record is constructed for each pseudo instruction and stored in the output buffer. Instructions with absolute operands are completely processed in the input phase. The output record is flagged as ready for output. Instructions with relocatable operands are partially processed. Final assignment of the operand address is made after analysis in the output phase. Address modes are identified in the input phase. All instructions with two-word relative addressing mode are flagged as one-word candidates. Instructions with no address mode field and those with a mode other than two-word relative are flagged as fixed length but not ready for output.

Pseudo instruction records are placed in the output buffer by subroutine INOUT. When an END record is encountered control is passed from PHASEC to the output phase.

#### 5.1.2 Data Strings

Data strings are completely processed in subroutine DATAST. A separate output record, identified as an ORG record, is constructed for each datum in the same format as the pseudo instruction record.

#### 5.1.3 Format Records

Format records are completely processed in subroutine PHASEC. ORG records containing two ASCII characters are constructed in the same format as the pseudo instruction record.

#### 5.1.4 Record Formats

##### 5.1.4.1 Input Record Instruction Codes

##### 5.1.4.2 Breakdown of Input Record Indicator

##### 5.1.4.3 Input Pseudo Instruction Record Formats

##### 5.1.4.4 Output Record Formats

##### 5.1.4.5 Assembler Instruction Type Codes

##### 5.1.4.6 Breakdown of Output Record Indicator

##### 5.1.4.7 Phase C and D/E Common Blocks



DOCUMENT CLASS IMS PAGE NO. 5-5  
PRODUCT NAME 1700 Mass Storage FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

INPUT RECORD INSTRUCTION CODES

<u>Instruction</u>	<u>Code</u>
BSS	1
ADC	2
CON	3
END	4
PARAMETER STORE	5
STATEMENT NO.	6
JMP	10
RTJ	11
LDA	12
AND	13
STA	14
STQ	15
RAO	16
ADD	17
SUB	18
LDQ	19
DVI	20
ADQ	21
MUI	22
EOR	23
KLDQ (I register load)	28
KSTQ (I register store)	29
ENA	30
INA	31
ENQ	32
INQ	33
LRS	34
LLS	35
QRS	36
QLS	37
ARS	38
ALS	39

DOCUMENT CLASS IMS PAGE NO. 5-6  
 PRODUCT NAME 1700 Mass Storage FORTRAN  
 PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

<u>Instruction</u>	<u>Code</u>
Conditional Jump	
A Jump $\neq$ 0	40
A Jump = 0	41
A Jump $<$ 0	42
A Jump $\geq$ 0	43
A Jump $\leq$ 0	44
A Jump $>$ 0	45
Q Jump $\neq$ 0	46
Q Jump = 0	47
AAQ A	60
TCQ A	61
TRA Q	62
TCA A	63
LAQ Q	64

DOCUMENT CLASS IMS PAGE NO. 5-7  
 PRODUCT NAME 1700 Mass Storage FORTRAN  
 PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

BREAKDOWN OF INPUT RECORD INDICATOR

<u>Bit No.</u>	<u>Holder Name</u>	<u>Meaning</u>
15	NL	Not used in input phase.
14	NT	This record is a label
13	NA	This record has an additive
12	NS	This record has an index
11	ND	Use indirect addressing mode
10	NR	Use two-word constant Addressing Mode
9	NQ	Assign index to Q register
8	NF	Not used in input phase
7-2		
Instruction code ≠ 2	NTYPE	
	= 0	Addressing mode to be assigned.
	= 1	Absolute operand. Use one-word direct addressing mode.
	= 2	Absolute operand. Use two-word constant addressing mode.
	= 3	Not used
	= 4	Operand is relocatable Address constant
	= 5	Operand is absolute Address constant
Instruction code = 2	NTYPE	
	= 0 thru 5	Instruction is address constant type 0 through 5. See description of address. Constant types under Subroutine PHASE 5 (5.1.5.)
1-0	NOWI	Number of words in the record minus 2.

INPUT PSEUDO INSTRUCTION RECORD FORMATS  
(Assembly Input Phase)

INSTRUCTION	WORD 1	WORD 2	WORD 3	WORD 4	WORD 5
Holder Name		NOPC	NOPT	NADD	NXPT
LABEL	Indicator	Label Pointer			
BSS	Indicator	Instr Code	Size		
ADC	Indicator	Instr Code	Address Pointer	Additive	
CON	Indicator	Instr Code	Constant		
END	Indicator	Instr Code			
Parameter Store	Indicator	Instr Code	Operand Pointer		
Statement Number	Indicator	Instr Code	Statement No.		
Storage Reference Instr Type Code $\neq$ 1 or 2	Indicator	Instr Code	Operand Pointer	Additive	Index Pointer
Storage Reference Instr Type Code = 1 or 2	Indicator	Instr Code	Operand		
Register Reference and Shift Instructions	Indicator	Instr Code	Operand		
Inter-Register Instr	Indicator	Instr Code			

CONTROL DATA CORPORATION • DEVELOPMENT DIV • SOFTWARE DOCUMENT

DOCUMENT CLASS IMS  
 PRODUCT NAME 1700 Mass Storage FORTRAN  
 PRODUCT NO. CD05 VERSION 2.0 MACHINE SERIES 1700  
 PAGE NO. 5-8



DOCUMENT CLASS IMS PAGE NO 5-10  
 PRODUCT NAME 1700 Mass Storage FORTRAN  
 PRODUCT NO. C005 VERSION 2-0 MACHINE SERIES 1700

ASSEMBLER INSTRUCTION TYPE CODES

<u>Code</u>	<u>Addressing Mode</u>	<u>Instruction</u>
1	One-word direct	Storage Reference
2	One-word indirect	
3	One-word relative	
4	One-word relative, indirect	
5	Two-word constant, 2nd word is an absolute address constant	
6	Two-word constant, 2nd word is a relocatable address constant	
7	Two-word absolute	
8	Two-word absolute, indirect	
9	Two-word relative	
10	Two-word relative, indirect	
11		Register reference and shifts
12		Conditional Skips A register $\neq 0$
13		A register = 0
14		A register $< 0$
15		A register $\geq 0$
16		A register $\leq 0$
17		A register $> 0$
18		Q register = 0
19		Q register $\neq 0$
		Conditional Jumps
20		A register $\neq 0$
21		A register = 0
22		A register $< 0$
23		A register $\geq 0$
24		A register $\leq 0$

CONTROL DATA CORPORATION  
LA JOLLA RESOURCE CENTER DIVISION

DOCUMENT CLASS IMS PAGE NO. 5-11  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

25	A register 0
26	Q register = 0
27	Q register ≠ 0
30	AAQ A
31	TCQ A
32	TRA Q
33	TCA A
34	LAQ Q
40	BSS
41	ADC Type 1
42	ADC Type 2
43	ADC Type 3
44	ADC Type 4
45	ADC Type 5
46	Integer Constant
47	Real Constant
48	Double Precision Constant
50	Data ORG
51	Data Continuation ORG
52	Program ORG
53	Program Continuation ORG
59	Statement No.
60	END

DOCUMENT CLASS IMS PAGE NO 5-12  
 PRODUCT NAME 1700 Mass Storage FORTRAN  
 PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

Breakdown of Output Record Indicator

<u>Bit No.</u>	<u>Holder Name</u>	<u>Meaning</u>
15	NL	Skip this record
14	NT	This record is a label
13	NA	Operand address has an additive.
12	NS	Operand address is indexed.
11	ND	This record is ready for output.
10	NR	This record is fixed length
9	NQ	Flag for address constant sub-type.
8	NF	Flag for address constant sub-type.
7-2	NTYPE	Assembler instruction type code. See list.
1-0	NOWI	Number of words in the record minus 2.



DOCUMENT CLASS IMS PAGE NO. 5-13  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005 V.2.0 MACHINE SERIES 1700

## LIST OF HOLDERS, COUNTERS, FLAGS BUFFERS AND TABLES

## IN PHASE C and D/E COMMON BLOCKS

<u>NAME</u>	<u>USE</u>
ICFLAG	Change flag set in ADMAX
ICOUNT	Program counter
ICT	Program counter increment
IDECR	Program address decrement set in ADMAX
IDVTAB(312)	Table of address holders built in CHOP
IFIX	Fixed buffer indicator set in AMOUT
INBUFF(203)	Input buffer
INCT	Index for INBUFF and NOBUFF. In phases D/E, circular buffer pointer to NOBUFF (always increases)
ISCOUN	Holder for ICOUNT saved from input phase
ITCOUN	Holder for ICOUNT saved from output phase
IUSE	Flag indicating use of I register in object program
IV(5)	Temporary storage
IXC	Occurrence flag for index to be assigned set in CHKWD and IXOPT
IXF	Holder for contents of object time I register
IXFLAG	Decision flag set in CHKWD
IXQ	Holder for contents of object time Q register
IXS1(10)	Table of saved indexes built in IXOPT
IXS2(10)	Table of occurrence flags for saved indexes built in IXOPT

## CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 5-14  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005 V.2.0 MACHINE SERIES 1700

<u>NAME</u>	<u>USE</u>
IXS3(10)	Table of equivalence table indexes for saved indexes built in IXOPT.
IXSX	Index for IXS1, IXS2 and IXS3.
LRS	Flag indicating instruction is LRS 16.
NA	Additive flag extracted from record indicator.
NADD	Additive to instruction operand address.
NADX	Index for IDVTAB.
NASCI	Holder for alpha output word used in SETPRT.
NATYPE	Assembly instruction type code in input phase. Index to alpha opcode table in output phase.
NBFL	Flag for last word in binary output record.
NBINC	Punch buffer index.
ND	Indirect flag extracted from record indicator in input phase. Flag indicating instruction is ready for output in output phase.
NDFL	Temporary holder for "ready for output" flag used in CL12.
NDSTAB(500)	Table of dummy parameter address holders built in CHOP.
NDXJ	Temporary holder for ICOUNT.
NF	Flag in the record indicator used in PHASE 5 and AMOUT to indicate sub-type of address constant.
NFTAB(160)	Table of futures built in AMOUT.
NL	Indicator flag to ignore this record. Used in AMOUT.

## CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS. \_\_\_\_\_ IMS \_\_\_\_\_ PAGE NO. 5-15  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005 V.2.0 \_\_\_\_\_ MACHINE SERIES 1700

NOBW	Number of binary words currently in the punch buffer
NOCT	NOBUFF index used in input phase In output phase D/E, next available entry in circular buffer.
NOBUFF (210) C (1152) D/E	Output buffer passed from the input phase to the output phase
NODS	Index to NDSTAB
NOFS	Index to NFTAB
NOPC	Instruction code in input phase Partial or complete binary word in output phase
NOPT	Pointer to operand location in the symbol table
NOUTOT	Total number of words in NOBUFF In Phases D/E, last entry in circular buffer
NOWI	Number of words in the input record extracted from the record indicator
NOWO	Number of words in the output record. Used in the input phase
NPCBF (57)	Punch buffer
NPTBF (55)	Print buffer
NPTBFX	Print buffer index
NQ	Flag extracted from the record indicator indicating the index for this instruction has been assigned to the Q register. Used in the output phase as a flag for an ADC sub-type.
NR	Two word instruction flag extracted from the record indicator in the input phase. Fixed length flag in the output phase.

## CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 5-16  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005 V.2.0 MACHINE SERIES 1700

<u>NAME</u>	<u>USE</u>
NRC(5)	Buffer for output record. Used in the input phase.
NRC1	Address relocation holder
NRC2	Address relocation holder for second word of a two-word instruction
NRM	Program relocation holder
NS	Flag extracted from the record indicator indicating the instruction address is indexed
NSETX	Print buffer index
NSUMD	Total number of storages in IDVTAB
NSUMDS	Total number of storages in NDSTAB
NSTYPE	Flag set when first executable instruction of the object program is encountered
NT	Flag extracted from the record indicator indicating this instruction is a label
NTYPE	Holder for address type extracted from the record indicator in the input phase. Used for assembler instruction type in the output phase.
NSUMVS	Total number of storages in NFTAB
NW2	Holder for second binary word of a two-word instruction
NW2FL	Flag indicating a two-word instruction
NX	Index for INBUFF and NOBUFF used in BKDWN

## CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 5-17  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005 V.2.0 MACHINE SERIES 1700

<u>NAME</u>	<u>USE</u>
NXCON	Holder for hex or decimal output word used in SETPRT
NXPT	Pointer to index address in the symbol table.
IHEAD	Pointer to 'beginning' of circular buffer (always increases).
INOB	Size of NOBUFF, used to reference circular buffer in phases D/E
INC	Actual subscript in NOBUFF, calculated from INCT modulo INOB, by subroutine INDEX

DOCUMENT CLASS IMS PAGE NO 5-18  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

### 5.1.5 Subroutine PHASEC

Subroutine PHASEC is the first subroutine entered in the FORTRAN Assembler. Symbol table location holders, other than common, are backgrounded to 7FFF<sub>16</sub>. The first input record is read here. Subsequent records are usually read in subroutine INOUT. They may be read in other routines if a forward scan is initiated at the bottom of the current input buffer.

If the input record is a data string or a format record the subroutine branches immediately to process the record. If the input record is a pseudo instruction record, subroutine BKDWN is called to determine the type of pseudo instruction.

Following are descriptions of the types of pseudo instructions encountered and their processing.

**LABEL** - Subroutine CHKWD is called to destroy any indexes saved in subroutine IXOPT since they will not be valid after a label is encountered. If this is a statement label, subroutine LABEL is called. If this is a variable or constant label and the first executable instruction has not been encountered, subroutine LABIN is called. If the first executable instruction has been encountered, subroutine LABEL is called.

**BSS** - Call subroutine BSS

**ADC** - If the address constant is an external and if this is the first encounter, the program counter is put in the symbol table. If this is not the first encounter, the output record is set up without recording the program counter.

If the address constant is not an external, it may be any one of five types.

**Type 1** - An entry in a table of addresses set up for a computed GO TO. Address is relative to referenced location computed in 16 bit arithmetic.

**Type 2** - A parameter in the calling sequence to a floating point subroutine when using relative addressing. By examining the run anywhere switch and the setting of the dummy argument identification bit in the symbol table, type 2 is broken down into four sub-types.

Parameter is not a dummy argument

Address is relative to referenced storage computed in 15 bit arithmetic with bit 15 set to zero.

DOCUMENT CLASS IMS PAGE NO 5-19  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

Parameter is a dummy argument

Run anywhere program - address is relative to referenced storage computed in 15 bit arithmetic with bit 15 set to one.

Non-run anywhere program - If the parameter has been referenced, this address is the location of the previous reference with bit 15 set. If the parameter has not been referenced, this location will be stuffed with the computed address of the dummy argument.

Type 3 - Parameter in a calling sequence to a subroutine other than floating point. Address is relative to referenced location computed in 15 bit arithmetic with bit 15 set to one.

Types 4 & 5 are parameters in a calling sequence to a subroutine other than floating point or to a floating subroutine when absolute addressing is used.

Type 4 is the relocatable address of the referenced location. Type 5 is the absolute address of the referenced location.

The actual address is not computed in subroutine PHASEC bits NQ and NF in the indicator are used to flag the setting of bit 15 and the type of address to be used.

CONSTANT - Call subroutine CON

PARAMETER STORE - This is a STA command with a special type code that tells the assembler the A register contains a computed address to be stored in a dummy parameter address holder. The first holder is always in the symbol table entry for this dummy parameter. The dummy parameter table is searched for any other holder that may have been created to maximize the use of one-word relative, indirect addressing. A store Address record is built for each holder. The first Store Address record contains a pointer to the entry in the symbol table. Subsequent records contain a pointer to the entry in the dummy parameter table.

## CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 5-20  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. COO5 V.2.0 MACHINE SERIES 1700

STATEMENT NO. - These are numbers generated consecutively by the FORTRAN Compiler for each FORTRAN source record. The output record applies to list output only.

STORAGE REFERENCE COMMAND - Call subroutine CL12. On return from subroutine CL12, if the storage reference command is an unconditional jump, the symbol table is searched for variable and constant holders and dummy parameter address holders that are ready for output. Variable and constant holders are ready for output when they have been used as the operands in a store type command. Dummy parameter address holders are ready for output after the first reference. Holders are identified as ready for output in Subroutine CHOP. LABEL and BSS records are constructed for each holder to be output. The current program location is entered in the symbol table.

The dummy parameter table is also searched for any additional dummy parameter address holders to be output. A BSS record is constructed for each holder to be output. The current program location is entered in the dummy parameter table.

REGISTER, SHIFT and INTER-REGISTER COMMANDS - Since these three types of commands have absolute operands, the binary output word is completely constructed. The output record is flagged "ready for output".

If the command is an LRS 16 the assembler examines the next record to see if it is an indexed DVI. If it is, and if the index in the DVI record is not in the I register holder, subroutine QXLD is called to construct the necessary index load and store commands to be inserted in the output buffer prior to the Long Right Shift Command. The index in the DVI command is placed in the I register holder.

SKIP - Call Subroutine Skip



## CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS \_\_\_\_\_ IMS \_\_\_\_\_ PAGE NO. \_\_\_\_\_ 5-21  
PRODUCT NAME \_\_\_\_\_ 1700 MASS STORAGE FORTRAN \_\_\_\_\_  
PRODUCT MODEL NO. \_\_\_\_\_ C005 V.2.0 \_\_\_\_\_ MACHINE SERIES \_\_\_\_\_ 1700

END -

The symbol table and the dummy parameter table are searched for holders to be identified as described under Storage Reference Commands. A dummy ENA record is output to avoid a possible BSS as the last command in the program.

In the 16K version of 1700 Fortran, the current symbol table page is written and the page size is changed from 2400 to 960. Control passes to the assembler output phases D/E.

DOCUMENT CLASS IMS PAGE NO 5-22  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT NO CO05 VERSION 2.0 MACHINE SERIES 1700

Subroutine BLDUP and INOUT are called to complete processing of the output record. Subroutine COUNT is called if the instruction to be output will occupy a storage location in the binary object program.

Records placed in the output buffer from subroutine PHASEC are flagged as "fixed length" or "ready for output". Bits NR and ND in the indication are used for this purpose. A fixed length instruction that is not ready for output is one with a forward storage reference that cannot be defined until after analysis in the output phase.

5.1.5.1 Flow Chart of Subroutine PHASEC

5.1.6 Subroutine BKDWN

The function of subroutine BKDWN is to extract and store the information in the pseudo instruction record. In the Assembly Input Phase, these are the records constructed in the previous FORTRAN Compiler pass. In the Assembly Output Phase, these are the records constructed in the Assembly Input phase.

See 5.1.4 for a detailed list of the input record breakdown.

5.1.6.1 Flow Chart of Subroutine BKDWN

5.1.7 Subroutine BLDUP

Subroutine BLDUP sets up the records that are output from the assembler input phase. See 5.1.4 for output record format.

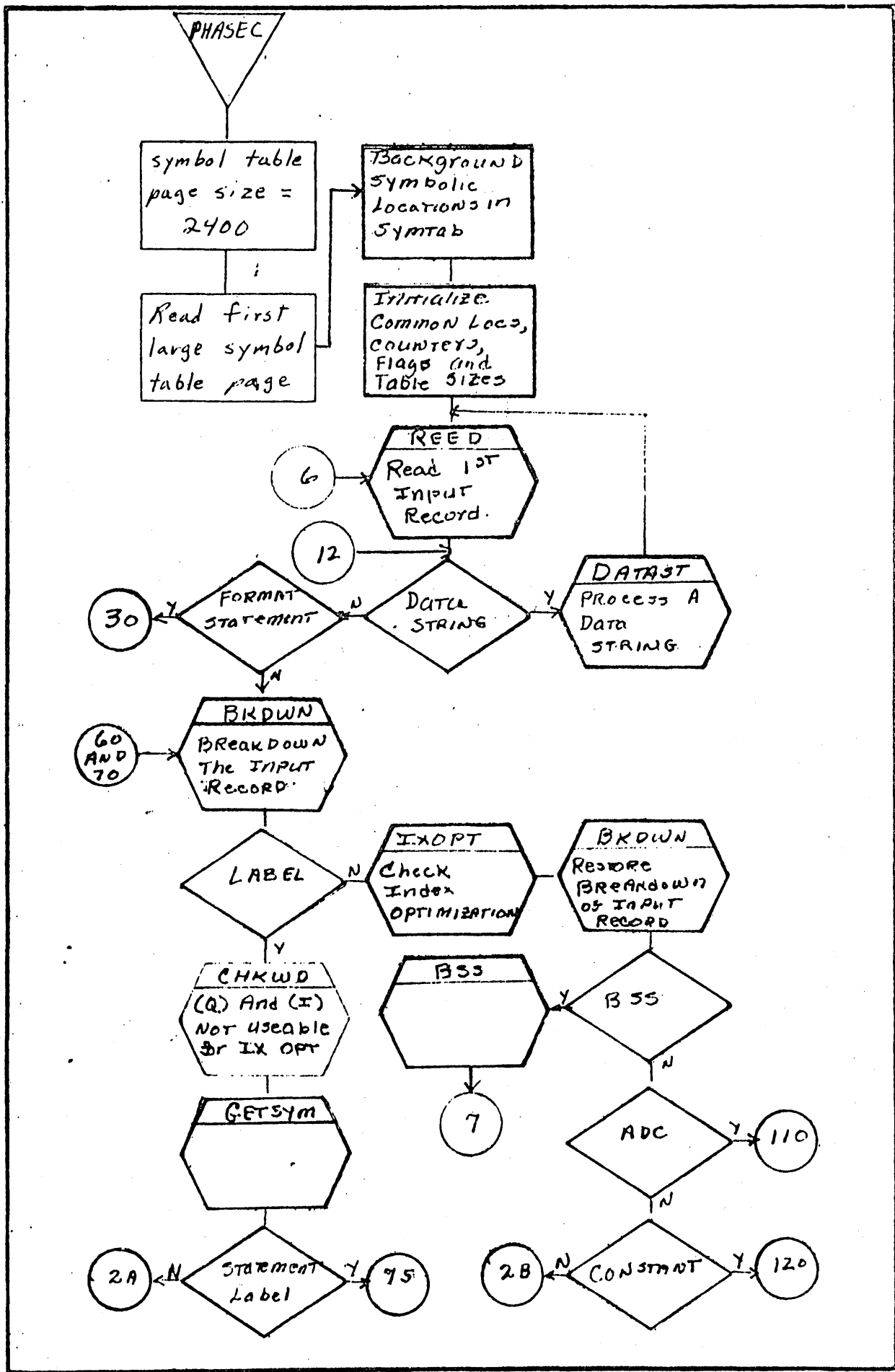
5.1.7.1 Flow Chart of Subroutine BLDUP

5.1.8 Subroutine BSS

Subroutine BSS constructs a BSS output record. The program's counter is incremented by the size of the BSS. The record is flagged ready for output.

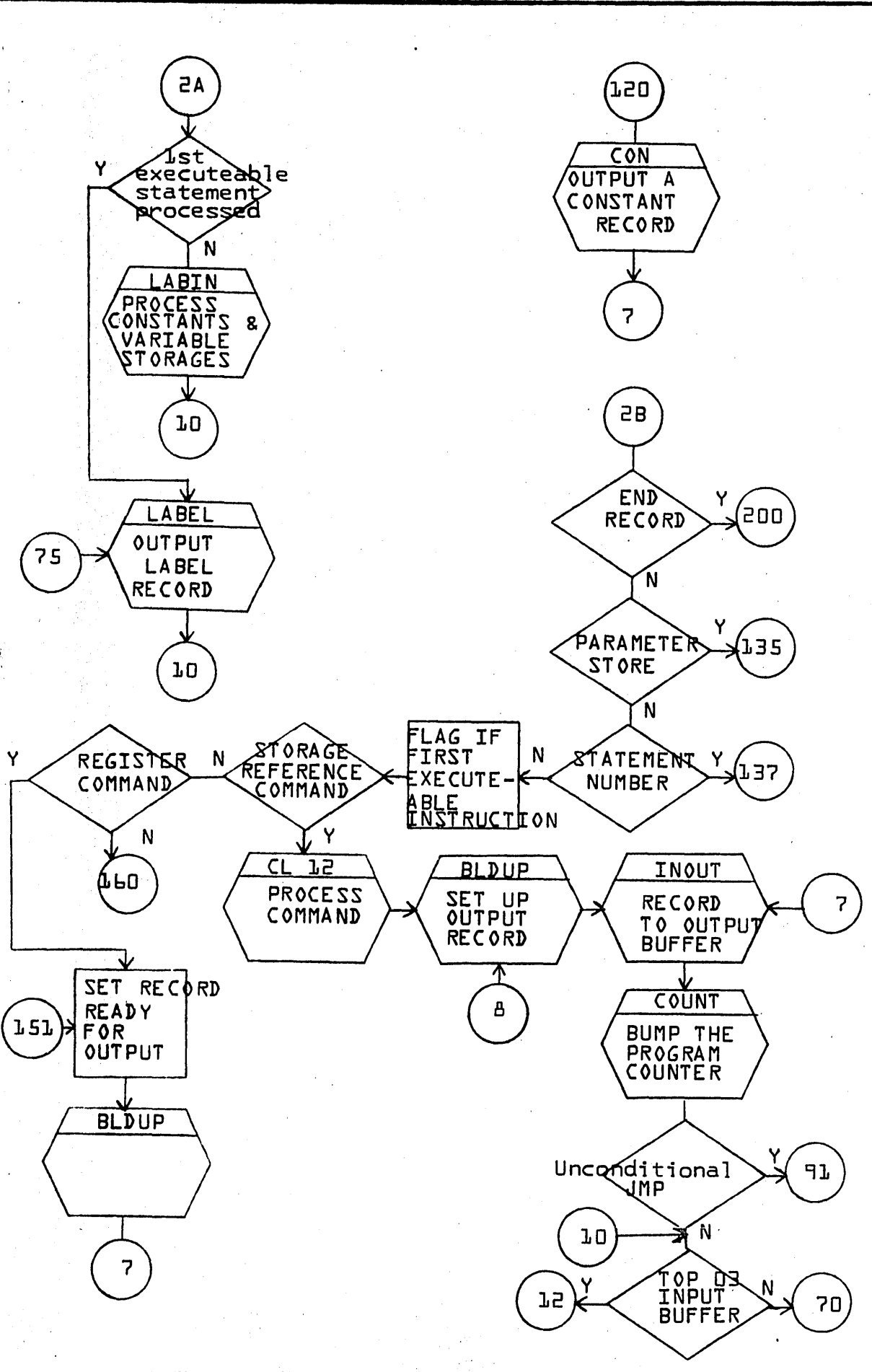
A BSS size of minus one in the input record identifies the I register holder which may be eliminated in the assembler output phase. In this case, the output record is not flagged ready for output.

5.1.8.1 Flow Chart of Subroutine BSS

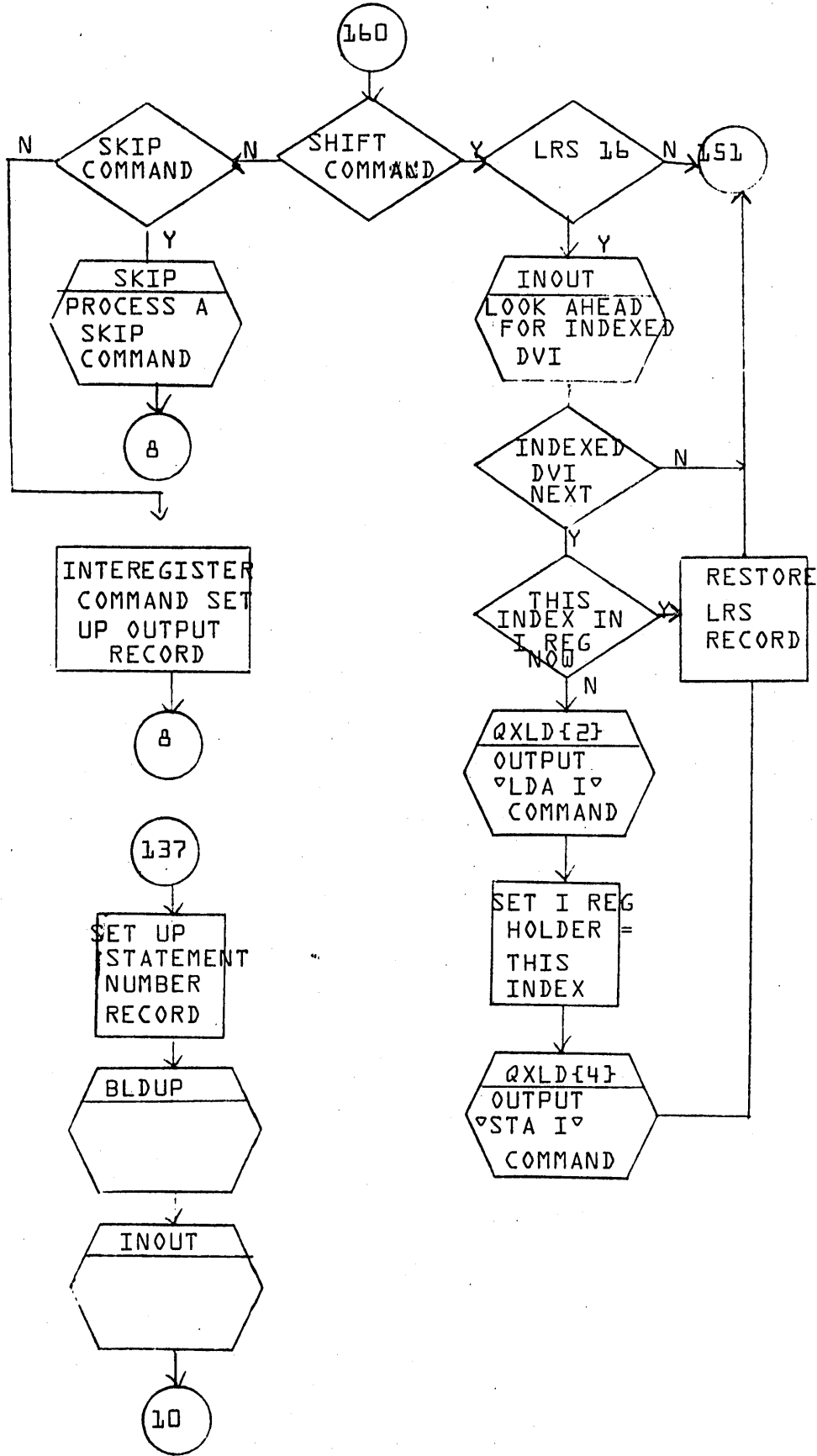


CONTROL DATA CORPORATION	PROJECT NO.	DATE
SOFTWARE DOCUMENT	APPROVED	
SAMPLE CODE	REV	
FLOWCHART		
DECISION TABLE		
DOCUMENT CLASS <b>FMS</b>	MACH. TYPE 1700	PROJECT MGR
DOCUMENT TITLE <b>PHASEC</b>	PAGE 1 OF 8	PROJECT NAME
NUMBER 5.1.5.1	ISSUE DATE	TASK NO.

2



DATE	
APPROVED	
REV	
PROJECT NO.	
PROJECT MGR.	
PROJECT NAME	
TASK NO.	
TASK NAME	
MACH. TYPE	1700
DOCUMENT CLASS	IMS
DOCUMENT TITLE	PHASE C
PAGE	2 of 8
NUMBER	5.1.5.1
ISSUE DATE	
DRAWN BY	
DATE	
<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT SAMPLE CODE <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> FLOWCHART DECISION TABLE OTHER	

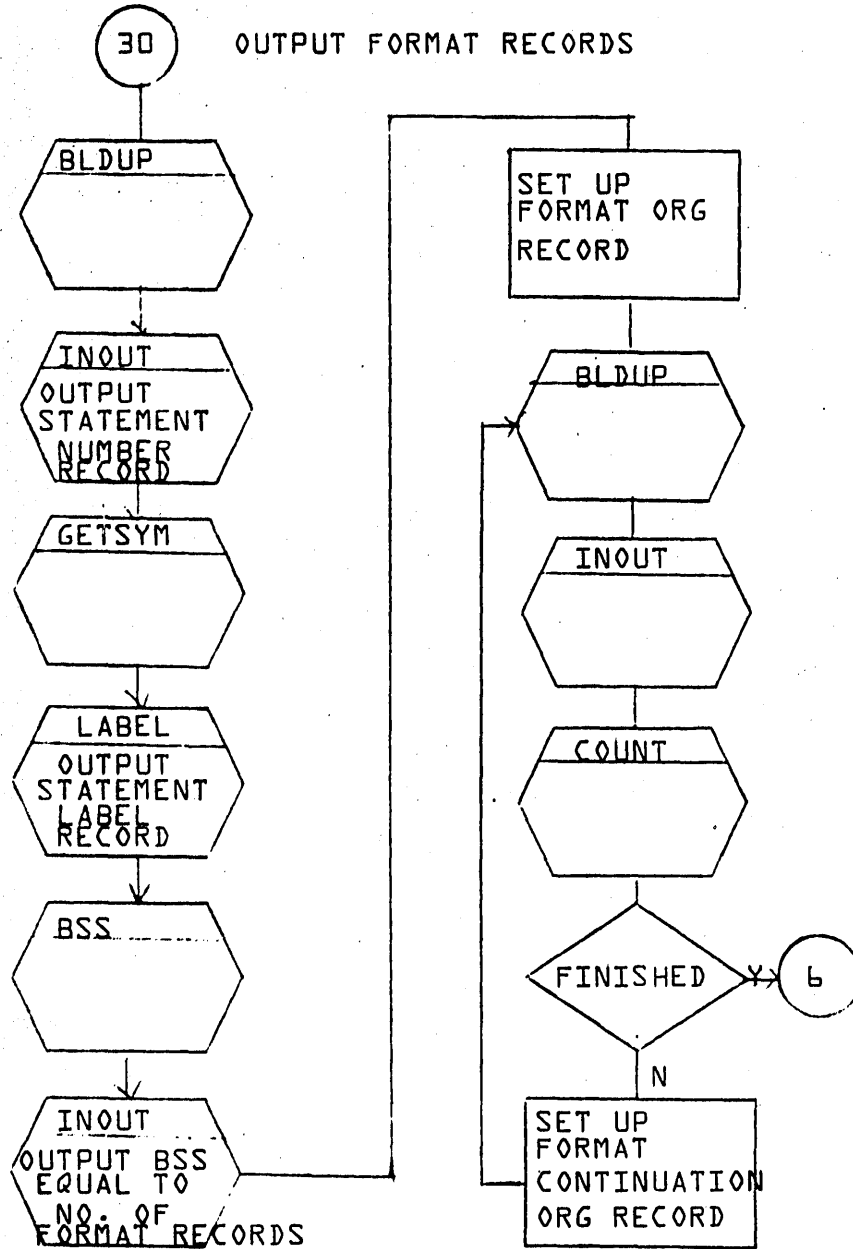


CONTROL DATA CORPORATION SOFTWARE DOCUMENT		MACH. TYPE 1700		PROJECT NO.	APPROVED	DATE
DOCUMENT TITLE Phase C		PAGE 3 of 8		PROJECT MGR.		
NUMBER		ISSUE DATE		PROJECT NAME		
DRAWN BY		DATE		TASK NO.		
				TASK NAME		

SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

1 2 3 4 5

A B C D



	REV	APPROVED	DATE
DOCUMENT CLASS	PROJECT NO.	PROJECT MGR.	PROJECT NAME
DOCUMENT TITLE	MACH. TYPE	ISSUE DATE	TASK NO.
IMS	1700	PAGE 4 OF 4	TASK NAME
PHASE C		DATE	
NUMBER			
DRAWN BY			

**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

SAMPLE CODE

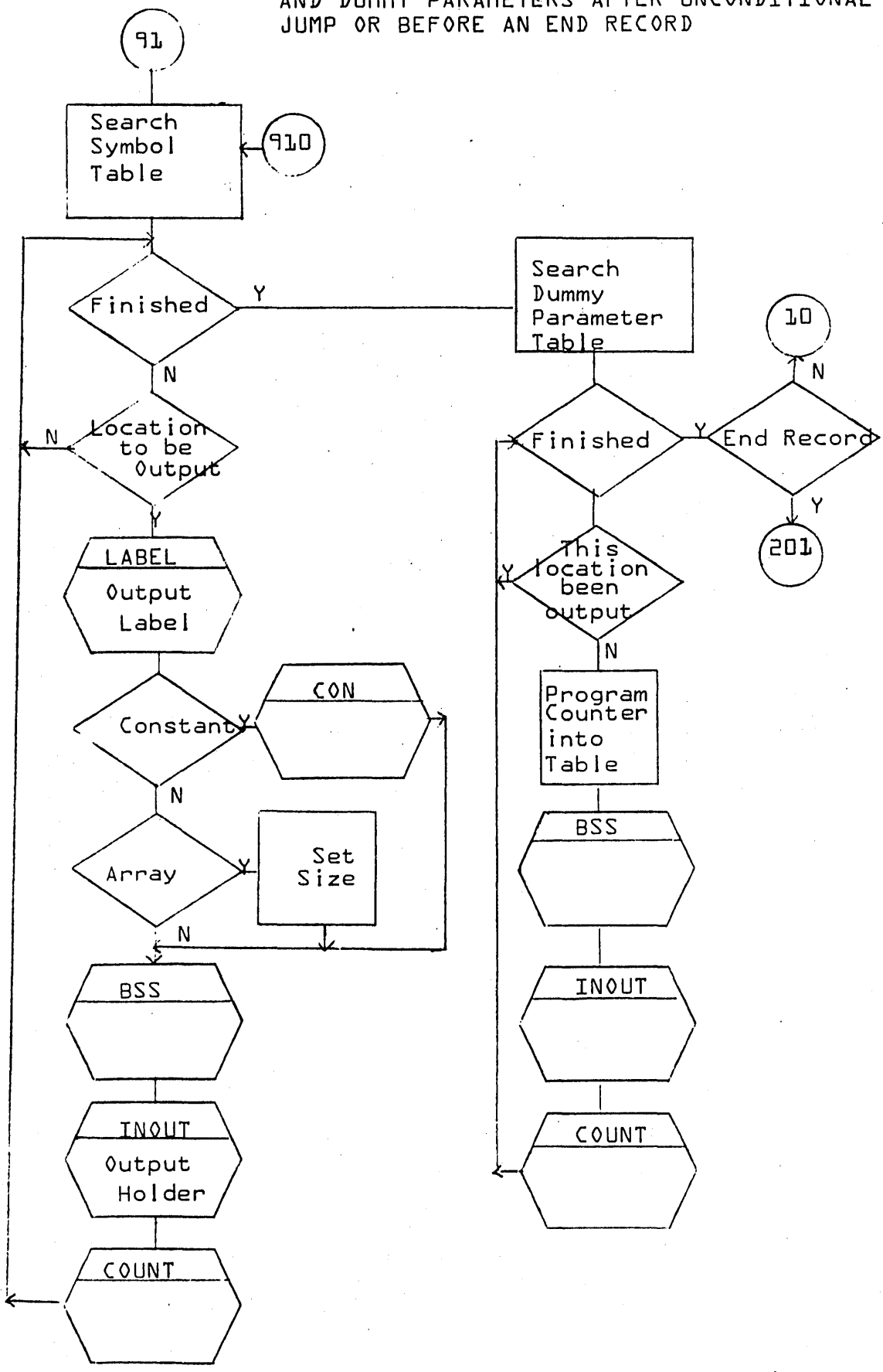
FLOWCHART

DECISION TABLE

OTHER

PRINTED IN USA

OUTPUT HOLDERS FOR VARIABLES, CONSTANTS AND DUMMY PARAMETERS AFTER UNCONDITIONAL JUMP OR BEFORE AN END RECORD



5  
4  
3  
2  
1

REV	APPROVED	DATE
PROJECT NO.	MACH. TYPE	1700
PROJECT MGR.	DOCUMENT CLASS	IMS
PROJECT NAME	DOCUMENT TITLE	Phase C
TASK NO.	NUMBER	5-1.5.1
TASK NAME	DRAWN BY	
	ISSUE DATE	
	DATE	
	PAGE 5 OF 8	

CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE

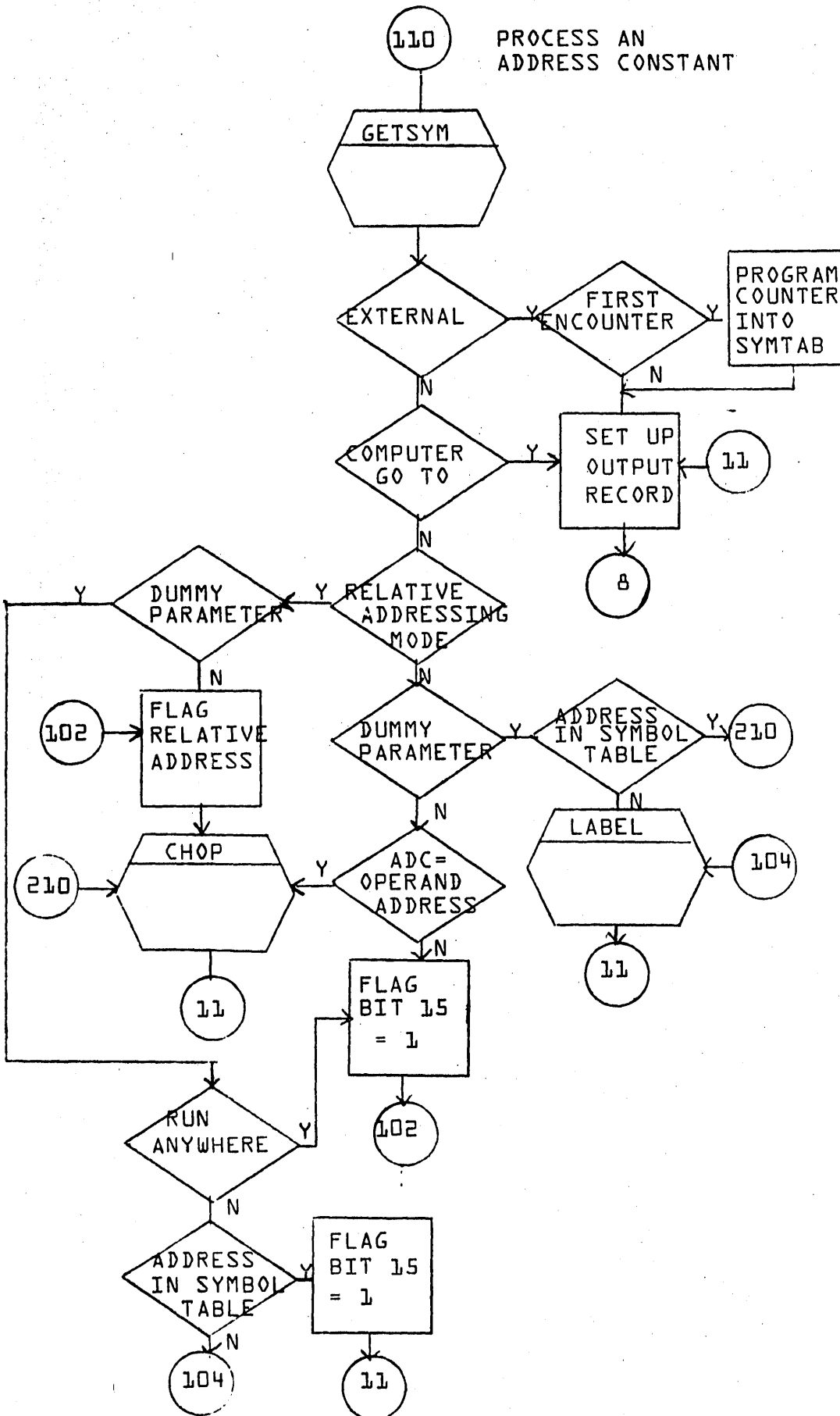
FLOWCHART

DECISION TABLE

OTHER

A B C D

PROCESS AN ADDRESS CONSTANT



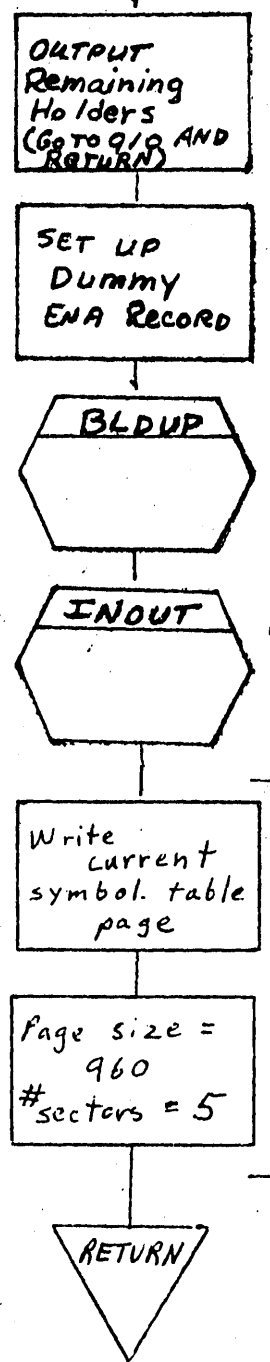
DATE	
APPROVED	
REV	
PROJECT NO.	
PROJECT MGR.	
PROJECT NAME	
TASK NO.	
TASK NAME	
MACH. TYPE	1700
DOCUMENT CLASS	IMS
DOCUMENT TITLE	PHASE C
PAGE	6
OF	8
ISSUE DATE	
NUMBER	
DRAWN BY	
DATE	
CONTROL DATA CORPORATION	
SOFTWARE DOCUMENT	
SAMPLE CODE	<input type="checkbox"/>
FLOWCHART	<input type="checkbox"/>
DECISION TABLE	<input type="checkbox"/>
OTHER	<input type="checkbox"/>

PRINTED IN USA





200 Process An End Record



2.0A version only

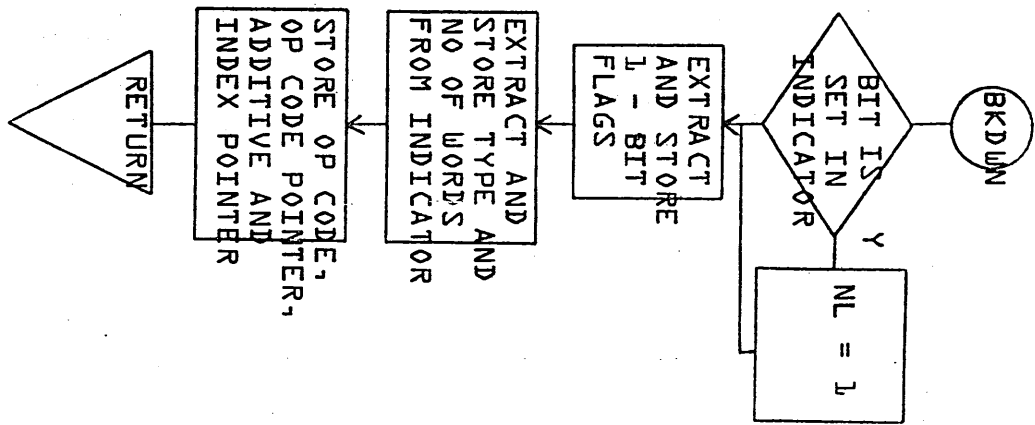
CONTROL DATA CORPORATION SOFTWARE DOCUMENT		DATE
SAMPLE CODE	PROJECT NO.	APPROVED
FLOWCHART	PROJECT MGR	REV
DECISION TABLE	PROJECT NAME	
OTHER	TASK NO.	
	TASK NAME	
DOCUMENT CLASS <b>IMS</b>	MACH TYPE <b>1700</b>	
DOCUMENT TITLE <b>PHASE C</b>	PAGE <b>8</b> OF <b>8</b>	
NUMBER <b>5.1.5.1</b>	ISSUE DATE	
DRAWN BY <b>N TALBOT</b>	DATE <b>10/91/66</b>	

A

B

C

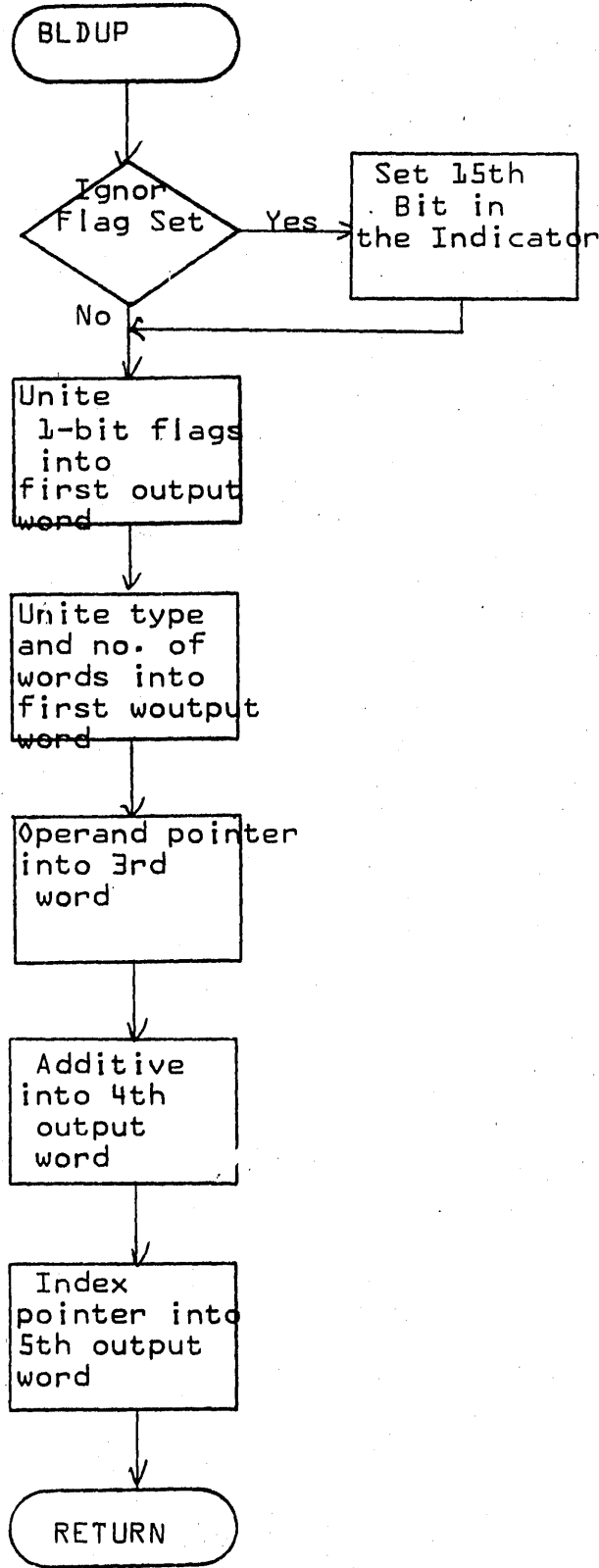
D



**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	Subroutine Bkdwn			PROJECT MGR.			
PAGE 1 OF 1				PROJECT NAME			
NUMBER	5.1.6.1	ISSUE DATE		TASK NO.			
DRAWN BY		DATE		TASK NAME			



CONTROL DATA CORPORATION SOFTWARE DOCUMENT		DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	
SAMPLE CODE <input type="checkbox"/>		DOCUMENT TITLE	Subroutine BLDUP		PROJECT MGR.		
FLOWCHART <input type="checkbox"/>		NUMBER		PAGE 1 of 1	PROJECT NAME		
DECISION TABLE <input type="checkbox"/>		ISSUE DATE			TASK NO.		
OTHER <input type="checkbox"/>		DRAWN BY			TASK NAME		
REV	APPROVED	DATE					

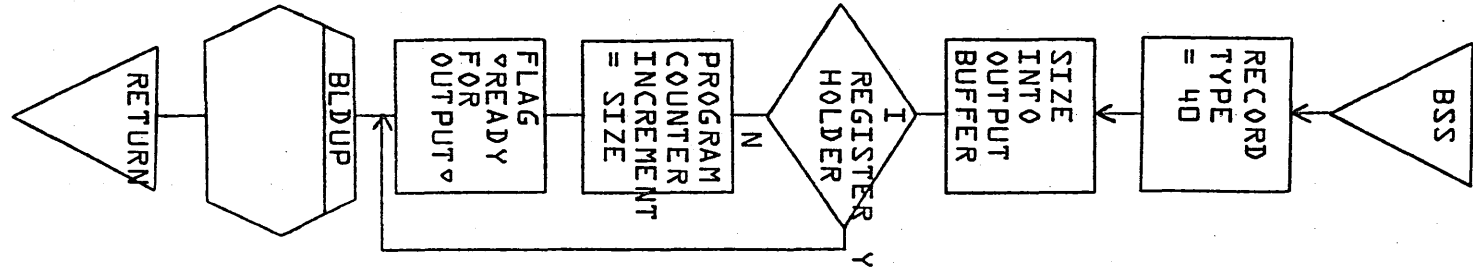
PRINTED IN U.S.A.

A

B

C

D



**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	Subroutine BSS			PROJECT MGR.			
PAGE 1 OF 1				PROJECT NAME			
NUMBER	5.1.8.1	ISSUE DATE		TASK NO.			
DRAWN BY		DATE		TASK NAME			

DOCUMENT CLASS IMS PAGE NO. 5-34  
PRODUCT NAME Mass Storage FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

### 5.1.9 Subroutine CHKWD

Subroutine CHKWD is used in the input phase of the FORTRAN Assembler. It checks all pseudo instructions to determine what effect, if any, they will have on current or future index register assignments. CHKWD is called from subroutine PHASE 5 whenever a label record is encountered and from subroutine IXOPT in both its primary and secondary scans of the input buffer.

The following pseudo instruction records may affect index register assignment:

Label - All index holders are cleared

Return Jump - Any index holders containing addresses in the COMMON area are cleared. If an address currently to be assigned to an index register is in the COMMON area, the decision flag is set.

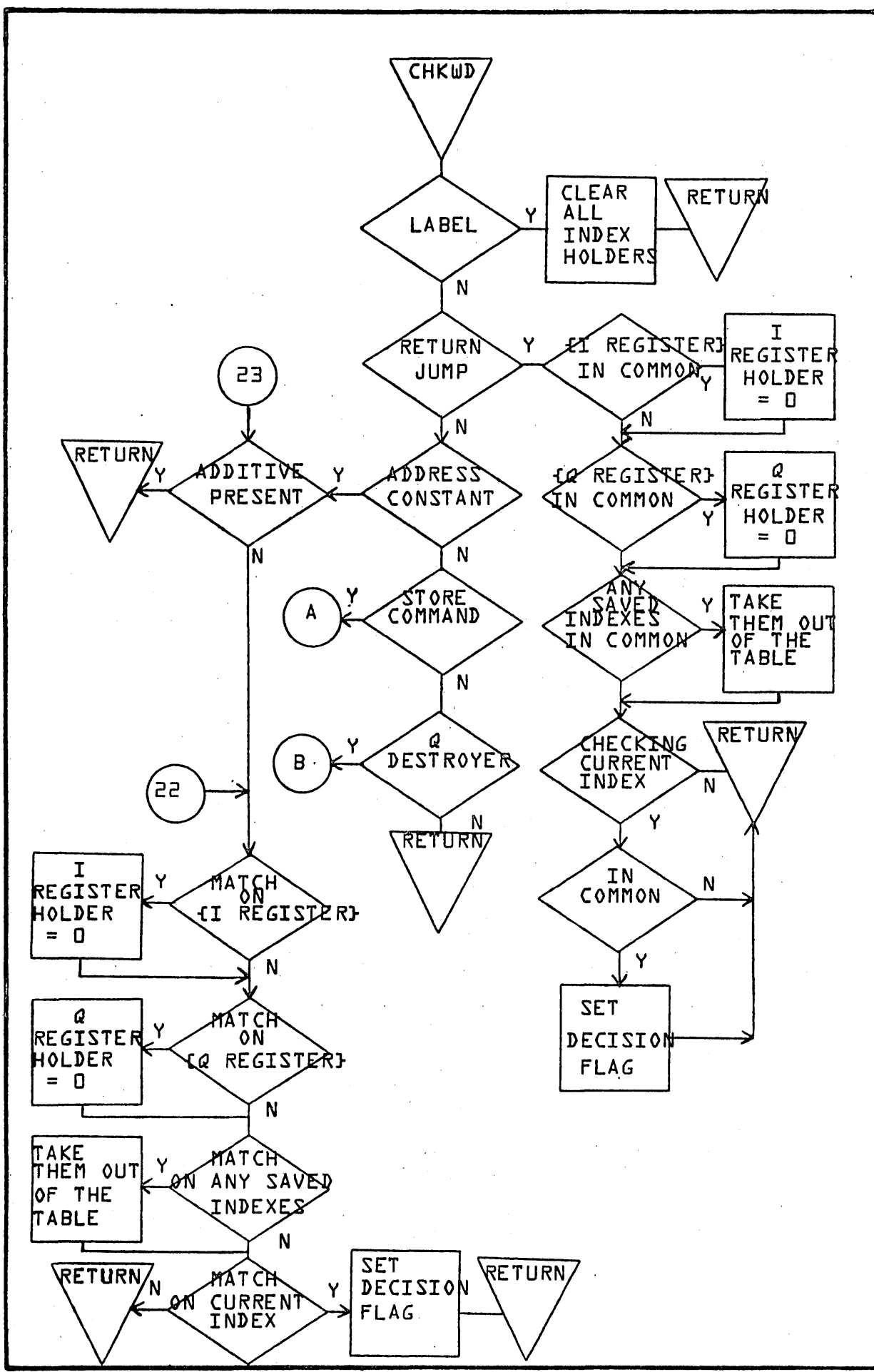
Address Constant - Any index holders containing this address constant are cleared. If an address currently to be assigned to an index register is this address constant, the decision flag is set.

Store Instruction - Any index holders containing the address of the operand of the store instruction are cleared. If the operand of the store instruction is in an equivalence chain, any index holders containing addresses in the same equivalence chain are cleared. If an address currently to be assigned to an index register is the same as the operand in the store instruction or is in the same equivalence chain, the decision flag is set.

Q Destroying Instruction - The Q register index holder is cleared and the occurrence flag is set.

The occurrence flag is set when an instruction affects index register assignment but is not decisive. The decision flag is set when an instruction is encountered that decides the index register assignment. These flags have meaning only if there is an address currently to be assigned to an index register.

#### 5.1.9.1 Flow Chart of Subroutine CHKWD



DATE	APPROVED	REV
PROJECT NO.	PROJECT MGR.	PROJECT NAME
TASK NO.	TASK NAME	
DOCUMENT CLASS	MACH. TYPE	ISSUE DATE
IMS	J700	
DOCUMENT TITLE	Subroutine Chkwd	PAGE 1 OF 2
NUMBER	5.1.9.1	
DRAWN BY		

CONTROL DATA CORPORATION  
 SOFTWARE DOCUMENT

SAMPLE CODE

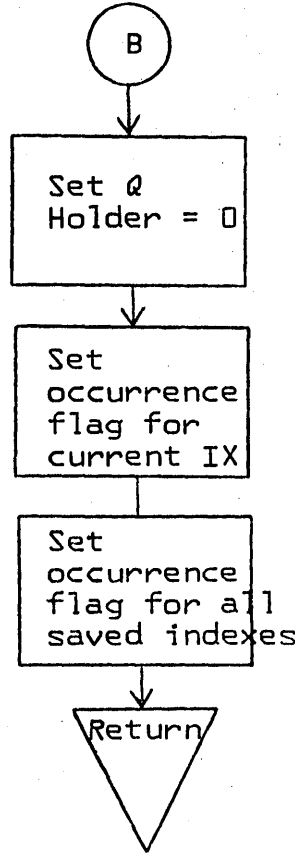
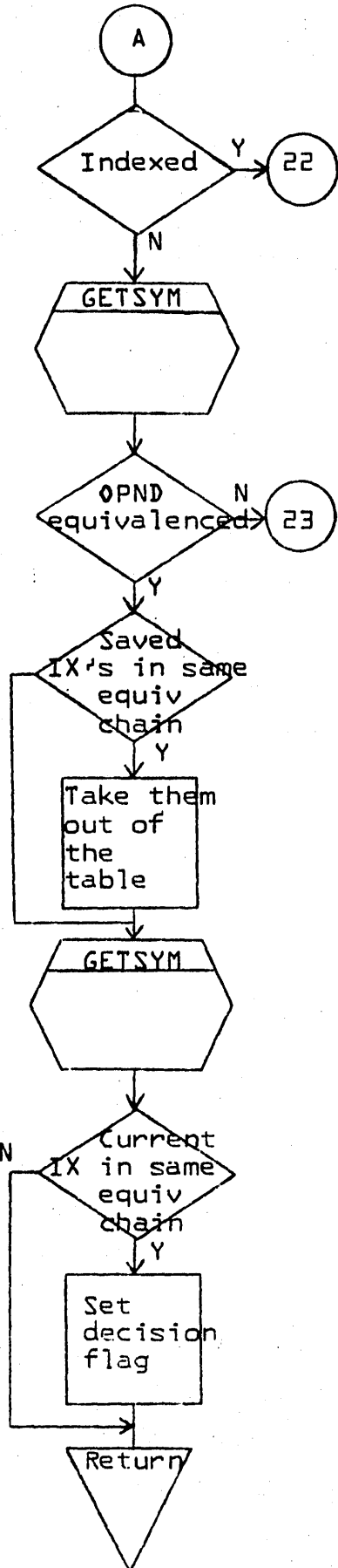
FLOWCHART

DECISION TABLE

OTHER

A B C D

5  
4  
3  
2  
1



CONTROL DATA CORPORATION		DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.		DATE	
SOFTWARE DOCUMENT		DOCUMENT TITLE	Subroutine Chkwd			PROJECT MGR.			
SAMPLE CODE		NUMBER	5.1.9.1	PAGE	2 of 2	PROJECT NAME			
FLOWCHART		ISSUE DATE		DATE		TASK NO.			
DECISION TABLE		DRAWN BY		DATE		TASK NAME			
OTHER									

A

B

C

D



DOCUMENT CLASS IMS PAGE NO. 5-37  
PRODUCT NAM 1700 Mass Storage FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

#### 5.1.10 Subroutine CHOP

The function of subroutine CHOP is to flag all defined operands for which no holder has been generated, to build tables of address holders, and to determine the addressing mode of all storage reference commands for which no address mode has been set in the previous pass.

An operand to be located within the body of the object program is flagged as defined if it is used in a store instruction or if it is an address constant. Holders for defined operands are generated following the next encountered unconditional jump instruction or at the end of the program.

An operand whose address is not within the body of the object program that is, COMMON locations, externals, functions and dummy parameters, is flagged as defined on first encounter. Holders for the operand addresses are generated following the next encountered unconditional jump instruction or at the end of the program.

Subroutine CHOP builds two tables of address holders, IDVTAB and NDSTAB.

Entries are made in IDVTAB for instructions referencing operands within the body of the object program when neither the operand holder nor any previous table entry for this operand can be reached with one-word relative direct or indirect addressing mode. The address of the referencing instruction and the address of the operand holder are entered in the table. Addresses of instructions referencing operands located in COMMON and a pointer to the operand address are also entered in IDVTAB on first encounter of the operand reference if the use count is greater than one. Subsequent references to the same operand are entered in IDVTAB if no previous reference can be reached with one-word, relative, indirect Addressing Mode.

Entries are made in NDSTAB for instructions referencing operands outside the body of the object program (excepting COMMON locations) when neither the operand address holder defined in the symbol table nor any previous table entry for this operand holder can be reached with one-word, relative, indirect addressing mode. Entries consist of the address of the referencing instruction and a pointer to the operand address.

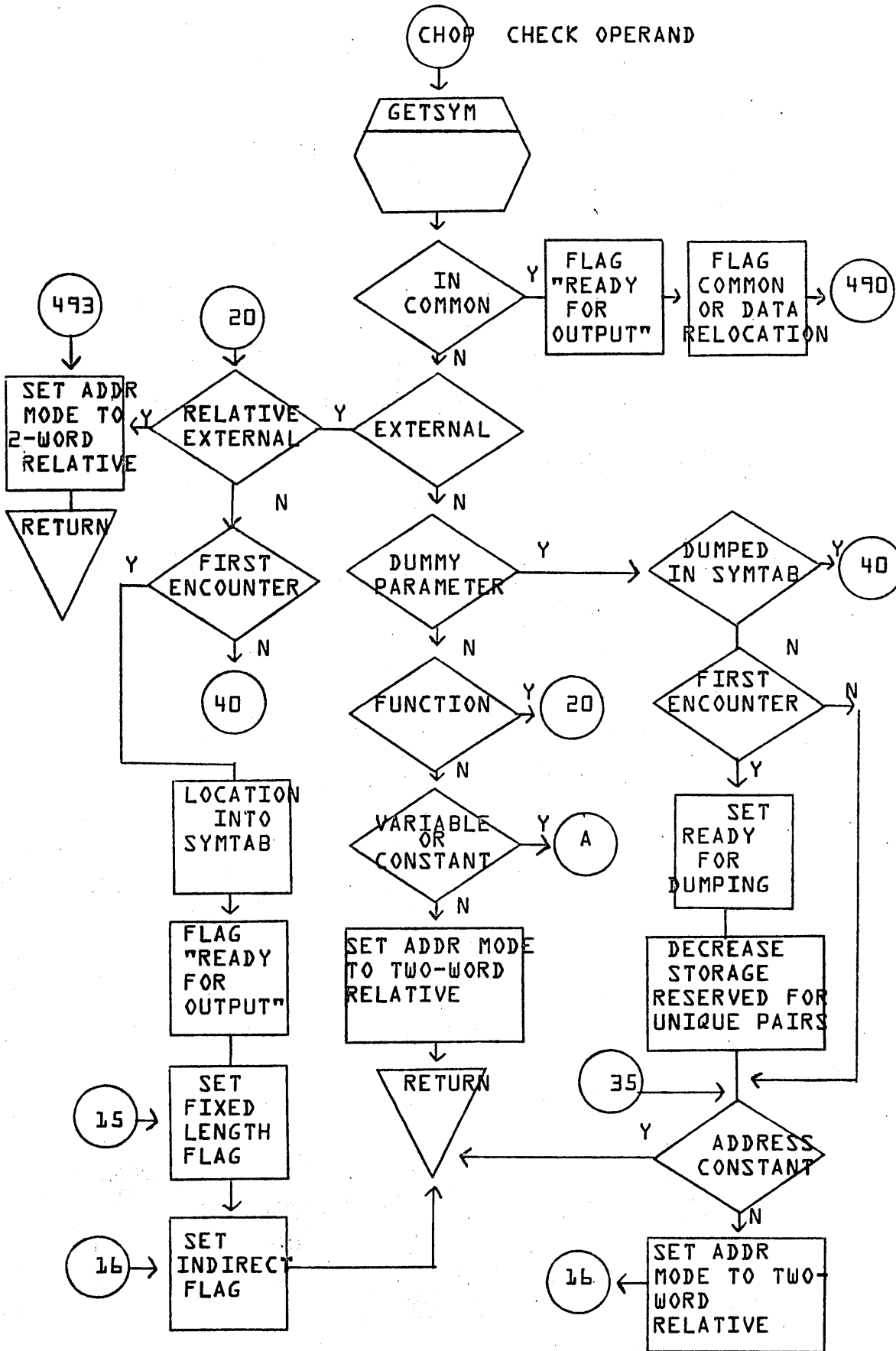
DOCUMENT CLASS IMS PAGE NO. 5-38  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. CD05 V2.0 MACHINE SERIES 1700

Addressing modes for instructions referencing operands located within the body of the object program are made one-word, relative, direct if the operand holder can be reached or one-word, relative, indirect if a table entry for this operand can be reached. If this instruction address is entered in IDVTAB, the addressing mode is two-word absolute. If the operand has not been defined, the addressing mode is two-word relative, direct.

Addressing modes for instructions referencing operand addresses outside the body of the object program are made one-word, relative, indirect if a previously defined address holder can be reached. If this instruction is entered in NDSTAB, the addressing mode is two-word, relative, indirect. In the case of an operand located in COMMON, the addressing mode is two-word absolute if the instruction is entered in IDVTAB.

5.1.10.1 Flow Chart of Subroutine CHOP.

5.1.10.2 Formats of IDVTAB and NDSTAB



DATE	
APPROVED	
REV	
PROJECT NO.	
PROJECT MGR.	
PROJECT NAME	
TASK NO.	
TASK NAME	
DOCUMENT CLASS	IMS
DOCUMENT TITLE	SUBROUTINE CHOP
MACH. TYPE	1700
PAGE	1
OF	3
NUMBER	5.1.10.1
ISSUE DATE	
DRAWN BY	
DATE	
CONTROL DATA CORPORATION	
SOFTWARE DOCUMENT	
SAMPLE CODE	<input type="checkbox"/>
FLOWCHART	<input type="checkbox"/>
DECISION TABLE	<input type="checkbox"/>
OTHER	<input type="checkbox"/>

5

4

3

2

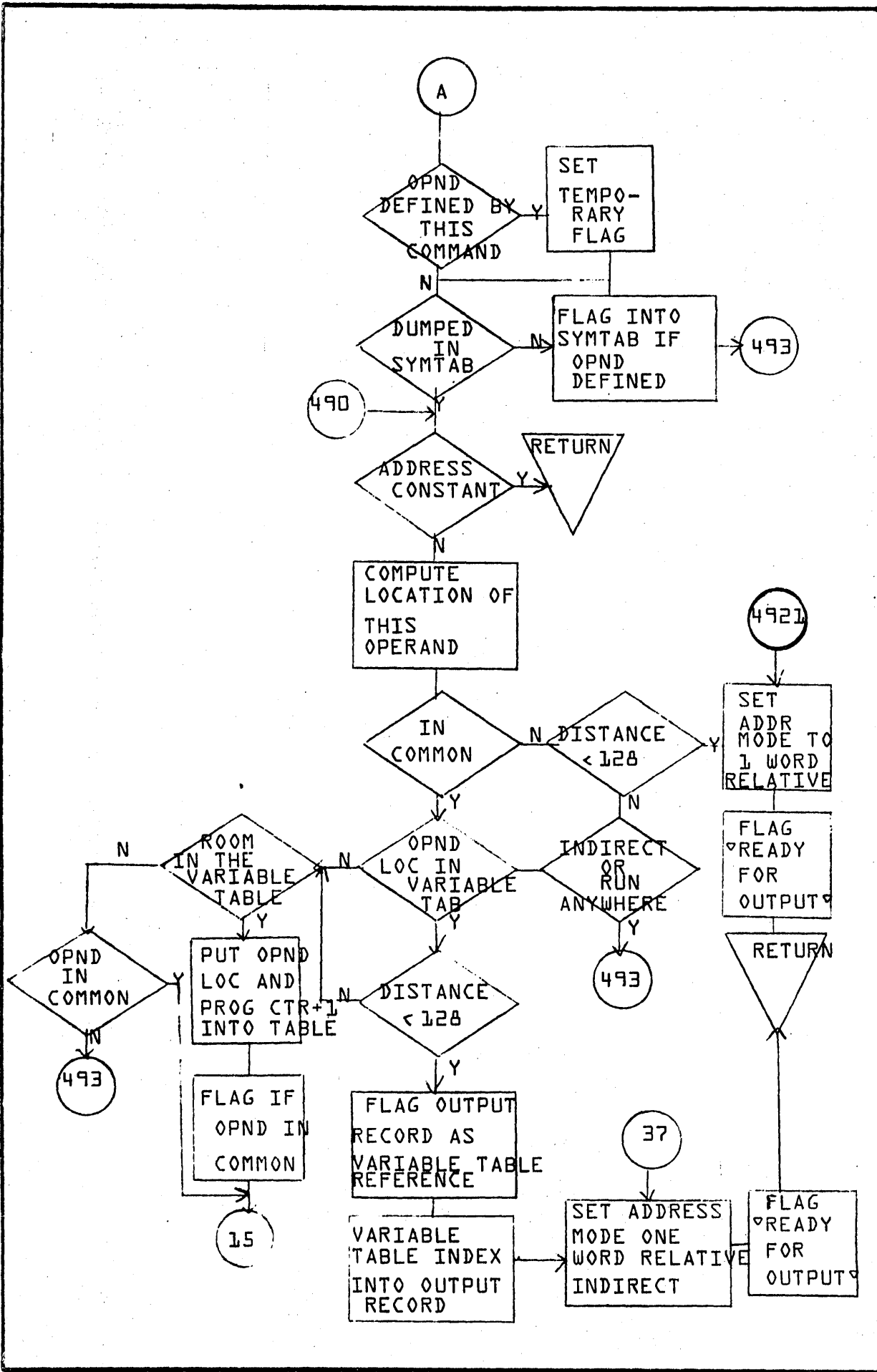
1

A

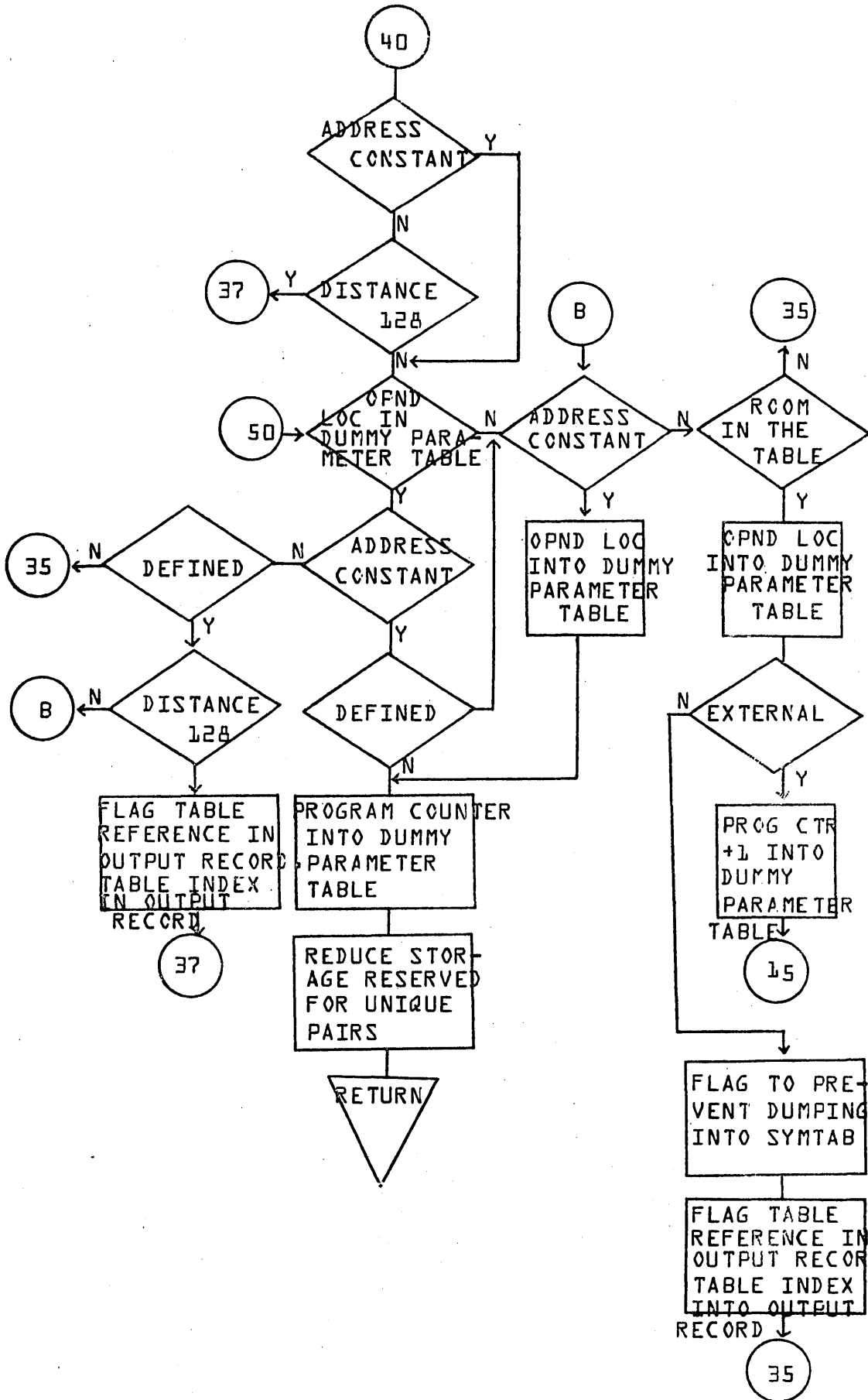
B

C

D



DATE	
APPROVED	
PROJECT NO.	
PROJECT MGR.	
PROJECT NAME	
TASK NO.	
TASK NAME	
DOCUMENT CLASS	IMS
DOCUMENT TITLE	SUBROUTINE CHOP
MACH. TYPE	1700
PAGE	2 OF 2
ISSUE DATE	
NUMBER	
DRAWN BY	
DATE	
<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT SAMPLE CODE <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> FLOWCHART DECISION TABLE OTHER	



DATE		APPROVED	
REV			
PROJECT NO.		PROJECT MGR.	
PROJECT NAME		TASK NO.	
TASK NAME			
MACH. TYPE	IMS	1700	
DOCUMENT TITLE	SUBROUTINE CHOP		
NUMBER	5.1.10.J	ISSUE DATE	
DRAWN BY		DATE	
CONTROL DATA CORPORATION	SOFTWARE DOCUMENT		
SAMPLE CODE	□ □ □ □		
FLOWCHART			
DECISION TABLE			
OTHER			

5

4

3

2

1

A

B

C

D

FORMAT OF IDVTAB

Address of operand Address of Referencing INSTR.
-Address of operand Address of Referencing INSTR.
-Address of operand -Address of Referencing INSTR.

operand is within the body of the Program

operand is in un-labelled Common

operand is in Labelled Common

FORMAT OF NDSTAB

Symbol Table Pointer Address of Address Holder
Symbol Table Pointer □

Points to Symbol Table entry for this Dummy parameter or External

Address Holder to be Generated after Next unconditional Jump.

<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS <i>IMS</i>	MACH. TYPE <i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE <i>FORMAT OF IDVTAB</i>		PROJECT MGR.			
	<i>AND NDSTAB</i>	PAGE <i>1</i> OF <i>1</i>	PROJECT NAME			
	NUMBER	ISSUE DATE	TASK NO.			
DRAWN BY	DATE	TASK NAME				

DOCUMENT CLASS IMS PAGE NO. 5-43  
PRODUCT NAME 1700 Mass Storage FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

#### 5.1.11 Subroutine CL12

Subroutine CL12 processes all storage reference instructions. If the operand of the instruction is an absolute address or a constant the binary output word is completely constructed and the output record is flagged ready for output. If the operand is a reference to a relocatable storage, the operation code field and the address mode field of the binary word are constructed. The address mode is determined either by the previous pass or by a call to Subroutine CHOP.

Special type codes are assigned to compiler generated instructions to save, restore and hold the I register for use in Subroutine ADMAX in the output phase of the Assembler.

#### 5.1.11.1 Flow Chart of Subroutine CL12

#### 5.1.12 Subroutine CON

Subroutine CON generates holders for constants as directed either by the previous pass or by Subroutine LABIN and CHOP in the Assembly input phase.

#### 5.1.12.1 Flow Chart of Subroutine CON

#### 5.1.13 Subroutine DATAST

Subroutine DATAST processes a string of data. Holders for data defined within the body of the object program are generated at the beginning of the object program by subroutine LABIN. Locations of data defined in labeled COMMON are assigned by the previous compiler pass.

An ORG record is constructed for each datum containing a pointer to the location of the array, the datum and address additive. ORG records are typed as program ORG or data ORG according to their location. A data word that is in a string of consecutively located data is typed as a program continuation ORG or a data continuation ORG according to the location of the array. ORG records are flagged ready for output.

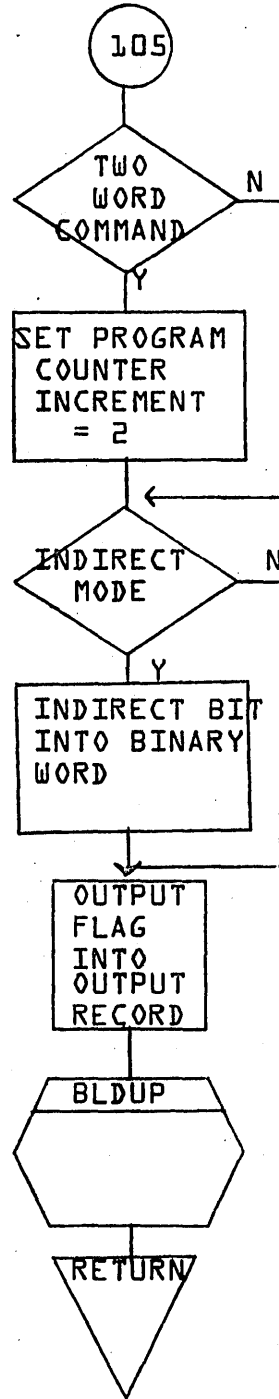
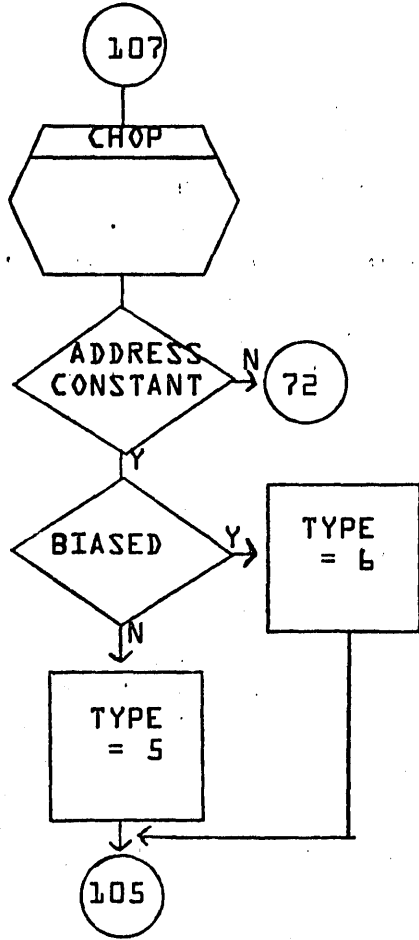
#### 5.1.13.1 Flow Chart of Subroutine DATAST

#### 5.1.14 Subroutine INOUT

Subroutine INOUT is the connecting subroutine between the input phase and the output phase of the FORTRAN Assembly pass. It reads records into the input buffer and writes records into the output buffer until the output buffer is full or an END record is encountered.







DATE	
APPROVED	
REV	
PROJECT NO.	
PROJECT MGR.	
PROJECT NAME	
TASK NO.	
TASK NAME	

DOCUMENT CLASS	IMS	MACH. TYPE	1700
DOCUMENT TITLE	SUBROUTINE CLJ2		
NUMBER	5.1.1.1	PAGE	2 OF 2
DRAWN BY	J.J.J.	ISSUE DATE	

CONTROL DATA CORPORATION  
 SOFTWARE DOCUMENT  
 SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

5

4

3

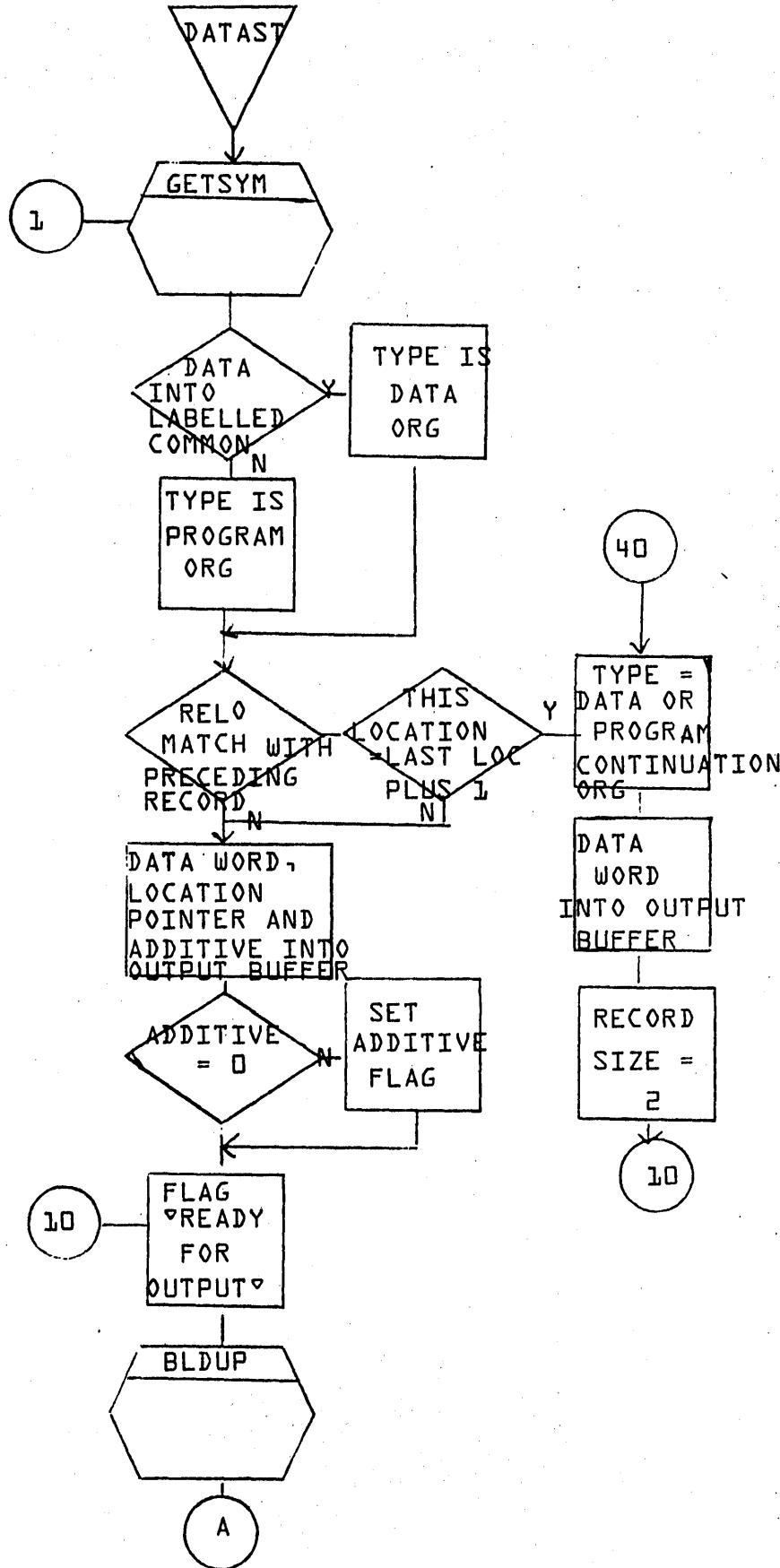
2

A

B

C

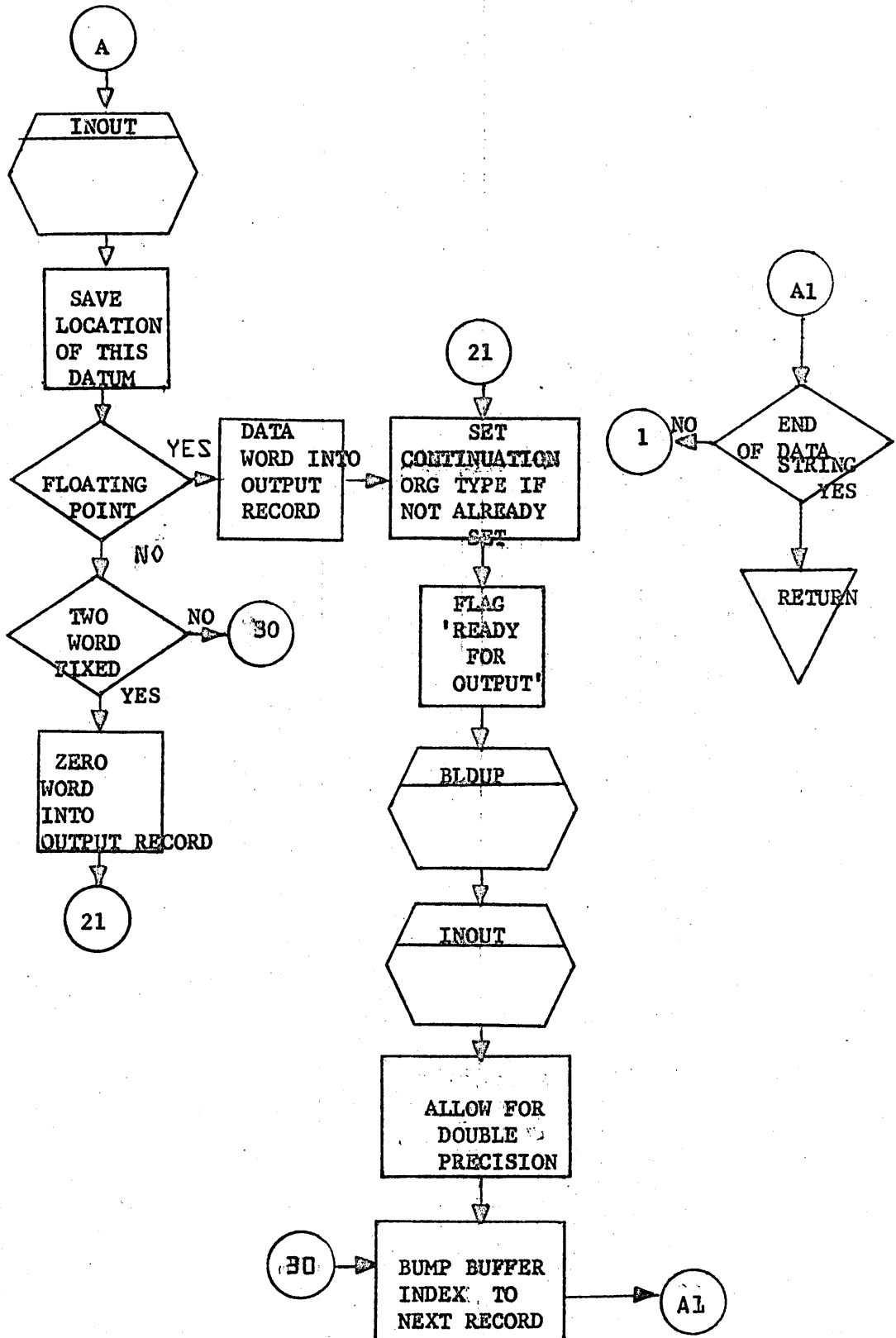
D



CONTROL DATA CORPORATION SOFTWARE DOCUMENT	DOCUMENT CLASS IMS	MACH. TYPE J700	PROJECT NO.	DATE
SAMPLE CODE	DOCUMENT TITLE SUBROUTINE DATAST	PROJECT MGR.	APPROVED	
FLOWCHART	PAGE 1 of 2	PROJECT NAME	REV	
DECISION TABLE	ISSUE DATE	TASK NO.		
OTHER	DRAWN BY	TASK NAME		

**CONTROL DATA**

LA JOLLA RESOURCE CENTER



TITLE		SUBROUTINE DATAST		DRG. NO.
DRAWN BY		PROJ.	DATE	REVISION
CSD 203 ANALYSIS				SHEET 2 OF 2

DOCUMENT CLASS IMS PAGE NO. 5-48  
PRODUCT NAME 1700 Mass Storage FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

When the output buffer is full or an END record is encountered, program control passes to the Assembly output phase through a call to Subroutine AMOUT. On return from Subroutine AMOUT program control is passed back to the assembly input phase unless an unprocessed END record is present in the output buffer. In this case, program control is returned to the output phase.

Subroutine INOUT also reads forward in the input buffer or re-fills the input buffer if necessary, to check for an indexed divide instruction following a long right shift instruction.

#### 5.1.14.1 Flow Chart of Subroutine INOUT

#### 5.1.15 Subroutine IXOPT

The function of Subroutine IXOPT is to assign index registers to indexed instructions. It is called from subroutine PHASE 5 for all pseudo instruction records except labels.

If the instruction being processed is not indexed, Subroutine CHKWD is called to determine its possible effect on future indexing. If the instruction is a store address with no additive present, the operand pointer is saved in the A register holder. Otherwise, the A register holder is cleared.

If the instruction being processed is indexed, any of the five following conditions will result in index register assignment without further analysis of the input buffer.

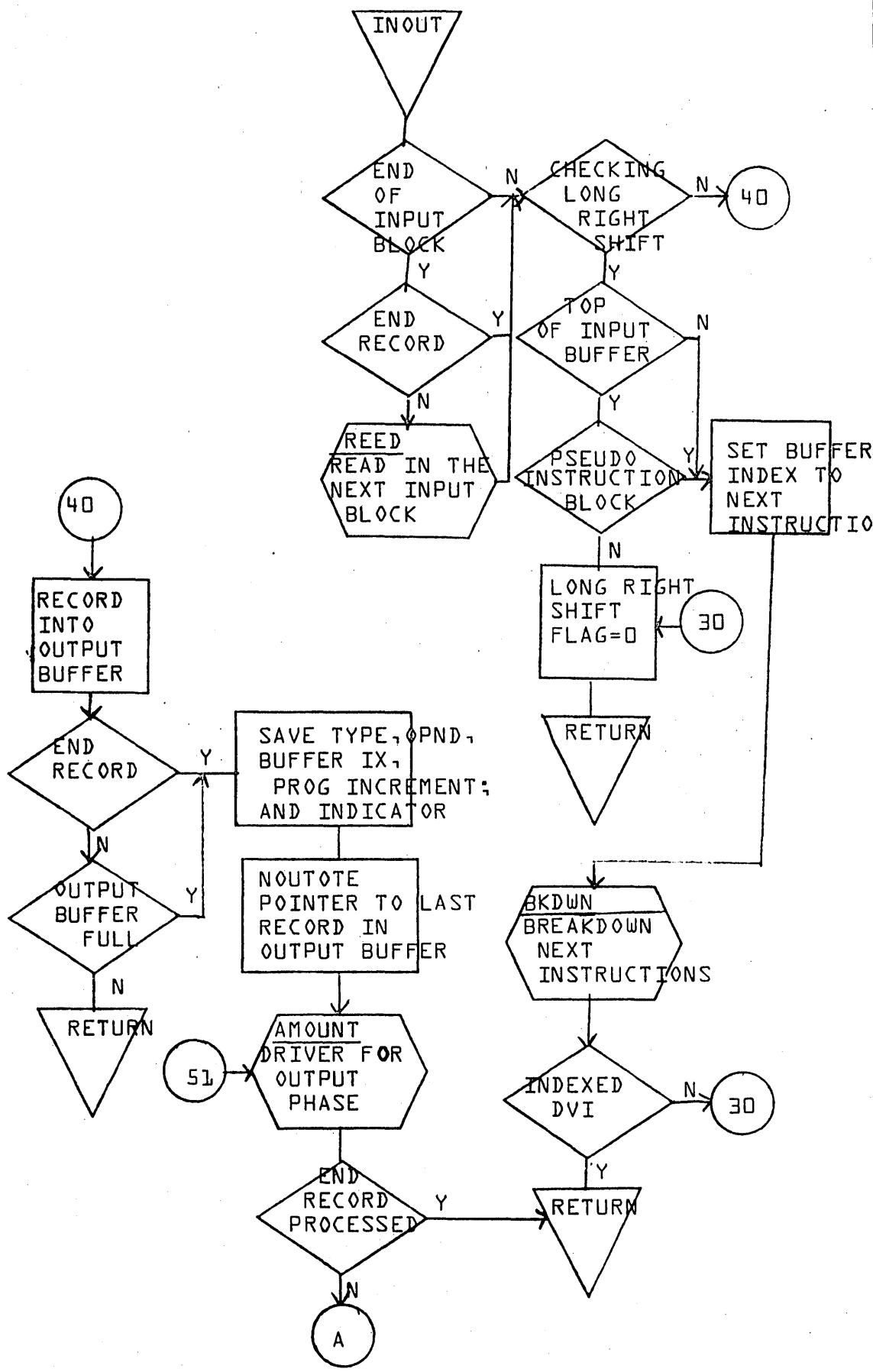
1. This index has been assigned to the Q register by the previous compiler pass.
2. This instruction is an indexed DVI. The index is assigned to the I register.
3. This index is currently in the Q register holder. It is assigned to the Q register.
4. This index is currently in the I register holder. It is assigned to the I register.

No indexing instructions are generated in the above four cases.

5. Subroutine CHKWD sets the decision flag. This index is assigned to the Q register. An LDQ instruction is generated unless the index is in the A register holder. In this case a TRA Q instruction is generated.

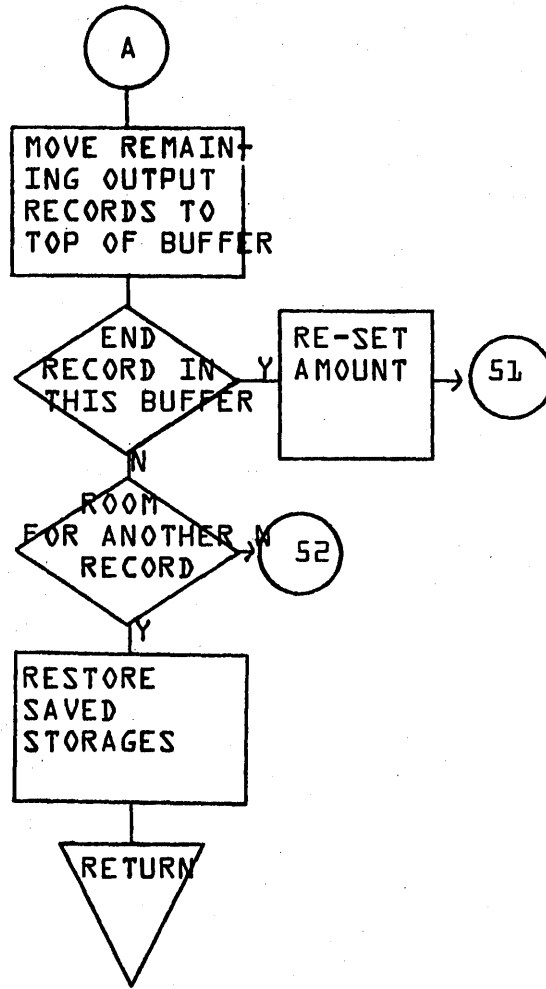
If none of the above five conditions is true, the input buffer is scanned forward until enough information is obtained to make the assignment.

The forward scan stops and the index register under consideration for assignment is assigned to the Q register when one of the



DATE		APPROVED		REV		PROJECT NO.		MACH. TYPE	1700	DOCUMENT CLASS	IMS
						PROJECT MGR.		DOCUMENT TITLE	SUBROUTINE INOUT	NUMBER	
						PROJECT NAME		PAGE	2	ISSUE DATE	
						TASK NO.		DRAWN BY		DATE	
						TASK NAME		CONTROL DATA CORPORATION SOFTWARE DOCUMENT SAMPLE CODE <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> FLOWCHART DECISION TABLE OTHER			

5  
4  
3  
2  
1



CONTROL DATA CORPORATION SOFTWARE DOCUMENT		DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.		APPROVED		DATE	
SAMPLE CODE <input type="checkbox"/>		DOCUMENT TITLE	SUBROUTINE INOUT			PROJECT MGR.					
FLOWCHART <input type="checkbox"/>		NUMBER	5.1.14.1	PAGE	2 OF 2	PROJECT NAME					
DECISION TABLE <input type="checkbox"/>		DRAWN BY		ISSUE DATE		TASK NO.					
OTHER <input type="checkbox"/>				DATE		TASK NAME					

A

B

C

D

DOCUMENT CLASS IMS PAGE NO 5-51  
PRODUCT NAME 1700 Mass Storage FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

following conditions is true.

1. The decision flag is set in Subroutine CHKWD.
2. The same register as that in the I register holder appears.
3. The same register as that in the Q register holder appears. A STQ I instruction is generated before the index load instruction.
4. A different index appears twice. An index is counted as appearing twice only if its occurrence flag has been set since its last appearance.
5. A DVI instruction with no index or with an index different from the one under consideration is encountered.
6. A label record is encountered.
7. The scan reaches the end of the input buffer with no decision.

An LDQ instruction is generated except when the index is in the A register holder. In this case a TRA Q instruction is generated.

The forward scan stops and the index under consideration for assignment is assigned to the I register when one of the following conditions is true.

1. The index under consideration appears again. It is counted as appearing again only if its occurrence flag has been set.
2. A DVI is encountered with an index the same as the one under consideration for assignment.

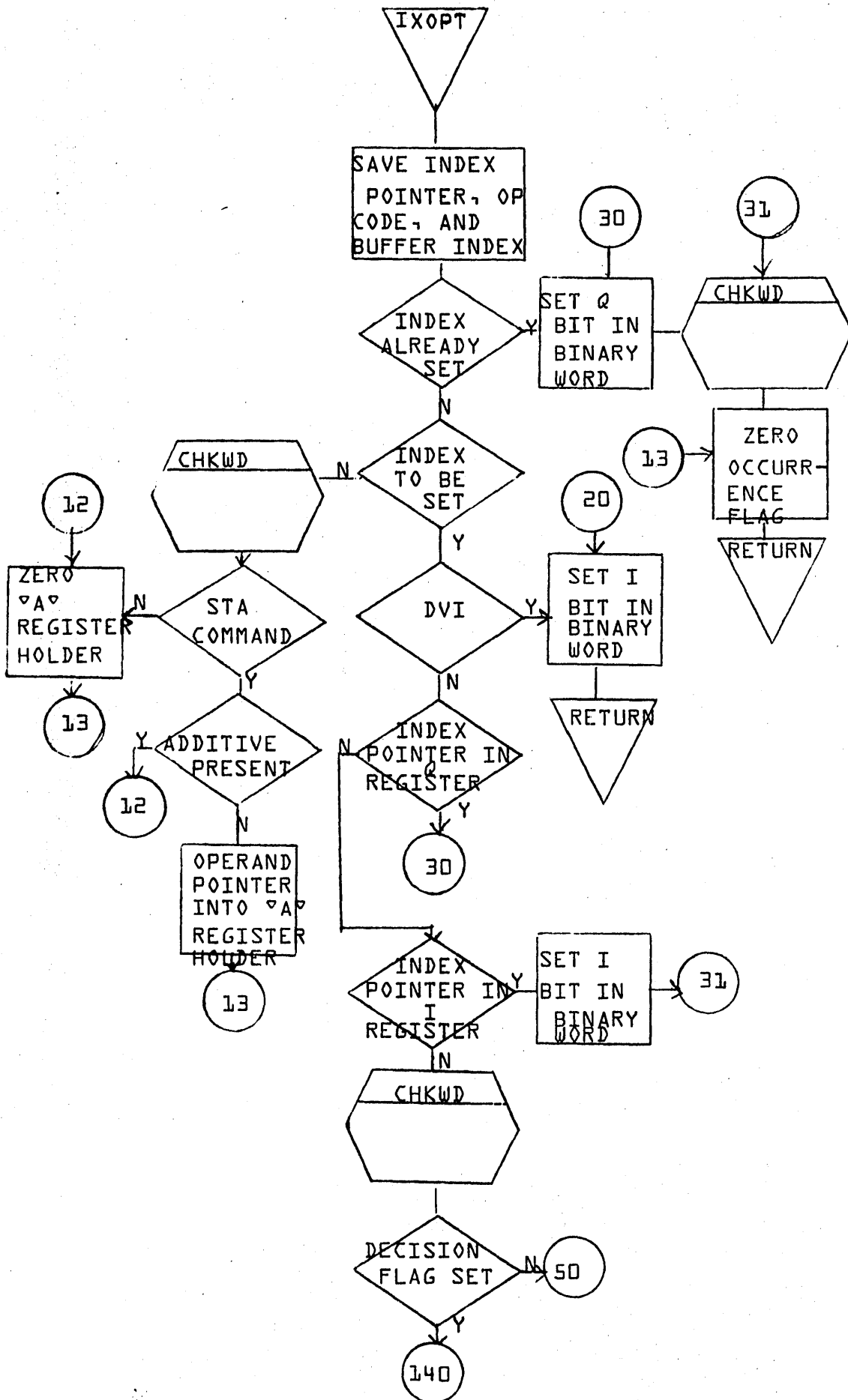
An STA I instruction is generated if the index under consideration is in the A register holder. LDA, STA I instructions are generated if the instruction being processed is an LDA. Otherwise, the instructions generated are LDQ, STQ I.

The occurrence flag is set for the index under consideration if a Q destroying instruction is encountered in the forward scan of the input buffer. The occurrence flags for the index under consideration and all saved indexes are set when an index appears that has not previously been encountered either in the buffer or in the Q or I register holders.

5.1.15.1 Flow Chart of Subroutine IXOPT

5.1.16 Subroutine LABEL

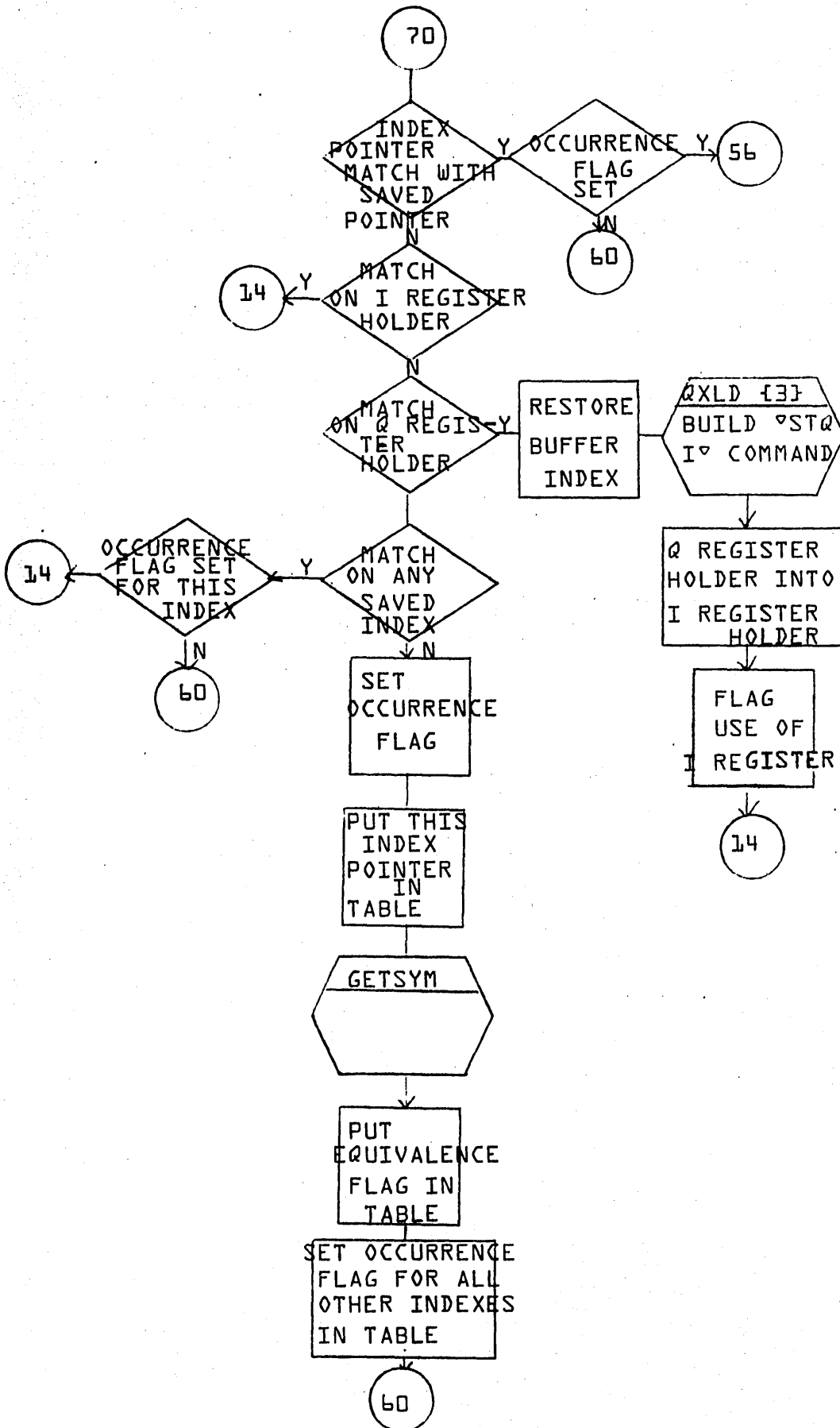
Subroutine LABEL constructs output records for statement and variable labels. The program counter is stored in the symbol table entry for the label being processed. The record is flagged ready for output.



DATE		APPROVED		REV	
PROJECT NO.		PROJECT MGR.		PROJECT NAME	
TASK NO.		TASK NAME		TASK NAME	
DOCUMENT CLASS	IMS	MACH. TYPE	1700	PAGE	OF 4
DOCUMENT TITLE	SUBROUTINE IXOPT	ISSUE DATE		DATE	
NUMBER		DRAWN BY			
CONTROL DATA CORPORATION SOFTWARE DOCUMENT <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> SAMPLE CODE <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> FLOWCHART <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> DECISION TABLE <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> OTHER <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>					







DATE	
APPROVED	
REV	
PROJECT NO.	
PROJECT MGR.	
PROJECT NAME	
TASK NO.	
TASK NAME	
DOCUMENT CLASS	IMS
MACH. TYPE	1700
DOCUMENT TITLE	SUBROUTINE IXOPT
NUMBER	PAGE 3 OF 4
ISSUE DATE	
DRAWN BY	
DATE	

CONTROL DATA CORPORATION  
 SOFTWARE DOCUMENT      
 SAMPLE CODE      
 FLOWCHART      
 DECISION TABLE      
 OTHER

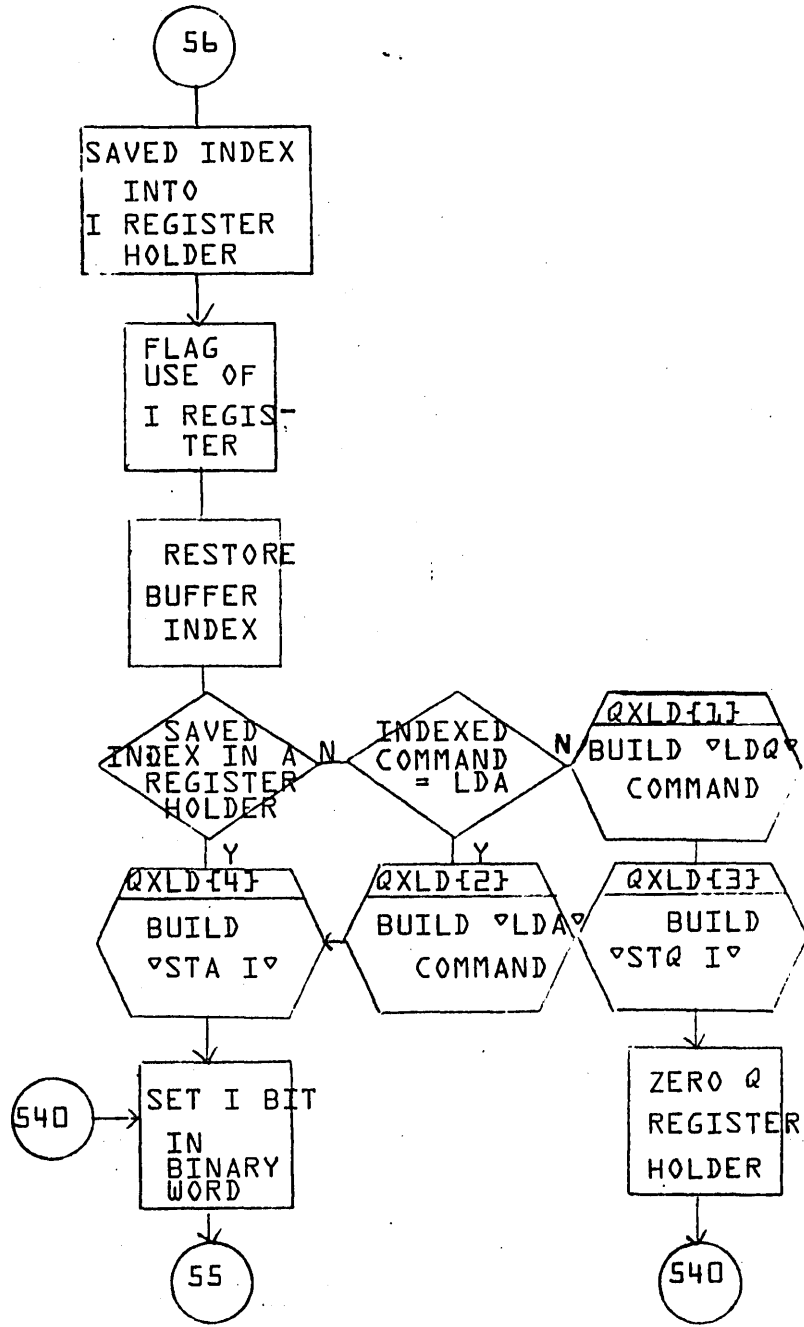
5

4

3

2

1



DATE		APPROVED		REV	
PROJECT NO.		PROJECT MGR.		PROJECT NAME	
MACH. TYPE J700		SUBROUTINE IXOPT		PAGE 4 of 4	
DOCUMENT CLASS IMS		ISSUE DATE		TASK NO.	
DOCUMENT TITLE		DRAWN BY		TASK NAME	
NUMBER		DATE		TASK NAME	

**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

SAMPLE CODE

FLOWCHART

DECISION TABLE

OTHER

A

B

C

D

DOCUMENT CLASS IMS PAGE NO. 5-56  
PRODUCT NAME 1700 Mass Storage FORTRAN  
PRODUCT NO. CD05 VERSION 2.0 MACHINE SERIES 1700

5.1.16.1 Flow Chart of Subroutine LABEL

5.1.17 Subroutine LABIN

Subroutine LABIN is called from subroutine PHASE C to assign storage locations to certain variables holders, constants and arrays. LABIN is called only if the first executable statement of the object program, excluding the jump at the beginning of a main program, has not been encountered.

Storage locations are assigned for all equivalenced arrays, all DATA arrays and all other arrays which occupy more than twenty locations. Arrays occupying twenty locations or less are assigned storage locations if their use count is less than one third the size of the array.

Storage locations are also assigned to variable holders which are used only once and to referenced constants.

All other variable holders, constants and arrays are ignored by subroutine LABIN. Storage assignments will be made for these items after they are used in the object program.

5.1.17.1 Flow Chart of Subroutine LABIN

5.1.18 Subroutine QXLD

The function of Subroutine QXLD is to generate indexing instructions.

According to the input parameter, QXLD will generate a LDA, LDQ, STA I, STQ I or TRA Q. If the instruction to be generated is a LDA or LDQ, Subroutine CHOP is called to determine the addressing mode. Subroutines BLDUP, INOUT and COUNT are called to output the instruction and bump the program counter.

5.1.18.1 Flow Chart of Subroutine QXLD

5.1.19 Subroutine SKIP

Subroutine SKIP generates one or two conditional skip instructions and a two-word relative unconditional jump instruction for every conditional jump instruction encountered in the input buffer.

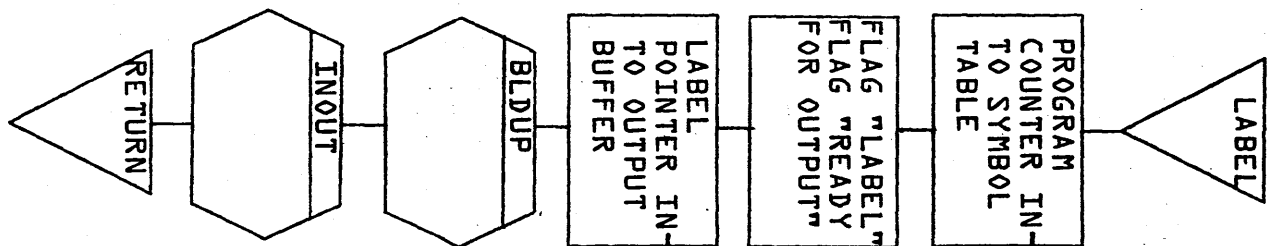
The skip records are flagged as fixed length. The jump instruction is a candidate for reduction to one word or elimination in the Assembler output phase.

Below are listed the conditional jump instructions received from the previous compiler pass and the corresponding instructions generated by the assembler:

DOCUMENT CLASS IMS PAGE NO. 5-57  
 PRODUCT NAME 1700 Mass Storage FORTRAN  
 PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

<u>Conditional Jump</u>	<u>Instruction Generated</u>
A Jump $\neq$ 0	SAZ 2 JMP
A Jump = 0	SAN 2 JMP
A Jump $<$ 0	SAP 2 JMP
A Jump $\geq$ 0	SAM 2 JMP
A Jump $\leq$ 0	SAZ 1 SAP 2 JMP
A Jump $>$ 0	SAZ 3 SAM 2 JMP
Q Jump = 0	SQN 2 JMP
Q Jump $\neq$ 0	SQZ 2 JMP

5.1.19.1 Flow Chart of Subroutine SKIP



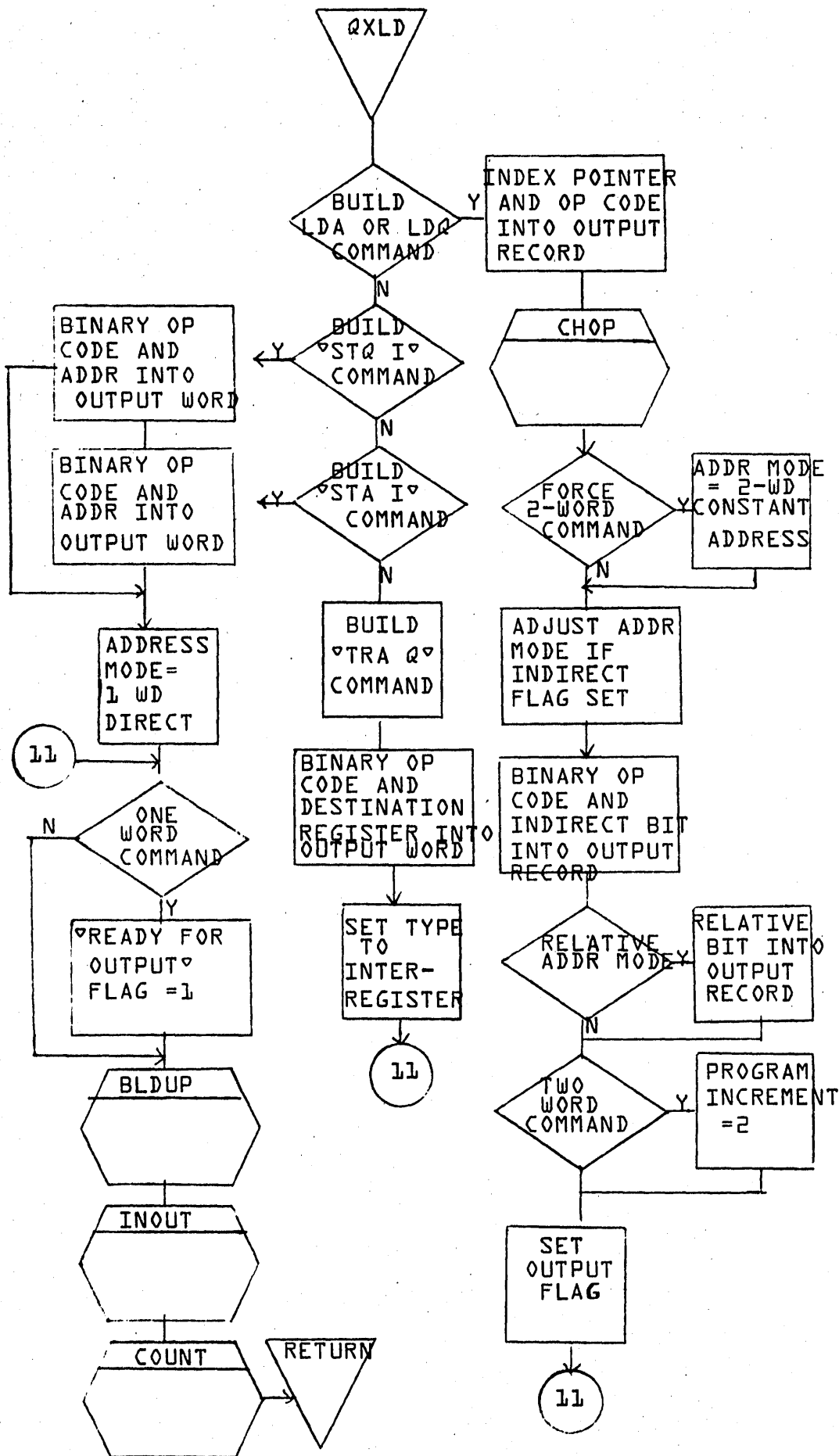
**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	SUBROUTINE LABEL			PROJECT MGR.			
PAGE 1 OF 1				PROJECT NAME			
NUMBER	5.1.16.1	ISSUE DATE		TASK NO.			
DRAWN BY		DATE		TASK NAME			

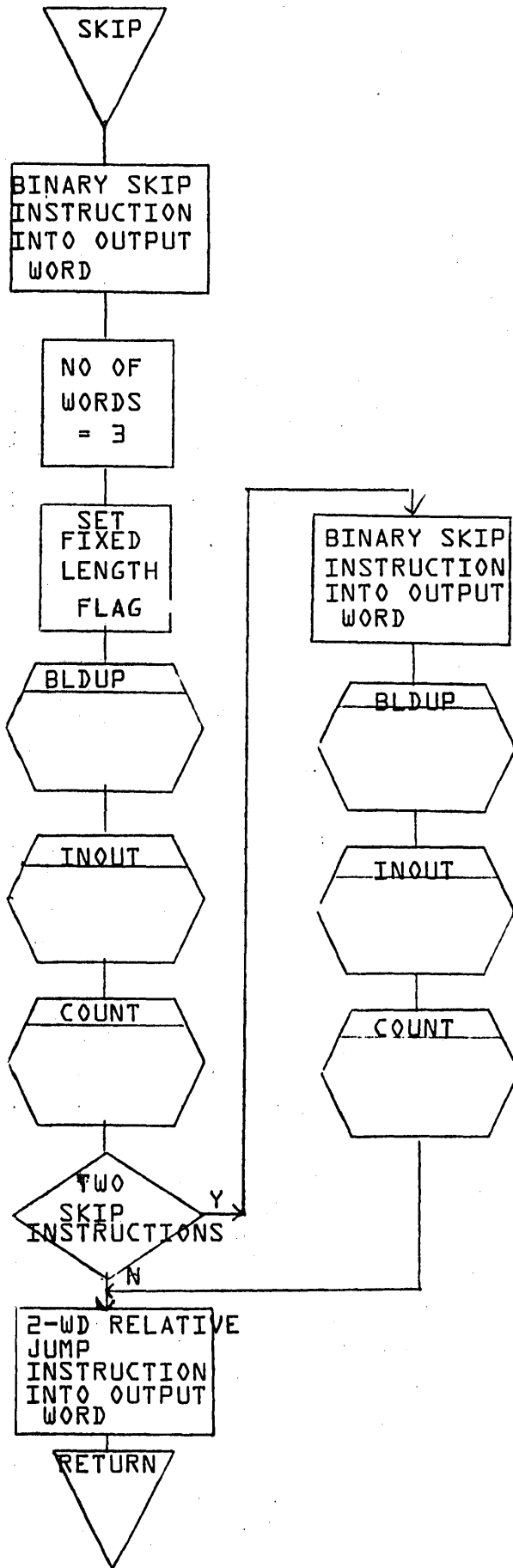
5-58





DATE	APPROVED	REV	PROJECT NO.	PROJECT MGR.	PROJECT NAME	TASK NO.	TASK NAME
			IMS	1700	SUBROUTINE QXLD	PAGE 1 OF 1	
CONTROL DATA CORPORATION		DOCUMENT CLASS		ISSUE DATE	DRAWN BY		
SOFTWARE DOCUMENT		NUMBER		DATE			
SAMPLE CODE		FLOWCHART		DECISION TABLE			OTHER
<input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>							





CONTROL DATA CORPORATION SOFTWARE DOCUMENT		DOCUMENT CLASS	IMS	MACH. TYPE	J700	PROJECT NO.		REV		APPROVED		DATE	
SAMPLE CODE		DOCUMENT TITLE	SKIP SUBROUTINE		PAGE	1 of 1	PROJECT MGR.						
FLOWCHART		NUMBER		ISSUE DATE		PROJECT NAME							
DECISION TABLE		DRAWN BY		DATE		TASK NO.							
OTHER						TASK NAME							

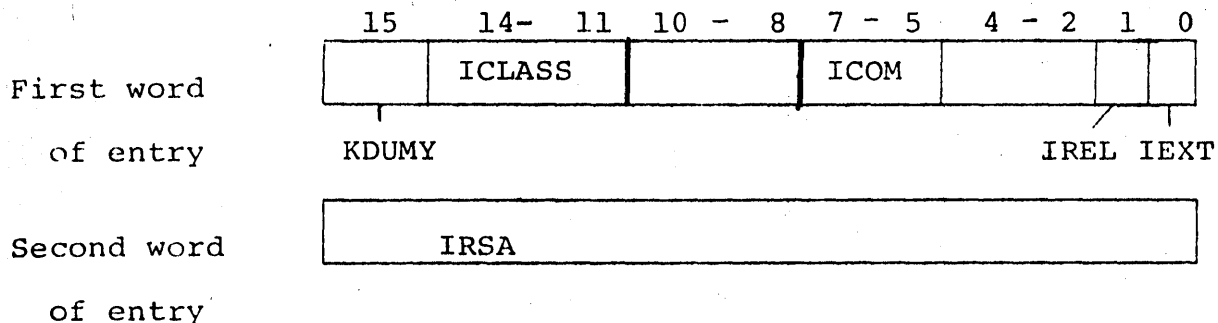
DOCUMENT CLASS IMS PAGE NO. 5-62  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. COO5 V.2.0 MACHINE SERIES 1700

## 5.2 ASSEMBLER OUTPUT PHASE D/E

In phase D/E the pseudo-instructions output by phase C are converted to relocatable code. Address optimization and final assignment take place in this phase. The 'D/E' notation is used because for a given compilation either phase D or phase E is executed, but not both. Phase E is loaded and executed whenever the A or M options (assembly listing) are requested, otherwise phase D is used. The relocatable code output by phase D/E is punched on cards and/or stored on the load-and-go file according to the P and X options respectively.

### 5.2.0.1 Distinction between Phases D and E.

Phases D and E are virtually identical with two major exceptions. First, subroutine SETPRT and all calls to SETPRT are omitted from Phase D. Second, in the 2.0A version of 1700 FORTRAN phase D uses a 'compressed' symbol table requiring two words per entry, as opposed to the five words required by the normal symbol table. The structure of this compressed symbol table is:



The meanings of the above variable names are found in the Chapter 7, where the normal symbol table is described. In the 2.0A version of Phase D, subroutine BEGINO packs the normal symbol table into the compressed symbol table. Subroutines GETSYM and SYMSCN are altered to reference the compressed table, and all routines except BEGINO and FINISH have access only to the compressed symbol table. In the 2.0A version of Phase D, subroutine AMOUT stores the IRSA entries of externals into INBUFF after the END record is processed, and subroutine FINISH retrieves these entries from INBUFF rather than from the normal symbol table.

DOCUMENT CLASS \_\_\_\_\_ IMS \_\_\_\_\_ PAGE NO. \_\_\_\_\_ 5-63  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. COO5 V.2.0 MACHINE SERIES 1700

### 5.2.0.2 Circular output buffer

Subroutine PHASE6 in phase D/E reads the output of phase C and stores it in the output buffer NOBUFF. For effective address optimization, NOBUFF must be filled at the 'bottom' as instructions at the 'top' are converted to relocatable code and output. To speed this process the storage assigned to NOBUFF is referenced by 'circular buffer pointers' that always increase. The NOBUFF subscript is calculated by reducing the given pointer modulo JNOB (NOBUFF size) using the function INDEX. In this way new information can be added to the output buffer without moving the unprocessed information in core, and the virtual size of NOBUFF is infinite. Subroutine PHASE6 refills NOBUFF from the phase C disk output file as necessary, and initializes and maintains the major circular buffer pointers IHEAD ('top' of buffer), NOUTOT ('bottom' of buffer), and NOCT (next available buffer location).

### 5.2.0.3 Summary of Phase D/E Flow

Records flagged ready for output are sent through the output routines starting with the first record in the buffer and continuing until a record is encountered that is not flagged ready for output. If this record is not sitting at the 'top' of the buffer, it is pushed to the 'top'. If no end record is present in the buffer, control is returned to PHASE6 to refill the buffer at the 'bottom.' On return, a forward scan of the buffer is initiated to find instructions that are one-word candidates. When a one-word candidate is encountered, subroutine ADMAX is called to decide whether or not to reduce the instruction to one word. The methods used to delete commands, maintain the program counter and make final assignment of operand addresses are fully detailed in the description of subroutines AMOUT and ADMAX. The scan of the output buffer is repeated once. At this point the record at the top of the buffer plus any following records flagged ready for output are forced out of the buffer. If no END record is present in the output buffer, control is returned to PHASE6 to fill the output buffer and the process is repeated.

DOCUMENT CLASS \_\_\_\_\_ IMS \_\_\_\_\_ PAGE NO. \_\_\_\_\_ 5-64  
PRODUCT NAME \_\_\_\_\_ 1700 MASS STORAGE FORTRAN \_\_\_\_\_  
PRODUCT MODEL NO. \_\_\_\_\_ C005 V.2.0 \_\_\_\_\_ MACHINE SERIES 1700 \_\_\_\_\_

### 5.2.1 Subroutine PHASE6

Subroutine PHASE6 is the main processor for phase D/E. It initializes the circular buffer pointers, calls subroutine BEGNO, fills the output buffer NOBUFF, and calls subroutine AMOUT. At every return from AMOUT, subroutine PHASE6 refills the output buffer at the 'bottom', overlaying the pseudo-instructions that have been processed by AMOUT. After AMOUT has processed the END record and returned, PHASE6 calls subroutine FINISH, punches the transfer card at the end of the relocatable output, and returns.

#### 5.2.1.1 Flowchart of Subroutine PHASE6

### 5.2.2 Subroutine BEGNO

Subroutine BEGNO prints and punches the initial records of the final compiler output. It prints the length of labelled and blank common, converts the name of the program being compiled to ASCII code, prints (phase E only) and punches the NAM record and the DAT and COM list. In the 16K version of 1700 FORTRAN, subroutine BEGNO halves the symbol table page size and packs the full symbol table into the two-word entry compressed table used by phase D.

#### 5.2.2.1 Flowchart of Subroutine BEGNO

### 5.2.3 Subroutine AMOUT

Subroutine AMOUT is the main processor for the output phase of the FORTRAN Assembler. Its function is to maximize use of one-word instructions, eliminate unneeded instructions, and send binary and ASCII output to the punch and list routines.

AMOUT makes a primary scan through the output buffer sending all instructions that are ready for output through processing and to the output routines until an instruction is encountered that is not ready for output. The output buffer is then moved up to eliminate the processed instructions and PHASE6 is reentered to fill the buffer.

If the instruction at the 'top' of the buffer is not ready for output, a secondary scan is initiated in which no output is generated. Every instruction that is a candidate for reduction to one word or elimina-

## CONTROL DATA CORPORATION

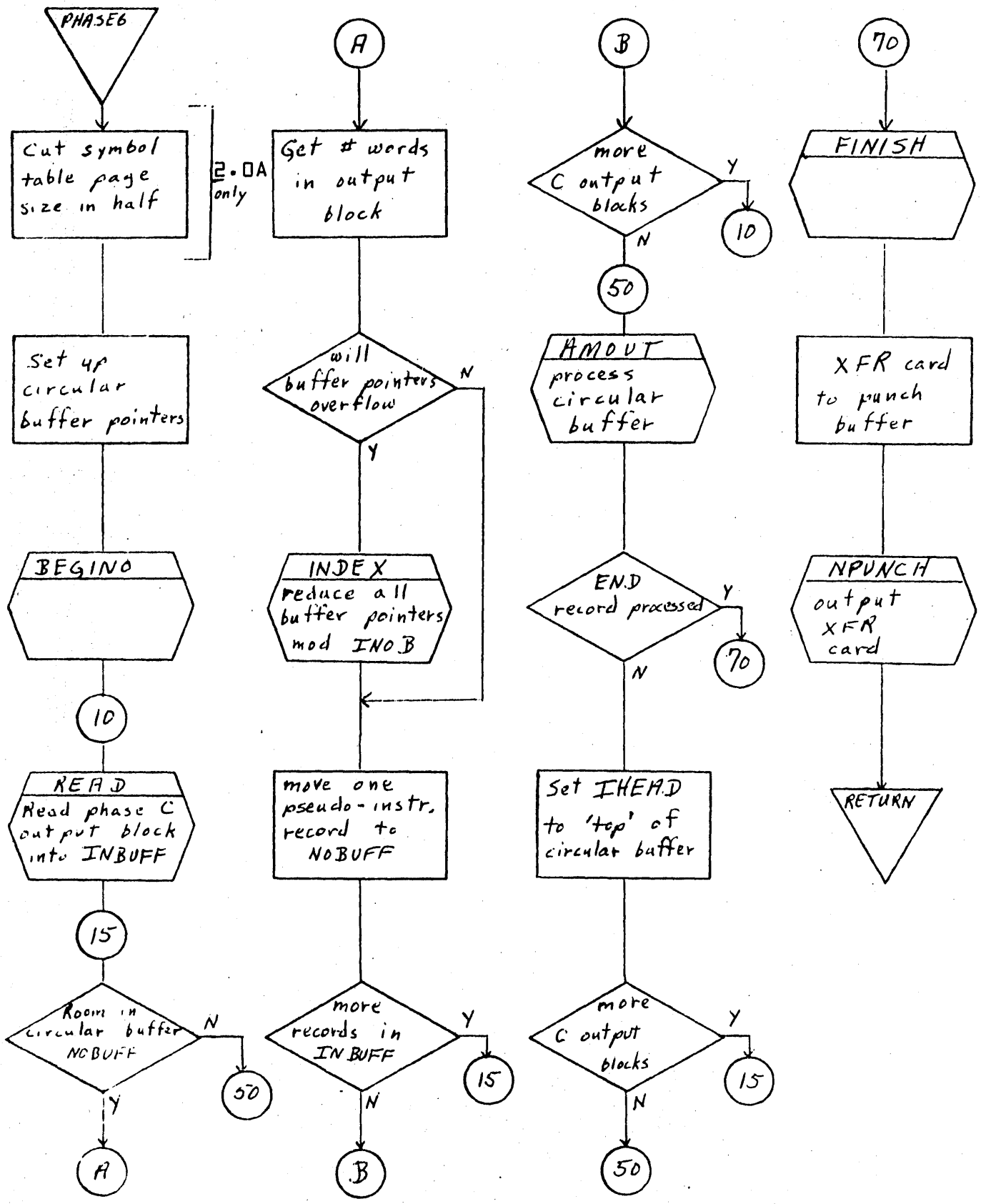
DIVISION

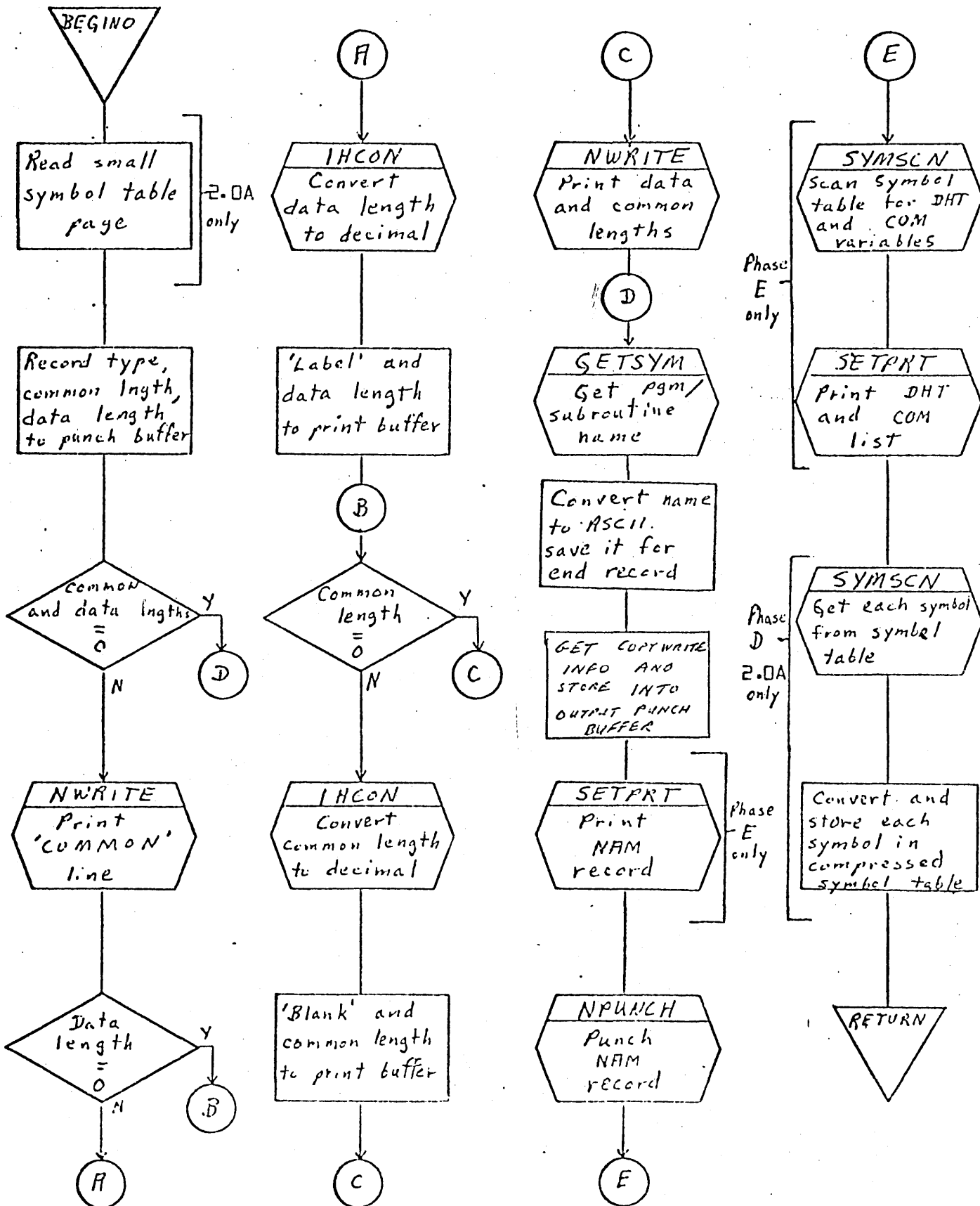
DOCUMENT CLASS \_\_\_\_\_ IMS \_\_\_\_\_ PAGE NO. \_\_\_\_\_ 5-65  
PRODUCT NAME \_\_\_\_\_ 1700 MASS STORAGE FORTRAN \_\_\_\_\_  
PRODUCT MODEL NO. \_\_\_\_\_ C005 V.2.0 \_\_\_\_\_ MACHINE SERIES \_\_\_\_\_ 1700 \_\_\_\_\_

tion is sent to Subroutine ADMAX. ADMAX sets a change flag if any instruction is reduced or eliminated. The entire output buffer is scanned once. The program counter is initialized for each scan and kept current throughout the scan.

In the 2.0A version, AMT is used to call AMOUT.

DOCUMENT CLASS IMS PAGE NO. \_\_\_\_\_  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. CO05 V.2.0 MACHINE SERIES 1700





DOCUMENT CLASS IMS PAGE NO 5-68  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

After the optimization scan, the instruction at the top of the buffer must be output. If the address reference has not yet been identified or if the distance from the current location to the address reference is more than  $1256_{10}$ , the instruction is stored in the future table. A binary output record to be read in at the current location at object time execution will be generated when the referenced location appears at the top of the output buffer.

If the distance from the current location to the address reference is equal to or less than  $1256_{10}$ , the output buffer is fixed to the point of the forward reference. This is accomplished by storing the location of the forward reference in IFIX. The number or size of instructions to be output may not be changed as long as the location of the instruction or the location of its forward reference is less than or equal to IFIX.

Processing of the instruction at the top of the buffer is completed and the record is flagged ready for output. AMOUT then goes back to the primary scan.

When the END record appears in the output buffer, IFIX is set negative at the end of the secondary scan to force all instructions out of the buffer.

#### 5.2.3.1 Flow Chart of Subroutine AMOUT

#### 5.2.3.2 Format of the Future Table, NFTAB

#### 5.2.4 Subroutine ADMAX

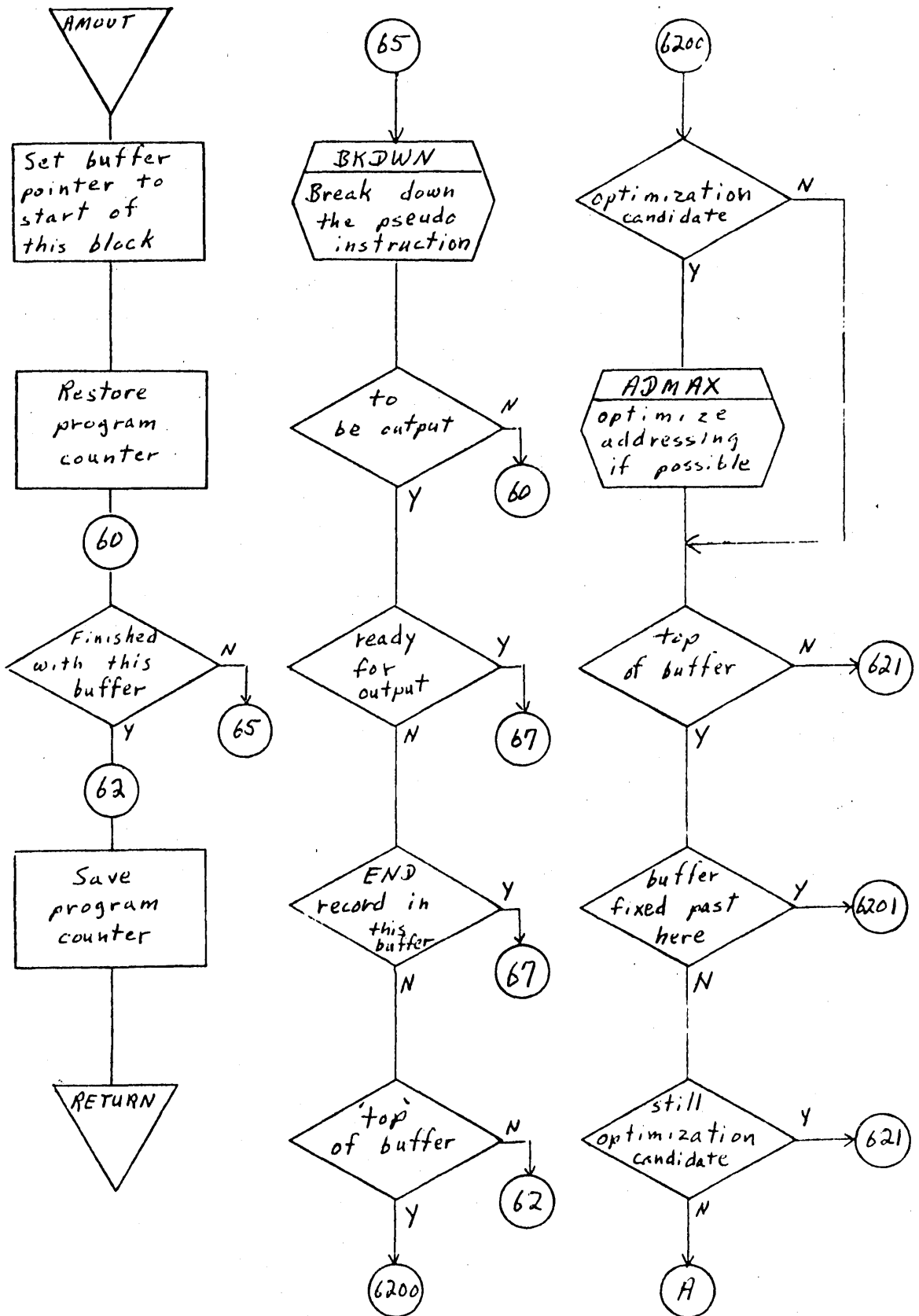
Subroutine ADMAX analyzes the instructions sent to it for possible reduction in size or elimination from the object program. It is called from subroutine AMOUT when a record is encountered that is not flagged as fixed length or ready for output.

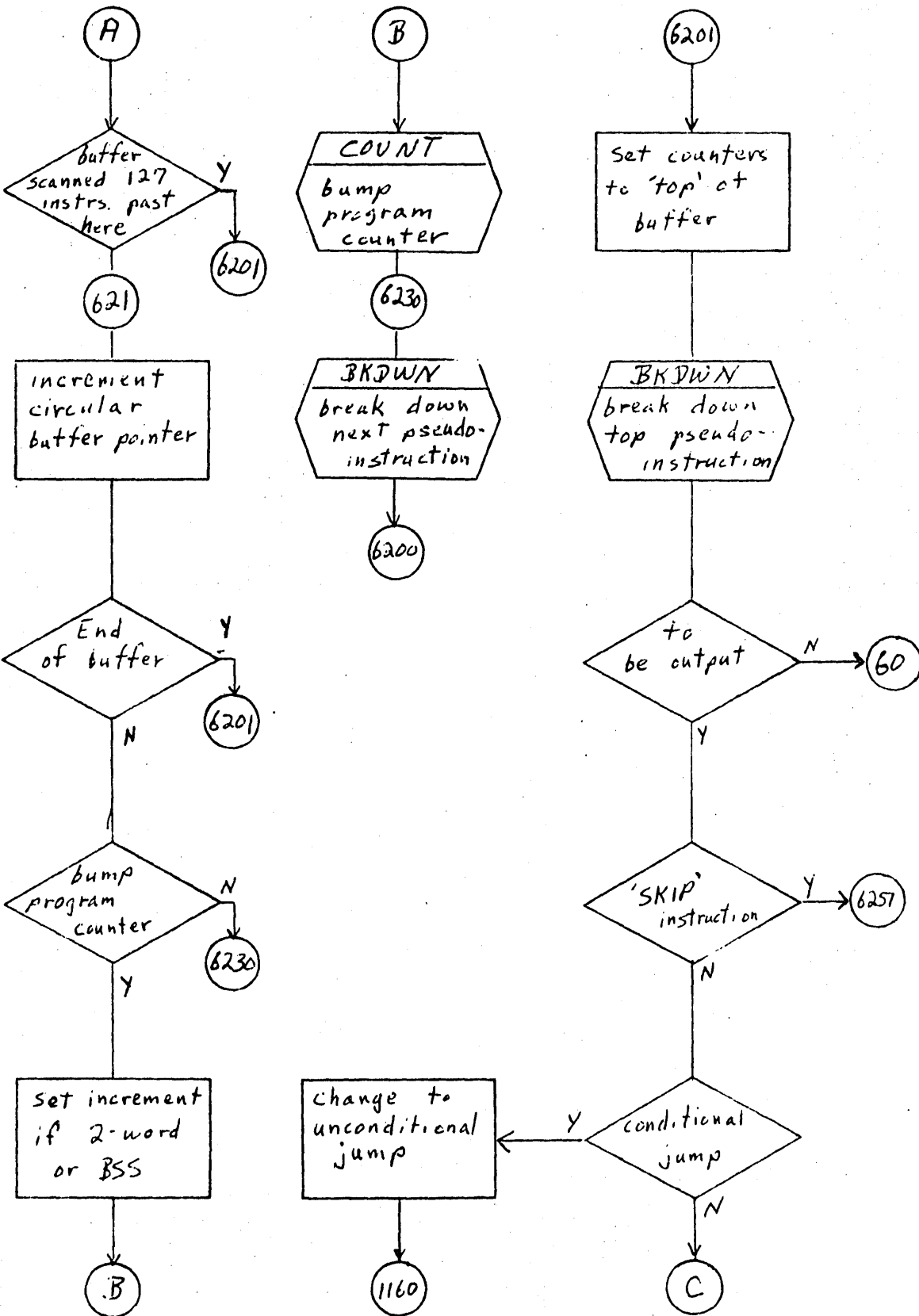
There are three situations which may produce reduction in size or elimination of an instruction:

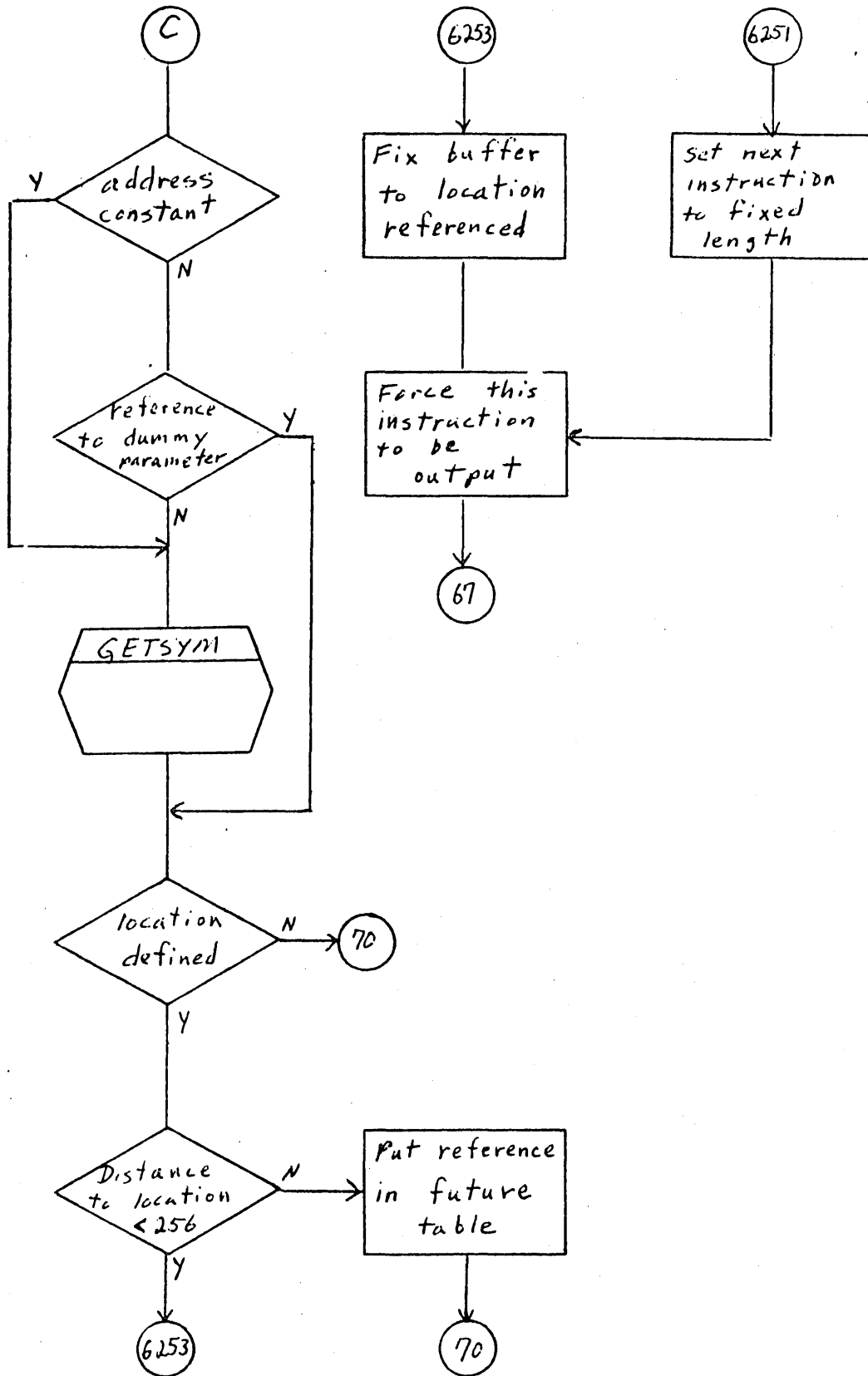
1. The I register is not used in the object program. The compiler generated instructions to save and restore the I register and the I register holder are eliminated.
2. The program counter is not fixed and the distance to the address reference is less than  $128_{10}$ . The type of the two-word instruction is changed to the corresponding one-word type; that is, two-word relative direct or indirect becomes one-word relative direct or indirect.

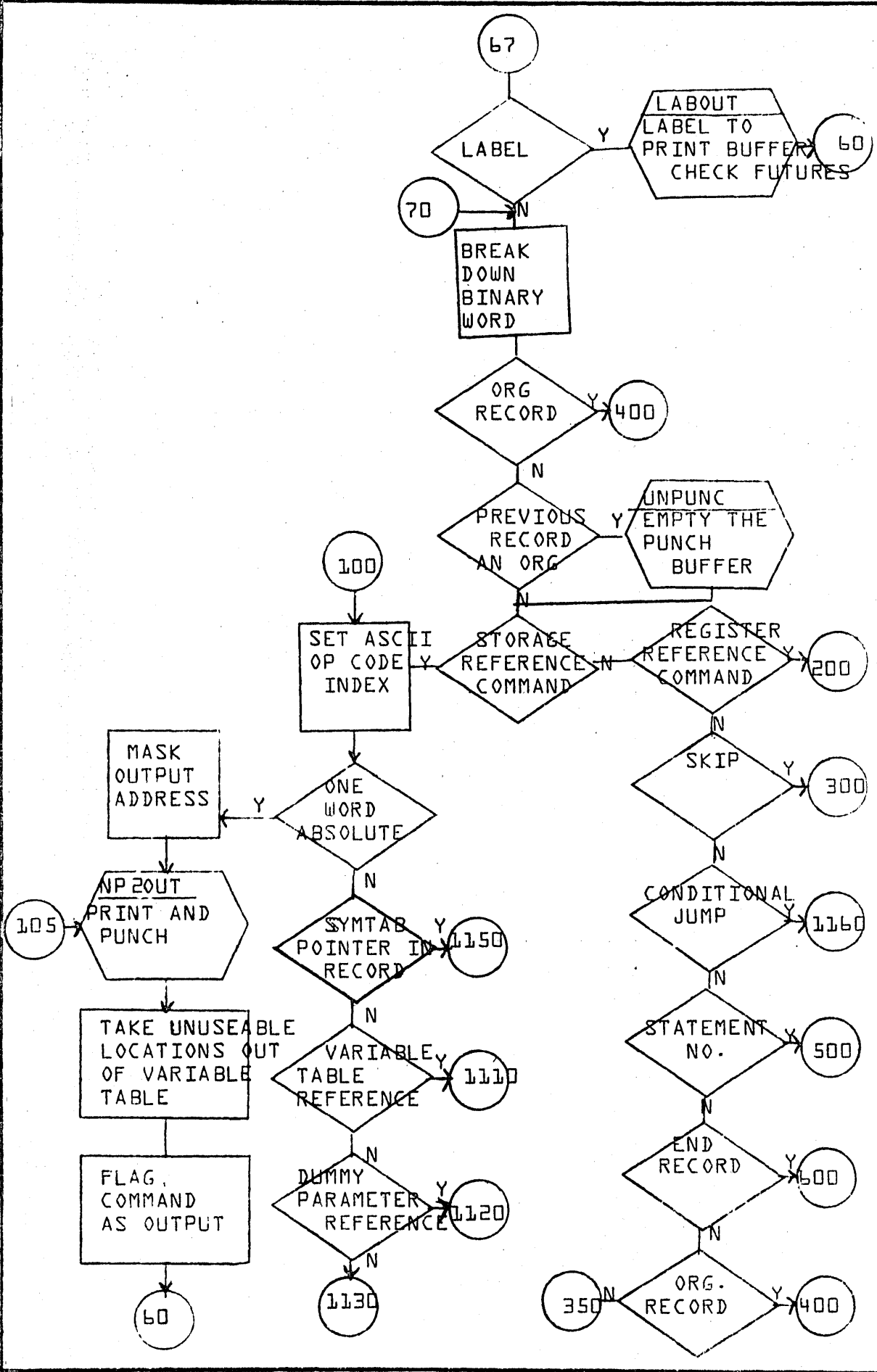


DOCUMENT CLASS \_\_\_\_\_ IMS \_\_\_\_\_ PAGE NO. \_\_\_\_\_  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. COO5 V.2.0 MACHINE SERIES 1700



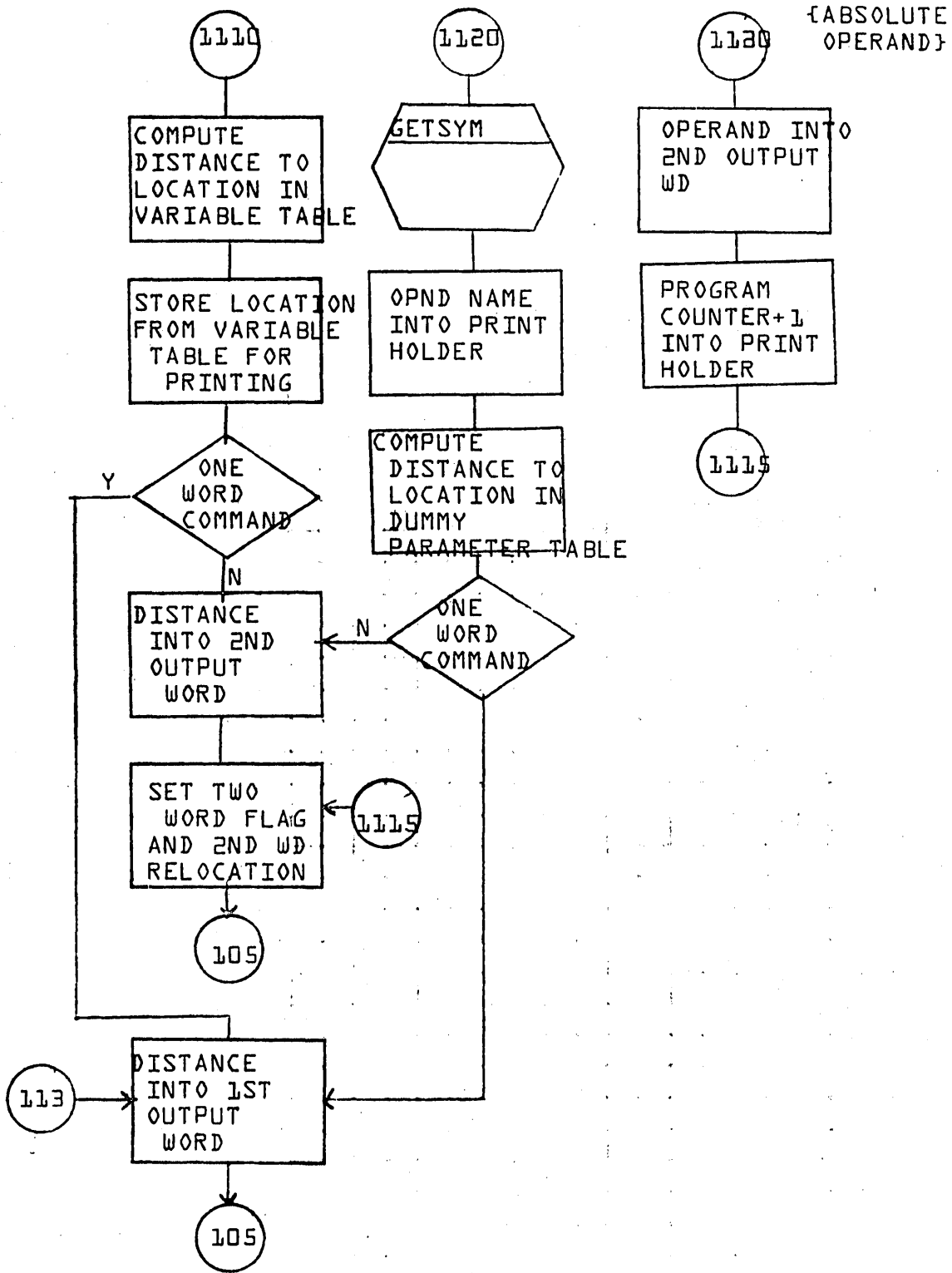




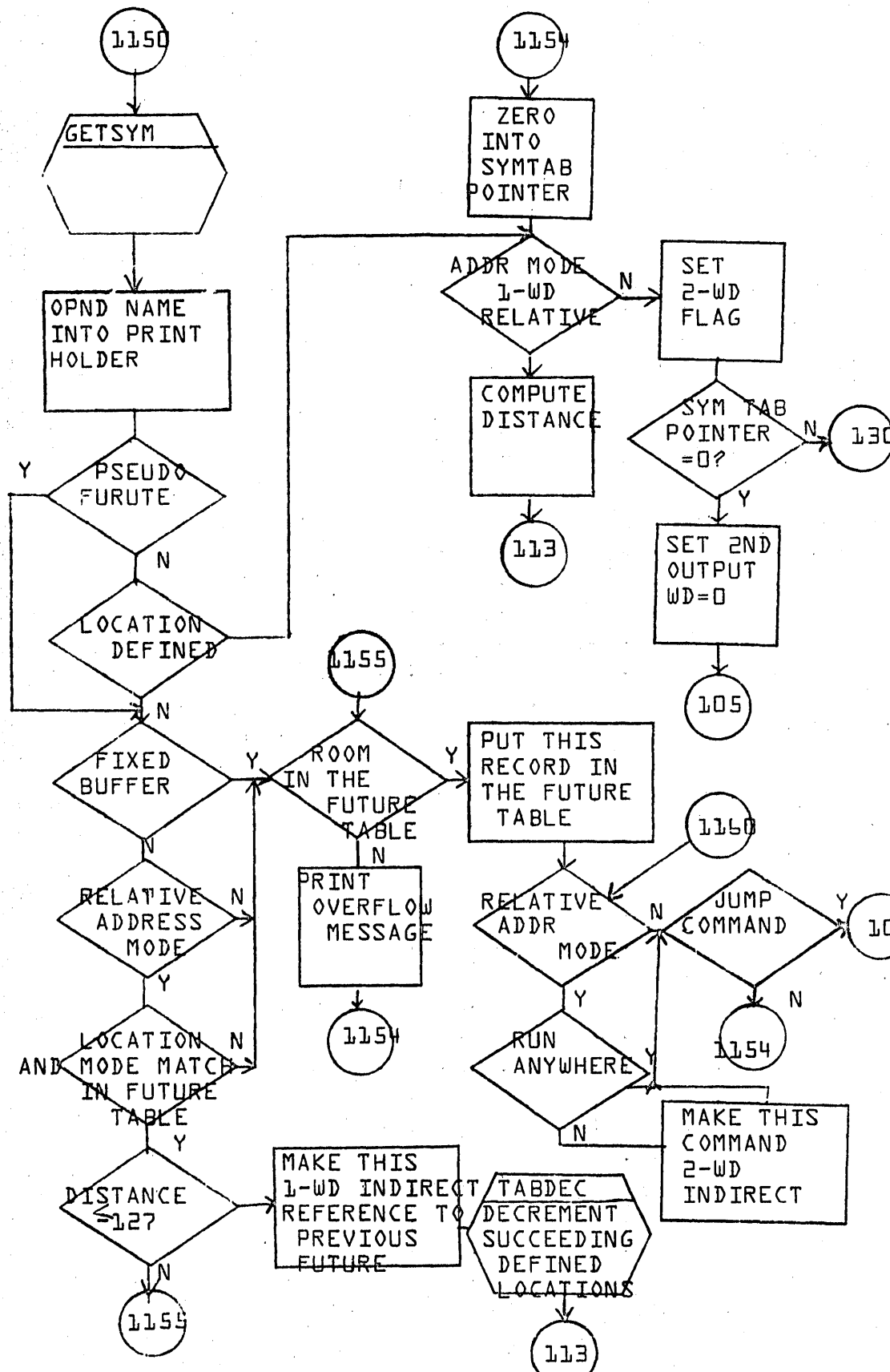


REV	APPROVED	DATE
PROJECT NO.	1700	
PROJECT MGR.		
PROJECT NAME		
TASK NO.		
TASK NAME		
DOCUMENT CLASS	IMS	
DOCUMENT TITLE	SUBROUTINE AMOUNT	
NUMBER		
DRAWN BY		
ISSUE DATE		
DATE		
CONTROL DATA CORPORATION SOFTWARE DOCUMENT <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> SAMPLE CODE <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> FLOWCHART <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> DECISION TABLE <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> OTHER <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>		

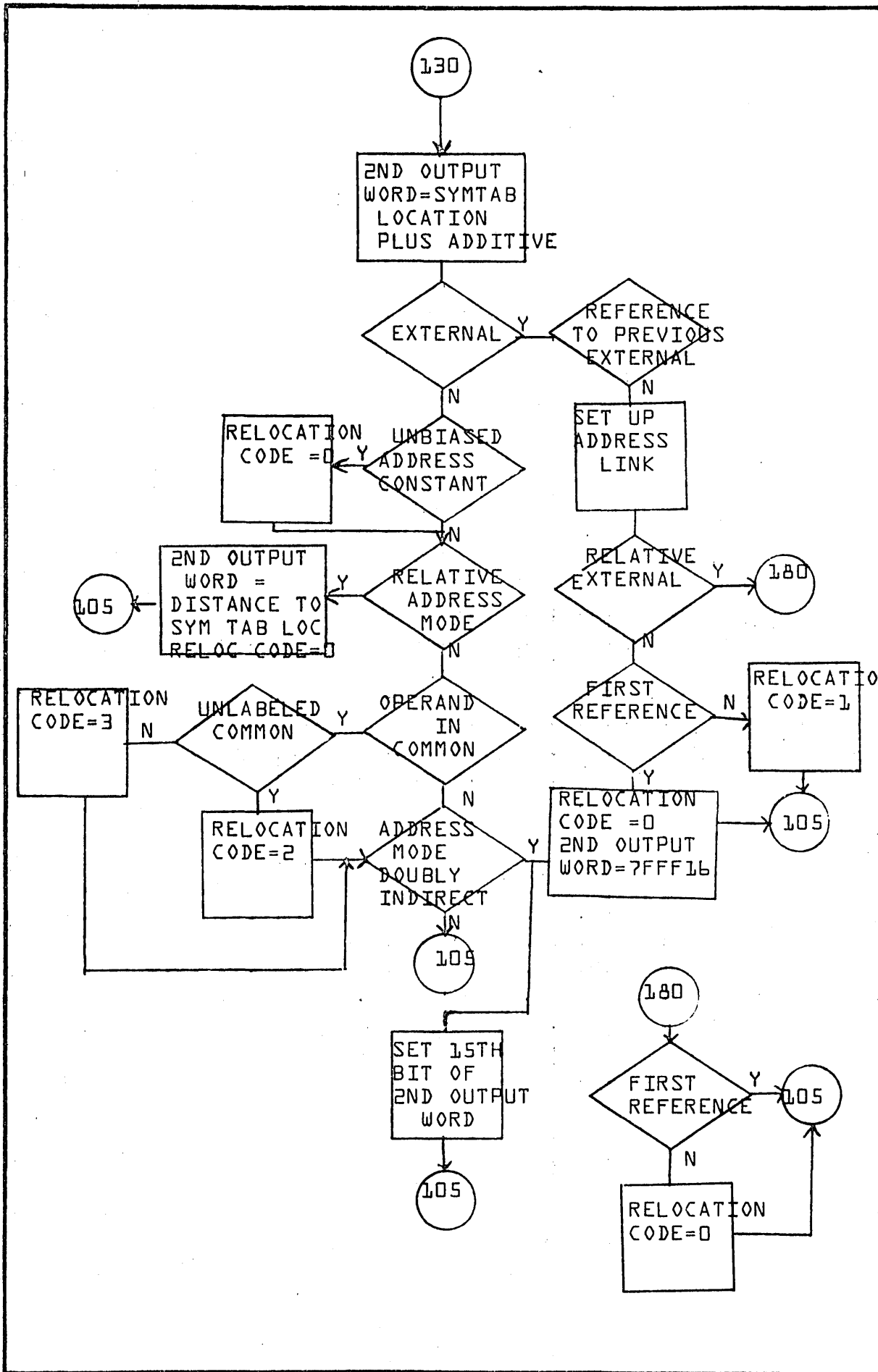
PRINTED IN U.S.A.



CONTROL DATA CORPORATION		DATE
SOFTWARE DOCUMENT	APPROVED	
SAMPLE CODE <input type="checkbox"/>	REV	
FLOWCHART <input checked="" type="checkbox"/>	PROJECT NO.	
DECISION TABLE <input type="checkbox"/>	PROJECT MGR.	
OTHER <input type="checkbox"/>	PROJECT NAME	
	TASK NO.	
	TASK NAME	
DOCUMENT CLASS	MACH. TYPE	DATE
IMS	1700	
DOCUMENT TITLE	SUBROUTINE AMOUNT	
NUMBER	PAGE 4 OF 11	
DRAWN BY	ISSUE DATE	



DATE	
APPROVED	
REV	
PROJECT NO.	
PROJECT MGR.	
PROJECT NAME	
TASK NO.	
TASK NAME	
DOCUMENT CLASS	MACH. 1700
DOCUMENT TITLE	IMS
NUMBER	1700
ISSUE DATE	
DRAWN BY	
DATE	
CONTROL DATA CORPORATION	
SOFTWARE DOCUMENT	
SAMPLE CODE	<input type="checkbox"/>
FLOWCHART	<input checked="" type="checkbox"/>
DECISION TABLE	<input type="checkbox"/>
OTHER	<input type="checkbox"/>
PAGE 5 OF 11	
SUBROUTINE AMOUNT	



REV	APPROVED	DATE
PROJECT NO.	PROJECT MGR.	PROJECT NAME
TASK NO.	TASK NAME	
DOCUMENT CLASS	MACH. TYPE	ISSUE DATE
IMS	1700	
DOCUMENT TITLE	SUBROUTINE AMOUNT	DATE
NUMBER	PAGE OF	DRAWN BY
	11	
	6	

CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE

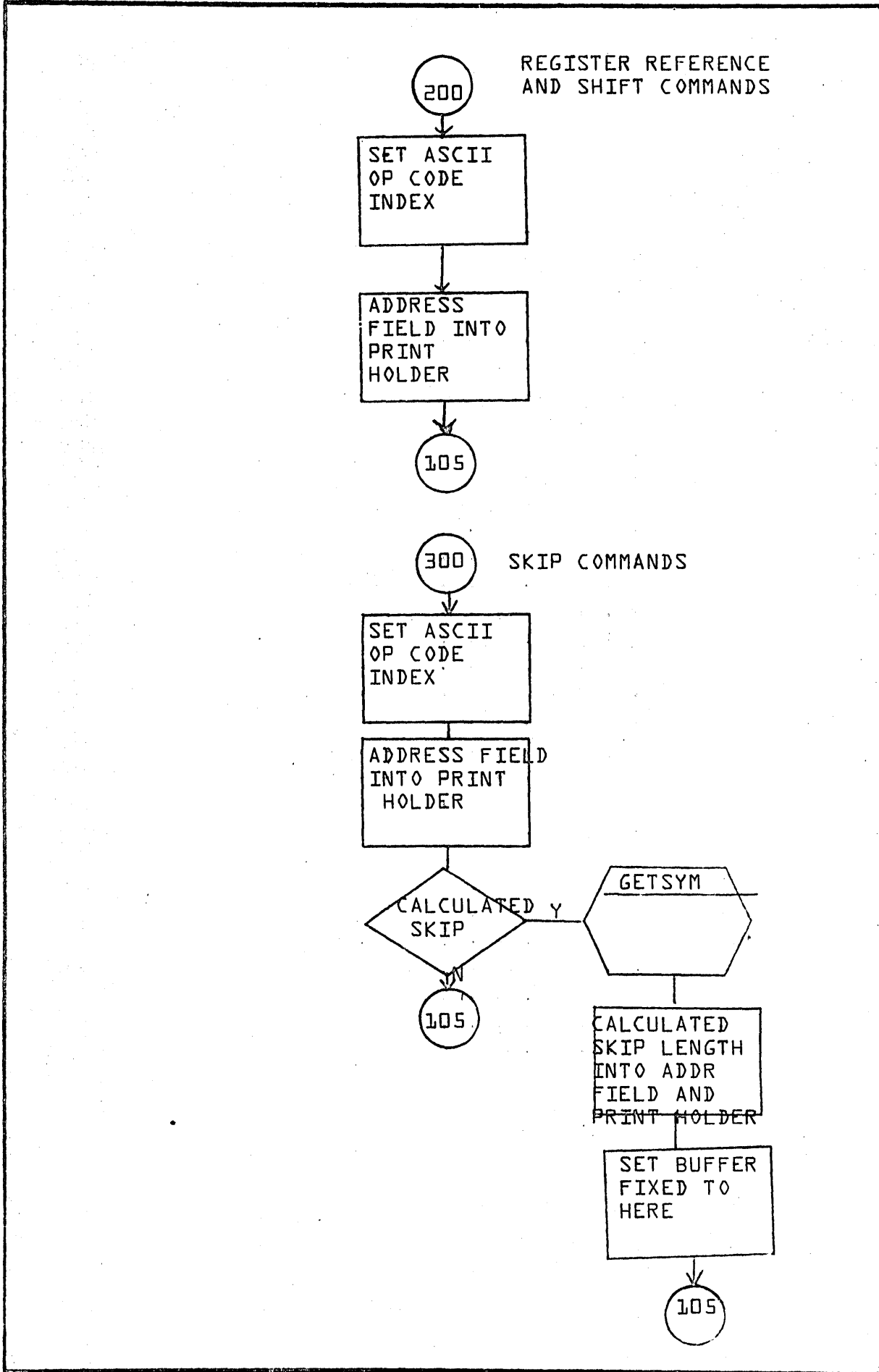
FLOWCHART

DECISION TABLE

OTHER

A B C D

1 2 3 4 5



REGISTER REFERENCE AND SHIFT COMMANDS

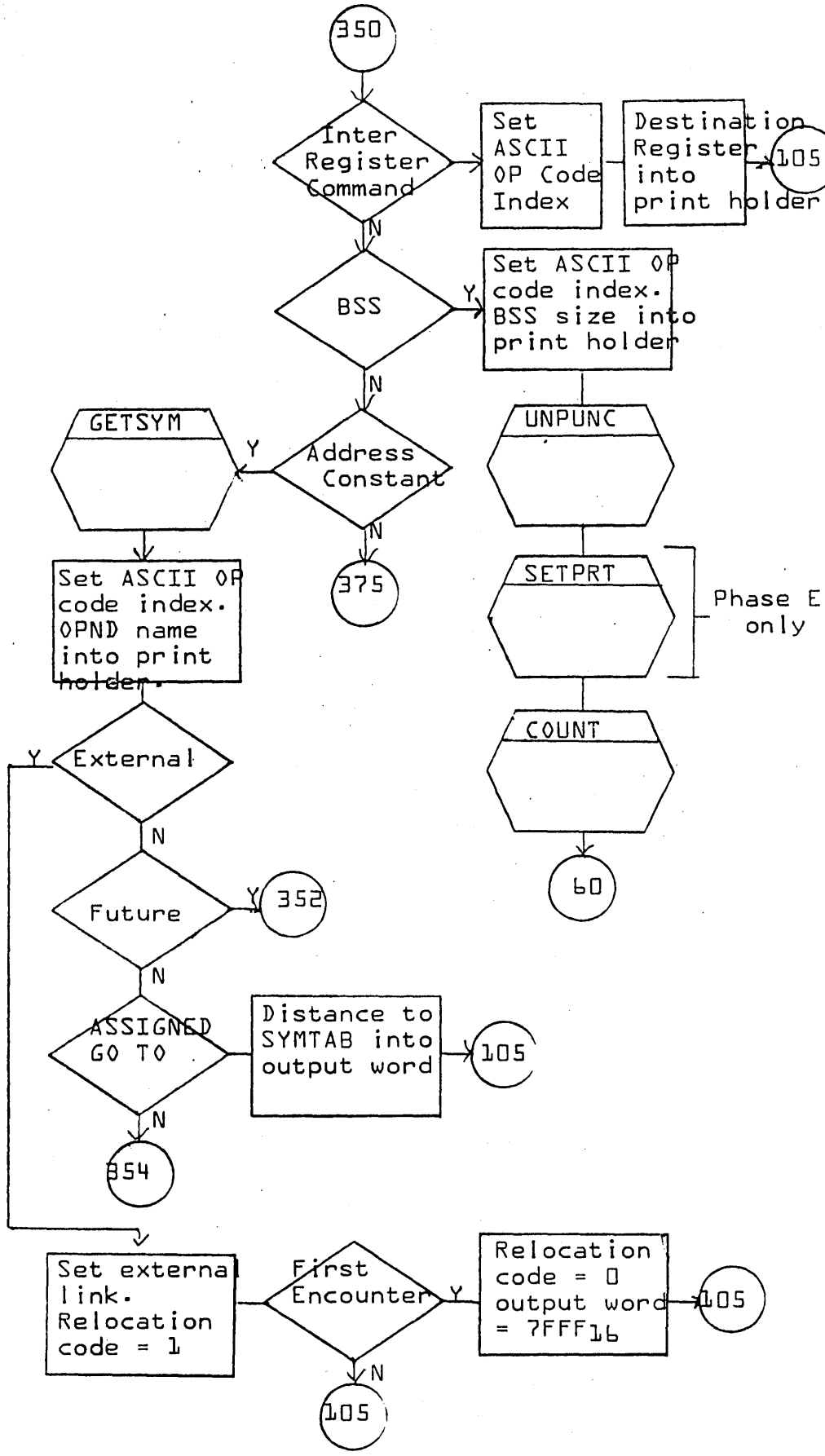
SKIP COMMANDS

DATE	APPROVED	REV
PROJECT NO.	PROJECT MGR.	PROJECT NAME
TASK NO.	TASK NAME	
DOCUMENT CLASS.	MACH. TYPE	AMOUNT
IMS	1700	
DOCUMENT TITLE	SUBROUTINE	
NUMBER	PAGE	OF
	7	11
DRAWN BY	ISSUE DATE	DATE

CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE  FLOWCHART  DECISION TABLE  OTHER





DATE	
APPROVED	
REV	
PROJECT NO.	
PROJECT MGR.	
PROJECT NAME	
TASK NO.	
TASK NAME	
DOCUMENT CLASS	IMS
DOCUMENT TITLE	MACH. TYPE 1700 Subroutine AMOUT
NUMBER	5-2.1.1
ISSUE DATE	
DRAWN BY	N. Talbot
DATE	11/1/66
TASK NAME	
CONTROL DATA CORPORATION	
SOFTWARE DOCUMENT	
SAMPLE CODE	<input type="checkbox"/>
FLOWCHART	<input type="checkbox"/>
DECISION TABLE	<input type="checkbox"/>
OTHER	<input type="checkbox"/>

A

B

C

D

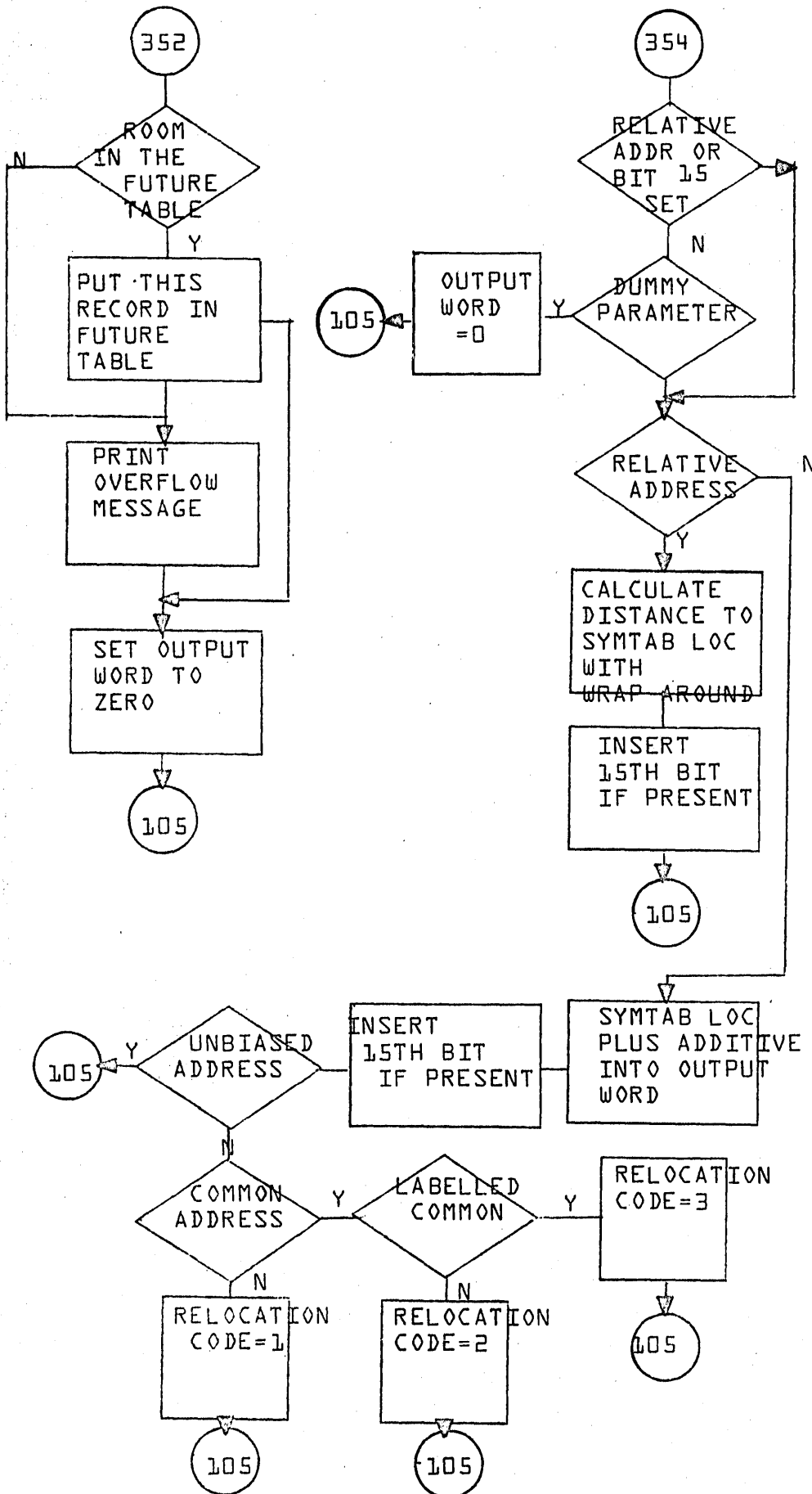
5

4

3

2

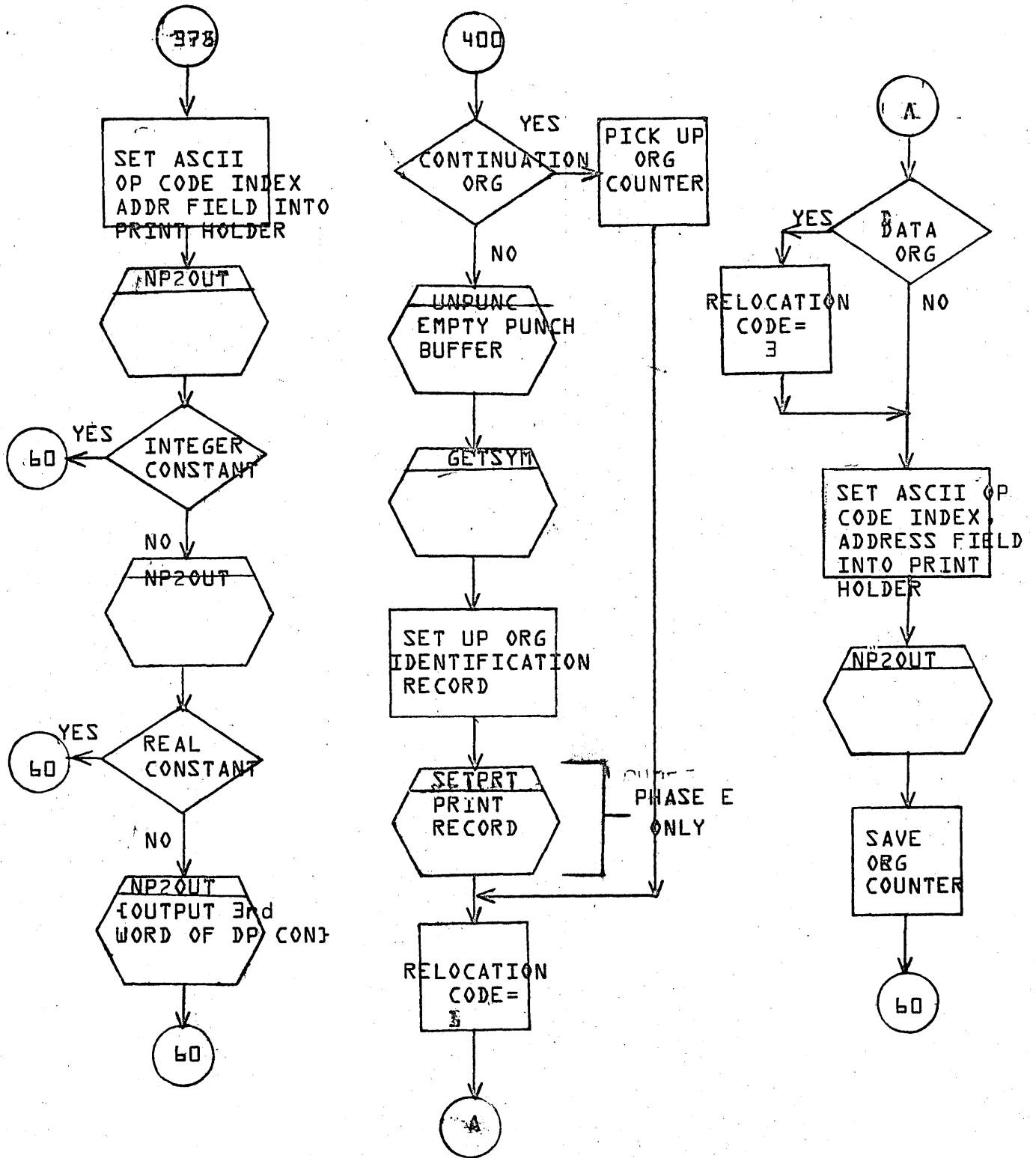
1



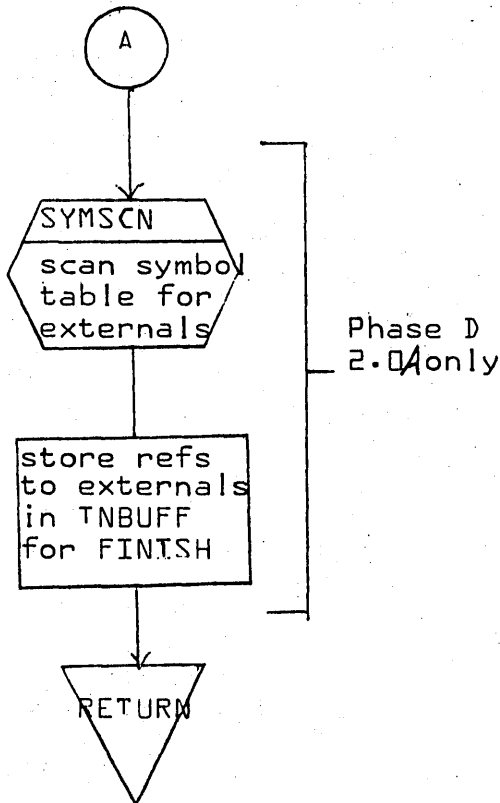
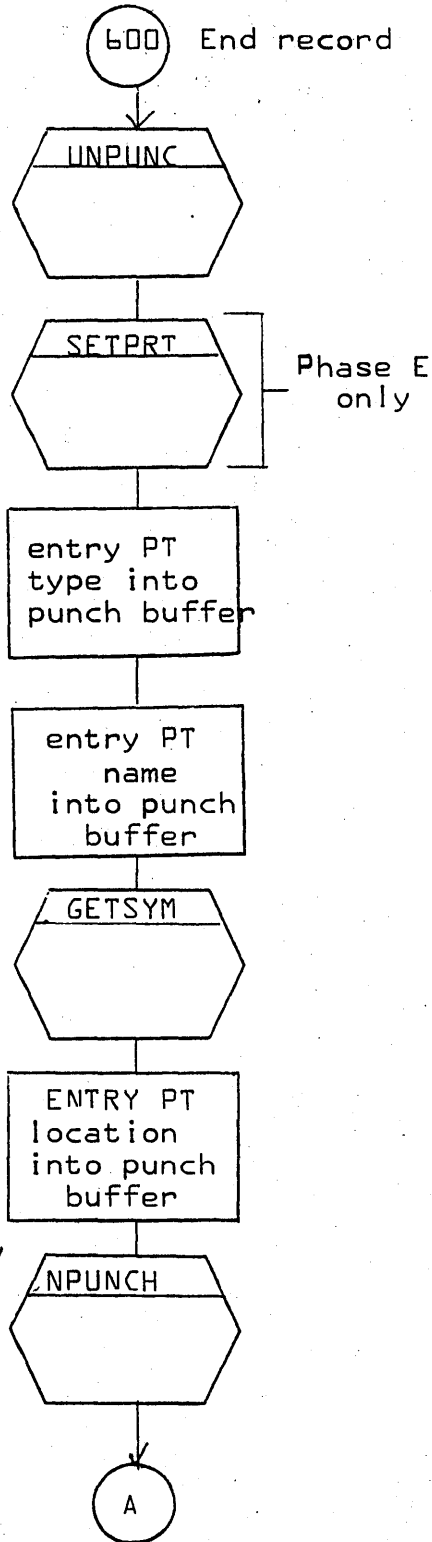
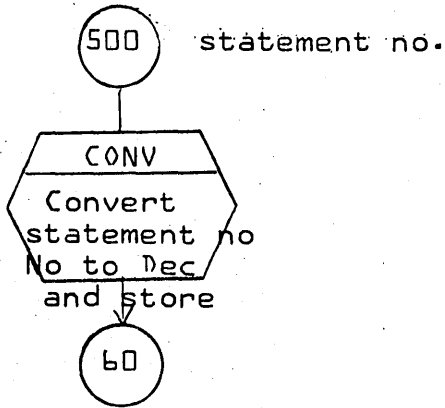
DATE	
APPROVED	
REV	
PROJECT NO.	
PROJECT MGR.	
PROJECT NAME	
TASK NO.	
TASK NAME	
MACH. TYPE	1700
SUBROUTINE AMOUNT	
PAGE	9
OF	11
ISSUE DATE	
NUMBER	
DRAWN BY	
DATE	
DOCUMENT CLASS	IMS
DOCUMENT TITLE	SUBROUTINE AMOUNT
CONTROL DATA CORPORATION	
SOFTWARE DOCUMENT	
SAMPLE CODE	<input type="checkbox"/>
FLOWCHART	<input checked="" type="checkbox"/>
DECISION TABLE	<input type="checkbox"/>
OTHER	<input type="checkbox"/>



LA JOLLA FACILITY



TITLE			SUBROUTINE AMOUT		DRG. NO.	
DRAWN BY			PROJ.		REVISION	
DATE			SHEET		10 OF 11	



DATE		PROJECT NO.	1700
APPROVED		PROJECT MGR.	
REV		PROJECT NAME	
		TASK NO.	
		TASK NAME	
		MACH. TYPE	AMS
		DOCUMENT CLASS	Subroutine AM011
		ISSUE DATE	PAGE 11 OF 11
		NUMBER	5.2.1.1
		DATE	11/26/66
		DRAWN BY	N Talbott

CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

PRINTED IN USA

A

FORMAT OF NFTAB

Symbol Table Pointer Address Additive Instruction Type Code Address of Referencing INSTR.
Symbol Table Pointer Address Additive -Instruction Type Code Address of Referencing INSTR.

Pointer to Symbol Table Entry for  
this operand  
This entry for an operand not yet  
defined

This entry for a defined operand with  
relative distance 7 255<sub>10</sub>

B

C

D

CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	FORMAT OF NFTAB			PROJECT MGR.			
	PAGE 1 OF 1			PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.					
DRAWN BY	DATE	TASK NAME					

DOCUMENT CLASS IMS PAGE NO 5-82  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

3. The program counter is not fixed, the instruction is a conditional jump and the distance to the address reference is less than  $16_{10}$ . The jump instruction is eliminated and the preceding skip or skips are changed to reflect the elimination of the jump.

The fifteenth bit of the indicator is set in a record containing an instruction to be eliminated. This will cause subroutine AMOUT to ignore the record.

A decrement count is set for instructions reduced in size or eliminated. Subroutine TABDEC is called to decrement all affected storage addresses.

#### 5.2.4.1 Flow Chart of Subroutine ADMAX

#### 5.2.5 Subroutine CONV

Subroutine CONV converts a binary word to decimal format. It is handed two parameters by the calling program. The first is the location of the word to be converted; the second is the location into which the converted word is to be stored.

#### 5.2.6 Subroutine COUNT

Subroutine COUNT increments the program counter. The increment holder, ICT, is always set at one unless re-set by the calling program.

#### 5.2.7 Subroutine FINISH

Subroutine FINISH lists the program length, referenced externals, and undefined symbols, if any. It also generates binary external records.

#### 5.2.7.1 Flow Chart of Subroutine FINISH

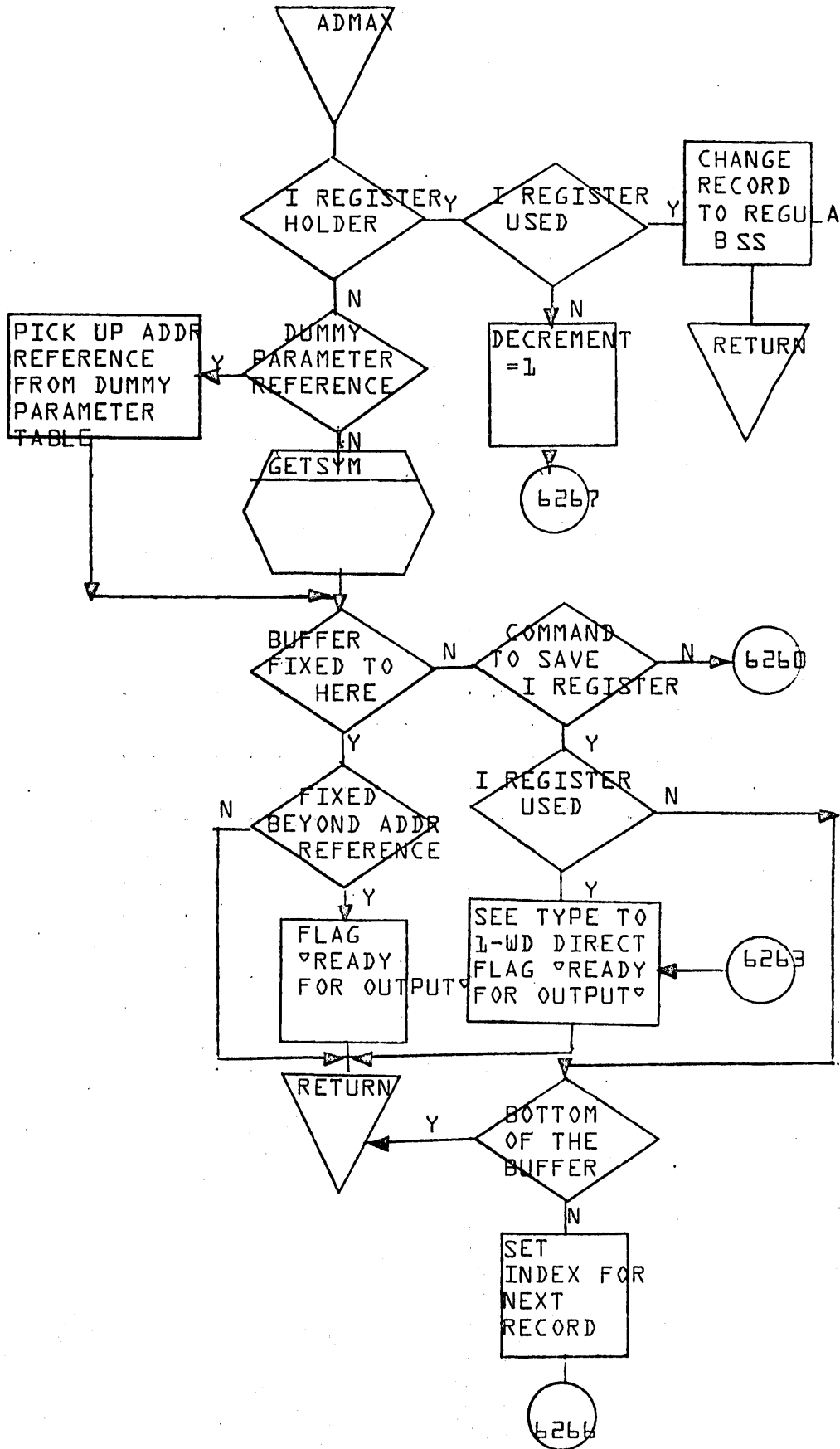
#### 5.2.8 Subroutine IACON

Subroutine IACON converts a two-word symbol name (ISYM - packed by CNVT) to ASCII format and stores it in the print buffer.

#### 5.2.9 Subroutine IHCON

Subroutine IHCON converts a binary word to hex format and stores it in the print buffer.

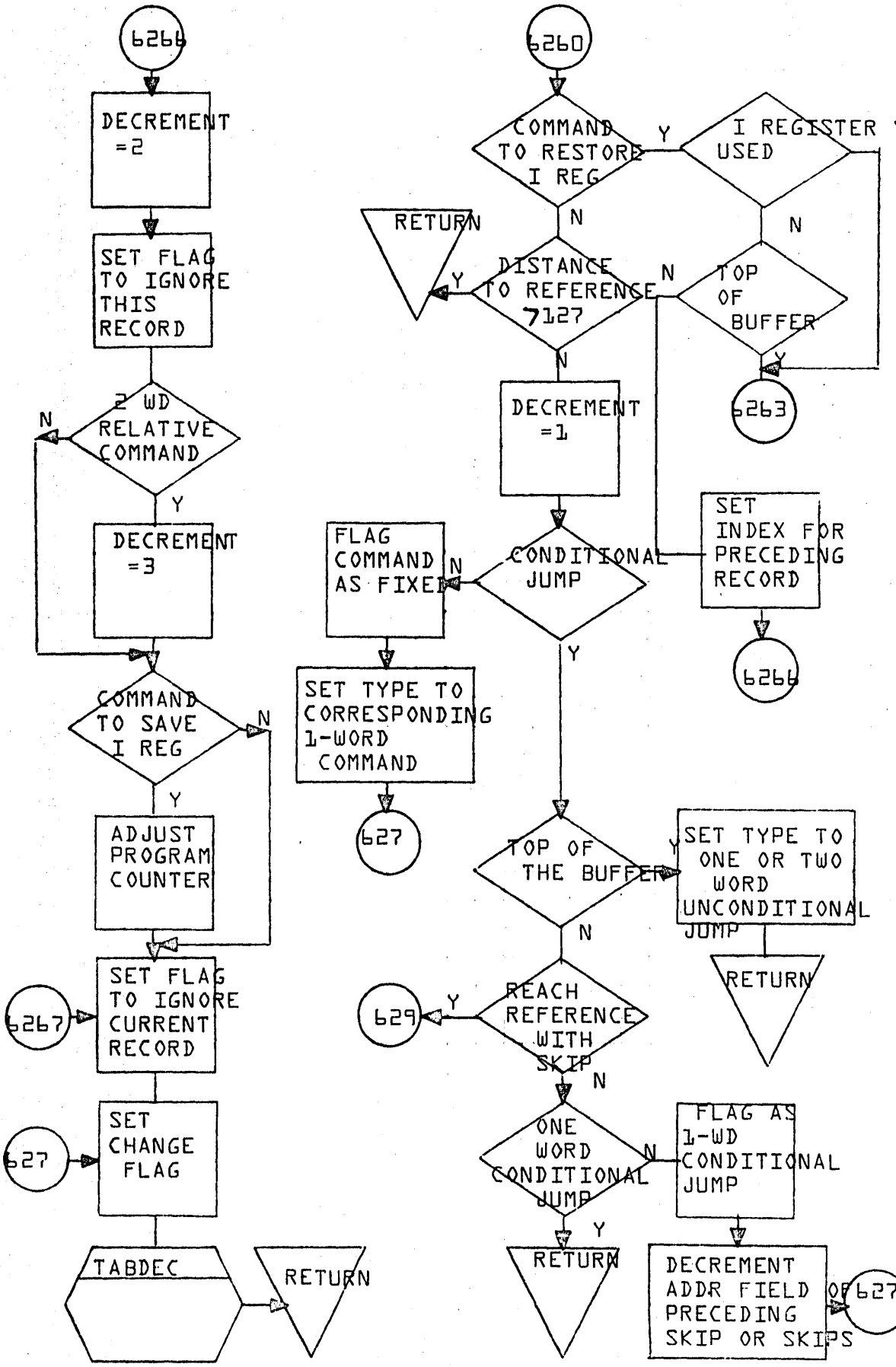
#### 5.2.10 Function INDEX (INX) converts the circular buffer pointer INX (which always increases) to a subscript of NOBUFF. That is, INDEX (INX) is equal to INX modulo INOB (NOBUFF size).



CONTROL DATA CORPORATION	DOCUMENT CLASS	IMS	MACH. TYPE	J700	PROJECT NO.		APPROVED		DATE
SOFTWARE DOCUMENT	DOCUMENT TITLE	SUBROUTINE ADMAX			PROJECT MGR.				
SAMPLE CODE	FLOWCHART	NUMBER	1 OF 3	PAGE	PROJECT NAME				
DECISION TABLE	OTHER	DRAWN BY		ISSUE DATE	TASK NO.				
				DATE	TASK NAME				

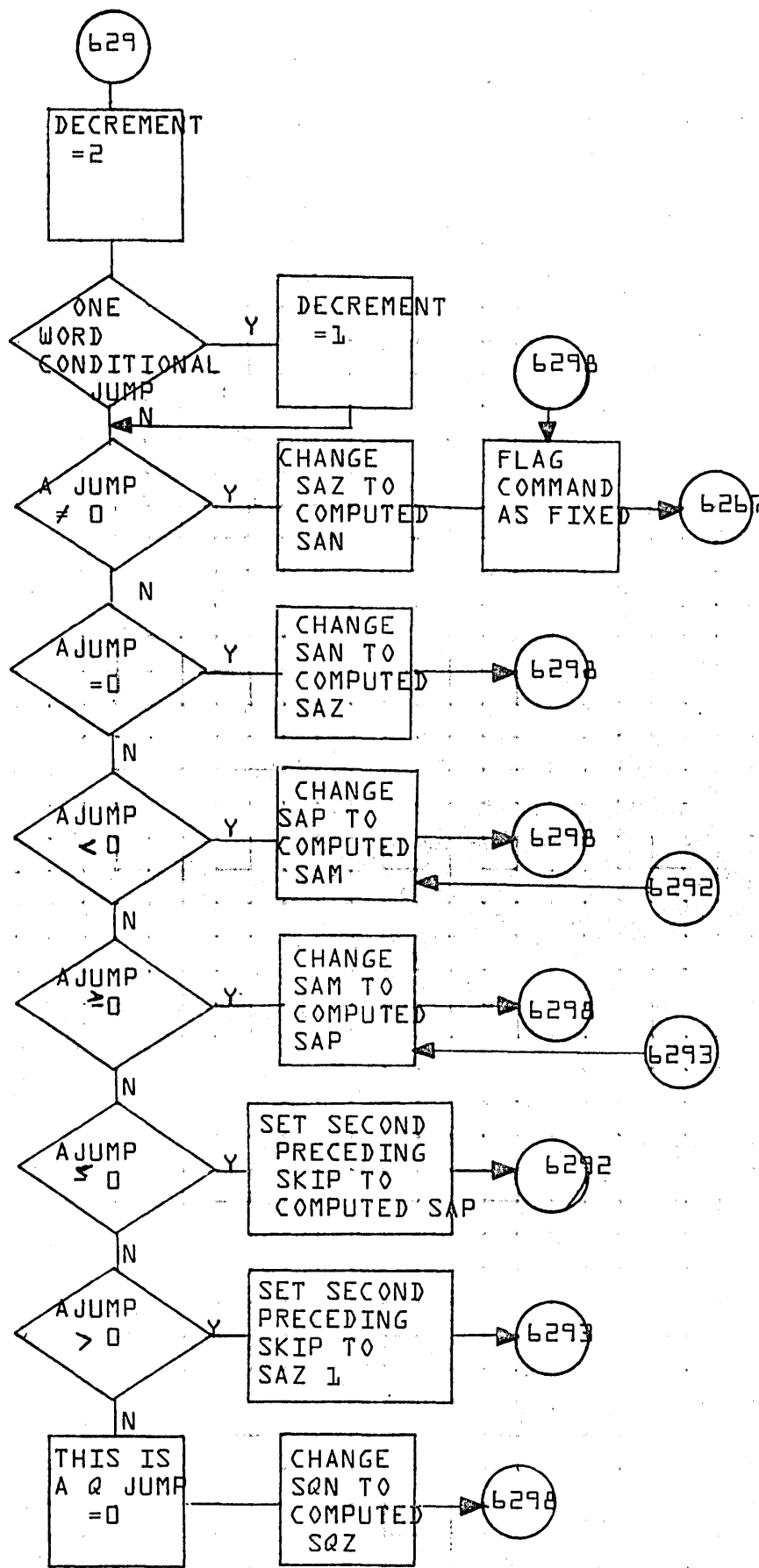
CONTROL DATA CORPORATION

1 2 3 4 5

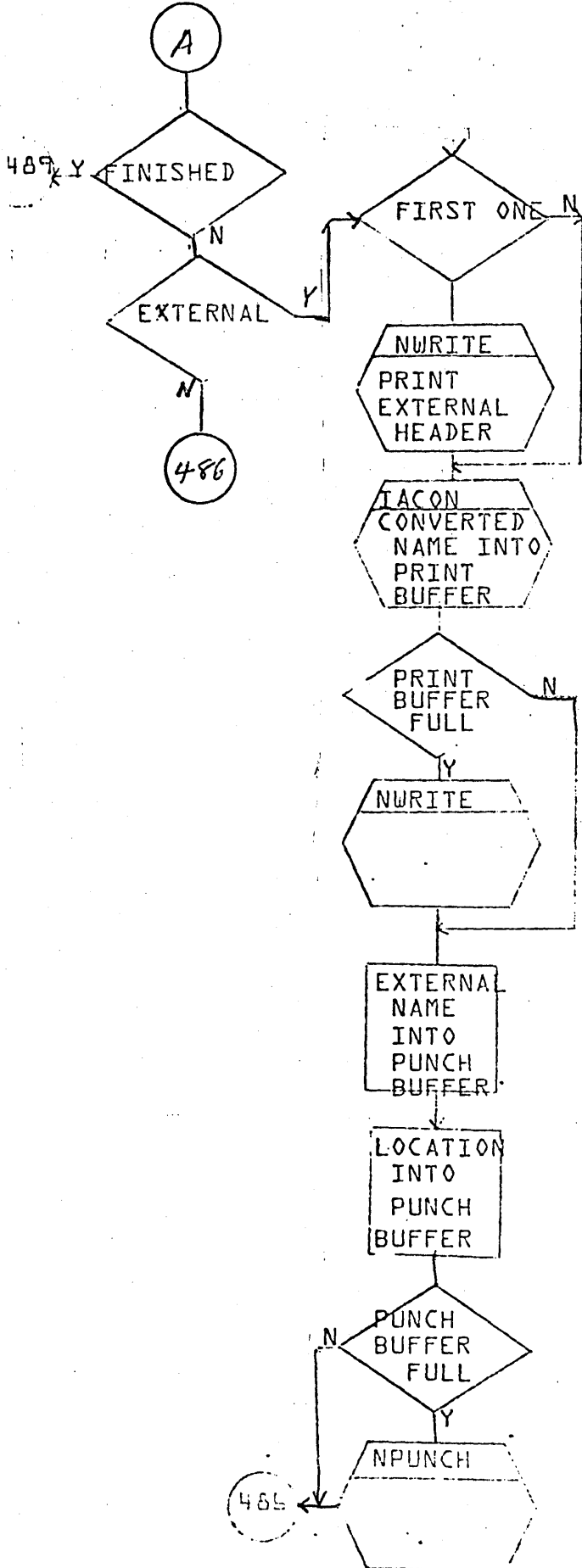
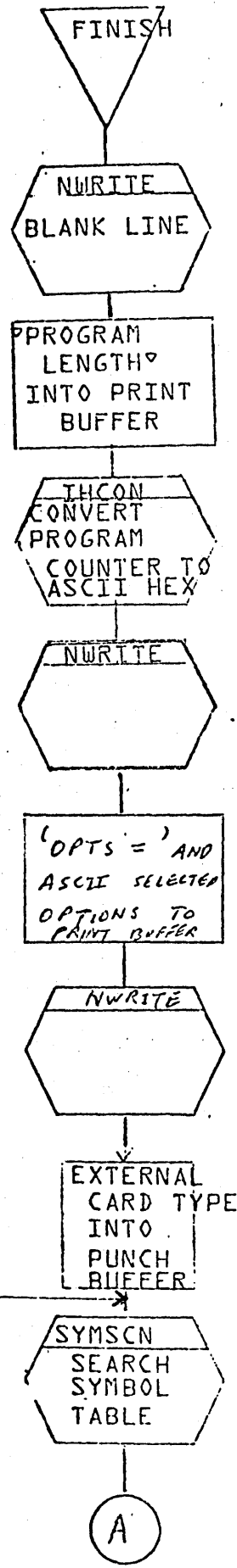


DATE		APPROVED	
REV		PROJECT NO.	
		PROJECT MGR.	
		PROJECT NAME	
		TASK NO.	
		TASK NAME	
DOCUMENT CLASS	IMS	MACH. TYPE	1700
DOCUMENT TITLE	SUBROUTINE ADMAX		
NUMBER		PAGE 2 OF	3
DRAWN BY		ISSUE DATE	
		DATE	
CONTROL DATA CORPORATION			
SOFTWARE DOCUMENT			
SAMPLE CODE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
FLOWCHART	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
DECISION TABLE	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
OTHER	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

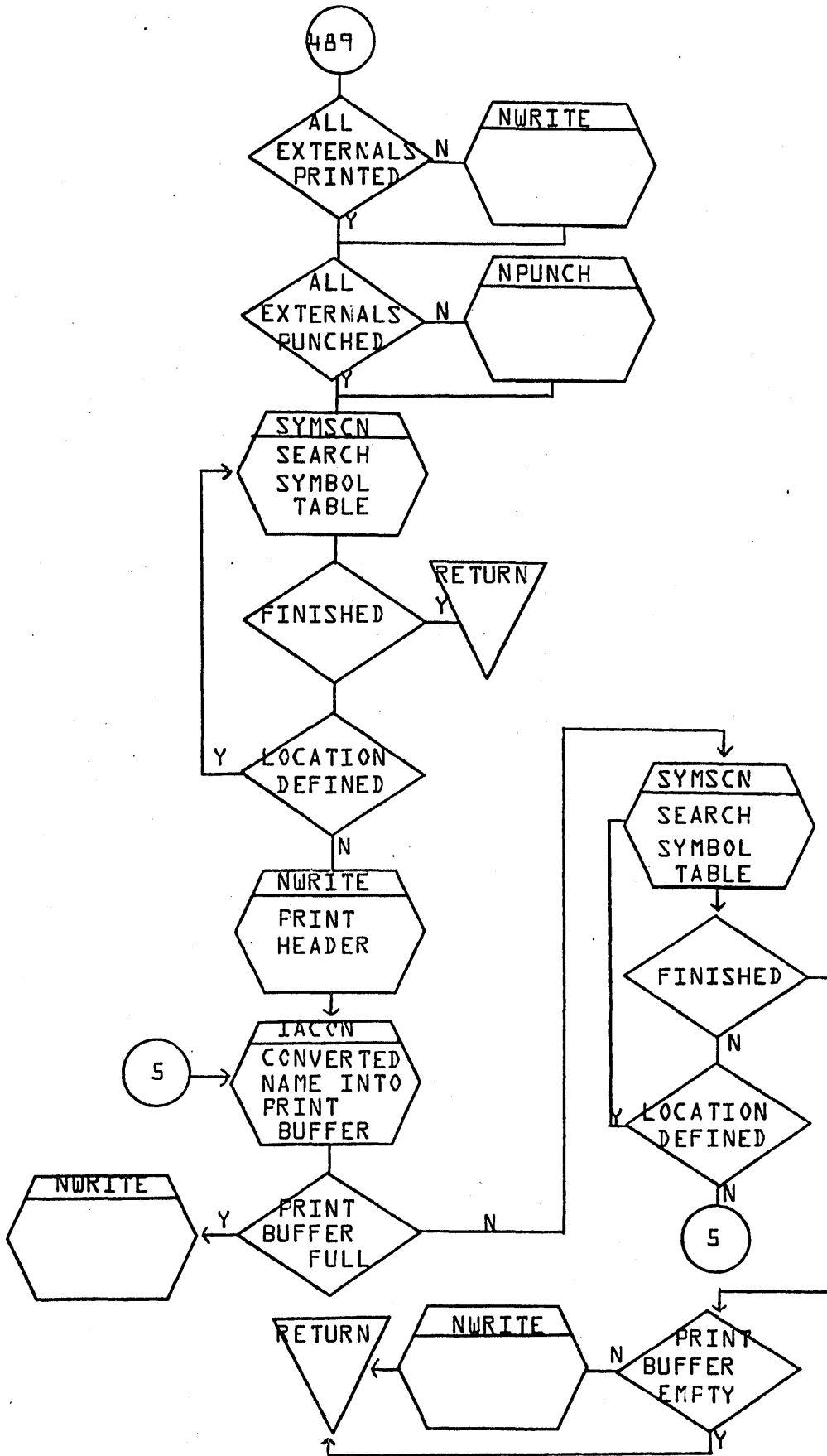




CONTROL DATA CORPORATION SOFTWARE DOCUMENT		DOCUMENT CLASS IMS	MACH. TYPE 1700	PROJECT NO.	REV	APPROVED	DATE
SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>		DOCUMENT TITLE SUBROUTINE ADMX	PAGE 3 OF 3	PROJECT MGR.			
		NUMBER	ISSUE DATE	PROJECT NAME			
DRAWN BY			DATE	TASK NO.			
				TASK NAME			



CONTROL DATA CORPORATION		DATE
SOFTWARE DOCUMENT	REV	APPROVED
SAMPLE CODE	PROJECT NO.	DATE
FLOWCHART	PROJECT MGR.	
DECISION TABLE	PROJECT NAME	
OTHER	TASK NO.	
	TASK NAME	
DOCUMENT CLASS	MACH. TYPE	PROJECT NO.
IMS	1700	
DOCUMENT TITLE	SUBROUTINE FINISH	
NUMBER	PAGE	OF
	2	2
DRAWN BY	ISSUE DATE	TASK NAME



CONTROL DATA CORPORATION		DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.		DATE	
SOFTWARE DOCUMENT		DOCUMENT TITLE	SUBROUTINE FINISH			PROJECT MGR.		REV	
SAMPLE CODE						PROJECT NAME		APPROVED	
FLOWCHART						TASK NO.			
DECISION TABLE		NUMBER	5.2.5.1	ISSUE DATE		TASK NAME			
OTHER		DRAWN BY		DATE					

A B C D

DOCUMENT CLASS IMS PAGE NO 5-88  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

### 5.2.11 Subroutine LABOUT

Subroutine LABOUT puts labels in the print buffer by calling subroutines IACON or IHCON as applicable. It also checks the future table, NFTAB, for references to this label. If any references are found, LABIN outputs a binary record originated at the location of the instruction referencing the label. The binary record contains the location of the label or the distance from the label to the reference depending on the addressing mode of the instruction.

#### 5.2.11.1 Flow Chart of Subroutine LABOUT

### 5.2.12 Subroutine NPUNCH

Subroutine NPUNCH outputs binary records from the punch buffer and clears the buffer.

### 5.2.13 Subroutine NP2OUT

Subroutine NP2OUT sends one or two output images to the binary output routine and to the list output routine (Phase E only) according to the size of the instruction. The program counter is bumped for each image that is output.

#### 5.2.13.1 Flow Chart of Subroutine NP2OUT

### 5.2.14 Subroutine NWRITE

Subroutine NWRITE outputs one list record from the print buffer and sets the buffer to ASCII blanks.

### 5.2.15 Subroutine PACK

Subroutine PACK is called from subroutine NWRITE to pack the output image in the print buffer.

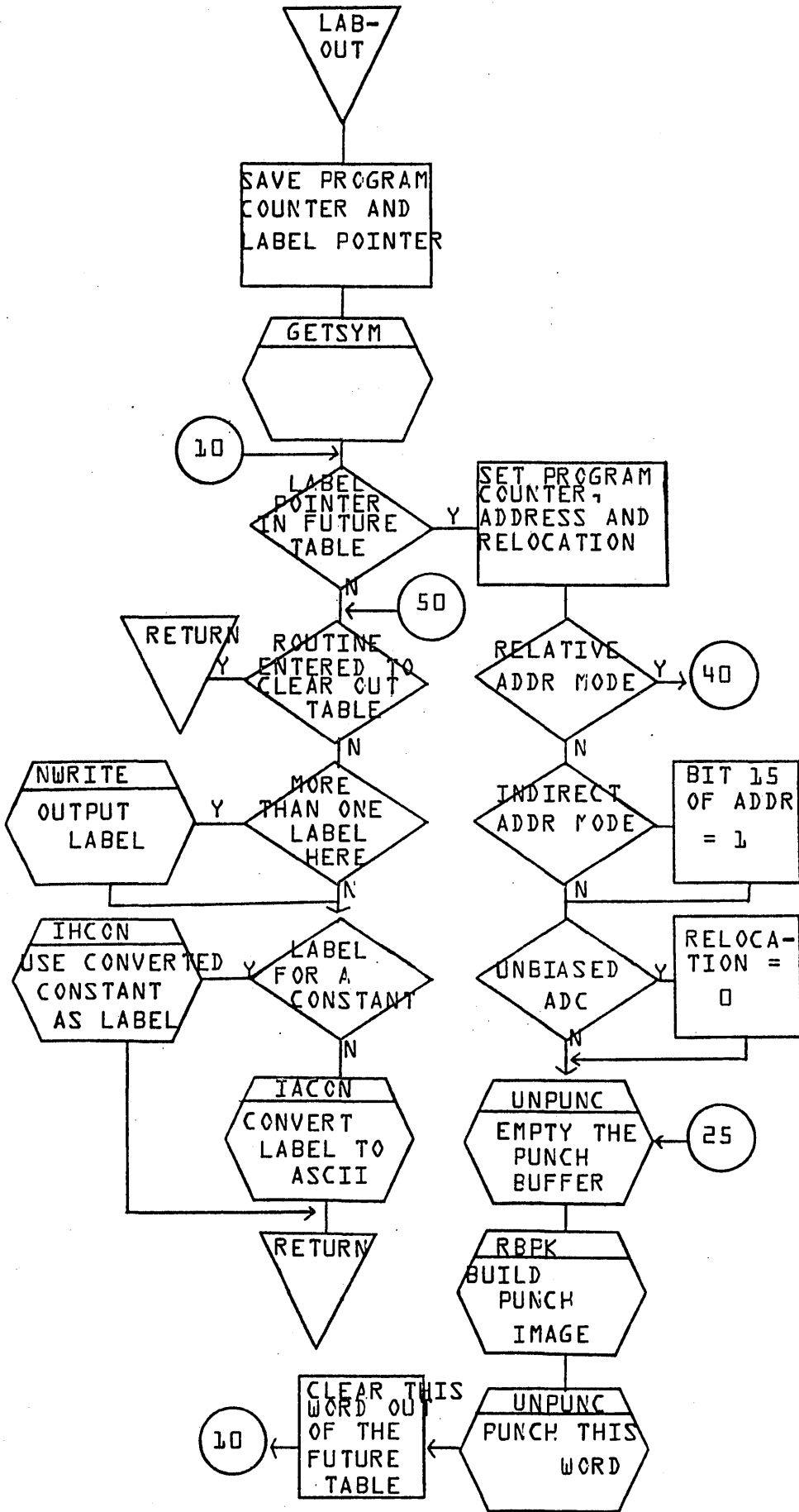
### 5.2.16 Subroutine RBDX

Subroutine RBDX is called from subroutine RBPK to compute the location and position of a binary word and its associated relocation byte in the binary output record. RBDX is called from subroutine UNPUNC to compute the location and position of the "last word flag" in the binary output record.

#### 5.2.16.1 Flow Chart of Subroutine RBDX

### 5.2.17 Subroutine RBPK

Subroutine RBPK packs one binary word into the punch buffer. When the buffer is full, subroutine UNPUNC is called to empty the buffer.



CONTROL DATA CORPORATION SOFTWARE DOCUMENT		DOCUMENT CLASS	IMS	MACH. TYPE	L700	PROJECT NO.	APPROVED	DATE
SAMPLE CODE <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>		DOCUMENT TITLE	SUBROUTINE LABOUT		PAGE 1 of 2	PROJECT MGR.		
FLOWCHART <input type="checkbox"/>		NUMBER	5.2.8.1	ISSUE DATE		PROJECT NAME		
DECISION TABLE <input type="checkbox"/>		DRAWN BY		DATE		TASK NO.		
OTHER <input type="checkbox"/>						TASK NAME		

A

B

C

D

5

4

3

2

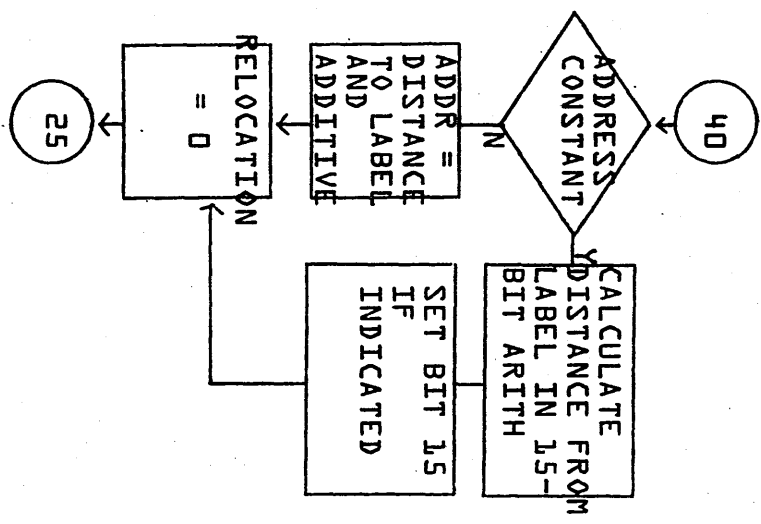
1

A

B

C

D

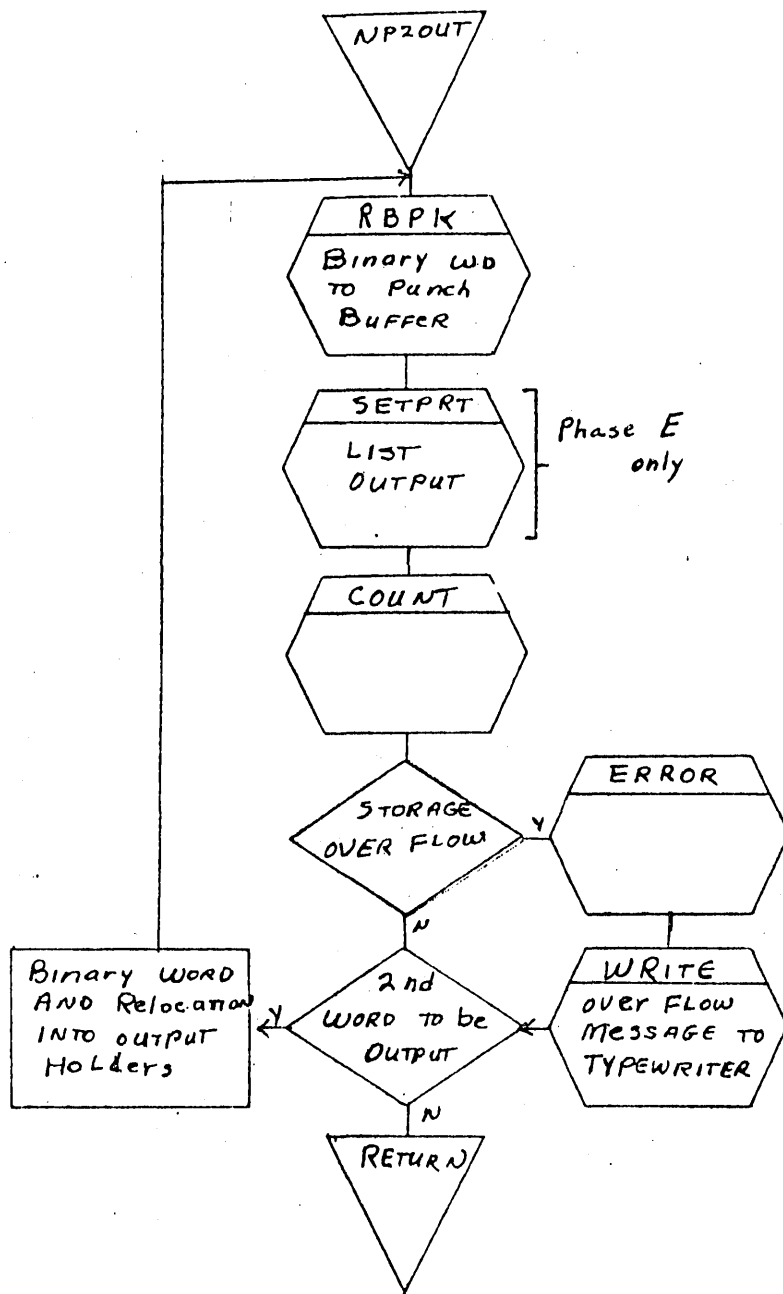


**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	SUBROUTINE LABOUT			PROJECT MGR.			
PAGE 2 OF 2				PROJECT NAME			
NUMBER	5.2.8.1	ISSUE DATE		TASK NO.			
DRAWN BY		DATE		TASK NAME			

5-90



REV	APPROVED	DATE

PROJECT NO.	PROJECT MGR	PROJECT NAME
TASK NO.	TASK NAME	

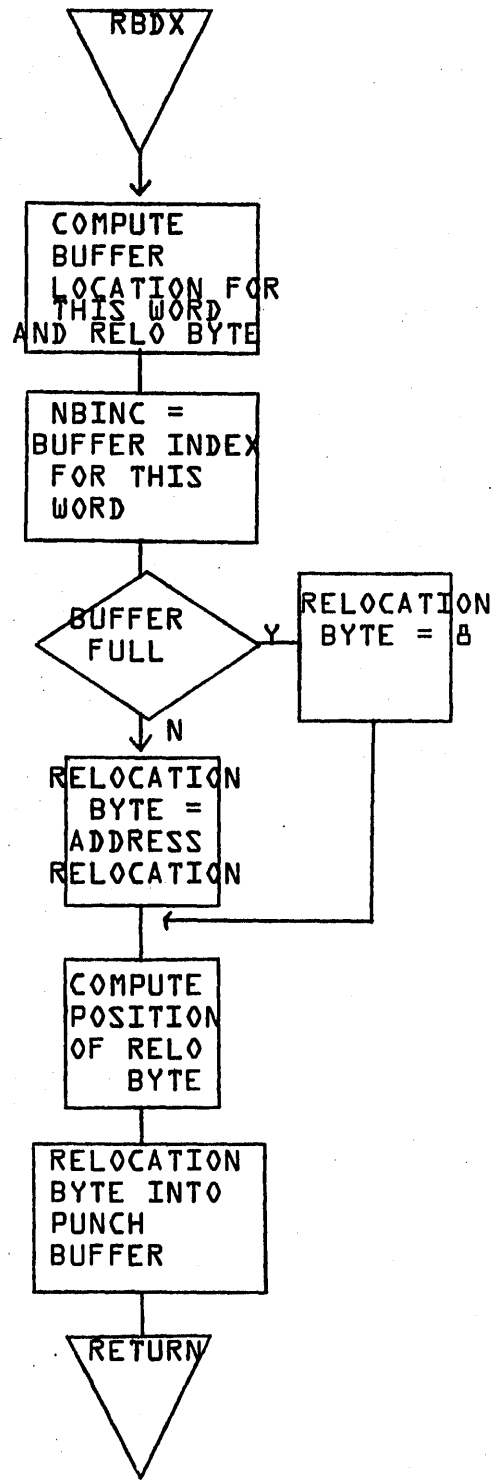
  

DOCUMENT CLASS	MACH. TYPE	1700
DOCUMENT TITLE	SUBROUTINE NP2OUT	
NUMBER	5.2.10.1	ISSUE DATE
DATE	11/9/66	

CONTROL DATA CORPORATION	<input type="checkbox"/>
SOFTWARE DOCUMENT	<input type="checkbox"/>
SAMPLE CODE	<input type="checkbox"/>
FLOWCHART	<input type="checkbox"/>
DECISION TABLE	<input type="checkbox"/>
OTHER	<input type="checkbox"/>

1 2 4



DATE	APPROVED	REV	PROJECT NO.	MACH. TYPE	1700	ISSUE DATE	DATE
			PROJECT MGR.	DOCUMENT CLASS	IMS	NUMBER	5.2.13.1
			PROJECT NAME	DOCUMENT TITLE	SUBROUTINE RBDX	PAGE	1 OF 1
			TASK NO.	DRAWN BY			
			TASK NAME				

CONTROL DATA CORPORATION  
 SOFTWARE DOCUMENT  
 SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER



DOCUMENT CLASS IMS PAGE NO. 5-93  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

5.2.17.1 Flow Chart of Subroutine RBPK

5.2.18 Subroutine SETPRT

Subroutine SETPRT constructs the opcode and address fields of a list output record and calls Subroutine NWRITE to output the record. SETPRT is in Phase E only.

5.2.18.1 Flow Chart of Subroutine SETPRT

5.2.19 Subroutine TABDEC

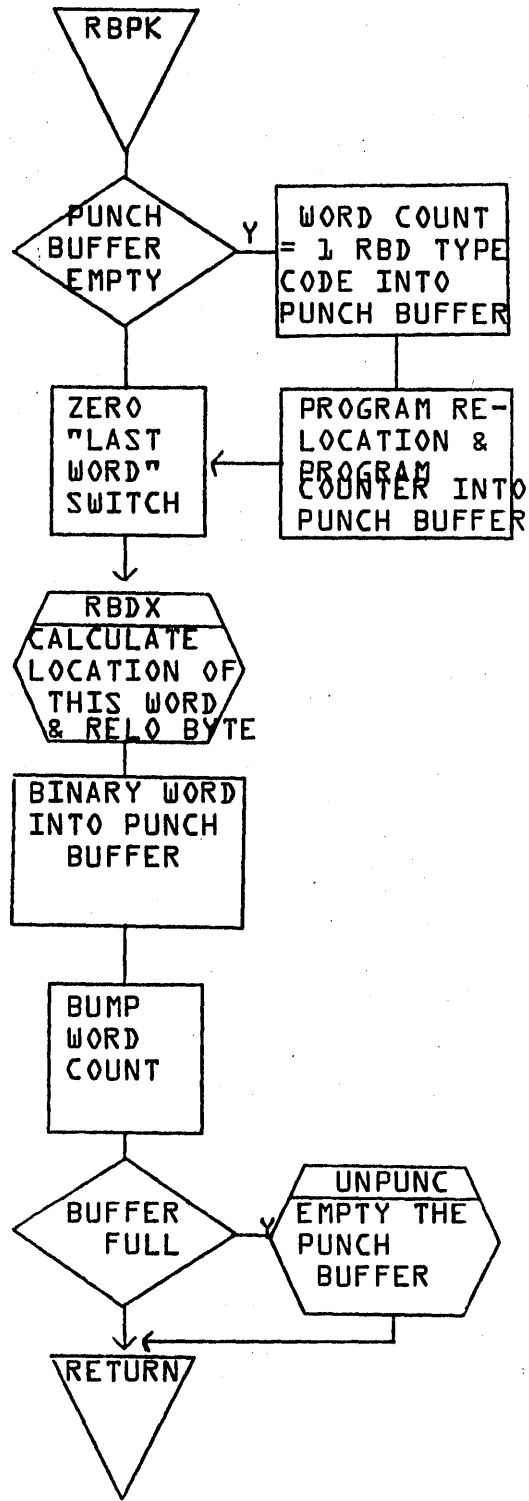
Subroutine TABDEC is called from subroutine ADMAX to decrement addresses that are affected by a change in the output buffer made in ADMAX. The symbol table, and the two tables of address holders, IDVTAB and NDSTAB, are searched for program addresses that are greater than the current program count. All such addresses found are decremented by the amount handed from ADMAX to TABDEC in the decrement holder, IDECR.

5.2.19.1 Flow Chart of Subroutine TABDEC

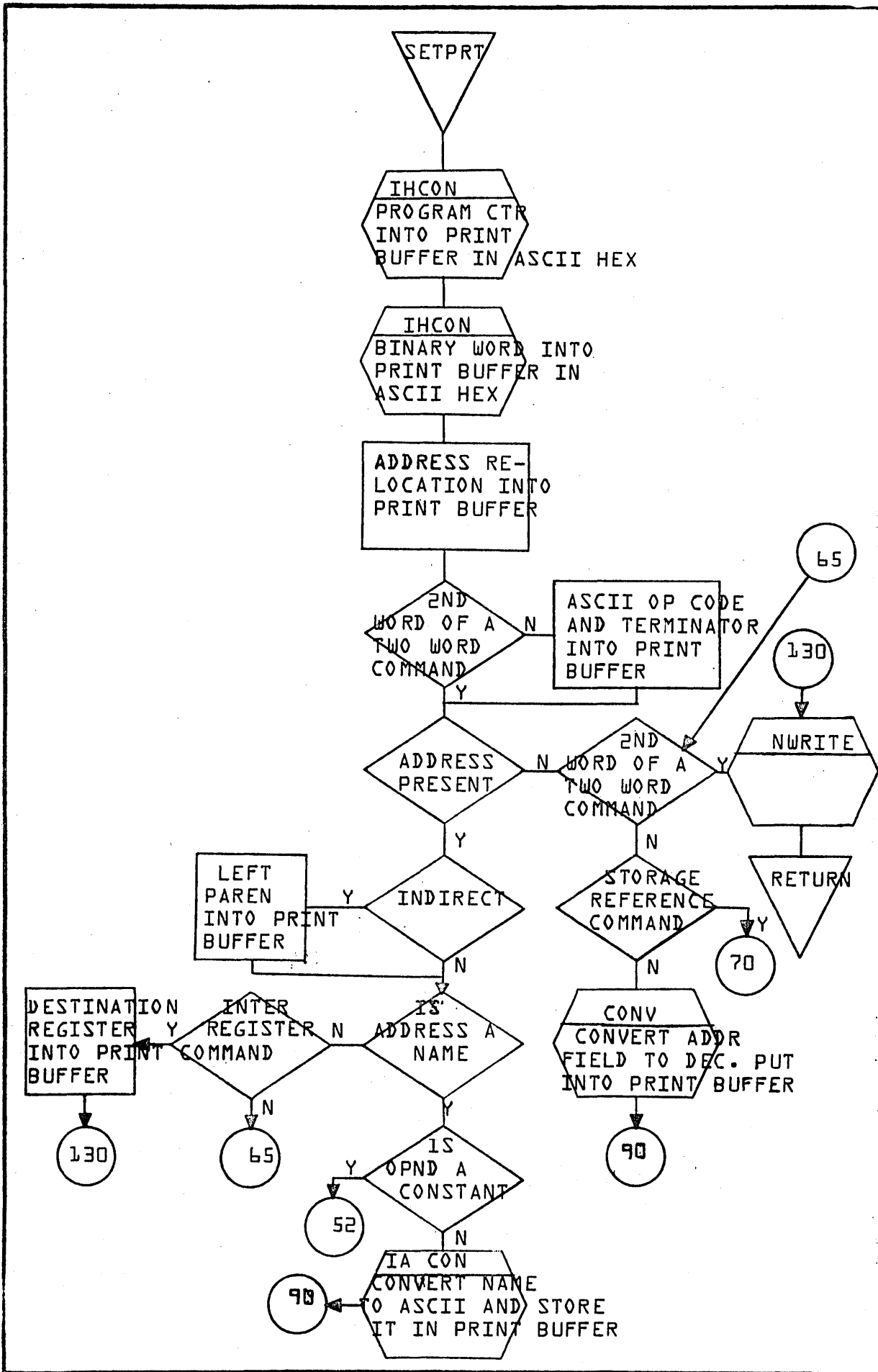
5.2.20 Subroutine UNPUNC

Subroutine UNPUNC calls subroutines RBDX and NPUNCH to output the contents of the punch buffer.

5.2.20.1 Flow Chart of Subroutine UNPUNC



CONTROL DATA CORPORATION SOFTWARE DOCUMENT		DOCUMENT CLASS	IMS	MACH. TYPE	J700	PROJECT NO.		REV		APPROVED		DATE	
SAMPLE CODE <input type="checkbox"/>		DOCUMENT TITLE	SUBROUTINE RBPK		PAGE	1	PROJECT MGR.						
FLOWCHART <input type="checkbox"/>		NUMBER	5.2.14.1	ISSUE DATE		PROJECT NAME							
DECISION TABLE <input type="checkbox"/>		DRAWN BY		DATE		TASK NO.							
OTHER <input type="checkbox"/>						TASK NAME							



DATE	
APPROVED	
REV	
PROJECT NO.	
PROJECT MGR.	
PROJECT NAME	
TASK NO.	
TASK NAME	
MACH. TYPE	1700
DOCUMENT CLASS	IMS
DOCUMENT TITLE	SUBROUTINE SETPRT
PAGE	1 of 2
ISSUE DATE	
NUMBER	
DRAWN BY	
DATE	

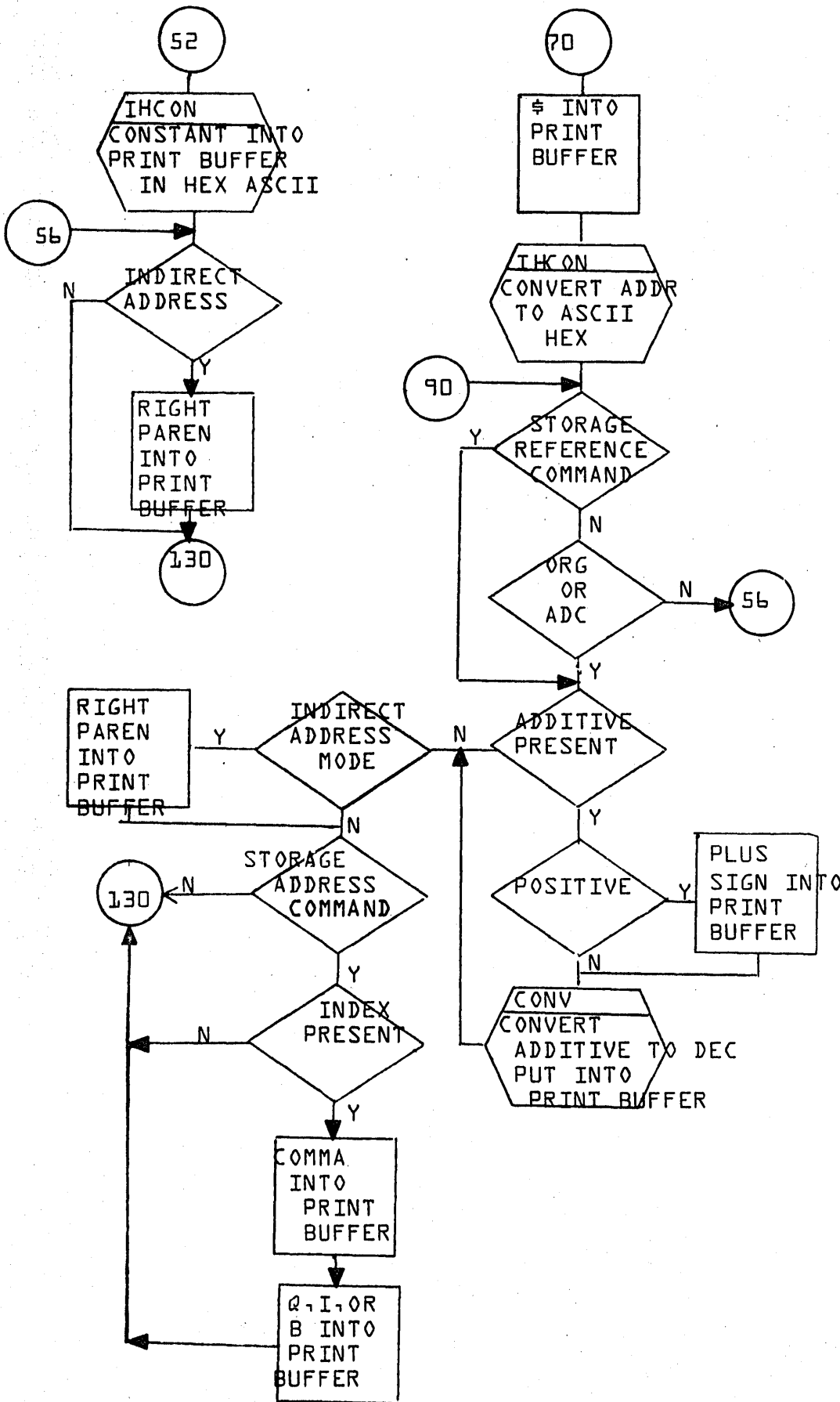
CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE

FLOWCHART

DECISION TABLE

OTHER



DATE	
APPROVED	
REV	
PROJECT NO.	1700
PROJECT MGR.	
PROJECT NAME	SUBROUTINE SETPRT
TASK NO.	2
TASK NAME	
ISSUE DATE	
DATE	
DRAWN BY	
DOCUMENT CLASS	IMS
MACH. TYPE	1700
DOCUMENT TITLE	SUBROUTINE SETPRT
NUMBER	2 OF 2

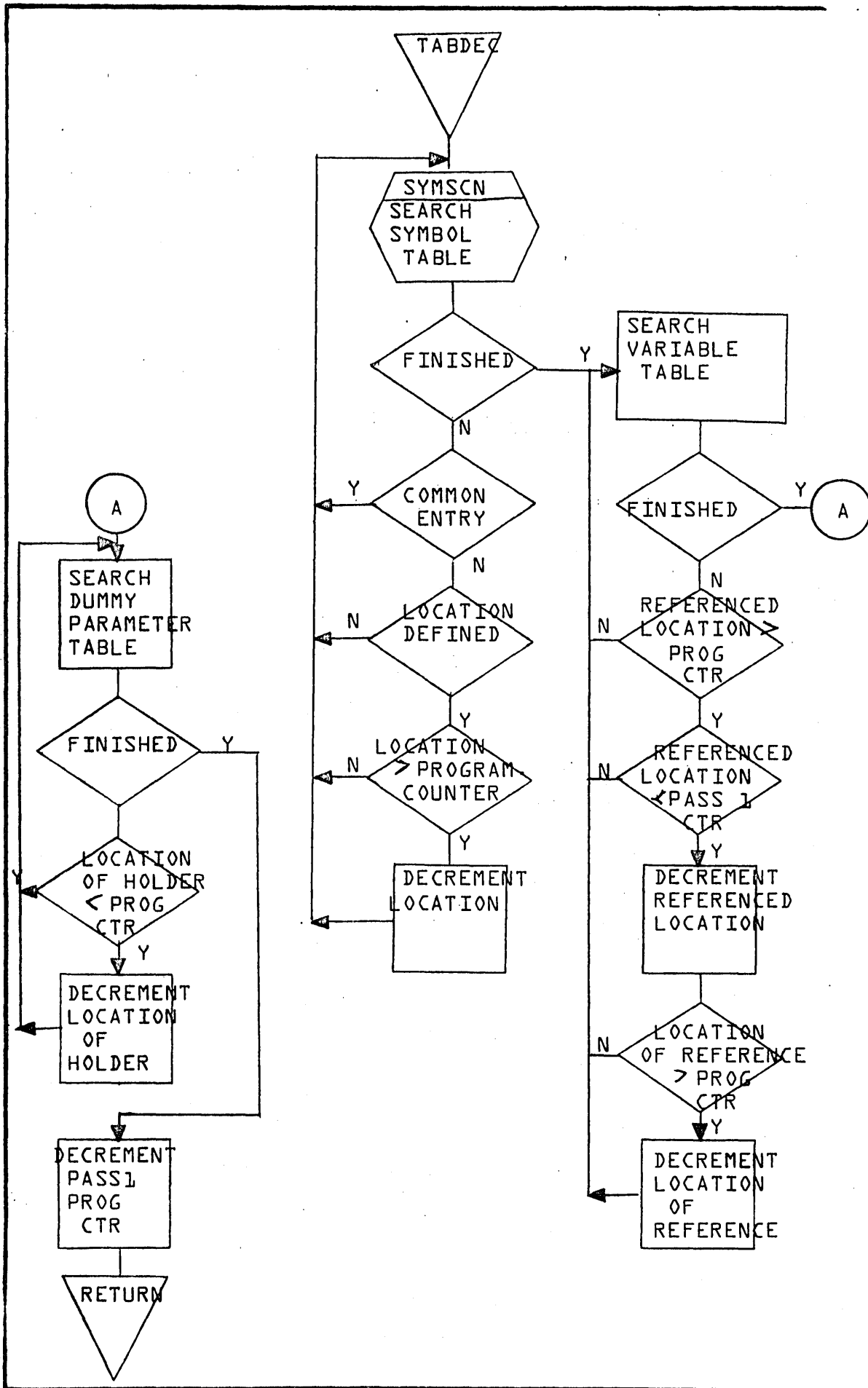
CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE

FLOWCHART

DECISION TABLE

OTHER



DATE	APPROVED	REV	PROJECT NO.	MACH. TYPE	DOCUMENT CLASS	DOCUMENT TITLE	ISSUE DATE	NUMBER	DRAWN BY	CONTROL DATA CORPORATION
			1700	IMS	SOFTWARE DOCUMENT	SUBROUTINE TABDEC	PAGE 1 OF 1			SAMPLE CODE <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> FLOWCHART DECISION TABLE OTHER
			PROJECT MGR.							
			TASK NO.							
			PROJECT NAME							
			TASK NAME							

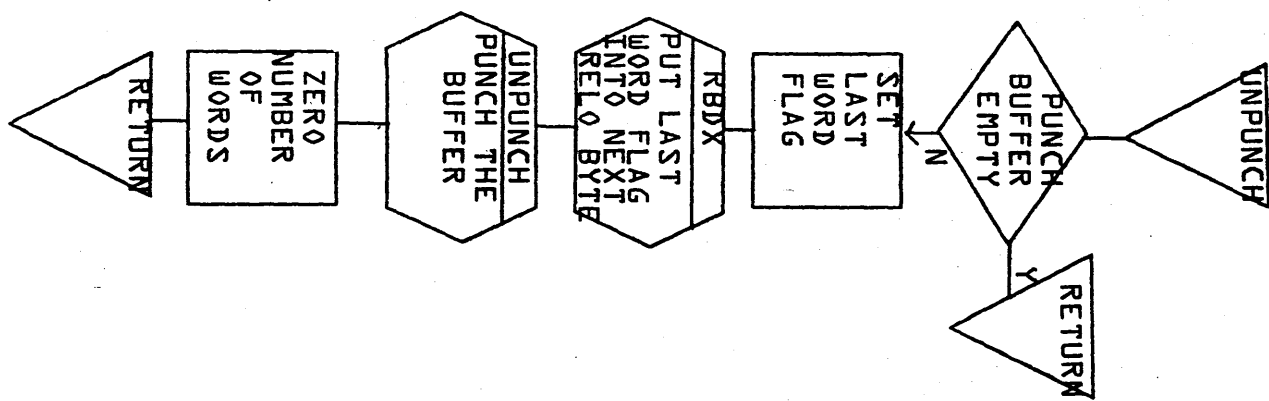
A B C D

A

B

C

D


**CONTROL DATA CORPORATION**  
 SOFTWARE DOCUMENT

 SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER 

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	SUBROUTINE UNPUNC			PROJECT MGR.			
		PAGE	1 OF 1	PROJECT NAME			
NUMBER	5.2.17.1	ISSUE DATE		TASK NO.			
DRAWN BY		DATE		TASK NAME			

5-98

DOCUMENT CLASS IMS PAGE NO. 6-1  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

CHAPTER 6

T A B L E O F C O N T E N T S

6.0 OBJECT TIME I/O ROUTINES

6.1 General Description

6.2 Calling Sequence for I/O Requests

6.2.1 Three Request I/O

6.2.2 One Request I/O

6.2.3 Auxiliary Input/Output Statements Calling Sequence

6.2.4 Basic External Functions

6.2.5 Special Calls

6.3 Discussion by Statement Type

6.3.1 READ and WRITE

6.3.2 Auxiliary Input/Output Statements

6.3.3 Basic External Functions

6.3.4 Special Calls

6.4 Discussion by Routine

6.4.1 QBQINI Routine

6.4.2 QBQX Routine

6.4.3 QBQUNI Routine

6.4.4 QBCMP Routine

6.4.5 QBRWBU Routine

6.4.6 QBFGET Routine

6.4.7 QBERRM Routine

6.4.8 QBMAGT Routine

6.4.9 QBIFRM Routine

6.4.10 QBFS Routine

6.4.11 QBTRAN Routine

6.4.12 QBQEND Routine

6.4.13 TAPCON Routine

6.4.14 IOCK Routine

6.4.15 PSSTOP Routine

6.4.16 QBPAND Routine

6.4.17 Subroutine QBEXP9

6.4.18 Subroutine QBEXP1

6.4.19 QBDFIO Routine

6.4.20 IFALT Routine

6.4.21 Subroutine QBDXP9

6.4.22 Subroutine QBDXP1

DOCUMENT CLASS IMS PAGE NO. 6-2  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3-1 A/B MACHINE SERIES 1700

b.5 I/O Restrictions

b.5.1 Structure of Unit History Table Entry

b.6 Error Message

b.6.1 General Discussion

b.6.2 An Error in the FORMAT Statement

b.6.3 Illegal Character in the Input Field

b.6.4 Input Data Exceeds Limits of 1700 Word

b.6.5 Improper Use of Unit

b.6.7 Illegal Formatted Input

b.6.8 Illegal List

b.6.9 File Defined Twice

b.6.10 Parameter Negative or Zero

b.6.11 Sector Address Too Large

b.6.12 File Not Defined

b.6.13 Logical Unit Not a Mass Storage Device

b.6.14 Record Number in READ or WRITE Request Incorrect

b.7 Records

b.7.1 Binary READ-WRITE Statements

b.7.2 Formatted READ-WRITE Statements



DOCUMENT CLASS IMS PAGE NO. 6-3  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. CO05 2.0 MACHINE SERIES 1700

6.0 OBJECT TIME I/O ROUTINES

6.1 GENERAL DESCRIPTION

I/O Requests of the following forms are handled by the I/O routines:

READ {u,f}L WRITE {u,f}L READ {k{n},f}L WRITE {k{n},f}L  
 READ {u,f} WRITE {u,f} READ {k{n}f} WRITE {k{n},f}  
 READ {u}L WRITE {u}L READ {k{n}}L WRITE {k{n}}L  
 READ {u} REWIND u READ {k{n}} WRITE {k{n}}  
 BACKSPACE u ENDFILE u OPEN {k,i,j,u,x}  
 EOF {u} IOCK {u} IFALT {m}

where u is the logical unit number of an input/output device, f is the number of a format statement, L is a list of parameters to be input or output, k is a file number, n is a sector number, i is the number of sectors per record, j is the number of records per file, x is a starting sector number, and m is the type of floating point fault.

6.2 CALLING SEQUENCE FOR I/O REQUESTS

6.2.1 Three Request I/O

I/O requests with parameter lists generate calls to three routines: the initialize routine, the transmission routine, and the terminate routine.

Initialize Routine

The initialize routine is named Q8QINI and has the following calling sequence:

RTJ Q8QINI

1. flag word
2. I/O request number
3. unit number or address of unit number
4. format statement address or address of format statement address.

for disk files, words 3 and 4 are:

3. file number or address of file number
4. sector number or address of sector number
5. format information as in 4 for non-disk

The bits of the flag word are designated to represent the following information:

BIT	SETTING
15	Not Used

## CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 6-4  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005 V2.0 MACHINE SERIES 1700

BIT	SETTING
14	1 formatted 0 unformatted
13	1 unit number 0 address of unit number
12	1 format statement address 0 address of format statement address
11	1 list 0 no list
10	1 read 0 not read
9	1 write 0 not write
8	1 disk I/O 0 no disk I/O
7	1 sector number address 0 address of sector number address
6-0	not used

Transmission Routine

The transmission routine is named Q8QX and has the following calling sequence:

RTJ      Q8QX  
 1. address of list element

Terminate Routine

The terminate routine is named Q8QEND and has the following calling sequence:

RTJ      Q8QEND

## 6.2.2 One Request I/O

I/O requests without parameter lists generate only a call to the initialize routine. The calling sequence is the same as described for Q8QINI above.

DOCUMENT CLASS IMS PAGE NO. 6-5  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005 V2.0 MACHINE SERIES 1700

### 6.2.3 Auxiliary Input/Output Statements Calling Sequence

REWIND u

RTJ Q8QWND  
 1. address of u

BACKSPACE u

RTJ Q8QBCK  
 1. address of u

END FILE u

RTJ Q8QFLE  
 1. address of u

OPEN - allocate disk addresses and open disk file

RTJ Q8QDFNF  
 1. I/O request number  
 2. address of file number  
 3. address of number of sectors per word  
 4. address of number of records per file  
 5. address of logical unit number  
 6. address of starting sector number

### 6.2.4 Basic External Functions

End of file check EOF

RTJ EOF  
 1. address of u

I/O error check IOCK

RTJ IOCK  
 1. address of u

Floating point error condition check IFALT

RTJ IFALT  
 1. number 0, 1, 2 in binary

### 6.2.5 Special Calls

Pause and stop statements

PAUSE

RTJ Q8QPSE

DOCUMENT CLASS IMS PAGE NO. 6-6  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005 V2.0 MACHINE SERIES 1700

PAUSE n

RTJ Q8QPSN

1. octal number n in binary

STOP

RTJ Q8STP

STOP n

RTJ Q8STPN

1. octal number n in binary

### 6.3 DISCUSSION BY STATEMENT TYPE

#### 6.3.1 READ and WRITE

I/O requests of the form READ and WRITE are handled by the same I/O routines. The initialize routine, Q8QINI, interprets the information in the calling sequence and initializes the required routines. The transmission routine transmits the address of the element to the processing routine. If no FORMAT statement is involved, the processing routine is Q8BINB; otherwise Q8IFRM is the processing routine. When all the elements have been processed, Q8QEND is called to terminate the input or output.

READ and WRITE requests without a list are handled by the initialize routine, Q8QINI. After the calling sequence has been interpreted, Q8QINI calls the required processing routine. When the request has been processed, Q8QINI calls Q8QEND to terminate the input or output.

#### 6.3.2 Auxiliary Input/Output Statements

A REWIND u statement is handled by the routine Q8QWND. A status request is made to determine if the unit may be rewound. If the statement is illegal, error message 6 is typed and exit is made to the Utility System. If the statement is legal, a rewind request is made.

A BACKSPACE u statement is handled by the routine Q8QBCK. A status request is made to determine if the unit may be backspaced. If the unit may not be backspaced, error message 6 is typed out and exit is made to the Operating System.

DOCUMENT CLASS IMS PAGE NO. 6-7  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005 V2.0 MACHINE SERIES 1700

An END FILE u statement is processed by the routine Q8QFLE. A status request is made to determine if the unit is capable of writing an end of file {EOF}. If an EOF may not be written, error message b is typed and exit is made to the Operating System.

Q8QWND, Q8QBCK, and Q8QFLE are grouped together in the program names TAPCON. They make use of common subroutines.

An OPEN d, i, j, u, x request is handled by the routine Q8DFNF which is in the program Q8DFIO. A table entry containing the file number, logical unit, and starting sector number is set up and the file space used is allocated. Several errors can result if the parameters are incorrect. The table entry is used by subsequent READ or WRITE requests for that file.

### 6.3.3

#### Basic External Functions

EOF{u} function is serviced by the routine EOF. If bit 12 of the unit history table entry for u is set, an end of file has been written by the previous WRITE request. The A register will contain one upon exit from EOF. Bit 12 not set results in the A register containing two upon exit from EOF.

EOF is grouped with Q8QFLE, Q8QWND, and Q8QBCK in the program TAPCON.

I0CK{u} function is serviced by the routine I0CK. If bit 2 of the unit history table entry for u is set, an error occurred during a previous I/O operation. The A register contains one upon exit from I0CK. Bit 2 not set results in the A register containing two upon exit from I0CK.

IFALT{I} function is serviced by the routine IFALT. The input parameter I may take on the following meanings:

I = 0	Check for overflow
I = 1	Check for divide fault
I = 2	Check for underflow

The values that the parameter may return are:

I = 1	Condition occurred
I = 2	Condition did not occur

DOCUMENT CLASS	IMS	PAGE NO.	6-8
PRODUCT NAME	1700 MASS STORAGE FORTRAN		
PRODUCT MODEL NO.	CO05 V2.0	MACHINE SERIES	1700

#### 6.3.4 Special Calls

A PAUSE statement is handled by the routine QBPSE. The message PAUSE is typed out and a typewriter read is executed. The program continues when a carriage return is typed in. PAUSE n is handled by the routine QBPSEN which is the same as QBPSE except PAUSE n, where n is a 5 digit octal number, is typed out.

A STOP statement is handled by the routine QBSTP. The message STOP is typed out and an exit request is made to the Operation System. STOP n is handled by QBSTPN which is identical to QBSTP except STOP n, where n is a 5 digit octal number, is typed out.

QBPSE, QBPSEN, QBSTP, and QBSTPN are grouped together in the program named PSSTOP.

#### 6.4 DISCUSSION BY ROUTINE

##### 6.4.1 QBQINI Routine

The purpose of this routine is to interpret the calling sequence and initialize the appropriate routines. The routine is written in 1700 assembly language.

Three entry points are declared: QBQINI, QBUNIT, and QBSKIP. The calling sequence to QBQINI is described in 6.2.1. QBUNIT contains the unit number of the I/O request. QBSKIP is called whenever the next record for formatted I/O is to be input or output.

The following externals are declared:

```

QBERRM in QBERRM
QBINTB in QBRWBU
QBEREM in QBERRM
QBTOX in QBQX
QBCMP0 in QBCMP
QBMAGT in QBMAGT
QBQEND in QBQEND
QBIFRM in QBIFRM
QBIGP in QBFGET
QBCMP1 in QBCMP
QBQUN2 in QBQUNI

```

##### 6.4.2 QBQX Routine

The purpose of this routine is to transmit the address of the element and call the appropriate processing routine {binary or format}. The routine is written in 1700 Assembly Language.

DOCUMENT CLASS IMS PAGE NO. 6-9  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3-1 A/B MACHINE SERIES 1700

The following entry points are declared: `QBQTOM`,  
`QBQX{QBQTRM}`, `QBQY`, `QBQZ`, and `QBMOVE`.

`QBQTOM` calling sequence:

LDA	flag word
RTJ	<code>QBQTOM</code>

`QBQX {QBQTRM}` is described in 6.2.1.

`QBMOVE` calling sequence:

RTJ	<code>QBMOVE</code>
ADC	value
ADC	switch

For Binary Transmission the following rules apply:

- A. RTJ `QBQX` {one word transmission}
  - 1. address of integer list element
- B. RTJ `QBQY` {two word transmission}
  - 1. address of single precision floating point element
- C. RTJ `QBQZ` {three word transmission}
  - 1. address of double precision floating point element

DOCUMENT CLASS IMS PAGE NO. 6-10  
PRODUCT NAME 1700 Mass Storage FORTRAN  
PRODUCT NO 005 VERSION 2.0 MACHINE SERIES 1700

For read, value is moved to the address of the list element plus switch. For write, the list element plus switch is moved to the address of value.

The following externals are declared:

Q8IFRM in Q8IFRM  
Q8BINB in Q8RWBU  
Q8QUN1 in Q8QUN1  
Q8UNIT in Q8QINI

#### 6.4.3 Q8QUN1 Routine

This routine is composed of three subroutines to insert data into and fetch data out of the unit history table. The routine is written in 1700 assembly language. The following entry points are declared: Q8QUN1, Q8QUN2, and Q8QUN3.

Q8QUN1 calling sequence:

LDQ unit  
RTJ Q8QUN1

Return with A register containing unit history table entry

Q8QUN2 calling sequence:

LDA value  
LDQ unit  
RTJ Q8QUN2

Value is logically ORed into the unit history table entry

Q8QUN3 calling sequence:

LDA value  
LDQ unit  
RTJ Q8QUN3

Value is stored in the unit history table entry

The following external is declared:

Q8EREM in Q8ERRM

#### 6.4.4 Q8CMP Routine

The routine is written in 1700 assembly language. Q8CMP has 2 entry



DOCUMENT CLASS IMS PAGE NO 6-1  
 PRODUCT NAME 1700 Mass Storage FORTRAN  
 PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

points, Q8CMPO and Q8CMP1. Q8CMPO initializes the input/output request. Q8CMP1 implements the input/output request retaining a composite error history which is left in the A register upon return, when entry is from Q8QEND.

Q8CMPO calling sequence:

```
LDA flag word
LDQ unit
RTJ Q8CMPO
```

Q8CMP1 calling sequence:

```
LDA indicator (non zero if entry from Q8QEND, otherwise 0)
RTJ Q8CMP1
```

Return with A = composite error history (if indicator non zero)

The following externals are declared:

```
Q8EREM in Q8ERRM
Q8BEGB in Q8RWBU
Q8LOCB in Q8RWBU
Q8CLRB in Q8RWBU
Q8RINT in Q8RWBU
```

#### 6.4.5 Q8RWBU Routine

This routine is written in 1700 assembly language. Q8RWBU has seven entry points: Q8IBUF, Q8INTB, Q8LOCB, Q8BEGB, Q8CLRB, Q8RWBU, and Q8BINB. Q8IBUF is the first word in the physical record buffer. As such Q8IBUF is zero for all physical records except for the last physical record when Q8IBUF is equal to the number of physical records in the logical record.

Q8INTB initializes buffer counters and, if the request is a READ, inputs the first record from the I/O device. Q8LOCB loads the address of the current buffer location into the A register. Q8BEGB loads the address of the first buffer location into the A register. Q8CLRB backgrounds the buffer with zeros for binary I/O and ASCII blanks for formatted I/O. Q8RINT reinitializes the buffer counters.

Q8RWBU loads a character from the conversion routine into the buffer for WRITE requests and from the buffer to the conversion routine for READ requests.

Q8BINB loads a binary word from the list into the buffer for WRITE requests and from the buffer into the list for READ requests.

## CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 6-12  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. COO5 V2.0 MACHINE SERIES 1700

Q8INTB calling sequence:

```
LDA flag word
RTJ Q8INTB
```

Q8LOCB calling sequence:

```
RTJ Q8LOCB
```

Return with A = address of current buffer location

Q8BEGB calling sequence:

```
RTJ Q8BEGB
```

Return with A = address of beginning buffer location

Q8CLRB calling sequence:

```
RTJ Q8CLRB
```

Q8RINT calling sequence:

```
RTJ Q8RINT
```

Q8RWBU calling sequence:

```
CALL Q8RWBU {CHAR} or RTJ Q8RWBU
ADC CHAR
```

CHAR is the location of the character to be stuffed into the buffer {WRITE} or the location to stuff the next buffer character into {READ}.

In a statement of the form

```
CALL Q8RWBU {CHAR}
```

CHAR may be a decimal number whose corresponding hexadecimal number gives the ASCII symbol to be stuffed into the buffer {WRITE}.

Q8BINB calling sequence:

```
RTJ Q8BINB
```

The following externals are declared:

```
Q8CMP1 in Q8CMP
Q8BEREM in Q8ERRM
```

DOCUMENT CLASS IMS PAGE NO. 6-13  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C805 V2.0 MACHINE SERIES 1700

#### 6.4.6 Q8FGET Routine

This routine is written in 1700 assembly language. Q8FGET consists of four entry points: Q8L0CF, Q8IGP, Q8FGET, and Q8FPUT. Q8L0CF loads the character counter into the A register. Q8IGP clears the

DOCUMENT CLASS IMS PAGE NO. 6-14  
PRODUCT NAME 1700 Mass Storage FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

character counter and stores the FORMAT statement address. Q8FGET fetches a character from the FORMAT statement. Q8FPUT stores a character into the FORMAT statement.

Q8LOCF calling sequence:

RTJ Q8LOCF

Return with A = character counter

Q8IGP calling sequence:

LDA format statement address

RTJ Q8IGP

Q8FGET calling sequence:

CALL Q8FGET (ADR, SAVE, REP) or RTJ Q8FGET  
ADC ADR, SAVE, REP

Where ADR is the address where the FORMAT statement character is to be stored; SAVE is a flag - 0 means ignore, 1 means save the current character count less 1 in FR, 2 means save the current character count less 1 in GR; REP is a flag - 0 means ignore, 1 means set the character count to FR, 2 means set the character count to GR.

Q8FPUT calling sequence:

CALL Q8FPUT (ADR) or RTJ Q8FPUT  
ADC ADR

Where ADR is the address of the character to be stored into the FORMAT statement.

There are no externals declared by Q8FGET.

#### 6.4.7

#### Q8ERRM Routine

The routine is written in 1700 assembly language. The purpose of this routine is to process error messages. Three entries are declared: Q8ERRM, Q8EREM, and Q8FERM. Q8ERRM converts the I/O request number and unit number from binary to ASCII decimal. Q8EREM processes error messages from 1700 assembly language routines. Q8FERM processes error messages from 1700 FORTRAN language routines.

DOCUMENT CLASS TMS PAGE NO. 6-15  
PRODUCT NAME Mass Storage FORTRAN  
PRODUCT NO. CU05 VERSION 2.0 MACHINE SERIES 1700

Q8ERRM calling sequence:

```
LDA I/O request number
LDQ unit
RTJ Q8ERRM
```

Q8EREM calling sequence:

```
LDA error type
RTJ Q8EREM
```

Q8FERM calling sequence:

```
CALL Q8FERM (I) or RTJ Q8FERM
ADC I
```

Where I is the error type number.

The following externals are declared:

```
Q8LOCF in Q8FGET
Q8LOCB in Q8RWBU
```

#### 6.4.8 Q8MAGT Routine

The purpose of this routine is to check the status of the magnetic tape unit specified for the READ/WRITE request. The routine is written in 1700 assembly language. There is one entry: Q8MAGT.

Q8MAGT calling sequence:

```
LDQ flag word
LDA unit
RTJ Q8MAGT
```

The following externals are declared:

```
Q8EREM in Q8ERRM
Q8QUN2 in Q8QUNI
Q8COMI in PSSTOP
Q8QWND in TAPCON
```

#### 6.4.9 Q8IFRM Routine

The purpose of this routine is to initialize certain cells and call either the format scanner (Q8FS) or the conversion and transmission routine (Q8TRAN) based on the setting of the input parameter IENTY. It is coded in FORTRAN and is defined as follows:

DOCUMENT CLASS IMS PAGE NO. 6-16  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. COO5\*3.1 A/B MACHINE SERIES 1700

SUBROUTINE Q&BIFRM {IENTY, IRW, ILIST}

The input parameters have the following meaning:

- IENTY - Switch which determines the phasing of the call. 0 = I/O list just beginning to be processed {entered from Q&BQINI}. 1 = I/O list being processed {entered from Q&BQX}.
- IRW - Read/Write Switch. 0 = Read, 1 = Write.
- ILIST - Any list associated with this I/O. 1 = list, 0 = no list.

6.4.10 Q&BFS Routine

The purpose of this routine is to scan the FORMAT statement, interpret individual subfields, and pass the resulting field specification parameters to the conversion and transmission routine, Q&BTRAN. It is coded in FORTRAN and is defined as follows:

SUBROUTINE Q&BFS {IRW, IFW, IND, ITYPE, IRF, IRG, IBCT, ISWR, ILIST, IANYL}

The input parameters have the following meaning:

- IRW - Read/Write switch. 0 = Read, 1 = Write.
- IFW - Number of characters in I/O field.
- IND - Number of characters to right of decimal point.
- ITYPE - Type of conversion to be done
  - 1 = A
  - 2 = R
  - 3 = E
  - 4 = F
  - 5 =  $\phi$
  - 6 = I
  - 7 = D
- IRF - Number indicating the number of times the current field is to be repeated.
- IRG - Number indicating the number of times the current group is to be repeated.
- IBCT - Number indicating the parenthesis count. +1 for all { and -1 for all }.
- ISWR - When this switch = 2 a new I/O is performed, otherwise not. It is a delayed read/write switch.
- ILIST - 0 = no list; 1 = list associated with this I/O statement.
- IANYL - 0 no conversion specification exists in FORMAT statement. Non-zero = at least 1 conversion specification exists.

DOCUMENT CLASS IMS PAGE NO 6-17  
PRODUCT NAME 1700 Mass Storage FORTRAN  
PRODUCT NO. COOS VERSION 2.0 Machine SERIES 1700

#### 6.4.11 Q8TRAN Routine

The purpose of this routine is to convert the next subfield for I/O based on the appropriate FORMAT specification and transmit the subfield to or from core. It controls all I/O done under FORMAT control.

It is coded in FORTRAN and is defined as follows:

```
SUBROUTINE Q8TRAN (IRW, IFW, IND, ITYPE, IRF, IRG, IBCT, ISWR,  
ILIST, IANYL)
```

The input parameters have the same meaning as described in section 6.4.10 for Q8FS.

#### 6.4.12 Q8QEND Routine

The purpose of Q8QEND is to terminate the READ/WRITE request and store the cumulative error history into the unit history table. The routine is written in 1700 assembly language. The only entry is Q8QEND.

Q8QEND calling sequence:

```
RTJ Q8QEND
```

The following externals are declared:

```
Q8CMP1 in Q8CMP  
Q8QUN1 in Q8QUNI  
Q8QUN2 in Q8QUNI  
Q8UNIT in Q8QINI
```

#### 6.4.13 TAPCON Routine

TAPCON consists of magnetic tape function routines. The routine is written in 1700 assembly language. Entries are Q8QBCK, Q8QFLE, Q8QWND, and EOF. Calling sequences and functions are described in 6.2.3, 6.2.4, 6.3.2, and 6.3.3.

The following externals are declared:

```
Q8EREM in Q8ERRM  
Q8CMP0 in Q8CMP  
Q8CMP1 in Q8CMP  
Q8QUN1 in Q8QUNI  
Q8QUN2 in Q8QUNI  
Q8QUN3 in Q8QUNI  
Q8IBUF in Q8QINI
```

DOCUMENT CLASS IMS PAGE NO 6-18  
PRODUCT NAME 1700 Mass Storage FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

#### 6.4.14 IOCK Routine

The routine is written in 1700 assembly language. IOCK is a routine to check for any I/O error. It is described in 6.2.4 and 6.3.3. It has one entry point: IOCK.

The following externals are declared:

Q8QUN1 in Q8QUNI  
Q8QUN3 in Q8QUNI

#### 6.4.15 PSSTOP Routine

PSSTOP consists of the special calls Q8PSE, Q8PSEN, Q8STP, Q8STPN, described in 6.2.5 and 6.3.4 plus the entry Q8COMI. The routine is written in 1700 assembly language. Q8COMI reads the teletypewriter for one character.

Q8COMI Calling Sequence

RTJ Q8GOMI

One external is declared: Q8PAND

#### 6.4.16 Q8PAND Routine

The purpose of Q8PAND is to type PAUSE or STOP followed by 5 octal digits if specified. The routine is written in 1700 assembly language. It has one entry point, Q8PAND.

Q8PAND calling sequence:

LDA NUM  
LDQ ADR  
RTJ Q8PAND

Where NUM is a binary number to be converted to 5 digit ASCII octal, or, if NUM 0, to be ignored. ADR is the address of PAUSE or STOP in ASCII.

There are no externals.

#### 6.4.17 Subroutine Q8EXP9

Routine converts a binary integer and decimal exponent into a floating point number. The integer is assumed to be greater than 0. Exponents may be positive, negative, or zero.



DOCUMENT CLASS \_\_\_\_\_ IMS \_\_\_\_\_ PAGE NO. 6-19  
 PRODUCT NAME 1700 Mass Storage FORTRAN  
 PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

Input parameters: P address of 2 word binary integer  
 P contains most significant 16 bits  
 P + 1 contains least significant 12 bits, right justified  
 P1 address of decimal exponent

Output parameters: P, P+1 contains floating point number (0 if underflow has occurred, \$7FFF, \$FFFF if overflow has occurred)  
 P1 error indicator  
 0 implies no error  
 ≠0 implies error

Routines Used: FLOT

Used by: Q8TRAN (FORTRAN object time conversion routine)

Language: 1700 Assembly Language

6.4.18

Subroutine Q8EXP1

Converts a floating point number to an octal fraction with decimal exponent.

Input parameter: P address of 4 word table P+1, P+2 contain floating point number to convert

Output parameters: P+1, P+2 contain octal fraction (lower 12 bits of each word)  
 P+3 value of decimal exponent

Used by: Q8TRAN

Routines used: FLOT

Other externals: Q8EXPT (in Q8EXP9)

Language: 1700 Assembly Language

DOCUMENT CLASS IMS PAGE NO. 6-20  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005 V2.0 MACHINE SERIES 1700

6.4.19

**Q8DFIO Routine**

Q8DFIO which is written in 1700 assembly language, handles the disk I/O necessary for OPEN statement. This routine may be split into two separate parts.

The first part, which has the entry point Q8DFIN, is called by Q8QINI to initialize the disk I/O.

Q8DFIN calling sequence:

```
RTJ  Q8DFIN
```

where

A contains the file number  
 Q contains the flag word shifted 7 left

This routine finds the corresponding file table entry, computes the starting sector number and places it in Q8DFAD {in Q8CMP} and returns with the logical unit in A.

The second part, which has the entry point Q8DFNF, is entered by a compiler generated call from the user's program when executing an OPEN statement.

Q8DFNF calling sequence:

```
RTJ  Q8DFNF
      1. address of file number
      2. address of number of sectors per record
      3. address of number of records per file
      4. address of disk logical unit number
      5. address of starting sector number
```

The disk file table starts in the low address of unused core and extends upwards. Three words are created for each file. The format is as follows:

```
WORD 0: file number
WORD 1: logical unit number
WORD 2: starting sector address {x}
```

The following externals are declared in Q8DFIO:

```
QBERRM  in QBERRM
QBEREM  in QBERRM
QBQINI  in QBQINI
Q8DFAD  in Q8CMP
QBQENS  in Q8CMP
```

DOCUMENT CLASS IMS PAGE NO. 6-21  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

6.4.20 IFALT Routine

This routine checks for a floating point error condition.

The input parameter I may take on the following values:

I = 0	Check for overflow
1	Check for divide fault
2	Check for underflow

The parameter I may return the following values:

I = 1	Condition occurred
2	Condition did not occur

This routine, which is written in 1700 Assembly Language, uses the contents of location #C8 in checking for the above conditions. The routines FLOAT and DFL0T store their error condition information in this low core location.

There are no externals.

6.4.21 Subroutine Q8DXP9

This routine converts a binary integer and decimal exponent into a double precision {3 word} floating point number. The integer is assumed to be greater than zero. Exponents may be positive, negative, or zero.

Input parameters:

P	address of 4 word binary integer
P	contains the most significant 4 bits
P+1	contains the next intermediate significant 12 bits right justified
P+2	contains the next intermediate significant 12 bits right justified
P+3	contains the least significant 12 bits right justified
P1	address of decimal exponent

Output parameters:

P	contains the double precision point number {0 if underflow has occurred #7FFF, #FFFF, #FFFF if overflow has occurred.
P+1	
P+2	
0	implies no error
≠0	implies error

DOCUMENT CLASS IMS PAGE NO. 6-21A  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

Routines Used: DFL0T

Used by: Q8TRAN {FORTRAN object time conversion routine}

Language: 1700 Assembly Language

6.4.22 Subroutine Q8DXP1

This routine obtains a double precision floating point number and converts it to an intermediate form of a decimal integer and a decimal exponent.

Input parameter: P address of b word table  
P+1 } contain the double precision  
P+2 } floating point number to be  
P+3 } converted

Output parameter: P+1 } contain the intermediate binary  
P+2 } fraction {Lower 12 bits of each  
P+3 } word}  
P+4 } {only the upper 4 bits of the  
lower 12 bits are used}  
P+5 contains the decimal exponent

Used by: Q8TRAN

Routines Used: DFL0T

Other Externals: Q8DXPT and Q8DXP2 {in Q8DXP1}

Language: 1700 Assembly Language

6.5 I/O Restrictions

The number of logical units which may be used in one program is limited to 30. This number is introduced to limit the number of cells necessary for storage of the unit history table. The unit history table must be available to give information on the EOF and IOCK requests.

DOCUMENT CLASS IMS PAGE NO. 6-22  
 PRODUCT NAME 1700 Mass Storage FORTRAN  
 PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

Each time a new logical unit is used in an I/O request, an entry will be made in the table. When use of more than 30 units is attempted, an error message will be typed. The transmission information on the I/O request is placed in the table upon completion of the request.

6.5.1 Structure of Unit History Table Entry

BIT	
15	Not used
14	1 Formatted 0 Unformatted
13	1 Rewind 0 Non rewind
12	1 EOF Read 0 No EOF Read
11	1 Backspace 0 Not backspace
10	1 Read 0 Not read
9	1 Write 0 Not write
8-3	Not used
2	1 Error occurred 0 Error free request
1-0	Not used

6.6 ERROR MESSAGE

6.6.1 General Discussion

Error messages will be output on the standard comment medium. The message form will be one of the following three types:

- Type 1: N  
I/O RQST n  
ffff
- Type 2: N  
I/O RQST n  
ffff  
8888
- Type 3: N  
I/O RQST n  
XX



DOCUMENT CLASS IMS PAGE NO. 6-23  
PRODUCT NAME 1700 Mass Storage FORTRAN  
PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

1. N is an error number which corresponds to an error message.
2. n is the I/O request statement number.
3. ffff is the current value of the format statement pointer. This value is in decimal.
4. gggg is the current value of the input field pointer. This value is in decimal.
5. xx is the decimal unit number of an improperly used device.

6.6.2 An Error in the FORMAT Statement

MESSAGE: 1  
I/O RQST n  
ffff

Illegal character in format statement

Action: program termination

6.6.3 Illegal Character in the Input Field

MESSAGE: 2  
I/O RQST n  
ffff  
8888

Action: program termination

6.6.4 Input Data Exceeds Limits of 1700 Word

MESSAGE: 3  
I/O RQST n  
ffff  
8888

Action: program termination

6.6.5 Improper Use of Unit

MESSAGE: N  
I/O RQST n  
unit number

DOCUMENT CLASS IMS PAGE NO. 6-24  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005 V2.0 MACHINE SERIES 1700

N	Description of Improper Use	Action
4	Attempt to read on a write unit or write on a read unit	Program termination
5	Read or write request after an end of file has been written without first doing an EOF check.	Program termination
6	Attempt to write an EOF, rewind, or backspace any unit other than a magnetic tape	Program termination
7	Write attempted on magnetic tape with no write enable	Type (cr) to continue
8	Attempt to use logical unit number greater than 30	Program termination
9	Backspace at load point	Program termination
10	End of magnetic tape sensed	Rewind tape. Type (cr) to continue

#### 6.6.7 Illegal Formatted Input

More elements are given than are contained in an input record.

MESSAGE: 12  
I/O RQST n  
ffff

Action: program termination

#### 6.6.8 Illegal List

A list is given but there are no conversion codes in the format statement.

MESSAGE: 13  
I/O RQST n  
ffff

Action: program termination



DOCUMENT CLASS IMS PAGE NO. 6-25  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. CO05 V2.0 MACHINE SERIES 1700

- 6.6.9 File defined twice  
More than one OPEN request given for the same file.  
MESSAGE: 14  
I/O RQST n  
file number  
ACTION: Program is terminated.
- 6.6.10 Parameter negative or zero  
One of the parameters in an OPEN statement is negative or zero.  
MESSAGE: 15  
I/O RQST n  
file number  
ACTION: Program is terminated
- 6.6.11 Sector address too large  
The starting sector address or ending address exceeds 215-1.  
MESSAGE: 16  
I/O RQST n  
file number  
ACTION: Program is terminated
- 6.6.12 File not defined  
A READ or WRITE request was given for a file which was not defined by an OPEN statement.  
MESSAGE: 17  
I/O RQST n  
file number  
ACTION: Program is terminated
- 6.6.13 Logical unit not a mass storage device  
MESSAGE: 18  
I/O RQST n  
file number  
ACTION: Program is terminated

CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 6-26  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005 V2.0 MACHINE SERIES 1700

6.6.14

Record number in READ or WRITE request incorrect. Resulting sector address is out of the range of the file or is zero.

MESSAGE: 19  
I/O RQST n  
file number

ACTION: Program is terminated

6.7 RECORDS

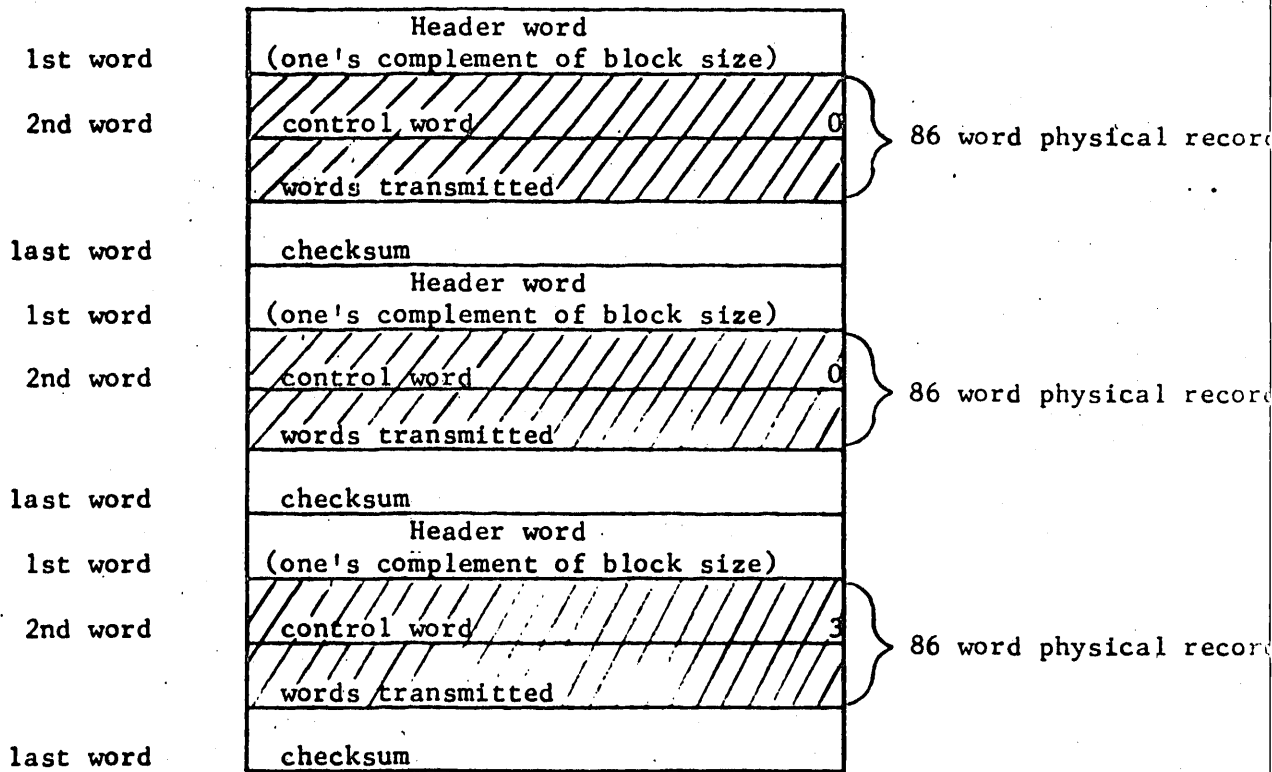
6.7.1 Binary READ-WRITE Statements

A binary write produces a logical record which may be composed of any number of physical records.

A READ (u) statement spaces the tape one logical record.

A WRITE (u) statement is illegal.

The first word will be interpreted as a control word and does not represent data. A WRITE (u)L will produce on paper tape a logical record as follows:



The mode is binary. The checksum when added to the sum of all the data words and the header word will result in zero. Overflow is ignored in the computation of the checksum.

DOCUMENT CLASS IMS PAGE NO. 6-28  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005 V2.0 MACHINE SERIES 1700

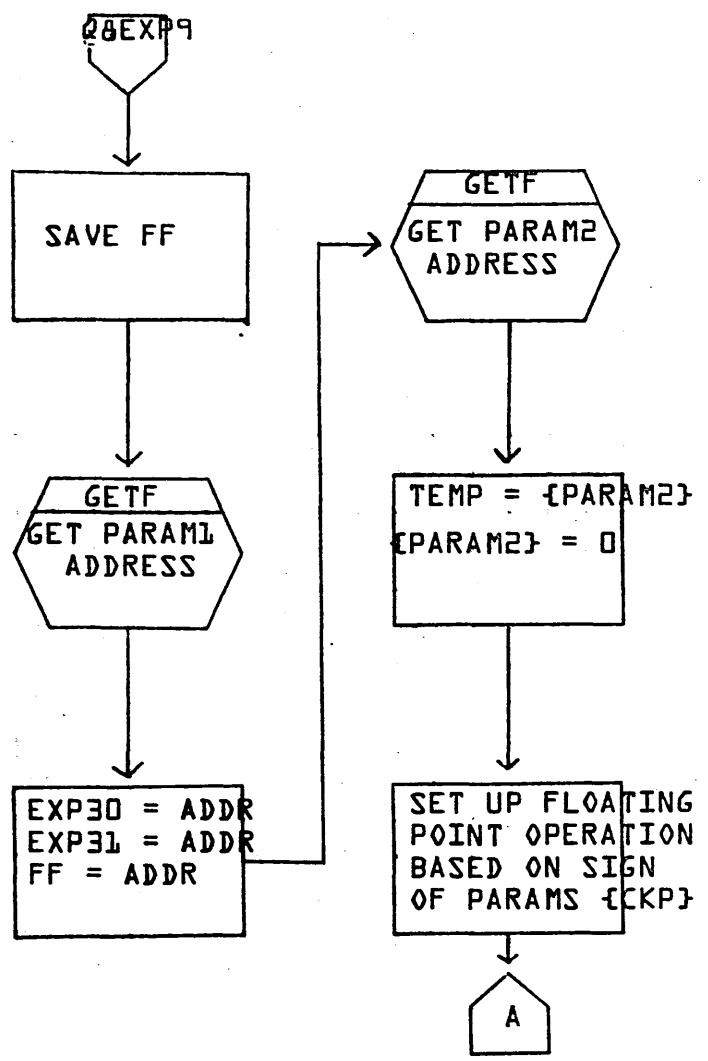
A READ {u}L will read one logical record with the above format. The header word and checksum are not read into the buffer.

A WRITE {u} will produce on magnetic tape a logical record which has the same form as the paper tape record except a record gap replaces the checksum and header word.

#### 6.7.2 Formatted READ-WRITE Statements

All formatted read-write statements produce a record of 60 or less words, 2 ASCII characters per word. Attempts to read more than 60 words {120 characters} in a record result in an error message and program termination. Attempts to write more than 60 words {120 characters} in a record results in truncation of the record at the 120th character.

A  
B  
C  
D



**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

SAMPLE CODE

FLOWCHART

DECISION TABLE

OTHER

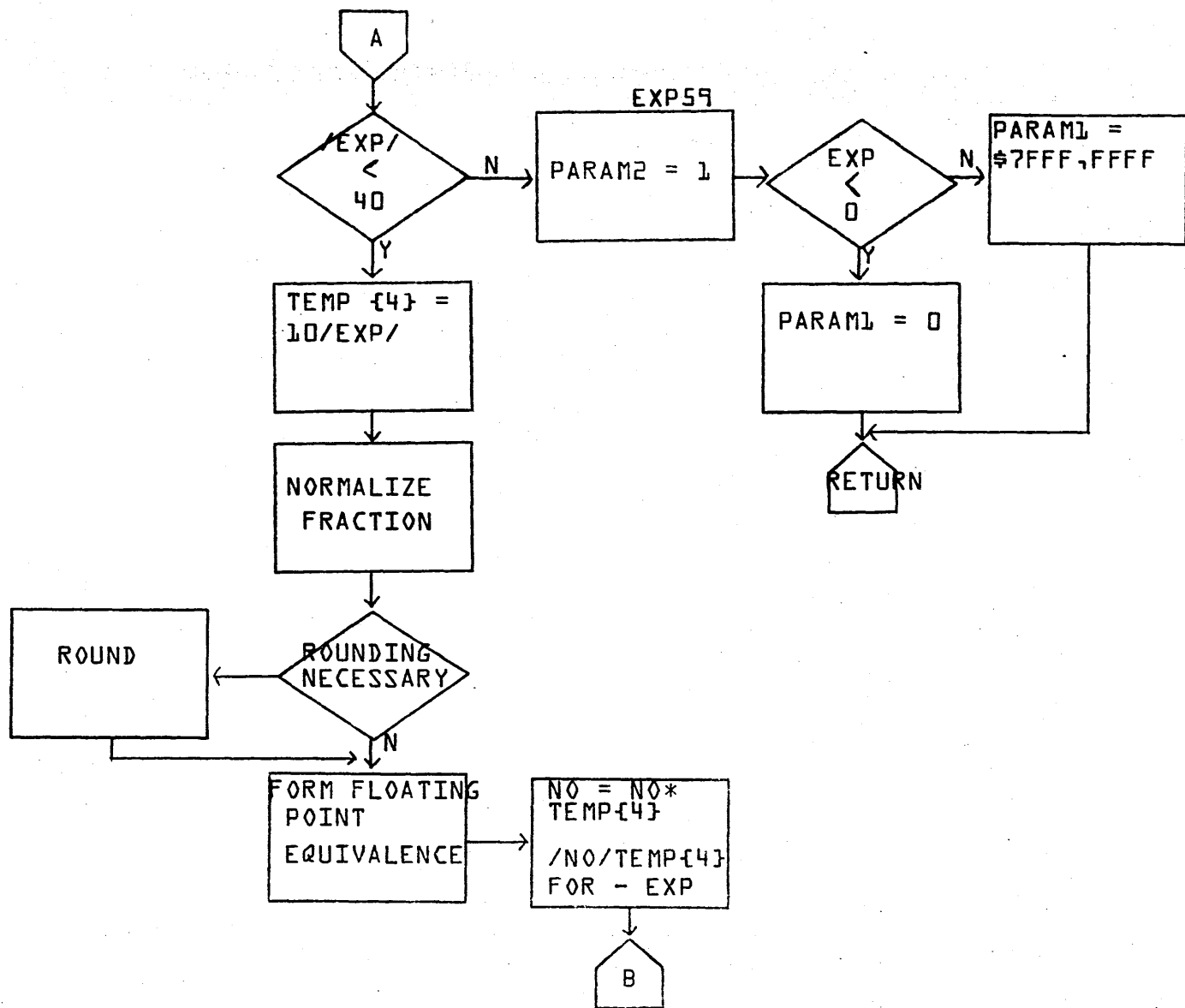
DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	Q8EXP9	PAGE 1 OF 3		PROJECT MGR.			
NUMBER	ISSUE DATE			PROJECT NAME			
DRAWN BY	DATE			TASK NO.			
				TASK NAME			

A

B

C

D



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	Q8EXP9			PROJECT MGR.			
		PAGE 2 OF 3			PROJECT NAME			
	NUMBER		ISSUE DATE		TASK NO.			
	DRAWN BY		DATE		TASK NAME			

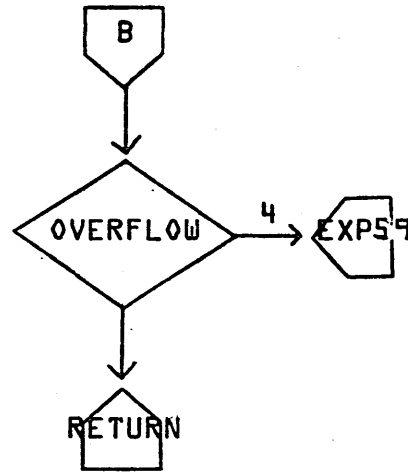
16-30

A

B

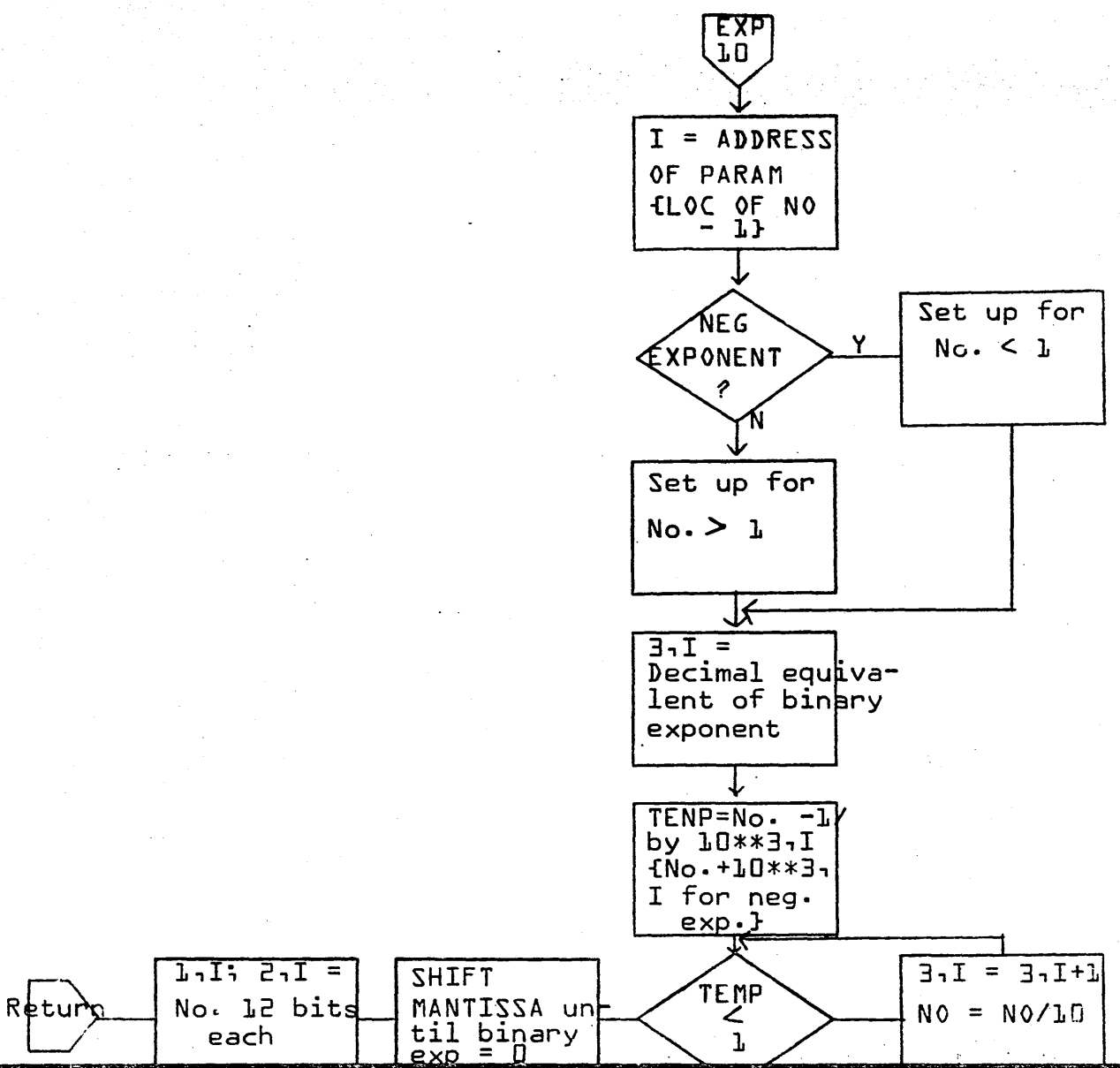
C

D



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	QBEXP9			PROJECT MGR.			
		PAGE 3 OF 3			PROJECT NAME			
	NUMBER		ISSUE DATE		TASK NO.			
	DRAWN BY		DATE		TASK NAME			

A  
B  
C  
D

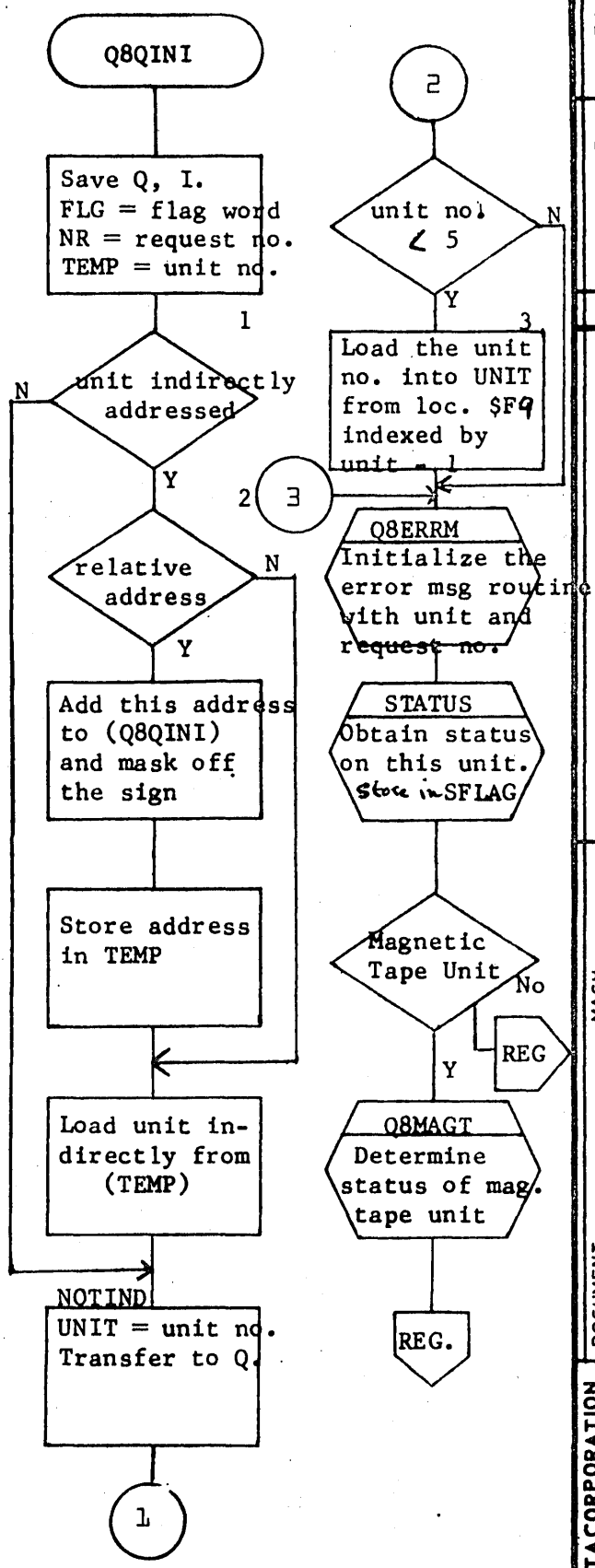
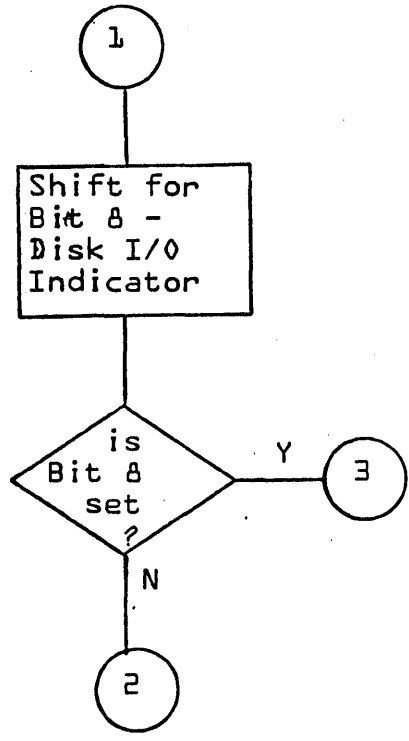


<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	06EXPL			PROJECT MGR.			
	PAGE 1 OF 1				PROJECT NAME			
	NUMBER	ISSUE DATE	TASK NO.					
	DRAWN BY		DATE		TASK NAME			

F-32

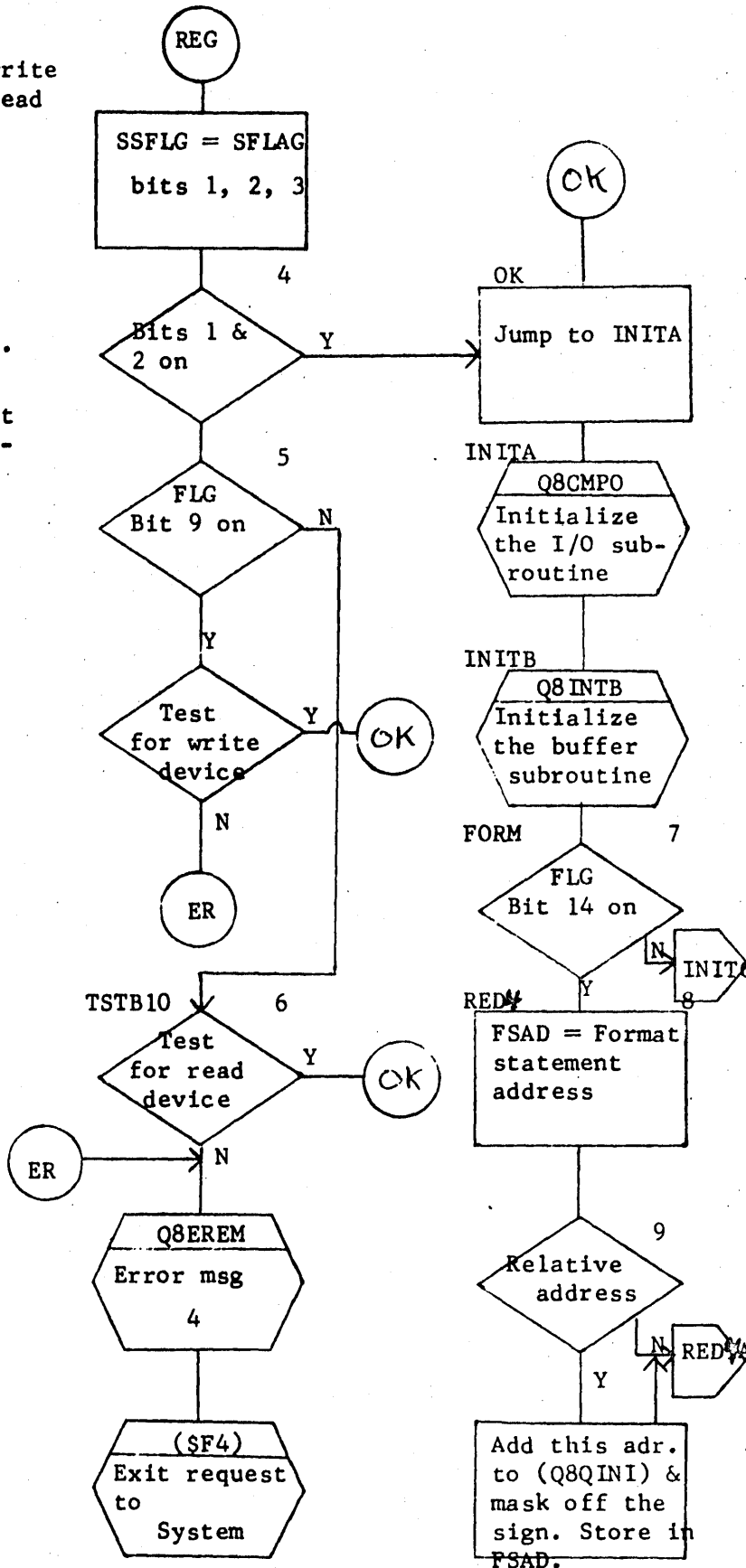


1. Flag word bit 14 off means this is the address of the unit number.
2. A negative address means this address is relative to the given address.
3. Location F9<sub>16</sub> is the start of the System Communication region and contains the device code for standard input, standard binary output, standard list output and comments, consecutively.



DATE	
APPROVED	
REV	
PROJECT NO.	
PROJECT MGR.	
PROJECT NAME	
TASK NO.	
TASK NAME	
MACH. TYPE	1700
DOCUMENT CLASS	TMS
DOCUMENT TITLE	Q8QINI
ISSUE DATE	PAGE 1 of 89
NUMBER	
DRAWN BY	DATE 3-6-79
<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT SAMPLE CODE <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> FLOWCHART DECISION TABLE OTHER	

4. Bit 1 on for read unit; bit 2 on for write unit; bit 3 on for read or write.
5. Flag word bit 9 on means write request.
6. Flag word bit 10 on means read request.
7. Flag word bit 14 on means format request.
8. FSAD can contain the address of the format statement direct, indirect or relative.
9. A negative address means this address is relative to the given address.



DATE	
APPROVED	
REV	
PROJECT NO.	
PROJECT MGR.	
PROJECT NAME	
TASK NO.	
TASK NAME	
DOCUMENT CLASS	
DOCUMENT TITLE	
Q8QINI NUMBER	
DRAWN BY	
MACH. TYPE	1700
ISSUE DATE	
PAGE 2 OF 89	
DATE	3-67

CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE

FLOWCHART

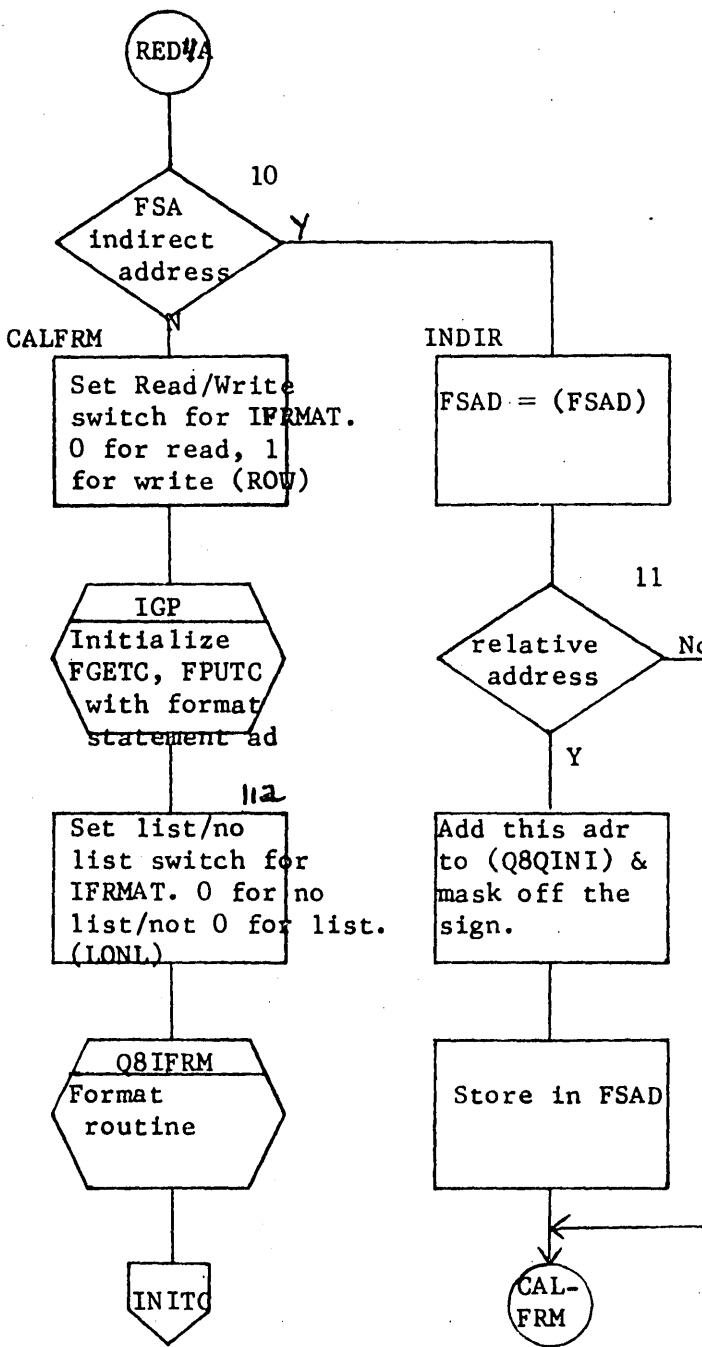
DECISION TABLE

OTHER

10. Flag word bit 12 off means this is the address of the format statement address.

11. A negative address means this address is relative to the given address.

112. This box belongs on page 2 at label FORM.



DATE	APPROVED	REV	PROJECT NO.	MACH. TYPE	DOCUMENT CLASS
				1700	INF
			PROJECT MGR.	Q8QINI	Q8QINI
			PROJECT NAME	PAGE 3 OF 89	NUMBER
			TASK NO.	ISSUE DATE	DRAWN BY
			TASK NAME	DATE	

CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

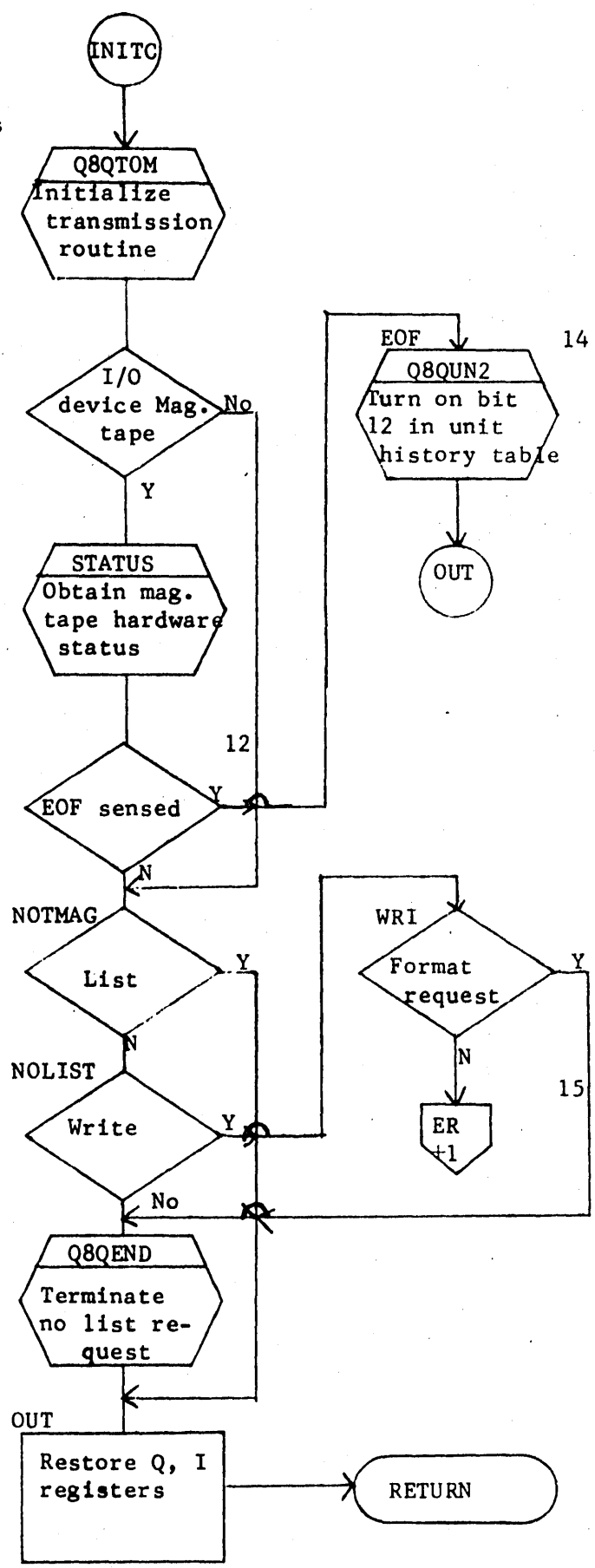
SAMPLE CODE

FLOWCHART

DECISION TABLE

OTHER

- 12. Bit 11 is on if an end of file was sensed by the magnetic tape status check.
- 13. LONL non-zero means this request has a list. Flag word bit 11 means list.
- 14. Bit 12 in the unit history table is used to flag end of file status on the device.
- 15. A formatted write request without list is a fatal error. Error message 11 is typed out.



DATE	
APPROVED	
REV	
PROJECT NO.	
PROJECT MGR.	
PROJECT NAME	
TASK NO.	
TASK NAME	
MACH. TYPE	1100
DOCUMENT CLASS	IMS
DOCUMENT TITLE	
Q8QINI NUMBER	4 of 89
ISSUE DATE	
DATE	3-6-7
DRAWN BY	
CONTROL DATA CORPORATION	
SOFTWARE DOCUMENT	
SAMPLE CODE	<input type="checkbox"/>
FLOWCHART	<input checked="" type="checkbox"/>
DECISION TABLE	<input type="checkbox"/>
OTHER	<input type="checkbox"/>

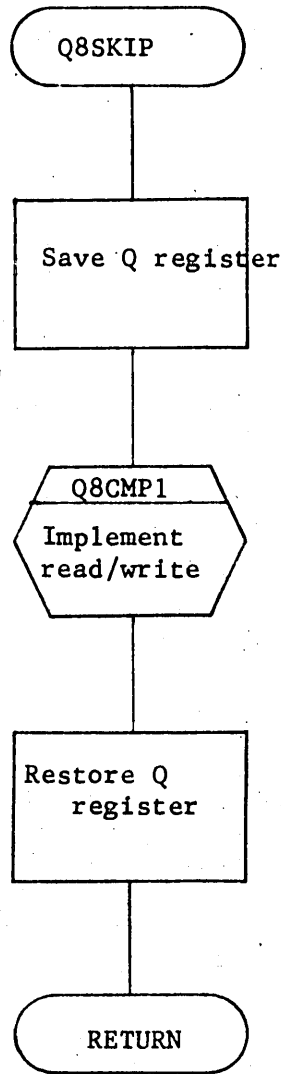
5

4

2

1

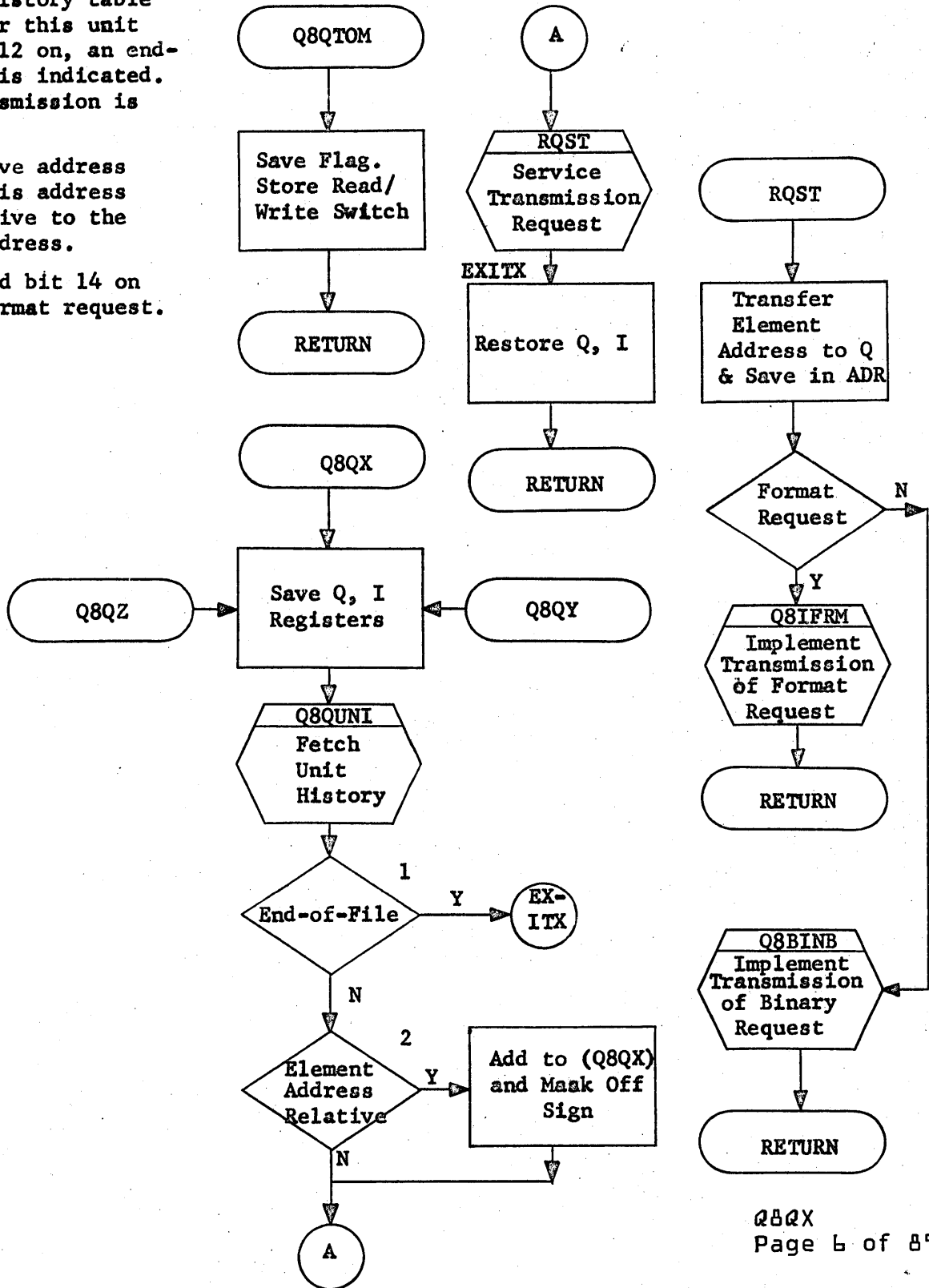
1. Q8SKIP is an entry point in Q8QINI but is an independent subroutine called by the format routines to skip to the next record. It is called whenever a slash occurs in the format statement or deemed necessary by the format routines.



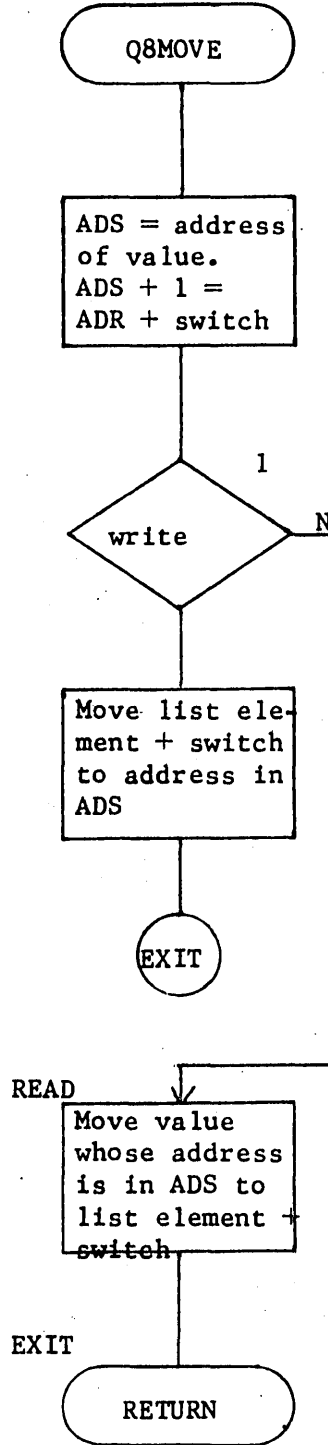
CONTROL DATA CORPORATION	DOCUMENT CLASS	MACH. TYPE	PROJECT NO.	REV	APPROVED	DATE
SOFTWARE DOCUMENT	<i>IMS</i>	<i>1700</i>				
SAMPLE CODE	DOCUMENT TITLE	PAGE	PROJECT MGR.			
<input type="checkbox"/> FLOWCHART	Q8SKIP	<i>5 of 89</i>				
<input checked="" type="checkbox"/> DECISION TABLE	NUMBER	ISSUE DATE	TASK NO.			
<input type="checkbox"/> OTHER	DRAWN BY	DATE	TASK NAME			
		<i>3-67</i>				

DOCUMENT CLASS IMS PAGE NO. 6-38  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

1. If the history table entry for this unit has bit 12 on, an end-of-file is indicated. All transmission is ignored.
2. A negative address means this address is relative to the given address.
3. Flag word bit 14 on means format request.



1. Flag word bit 9 on means write.
2. Q8MOVE is an entry point in Q8QX.



	APPROVED	DATE			
REV					
PROJECT NO.					
PROJECT MGR.					
PROJECT NAME					
TASK NO.					
TASK NAME					
MACH. TYPE	1700	PAGE 7 OF 89	ISSUE DATE	DATE	5-67
DOCUMENT CLASS	IAC				
DOCUMENT TITLE					
NUMBER	080X				
DRAWN BY					

**CONTROL DATA CORPORATION**

SOFTWARE DOCUMENT

SAMPLE CODE

FLOWCHART

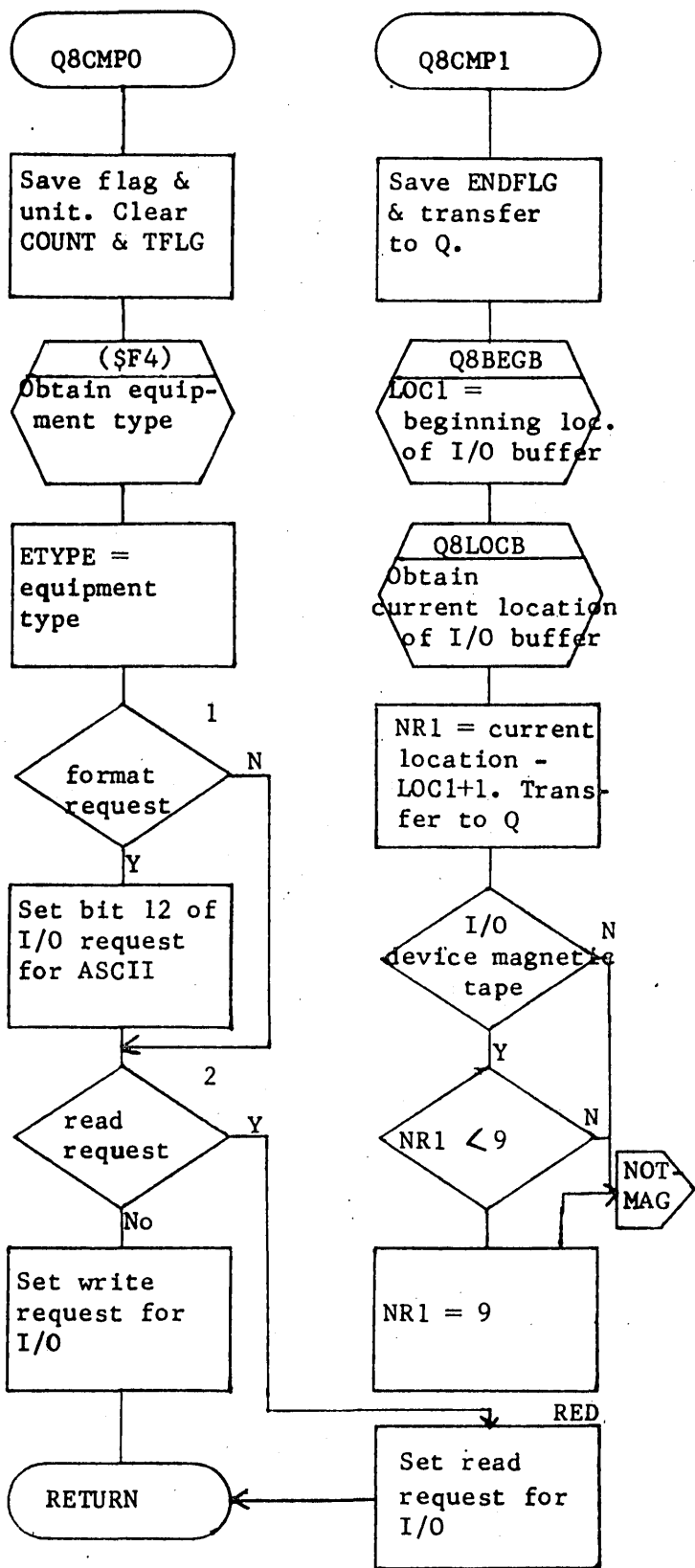
DECISION TABLE

OTHER





1. Flag word bit 14 on means formatted I/O.
2. Flag word bit 10 on means read request.
3. Magnetic tape transfers must be at least 9 words in length due to hardware constraints.

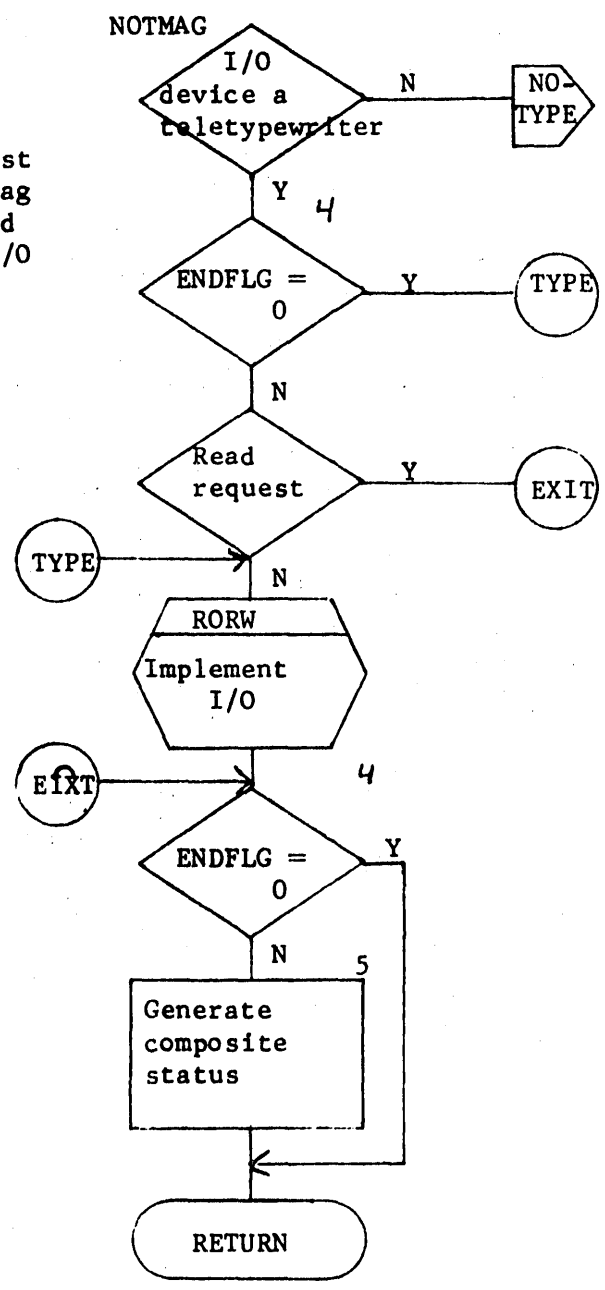


	DATE		APPROVED		REV
PROJECT NO.		PROJECT MGR.		PROJECT NAME	
TASK NO.		TASK NAME			
DOCUMENT CLASS	MACH. TYPE 1700	DOCUMENT TITLE	Q8CMP0	ISSUE DATE	DATE 5-67
SAMPLE CODE		FLOWCHART	<input checked="" type="checkbox"/>	DECISION TABLE	<input type="checkbox"/>
OTHER			<input type="checkbox"/>		

CONTROL DATA CORPORATION  
 SOFTWARE DOCUMENT

NUMBER 98000  
 DRAWN BY

- 4. ENDFLG indicates entry from Q8QEND if non-zero.
- 5. Composite status consist of bits 9-10 of the flag word for read/write and bits 13-14-15 of the I/O request status.



	REV	APPROVED	DATE
DOCUMENT CLASS	PROJECT NO.	PROJECT MGR.	PROJECT NAME
DOCUMENT TITLE	ISSUE DATE	TASK NO.	TASK NAME
Q8CMP0	PAGE 10 OF 89		
NUMBER	DATE 5-6-7		
DRAWN BY			

**CONTROL DATA CORPORATION**

SOFTWARE DOCUMENT

SAMPLE CODE

FLOWCHART

DECISION TABLE

OTHER

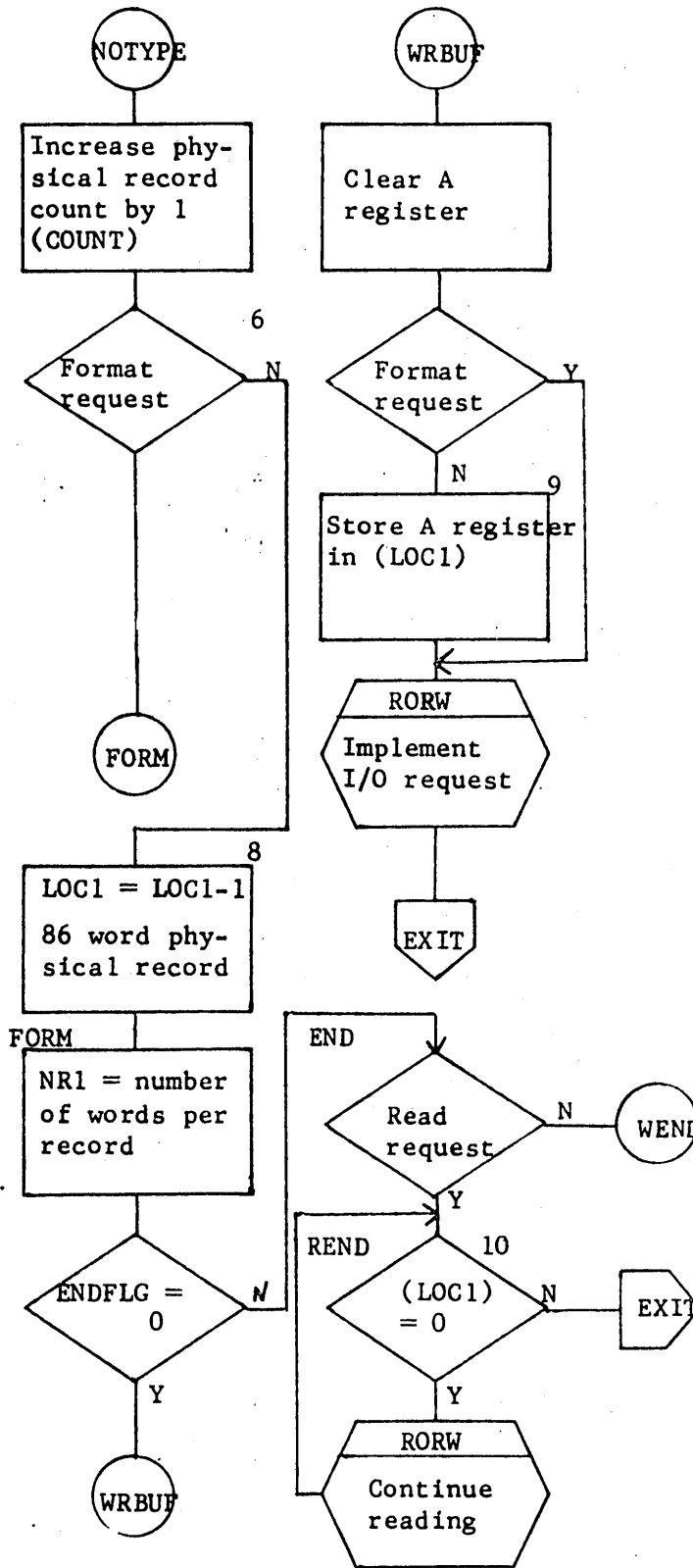
6. Flag word bit 14 on means format request.

8. All binary I/O requests are in logical records of 86 words each.

The first word of each 86 word record is zero for all records except the last which contains the number of physical records per logical record (COUNT).

9. Clear 1st word of binary physical record.

10. 1st word zero means more physical records to follow. Q8QEND requires input of all physical records until entire logical record has been read.



DATE		APPROVED		REV		PROJECT NO.		MACH. TYPE	1400
						PROJECT MGR.		DOCUMENT TITLE	Q8CMP0
						PROJECT NAME		PAGE	1 of 89
						TASK NO.		NUMBER	
						TASK NAME		ISSUE DATE	
								DRAWN BY	JAK
								DATE	5-1-69

CONTROL DATA CORPORATION  
 SOFTWARE DOCUMENT

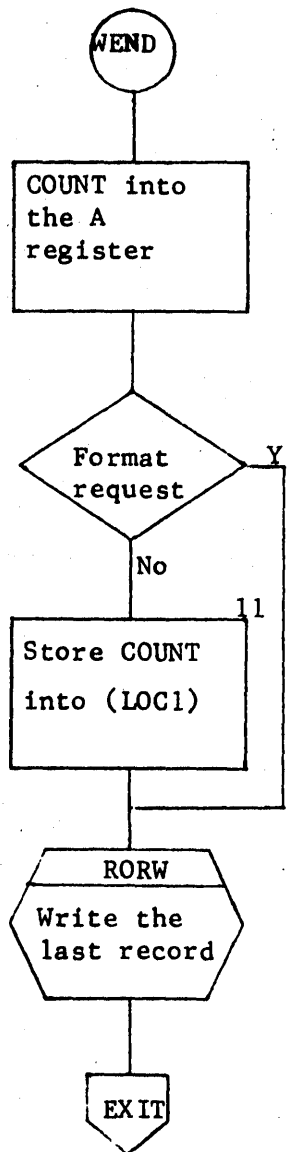
SAMPLE CODE

FLOWCHART

DECISION TABLE

OTHER

11. Q8QEND means terminate the write request. For binary this means the physical record count is stored in the first word of this, the last, record. For format, output the record.

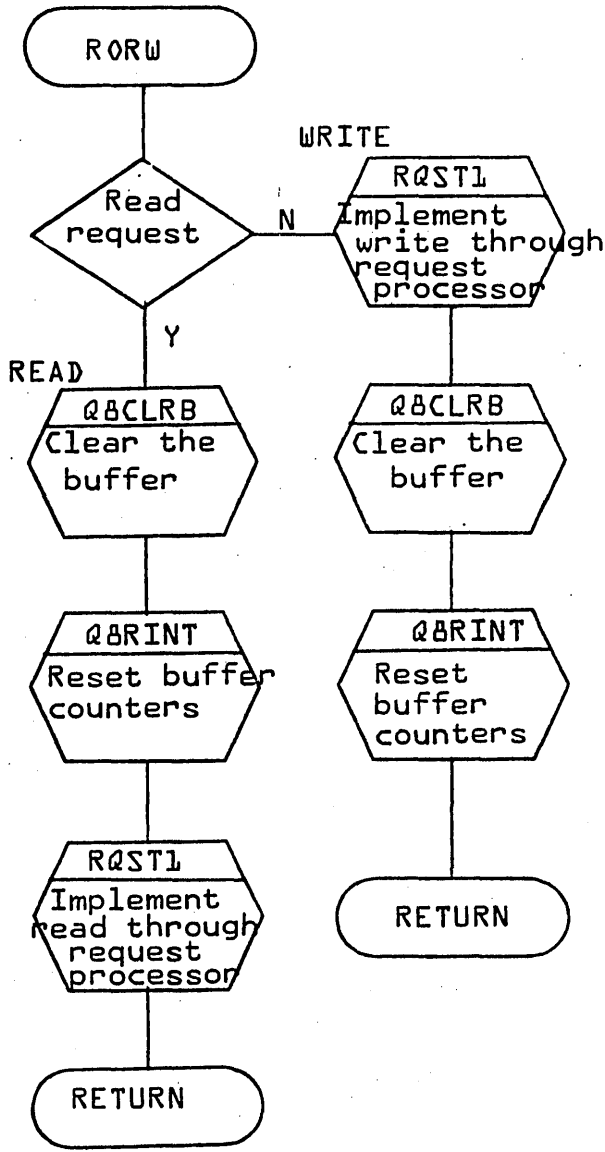


CONTROL DATA CORPORATION		DOCUMENT CLASS	MACH. TYPE	PROJECT NO.	DATE
SOFTWARE DOCUMENT		DOCUMENT TITLE	ISSUE DATE	PROJECT MGR.	APPROVED
SAMPLE CODE		Q8CMP0	PAGE 12 OF 89	PROJECT NAME	
FLOWCHART			ISSUE DATE	TASK NO.	
DECISION TABLE			DATE	TASK NAME	
OTHER					

5

4

2



CONTROL DATA CORPORATION		DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.		APPROVED	DATE
SOFTWARE DOCUMENT		DOCUMENT TITLE	QBCMPO			PROJECT MGR.		REV	
SAMPLE CODE		NUMBER	13	ISSUE DATE	89	PROJECT NAME			
FLOWCHART		DRAWN BY		DATE		TASK NO.			
DECISION TABLE						TASK NAME			
OTHER									

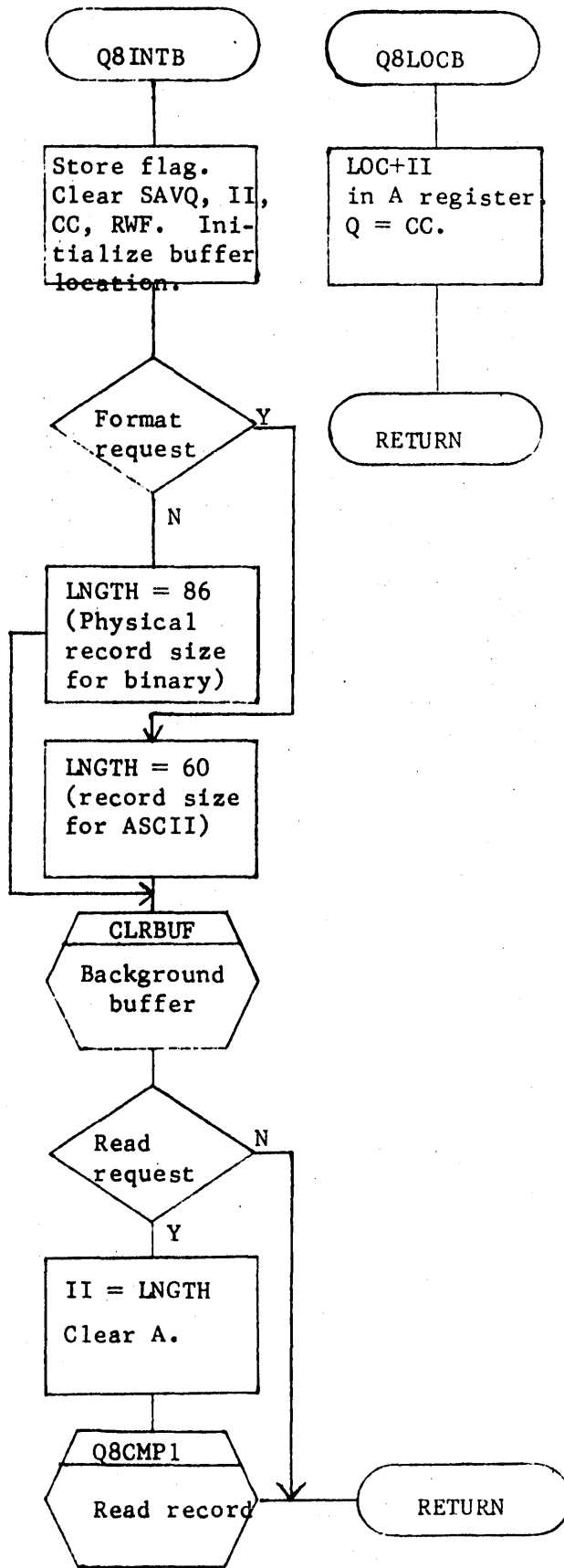
A

B

C

D

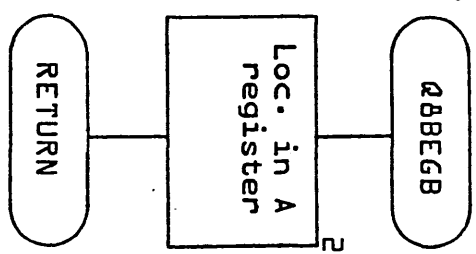
1. LOC+II is the current location of the input/output buffer.



CONTROL DATA CORPORATION		DOCUMENT CLASS	MACH TYPE	PROJECT NO.	APPROVED	DATE
SOFTWARE DOCUMENT		DOCUMENT TITLE		PROJECT MGR.		
SAMPLE CODE		Q8RWRU	PAGE 1 of 89	PROJECT NAME		
FLOWCHART		NUMBER	ISSUE DATE	TASK NO.		
DECISION TABLE		DRAWN BY	DATE	TASK NAME		
OTHER						

A  
B  
C  
D

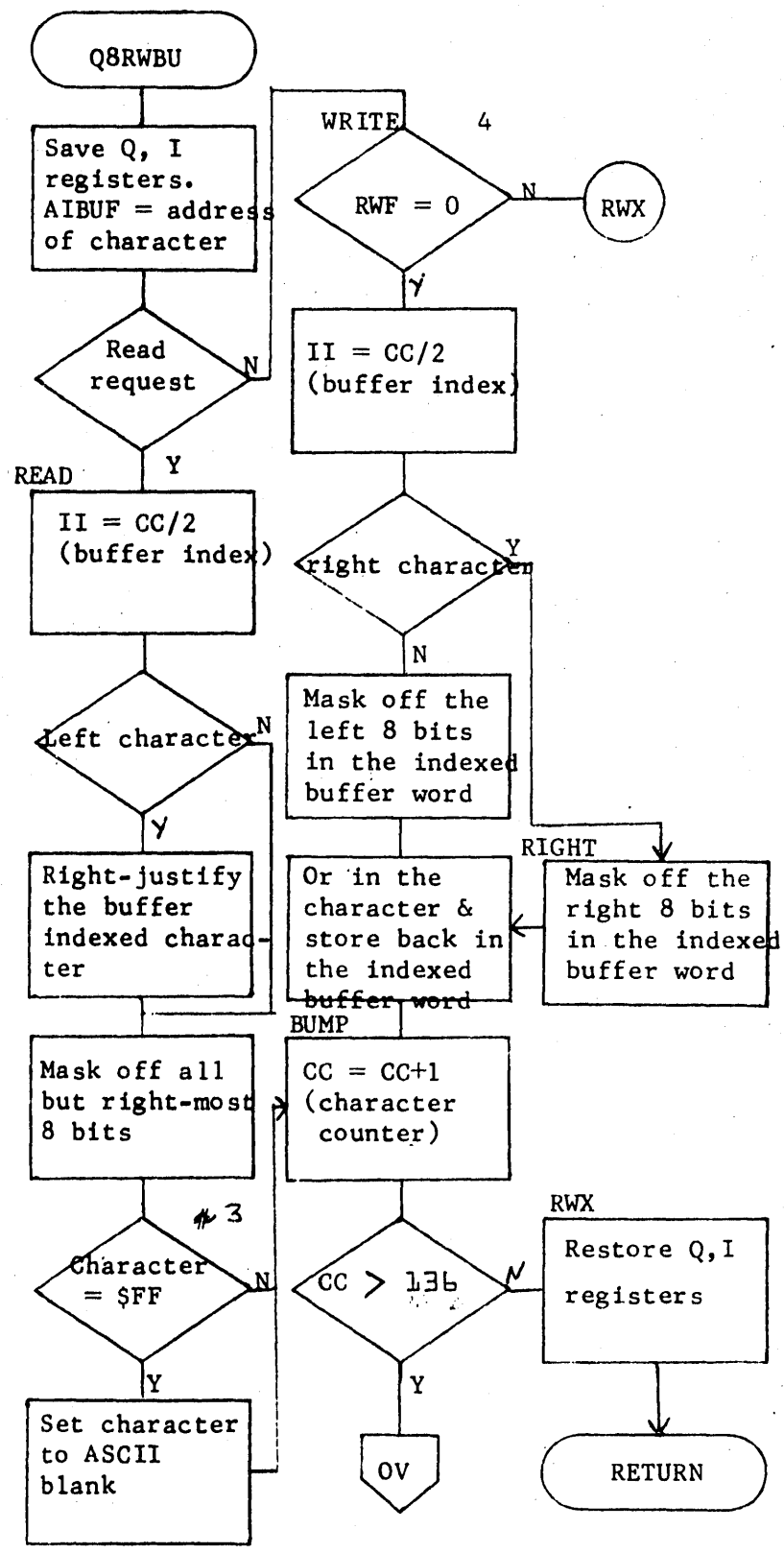
2. LOC is the beginning location of the input/output buffer.



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	Q8RWBU	PAGE 15 OF 89		PROJECT MGR.			
	NUMBER		ISSUE DATE		PROJECT NAME			
	DRAWN BY		DATE		TASK NO.			
					TASK NAME			

5-4

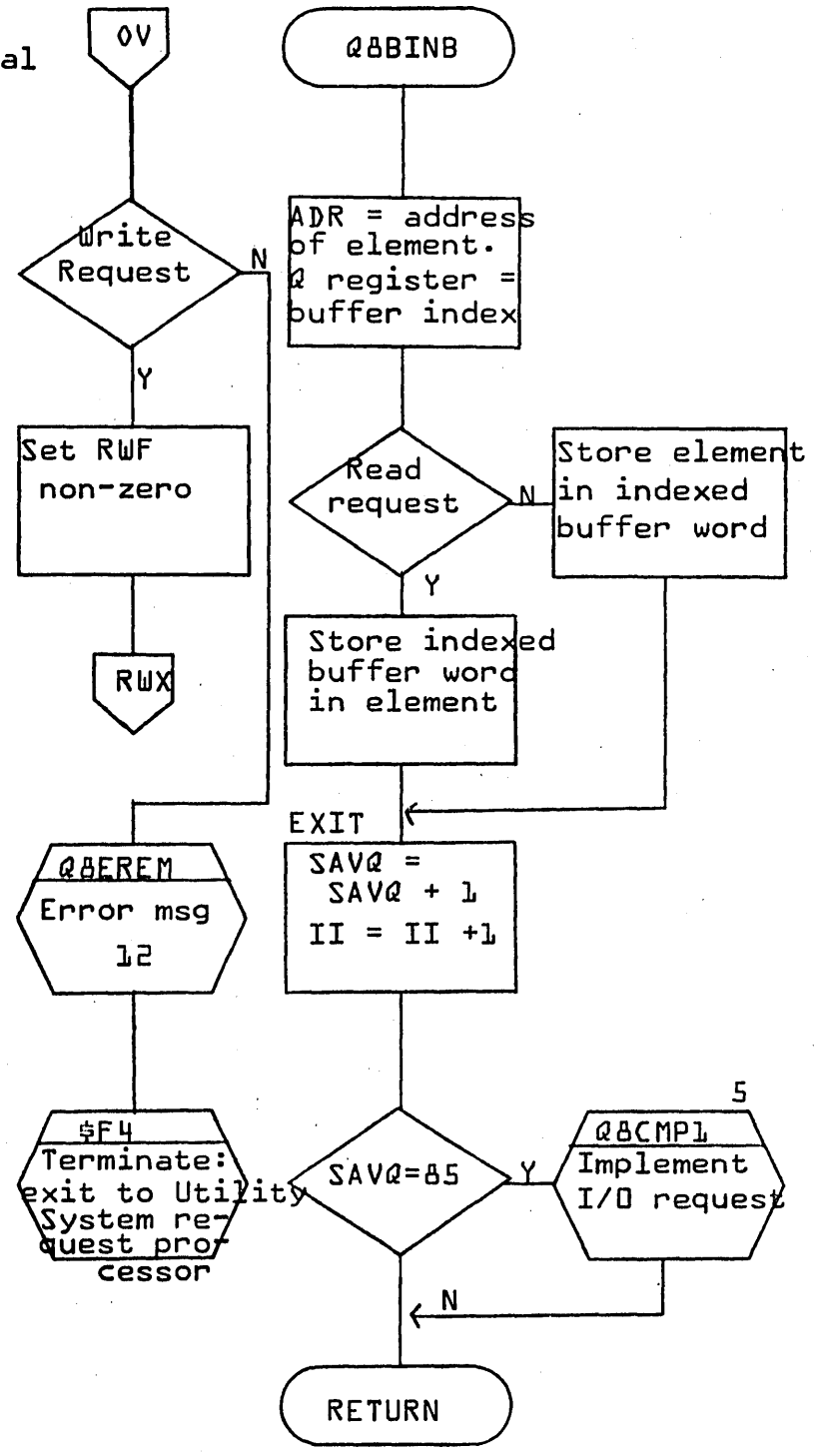
- 3. An FF<sub>16</sub> will be encountered when a read was terminated at an odd numbered character. The I/O drivers fill the word with FF<sub>16</sub>.
- 4. RWF flags more than 120 characters transferred. On a write request truncation occurs at 126 characters.



DATE	
APPROVED	
REV	
PROJECT NO.	
PROJECT MGR.	
PROJECT NAME	
TASK NO.	
TASK NAME	
DOCUMENT CLASS	MACH. TYPE <i>1/CC</i>
DOCUMENT TITLE	<i>Q8RWBU</i>
NUMBER	PAGE <i>6</i> OF <i>89</i>
DRAWN BY	ISSUE DATE
	DATE <i>5-6-77</i>
CONTROL DATA CORPORATION	
SOFTWARE DOCUMENT	
SAMPLE CODE	<input type="checkbox"/>
FLOWCHART	<input checked="" type="checkbox"/>
DECISION TABLE	<input type="checkbox"/>
OTHER	<input type="checkbox"/>



5. When the binary buffer is full, the I/O request is implemented. A physical record is read or written.



DATE	
APPROVED	
REV	
PROJECT NO.	
PROJECT MGR.	
PROJECT NAME	
TASK NO.	
TASK NAME	
DOCUMENT CLASS	IMS
DOCUMENT TITLE	Q8RWBU
MACH. TYPE	1700
PAGE 17 of 89	
ISSUE DATE	
DATE	
DRAWN BY	

CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

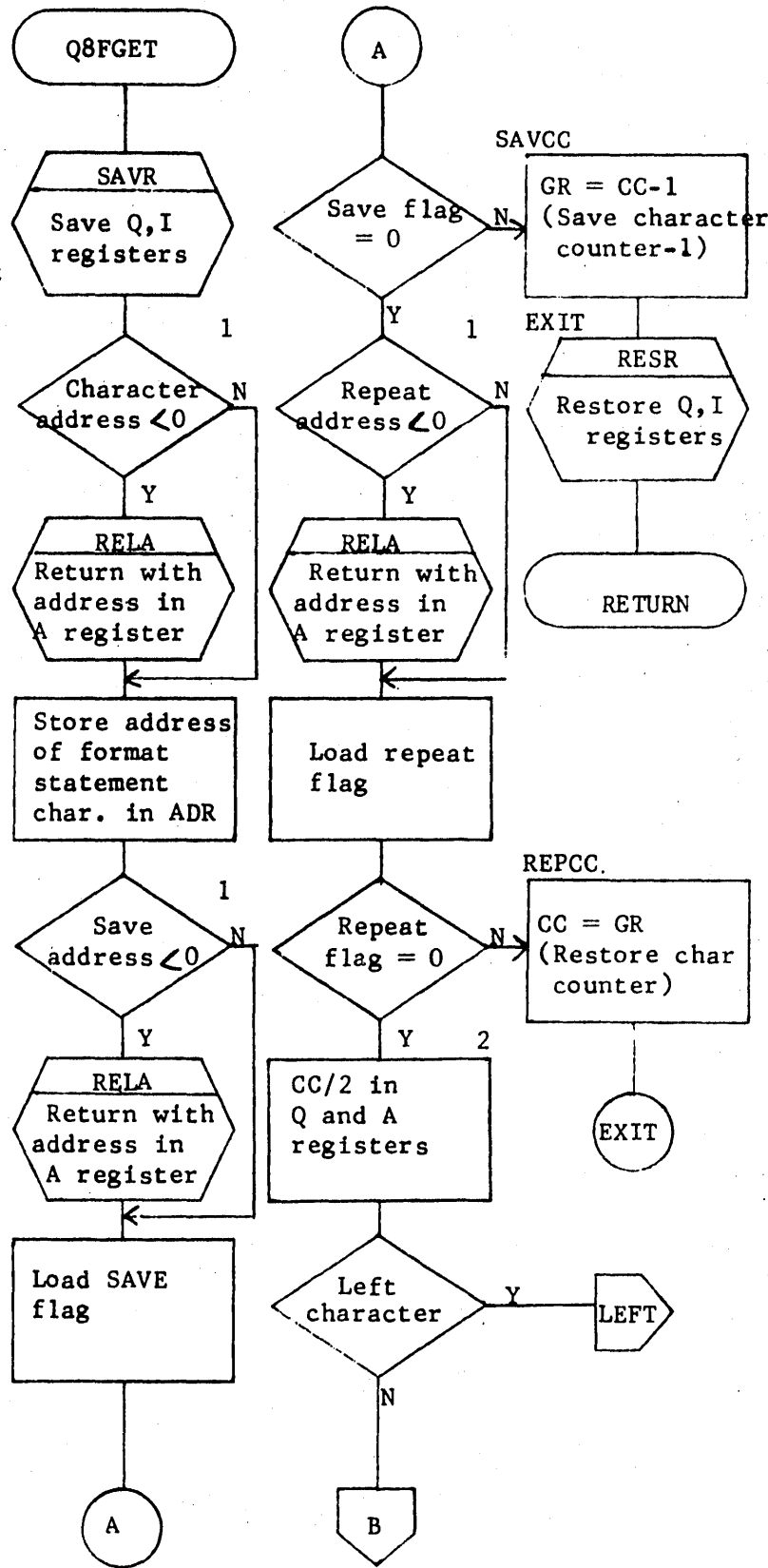
SAMPLE CODE

FLOWCHART

DECISION TABLE

OTHER

1. A negative address means this address is relative to the given address.
2. Loading the Q register with CC and shifting right one place has the effect of loading Q with the format statement word index and leaving a pointer to the right character. CC, the character counter, is odd for right characters, even for left characters.



DATE	
APPROVED	
REV	
PROJECT NO.	
PROJECT MGR.	
PROJECT NAME	
TASK NO.	
TASK NAME	
DOCUMENT CLASS	MACH. TYPE 1700
DOCUMENT TITLE	Q8FGET
NUMBER	PAGE 18 OF 89
ISSUE DATE	
DRAWN BY	DATE 5-67

**CONTROL DATA CORPORATION**

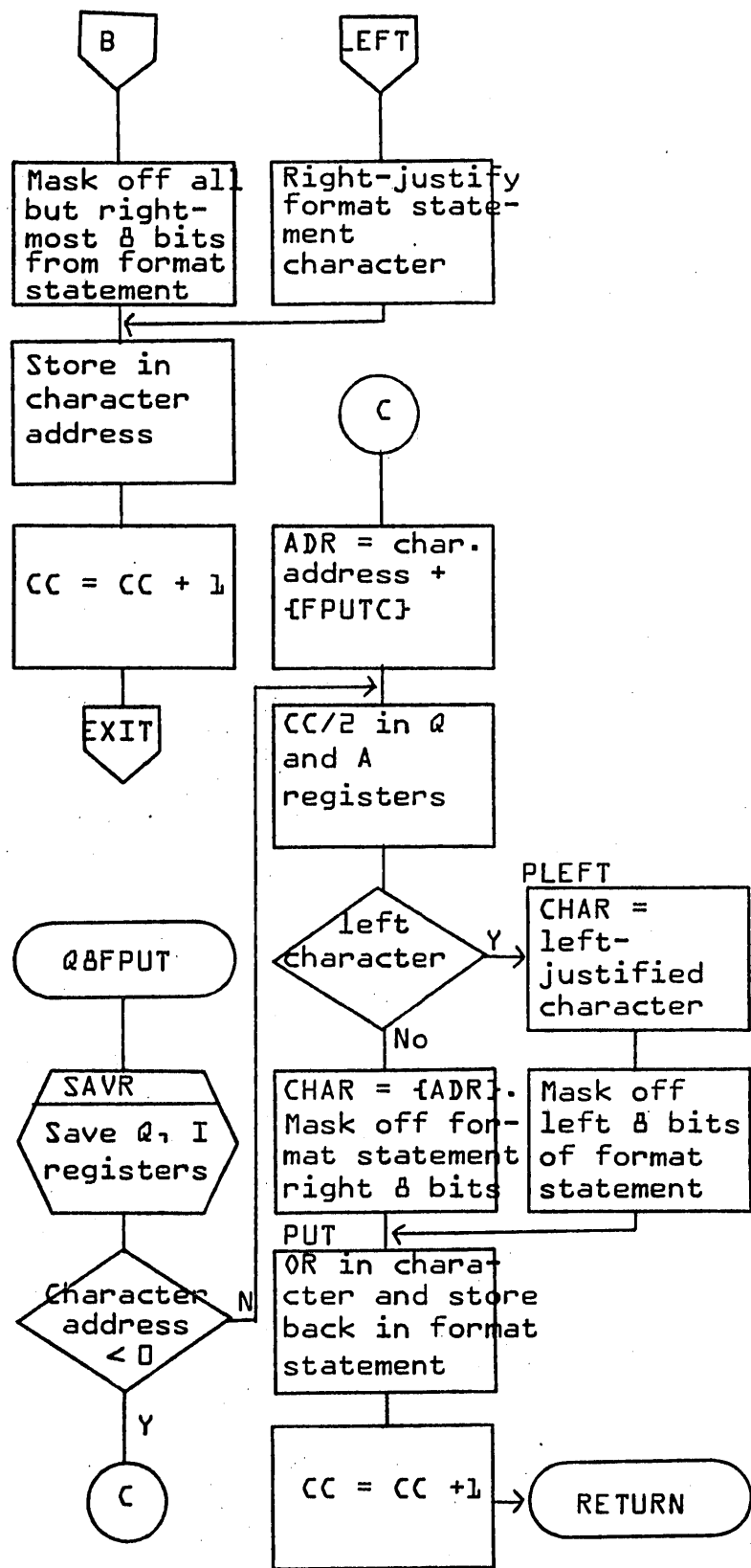
SOFTWARE DOCUMENT

SAMPLE CODE

FLOWCHART

DECISION TABLE

OTHER



CONTROL DATA CORPORATION SOFTWARE DOCUMENT	DOCUMENT CLASS IMS	MACH. TYPE 1700	PROJECT NO.	APPROVED	DATE
SAMPLE CODE FLOWCHART DECISION TABLE OTHER	DOCUMENT TITLE Q&BGET	PAGE 1 of 89	PROJECT MGR.	REV	
	NUMBER	ISSUE DATE	PROJECT NAME		
	DRAWN BY	DATE	TASK NO.		
			TASK NAME		

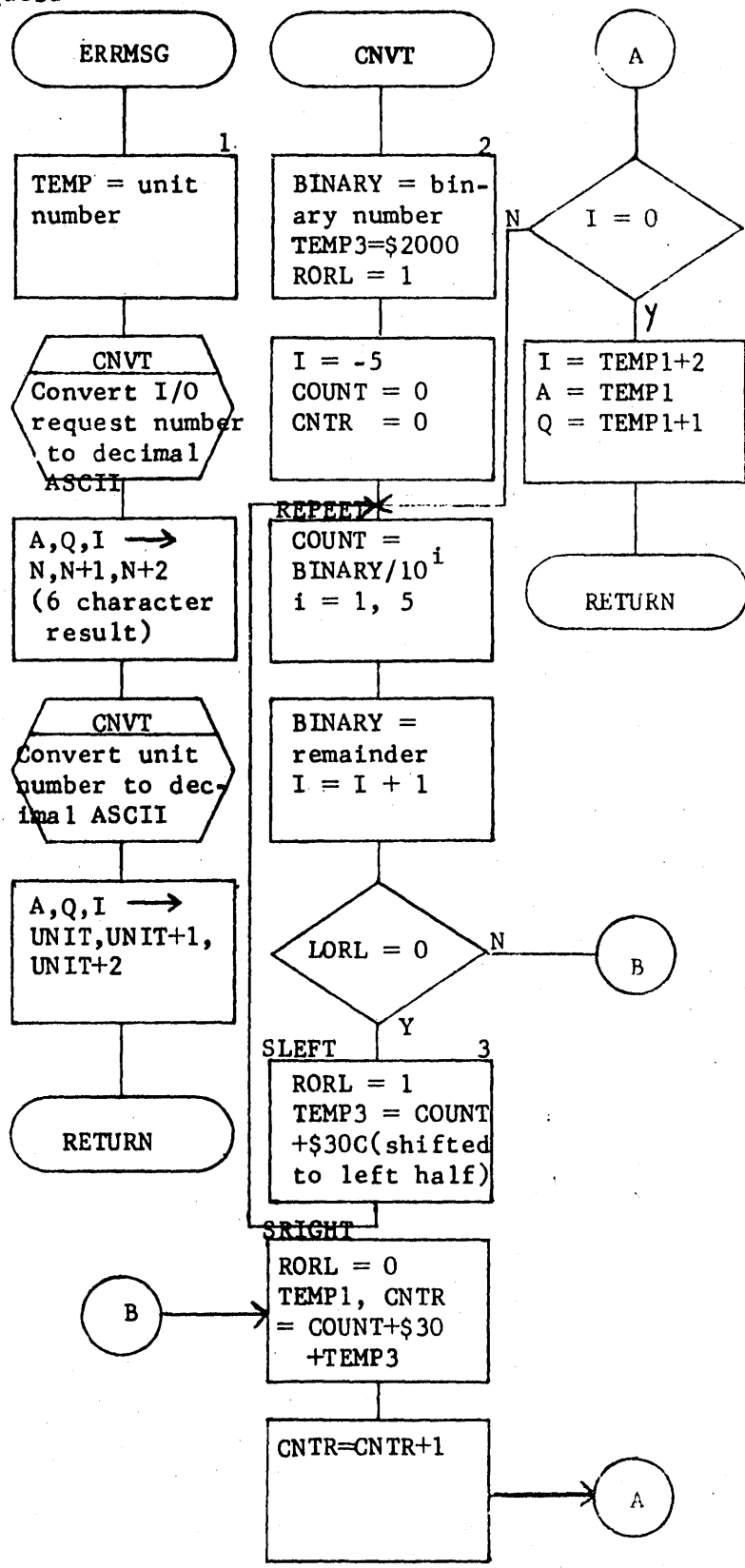
A

B

C

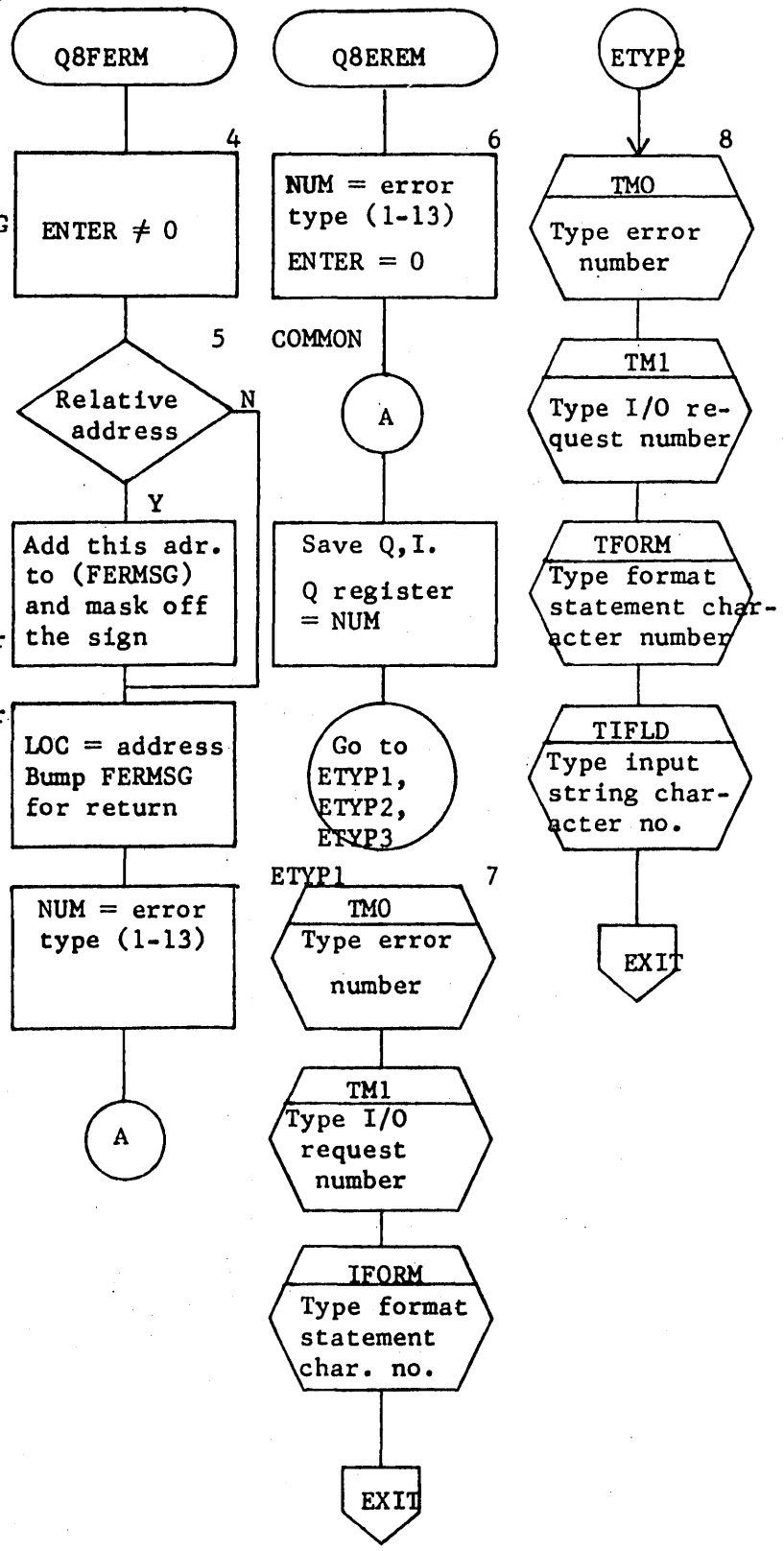
D

1. Enter with I/O request number in A and unit number in Q.
2. LORL is a flag; 1 for right half store; 0 for left half store.
3. \$30 is ASCII zero.



DATE	
APPROVED	
REV	
PROJECT NO.	
PROJECT MGR.	
PROJECT NAME	
TASK NO.	
TASK NAME	
MACH. TYPE	1700
DOCUMENT TITLE	Q8ERRM
ISSUE DATE	PAGE 20 OF 89
DATE	3-67
DOCUMENT CLASS	MS
NUMBER	
DRAWN BY	
CONTROL DATA CORPORATION	
SOFTWARE DOCUMENT	
SAMPLE CODE	<input type="checkbox"/>
FLOWCHART	<input checked="" type="checkbox"/>
DECISION TABLE	<input type="checkbox"/>
OTHER	<input type="checkbox"/>

- 4. FERMSG is entered from the format package which is coded in FORTRAN. The flag, ENTER, is set non-zero to indicate FERMSG was entered.
- 5. A negative address means this address is relative to the given address.
- 6. ERRMSG is entered from the 1700 machine language routines of the I/O package.
- 7. ETYP1 is the error types 1, 12, 13.
- 8. ETYP2 is the error format for error types 2 and 3.



	DATE				
	APPROVED				
	REV				
PROJECT NO.	PROJECT MGR.	PROJECT NAME	TASK NO.	TASK NAME	
MACH. TYPE	1700	PAGE 21 of 89	ISSUE DATE	DATE 3-6-67	
DOCUMENT CLASS	Q8ERRM	NUMBER	DRAWN BY		
DOCUMENT TITLE					

**CONTROL DATA CORPORATION**

SOFTWARE DOCUMENT

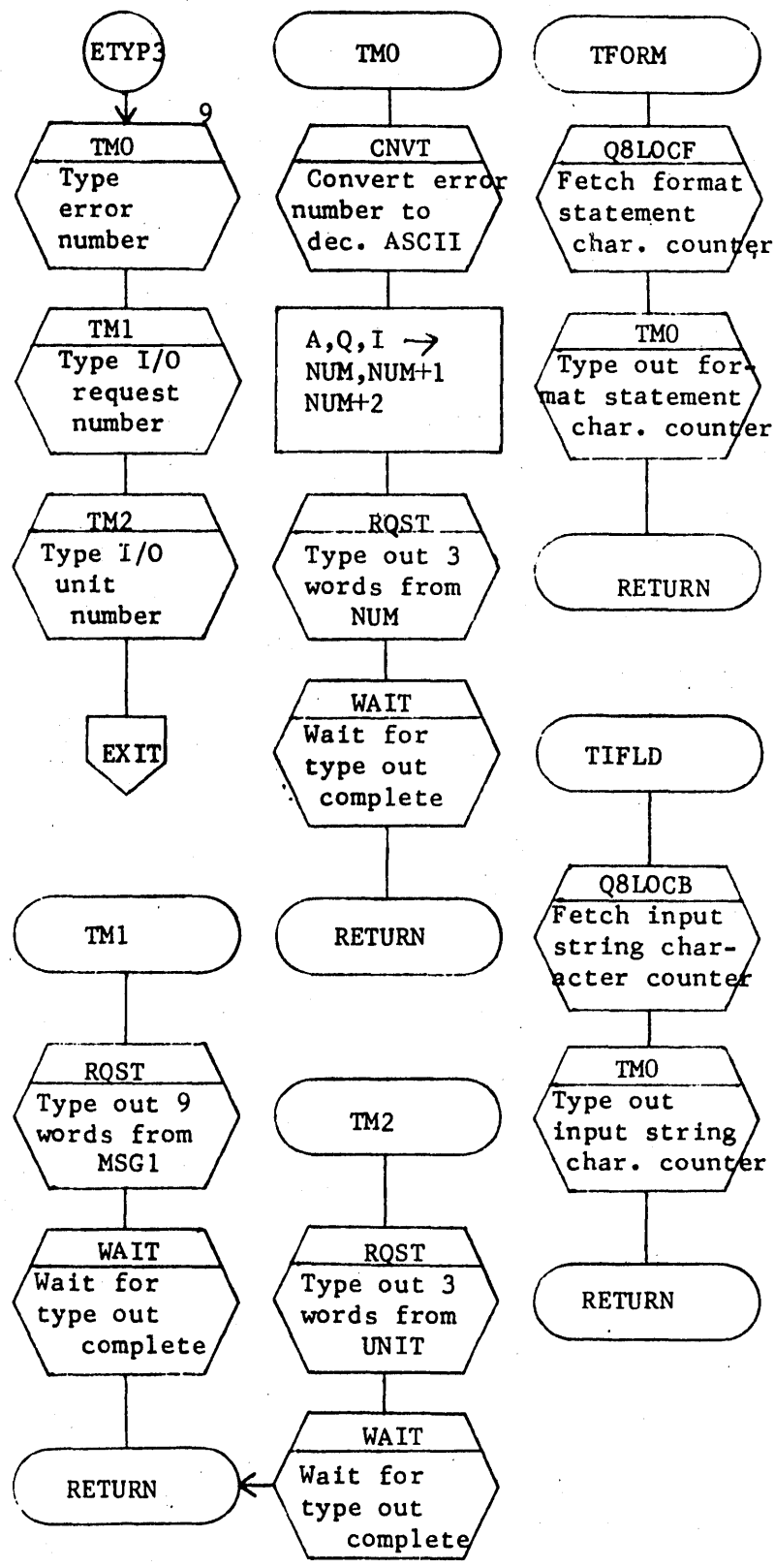
SAMPLE CODE

FLOWCHART

DECISION TABLE

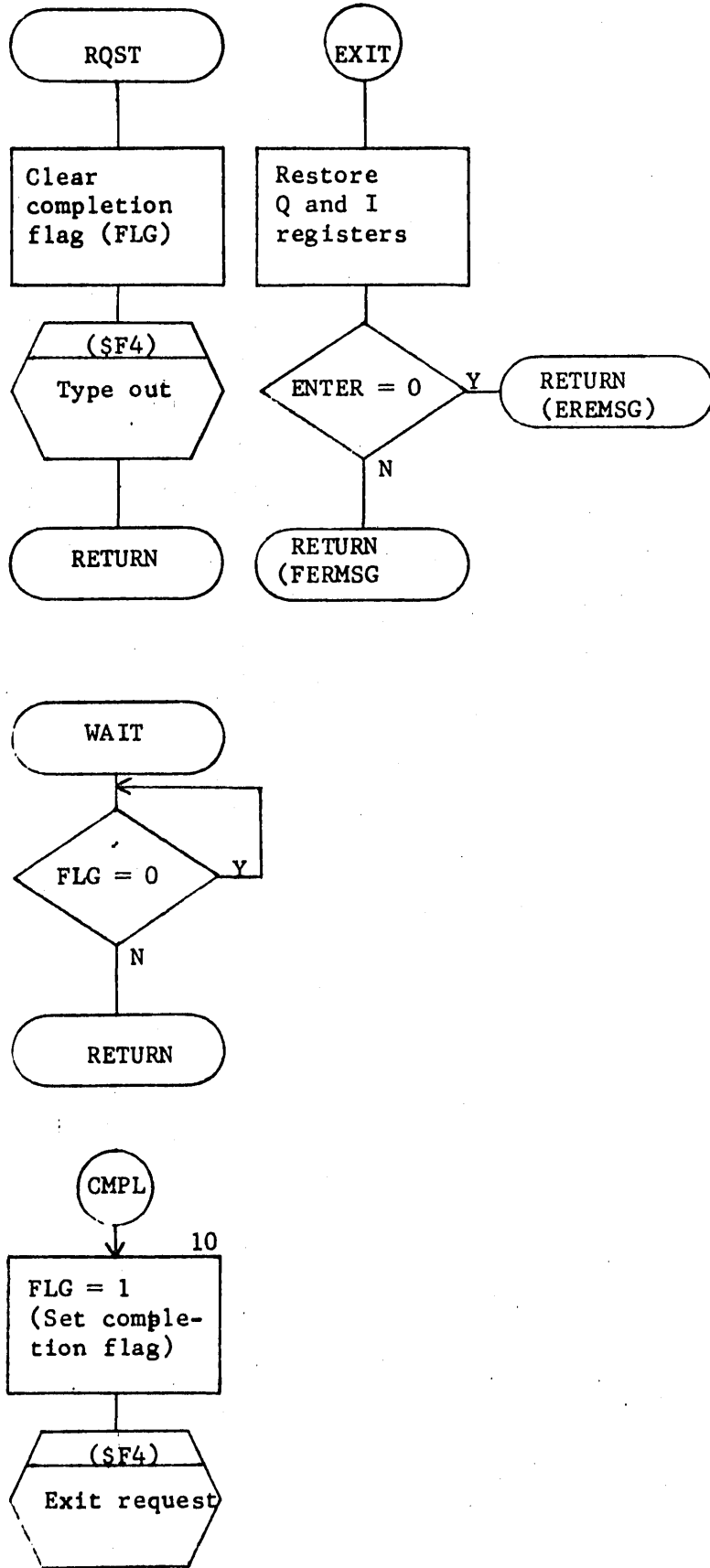
OTHER

9. ETYP3 is the error format for error types 4, 5, 6, 7, 8, 9, 10, and 11.



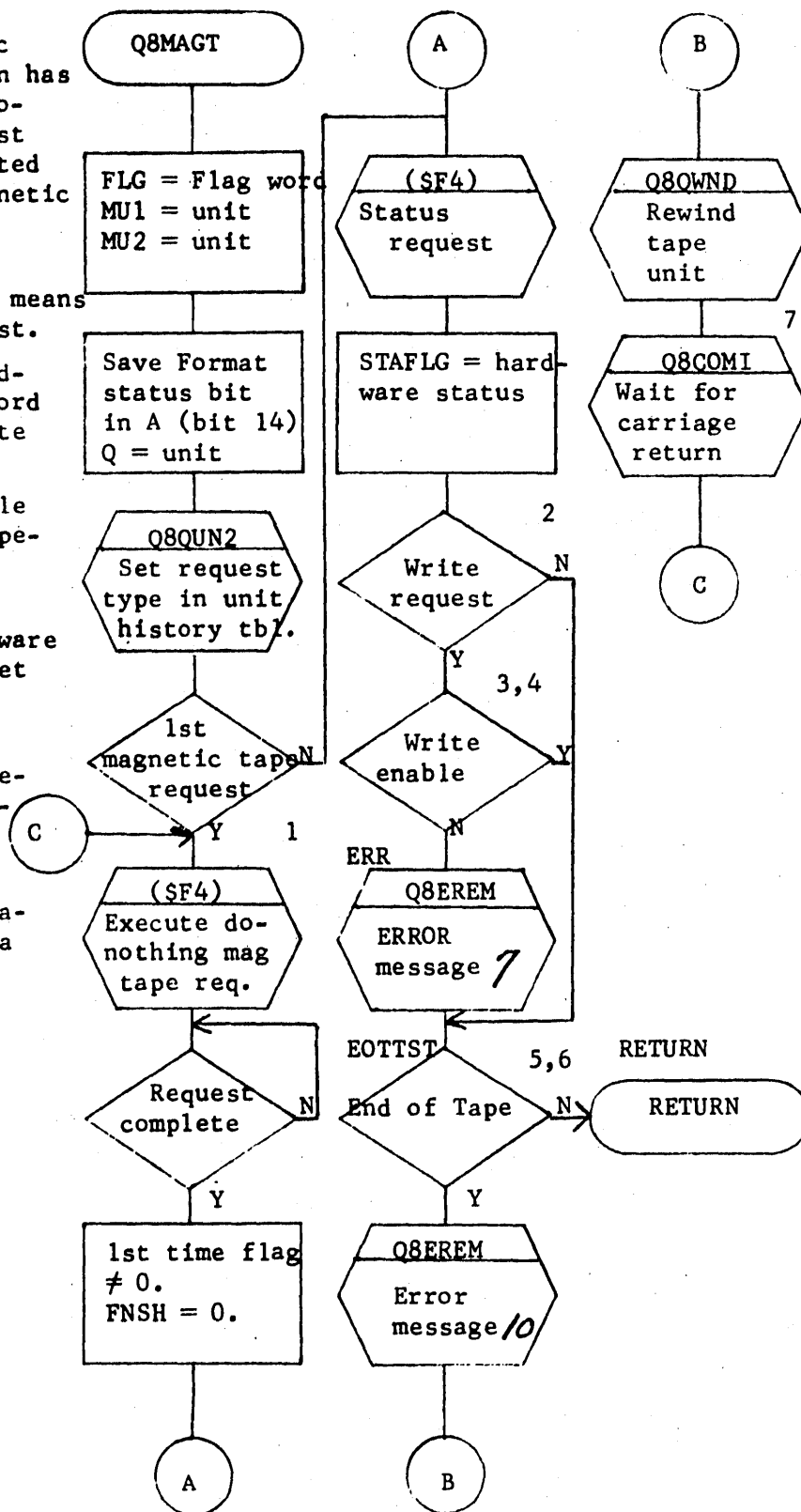
DATE	
APPROVED	
REV	
PROJECT NO.	
PROJECT MGR.	
PROJECT NAME	
TASK NO.	
TASK NAME	
DOCUMENT CLASS	
DOCUMENT TITLE	
MACH. TYPE	
Q8ERRM NUMBER	89
ISSUE DATE	
DATE	
DRAWN BY	
<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT SAMPLE CODE <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> FLOWCHART DECISION TABLE OTHER	

10. CMPL is the completion address entry point from the System request.



CONTROL DATA CORPORATION		DOCUMENT CLASS	MACH. TYPE	PROJECT NO.	APPROVED	DATE
SOFTWARE DOCUMENT		<i>IMC</i>	<i>1100</i>			
SAMPLE CODE	DOCUMENT TITLE	Q8ERRM	PAGE 23 OF 89	PROJECT MGR.		
<input type="checkbox"/>	FLOWCHART		ISSUE DATE	PROJECT NAME		
<input checked="" type="checkbox"/>	DECISION TABLE		DATE <i>3-1-89</i>	TASK NO.		
<input type="checkbox"/>	OTHER		DRAWN BY	TASK NAME		

1. If no magnetic tape operation has occurred, a do-nothing request must be executed to obtain magnetic tape status.
2. Bit 9 of the flag word set means a write request.
3. Bit 15 of hardware status word set means write is enabled.
4. No write enable results in type-out of error message 7.
5. Bit 9 of hardware status word set means end of tape.
6. End of tape results in type-out of error message 10.
7. Wait for operator to mount a new tape.



DATE	
APPROVED	
REV	
PROJECT NO.	
PROJECT MGR.	
PROJECT NAME	
TASK NO.	
TASK NAME	
MACH TYPE	1144
DOCUMENT TITLE	Q8MAGT
ISSUE DATE	PAGE 2 of 89
DATE	5/6/89
DRAWN BY	
CONTROL DATA CORPORATION	
SOFTWARE DOCUMENT	
SAMPLE CODE	<input type="checkbox"/>
FLOWCHART	<input checked="" type="checkbox"/>
DECISION TABLE	<input type="checkbox"/>
OTHER	<input type="checkbox"/>



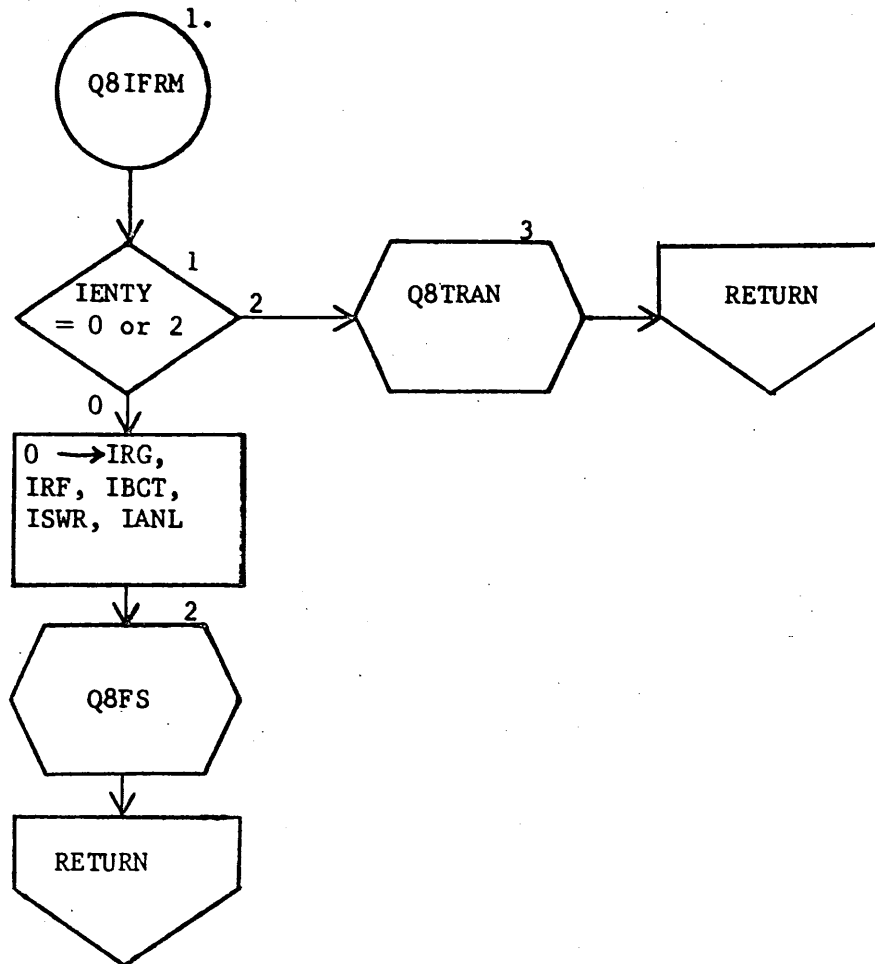
1. This routine is called from Q8QINI and Q8QX. The calling sequence to this routine has two parameters. They are:

IENTRY - 0 if routine called from Q8QINI

IENTRY - 1 if routine called from Q8QX.

2. Call format scanner. This routine interprets FORMAT statement parameters and sets them up in a form that the transmission routine can use.

3. Call transmission routine. This routine converts the next I/O field according to the format specifications.



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE   
FLOWCHART   
DECISION TABLE   
OTHER

DOCUMENT CLASS	MACH. TYPE	PROJECT NO.	REV	APPROVED	DATE
141	1100				
DOCUMENT TITLE		PROJECT MGR.			
Q8IFRM	PAGE 25 OF 89	PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.			
DRAWN BY	DATE 5-6-74	TASK NAME			

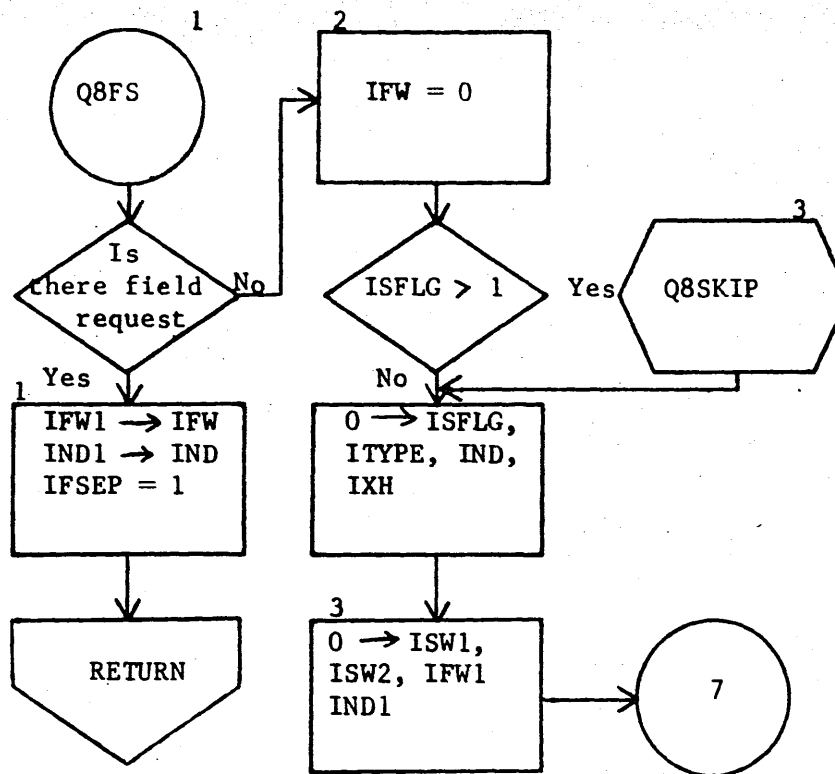
1. This routine is called from Q8IFRM and Q8TRAN. Its function is to interpret the FORMAT statement and put subfield parameters into boxes accessible by the transmit routine Q8TRAN. The parameters passed are as follows:

- IFW = field width
- IND = no. decimal places
- ITYPE = type of conversion
- IRF = field repeat count
- IRG = grout repeat count
- IBCT = bracket count
- ISWR = I/O switch
- ILIST = FORMAT statement accompanied by I/O list
- IANYL = switch to denote existence of some conversion specification.

2. Gets next character from read buffer or puts next character in output buffer. Its 3 parameters are:

- I3 = next character loc.
- I4 = save char. position/ or No.
- I5 = reposition/or no

3. Routine initiates I/O.



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

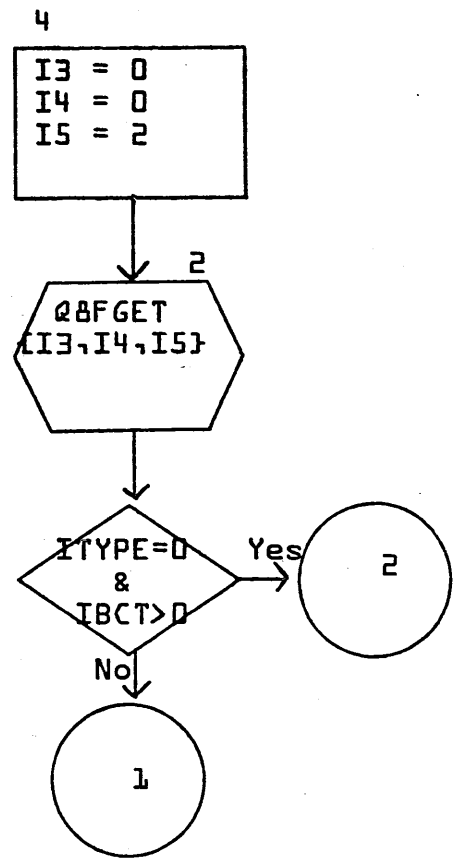
DOCUMENT CLASS <i>ZMS</i>	MACH TYPE <i>1/100</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE <b>Q8FS</b>	PAGE 26 OF 89	PROJECT MGR.			
NUMBER	ISSUE DATE	TASK NO.			
DRAWN BY	DATE <i>5-6-74</i>	TASK NAME			

A

B

C

D



**CONTROL DATA CORPORATION**  
 SOFTWARE DOCUMENT

SAMPLE CODE

FLOWCHART

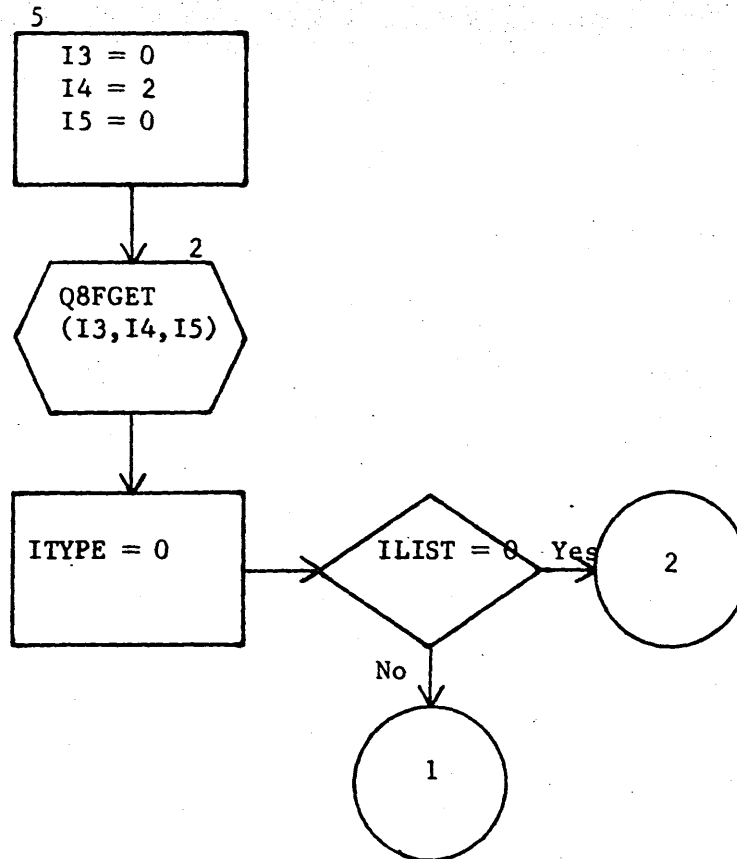
DECISION TABLE

OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	QBF S			PROJECT MGR.			
NUMBER	PAGE 27 OF 89			PROJECT NAME			
	ISSUE DATE	TASK NO.					
DRAWN BY	DATE	TASK NAME					

6-59

4. Error 1 - Invalid character in FORMAT statement.



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE

FLOWCHART

DECISION TABLE

OTHER

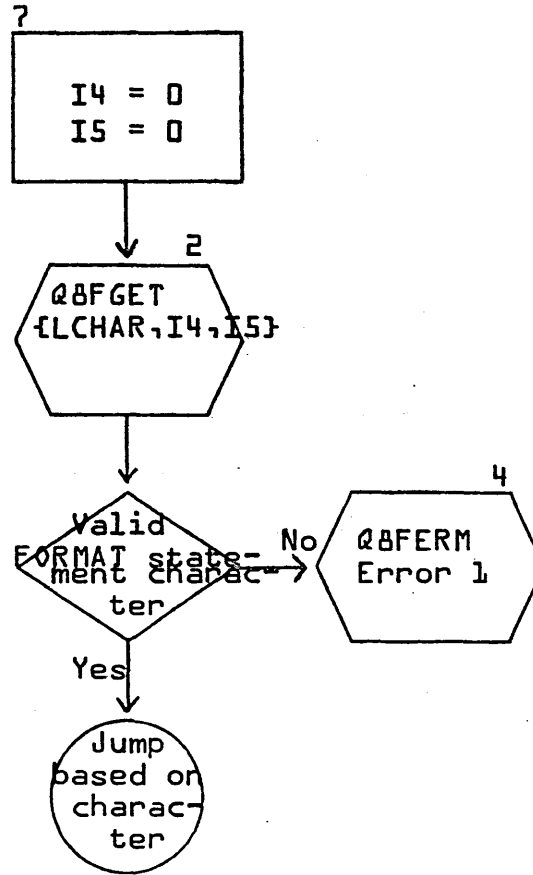
DOCUMENT CLASS	<i>IMS</i>	MACH. TYPE	<i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE				PROJECT MGR.			
NUMBER	<i>Q8FS</i>	PAGE	<i>28</i> OF <i>89</i>	PROJECT NAME			
		ISSUE DATE		TASK NO.			
DRAWN BY		DATE	<i>3-67</i>	TASK NAME			

A

B

C

D



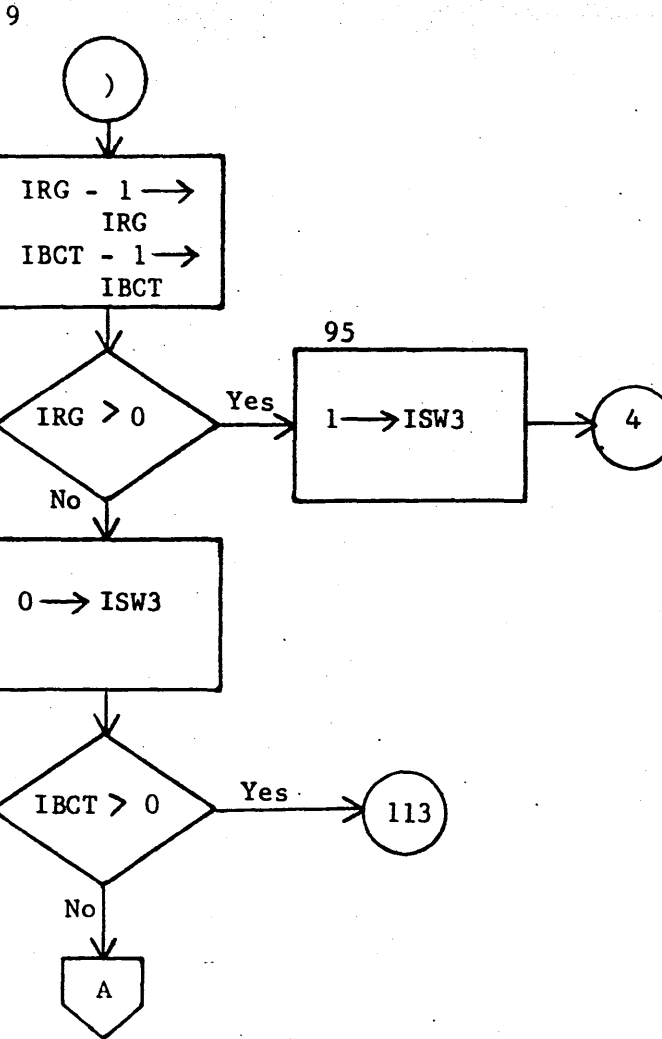
**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	Q8FS	PAGE	29 OF 89	PROJECT MGR.			
NUMBER		ISSUE DATE		PROJECT NAME			
DRAWN BY		DATE		TASK NO.			
				TASK NAME			

6-67

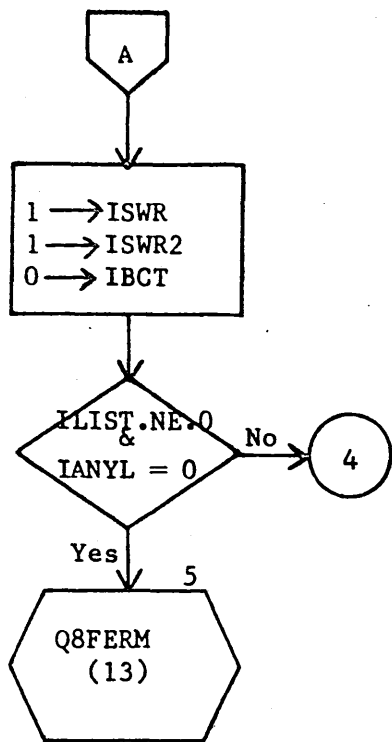
5. Error 13 - I/O list with READ, WRITE statement but no conversion specification in FORMAT statement.



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE   
FLOWCHART   
DECISION TABLE   
OTHER

DOCUMENT CLASS <i>IMS</i>	MACH. TYPE <i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE <i>Q8FS</i>	PAGE 30 OF 89	PROJECT MGR.			
NUMBER	ISSUE DATE	PROJECT NAME			
DRAWN BY	DATE <i>5-67</i>	TASK NO.			
		TASK NAME			

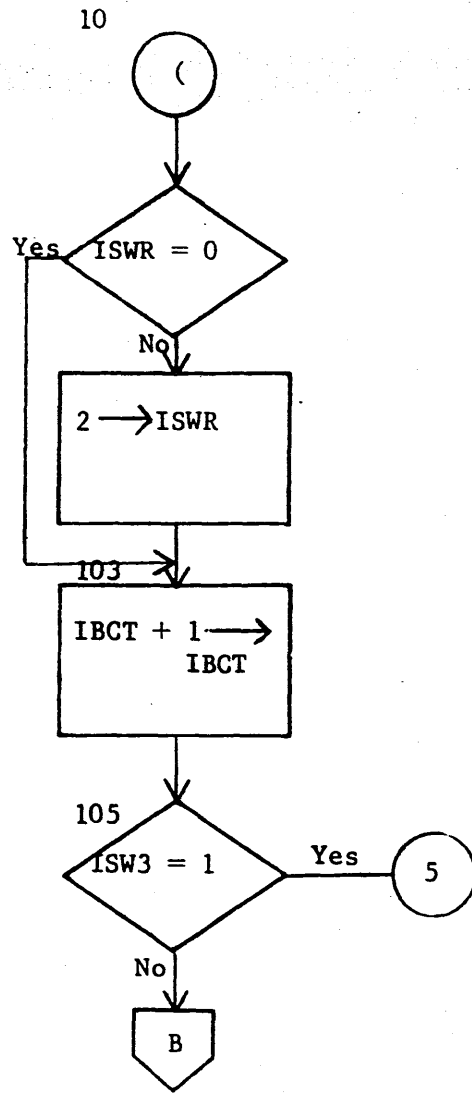


**CONTROL DATA CORPORATION**

SOFTWARE DOCUMENT

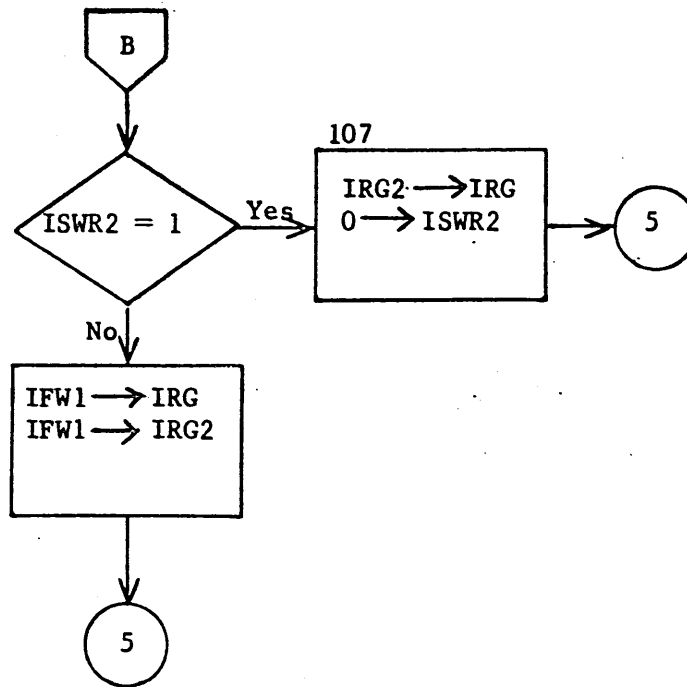
- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS <i>IMS</i>	MACH. TYPE <i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE		PROJECT MGR.			
<i>Q8FS</i>	<i>PAGE 31 OF 89</i>	PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.			
DRAWN BY	DATE <i>3-67</i>	TASK NAME			



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	MACH. TYPE	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE		PROJECT MGR.			
	NUMBER	ISSUE DATE	PROJECT NAME			
	DRAWN BY	DATE	TASK NO.			
			TASK NAME			



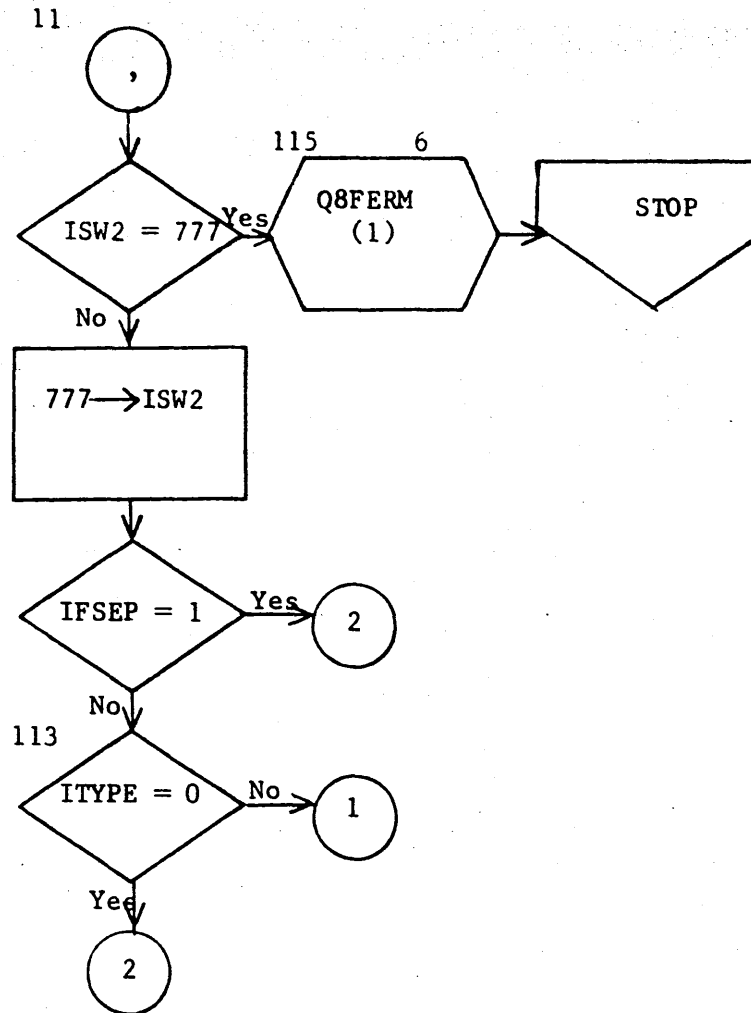


**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

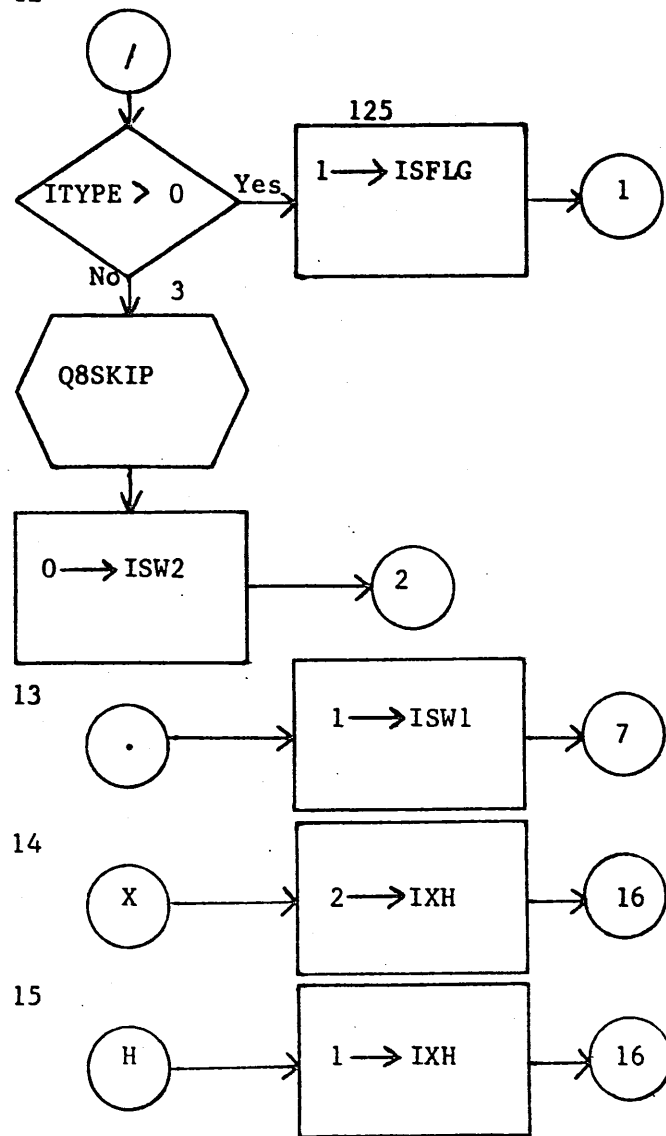
DOCUMENT CLASS <i>ZMS</i>	MACH. TYPE <i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE		PROJECT MGR.			
<b>Q8FS</b>	PAGE <i>33</i> OF <i>89</i>	PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.			
DRAWN BY	DATE <i>5-67</i>	TASK NAME			

6. Error 1 - two consecutive commas in FORMAT specification is not allowed.



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	MACH. TYPE	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE		PROJECT MGR.			
	08FS	PAGE 34 OF 89	PROJECT NAME			
	NUMBER	ISSUE DATE	TASK NO.			
	DRAWN BY	DATE 3-6-7	TASK NAME			

12



13

14

15

**CONTROL DATA CORPORATION**

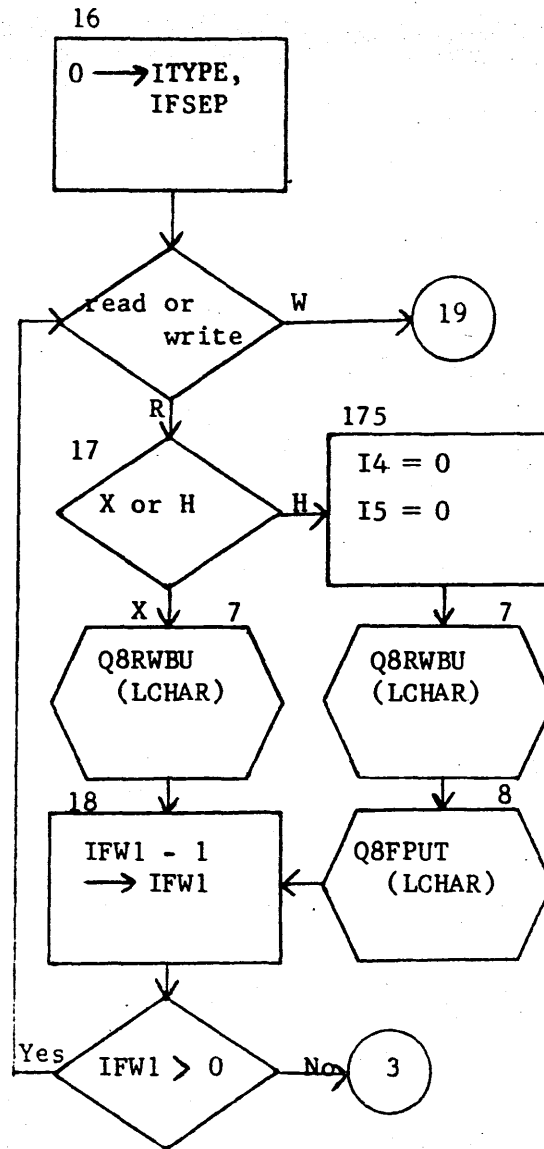
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	MACH. TYPE	PROJECT NO.	REV	APPROVED	DATE
ZM2	1706				
DOCUMENT TITLE		PROJECT MGR.			
Q8FS	PAGE 35 OF 89	PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.			
DRAWN BY	DATE 3-6-71	TASK NAME			

7. This routine gets the next character from the input buffer or puts the next character into the output buffer.

8. This routine puts a character into the next position of the FORMAT statement.

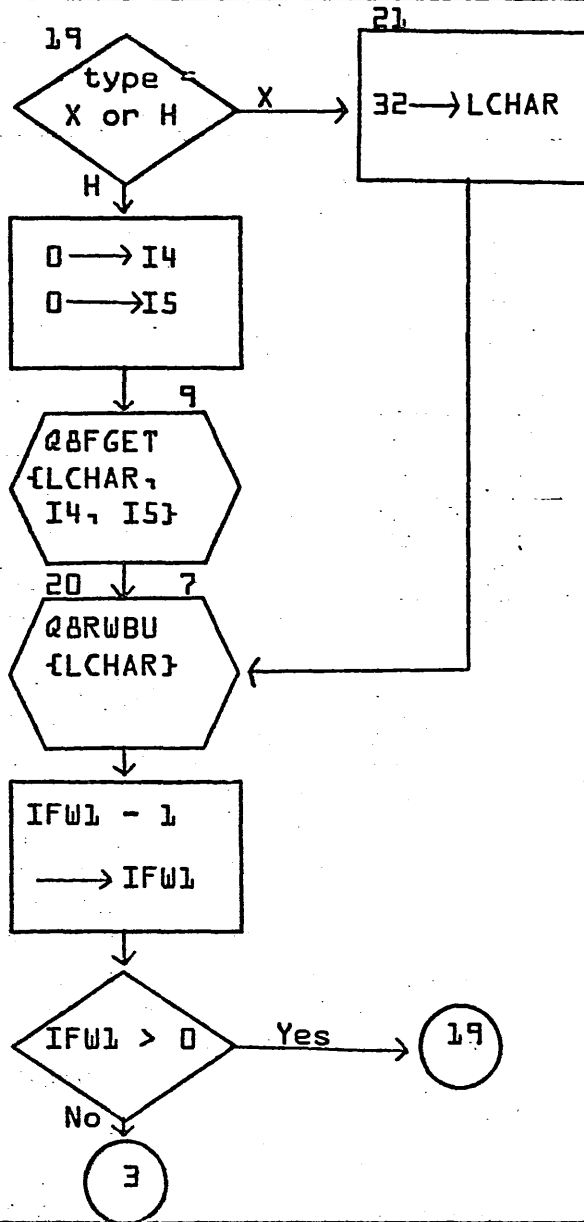


CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE   
FLOWCHART   
DECISION TABLE   
OTHER

DOCUMENT CLASS	MACH. TYPE	PROJECT NO.	REV	APPROVED	DATE
IMS	1700				
DOCUMENT TITLE		PROJECT MGR.			
Q8FS	PAGE 36 OF 89	PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.			
DRAWN BY	DATE 3-67	TASK NAME			

9. This routine gets the next character from the FORMAT statement.

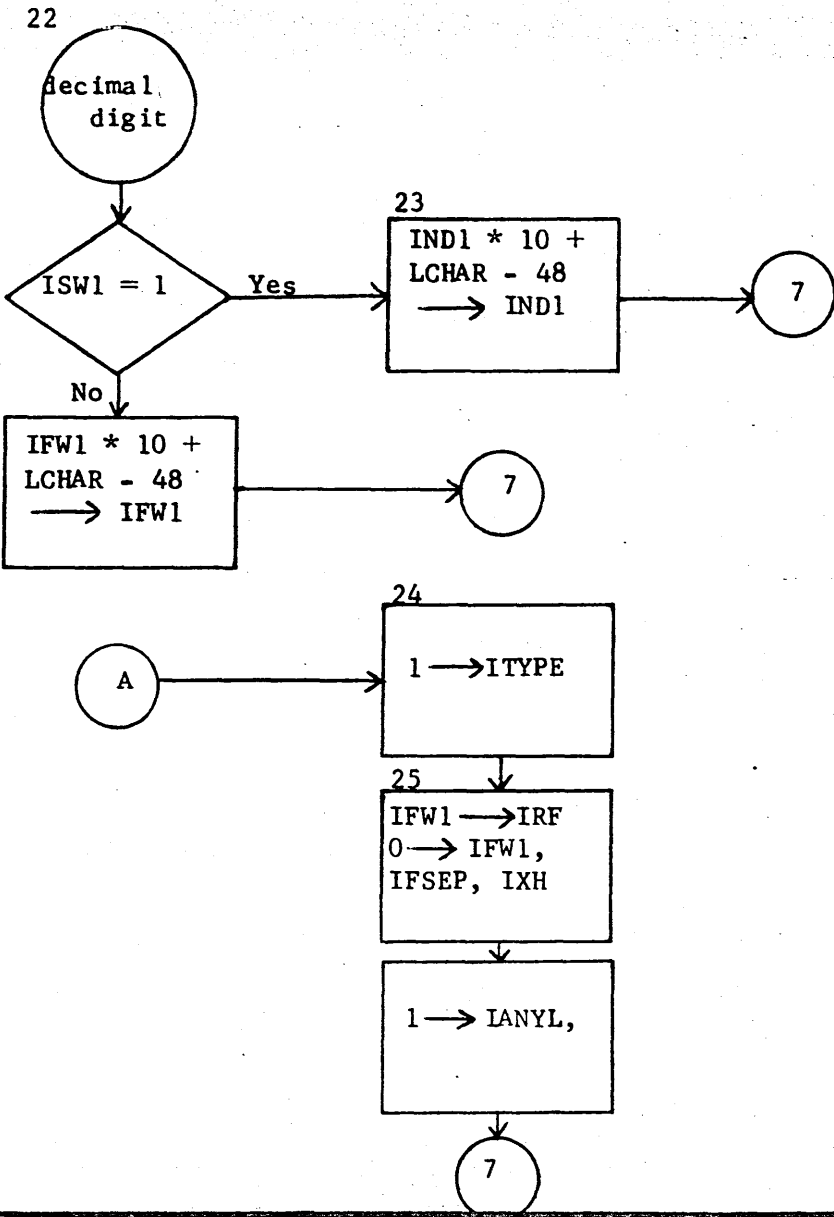


CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	Q8FS			PROJECT MGR.			
NUMBER		ISSUE DATE	PAGE 37 OF 89	PROJECT NAME			
DRAWN BY		DATE		TASK NO.			
				TASK NAME			

5-69-9

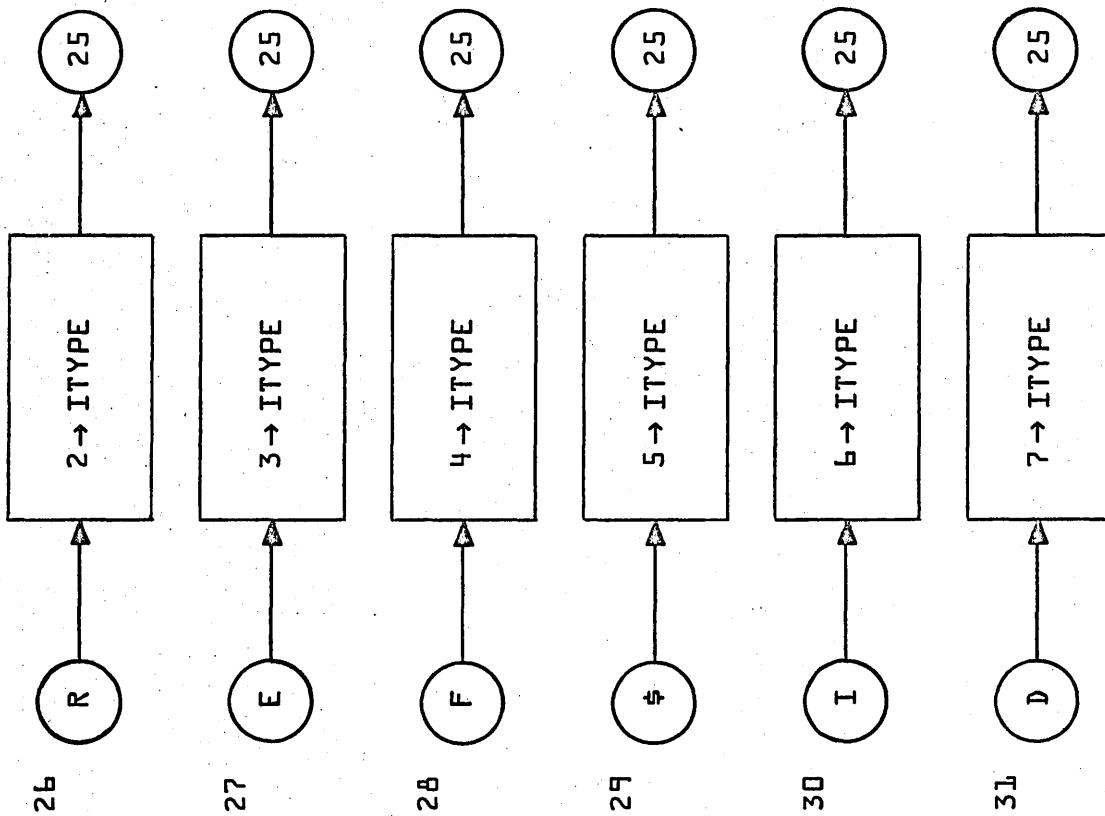


**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

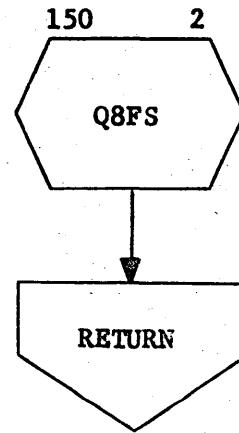
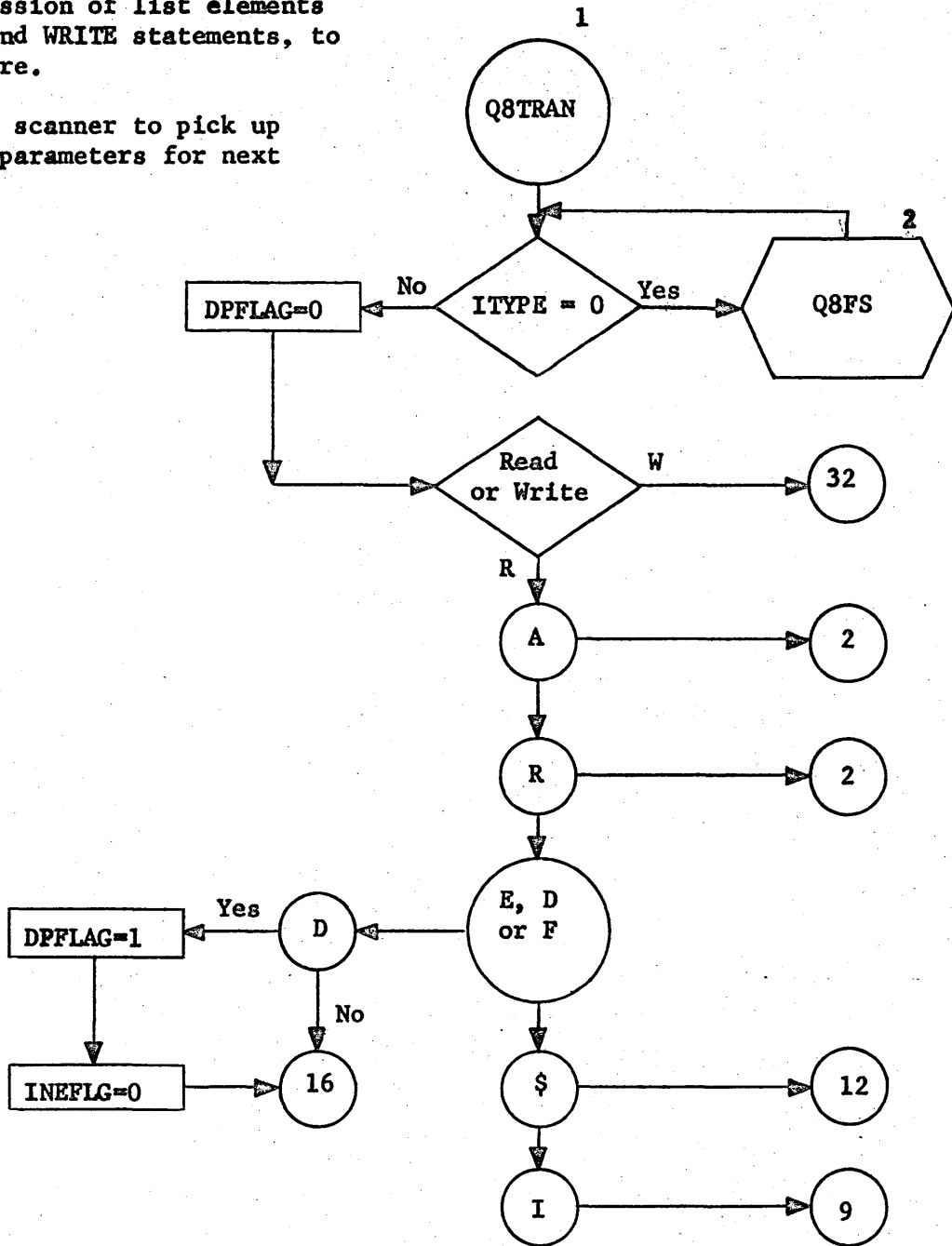
- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS <i>IMS</i>	MACH. TYPE <i>1500</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE <b>Q8FS</b>	PAGE <b>38</b> OF <b>89</b>	PROJECT MGR.			
NUMBER	ISSUE DATE	TASK NO.			
DRAWN BY	DATE <i>3-6-54</i>	TASK NAME			

DOCUMENT CLASS IMS PAGE NO. 6-71  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700



1. This routine controls the conversion and transmission of list elements from READ and WRITE statements, to and from core.
2. Call format scanner to pick up conversion parameters for next subfield.

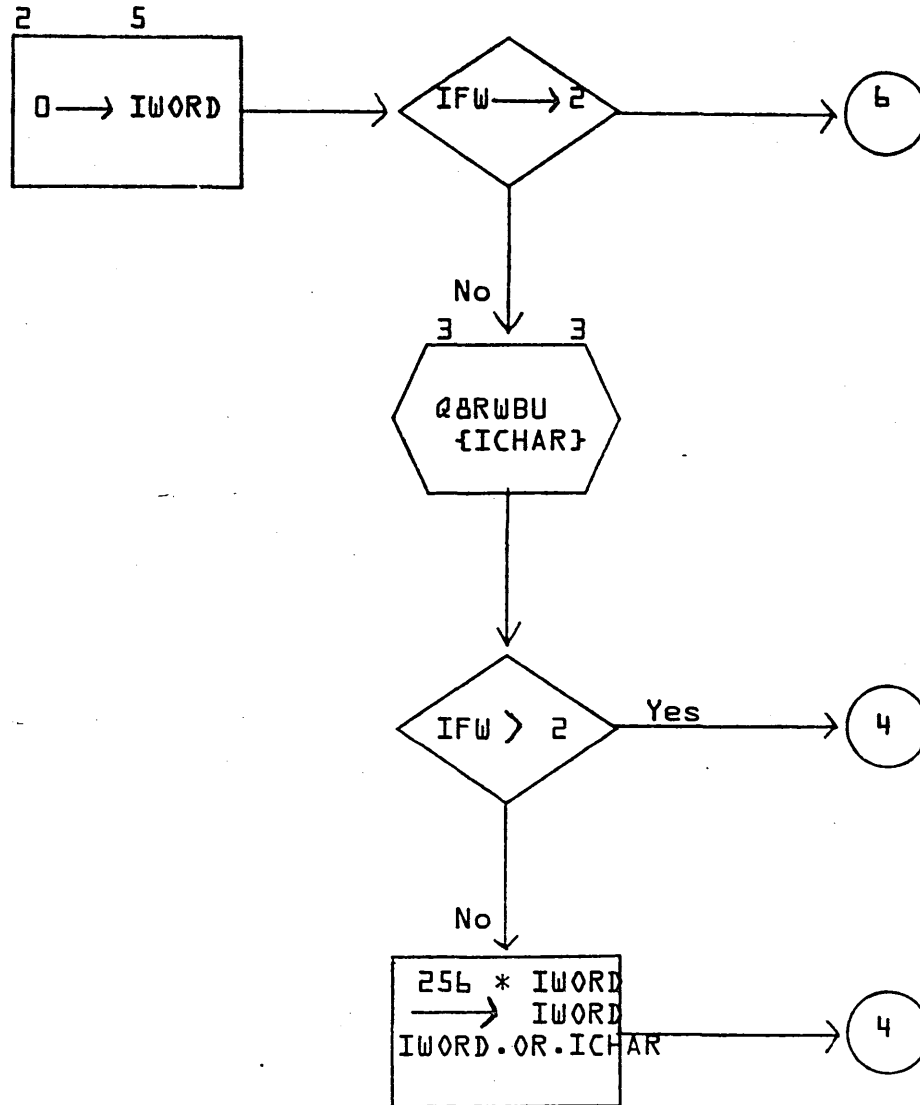


DOCUMENT CLASS IMS PAGE NO. E-72  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. CD05\*3.1 A/B MACHINE SERIES 1700

CONTROL DATA CORPORATION  
 LA JOLLA RESOURCE CENTER  
 DIVISION



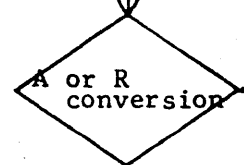
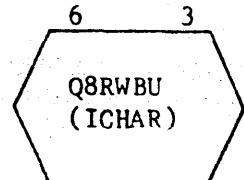
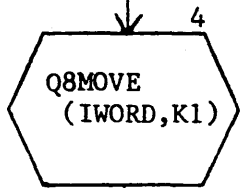
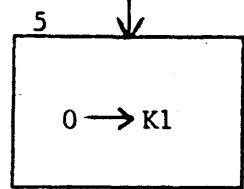
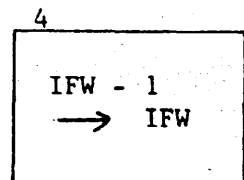
- 3. Pick up next character from input string or on output, place next character into output buffer.
- 4. This routine moves one word to or from core based on nature of I/O currently being executed. The first parameter is the word to be moved, and the second parameter is the increment to be added to the base location to determine the actual core location.
- 5. A and R input core - versions comes here.



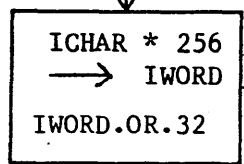
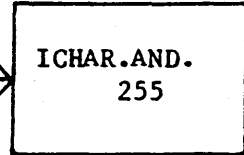
**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	Q8TRAN			PROJECT MGR.			
NUMBER		ISSUE DATE	PAGE 41 OF 89	PROJECT NAME			
DRAWN BY		DATE		TASK NO.			
				TASK NAME			



R

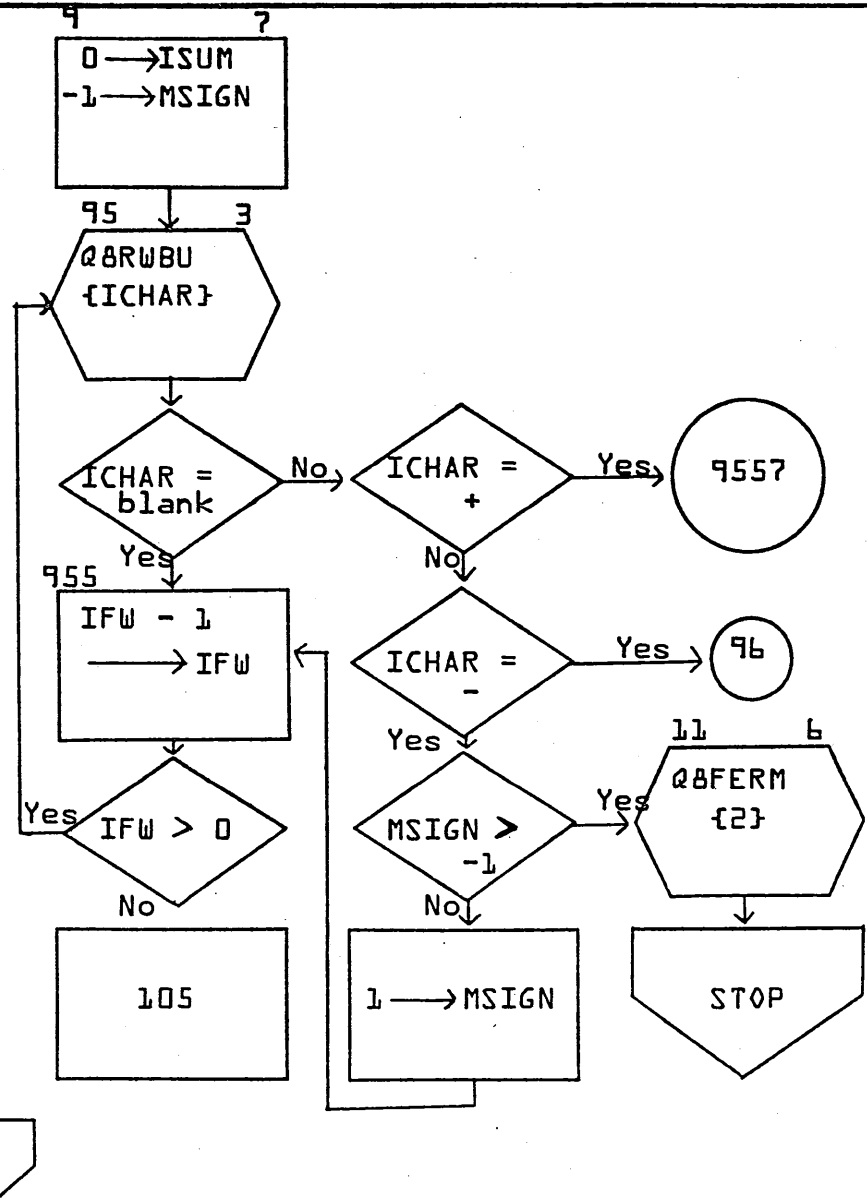


**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	LM	MACH. TYPE	700	PROJECT NO.		REV		APPROVED		DATE
DOCUMENT TITLE	Q8TRAN			PROJECT MGR.						
NUMBER		ISSUE DATE	PAGE 42 OF 89	PROJECT NAME						
DRAWN BY		DATE	3-17	TASK NO.						
				TASK NAME						

6. Error 2. Bad character in input string.  
 7. Integer input conversion comes here.



C

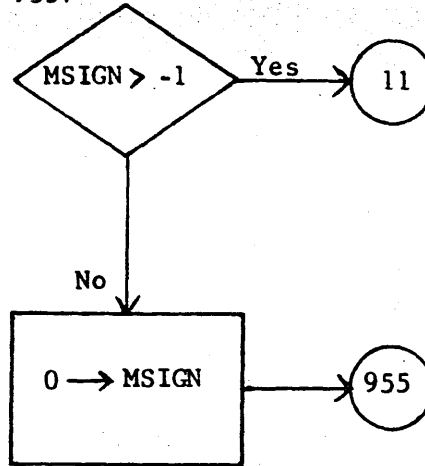
CONTROL DATA CORPORATION  
 SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	Q8TRAN			PROJECT MGR.			
NUMBER		ISSUE DATE	PAGE 43 OF 89	PROJECT NAME			
DRAWN BY		DATE		TASK NO.			
				TASK NAME			

C

9557

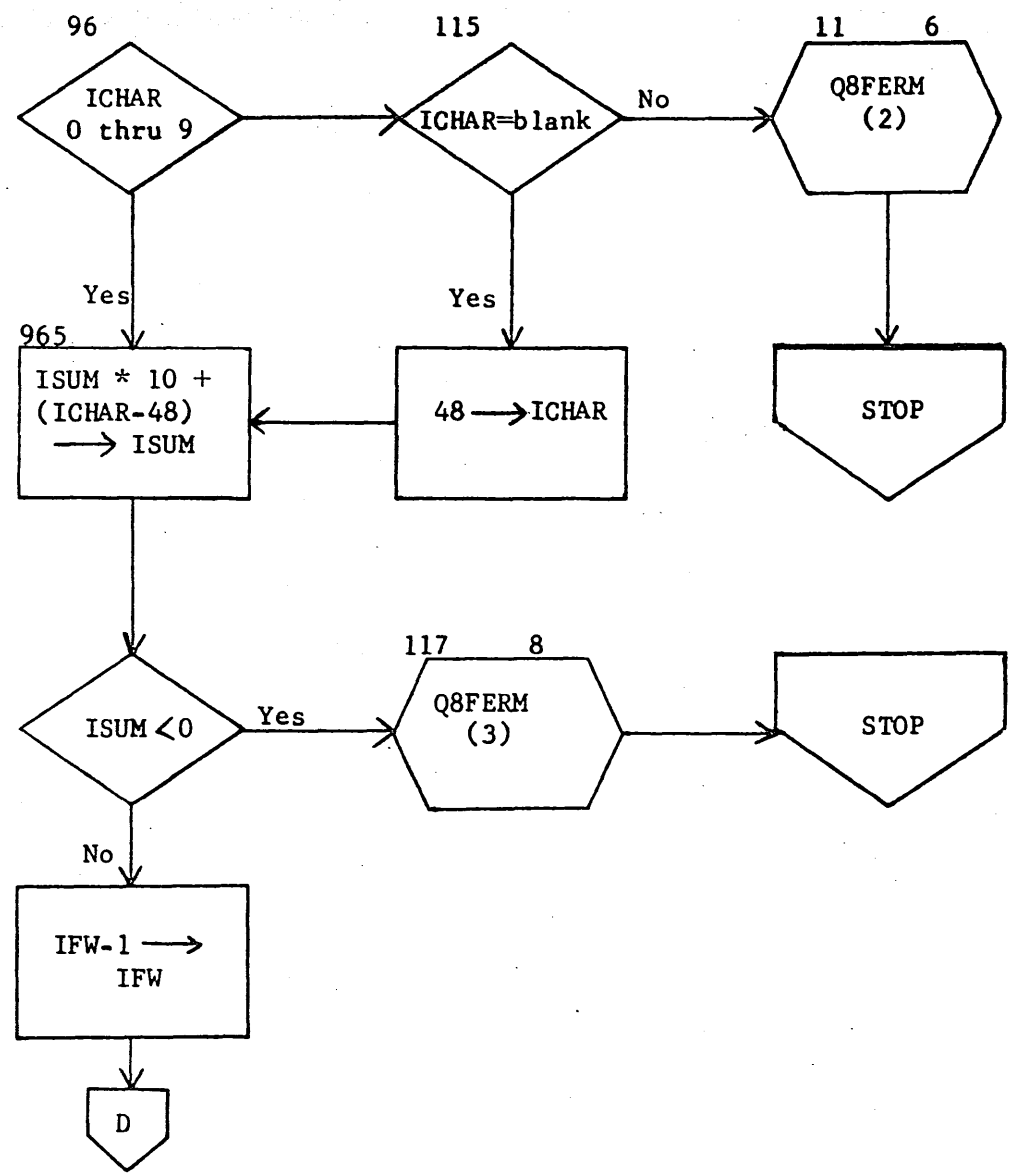


**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS <i>TRANS</i>	MACH. TYPE <i>TRANS</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE <b>Q8TRAN</b>	PAGE <b>44</b> OF <b>89</b>	PROJECT MGR.			
NUMBER	ISSUE DATE	TASK NO.			
DRAWN BY	DATE <b>5-1-7</b>	TASK NAME			

8. Error 3. Magnitude of number being processed exceeds the full specification.



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

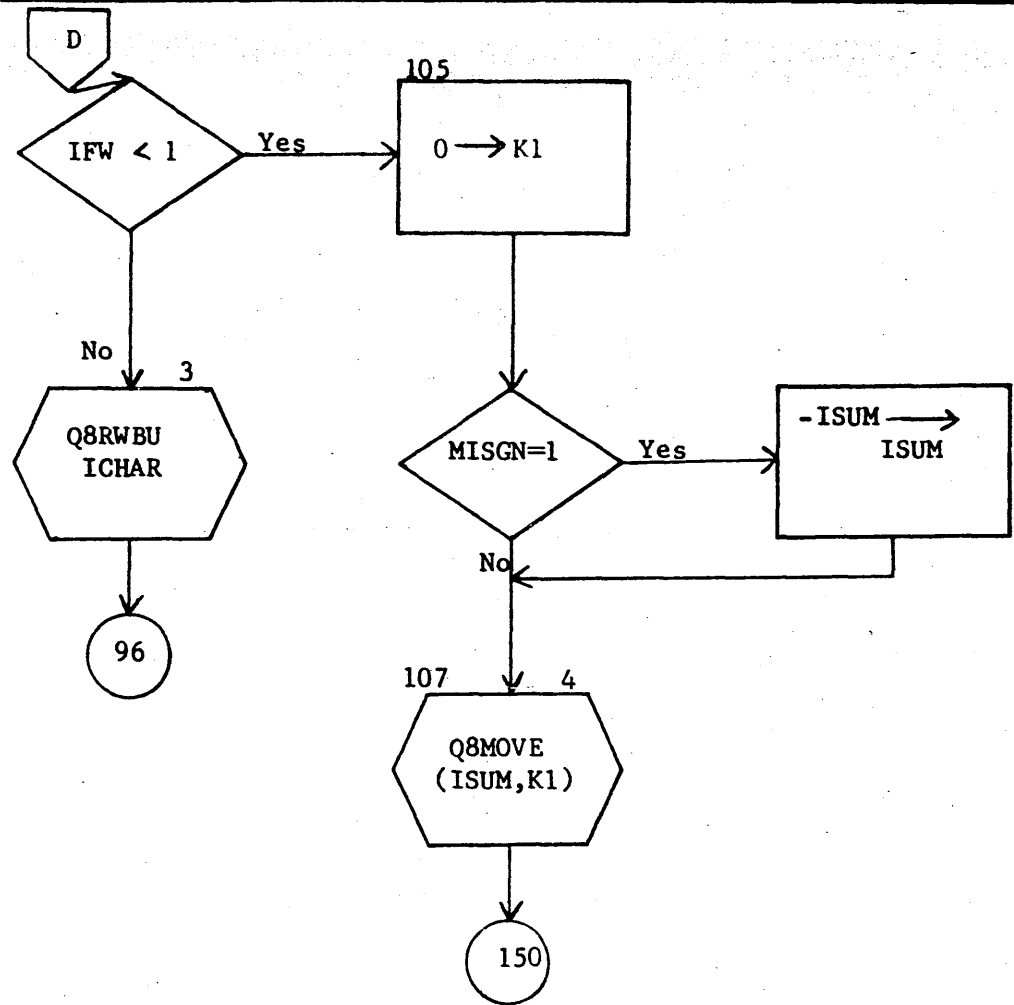
SAMPLE CODE

FLOWCHART

DECISION TABLE

OTHER

DOCUMENT CLASS	<i>JMS</i>	MACH. TYPE	<i>1700</i>	PROJECT NO.	REV.	APPROVED	DATE
DOCUMENT TITLE	Q8TRAN			PROJECT MGR.			
NUMBER	PAGE	OF		PROJECT NAME			
	<i>45</i>	<i>49</i>		TASK NO.			
DRAWN BY	ISSUE DATE	DATE	<i>5-6-67</i>	TASK NAME			

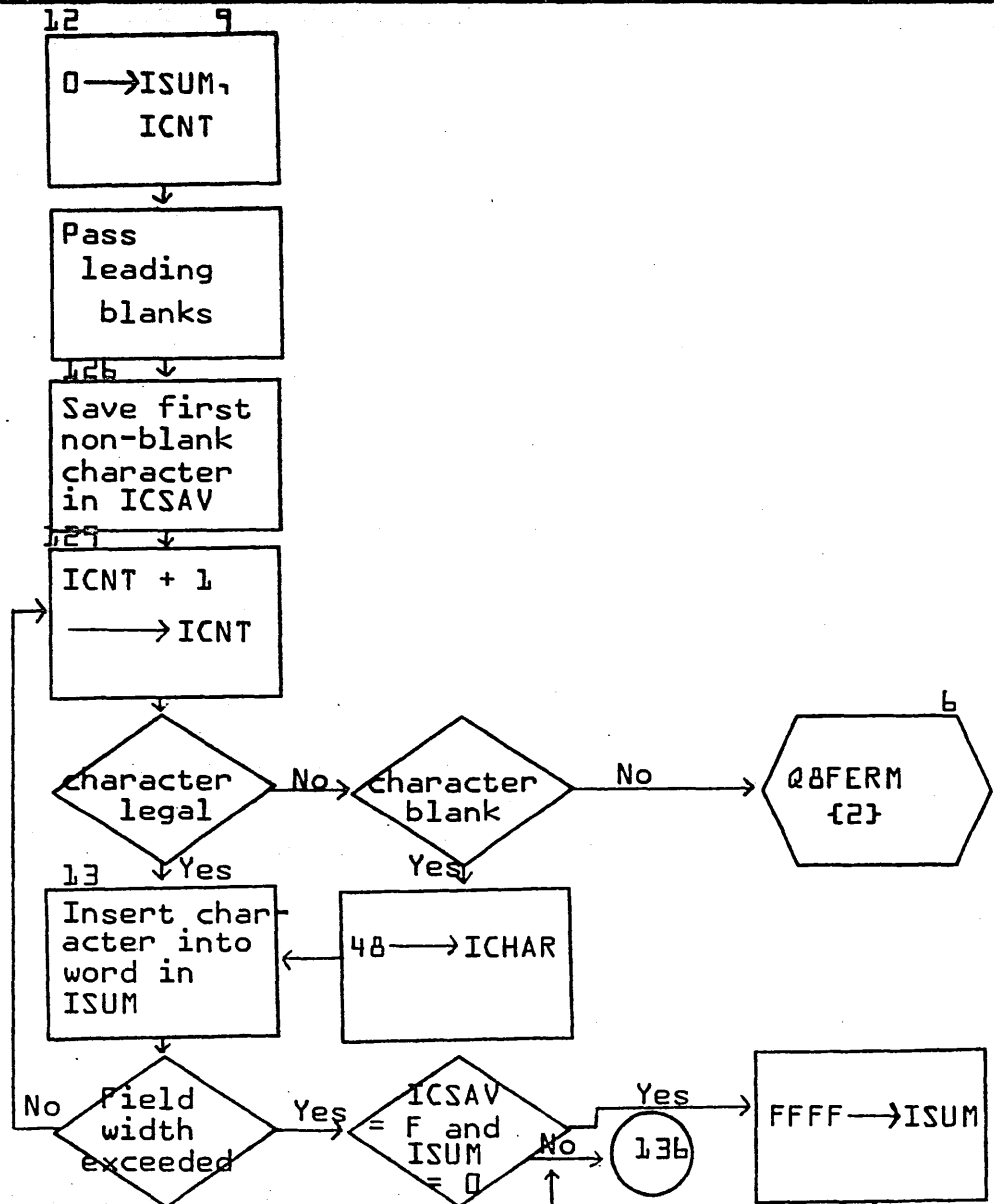


**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

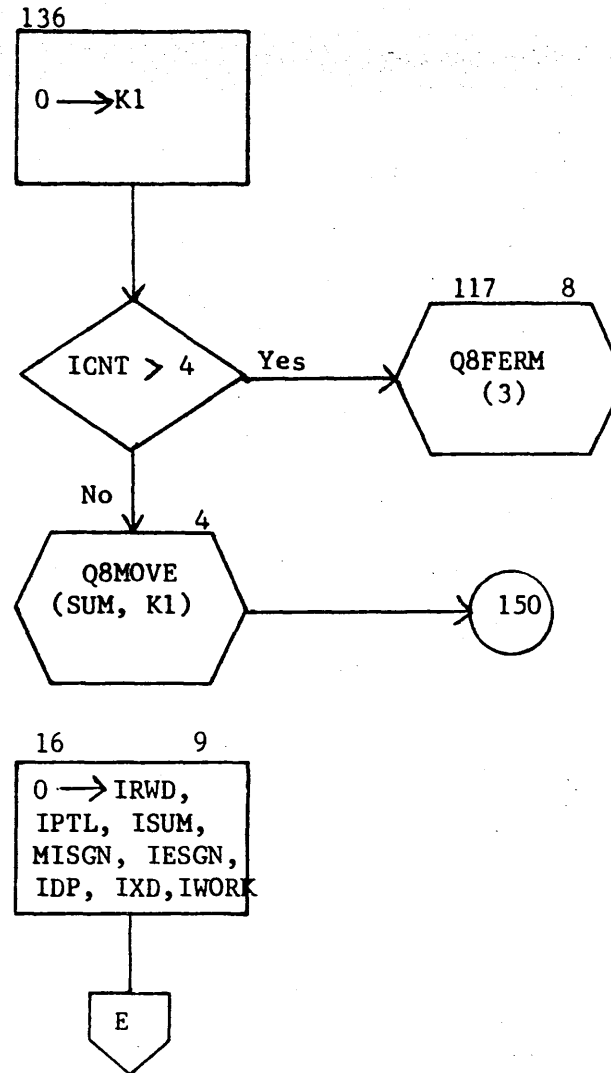
DOCUMENT CLASS <i>TMS</i>	MACH. TYPE <i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE <b>Q8TRAN</b>	PAGE <i>460F</i> OF <i>89</i>	PROJECT MGR.			
NUMBER	ISSUE DATE	TASK NO.			
DRAWN BY	DATE <i>3-67</i>	TASK NAME			

9. Hexadecimal input conversion comes here.



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	QBFTRAN			PROJECT MGR.			
	NUMBER	PAGE 47 OF 89			PROJECT NAME			
		ISSUE DATE				TASK NO.		
	DRAWN BY	DATE			TASK NAME			
	<div style="text-align: right;">6-79</div>							

9. E and F input conversions  
come here.



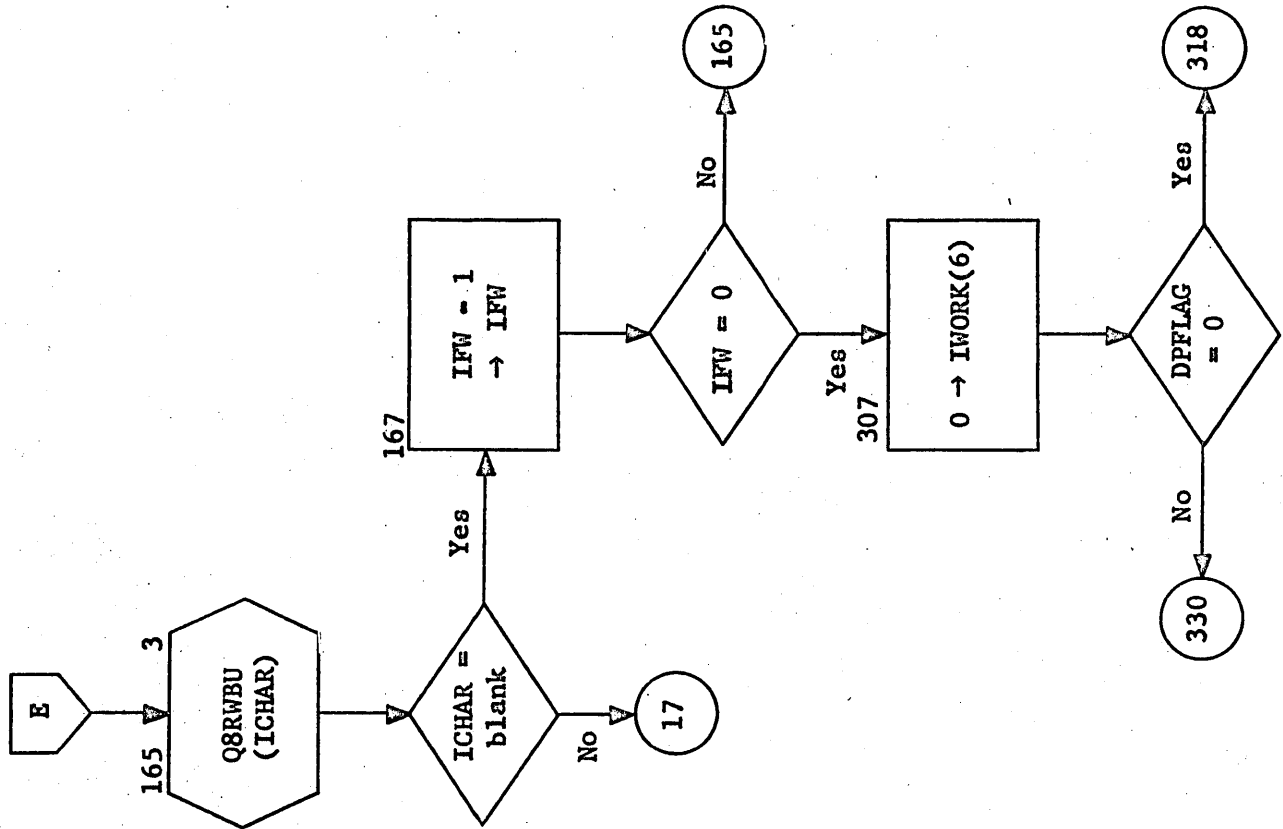
CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

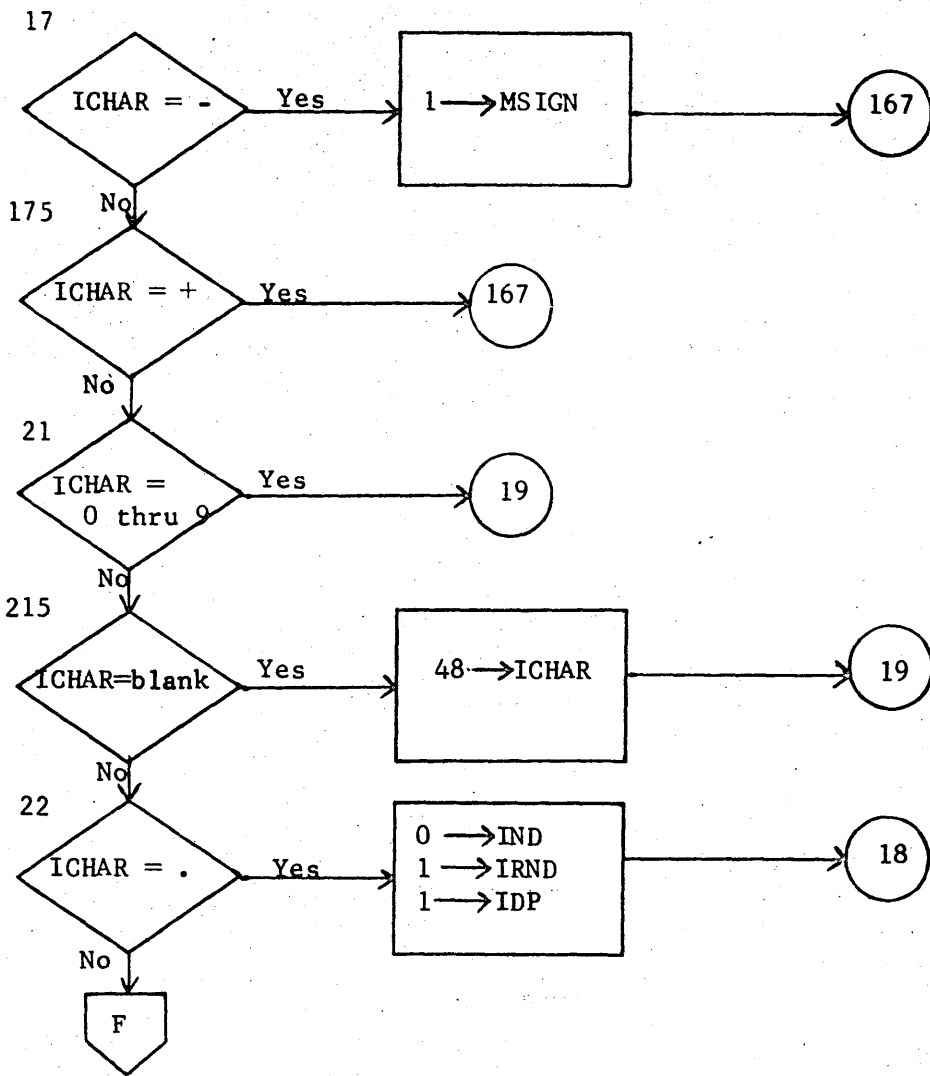
SAMPLE CODE   
FLOWCHART   
DECISION TABLE   
OTHER

DOCUMENT CLASS	MACH. TYPE	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE		PROJECT MGR.			
Q8TRAN	PAGE 48 OF 89	PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.			
DRAWN BY	DATE 3-27	TASK NAME			



DOCUMENT CLASS IMS PAGE NO. 6-81  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. CO05\*3.1 A/B MACHINE SERIES 1700



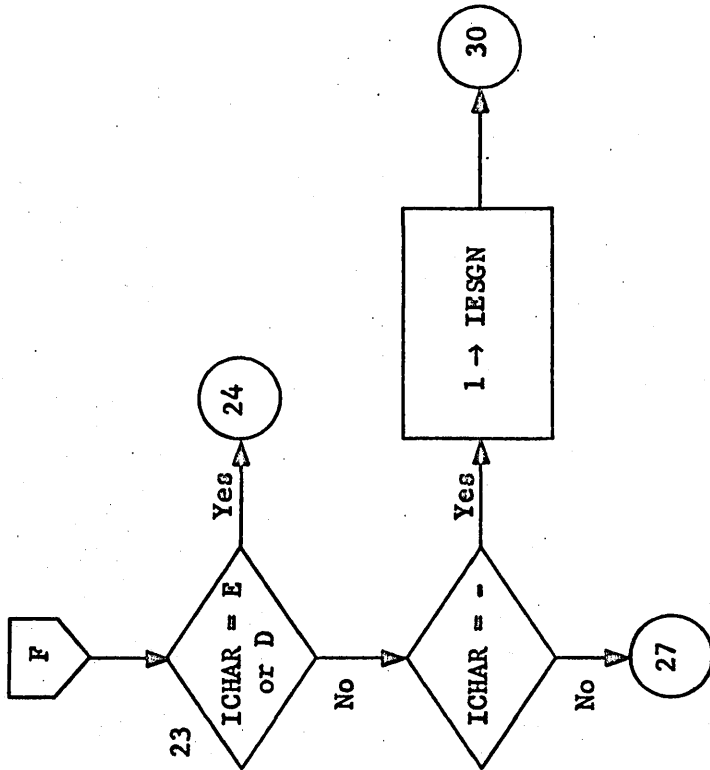


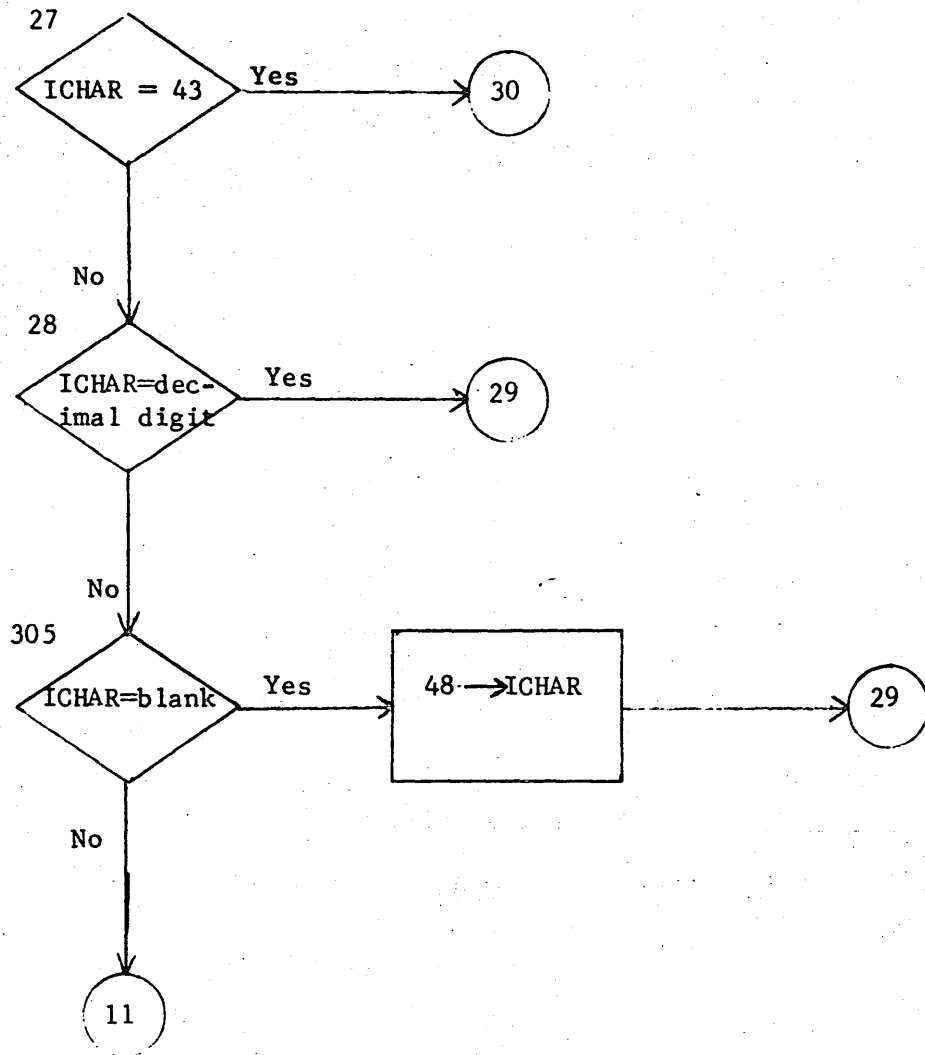
**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

DOCUMENT CLASS	MACH. TYPE	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE		PROJECT MGR.			
Q8TRAN	PAGE 50 OF 89	PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.			
DRAWN BY	DATE	TASK NAME			

DOCUMENT CLASS IMS PAGE NO. 6-83  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700





**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

DOCUMENT CLASS <i>JM</i>	MACH. TYPE <i>1100</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE		PROJECT MGR.			
<i>Q8TRAN</i>	<i>PAGE 52 OF 89</i>	PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.			
DRAWN BY	DATE <i>3/1/64</i>	TASK NAME			

29  
 IPTL 8 10 +  
 ICHAR - 48 →  
 IPTL

IPTL > 99

Yes → 117

No

30  
 IFW-1 → IFW

IFW = 0

Yes → 31

No

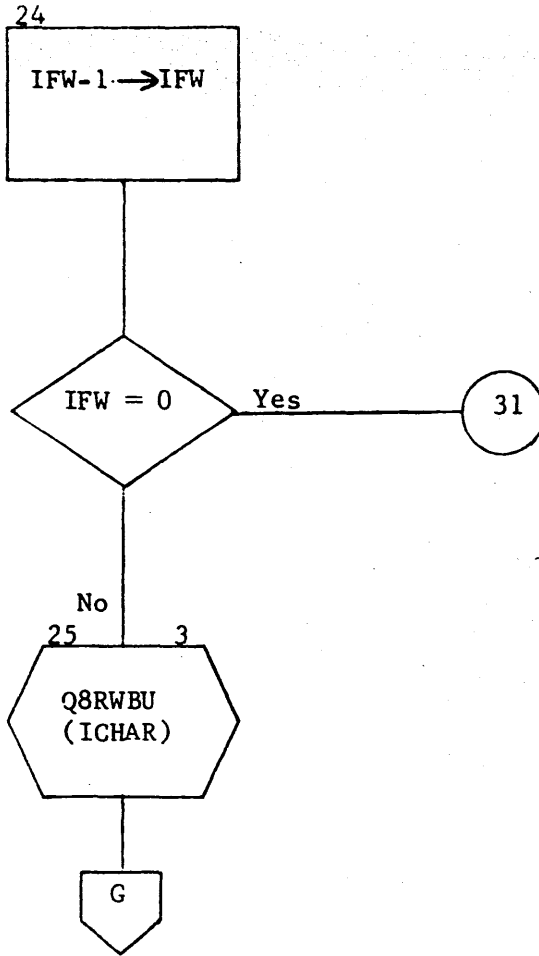
Q8RWB  
 (ICHAR)

→ 28

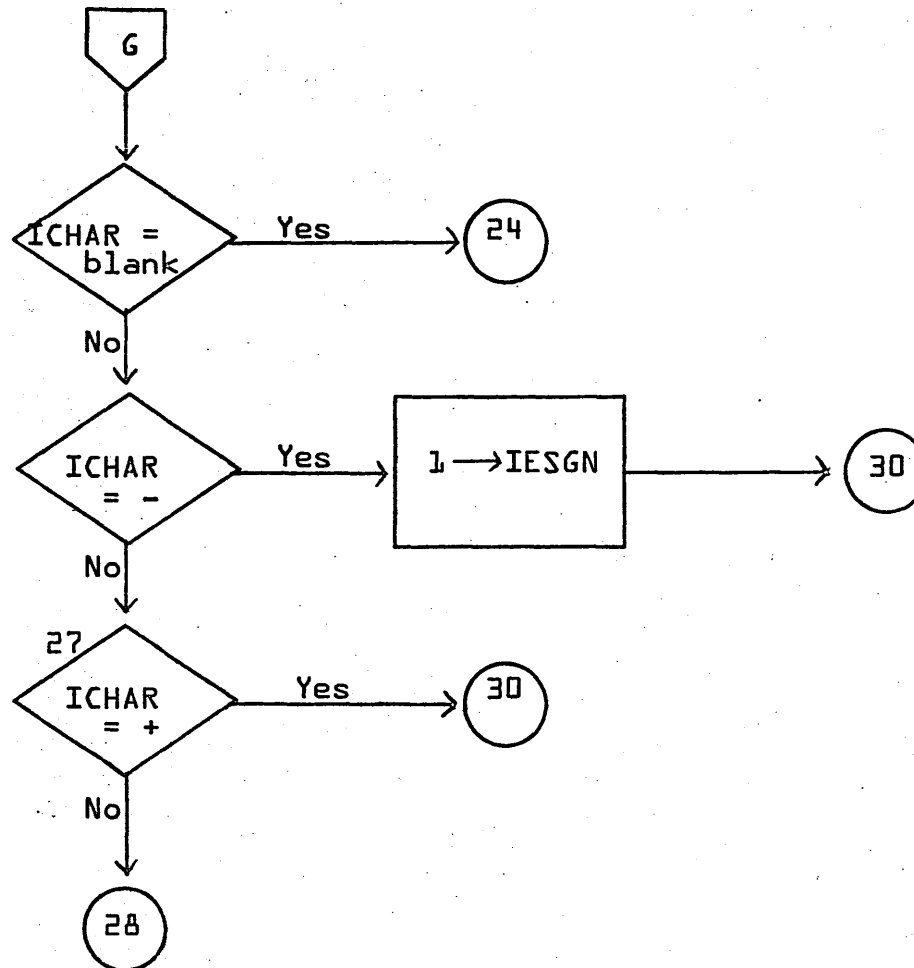
CONTROL DATA CORPORATION  
 SOFTWARE DOCUMENT

DOCUMENT CLASS	<i>IM</i>	MACH. TYPE	<i>1100</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	Q8TRAN			PROJECT MGR.			
NUMBER		ISSUE DATE	PAGE 53 of 89	PROJECT NAME			
DRAWN BY		DATE	<i>5-6-77</i>	TASK NO.			
				TASK NAME			

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS <i>LM</i>	MACH. TYPE <i>100</i>	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE		PROJECT MGR.			
	Q8TRAN	PAGE 54 of 89	PROJECT NAME			
	NUMBER	ISSUE DATE	TASK NO.			
	DRAWN BY	DATE <i>3-6-1</i>	TASK NAME			



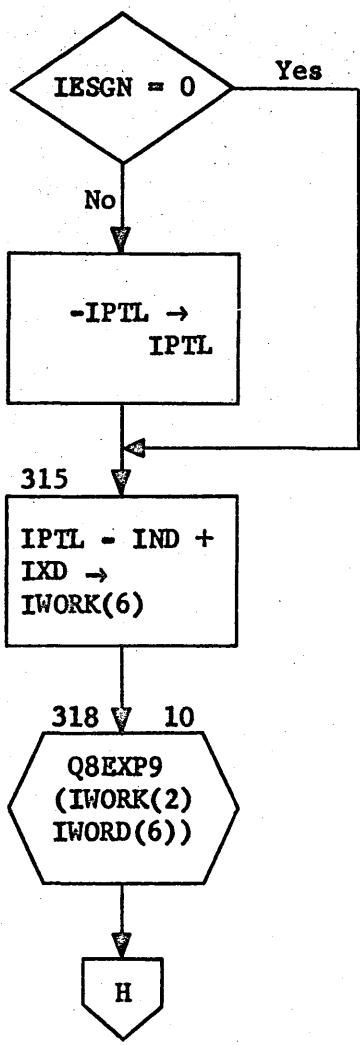
**CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT**

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

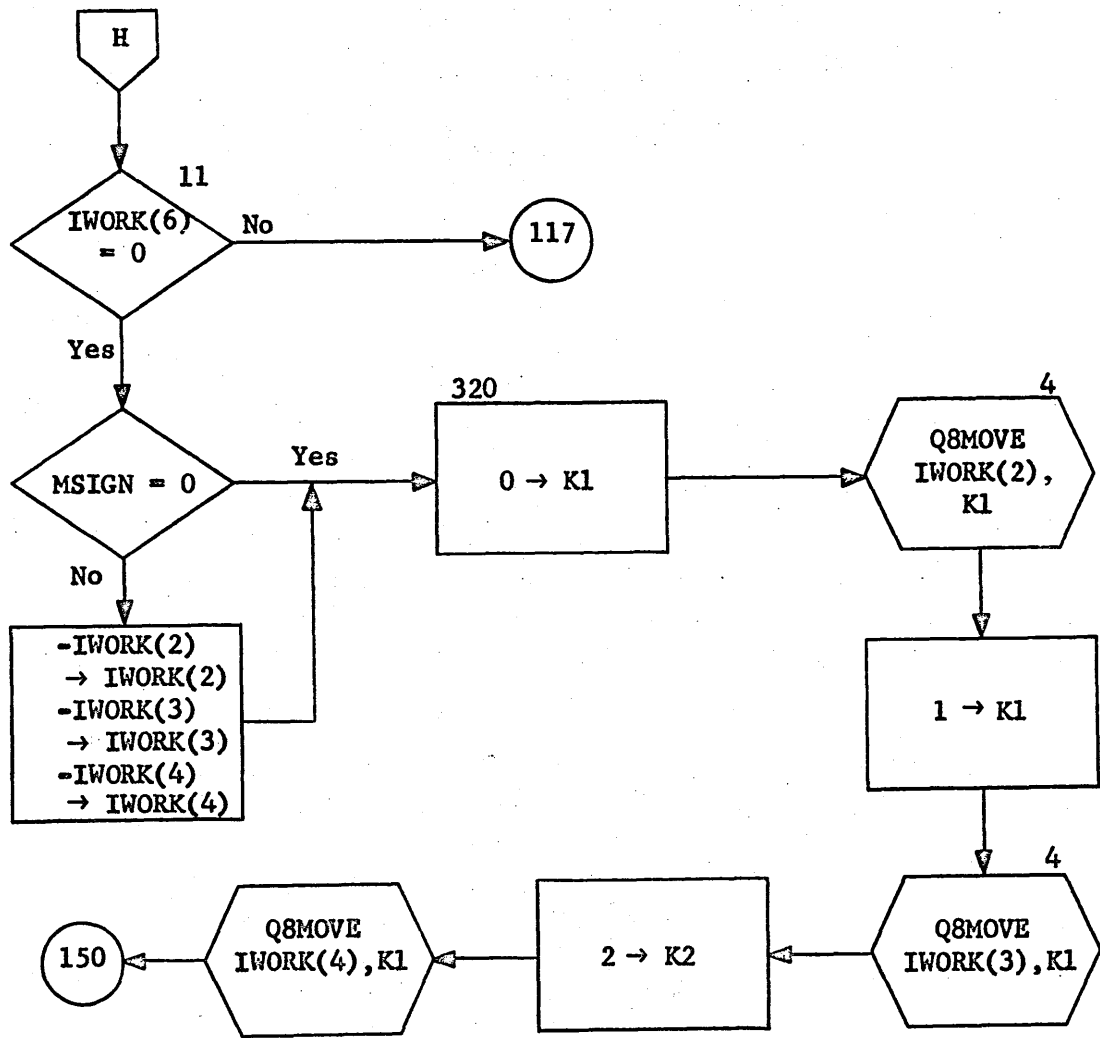
DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	Q8TRAN			PROJECT MGR.			
NUMBER		ISSUE DATE	PAGE 55 OF 89	PROJECT NAME			
DRAWN BY		DATE		TASK NO.			
				TASK NAME			

5-87

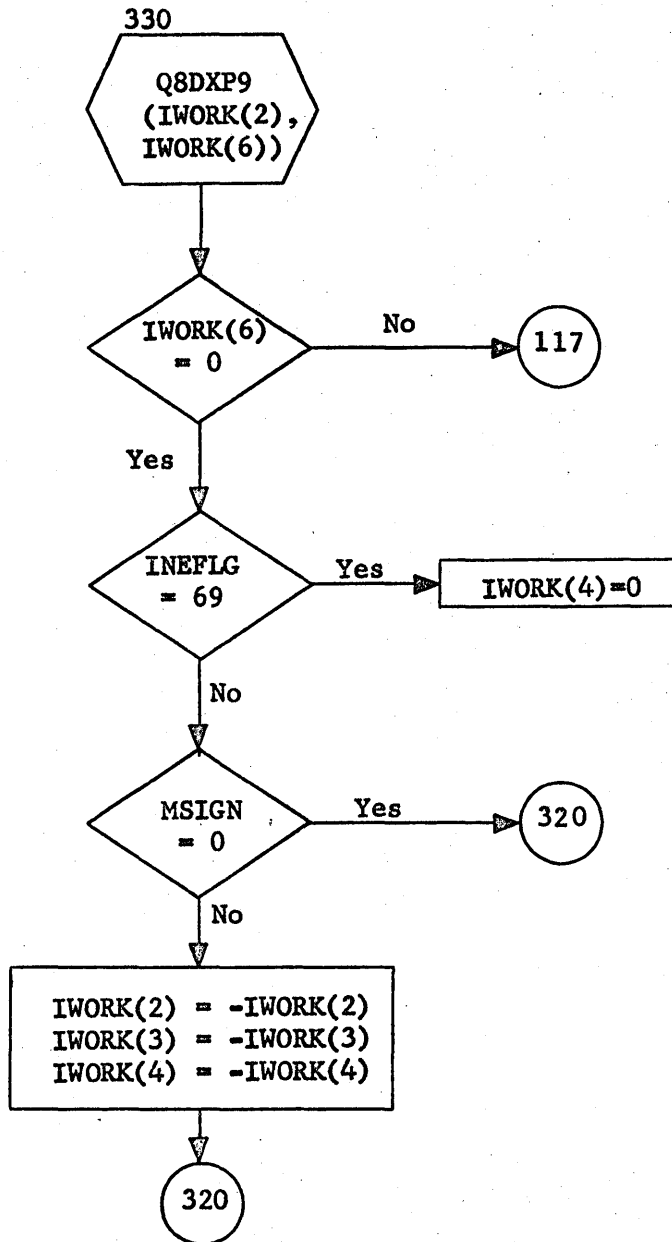
10. This routine picks up a decimal integer at IWORD(2), IWORK(3), IWORK(4), and IWORK(5) and the corresponding decimal exponent at IWORK(6) and converts these values into the binary equivalent in the real number form required by the floating point package.
11. If magnitude of number to be converted is excessive, IWORK(6) is set to non-zero.



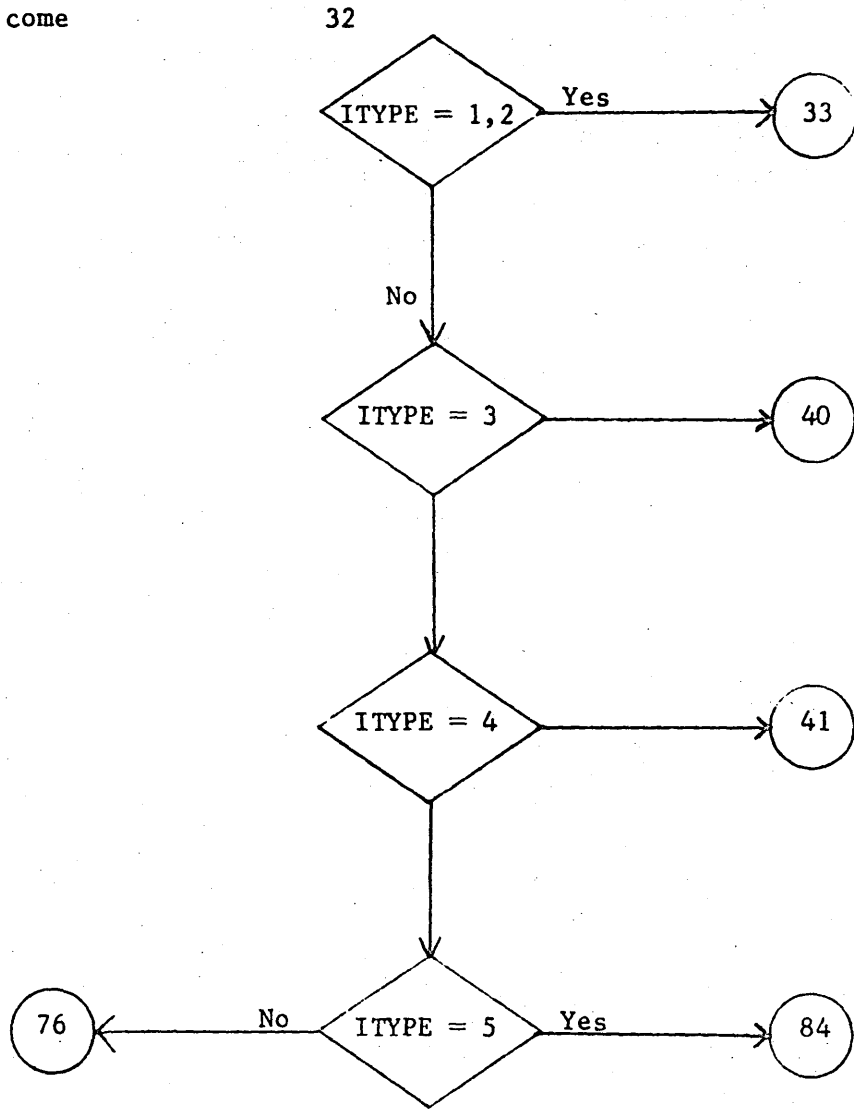




DOCUMENT CLASS IMS PAGE NO. 6-89A  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700



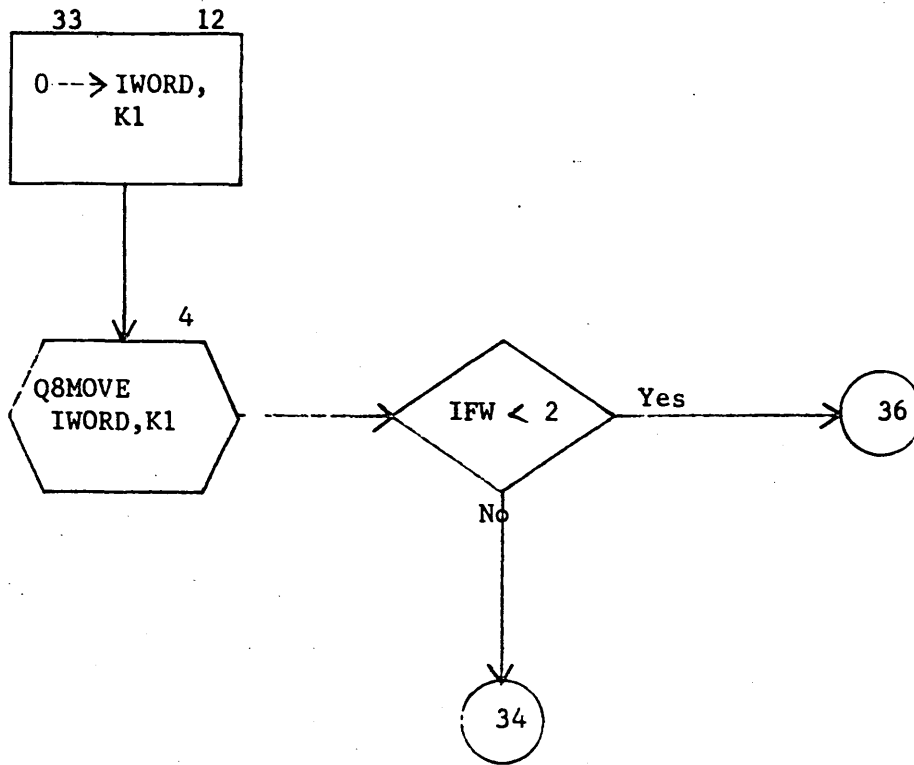
A and R output conversions come here.



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

DOCUMENT CLASS	<i>T.M.C.</i>	MACH. TYPE	<i>1100</i>	PROJECT NO.	REV.	APPROVED	DATE
DOCUMENT TITLE				PROJECT MGR.			
NUMBER	<i>Q8TRAN</i>	ISSUE DATE	<i>PAGE 58 OF 89</i>	PROJECT NAME			
DRAWN BY		DATE	<i>3-17</i>	TASK NO.			
				TASK NAME			

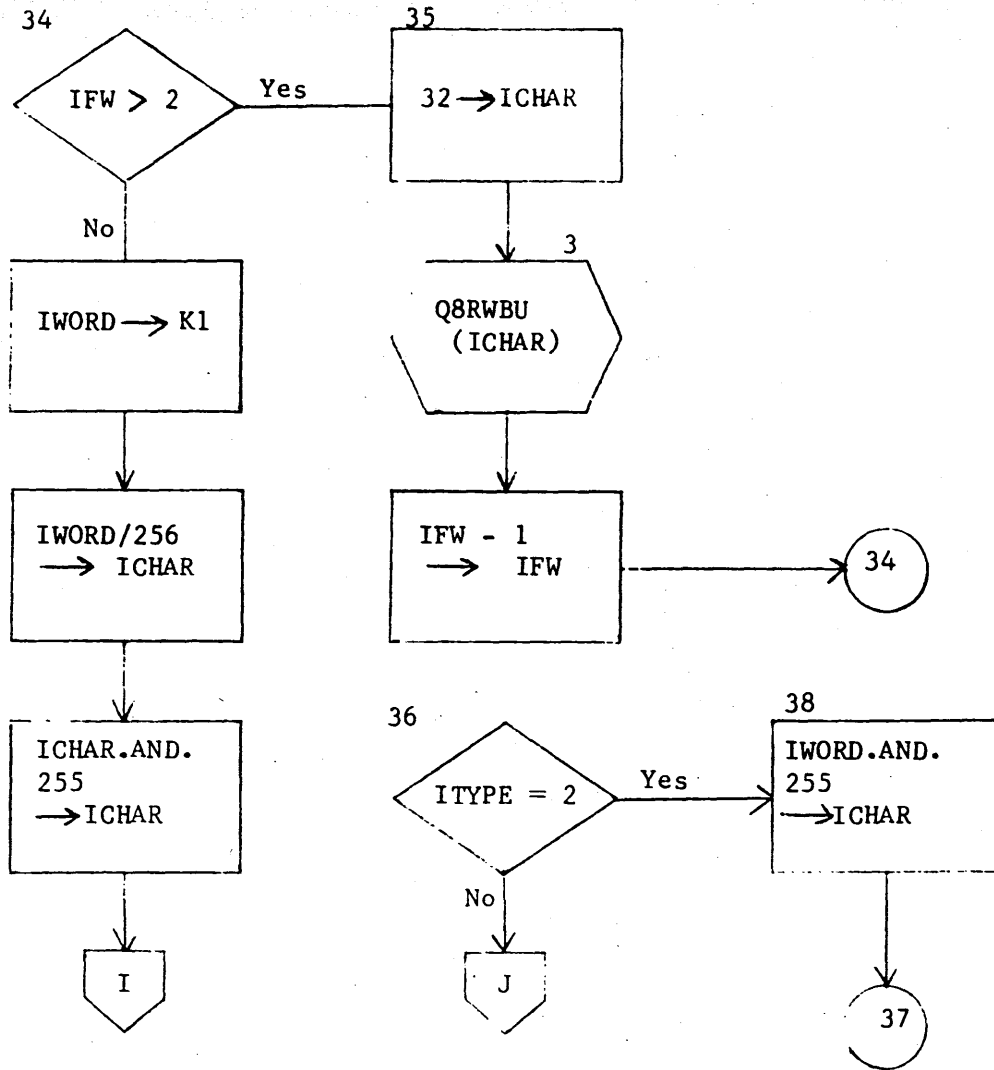




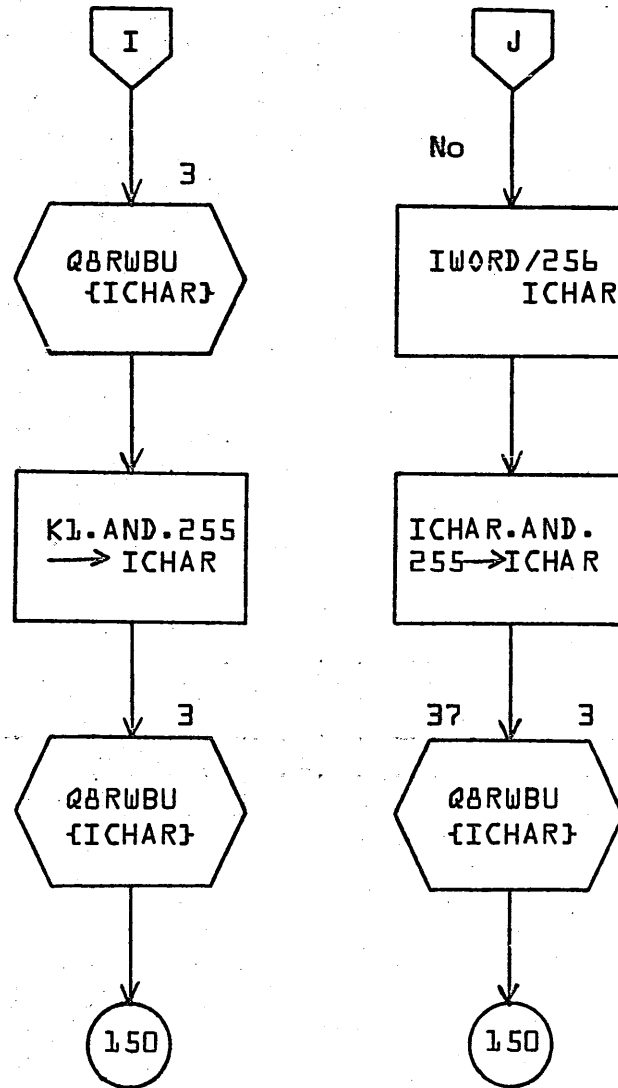
**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	MACH. TYPE	PROJECT NO.	REV	APPROVED	DATE
1M	1100				
DOCUMENT TITLE		PROJECT MGR.			
Q8TRAN	PAGE 59 OF 89	PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.			
DRAWN BY	DATE 3-6-71	TASK NAME			



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	MACH. TYPE	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE		PROJECT MGR.			
	NUMBER	ISSUE DATE	PROJECT NAME			
	DRAWN BY	DATE	TASK NO.			
			TASK NAME			

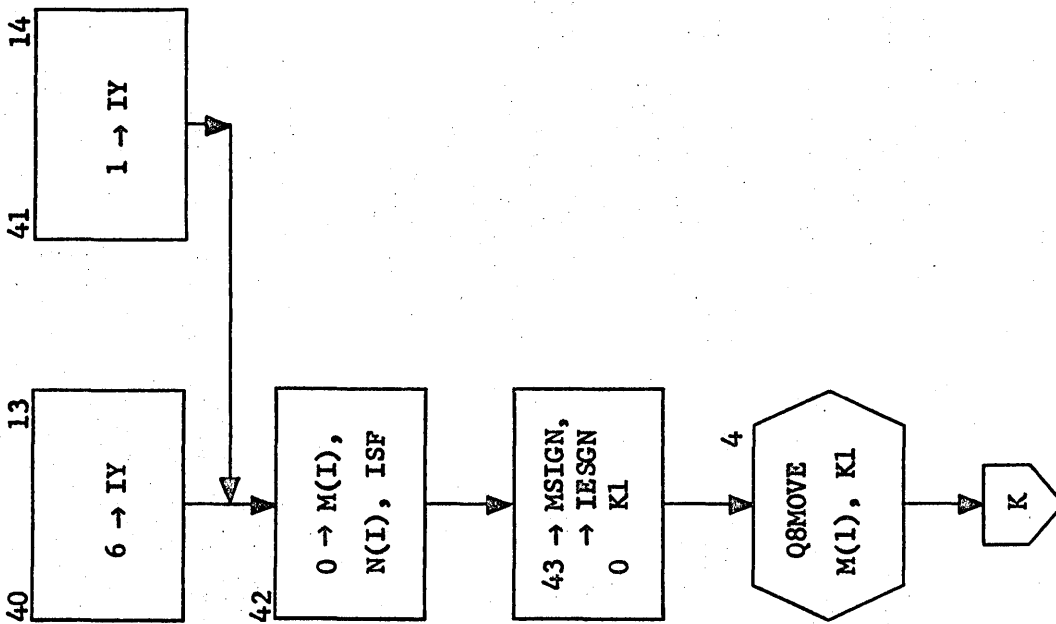


**CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT**

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	Q8TRAN			PROJECT MGR.			
NUMBER		ISSUE DATE	PAGE 1 OF 89	PROJECT NAME			
DRAWN BY		DATE		TASK NO.			
				TASK NAME			

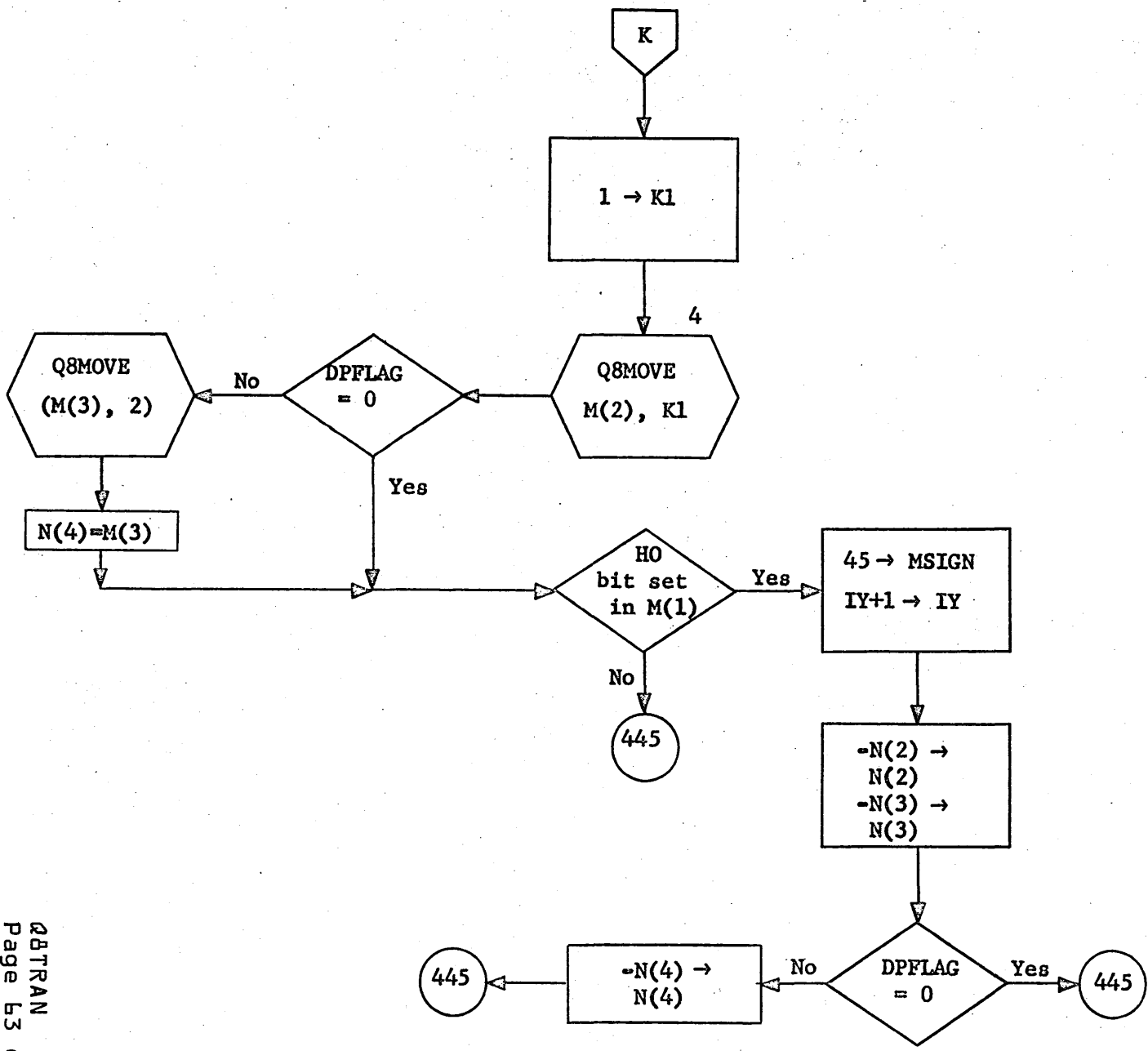
DOCUMENT CLASS IMS PAGE NO. 6-94  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700



13. E and D conversions come here.

14. F conversion comes here.

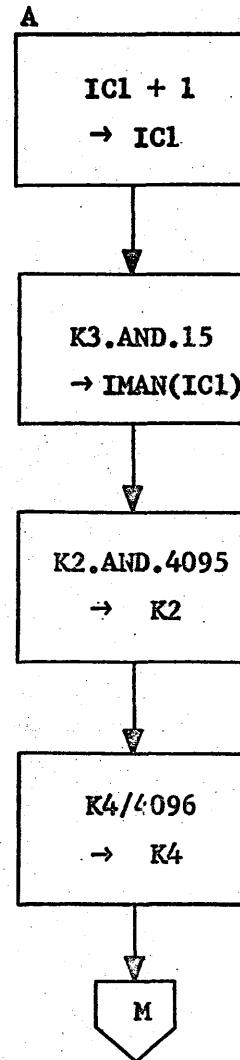
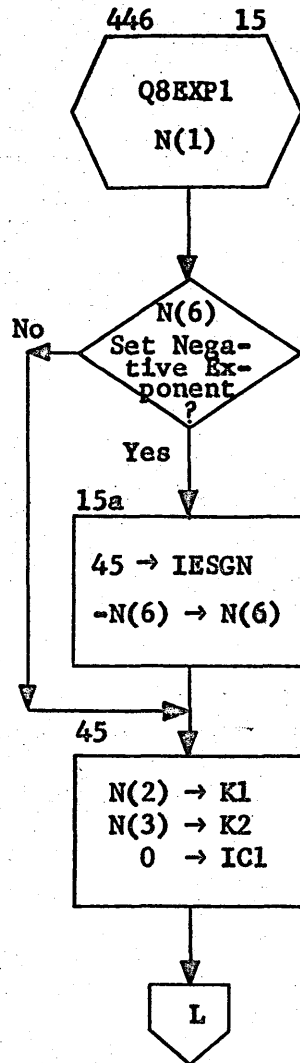
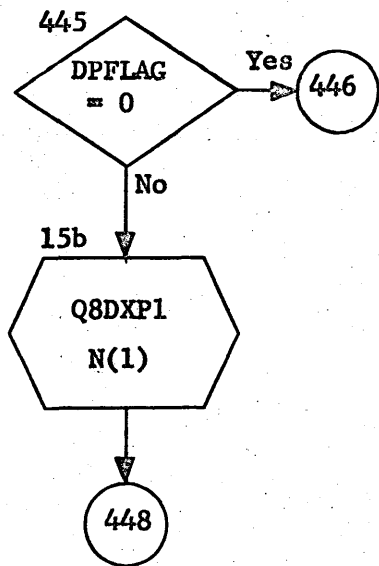




15. This routine takes floating point number in the form required by the floating point package and converts it to an intermediate form of a decimal integer representing the mantissa in N(2), N(3) and a decimal exponent in N(6).

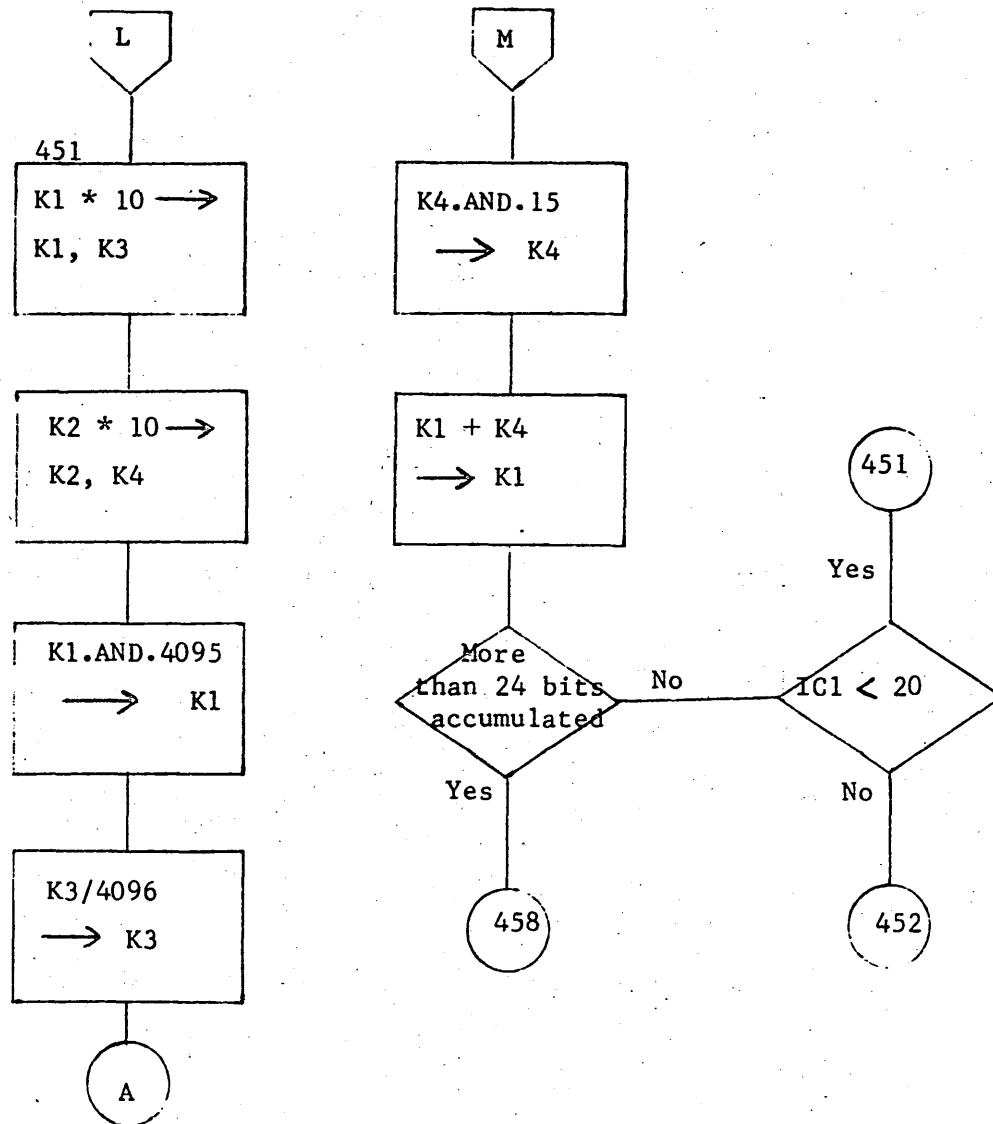
15a. IESGN = 45 means negative exponent  
IESGN = 43 means positive exponent

15b. This routine takes a double precision floating point number in the form required by the floating point package and converts it to an intermediate form of a decimal integer representing the mantissa in N(2), N(3), N(4), and N(5) and the decimal exponent in N(6).



DOCUMENT CLASS IMS PAGE NO. 6-9b  
 PRODUCT NAME J700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. CO05\*3.1 A/B MACHINE SERIES J700

CONTROL DATA CORPORATION  
 LA JOLLA RESOURCE CENTER  
 DIVISION



**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

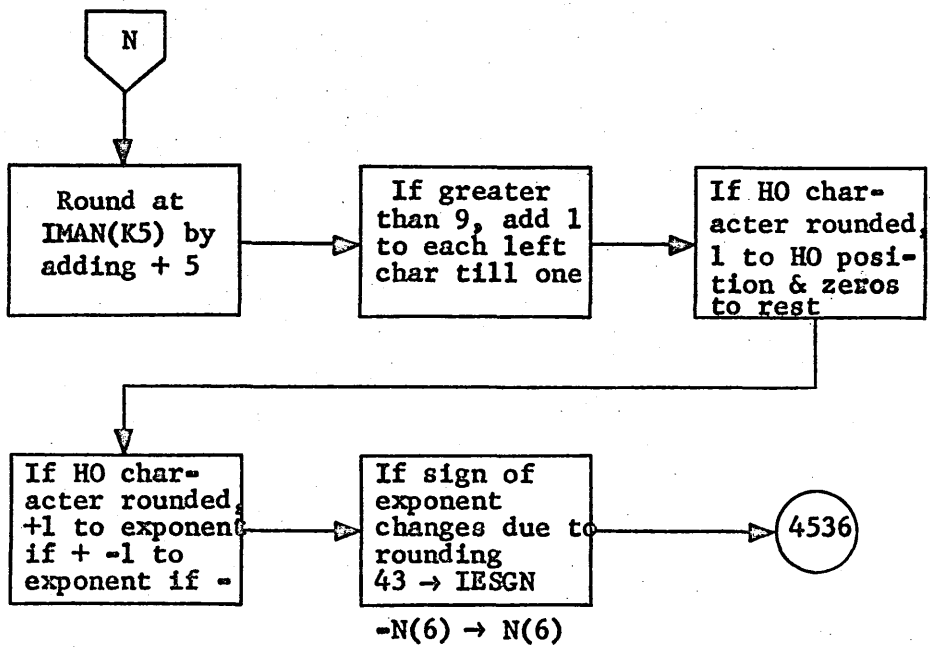
- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

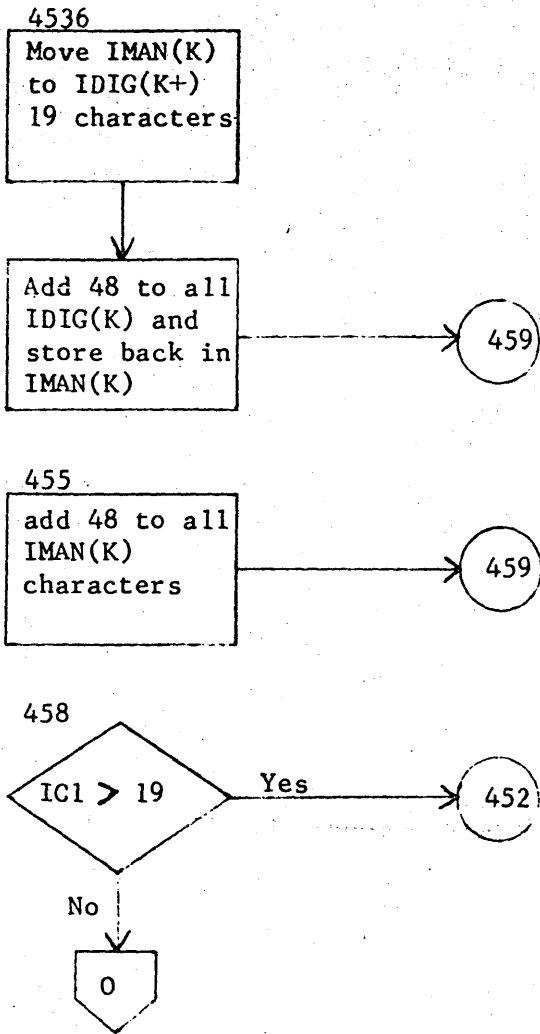
DOCUMENT CLASS	<i>TH</i>	MACH. TYPE	<i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	Q8TRAN			PROJECT MGR.			
NUMBER		ISSUE DATE	PAGE 65 OF 89	PROJECT NAME			
DRAWN BY		DATE	<i>7-65</i>	TASK NO.			
				TASK NAME			



CONTROL DATA CORPORATION  
LA JOLLA RESOURCE CENTER DIVISION

DOCUMENT CLASS IMS PAGE NO. E-99  
PRODUCT NAME J700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. 5005\*3.1 A/B MACHINE SERIES J700





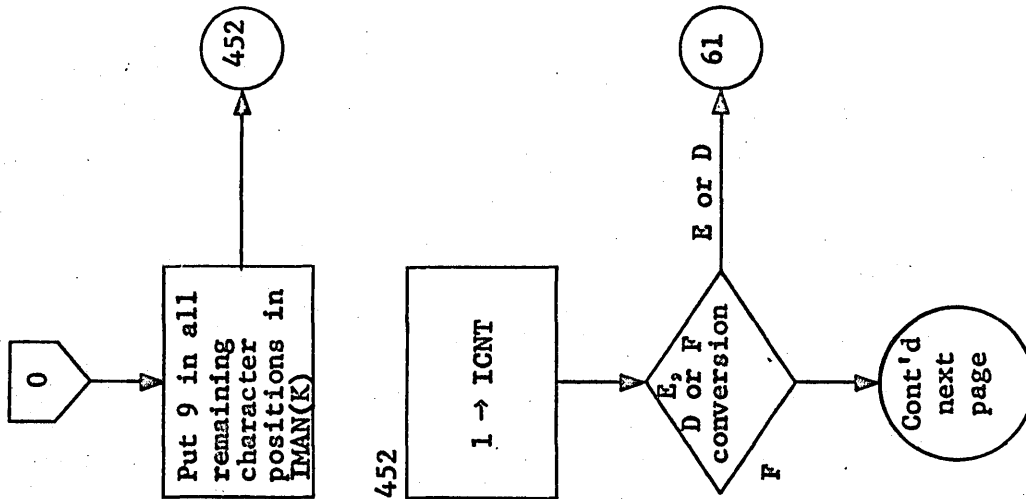
**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

DOCUMENT CLASS	MACH. TYPE	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE		PROJECT MGR.			
Q8TRAN	PAGE 68 OF 89	PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.			
DRAWN BY	DATE	TASK NAME			

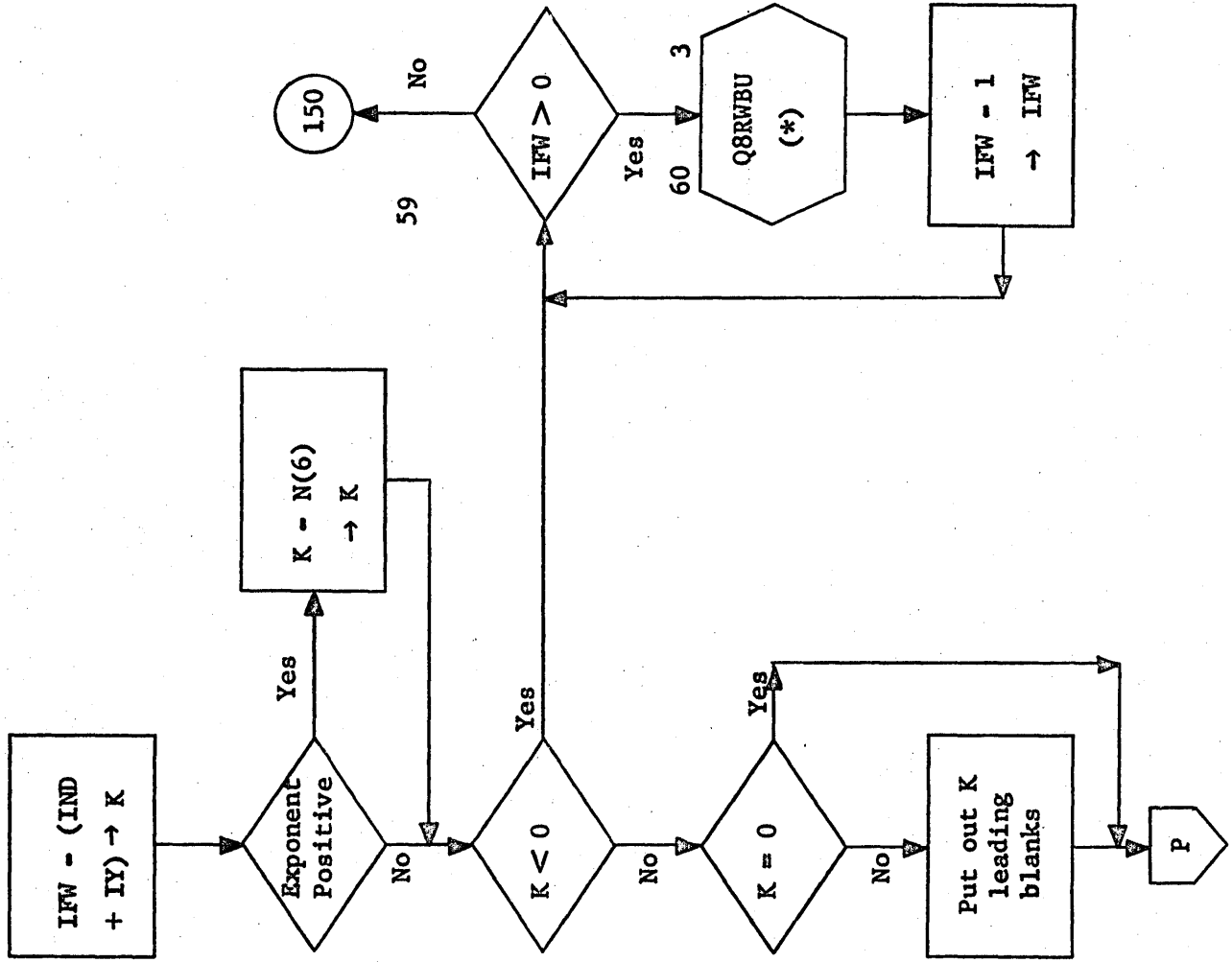
- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

6-100

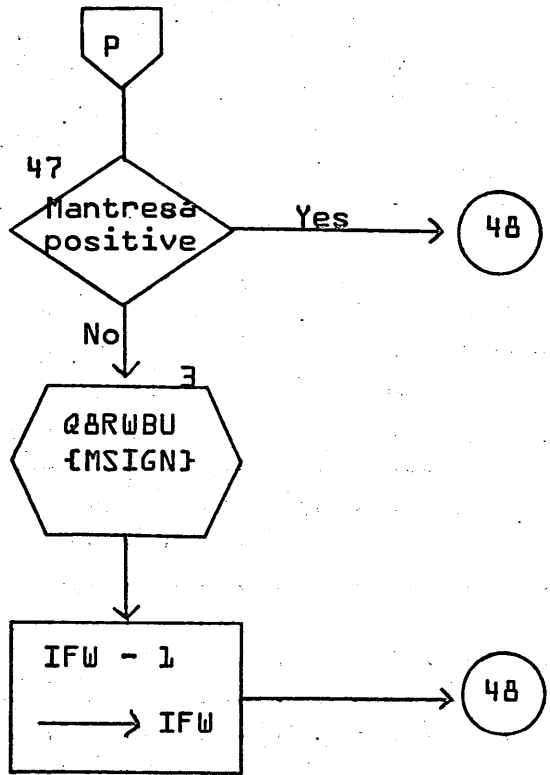
DOCUMENT CLASS IMS PAGE NO. 6-101  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700



DOCUMENT CLASS IMS PAGE NO. 6-102  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700







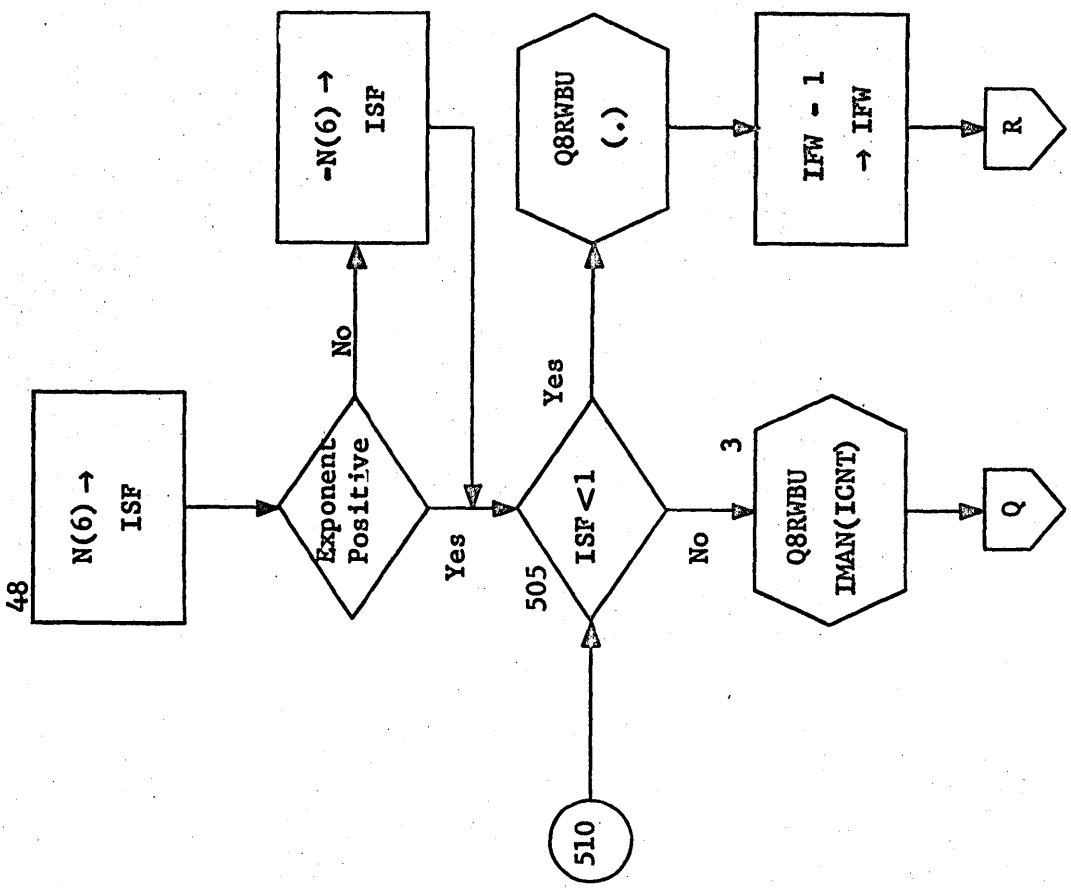
**CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT**

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	Q&RWBUB {MSIGN}			PROJECT MGR.			
NUMBER	Q&RWBUB	ISSUE DATE	PAGE 71 OF 89	PROJECT NAME			
DRAWN BY		DATE		TASK NO.			
				TASK NAME			

E0125

DOCUMENT CLASS IMS PAGE NO. 6-104  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

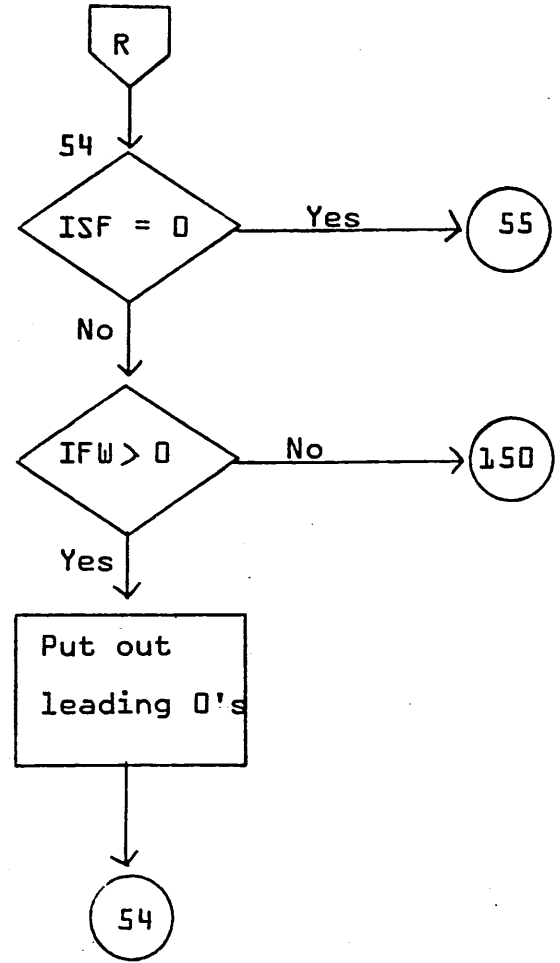
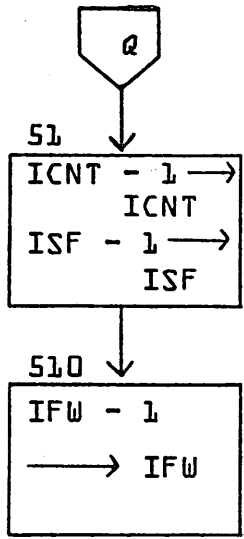


A

B

C

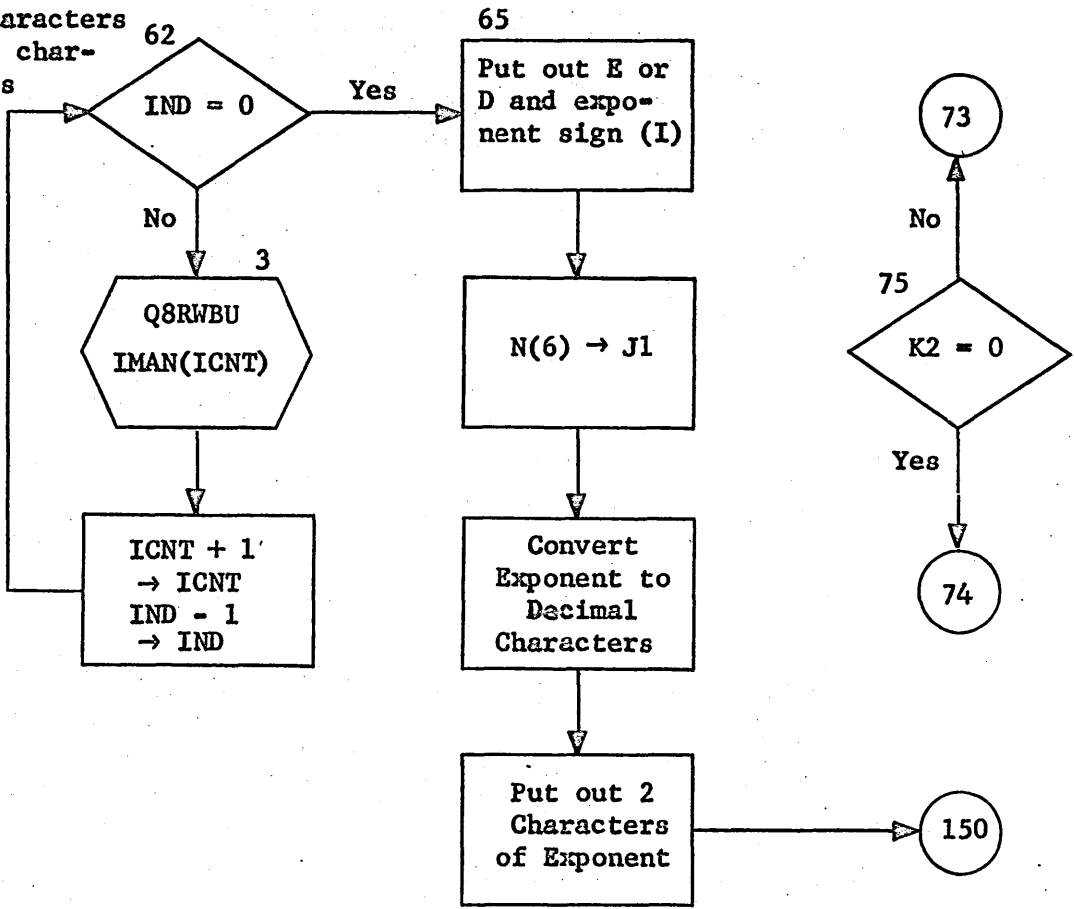
D



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	Q8TRAN			PROJECT MGR.			
	NUMBER	PAGE 73 OF 89			PROJECT NAME			
		ISSUE DATE			TASK NO.			
	DRAWN BY	DATE			TASK NAME			



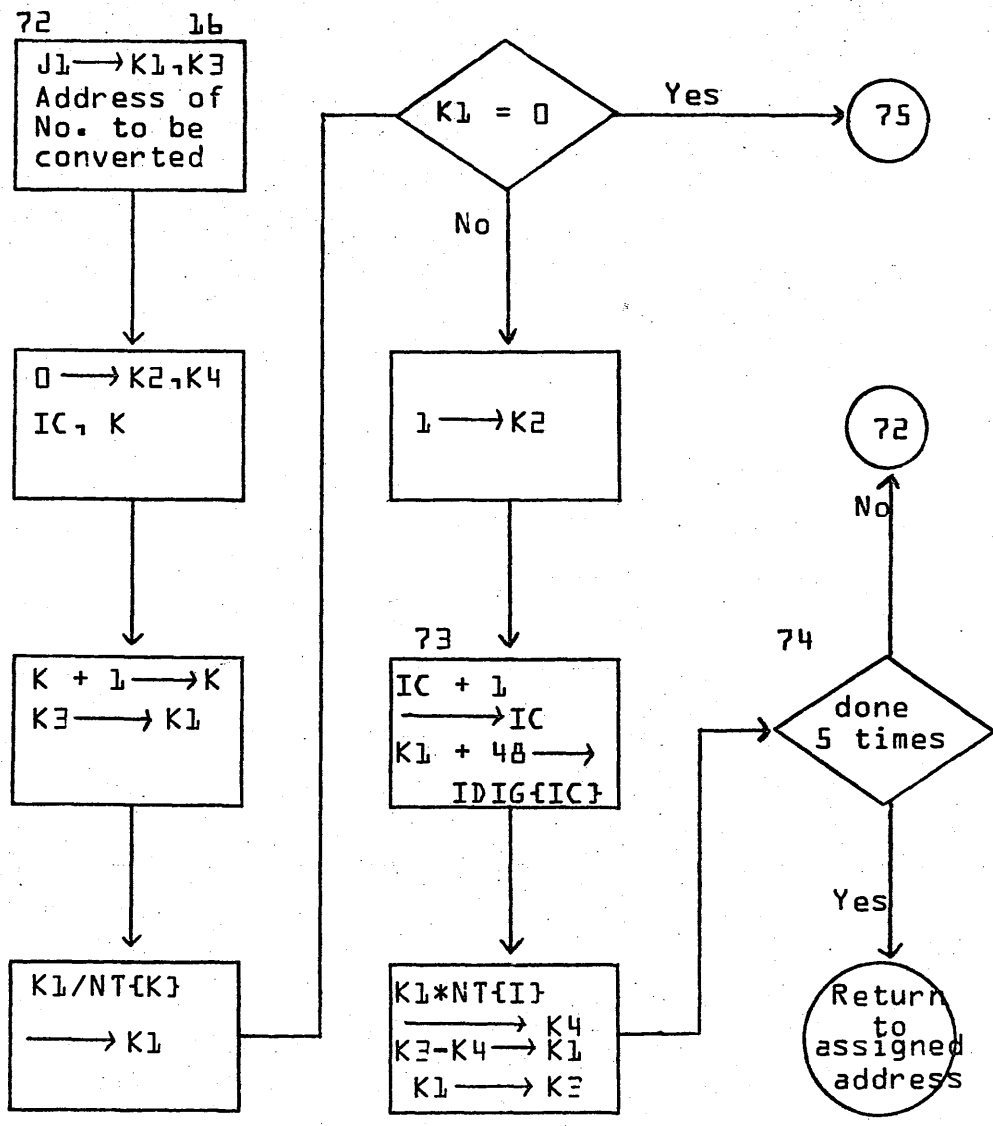
16. This section of code converts a binary integer placed in J1 into the equivalent characters in IDIG(I). IC is the character count. Return is made to IAD.



DOCUMENT CLASS IMS PAGE NO. B-107  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. CDD5\*3.1 A/B MACHINE SERIES 1700

CONTROL DATA CORPORATION  
 LA JOLLA RESOURCE CENTER DIVISION

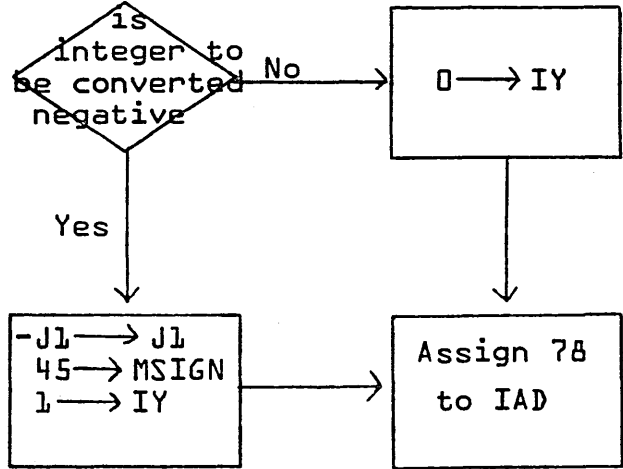
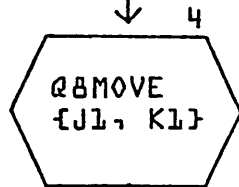
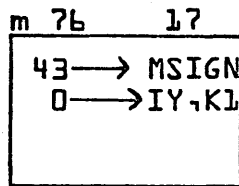
A  
B  
C  
D



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	Q8TRAN			PROJECT MGR.			
	NUMBER	PAGE 76 OF 89			PROJECT NAME			
		ISSUE DATE	TASK NO.					
	DRAWN BY	DATE			TASK NAME			

F-108

17. Integer output conversion comes here.



72

<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	Q8TRAN			PROJECT MGR.			
	NUMBER		ISSUE DATE	PAGE 77 OF 89	PROJECT NAME			
	DRAWN BY		DATE		TASK NO.			
					TASK NAME			

78  
 If value zero,  
 make IC=1 &  
 48 IDIG(1)

Put out field  
 of \*'s if  
 Field width  
 too small

Put out  
 leading  
 blanks

Put out -  
 if integer  
 value nega-  
 tive

Put out dec-  
 imal digits  
 required

150

6-110

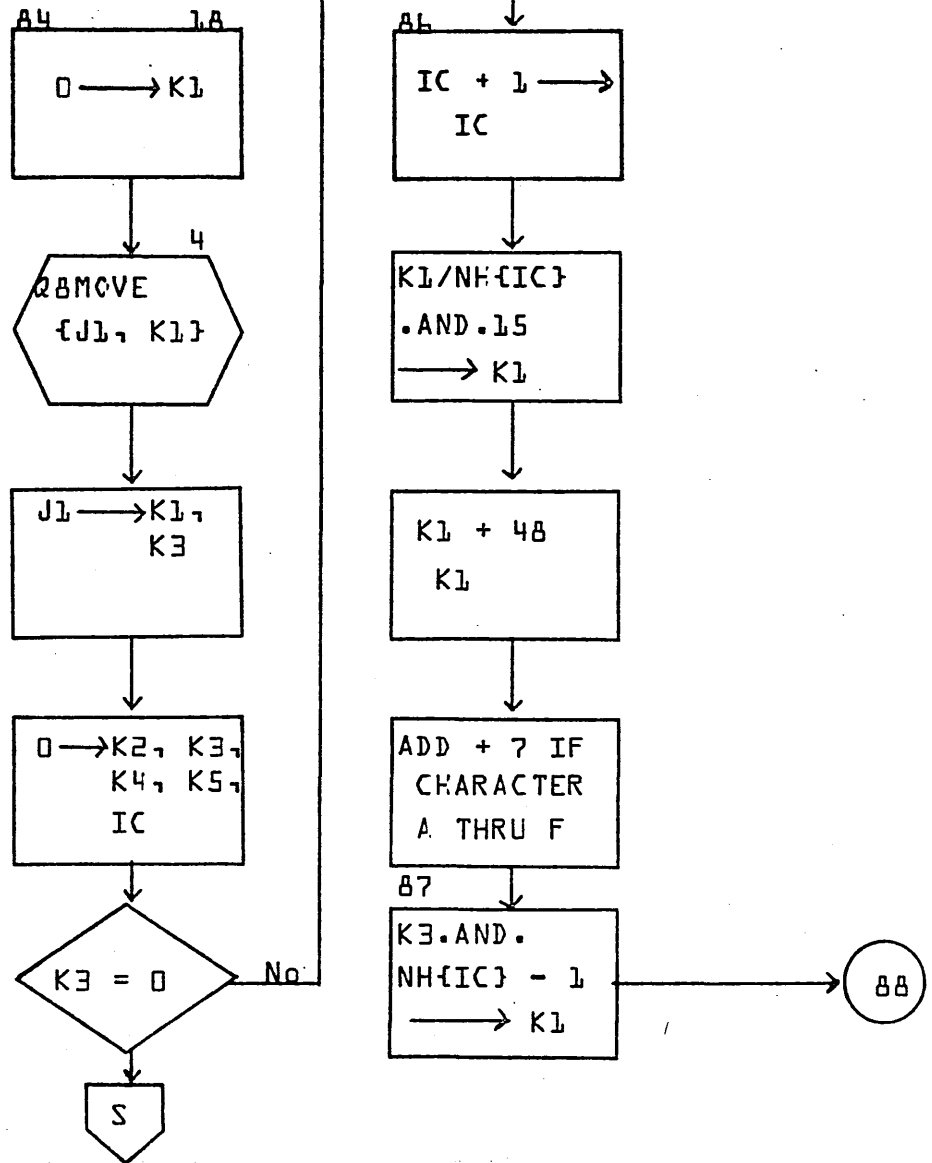
CONTROL DATA CORPORATION  
 SOFTWARE DOCUMENT

SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

DOCUMENT CLASS	MACH. TYPE	PROJECT NO.	REV	APPROVED	DATE
141					
DOCUMENT TITLE		PROJECT MGR.			
Q8TRAN	PAGE 78 OF 89	PROJECT NAME			
NUMBER	ISSUE DATE	TASK NO.			
DRAWN BY	DATE	TASK NAME			



18. Hexadecimal output conversion comes here.



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

- SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	QBTRAN			PROJECT MGR.			
NUMBER		ISSUE DATE	PAGE 79 OF 85	PROJECT NAME			
DRAWN BY		DATE		TASK NO.			
				TASK NAME			

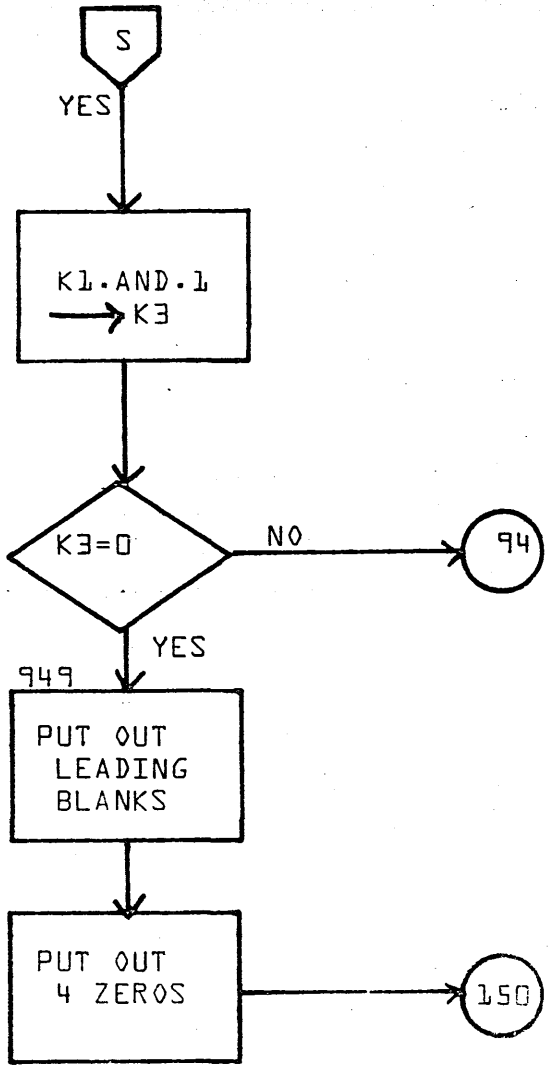
T-11-5

A

B

C

D

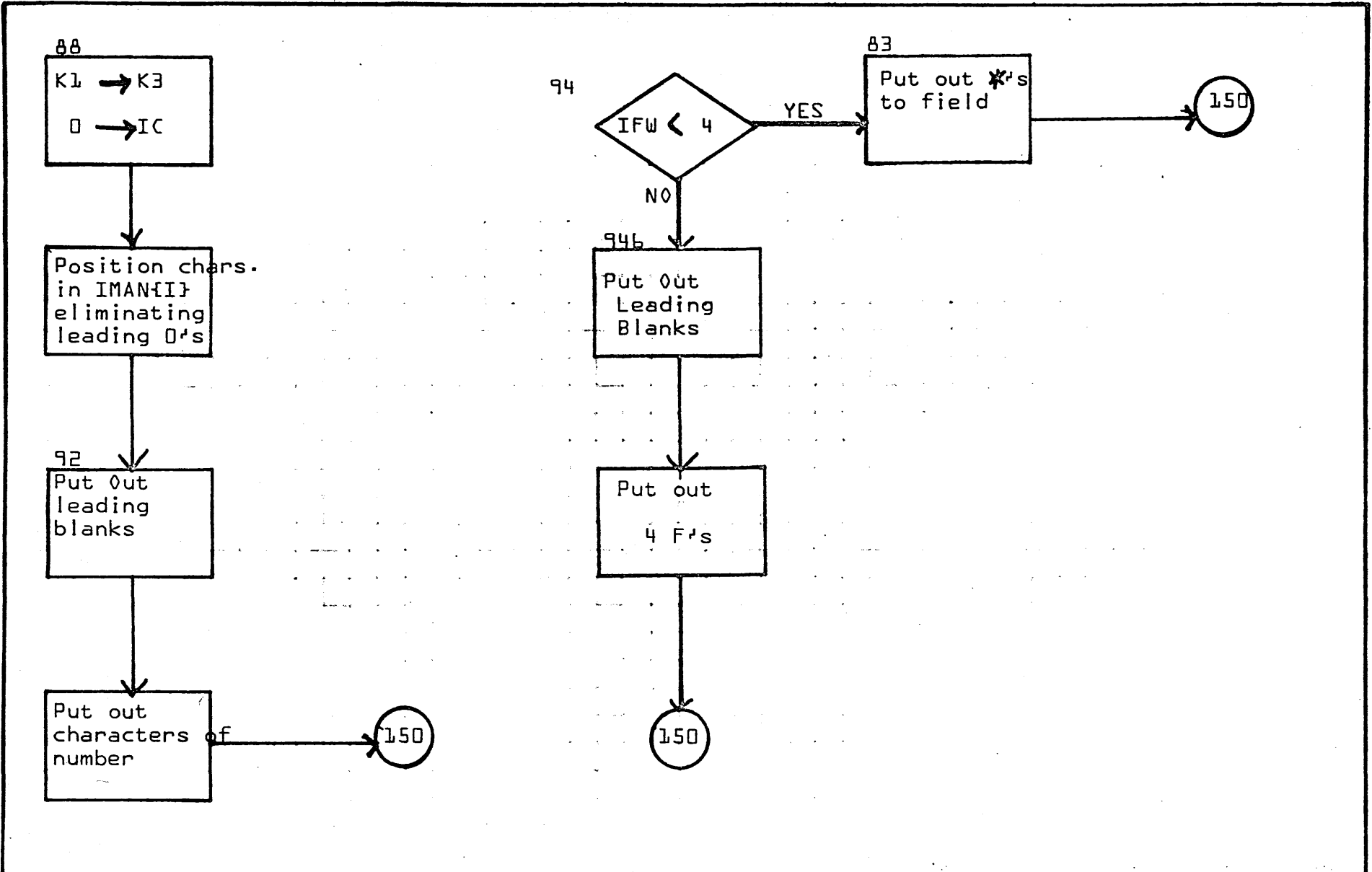


**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	<i>IMS</i>	MACH. TYPE	<i>1700</i>	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	<i>B8TRAN</i>	ISSUE DATE	<i>PAGE 80 OF 89</i>	PROJECT MGR.			
NUMBER		DATE		PROJECT NAME			
DRAWN BY		DATE		TASK NO.			
				TASK NAME			

A  
B  
C  
D



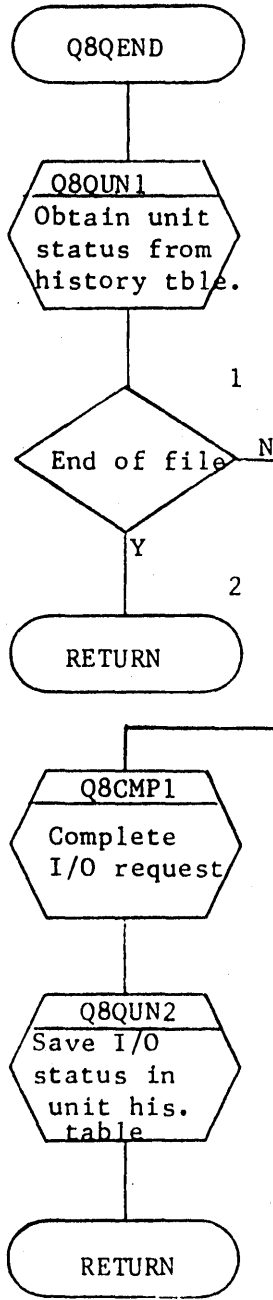
CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	QPTRAN	PAGE	81 OF 89	PROJECT MGR.			
NUMBER		ISSUE DATE		PROJECT NAME			
DRAWN BY		DATE		TASK NO.			
				TASK NAME			

6-113

1. Bit 12 in unit history table entry set means an end of file was detected during this I/O request.
2. Q8QEND entry when an end of file has been read is to be ignored.

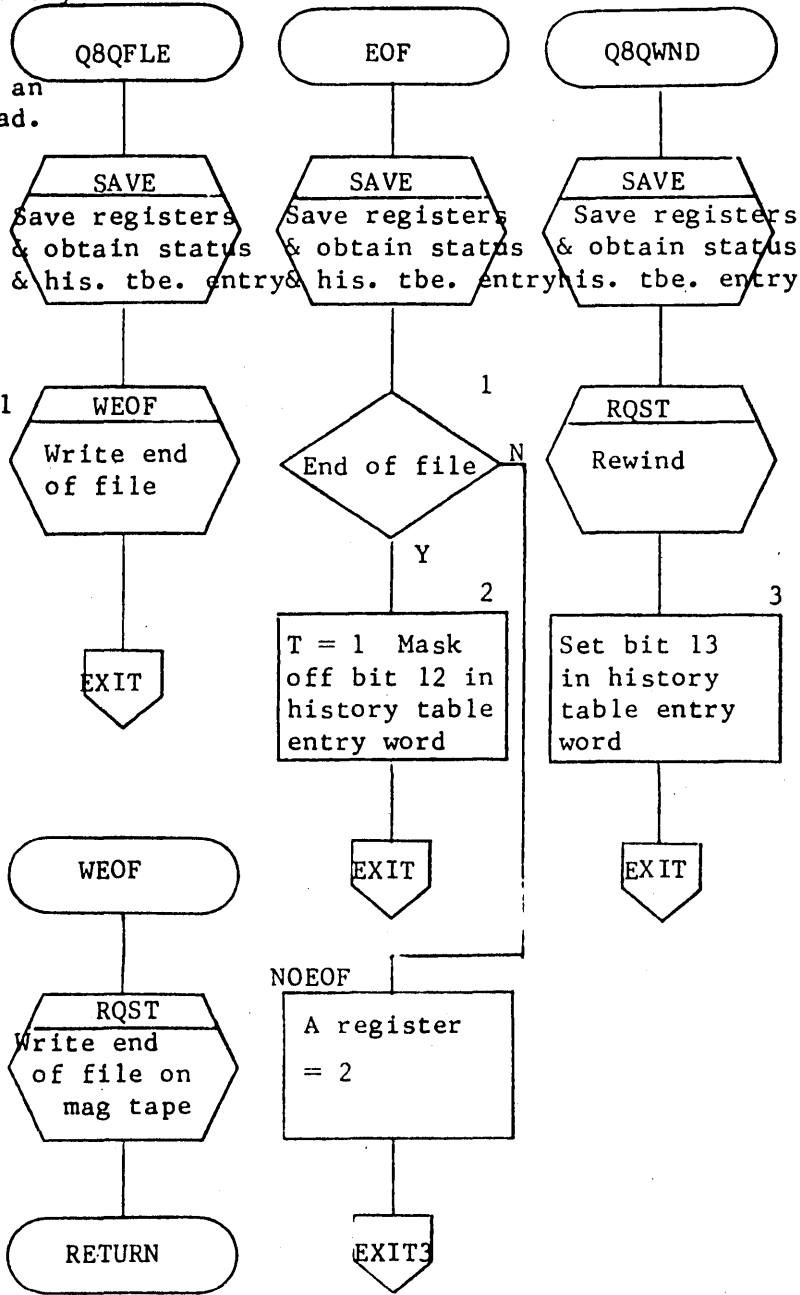


CONTROL DATA CORPORATION	DOCUMENT CLASS	MACH TYPE	PROJECT NO.	DATE
SOFTWARE DOCUMENT	DOCUMENT TITLE		PROJECT MGR.	APPROVED
SAMPLE CODE	Q8QEND	PAGE 82 OF 89	PROJECT NAME	REV
FLOWCHART	ISSUE DATE		TASK NO.	
DECISION TABLE			TASK NAME	
OTHER	DRAWN BY			

1. The history table entry word for the indicated unit will have bit 12 set if an end of file was read.

2. The A register contains T when EOF returns to the calling program.

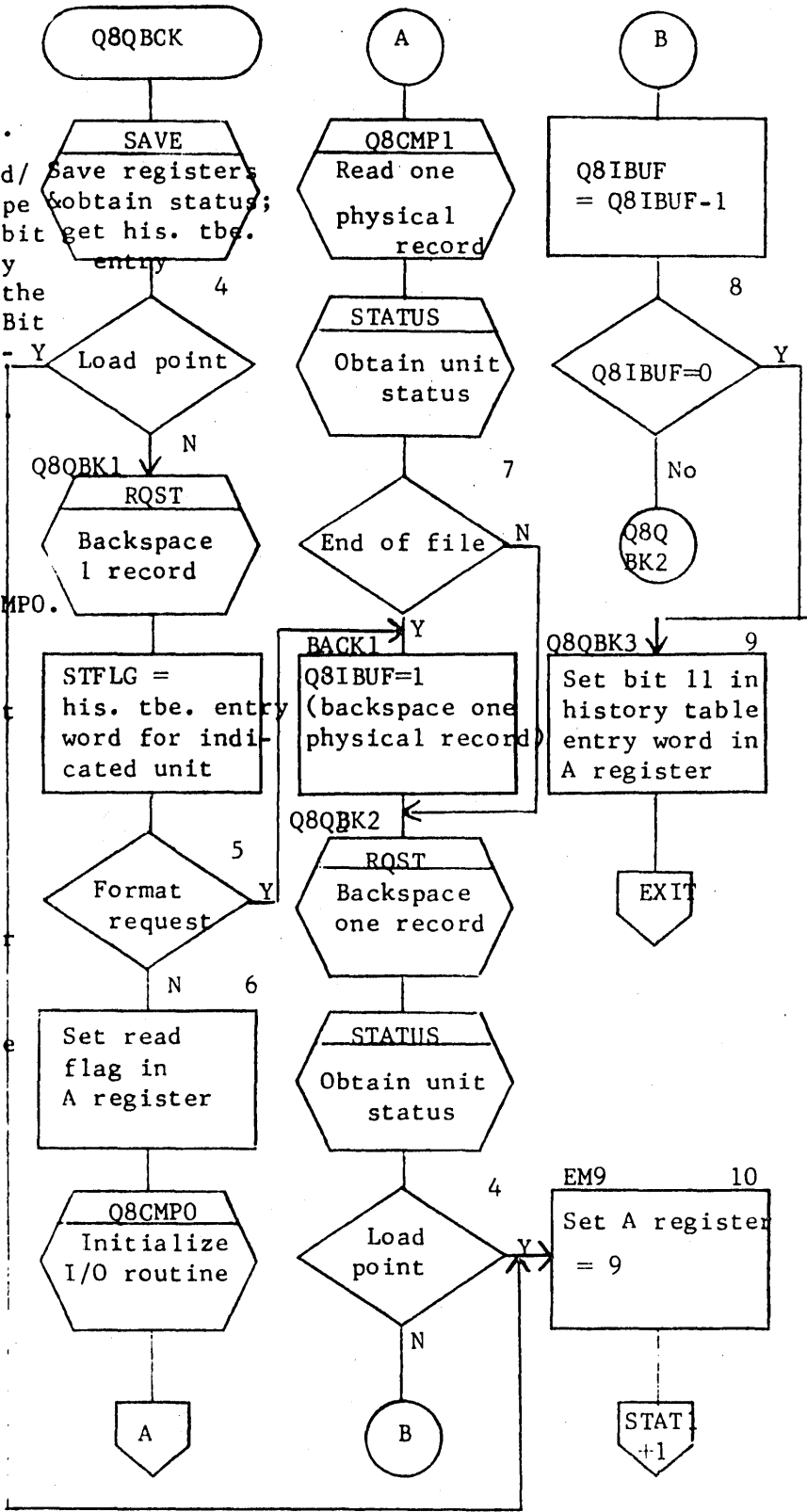
3. The history table entry word for the indicated unit will have bit 13 set after a rewind is implemented.



DATE	
APPROVED	
REV	
PROJECT NO.	
PROJECT MGR	
PROJECT NAME	
TASK NO.	
TASK NAME	
MACH. TYPE	
ISSUE DATE	
DATE	
DOCUMENT CLASS	
DOCUMENT TITLE	
NUMBER	
DRAWN BY	

CONTROL DATA CORPORATION  
 SOFTWARE DOCUMENT  
 SAMPLE CODE      
 FLOWCHART  
 DECISION TABLE  
 OTHER

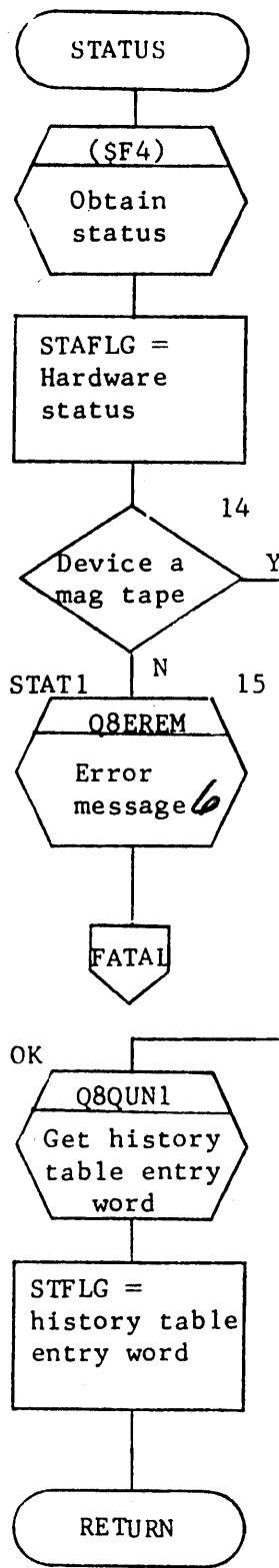
4. Bit 10 in the magnetic tape hardware status word (STAF LG) set means the tape is at load point.
5. The previous read/write request type is contained in bit 14 of the history table entry for the indicated unit. Bit 14 set means format request (1 physical record).
6. The A register now contains the history table entry with bit 10 set to indicate read to Q8CMPO.
7. Bit 11 of the hardware status word (STAF LG) set means an end of file was read.
8. For binary records Q8IBUF contains the number of physical records per logical records.
9. Bit 11 set in the history table entry indicates backspace.
10. Backspace at load point results in error message 9 and program termination.



DATE	
APPROVED	
REV	
PROJECT NO.	
PROJECT MGR.	
PROJECT NAME	
TASK NO.	
TASK NAME	
MACH. TYPE	
DOCUMENT CLASS	
DOCUMENT TITLE	
PAGE	84 of 89
ISSUE DATE	
NUMBER	
DRAWN BY	
DATE	
DOCUMENT CLASS	SOFTWARE DOCUMENT
SAMPLE CODE	<input type="checkbox"/>
FLOWCHART	<input checked="" type="checkbox"/>
DECISION TABLE	<input type="checkbox"/>
OTHER	<input type="checkbox"/>



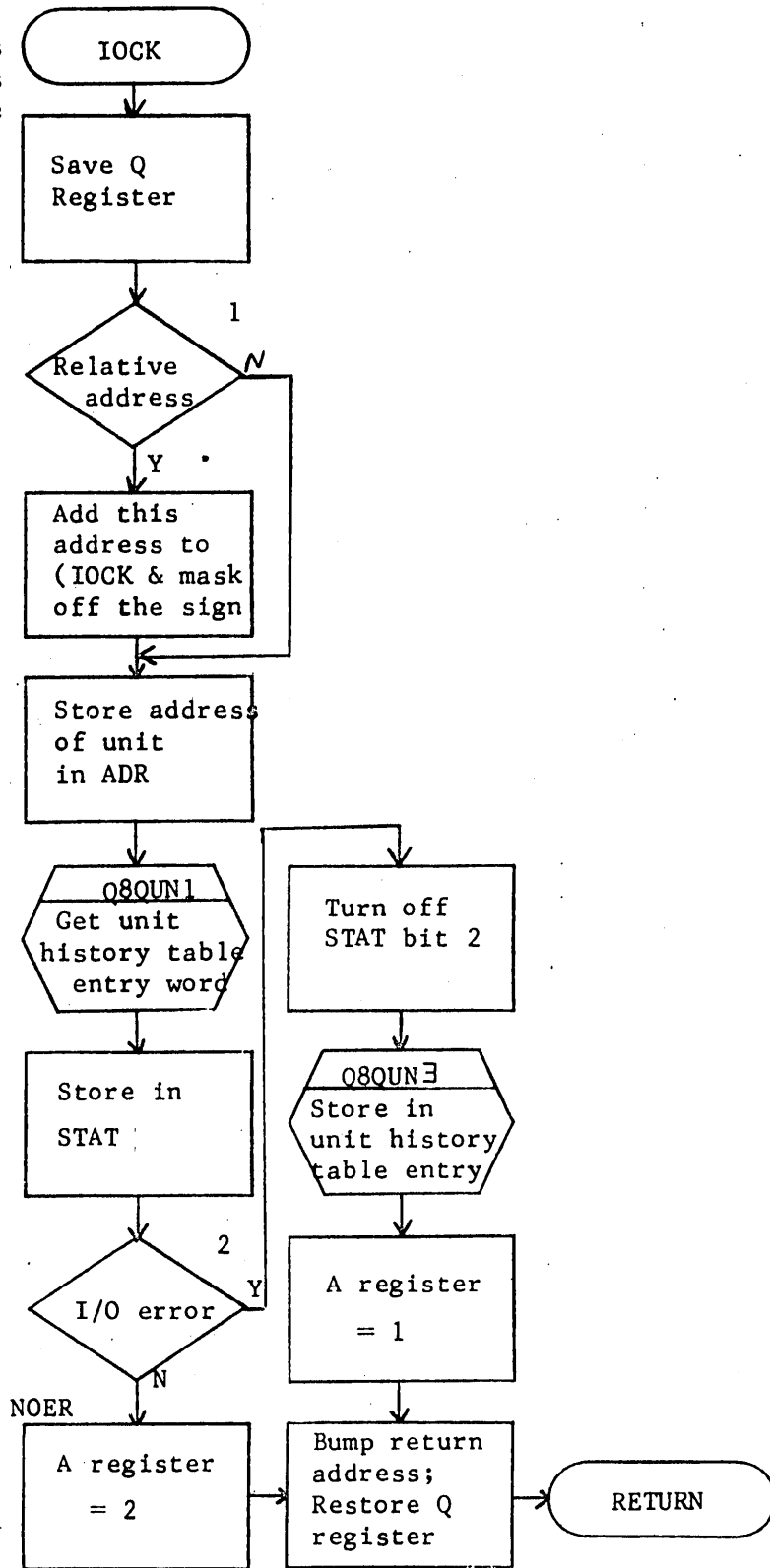
- 14. System Status request results in the Q register containing the device code in bits 4-7. Mag tape device code is 9.
- 15. Device not a mag tape results in error message 6 and program termination.



CONTROL DATA CORPORATION		DATE
SOFTWARE DOCUMENT	APPROVED	
SAMPLE CODE	REV	
FLOWCHART	PROJECT NO.	
DECISION TABLE	PROJECT MGR	
OTHER	PROJECT NAME	
	TASK NO.	
	TASK NAME	
DOCUMENT CLASS	MACH. TYPE	
DOCUMENT TITLE	ISSUE DATE	
TAPCON NUMBER	PAGE 86 OF 89	
DRAWN BY	DATE	



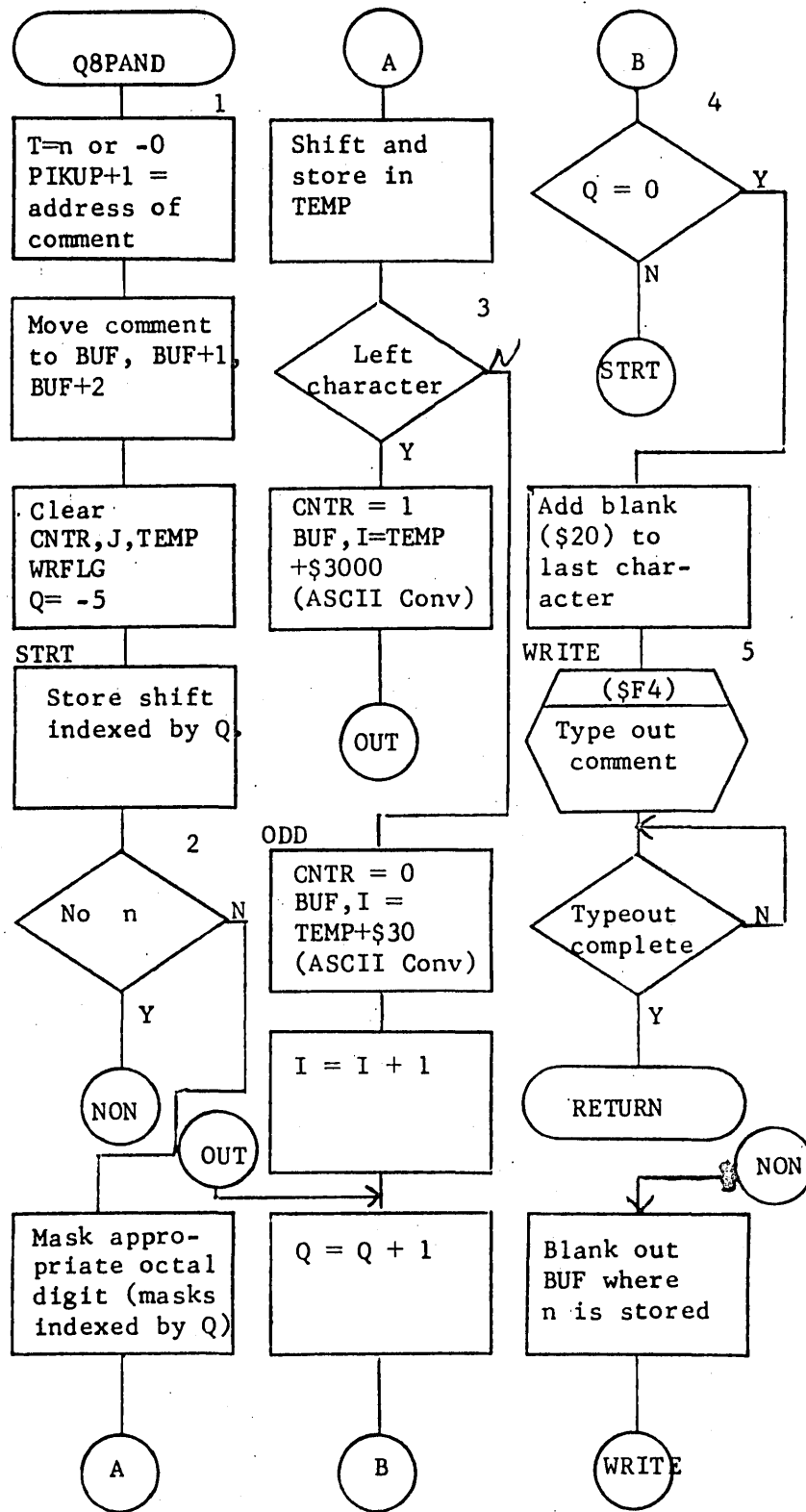
1. A negative address means this address is relative to the given address.
2. Bit 2 in the unit history table entry word set indicates an I/O error occurred.



DATE	
APPROVED	
REV	
PROJECT NO.	
PROJECT MGR.	
PROJECT NAME	
TASK NO.	
TASK NAME	
DOCUMENT CLASS	IMS
DOCUMENT TITLE	IOCK
NUMBER	87 of 89
ISSUE DATE	
DATE	
DRAWN BY	
MACH. TYPE	1700
CONTROL DATA CORPORATION SOFTWARE DOCUMENT	
SAMPLE CODE	
FLOWCHART	
DECISION TABLE	
OTHER	



1. Either PAUSE or STOP is typed out.
2. T 0 means no n is required.
3. CNTR is a flag indicating left or right half of ASCII equivalent of the octal 5 digit number. CNTR=0 for left half.
4. Conversion to ASCII octal complete?
5. PAUSE, PAUSE n, STOP, STOP n.



DATE	
APPROVED	
REV	
PROJECT NO.	
PROJECT MGR.	
PROJECT NAME	
TASK NO.	
TASK NAME	
DOCUMENT CLASS	
DOCUMENT TITLE	
Q8PAND	
NUMBER	
DRAWN BY	
ISSUE DATE	
DATE	
MACH. TYPE	
PAGE 89 of 89	

**CONTROL DATA CORPORATION**

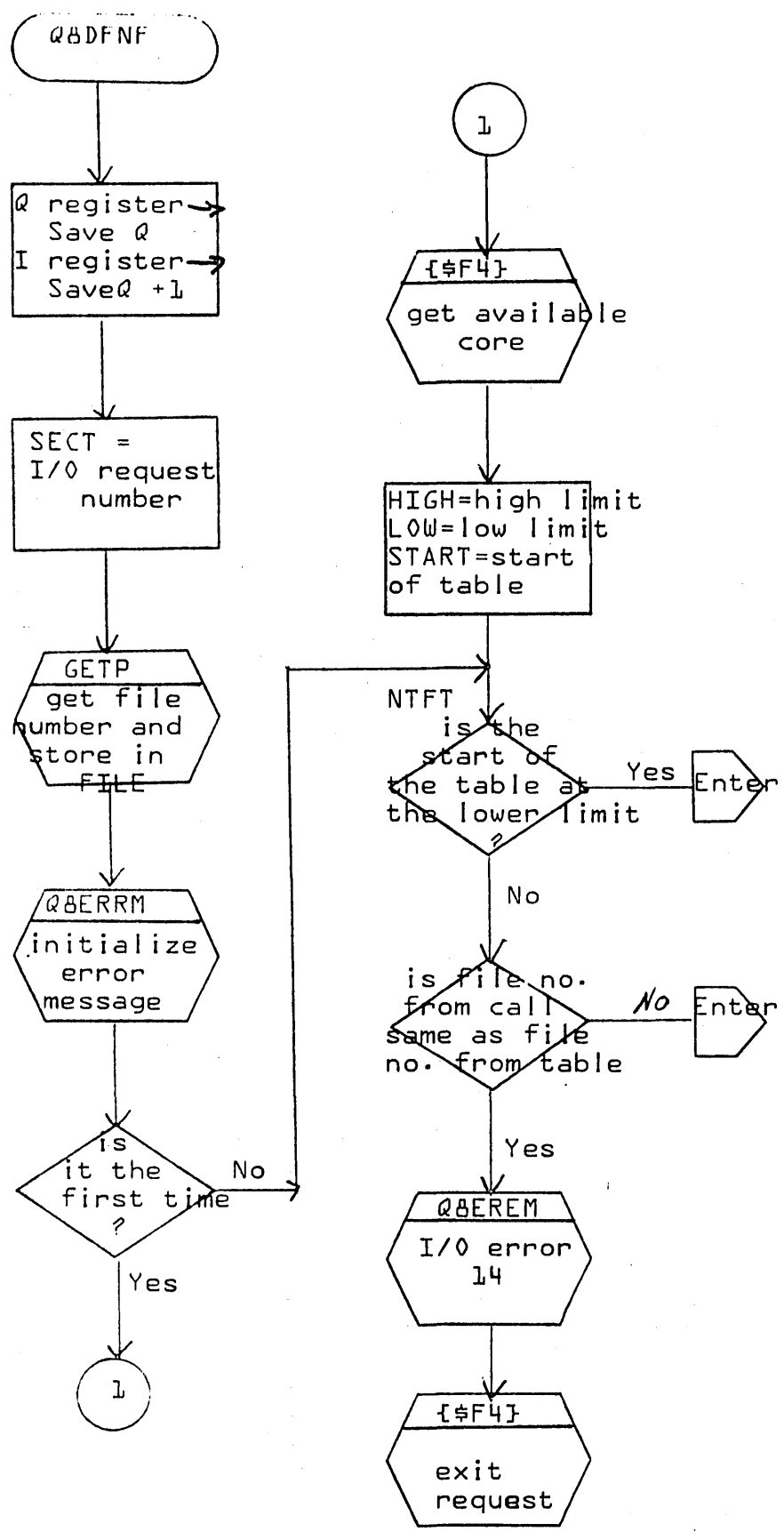
SOFTWARE DOCUMENT

SAMPLE CODE

FLOWCHART

DECISION TABLE

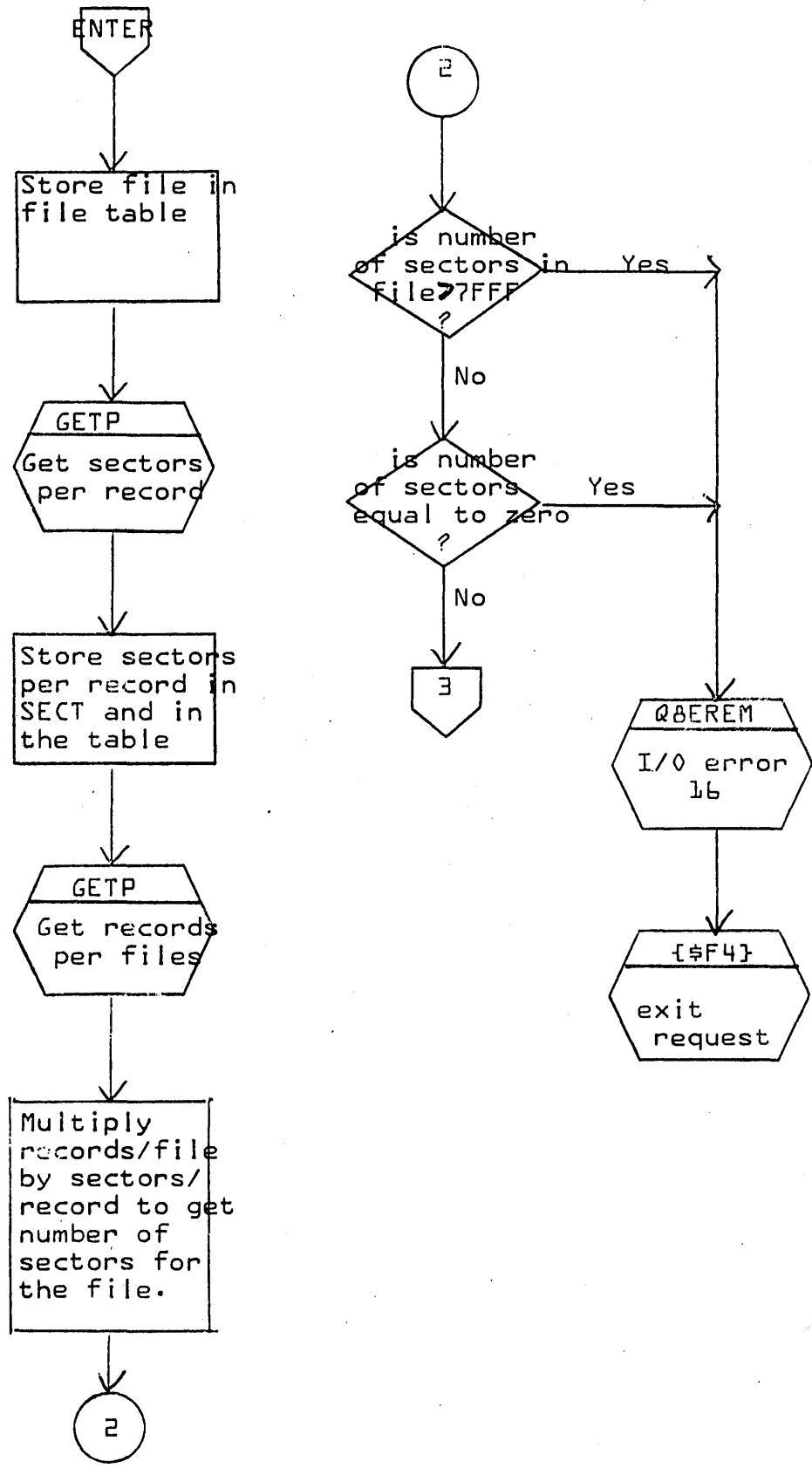
OTHER



CONTROL DATA CORPORATION SOFTWARE DOCUMENT		DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.		APPROVED		DATE	
SAMPLE CODE		DOCUMENT TITLE	Q8DFI0			PROJECT MGR.		REV			
FLOWCHART		NUMBER	100			PROJECT NAME					
DECISION TABLE		DRAWN BY				TASK NO.					
OTHER						TASK NAME					

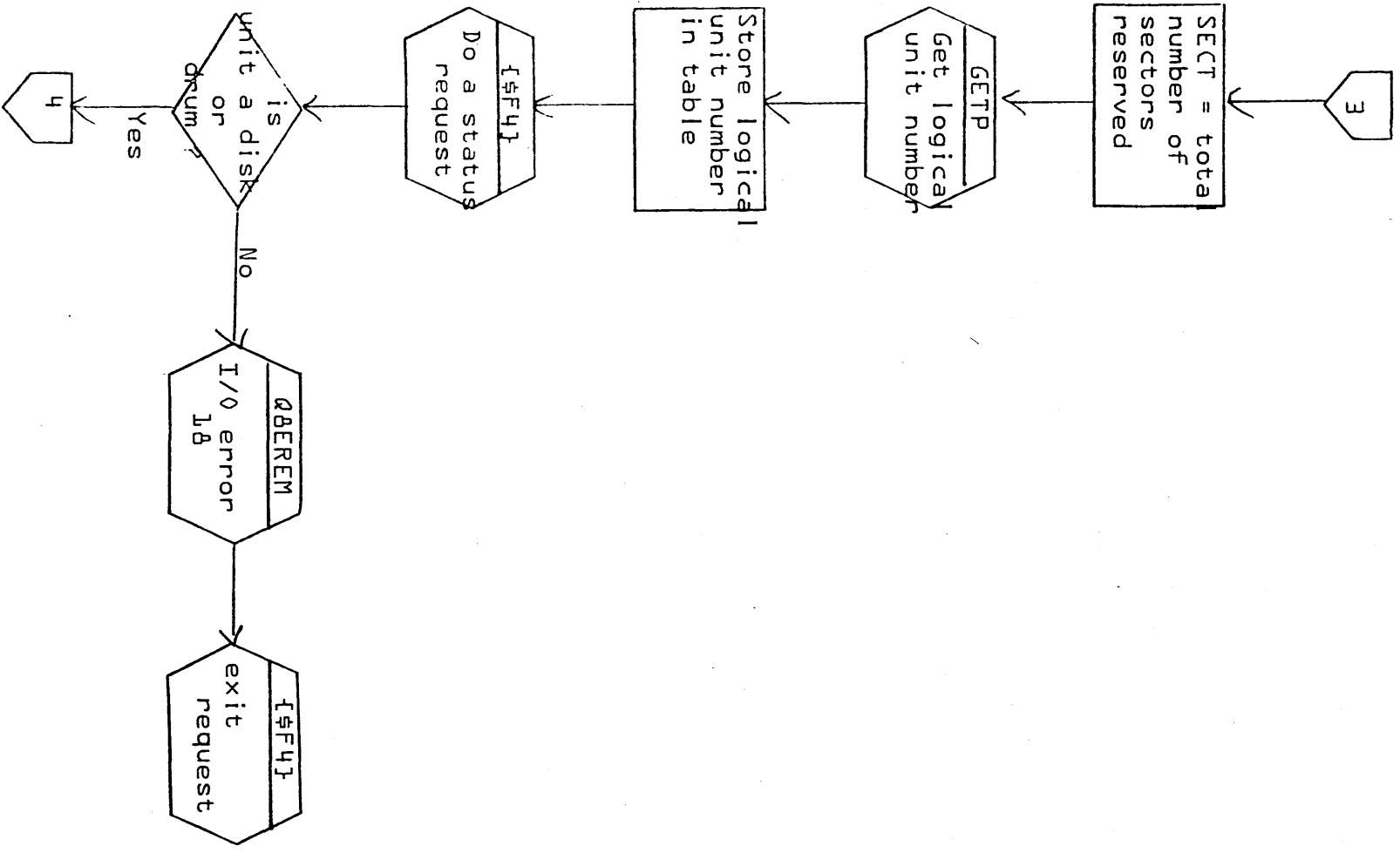
PRINTED IN USA

1 2 3 4 5



CONTROL DATA CORPORATION SOFTWARE DOCUMENT		DOCUMENT CLASS IMS	MACH. TYPE 1700	PROJECT NO.	APPROVED	DATE
SAMPLE CODE <input type="checkbox"/>		DOCUMENT TITLE Q8DF10	PAGE 2 of 8	PROJECT MGR.		
FLOWCHART <input type="checkbox"/>		NUMBER	ISSUE DATE	PROJECT NAME		
DECISION TABLE <input type="checkbox"/>		DRAWN BY	DATE	TASK NO.		
OTHER <input type="checkbox"/>				TASK NAME		

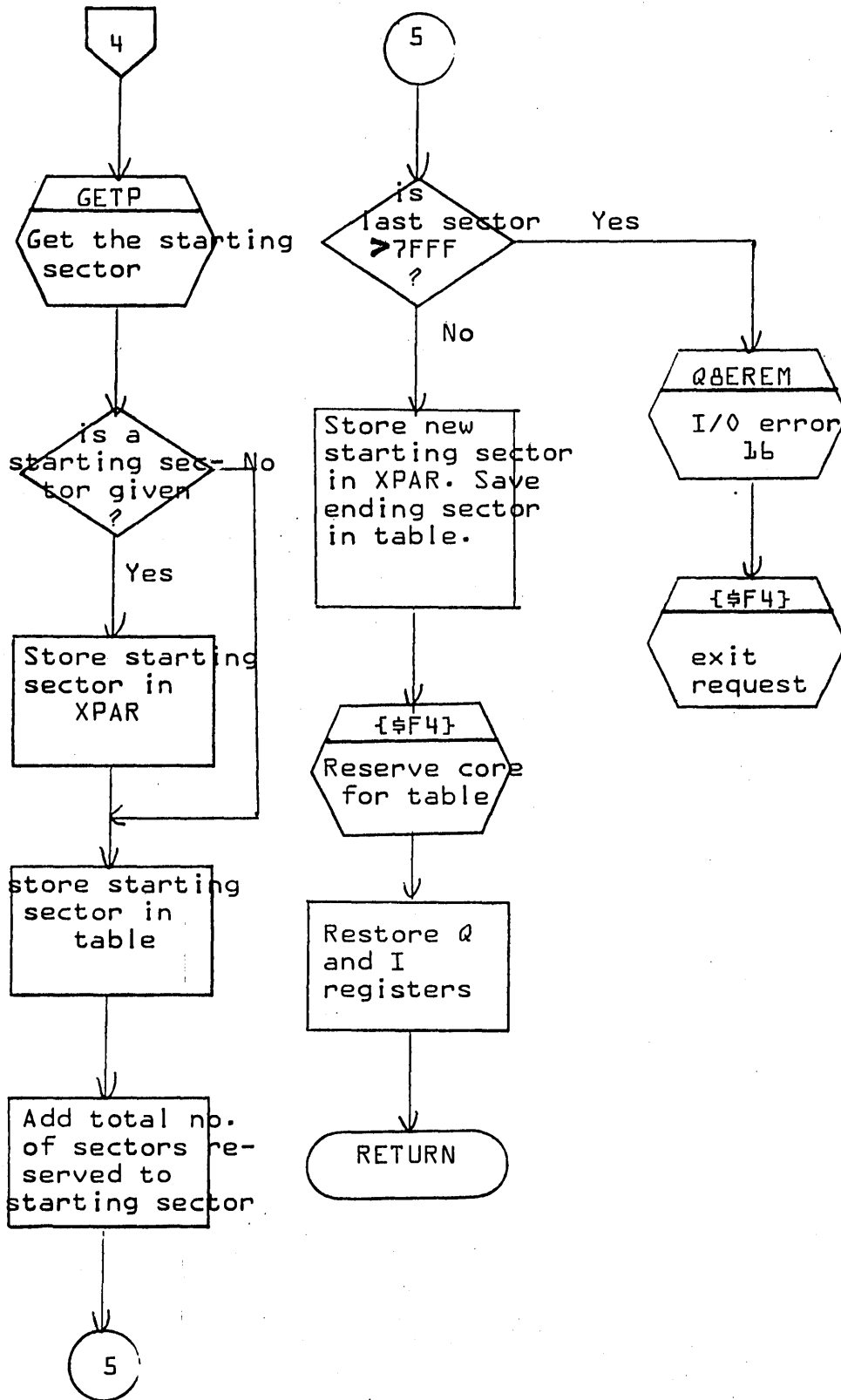
A B C D



A  
B  
C  
D

<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	QADFIO		PAGE 3 OF 8	PROJECT MGR.			
	NUMBER		ISSUE DATE		PROJECT NAME			
	DRAWN BY		DATE		TASK NO.			
					TASK NAME			

6-124



CONTROL DATA CORPORATION		DOCUMENT CLASS	IMS	MACH. TYPE	J700	PROJECT NO.		REV		APPROVED		DATE	
SOFTWARE DOCUMENT		DOCUMENT TITLE	Q8DFIO			PROJECT MGR.							
SAMPLE CODE <input type="checkbox"/>		NUMBER				PROJECT NAME							
FLOWCHART <input type="checkbox"/>		ISSUE DATE				TASK NO.							
DECISION TABLE <input type="checkbox"/>		DRAWN BY				TASK NAME							
OTHER <input type="checkbox"/>													

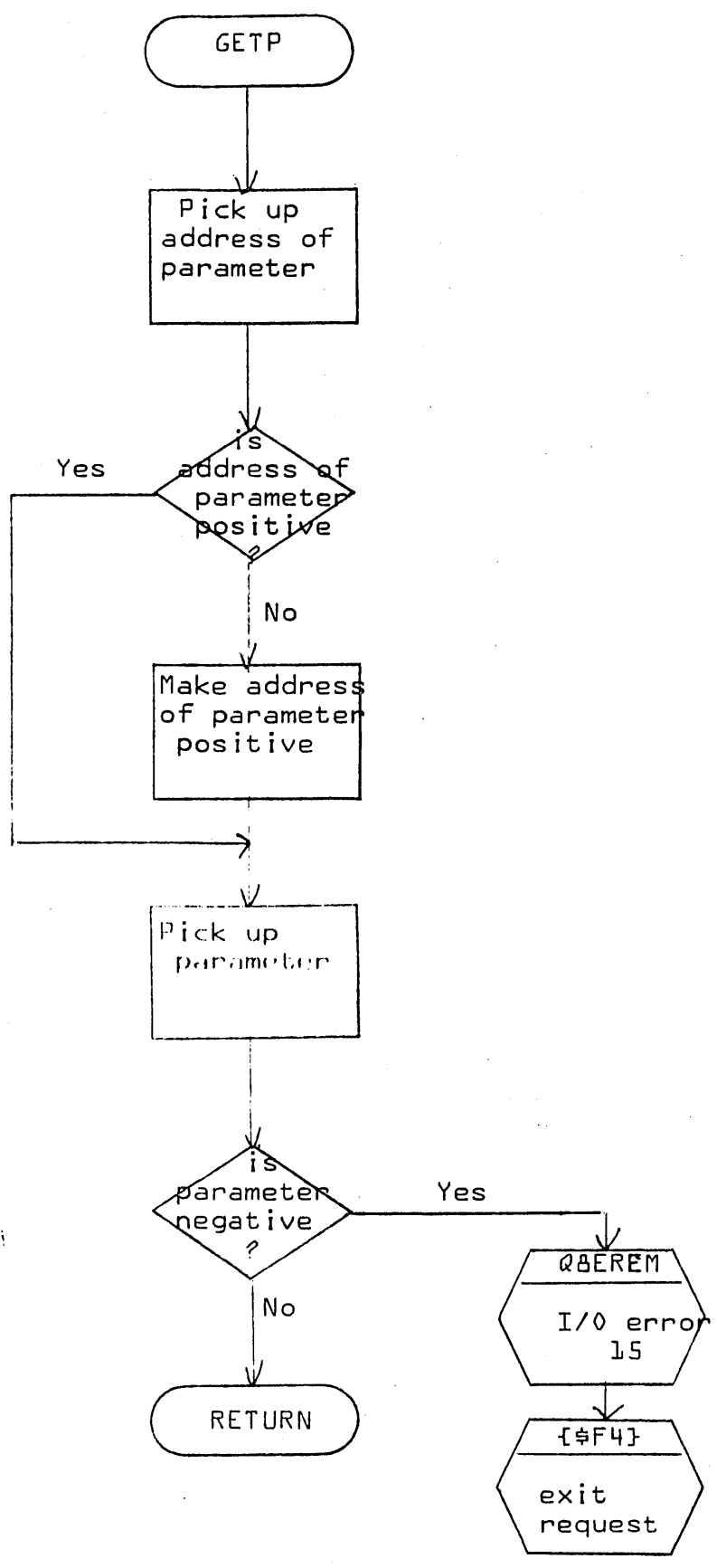
A

B

C

D

1 2 3 4 5



DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.		REV	APPROVED	DATE
DOCUMENT TITLE	Q8DFI0		PAGE 5 OF 8	PROJECT MGR.				
NUMBER		ISSUE DATE		PROJECT NAME				
DRAWN BY		DATE		TASK NO.				
				TASK NAME				

CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE

FLOWCHART

DECISION TABLE

OTHER

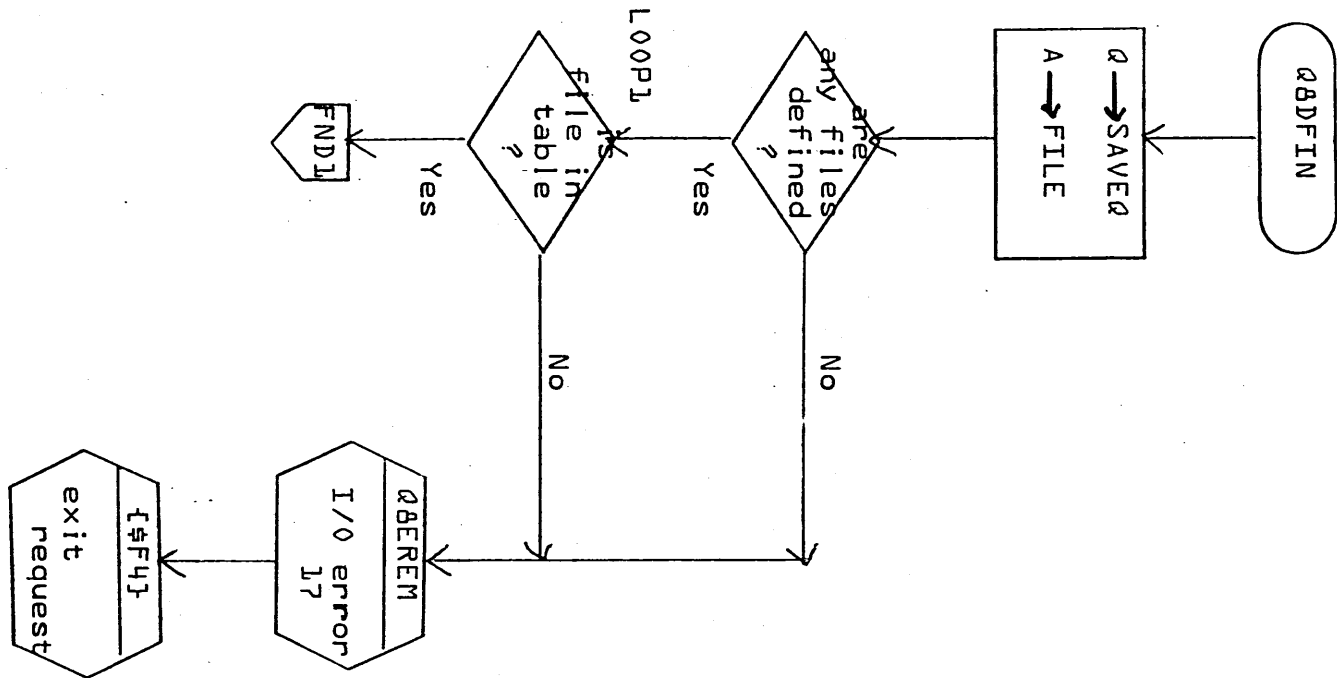


A

B

C

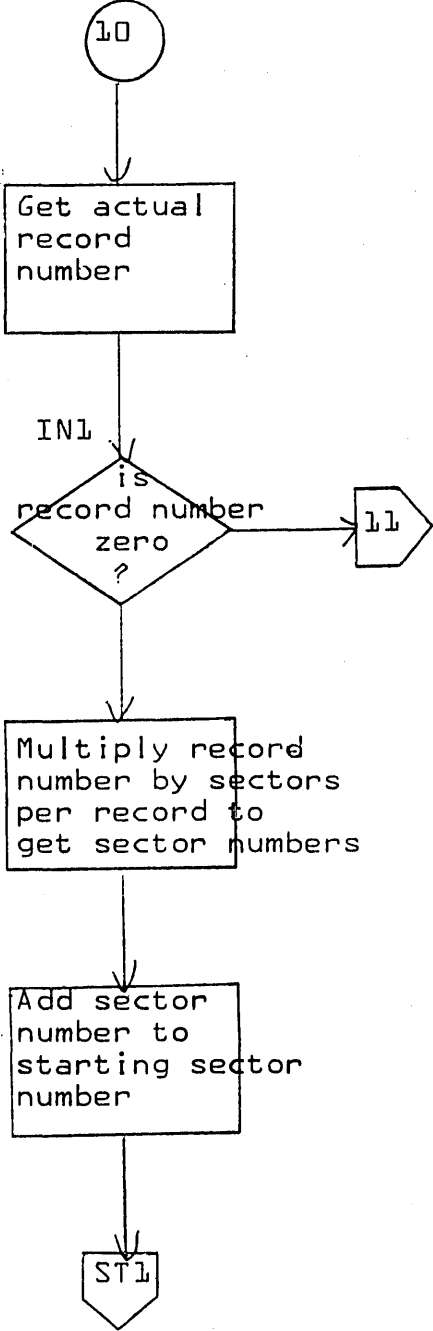
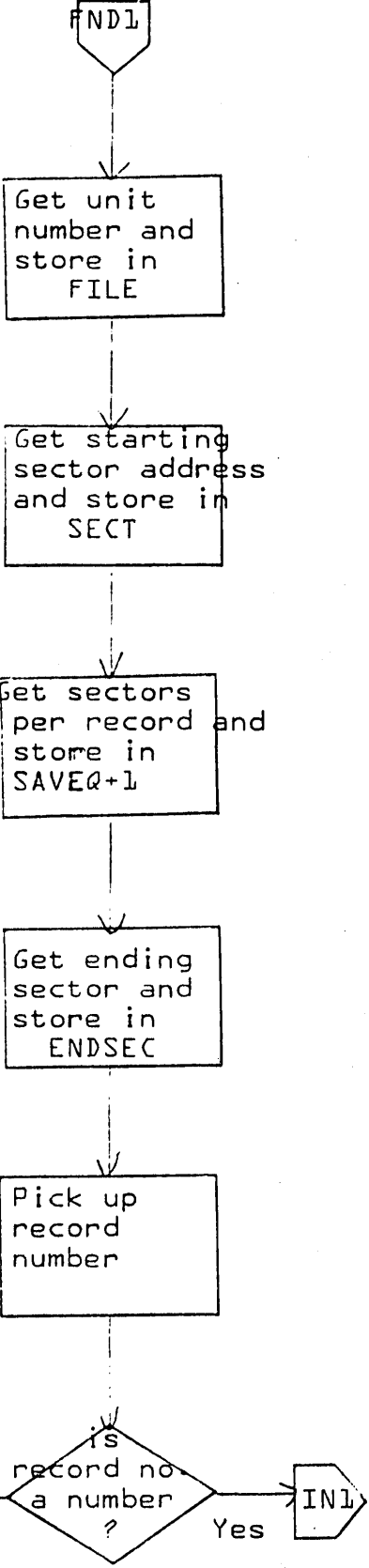
D



**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	Q8DFIO			PROJECT MGR.			
NUMBER		ISSUE DATE	PAGE 6 OF 8	PROJECT NAME			
DRAWN BY		DATE		TASK NO.			
				TASK NAME			



CONTROL DATA CORPORATION SOFTWARE DOCUMENT		DOCUMENT CLASS IMS	MACH. TYPE J700	PROJECT NO.	APPROVED	DATE
SAMPLE CODE <input type="checkbox"/>		DOCUMENT TITLE QADE10	PAGE 7 OF 8	PROJECT MGR.		
FLOWCHART <input type="checkbox"/>		NUMBER	ISSUE DATE	PROJECT NAME		
DECISION TABLE <input type="checkbox"/>		DRAWN BY	DATE	TASK NO.		
OTHER <input type="checkbox"/>				TASK NAME		
REV						

A

B

C

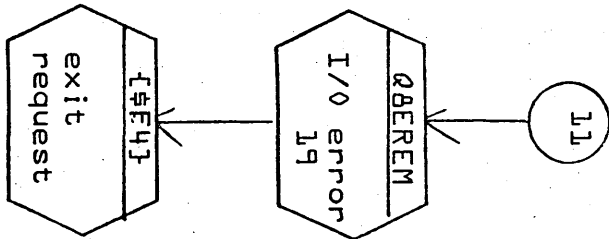
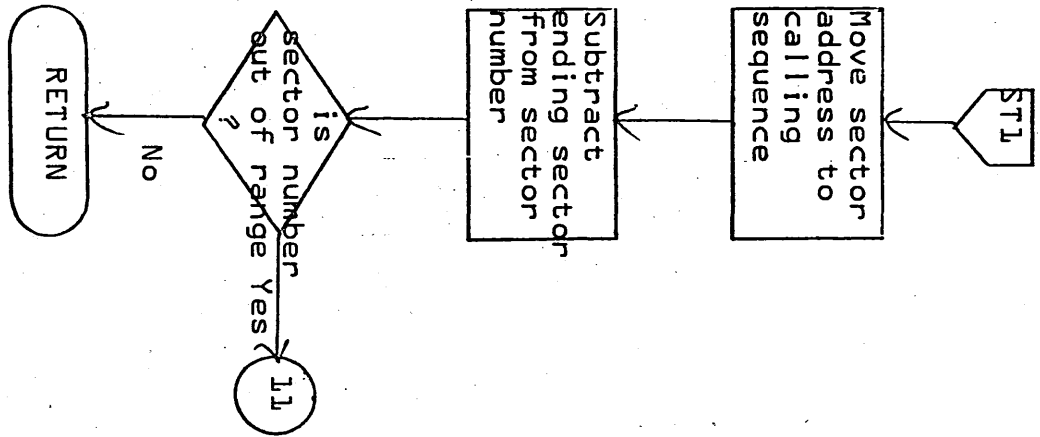
D

A

B

C

D



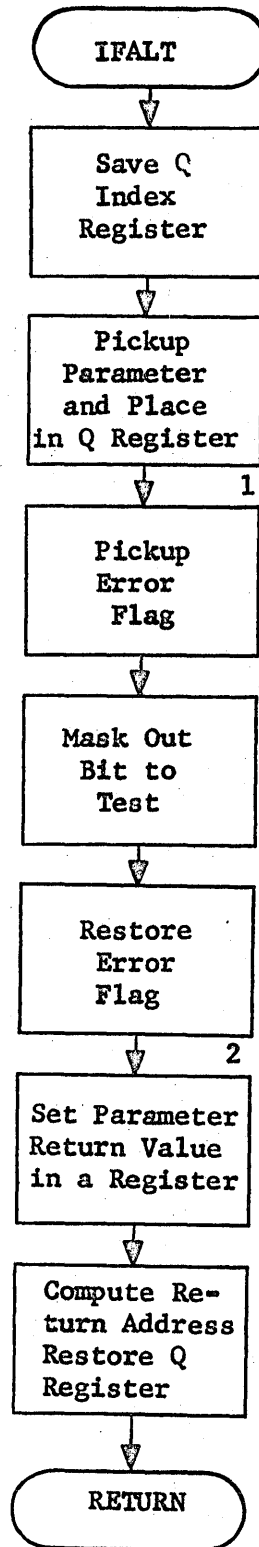
**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

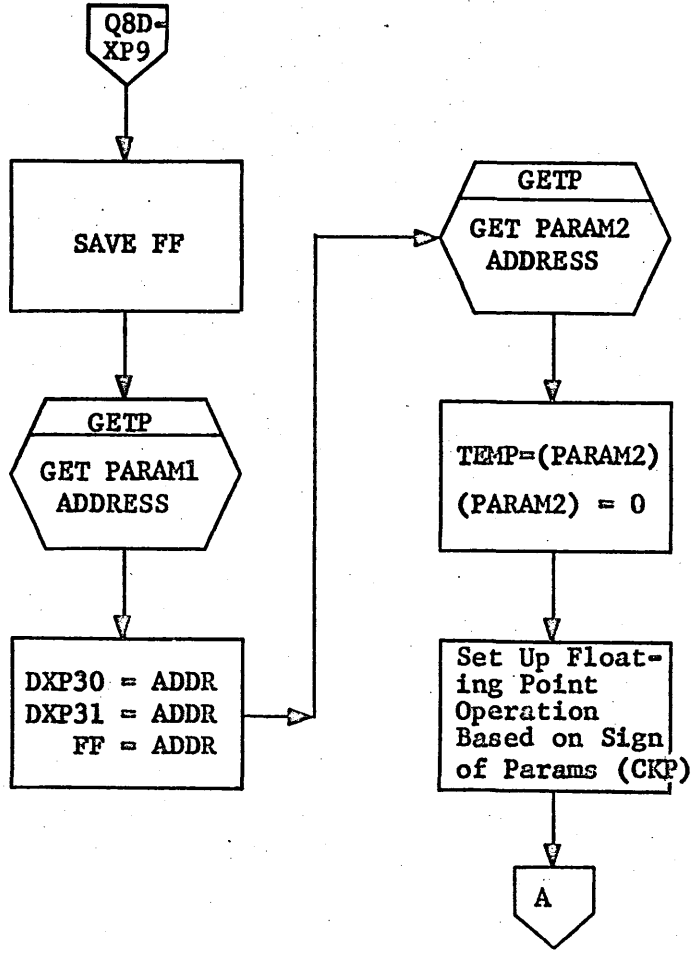
- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

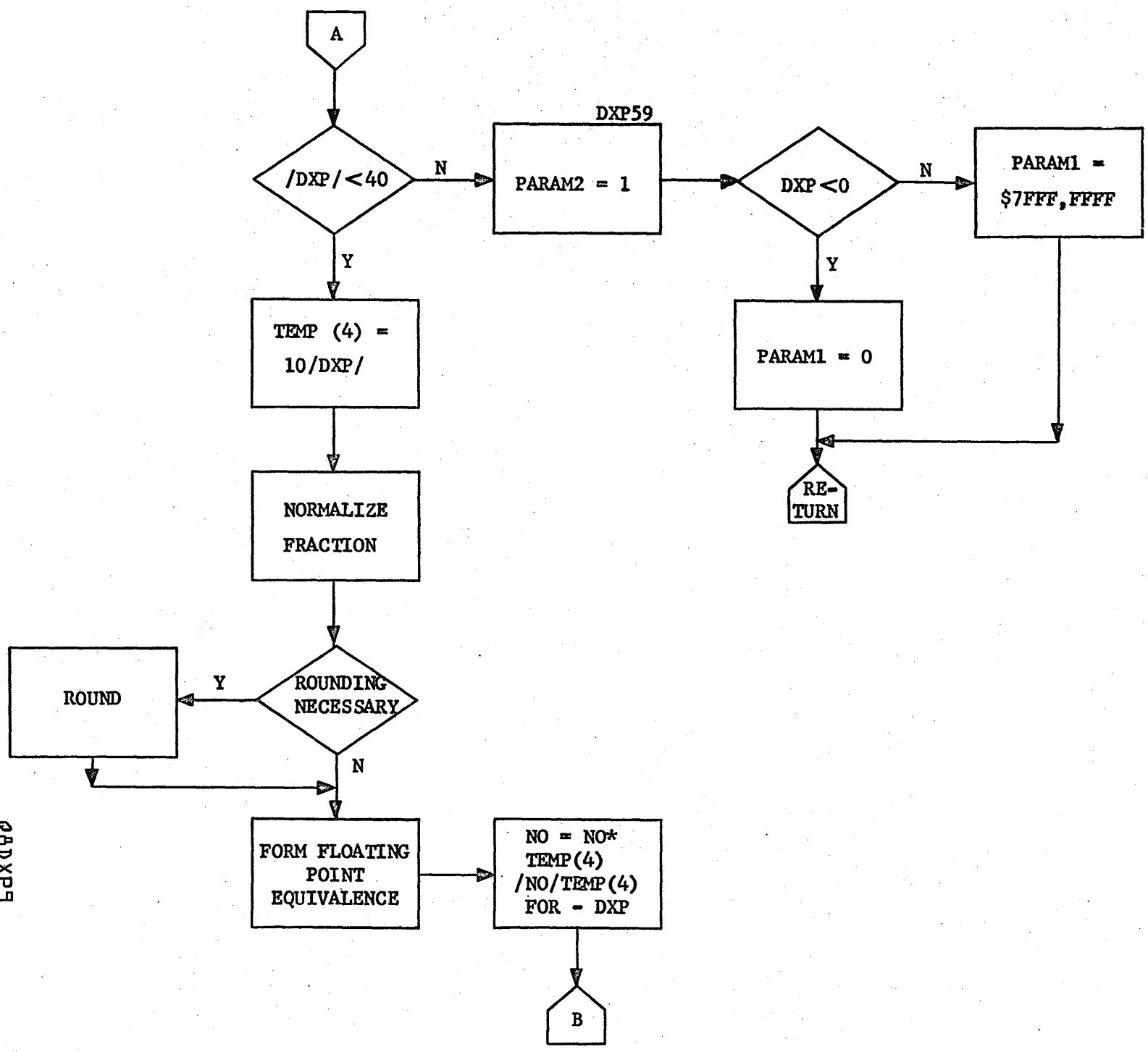
DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	QBDFIO			PROJECT MGR.			
NUMBER		ISSUE DATE	PAGE 8 OF 8	PROJECT NAME			
DRAWN BY		DATE		TASK NO.			
				TASK NAME			

DOCUMENT CLASS IMS PAGE NO. 6-130  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

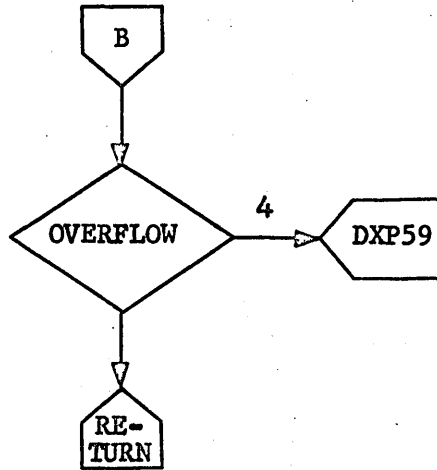
1. The routines FLOAT and DFLOT puts the error flag in low core location \$C8 from which IFALT can pick it up.
2. IFALT returns the information as to whether the error occurred in the A register.

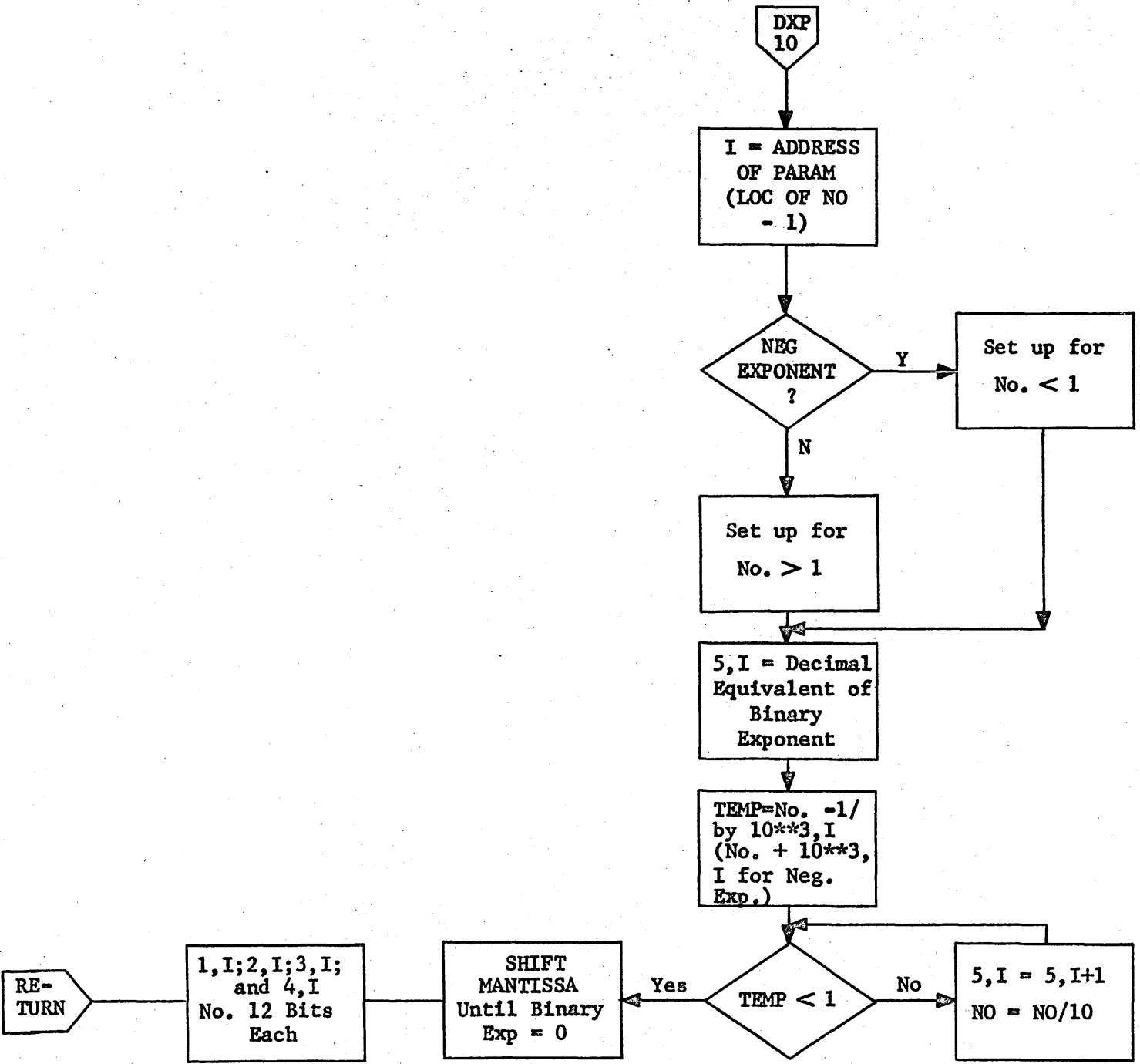






DOCUMENT CLASS IMS PAGE NO. 6-133  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. COO5\*3.1 A/B MACHINE SERIES 1700







DOCUMENT CLASS IMS PAGE NO. 7-1  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. CO05 V2.0 MACHINE SERIES 1700

## CHAPTER 7

## TABLE OF CONTENTS

## 7.0 Tables

## 7.1 Symbol Table {SYMTAB}

- 7.1.1 Format
- 7.1.2 Description

## 7.2 Loop Structure Table {L00PT}

- 7.2.1 Format
- 7.2.2 Description

## 7.3 FORTRAN Internal Code

## 7.4 Other Tables

- 7.4.1 Specification Table {ISTAB}
- 7.4.2 Table of Symbol Table Presets {ISET}

## 7.5 Output Format, PHASE A

- 7.5.1 General Form
- 7.5.2 Statement Types
- 7.5.3 Arithmetic Tree
  - 7.5.3.1 General Form
  - 7.5.3.2 Operand Types

## 7.6 Loader Maps

- 7.6.1 MS FORTRAN V2.0A Loader Maps
- 7.6.2 MS FORTRAN V2.0B Loader Maps

## 7.7 Description of Common Blocks

- 7.7.1 Master Labelled Common Block
- 7.7.2 Specification Table Labelled Common Block
- 7.7.3 Phase A Blank Common Block
- 7.7.4 Phase 4 Labelled Common Block
- 7.7.5 Phase 4 Blank Common Block
- 7.7.6 Phase C, D, E Labelled Common Block
- 7.7.7 Phase C and phase D/E Blank Common Block
- 7.7.8 Compressed Symbol Table Blank Common Block



DOCUMENT CLASS IMS PAGE NO. 7-3  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

7.1.2 Meaning and Use of Symbol Table Entries

ITEM NAME	PHASES IN WHICH SET	PHASES IN WHICH USED	DESCRIPTION AND USE
*ISYM	A	A, B, C, E	The symbol name. After Pass 2, an index to the symbol's entry in SYMTAB is substituted for all symbol references. External names and constants are retained in this entry throughout compilation.
ICLASS	A	A, B, C, E	Specifies the type of entry. 0 = Unassigned 1 = Variable or FUNCTION definition name 2 = Constant 3 = Intrinsic function 4 = Statement function 5 = External function 6 = External subroutine 7 = Statement label 8 = Subroutine definition name 9 = Program definition name
ITYPE	A	A, B, C, E	If applicable, specifies the arithmetic mode of the symbol. 0 = Unassigned 1 = Integer 2 = Real {floating point} 3 = Double Precision
*IPART	A	A, B	If 0, symbol is not a byte. If 1, symbol is a BYTE. If 2, symbol is a SIGNED BYTE.
IPARTL	A	A, B	The left most bit of a byte.
IPARTR	A	A, B	The right most bit of a byte.
IDUM	A	A, B, C, E	If non-zero, symbol is a dummy argument.
IDIM	A	A, B, C, E	When ICLASS is 1, entry contains the dimensionality of the array {if undimensioned = 0}. The actual dimensions are in the ISTAB table. The index to this table for this array is in the ISTABX table.

\*See note 1 {page 7-5}

CONTROL DATA CORPORATION  
LA JOLLA RESOURCE CENTER

---

DIVISION

DOCUMENT CLASS IMS PAGE NO. 7-3A  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

ITEM NAME	PHASES IN WHICH SET	PHASES IN WHICH USED	DESCRIPTION AND USE
KRFCNT	A,B	A,B,C,E	Number of references to this array or variable or constant. Used in placing this entity optimally vis-a-vis one word relative addressing.

DOCUMENT CLASS IMS PAGE NO. 7-4  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

ITEM NAME	PHASES IN WHICH SET	PHASES IN WHICH USED	DESCRIPTION AND USE
ISNOL	A	A,B	If ICLASS is 7, the statement number of this label.
IREF	A	B	If ICLASS is 2, non-zero means the constant is to be materialized.
ICOM	A	A,B,C,D,E	ICOM is the index to the ICOMT entry for the COMMON variable. If zero, this entry is not in COMMON.
IEQVX	A	A,B	IEQVX is the IEQV table index for this equivalenced entry. Used in generating definition points and for storage allocation.
KDUMY	A,B	A,B	This entry is a dummy argument-increment entry.
IDATAS	A	E	If non-zero, the variable was initialized in a data statement.
IRSA	C,D,E	C,D,E	The relative storage location of this entry.
ICOMTX	A	A,B	The SYMTAB index to a thread of the variables and arrays in a COMMON block. The first array of the thread is ICOMBX in the table ICOMT. ICOMTX is the SYMTAB index to the array declared prior to this array.
IEXT	A	A,B,C,D,E	If non-zero, this symbol appears in an EXTERNAL statement.
ISNGL	A	A,B	If non-zero, this symbol appears in a SINGLE statement.
IREL	A	A,B,C,D,E	If non-zero, this symbol appears in a RELATIVE statement.
INDUCV	A	A	Non-zero if variable is currently an induction variable.



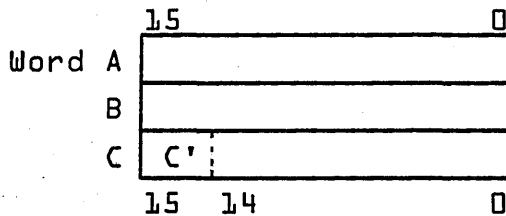
CONTROL DATA CORPORATION  
 LA JOLLA RESOURCE CENTER DIVISION

DOCUMENT CLASS IMS PAGE NO. 7-5  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. COO5\*3.1 A/B MACHINE SERIES 1700

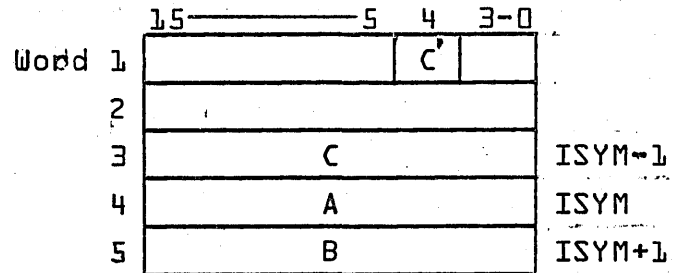
ITEM NAME	PHASES IN WHICH SET	PHASES IN WHICH USED	DESCRIPTION AND USE
IARGNO	A	A,B	Number of arguments in function call.
ISFARG	A	A,B	Set for statement function argument.
KELSIZ	A,B	A,B	If ICLASS is 1, the element size minus one.
ITILF	A	A	If ICLASS is 3, is the index to the in-line functions table.

Note 1: Double precision constants are 3 words long and are stored in the symbol table as follows:

Double Precision Constant



Symbol Table Entry

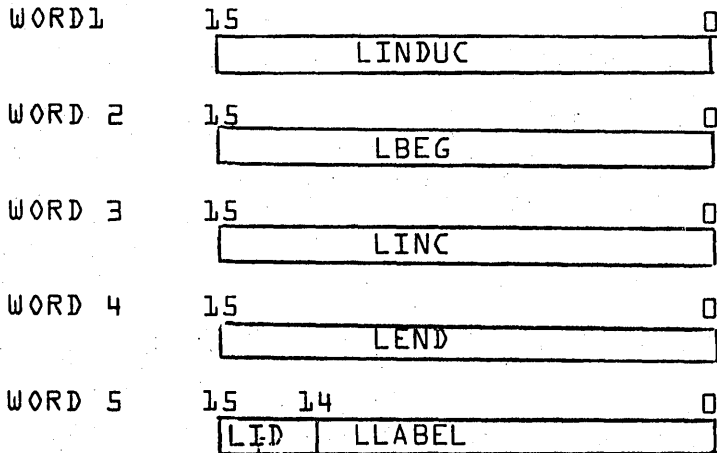


C' = High order bit of word 3 which must be preserved in the high order bit of IPART.

DOCUMENT CLASS IMS PAGE NO. 7-6  
 PRODUCT NAME 1700 Mass Storage FORTRAN  
 PRODUCT MODEL NO. C005 V2.0 MACHINE SERIES 1700

7.2 Loop Structure Table {Loopt}

7.2.1 Loop Structure Table Format



7.2.2 LOOP Structure Table {L00PT}

The loop structure table is available in pass A. A BEGIN DO entry is entered in it when Pass A encounters a DO statement. An END DO entry is entered upon encountering a label which terminates the loop. L00PT allows room for 3D loops. The L00PT index is L00PTX.

BEGIN DO ENTRY

ITEM	NO. BITS	DESCRIPTION AND USE
LINDUC	15	The SYMTAB index to the induction variable.
LBEG	15	The SYMTAB index to the initial value.
LINC	15	The SYMTAB index to the increment.
LID	1	≠0 decrementing loop
LLABEL	14	The SYMTAB index to label entry for terminating statement.



DOCUMENT CLASS IMS PAGE NO. 7-7  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

7.3 FORTRAN Internal Code

	CHARACTER	CODE {Decimal}
Digits	0-9 . . . . .	0-9
Letters	A-Z . . . . .	10-35
Dollar sign	\$ . . . . .	36
Period	. . . . .	37
Plus sign	+ . . . . .	38
Minus sign	- . . . . .	39
Equal sign	= . . . . .	40
Left parenthesis	{ . . . . .	41
Right parenthesis	} . . . . .	42
Comma	, . . . . .	43
Slash	/ . . . . .	44
Asterisk	* . . . . .	45
Blank	△ . . . . .	46
End of statement	△ . . . . .	47
Single quote	' . . . . .	48

All other characters should be translated to blank and an appropriate message output.

7.4 Other Tables

7.4.1 Specification Table {ISTAB}

The current index for this table is ISTAB2. The maximum table size {to be compared against ISTAB2} is ISTABS and is 150. The entries are pointers to integer constants which are dimensions. For a given symbol, its SYMTAB entry is used to compute the entry in ISTABX table {ISYMX/{2\*ISYMF1}+1} which gives the index to ISTAB of the first dimension. The SYMTAB entry IDIM gives the dimensionality.

DOCUMENT CLASS IMS PAGE NO. 7-8  
PRODUCT NAME 1700 Mass Storage FORTRAN  
PRODUCT MODEL NO. C005 V2.0 MACHINE SERIES 1700

## 7.4.2 Table of Symbol Table Presets {ISET}

<u>Entry</u>	<u>Pointer To</u>
1	constant 1
2	constant 1.0
3	Subroutine Q8QFLT
4	Subroutine Q8QFIX
5	Subroutine FL0T
6	constant 0
7	Subroutine Q8QF2F
8	Subroutine Q8QF2I
9	Subroutine Q8QI2F
10	Subroutine Q8STP
11	Subroutine Q8STPN
12	Subroutine Q8PSE
13	Subroutine Q8PSEN
14	Subroutine Q8PKUP
15	Subroutine Q8PREP
16	Subroutine Q8QFLE
17	Subroutine Q8QWND
18	Subroutine Q8QBCK
19	Subroutine Q8QINI
20	Variable Q8QX1
21	Variable Q8QX2
22	Variable Q8QX3
23	Subroutine Q8QX
24	Subroutine Q8QEND
25	Subroutine Q8DFNF
26	Subroutine Q8QY

DOCUMENT CLASS IMS PAGE NO 7-9  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

7.5 OUTPUT FORMAT, *PHHSE H*

7.5.1 Statement Format of *PHHSE H* Output Files

The first four words are the same for all statements. Word 1 contains the number of words in this entry. This is zero for the first entry of the file. Word 2 is the generated statement number. Word three is the type number of the statement. Word 4 is a switch used to signal the generated statement or statements which comprise the second half of a logical IF. The file is terminated by an END card entry.

Statements with statement labels will have a negative statement number (its absolute value being sequential in the file). Word 5 will then be the SYMTAB index to the label.

The remaining words in each entry are described below. Statement labels, variables, and arrays are represented by their SYMTAB indices.

ASSEM	parameter <sub>1</sub> , parameter <sub>2</sub> , ..., parameter n
BLOCK DATA	no additional file entries
SUBROUTINE and FUNCTION	SYMTAB pointer to SUBROUTINE name (in complemented form) SYMTAB to FUNCTION (if applicable) SYMTAB pointer to formal parameter 1 : SYMTAB pointer to formal parameter n
DATA	pointer to entry to be set value pointer value : :

FORMAT - contains the part of the FORMAT STATEMENT beginning with the first left parenthesis and ending with the last right parenthesis. This is represented in internal code, two characters per word.

REPLACEMENT	tree <sub>1</sub> tree <sub>2</sub>
-------------	--

where tree<sub>1</sub> is the tree of the variable to be stored into  
 tree<sub>2</sub> is the tree of the arithmetic expression

## CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 7-10  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005 V2.0 MACHINE SERIES 1700

STATEMENT FUNCTION	SYMTAB pointer to statement function name
	SYMTAB pointer to formal parameter 1
	.
	.
	SYMTAB pointer to formal parameter n tree
ASSIGN I to n	I SYMTAB pointer to ASSIGN variable
	n SYMTAB pointer to entry assigned tree
CALL	tree
RETURN	no additional file entries
Unconditional GO TO	SYMTAB pointer to label
COMPUTED GO TO	SYMTAB pointer to label 1
	.
	.
	SYMTAB pointer to label n tree
Assigned GO TO	variable
CONTINUE	no additional file entries
STOP	no additional file entries
STOP n	SYMTAB pointer to n
PAUSE	no additional file entries
PAUSE n	SYMTAB pointer to n
END FILE n	translated to CALL @B@FLE{N}
REWIND n	CALL @B@WND{N}
BACKSPACE n	CALL @B@BCK{N}
END	no additional file entry needed
	If END is not directly preceded by STOP or RETURN, a STOP or RETURN statement is generated before END.

CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 7-11  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005 V2.0 MACHINE SERIES 1700

BEGIN DO

SYMTAB indices for index variable,  
initial increment and final values,  
termination label

END DO

Index to LOOP END DO

ARITHMETIC IF

SYMTAB index to label<sub>1</sub>

SYMTAB index to label<sub>2</sub>

SYMTAB index to label<sub>3</sub>

tree

CONTROL DATA CORPORATION  
LA JOLLA RESOURCE CENTER DIVISION

DOCUMENT CLASS IMS PAGE NO. 7-12  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

LOGICAL IF tree

7.5.2 Statement Types

TYPE	STATEMENT NAME
0	DIMENSION
1	COMMON
2	INTEGER
3	REAL
4	INTEGER FUNCTION
5	REAL FUNCTION
6	PROGRAM
7	SINGLE
8	BYTE
9	SIGNED BYTE
10	EXTERNAL
11	RELATIVE
12	EQUIVALENCE
13	BLOCK DATA
14	FUNCTION
15	SUBROUTINE
16	DATA
17	FORMAT
18	Replacement Statement
19	Statement Function
20	ASSIGN
21	CALL
22	RETURN
23	UNUSED
24	GO TO {Unconditional}
25	GO TO {Computed}
26	GO TO {Assigned}
27	CONTINUE
28	STOP
29	STOP n
30	PAUSE
31	PAUSE n
32	END
33	ENDFILE
34	REWIND
35	BACKSPACE
36	READ {Unformatted}
37	READ {Formatted}
38	WRITE {Unformatted}
39	WRITE {Formatted}
40	BEGIN DO
41	END DO
42	Arithmetic IF

DOCUMENT CLASS IMS PAGE NO. 7-13  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. CO05\*3.1 A/B MACHINE SERIES 1700

TYPE	STATEMENT NAME
43	Logical IF
44	ASSEM
45	OPEN
46	DOUBLE PRECISION
47	DOUBLE PRECISION FUNCTION

7.5.3 ARITHMETIC TREE

7.5.3.1 General Form of Arithmetic Tree

WORD 1 Tree Indicator {-1}  
 WORD 2 No. words in expression {from WORD 4 to end}  
 WORD 3 Mode of expression {0=?, 1=integer, 2=real, 3=double}  
 WORD 4 Operator {see list}  
 WORD 5 {assuming operator takes only one word} No. Operands  
 WORD 6 WORD 6+{word 5}-1 operand pointers {base operator  
 this level}  
 WORD {NEXT} operand normal/operand inverse {0,0≠0}  
 WORD {NEXT}+1} operand {see list}

NOTE: each operand {operator} except the first is preceded by a normal/inverse indicator.

Example I+R+D

WORD 1	-1
WORD 2	14
WORD 3	3
WORD 4	11
WORD 5	3
WORD 6	5
WORD 7	8
WORD 8	11
WORD 9	0
WORD 10	24
WORD 11	D
WORD 12	0
WORD 13	24
WORD 14	R
WORD 15	0
WORD 16	24
WORD 17	I

5

B+{OPERAND POINTER} {1}operand  
 = 11

CONTROL DATA CORPORATION  
LA JOLLA RESOURCE CENTER DIVISION

DOCUMENT CLASS IMS PAGE NO. 7-14  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. COO5\*3.1 A/B MACHINE SERIES 1700

8

b+{OPERAND POINTER}<sub>{2}</sub>operand  
= 14

11

b+{OPERAND POINTER}<sub>{3}</sub>operand  
= 17

Types 22 - 24, 29, 34, 35 are two word entries

WORD 1 operand type  
WORD 2 symbol table pointer

Types 18 - 21 take two words to express type

WORD 1 operand type  
WORD 2 symbol table pointer  
WORD 3 no. operands etc.

Types 25 - 30 are of the following form

WORD 1 25, 30  
WORD 2 symbol table pointer  
WORD 3 24, 29 for subscript variable  
WORD 4 symbol table pointer

Types 26 and 31 are of the following form

WORD 1 26, 31  
WORD 2 symbol table pointer  
WORD 3 symbol table pointer for constant subscript

Types 27 and 32 are of the following form

WORD 1 27, 32  
WORD 2 symbol table pointer  
WORD 3 symbol table pointer for constant portion  
WORD 4 24, 29 for subscript variable  
WORD 5 symbol table pointer for subscript variable



DOCUMENT CLASS IMS PAGE NO. 7-15  
 PRODUCT NAME 700 MASS STORAGE FORTRAN  
 PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

Types 28 and 33 are of the following form

WORD 1	28,33
WORD 2	symbol table pointer
WORD 3	symbol table pointer for constant portion
WORD 4	operator (+ or *)
WORD 5+ff	no. operands, pointers, operands

7.5.3.2

Operand Types (Operators are Also Operands)

1. COMMA - does not appear in File 2
2. AND
3. OR
4. NOT - does not appear in
5. LT
6. GT
7. LE
8. GE
9. EQ
10. NE
11. +
12. - does not appear in File 2 - represented by inverse +
13. \*
14. / does not appear in File 2 - represented by inverse \*
15. \*\*
16. ( does not appear in File 2
17. ) does not appear in File 2
18. function
19. unused
20. subroutine
21. unused
22. function with no argument
23. subroutine with no argument
24. non-subscripted variable
25. variable only subscripted variable
26. increment only subscripted variable
27. variable and increment subscripted variable
28. complex subscripted variable
29. non-subscripted partial variable
30. variable only subscripted partial variable
31. increment only subscripted partial variable
32. variable and increment subscripted partial variable
33. complex subscripted partial variable
34. Hollerith constant
35. numeric constant
36. calling sequence label
37. material constant

DOCUMENT CLASS IMS PAGE NO. 7-16  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. 0005\*3.1 A/B MACHINE SERIES 1700

7.6.1 Mass Storage Fortran 3.1A Loader Maps

7.6.1.1 Phase A1

FTN	3E63
GOA	4515
CNVT	4576
CONV	4588
DIAG	45EE
DXP9	4686
DFLOT	47AF
DUMVOL	49EA
GETSYM	4A1A
GPUT	4A53
IOPRBA	4A7C
PACK	4CDB
Q8PRMS	4D00
STORE	4D1A
SYMBOL	4D60
SAVEID	4E21
LOCLAL	4EC6
DUMYAL	4F75
STCHAR	4FDC
GETC	500E
ENDD0	5027
GETF	512C
GNST	5472
IGETCF	564E
OPTION	5667
OUTENT	56D5
PHASEA	5709
PLABEL	5C31
Q8QBDS	5C87
RDLABL	5C87
TYPE	5D25
ENDLOC	5F52

7.6.1.2 Phase A2

FTN	3E63
GOA	4515
CNVT	4576
CONV	4588
DIAG	45EE
DXP9	4686
DFLOT	47AF
DUMVOL	49EA
GETSYM	4A1A
GPUT	4A53

CONTROL DATA CORPORATION  
LA JOLLA RESOURCE CENTER

DIVISION

DOCUMENT CLASS IMS PAGE NO. 7-17  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. COO5\*3.1 A/B MACHINE SERIES 1700

IOPRBA	4A7C
PACK	4CDB
QBPRMS	4D00
STORE	4D1A
SYMBOL	4D60
SAVEID	4E21
LOCLA2	4EC6
DUMYA2	4F78
ARITH	4F85
COMNPR	55FE
GETC	569A
GETF	56B3
DIMPR	59F9
SUBSCR	5BA7
TYPEPR	5E6C
ENDLOC	5E83

7.6.1.3 Phase A3

FTN	3E63
GOA	4515
CNVT	4576
CONV	45BB
DIAG	45EE
DXP9	4686
DFLOT	47AF
DUMVOL	49EA
GETSYM	4A1A
GPUT	4A53
IOPRBA	4A7C
PACK	4CDB
QBPRMS	4D00
STORE	4D1A
SYMBOL	4D60
SAVEID	4E21
LOCLA3	4EC6
DUMYA3	4F78
BYEQPR	4F85
CHECKF	517C
CONSUB	521F
DATAPR	52A6
OUTENT	5777
FGETC	57AB
FORK	586F
GETC	5A09
GETF	5A22
STCHAR	5D68
FREE	5D9A
ENDLOC	62B9

CONTROL DATA CORPORATION  
LA JOLLA RESOURCE CENTER DIVISION

---

DOCUMENT CLASS IMS PAGE NO. 7-18  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. CD05\*3.1 A/B MACHINE SERIES 1700

7.6.1.4 Phase A4

FTN	3E63
GOA	4515
CNVT	4576
CONV	4588
DIAG	45EE
DXP9	4686
DFLOT	47AF
DUMVOL	49EA
GETSYM	4A1A
GPUR	4A53
IOPRBA	4A7C
PACK	4CDB
Q8PRMS	4D00
STORE	4D1A
SYMBOL	4D60
SAVEID	4E21
LOCLA4	4EC6
DUMYA4	4F78
ARAYSZ	4F7F
ASGNPR	5001
BDOPR	5047
CFIVOC	5184
CKIVC	51E3
CKNAME	51F3
CPLOOP	5203
ENDDO	5280
GETC	5385
GETF	53CE
IOSPR	5714
OUTENT	5DE7
RDLABL	5E18
STCHAR	5EB9
ENDLOC	5EEB

7.6.1.5 Phase A5

FTN	3E63
GOA	4515
CNVT	4576
CONV	4588
DIAG	45EE
DXP9	4686
DFLOT	47AF
DUMVOL	49EA
GETSYM	4A1A
GPUR	4A53
IOPRBA	4A7C

CONTROL DATA CORPORATION  
LA JOLLA RESOURCE CENTER

DIVISION

DOCUMENT CLASS IMS PAGE NO. 7-19  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

PACK	4CDB
QBPRMS	4D00
STORE	4D1A
SYMBOL	4D60
SAVEID	4E21
LOCLAS	4EC6
DUMYAS	4F75
ARITH	4F82
GETC	55FC
GETF	5615
SUBSCR	5958
ENDLOC	5C20

7.6.1.6 Phase A6

FTN	3E63
GOA	4515
CNVT	4576
CONV	4588
DIAG	45EE
DXP9	4686
DFLOT	47AF
DUMVOL	49EA
GETSYM	4A1A
GPUR	4A53
IOPRBA	4A7C
PACK	4CDB
QBPRMS	4D00
STORE	4D1A
SYMBOL	4D60
SAVEID	4E21
LOCLAB	4EC6
DUMYAB	4F78
ERBPR	4F89
STCHAR	4FDC
GETC	500E
CFIVOC	5027
CKIVC	5086
GETF	5096
MODMXR	53DC
RDLABL	5B41
SUBPPR	5BDF
TREE	5C93
ENDLOC	6182

7.6.1.7 Phase A7

FTN	3E63
GOA	4515

CONTROL DATA CORPORATION  
LA JOLLA RESOURCE CENTER

DIVISION

DOCUMENT CLASS IMS PAGE NO. 7-20  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

CNVT	4576
CONV	4588
DIAG	45EE
DXP9	4686
DFLOT	47AF
DUMVOL	49EA
GETSYM	4A1A
GPUT	4A53
IOPRBA	4A7C
PACK	4CDB
QBPRMS	4D00
STORE	4D1A
SYMBOL	4D60
SAVEID	4E21
LOCLA7	4EC6
DUMYA7	4F78
ASEMPR	4F78
EXRLPR	5141
GETC	51A1
GETF	51BA
IGETCF	5500
PEQVS	5519
PRNTNM	593A
PUNT	59C9
RDLABL	5A01
SYMSCN	5A9F
ENDLOC	5ABB

7.6.1.8 Phase B1

FTN	3E63
GOB	457F
CNVT	4597
DUMMY	45DC
FCMSTK	46EF
GETSYM	478D
IOPRBB	47C6
KCPART	4997
KOUTPT	49C8
KPCSTK	49DA
KPC3PR	4F87
KSYMGN	4F9F
LABKPC	4FE7
LABLER	4FFB
PUNT	5019
QBPRMS	502F
STOREB	5049
SYMBOL	507D
TSALOC	511A

CONTROL DATA CORPORATION  
LA JOLLA RESOURCE CENTER

DIVISION

DOCUMENT CLASS IMS PAGE NO. 7-21  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

LOCLB1	51B5
DUMYB1	5248
ARAYSZ	5271
ASSEM	52F3
BANANA	535C
BGINDO	5426
END	5536
ENTCOD	5587
HELEN	5636
INXRST	5790
NOPROC	57A4
PHASEB	57D5
READIR	5C76
SUBFUN	5CCE
SYMSCN	5D35
ENDLOC	5D51

7.6.1.9 Phase B2

FTN	3E63
GOB	457F
CNVT	4597
DUMMY	45DC
FCMSTK	46EF
GETSYM	478D
IOPRBB	47C6
KCPART	4997
KOUTPT	49CB
KPCSTK	49DA
KPC3PR	4F87
KSYMGN	4F9F
LABKPC	4FE7
LABLER	4FFB
PUNT	5019
QBPRMS	502F
STOREB	5049
SYMBOL	507D
TSALOC	511A
LOCLB2	51B5
ACP	524A
AFIDL	5759
ASUPER	578A
CGOTO	5875
FINK	58D6
INTRAM	598C
PARTSB	5B98
SUBPR1	5C4F
SUBPR2	5C93
SUBPR3	5D21
ENDLOC	5D68

CONTROL DATA CORPORATION  
LA JOLLA RESOURCE CENTER DIVISION

DOCUMENT CLASS IMS PAGE NO. 7-22  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

7.6.1.10 Phase B3

FTN	3E63
GOB	457F
CNVT	4597
DUMMY	45DC
FCMSTK	46EF
GETSYM	478D
IOPRBB	47C6
KCPART	4997
KOUTPT	49C8
KPCSTK	49DA
KPC3PR	4F82
KSYMG	4F9F
LABKPC	4FE7
LABLER	4FFB
PUNT	5019
QBPRMS	502F
STOREB	5049
SYMBOL	507D
TSALOC	511A
LOCLB3	51B5
ACP	5248
ARITHR	5757
ASUPER	59C7
FINK	5A82
INTRAM	5B38
PARISB	5D47
SUBPR1	5DFB
SUBPR2	5E3F
SUBPR3	5ECD
ENDLOC	5F14

7.6.1.11 Phase C1

FTN	3E63
GOC	4A55
BKDN	4A6C
BLDUP	4ACB
BSS	4B0E
CHKWD	4B2C
CHOP	4CB6
CL12	4EE1
CON	4FD7
COUNT	5031
DATAST	5048
GETSYM	5130
INOUT	51D4
IXOPT	5243



CONTROL DATA CORPORATION  
LA JOLLA RESOURCE CENTER

DIVISION

DOCUMENT CLASS IMS PAGE NO. 7-23  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. 005\*3.1 A/B MACHINE SERIES 1700

PHASEC	5384
LABEL	5734
LABIN	5756
QXLD	57BC
REED	584C
SKIP	5880
SYMSCN	5906
IOPRBC	5922
QBPRMS	5BAD
ENDLOC	5BC7

7.6.1.12 Phase D1

FTN	3E63
GOOD	42D5
INDEX	42FA
IOPRBD	4316
NPUNCH	45C7
QBPRMS	470F
PHASE6	4729
LOCLD1	47D1
DUMYD1	4895
AMT	48A3
AMOUT	48AC
ADMAX	4EB6
BKDWN	50B9
COUNT	5122
LABOUT	5139
NP2OUT	521C
RBDX	524B
RBPK	528D
TABDEC	52B7
UNPUNC	5341
GETSYM	5357
SYMSCN	5393
ENDLOC	53BA

7.6.1.13 Phase D2

FTN	3E63
GOOD	42D5
INDEX	42FA
IOPRBD	4316
NPUNCH	45C7
QBPRMS	470F
PHASE6	4729
LOCLD2	47D1
DUMYD2	4896

CONTROL DATA CORPORATION  
LA JOLLA RESOURCE CENTER \_\_\_\_\_ DIVISION

DOCUMENT CLASS IMS PAGE NO. 7-24  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

AMT	489D
GETSYM	48A4
IACON	48D2
IHCON	492C
NWRITE	4959
PACK	4994
SYMSCN	49B9
BEGIN0	49D5
FINISH	48B9
ENDLOC	4D91

7.6.1.14 Phase E1

FTN	3E63
GOE	42D5
INDEX	42FA
IOPRBD	4316
NPUNCH	45C7
Q8PRMS	470F
PHASE6	4729
LOCLD1	47D1
DUMYD1	4895
AMT	48A3
AMOUT	48AC
ADMAX	4EBB
BKDWN	50BE
COUNT	5127
LABOUT	513E
NP2OUT	5260
RBDX	5298
RBPk	52DB
TABDEC	5305
UNPUNC	538F
CONV	53A5
GETSYM	53DE
IACON	542B
IHCON	5485
NWRITE	5481
PACK	54EC
SETPRT	5511
SYMSCN	5699
ENDLOC	56B5

7.6.1.15 Phase E2

FTN	3E63
GOE	42D5
INDEX	42FA

CONTROL DATA CORPORATION  
LA JOLLA RESOURCE CENTER DIVISION

DOCUMENT CLASS IMS PAGE NO. 7-25  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. COO5\*3.1 A/B MACHINE SERIES 1700

IOPRBD	4316
NPUNCH	45C7
QBPRMS	470F
PHASE6	4729
LOCLD2	47D1
DUMYD2	4896
AMT	489D
CONV	48A4
GETSYM	48DD
IACON	492A
IHCON	4984
NWRITE	4980
PACK	49EB
SETPRT	4A10
SYMSCN	4B98
BEGIN0	4BB4
FINISH	4D3E
ENDLOC	4EEA

7.6.2 Mass Storage Fortran 3.1B Loader Maps

7.6.2.1 Phase A1

FTN	3E63
GOA	4519
CFIVOC	457A
CKNAME	45D9
CNVT	45E9
CONV	462E
DIAG	4661
DXP9	46F9
DFLOT	4822
DUMVOL	4A5D
GETC	4ABD
GETF	4AB8
GETSYM	4E22
GPUT	4E5B
IGETCF	4E84
IOPRBA	4E9D
PACK	51AF
QBPRMS	51D4
RDLABL	51EE
STORE	528C
SYMBOL	52D2
ENDD0	5393
GNST	5498
OPTION	5674
OUTENT	56E2
PHASEA	5716

CONTROL DATA CORPORATION  
LA JOLLA RESOURCE CENTER

DIVISION

DOCUMENT CLASS IMS PAGE NO. 7-26  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

PLABEL	5C3E
STCHAR	5C94
TYPE	5CC6
SAVEID	5EF3
LOCLA1	5F98
DUMYA1	6056
QBQBD5	60BD
ENDLOC	60BD

7.6.2.2 Phase A2

FTN	3E63
GOA	4519
CFIVOC	457A
CKNAME	45D9
CNVT	45E9
CONV	462E
DIAG	4661
DXP9	46F9
DFLOT	4822
DUMVOL	4A5D
GETC	4A8D
GETF	4A88
GETSYM	4E22
GPUT	4E5B
IGETCF	4E84
IOPRBA	4E9D
PACK	51AF
QBPRMS	51D4
RDLABL	51EE
STORE	528C
SYMBOL	52D2
ENDDO	5393
GNST	5498
OPTION	5674
OUTENT	56E2
PHASEA	5716
PLABEL	5C3E
STCHAR	5C94
TYPE	5CC6
SAVEID	5EF3
LOCLA2	5F98
DUMYA2	6056
BYEQPR	60BD
CHECKF	62B4
COMNPR	6357
CONSUB	63F3
DATAPR	647A
DIMPR	694B

CONTROL DATA CORPORATION  
LA JOLLA RESOURCE CENTER DIVISION

DOCUMENT CLASS IMS PAGE NO. 7-27  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. 005\*3.1 A/B MACHINE SERIES 1700

EXRLPR	6AF9
FGETC	6B59
FORK	6C1D
PEQVS	6DB7
PRNTNM	71E5
SUBPPR	7274
SYMSCN	731C
TYPEPR	7338
ENDLOC	734F

7.6.2.3 Phase A3

FTN	3E63
GOA	4519
CFIVOC	457A
CKNAME	45D9
CNVT	45E9
CONV	462E
DIAG	4661
DXP9	46F9
DFLCT	4822
DUMVOL	4A5D
GETC	4A8D
GETF	4AB8
GETSYM	4E22
GPUT	4E5B
IGETCF	4E84
IOPRBA	4E9D
PACK	51AF
QBPRMS	51D4
RDLABL	51EE
STORE	528C
SYMBOL	52D2
ENDDO	5393
GNST	5498
OPTION	5674
OUTENT	56E2
PHASEA	5716
PLABEL	5C3E
STCHAR	5C94
TYPE	5CC6
SAVEID	5EF3
LOCLAE	5F98
DUMYAE	6056
ARAYSZ	608D
ASEMPR	613F
ASGNPR	6308
BDOPR	634E
CHECKF	6488

CONTROL DATA CORPORATION  
LA JOLLA RESOURCE CENTER DIVISION

DOCUMENT CLASS IMS PAGE NO. 7-28  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

CKIVC	652E
CONSUB	653E
CPL00P	65C5
FGETC	6672
F0RK	6736
ERBPR	68D0
MODMXR	6923
PUNT	7088
ENDLOC	70C0

7.6.2.4 Phase A4

FTN	3E63
GOA	4519
CFIVOC	457A
CKNAME	45D9
CNVT	45E9
CONV	462E
DIAG	4661
DXP9	46F9
DFLOT	4822
DUMVOL	4A5D
GETC	4A8D
GETF	4AB8
GETSYM	4E22
GPUT	4E5B
IGETCF	4E84
IOPRBA	4E9D
PACK	51AF
Q8PRMS	51D4
RDLABL	51EE
STORE	528C
SYMBOL	52D2
ENDD0	5393
GNST	5498
OPTION	5674
OUTENT	56E2
PHASEA	5716
PLABEL	5C3E
STCHAR	5C94
TYPE	5CC6
SAVEID	5EF3
LOCLA4	5F98
DUMYA4	6056
ARITH	608D
SUBSCR	6769
TREE	6A2E
ENDLOC	6F43

CONTROL DATA CORPORATION  
LA JOLLA RESOURCE CENTER

DIVISION

DOCUMENT CLASS IMS PAGE NO. 7-29  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

7.6.2.5 Phase A5

FTN	3E63
GOA	4519
CFIVOC	457A
CKNAME	45D9
CNVT	45E9
CONV	462E
DIAG	4661
DXP9	46F9
DFLCT	4822
DUMVOL	4A5D
GETC	4A8D
GETF	4AB8
GETSYM	4E22
GPUT	4E5B
IGETCF	4E84
IOPRBA	4E9D
PACK	51AF
QBPRMS	51D4
RDLABL	51EE
STORE	528C
SYMBOL	52D2
ENDD0	5393
GNST	5498
OPTION	5674
OUTENT	56E2
PHASEA	5716
PLABEL	5C3E
STCHAR	5C94
TYPE	5CC6
SAVEID	5EF3
LOCLAS	5F98
DUMYA5	6056
BDOPR	60BD
CKIVC	61FA
IOSPR	620A
ENDLOC	68F1

7.6.2.6 Phase B1

FTN	3E63
GOB	4583
CNVT	4599

CONTROL DATA CORPORATION  
LA JOLLA RESOURCE CENTER DIVISION

DOCUMENT CLASS IMS PAGE NO. 7-30  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

DUMMY	45DE
FCMSTK	56F1
GETSYM	478F
IOPRBB	47C8
KCPART	4D52
KOUTPT	4D83
KPCSTK	4D95
KPC3PR	5342
KSYMGN	535A
LABKPC	53A2
LABLER	53B6
PUNT	53D4
QBPRMS	53EA
STOREB	5404
SYMBOL	5438
TSALOC	54D5
ARAYSZ	5570
ASSEM	55F2
BANANA	565B
BGINDO	5725
END	5835
ENTCOD	5886
HELEN	5935
INXRST	5A8F
NOPROC	5AA3
PHASEB	5AD4
READIR	5F76
SUBFUN	5FCE
SYMSCN	6035
ACP	6051
AFIDL	6560
ASUPER	66C1
CGOTO	667C
FINK	66DD
INTRAM	6793
PARTSB	69A2
SUBPR1	6A56
SUBPR2	6A9A
SUBPR3	6B28
ARITHR	6B6F
ENDLOC	6DDF

7.6.2.7 Phase C1

FTN	3E63
GOC	4A59
BKDWN	4A77
BLDUP	4AD6
BSS	4B19



CONTROL DATA CORPORATION  
LA JOLLA RESOURCE CENTER DIVISION

DOCUMENT CLASS IMS PAGE NO. 7-31  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. CO05\*3.1 A/B MACHINE SERIES 1700

CHKWD	4B37
CHOP	4CC1
CL12	4EEC
CON	4FE2
COUNT	503C
DATAS	5053
GETSYM	513B
INOUT	51DF
IOPRBC	524E
IXOPT	60FC
LABEL	623D
LABIN	625F
PHASEC	62C5
QBPRMS	6626
QXLD	6640
REED	66D0
SKIP	6734
SYMSCN	678A
ENDLOC	67A6

7.6.2.8 Phase D1

FTN	3E63
GOOD	4A59
AMOUT	4A7C
ADMAX	507A
GEGINO	527D
BKDWN	53C5
COUNT	542E
FINISH	5445
GETSYM	55F1
IACON	5695
IHCON	56EF
INDEX	571C
IOPRBD	5738
LABOUT	5CE6
NP2OUT	5DC9
NPUNCH	5DF8
NWRITE	5F40
PACK	5F7B
PHASE6	5FA0
QBPRMS	6043
RBDX	605D
RBPK	609F
SYMSCN	60C9
TABDEC	60E5
UNPUNC	616F
ENDLOC	6185

CONTROL DATA CORPORATION  
LA JOLLA RESOURCE CENTER DIVISION

---

DOCUMENT CLASS IMS PAGE NO. 7-31A  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. COO5\*3.1 A/B MACHINE SERIES 1700

7.b.2.9 Phase E1

FTN	3E63
GOE	4A59
AMOUT	4A7C
ADMAX	508B
BEGIN0	528E
BKDN	540A
CONV	5473
COUNT	54AC
FINISH	54C3
GETSYM	566F
IACON	5713
IHCON	576D
INDEX	5799
IOPRBD	57B5
LABOUT	5D63
NP2OUT	5E85
NPUNCH	5EBD
NWRITE	6005
PACK	6040
PHASE6	6065
QBPRMS	6108
RBDX	6122
RBPK	6165
SETPRT	618F
SYMSCN	6317
TABDEC	6333
UNPUNC	638D
ENDLOC	63D3

## CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 7-32  
 PRODUCT NAME 1700 Mass Storage FORTRAN  
 PRODUCT MODEL NO. CD05#3.0-ED06#4.0 MACHINE SERIES 1700

LOCL#3	5C29	FORTRAN	3.0B	SUMMARY-REL	20F
DUMY#3	5CE7	FORTRAN	3.0B	SUMMARY-REL	21F
ARAYSZ	5D4E	FORTRAN	3.0B	SUMMARY-REL	42A
ASEMPK	5D69	FORTRAN	3.0B	SUMMARY-REL	40A
ASGNPK	5F6H	FORTRAN	3.0B	SUMMARY-REL	32A
BDOPK	5FB1	FORTRAN	3.0B	SUMMARY-REL	33A
CHECKF	60EB	FORTRAN	3.0B	SUMMARY-REL	29A
CKIVC	618E	FORTRAN	3.0B	SUMMARY-REL	35A
CONSD	619E	FORTRAN	3.0B	SUMMARY-REL	30A
CPLOOP	6225	FORTRAN	3.0B	SUMMARY-REL	43A
DATAPK	62CB	FORTRAN	3.0B	SUMMARY-REL	31A
FGETC	645B	FORTRAN	3.0B	SUMMARY-REL	21A
FORK	64EB	FORTRAN	3.0B	SUMMARY-REL	22A
ERBPK	668R	FORTRAN	3.0B	SUMMARY-REL	38A
MODMAP	66DB	FORTRAN	3.0B	SUMMARY-REL	39A
PUNT	6B3C	FORTRAN	3.0B	SUMMARY-REL	27A
ENDLUC	6B75	FORTRAN	3.0B	SUMMARY-REL	17F

IN

\*K, IO

IN

\*N, FURIA3, 99B

IN

\*K, IO

IN

\*P

FTN	3DC4	COPYRIGHT CONTROL DATA CORP. 1972			
GUA	447A	FORTRAN	3.0B	SUMMARY-REL	02F
CFIVUC	44UR	FORTRAN	3.0B	SUMMARY-REL	34A
CKNAME	4539	FORTRAN	3.0B	SUMMARY-REL	36A
CNVT	4549	FORTRAN	3.0B	SUMMARY-REL	01A
CONV	4587	FORTRAN	3.0B	SUMMARY-REL	03F

DIAG	45BA	FORTRAN	3.0B	SUMMARY-REL	04F
EXP9	464A	FORTRAN	3.0B	SUMMARY-REL	05F
FLOAI	46F3	FORTRAN	3.0B	SUMMARY-REL	06F
GETC	483D	FORTRAN	3.0B	SUMMARY-REL	14F
GETF	480H	FORTRAN	3.0B	SUMMARY-REL	04A
GETSYM	4B03	FORTRAN	3.0B	SUMMARY-REL	07F
GPOT	4B9C	FORTRAN	3.0B	SUMMARY-REL	02A
IGETCP	4BC5	FORTRAN	3.0B	SUMMARY-REL	15F
IOPRBA	4BDE	FORTRAN	3.0B	SUMMARY-REL	08F
PACK	4EE6	FORTRAN	3.0B	SUMMARY-REL	09F
QBFRMS	4F0R	FORTRAN	3.0B	SUMMARY-REL	10F



## CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS TMS PAGE NO. 7.33  
 PRODUCT NAME 1700 Mass Storage FORTRAN  
 PRODUCT MODEL NO. C005\*3.0-E006\*4.0 MACHINE SERIES 1700

RDLABL	4F25	FORTAN	3.0B	SUMMARY-REL	10A
STORE	4FC3	FORTAN	3.0B	SUMMARY-REL	11F
SYMBOL	4FF7	FORTAN	3.0B	SUMMARY-REL	03A
ENDDU	5099	FORTAN	3.0B	SUMMARY-REL	29A
GNST	51A0	FORTAN	3.0B	SUMMARY-REL	05A
OPTION	5301	FORTAN	3.0B	SUMMARY-REL	16F
QUIENT	53CF	FORTAN	3.0B	SUMMARY-REL	06A
PHASEA	5403	FORTAN	3.0B	SUMMARY-REL	07A
PLABEL	58FA	FORTAN	3.0B	SUMMARY-REL	08A
STCHAR	594F	FORTAN	3.0B	SUMMARY-REL	11A
TYPE	5980	FORTAN	3.0B	SUMMARY-REL	12A
SAVEID	5B83	FORTAN	3.0B	SUMMARY-REL	13A
LOCIA4	5C29	FORTAN	3.0B	SUMMARY-REL	22F
DUMYA4	5CE7	FORTAN	3.0B	SUMMARY-REL	23F
ARITH	5D4E	FORTAN	3.0B	SUMMARY-REL	14A
SUBSCH	63U3	FORTAN	3.0B	SUMMARY-REL	17A
TREE	6690	FORTAN	3.0B	SUMMARY-REL	41A
ENDLUC	6B95	FORTAN	3.0B	SUMMARY-REL	17F

IN

\*K, I<sub>0</sub>  
IN\*N, FORIA499, B  
IN\*K, I<sub>0</sub>  
IN

\*P

FTN	3DC4	COPYRIGHT CONTROL DATA CORP. 1972			
GOA	447A	FORTAN	3.0B	SUMMARY-REL	02F
CFIVUC	440A	FORTAN	3.0B	SUMMARY-REL	34A
CKNAME	4539	FORTAN	3.0B	SUMMARY-REL	36A
CNVT	4549	FORTAN	3.0B	SUMMARY-REL	01A
CONV	4587	FORTAN	3.0B	SUMMARY-REL	03F
DIAG	45BA	FORTAN	3.0B	SUMMARY-REL	04F
EXP9	464A	FORTAN	3.0B	SUMMARY-REL	05F
FLOAT	46F3	FORTAN	3.0B	SUMMARY-REL	06F
GETC	4830	FORTAN	3.0B	SUMMARY-REL	14F
GETF	4868	FORTAN	3.0B	SUMMARY-REL	04A
GETSYM	4893	FORTAN	3.0B	SUMMARY-REL	07F
GPOT	4B9C	FORTAN	3.0B	SUMMARY-REL	02A
IGETCH	4BC5	FORTAN	3.0B	SUMMARY-REL	15F
IOFRBA	4BDE	FORTAN	3.0B	SUMMARY-REL	08F
PACK	4EE6	FORTAN	3.0B	SUMMARY-REL	09F
QBERMS	4F0F	FORTAN	3.0B	SUMMARY-REL	10F
RDLABL	4F25	FORTAN	3.0B	SUMMARY-REL	10A
STORE	4FC3	FORTAN	3.0B	SUMMARY-REL	11F

## CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 7.34  
 PRODUCT NAME 1700 Mass Storage FORTRAN  
 PRODUCT MODEL NO. C005\*3.0-ED06\*4.0 MACHINE SERIES 1700

SYMBOL	4FF7	FORTRAN	3.0B	SUMMARY-REL	03A
ENDDU	5099	FORTRAN	3.0B	SUMMARY-REL	29A
GNST	51A0	FORTRAN	3.0B	SUMMARY-REL	05A
OPTION	5301	FORTRAN	3.0B	SUMMARY-REL	16F
OUTENT	53CF	FORTRAN	3.0B	SUMMARY-REL	06A
PHASEA	5403	FORTRAN	3.0B	SUMMARY-REL	07A
PLABEL	58FB	FORTRAN	3.0B	SUMMARY-REL	08A
STCHAR	594E	FORTRAN	3.0B	SUMMARY-REL	11A
TYPE	5980	FORTRAN	3.0B	SUMMARY-REL	12A
SAVEID	5B03	FORTRAN	3.0B	SUMMARY-REL	13A
LOCAS	5C29	FORTRAN	3.0B	SUMMARY-REL	24F
DUMYAS	5CE7	FORTRAN	3.0B	SUMMARY-REL	25F
BDOPR	5D4E	FORTRAN	3.0B	SUMMARY-REL	33A
CKIVC	5E83	FORTRAN	3.0B	SUMMARY-REL	35A
IUSPR	5E92	FORTRAN	3.0B	SUMMARY-REL	37A
ENDLOC	6545	FORTRAN	3.0B	SUMMARY-REL	17F

IN

\*K,I6

IN

\*N,FORTIAS,,,B

IN

\*K,I6

IN

\*P

FTN	3DC4	COPYRIGHT CONTROL DATA CORP. 1972			
G0B	44D4	FORTRAN	3.0B	SUMMARY-REL	26F
CNVT	44EA	FORTRAN	3.0B	SUMMARY-REL	01A
DUMMY	4528	FORTRAN	3.0B	SUMMARY-REL	01B
FCMSIK	4639	FORTRAN	3.0B	SUMMARY-REL	02B
GETSYM	46C2	FORTRAN	3.0B	SUMMARY-REL	07F
IOPR0B	46FH	FORTRAN	3.0B	SUMMARY-REL	27F
KCPART	4C85	FORTRAN	3.0B	SUMMARY-REL	03B
KOUTPT	4C86	FORTRAN	3.0B	SUMMARY-REL	04B
KPCSTK	4CC8	FORTRAN	3.0B	SUMMARY-REL	05B
KPC3P	5099	FORTRAN	3.0B	SUMMARY-REL	06B
KSYMGN	50B1	FORTRAN	3.0B	SUMMARY-REL	07B
LABKFC	50F9	FORTRAN	3.0B	SUMMARY-REL	08B
LABLER	510D	FORTRAN	3.0B	SUMMARY-REL	09B
PUNT	512R	FORTRAN	3.0B	SUMMARY-REL	10B
QBPRMS	5141	FORTRAN	3.0B	SUMMARY-REL	10F
STORE	515B	FORTRAN	3.0B	SUMMARY-REL	11F
SYMBOL	518F	FORTRAN	3.0B	SUMMARY-REL	11B
TSAUC	522C	FORTRAN	3.0B	SUMMARY-REL	12B
ARAY3Z	52B7	FORTRAN	3.0B	SUMMARY-REL	42A

## CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS TMS PAGE NO. 7-35  
 PRODUCT NAME 1700 Mass Storage FORTRAN  
 PRODUCT MODEL NO. COO5\*3.0-EOO6\*4.0 MACHINE SERIES 1700

ASSEM	5322	FORTRAN	3.0B	SUMMARY-REL	13B
BANANA	53B9	FORTRAN	3.0B	SUMMARY-REL	14B
BGINDO	544D	FORTRAN	3.0B	SUMMARY-REL	15B
END	5536	FORTRAN	3.0B	SUMMARY-REL	16B
ENTCUD	559E	FORTRAN	3.0B	SUMMARY-REL	17B
HELEN	5648	FORTRAN	3.0B	SUMMARY-REL	18B
INARST	57A1	FORTRAN	3.0B	SUMMARY-REL	19B
NOFROC	57B5	FORTRAN	3.0B	SUMMARY-REL	20B
PHASEB	57E6	FORTRAN	3.0B	SUMMARY-REL	21B
READIR	5C3C	FORTRAN	3.0B	SUMMARY-REL	22B
SUBFUN	5C74	FORTRAN	3.0B	SUMMARY-REL	23B
SYMSUN	5CFC	FORTRAN	3.0B	SUMMARY-REL	28A

ACF	5D19	FORTRAN	3.0B	SUMMARY-REL	24B
AFIDL	6104	FORTRAN	3.0B	SUMMARY-REL	25B
ASUPER	61BE	FORTRAN	3.0B	SUMMARY-REL	26B
CGUTU	6275	FORTRAN	3.0B	SUMMARY-REL	27B
FINK	62D0	FORTRAN	3.0B	SUMMARY-REL	28B
INTRAM	6366	FORTRAN	3.0B	SUMMARY-REL	29B
PARTSB	655F	FORTRAN	3.0B	SUMMARY-REL	30B
SUBPR1	660F	FORTRAN	3.0B	SUMMARY-REL	31B
SUBPR2	664D	FORTRAN	3.0B	SUMMARY-REL	32B
SUBPR3	66DA	FORTRAN	3.0B	SUMMARY-REL	33B
ARITHR	6721	FORTRAN	3.0B	SUMMARY-REL	34B
ENDLUC	68E5	FORTRAN	3.0B	SUMMARY-REL	17F

IN

\*K, 10

IN

\*N, FURIBL, 9, 8

IN

\*K, 10

IN

\*p

FTN	30C4	COPYRIGHT CONTROL DATA CORP. 1972			
GOC	49BA	FORTRAN	3.0B	SUMMARY-REL	28F
BKDOWN	490B	FORTRAN	3.0B	SUMMARY-REL	01C
BLDUP	4A39	FORTRAN	3.0B	SUMMARY-REL	02C
BSS	4A7C	FORTRAN	3.0B	SUMMARY-REL	03C
CHKWD	4A9A	FORTRAN	3.0B	SUMMARY-REL	04C
CHUP	4C1A	FORTRAN	3.0B	SUMMARY-REL	05C
CL12	4E3B	FORTRAN	3.0B	SUMMARY-REL	06C
CON	4F2F	FORTRAN	3.0B	SUMMARY-REL	07C
COUNI	4F6A	FORTRAN	3.0B	SUMMARY-REL	08C

CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS TMS PAGE NO. 7-36  
 PRODUCT NAME 1700 Mass Storage FORTRAN  
 PRODUCT MODEL NO. C005\*3.0-E006\*4.0 MACHINE SERIES 1700

DATAST	4F7E	FORTRAN	3.0B	SUMMARY-REL	09C
GEISYM	504D	FORTRAN	3.0B	SUMMARY-REL	10C
INOUT	50F1	FORTRAN	3.0B	SUMMARY-REL	11C
IOPRBC	5161	FORTRAN	3.0B	SUMMARY-REL	29F
IXUPI	600F	FORTRAN	3.0B	SUMMARY-REL	12C
LABEL	614B	FORTRAN	3.0B	SUMMARY-REL	14C
LABIN	616D	FORTRAN	3.0B	SUMMARY-REL	15C
PHASEC	61D4	FORTRAN	3.0B	SUMMARY-REL	13C
QBFRMS	6529	FORTRAN	3.0B	SUMMARY-REL	10F
QXCD	6543	FORTRAN	3.0B	SUMMARY-REL	16C
REED	65U3	FORTRAN	3.0B	SUMMARY-REL	17C
SKIP	6630	FORTRAN	3.0B	SUMMARY-REL	18C
SYMSUN	6686	FORTRAN	3.0B	SUMMARY-REL	19C
ENDLOC	66A3	FORTRAN	3.0B	SUMMARY-REL	17F

IN

\*K, I8

IN

\*N, FORIC1, ., B

IN

\*K, I0

IN

\*P

FTN	3DC4	COPYRIGHT CONTROL DATA CORP. 1972			
GOOD	49BA	FORTRAN	3.0B	SUMMARY-REL	30F
AMOUT	49DD	FORTRAN	3.0B	SUMMARY-REL	01D
ADMAX	4FC5	FORTRAN	3.0B	SUMMARY-REL	02D
BEGIN0	51C7	FORTRAN	3.0B	SUMMARY-REL	03D
RKDNW	530F	FORTRAN	3.0B	SUMMARY-REL	04D
COUNT	537A	FORTRAN	3.0B	SUMMARY-REL	05D
FINISH	5392	FORTRAN	3.0B	SUMMARY-REL	06D
GEISYM	553B	FORTRAN	3.0B	SUMMARY-REL	10C
IACON	55UF	FORTRAN	3.0B	SUMMARY-REL	07D
INCON	563B	FORTRAN	3.0B	SUMMARY-REL	08D
INDEX	5666	FORTRAN	3.0B	SUMMARY-REL	09D
IOPRBD	5683	FORTRAN	3.0B	SUMMARY-REL	31F
LABOUT	5C31	FORTRAN	3.0B	SUMMARY-REL	10D
NPZOUT	5D11	FORTRAN	3.0B	SUMMARY-REL	11D
NPUNCH	5D4D	FORTRAN	3.0B	SUMMARY-REL	12D
NWRITE	5E81	FORTRAN	3.0B	SUMMARY-REL	13D
PACK	5E8C	FORTRAN	3.0B	SUMMARY-REL	09F
PHASE6	5EE1	FORTRAN	3.0B	SUMMARY-REL	14D
QBFRMS	5F7E	FORTRAN	3.0B	SUMMARY-REL	10F



DOCUMENT CLASS IMS PAGE NO. 7-37  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. COO5\*3.1, A/B MACHINE SERIES 1700

7.7.4 Phase 4 Labeled Common Block

1. INDTAB            preset table of constants. Those indicator words {see output description} which are used several times are preset in this table for easy reference.
2. INFTB            Intrinsic function table. A preset loopup table for determining whether a function is intrinsic {see HELEN}.
3. NBSS, NADC       constants preset to the instruction code values: i.e., NBSS = 1, NINA = 31, NTRA@ = b2, etc.
4. INFTBL           INFTB table entry length.
5. INFTBN           INFTBN table length.
6. INFTBX           index to INFTB table {not preset}.
7. KODNAM           part of INFTB entry containing function name as coded.
  
9. KFTYPE           part of INFTB entry containing function type:  
                      1 = integer  
                      2 = real  
                      3 = double precision
10. NLINE           part of INFTB entry containing in-line/non-in-line flag.
11. NPMTRS           part of INFTB entry containing number of parameters of function.
12. KSPTAB           preset table of special pointers to sybmol table; e.g., 0 constant, 1.0 constant, and pointers to implied subroutines.

CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS TMS PAGE NO. 7-38  
PRODUCT NAME 1700 Mass Storage FORTRAN  
PRODUCT MODEL NO. C005\*3.0-E006\*4.0 MACHINE SERIES 1700

ENDLOC 02E4 FORTRAN 3.0B SUMMARY-REL 17F  
IN  
\*K,10  
IN  
\*N,FURIE1,9,9B  
IN  
\*K,10  
IN  
\*L,FIN  
IN  
\*K,PS  
IN  
\*V,11  
IN

DOCUMENT CLASS IMS PAGE NO. 7-39  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT NO. CD05 VERSION 2.0 MACHINE SERIES 1700

15. LTYP working cell of BGINDO and BANANA.
16. MBEGIN working cell of PHASE 4 holds label of first address of statement function
17. IADDIT parameter to INTRAM (additive)
19. INTRAX INTRAS table index
20. IPTR parameter to INTRAM (operand pointer)
21. ISBSCP parameter to INTRAM (subscript pointer)
22. JXX index to input buffer set by READIR
23. KBEGIN working cell of PHASE 4 holds label of first address in program
24. KBIAS hold label of cell containing program base (PHASE4)
25. KBYTX byte index for constructing floating point calls (FCMSTK)
26. KENTER working cell of PHASE 4
27. KENTRY working cell of PHASE 4
28. KEXEC working cell of PHASE 4
29. KEXTYP set by ACP as flag to give expression type of tree just processed
30. KFCETX used for KPCSTK
31. KFCSW floating calling sequence switch (FCMSTK)
32. KFFSAV contains label of cell used to save index FF
33. KFINDX index used in building floating calling sequences (FCMSTK)
34. KFLAM current floating addressing mode (FCMSTK)
35. KINTYP instruction type (parameter to INTRAM)

CONTROL DATA CORPORATION  
LA JOLLA RESOURCE CENTER \_\_\_\_\_ DIVISION

DOCUMENT CLASS IMS PAGE NO. 7-40  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. COO5\*3.1 A/B MACHINE SERIES 1700

- 36. KLLTBX            KLLTB table index
- 37. KNTR0L           control parameter to INTRAM
- 38. K0BX             output buffer index
- 39. KPRNAM           program name. Working cell of PHASE 4
- 40. K0SAV            contains label of cell used to save  
                         @ register
- 41. KRETRN           contains label of first cell of  
                         RETURN code sequence
- 42. KRTNS            set in PHASEB
  
- 45. KSFNAM           statement function name holder
  
- 47. KSTYP            set in PHASEB
- 48. KTCATX           KTCAT index
- 49. LEVEL            index to level tables for ACP
- 50. LIFTX            index to level tables for LOGLIF
- 51. LOGIF            logical if switch {LOGLIF}
- 52. IDMM             Holder for local variables  
                         equivalenced to common
- 53. IDPFLG           Indicates location of 'current' dp  
                         value.
  - 0 - dp value is in dp pseudo-  
                         accumulator
  - 1 - dp value is in locations  
                         C5-C7

DOCUMENT CLASS IMS PAGE NO 7-41  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

## 7.7.6 Phase C, D, E Labelled Common Block

NPCBF binary output buffer  
 IDVTAB table of pointers to places in program where addresses of variables may be found  
 NPTBF print image  
 IXS1  
 ISX2 index history of program being compiled

## 7.7.7 Phase C and D/E Blank Common Block (some array dimensions differ in phase D/E)

INBUFF input buffer  
 NOBUFF intermediate output buffer  
 NRCDD intermediate record holder  
 NFTAB future table  
 IV table of temporary holders  
 NSUMD maximum length of IDVTAB  
 NDFL modify, don't modify addressing switch  
 NADX IDVTAB index  
 NSUMVS maximum size of future table  
 IXC current index  
 IXFLAG effect of current command on index  
 IXF current contents of \$FF  
 IXQ current contents of Q  
 IXXS index history tables index  
 IUSE \$FF has been used switch  
 IFIX point to which addressing is fixed  
 NOUTOT number of words in intermediate output buffer (circular in D/E)  
 NOWO number of words in current intermediate output record (-2)  
 NL ignore record switch  
 NT record is label switch  
 NA record has additive switch  
 NS record has subscript switch  
 ND don't modify addressing switch  
 NR 2 word instruction switch  
 NQ command uses Q register switch  
 NF ADC type indicator  
 NTYPE command type - second phase  
 NOWI number of words in input record (-2)  
 NOPC opcode  
 NOPT address pointer  
 NADD additive pointer  
 NXPT variable subscript pointer  
 NATYPE command type - first phase  
 ICT number of words command occupies  
 INCT input buffer index; in phase D/E circular buffer index  
 NOCT output buffer index; in phase D/E, next available circular buffer location

DOCUMENT CLASS IMS PAGE NO. 7-42  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

NSTYPE	first executable instruction encountered switch
LRS	long right shift 16 needed indicator
NOBW	no. words in binary output, this record
NBINC	current index into binary output image
NPC1	
NSETX	top of list output buffer
NRM	relocation of current command
NPTBFX	list output buffer
NRC1	
NRC2	relocation - words 1, 2 of current command
NW2FL	processing second half of 2 word command
NW2	second binary word of 2 word command
NBFL	last word flag - binary output
NXCON	constant value - used by SETPRT
NASCI	pointer for address field - used by SETPRT
ISCOUN	address counter - second phase
ICOUNT	address counter - both phases
NOFS	future table index
NODS	NDSTAB index
NDXJ	temporary holder for ICOUNT
NX	INBUFF, NOBUFF index
IHEAD	'top' of circular buffer NOBUFF
INOB	size of NOBUFF
INC	NOBUFF subscript (not 'circular')
NDSTAB	table of references to dummy parameters
IXS3	Index history of program being compiled

CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 7-43  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005 V.2.0 MACHINE SERIES 1700

7.7.8

Compressed Symbol Table Blank Common Block  
(phase D 2.0A only)

ICSYFL	Length of compressed symbol table entry
ICSYNS	Number of sectors needed for compressed symbol table page
ICSYMP	Compressed symbol table page indicator
ICSYPC	Current compressed symbol table page
ICSYPS	Compressed Symbol table page size
CSYMTB	Compressed Symbol Table

C

C

C



DOCUMENT CLASS \_\_\_\_\_ IMS \_\_\_\_\_ PAGE NO 8-1  
 PRODUCT NAME 17 Mass Storage FORTRAN  
 PRODUCT NO. C008 VERSION 2.0 MACHINE SERIES 1700

## CHAPTER 8

## TABLE OF CONTENTS

- 8.0 Object Time Arithmetic Routines
  - 8.1 Introduction
  - 8.2 General Description
    - 8.2.1 Temporary Storage
    - 8.2.2 Calling Sequence
    - 8.2.3 Returning Sequence
  - 8.3 Function Description
    - 8.3.1 Trigonometric Sine (SIN)
      - 8.3.1.1 Range Reduction
      - 8.3.1.2 Series for SIN(X), with  $X \leq \pi/2$
    - 8.3.2 Trigonometric Cosine (COS)
      - 8.3.2.1 Temporary Storage for SIN and COS
    - 8.3.3 Hyperbolic Tangent (TANH)
      - 8.3.3.1 If  $0 < X < \frac{1}{2}$
      - 8.3.3.2 Temporary Storage for TANH
    - 8.3.4 Arctangent (ATAN)
      - 8.3.4.1 Range Reduction
      - 8.3.4.2 Series for Arctan(X),  $X \leq \tan(\pi/16)$
      - 8.3.4.3 Temporary Storage for ATAN
    - 8.3.5 Exponential (EXP)
      - 8.3.5.1 Range Reduction
      - 8.3.5.2 Series for  $e^x$  for  $X \leq \ln/2$
      - 8.3.5.3 Temporary Storage for EXP
    - 8.3.6 Natural Logarithm (ALOG)
      - 8.3.6.1 Range Reduction
      - 8.3.6.2 Series for  $\ln\left(\frac{1+X}{1-X}\right)$  for  $|X| \leq 3 - 2\sqrt{2}$
      - 8.3.6.3 Temporary Storage for ALOG
    - 8.3.7 Square Root (SQRT)
      - 8.3.7.1 Range Reduction
      - 8.3.7.2 Iterative Method
      - 8.3.7.3 Temporary Storage for SQRT
  - 8.4 Type Out Message
    - 8.4.1 Routine Argument Answer
  - 8.5 References
  - 8.6 Exponentiation - Subroutine QBEXPN

DOCUMENT CLASS IMS PAGE NO. 8-2  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005 V2.0 MACHINE SERIES 1700

## 8.0 OBJECT TIME ARITHMETIC ROUTINES

### 8.1 Introduction

This chapter deals with the following object time routines:

<u>External Function</u>	<u>Symbolic Name</u>
Trigonometric sine	SIN
Trigonometric cosine	COS
Hyperbolic tangent	TANH
Exponential	EXP
Natural logarithm	ALOG
Arctangent	ATAN
Square Root	SQRT

The above routines are designed to operate under the 1700 Mass Storage Fortran. They are written in 1700 Assembly Language for optimization of code and time of execution. However, they can be used by any other program working under the 1700 Operating System as long as the proper calling sequence is generated, and cells #C5 and #E5 are reserved for the floating point package and temporary storage.

### 8.2 General Description

The argument to any of the functions is a real value number written in the 1700 Mass Storage Fortran floating point notation. The result is also a real value number in floating point notation.

#### 8.2.1 Temporary Storage

These routines use the unprotected communications area from #D8 to #E2 as temporary storage area.

#### 8.2.2 Calling Sequence

Each object time routine is entered via an RTJ instruction to its symbolic name, followed by a cell containing the address of core where the value of the argument can be found in floating point notation.

DOCUMENT CLASS IMS PAGE NO. 8-3  
 PRODUCT NAME 1700 Mass Storage FORTRAN  
 PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

### 8.2.3 Returning Sequence

Each object time routine will return control to the user at the program location immediately following the parameter address with the calculated value of the function in the floating point pseudo-accumulator.

### 8.3 Function Description

The mathematical method used to approximate the functions will be found in Appendix A of this document. Basically it is a truncated Taylor-Mclaurin series with Chebyshev expansion for the higher terms of the series.

#### 8.3.1 Trigonometric Sine (SIN)

##### 8.3.1.1 Range Reduction

Let Z be the value of the argument.

If  $|Z| > 2^{21}$  then the answer will be assumed zero.

If:

$Z < 0$  let  $U = -Z$  and  $\sin(Z) = -\sin(U)$   
 $Z \geq 0$  let  $U = Z$  and  $\sin(Z) = \sin(U)$

Let I be integral part of  $U/2\pi$  and let

$Y = U - I \cdot 2\pi$ , then  $Y \leq 2\pi$ , and  $\sin(U) = \sin(Y)$

If:

$Y > \pi$ , let  $T = Y - \pi$  then  $\sin(Y) = -\sin(T)$   
 $Y \leq \pi$ , let  $T = Y$  then  $\sin(Y) = \sin(T)$

If:

$T > \pi/2$ , let  $X = \pi - T$   
 $T \leq \pi/2$ , let  $X = T$   
 and  $\sin(T) = \sin(X)$ , with  $X \leq \pi/2$

##### 8.3.1.2 Series for SIN(X) with $X \leq \pi/2$

$$\text{SIN}(X) = C_0 X + C_1 X^3 + C_2 X^5 + C_3 X^7 + C_4 X^9$$

with  $C_0 = 1.0$   
 $C_1 = -1.66666647$   
 $C_2 = .83328836E-3$   
 $C_3 = -1.9799327E-3$   
 $C_4 = -2.5857445E-6$

DOCUMENT CLASS TMS PAGE NO. 8-4  
 PRODUCT NAME 1700 Mass Storage FORTRAN  
 PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

## 8.3.2 Trigonometric Cosine (COS)

Since  $\text{COS}(Z) = \text{SIN}(Z + \pi/2)$  then object time routine COS will simply be a second entry to the SIN routine.

## 8.3.2.1 Temporary Storage for SIN and COS

Eight cells of temporary storage are required by either SIN and COS. They are:

PARADD	for parameter address \$DE
RETADD	for return address \$D9
FLAG	for sign indicator \$D8
X and X <sub>2</sub>	for temporary floating point numbers, \$DA to \$DD
QS	for saving Q #E2

## 8.3.3 Hyperbolic Tangent (TANH)

8.3.3.1 If  $0 < |X| < \frac{1}{2}$ ,

$$\text{TANH}(X) = X + A_1 X^{**3} + A_2 X^{**5} + A_3 X^{**7} + A_4 X^{**9}$$

where

$$A_1 = -0.33333227$$

$$A_2 = 0.13337246$$

$$A_3 = -0.053388615$$

$$A_4 = 0.025628253$$

If  $\frac{1}{2} \leq |X| < 10$

$$\text{TANH}(X) = 1.0 - 2.0 / (\text{EXP}(2.0 * X) + 1.0).$$

If  $|X| \geq 10$ ,  $\text{TANH}(X) = \pm 1.0$

If  $X = 0$ ,  $\text{TANH}(X) = 0$

8.3.3.2 Temporary Storage for TANH

Eight cells of temporary storage are required.

They are:

QS	for saving Q	#E2
J	for indexing in series 1000	#DF
X, XS, T1	for floating point numbers, two cells each	
#DD, #DA, #DB		

8.3.4 Arctangent (ATAN)

8.3.4.1 Range Reduction

Let Z be value of input argument.

If:

$Z < 0$ , let  $U = -Z$ , and  $\arctan(Z) = -\arctan(U)$   
 $Z = 0$ , let  $U = Z$ , and  $\arctan(Z) = \arctan(U)$

If:

$U < 1$ , let  $Y = U$ ,  $A = 0$ , and  $B = 1$   
 $U > 1$ , let  $Y = 1/U$ ,  $A = \pi/2$ , and  $B = -1$

If:

$Y \leq \tan(\pi/8)$  let  $T = \pi/16$   
 $Y > \tan(\pi/8)$  let  $T = 3\pi/16$

DOCUMENT CLASS IMS PAGE NO 8-6  
 PRODUCT NAME 1700 MASS Storage FORTRAN  
 PRODUCT NO. COB5 VERSION 2.0 MACHINE SERIES 1700

and  $\arctan(U) = A + B(T + \arctan(X))$

where  $X = \frac{Y - \tan(T)}{1 + Y \tan(T)}$

8.3.4.2 Series for  $\arctan(X)$ ,  $X \leq \tan(\pi/16)$

$$\arctan(X) = C_0 X + C_1 X^3 + C_2 X^5$$

with  $C_0 = .99999900$   
 $C_1 = .33313333$   
 $C_2 = .19000000$

8.3.4.3 Temporary Storage for ATAN

Eleven cells of temporary storage are required.

They are:

PARADD	for parameter address	#E2
ARCRET	for return address	\$D9
ARCFLG	for path decision	\$D8
X, X <sub>2</sub> , AF, BF	for floating point numbers,	\$DA to \$E1

8.3.5 Exponential (EXP)

8.3.5.1 Range Reduction

Let Z be input value of argument

If:

Z < 0 let Y = - Z, and  $e^Z = 1/e^Y$

Z > 0 let Y = Z, and  $e^Z = e^Y$

Z = 0 put answer equal to 1 and exit.

Now  $e^Y = 2^N e^X$  if X and N are defined as follows:

Let  $T = Y/\ln 2$   
 $N = T + \frac{1}{2}$  take integral part  
 $W = T - N$   
 $X = W \ln 2$

Therefore the maximum absolute value for Y is equal to 88.02968 which will generate the maximum floating point number. If input value is greater than this maximum value then a message will be typed out and the answer is given as the maximum floating number permissible in 1700 Fortran.

DOCUMENT CLASS IMS PAGE NO 8-7  
 PRODUCT NAME 1700 Mass Storage FORTRAN  
 PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

8.3.5.2 Series for  $e^X$  for  $X \leq \ln 2/2$

Series is normal Taylor-Mclaurin series truncated at  $n = 7$ ; i.e.,

$$e^X = \sum_{n=0}^6 \frac{X^n}{n!}$$

8.3.5.3 Temporary Storage for EXP

seven cells of temporary storage are required. They are:

PARADD	for parameter address \$EO
RETEXP	for return address \$DA
FLAG,N	for indicator \$DE, \$DC
Y	for floating point number, \$D8, \$D9
QS	for saving Q \$E1

8.3.6 Natural Logarithm (ALOG)

8.3.6.1 Range Reduction

Let Z be input value of argument.

If  $Z \leq 0$  then the answer will be the maximum floating point number possible in 1700.

Now  $Z = 2^N Y$

and  $\ln(Z) = (N - \frac{1}{2}) \ln 2 + \ln \left( \frac{1+X}{1-X} \right)$

where  $X = \frac{Y - \sqrt{2}/2}{Y + \sqrt{2}/2}$

Thus  $|X| \leq 3 - 2\sqrt{2}$

8.3.6.2 Series for  $\ln \left( \frac{1+X}{1-X} \right)$  for  $|X| \leq 3 - 2\sqrt{2}$

$$\ln \left( \frac{1+X}{1-X} \right) = C_0 X + C_1 X^3 + C_2 X^5$$

with  $C_0 = 2.0312500$   
 $C_1 = 0.41666667$   
 $C_2 = 0.90000000$

DOCUMENT CLASS IMS PAGE NO 8-8  
 PRODUCT NAME 1700 Mass Storage FORTRAN  
 PRODUCT NO. 0005 VERSION 2.0 MACHINE SERIES 1700

8.3.6.3 Temporary Storage for ALOG

Nine cells of temporary storage are required.

They are:

PARADD for parameter address #E0  
 LNRET for return address \$D8  
 N,X,X2 for floating point numbers \$D9 to \$DE  
 OS for saving Q \$E1

8.3.7 Square Root (SQRT)

8.3.7.1 Range Reduction

Let Z be input value of argument

If  $Z < 0$  let  $Y = -Z$  and  $\sqrt{Z} = -\sqrt{Y}$

also type out error message.

If  $Z \geq 0$  let  $Y = Z$

Now  $Y = 2^{2N} X$  and  $\sqrt{Y} = 2^N \sqrt{X}$  where  $\frac{1}{2} \leq X < 1$

The Padé approximation for a first estimate with a maximum relative error  $< 2.3 (E-4)$  is

$$y_i = \frac{25}{7} - \frac{\left[ \frac{5000}{343} \left( x + \frac{15}{49} \right) \right]}{\left( x + \frac{15}{49} \right) \left( x + \frac{235}{49} \right) - \frac{400}{2401}}$$

8.3.7.2 Iterative method

The square root is computed via a Newton-Raphson iteration starting with a first approximation,  $y_i$  above. Successive approximations are found from:

$$y_{i+1} = \frac{1}{2} \left( y_i + \frac{X}{y_i} \right)$$

until  $\left| y_{i+1} - y_i \right| \leq 2^{-23}$



DOCUMENT CLASS IMS PAGE NO. 89  
 PRODUCT NAME 1700 Mass Storage FORTRAN  
 PRODUCT NO. CO05 VERSION 2.0 MACHINE SERIES 1700

8.3.7.3 Temporary Storage for SQRT

Ten cells of temporary storage are required:

They are:

PARADD for parameter address #E0  
 QS for saving Q \$E2  
 SQRET for return address \$DA  
 SQEXP for value of N \$DC  
 SQFLG for negative input \$DB  
 X, Y0, Y1 for floating point numbers, \$DD to \$EO and \$D8,\$D9

8.4 Type Out Message

Any illegal input value will cause a predetermined value to be given as an answer and the program will exit normally. It is left to the user to take corrective action.

8.4.1

Routine	Argument	Answer
SIN or COS	$ z  > 2^{21}$	0
EXP	$ z  > 88.02968$	$\infty$
ALOG	$z \leq 0$	$\infty$
SQRT	$z < 0$	$-\sqrt{ z }$

8.5 References:

CONTROL DATA 6600 Computer System, Programming System/Library Functions, A Study of Mathematical Approximations, Pub. No. 60114500.

HANDBOOK OF MATHEMATICAL FUNCTIONS, National Bureau of Standards Applied Mathematics Series-55, Library of Congress catalog number 64-60036.

DOCUMENT CLASS IMS PAGE NO. 8-10  
 PRODUCT NAME 1700 Mass Storage FORTRAN  
 PRODUCT MODEL NO. C005 2.0 MACHINE SERIES 1700

### 8.3.8 Exponentiation {\*\*} Subroutine Q8EXPN

This subroutine is called by programs which use the exponentiation operator, i.e. the symbol \*\*. This routine is written in 1700 Assembly Language. It has the following entry points:

Q8QF2I	floating number to integer power
Q8QI2F	integer number to integer power
Q8QF2F	floating number to floating power
RETAD	
QSAVE	

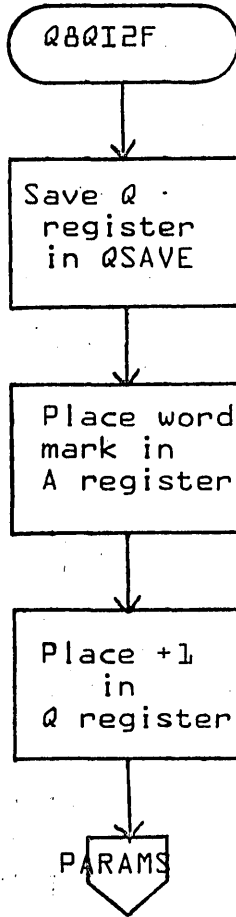
The following externals are declared:

FL0T  
 ALOG  
 EXP

The calls required for temporary storage are:

FL0FLG	for floating/fixed flag \$DC
FRESLT	for intermediate results {floating} \$DD to \$DE
SIGN	for sign of exponent \$DF
COEFF	for address of coefficient \$E1
EXP0	for address or value of exponent \$E2
XLOGC	holds intermediate results \$E1
MLTPR	holds powers of coefficient \$E3 {floating point number - 2 words} \$E4-L & G saved
IRESLT	for intermediate results {fixed} \$E5
FL0VEL	for floating point package overflow \$CB
FLACC	\$C5

# Integer to integer



**CONTROL DATA CORPORATION**  
 SOFTWARE DOCUMENT

SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

DOCUMENT CLASS **IMS** MACH. TYPE **1700**  
 DOCUMENT TITLE **QBEXPN** PAGE **1** of **18**  
 NUMBER **1** ISSUE DATE **1/8**  
 DRAWN BY **1** DATE **1/8**

PROJECT NO.	PROJECT MGR.	PROJECT NAME	TASK NO.	TASK NAME	REV	APPROVED	DATE

5

4

3

2

1

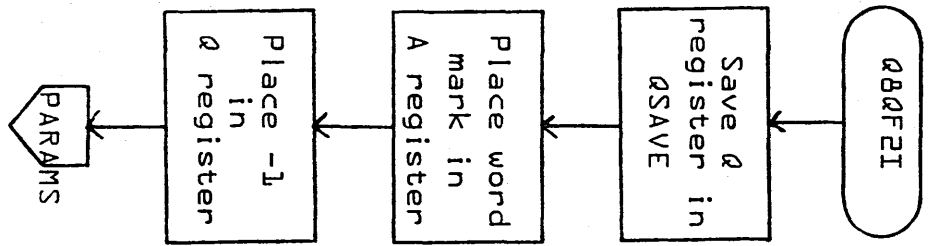
A

B

C

D

Floating to integer

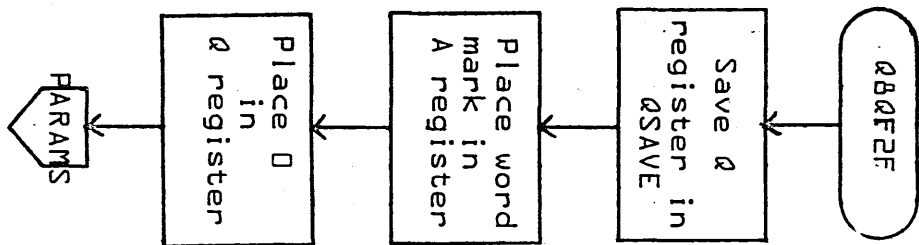


**CONTROL DATA CORPORATION**  
 SOFTWARE DOCUMENT

SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

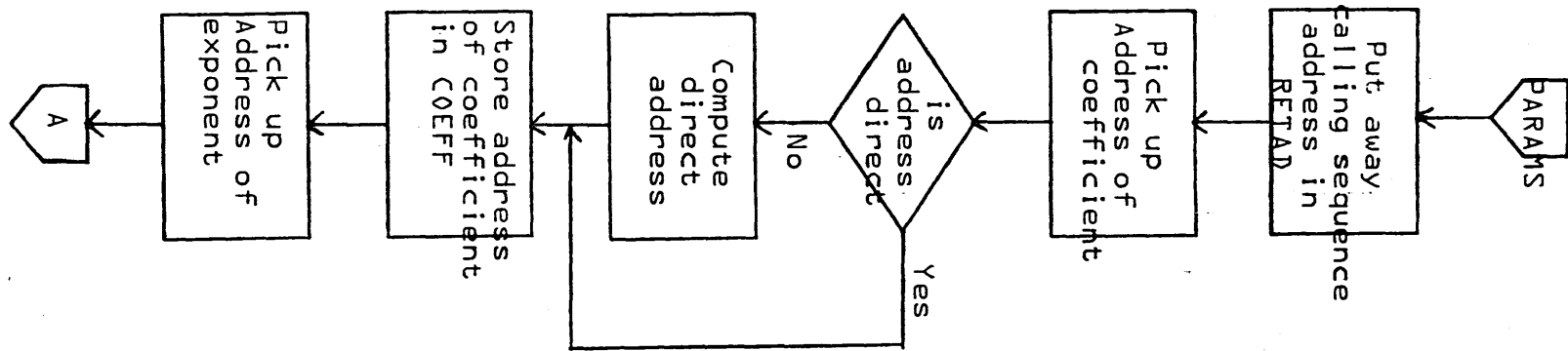
DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	QBEXPN PAGE 2 OF 18			PROJECT MGR.			
NUMBER		ISSUE DATE		TASK NO.			
DRAWN BY		DATE		TASK NAME			

Floating to floating



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	QBEXPN			PROJECT MGR.			
	NUMBER	PAGE 3 OF 18			PROJECT NAME			
		ISSUE DATE				TASK NO.		
	DRAWN BY	DATE			TASK NAME			

8-13



**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

SAMPLE CODE

FLOWCHART

DECISION TABLE

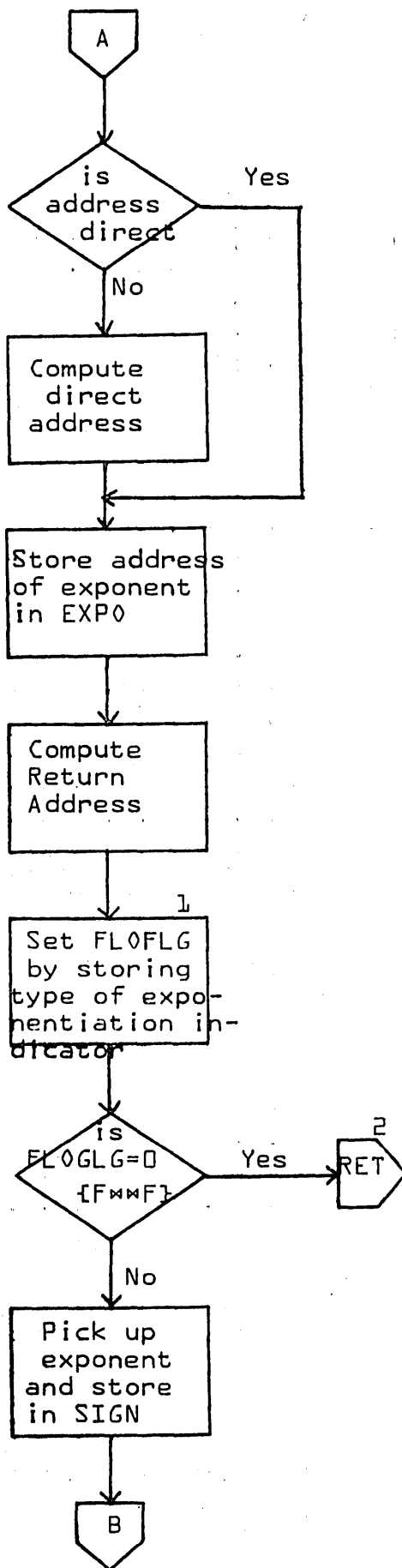
OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	QBEXPN		PAGE 4 OF 18	PROJECT MGR.			
NUMBER		ISSUE DATE		PROJECT NAME			
DRAWN BY		DATE		TASK NO.			
				TASK NAME			

1. FLOFLG equals

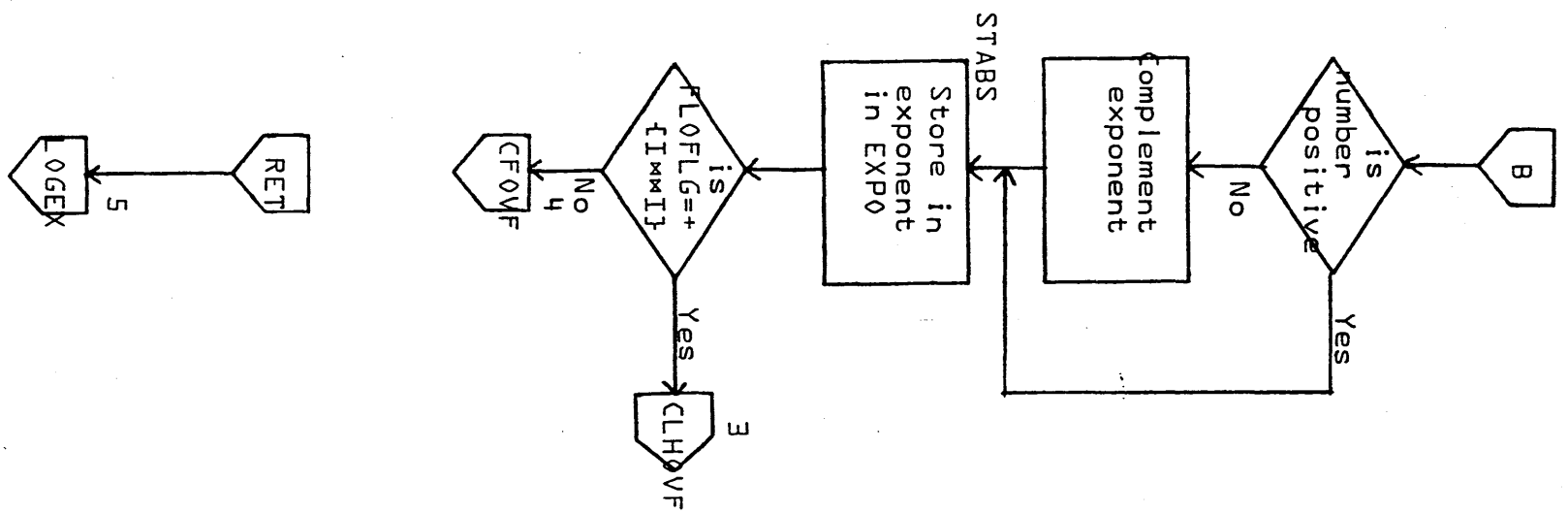
- 1 FMMI
- 0 FMMF
- +1 IMMI

2. From this place a jump is made to LOGEX.



CONTROL DATA CORPORATION SOFTWARE DOCUMENT		DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.		REV		DATE	
SAMPLE CODE <input type="checkbox"/>		DOCUMENT TITLE	Q8EXPN			PROJECT MGR.					
FLOWCHART <input type="checkbox"/>		NUMBER			PAGES of 18	PROJECT NAME					
DECISION TABLE <input type="checkbox"/>		DRAWN BY			ISSUE DATE	TASK NO.					
OTHER <input type="checkbox"/>					DATE	TASK NAME					

- 3. Process  
IMMI
- 4. Process  
FMFI
- 5. Process  
FMWF



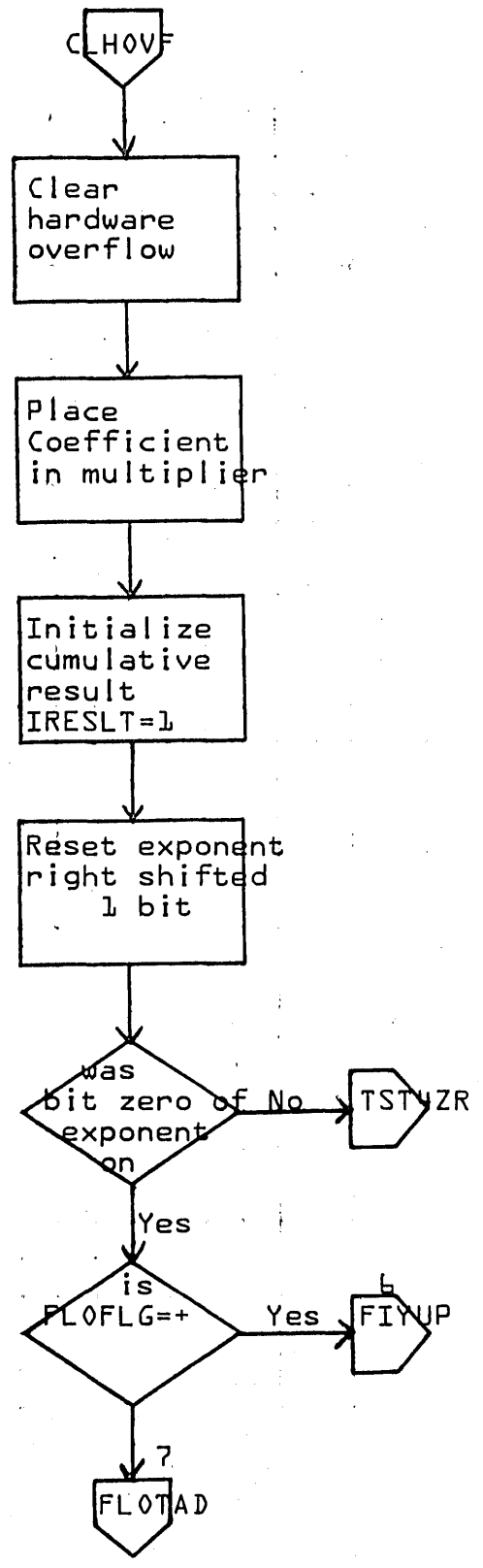
<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	QBEXPN		PAGE 6 OF 18	PROJECT MGR.			
	NUMBER		ISSUE DATE		PROJECT NAME			
	DRAWN BY		DATE		TASK NO.			
					TASK NAME			

B-16



Process integer to integer

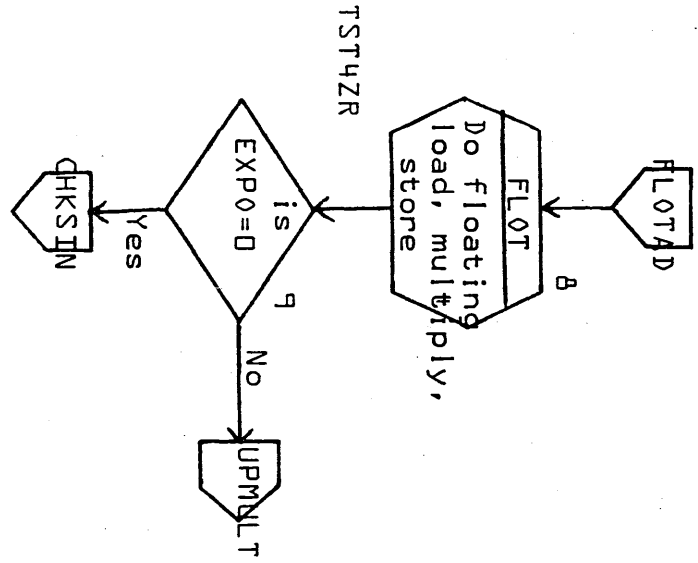
- 6. Select I<sub>MMI</sub>
- 7. Select F<sub>MMI</sub>



CONTROL DATA CORPORATION		DOCUMENT CLASS	MACH. TYPE	PROJECT NO.	REV	APPROVED	DATE
SOFTWARE DOCUMENT		IMS	1700				
SAMPLE CODE		DOCUMENT TITLE		PROJECT MGR.			
FLOWCHART		Q8EXPN		PROJECT NAME			
DECISION TABLE		NUMBER	PAGE 7 of 18	TASK NO.			
OTHER		DRAWN BY	ISSUE DATE	TASK NAME			

3. Floating  
 FRESLT =  
 FRESLT \* MLTPR

9. Test exponent to see  
 if all significant  
 bits have been  
 shifted out.



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	Q8EXPN			PROJECT MGR.			
		PAGE 8 OF 18			PROJECT NAME			
	NUMBER		ISSUE DATE		TASK NO.			
	DRAWN BY		DATE		TASK NAME			

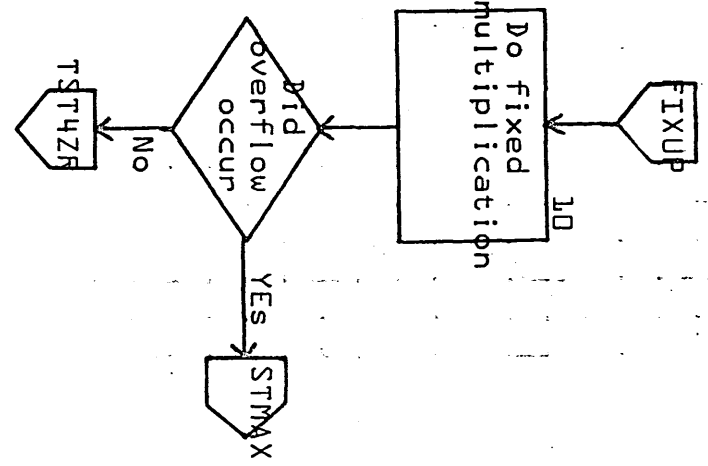
A

B

C

D

10. Fixed  
 IRESLT  
 = IRESLT \* MLTPR



**CONTROL DATA CORPORATION**  
 SOFTWARE DOCUMENT

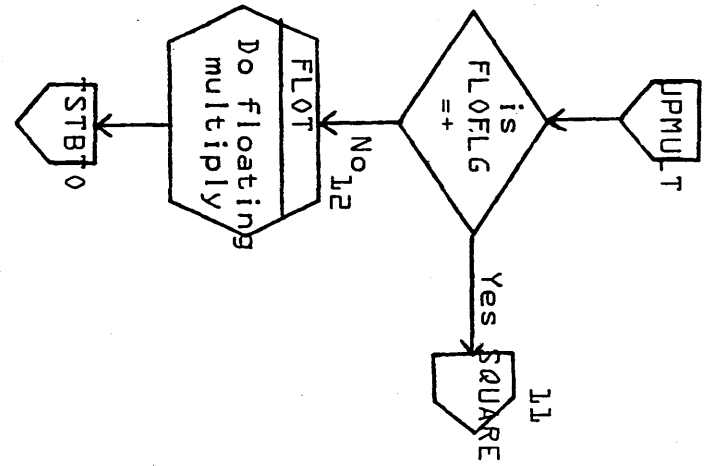
- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	QBEXPN			PROJECT MGR.			
NUMBER		ISSUE DATE	PAGE 9 OF 18	PROJECT NAME			
DRAWN BY		DATE		TASK NO.			
				TASK NAME			

8-19

1.1. Processing  
I M M I

1.2. Square floating  
multiplier  
MLTPR  
=MLTPR\*MLTPR



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE

FLOWCHART

DECISION TABLE

OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	QBEXPN			PROJECT MGR.			
		PAGE	10	PROJECT NAME			
NUMBER		ISSUE DATE	OF 18	TASK NO.			
DRAWN BY		DATE		TASK NAME			

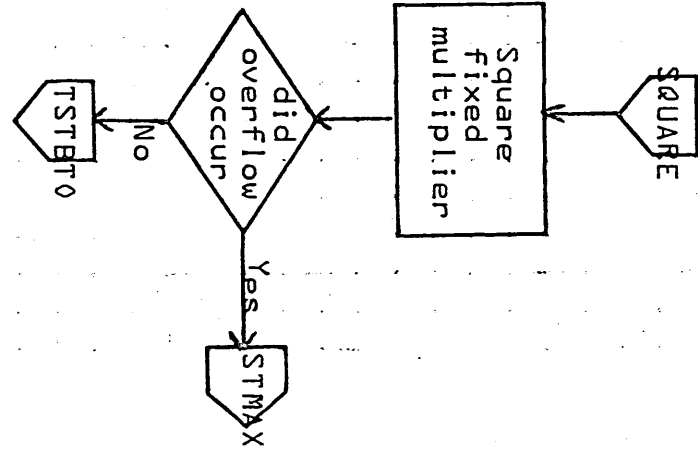
8-210

A

B

C

D



**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

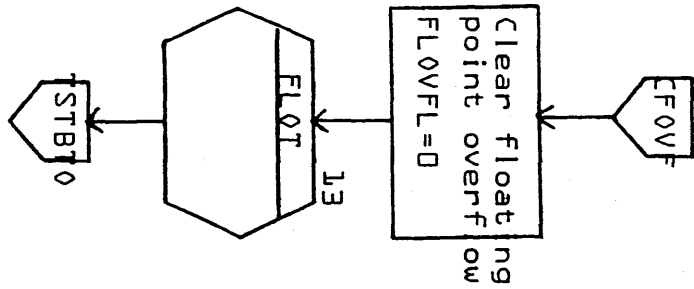
SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	Q8EXPN			PROJECT MGR.			
	PAGE 11 OF 18			PROJECT NAME			
NUMBER	ISSUE DATE			TASK NO.			
DRAWN BY	DATE			TASK NAME			

A-21

Process floating to integer

13. Place coefficient  
in multiplier  
and initialize  
cumulative result.  
MLTPR=COEFF  
FRESLT=1.0



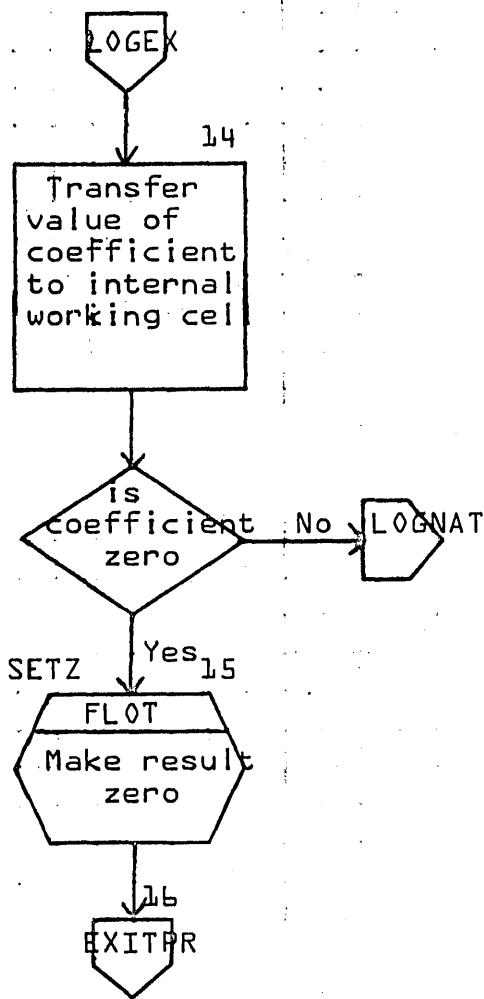
**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	Q8EXPN PAGE 13 OF 18			PROJECT MGR.			
NUMBER		ISSUE DATE		TASK NO.			
DRAWN BY		DATE		TASK NAME			

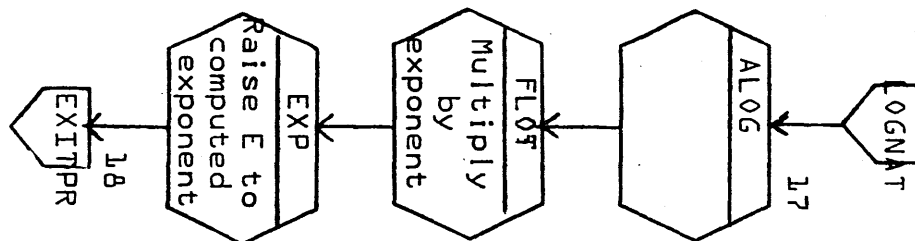
Process floating to floating

- 14. First word into FRESLT;  
Second word into FRESLT+1.
- 15. Make result zero to avoid a fault in the natural log routine.
- 16. Exit



CONTROL DATA CORPORATION SOFTWARE DOCUMENT		DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.		APPROVED		DATE	
SAMPLE CODE <input type="checkbox"/>		DOCUMENT TITLE	08EXPN			PROJECT MGR.		REV			
FLOWCHART <input type="checkbox"/>		NUMBER		ISSUE DATE		PROJECT NAME					
DECISION TABLE <input type="checkbox"/>		DRAWN BY		DATE		TASK NO.					
OTHER <input type="checkbox"/>						TASK NAME					

17. Take natural log  
of coefficient.  
14. Exit

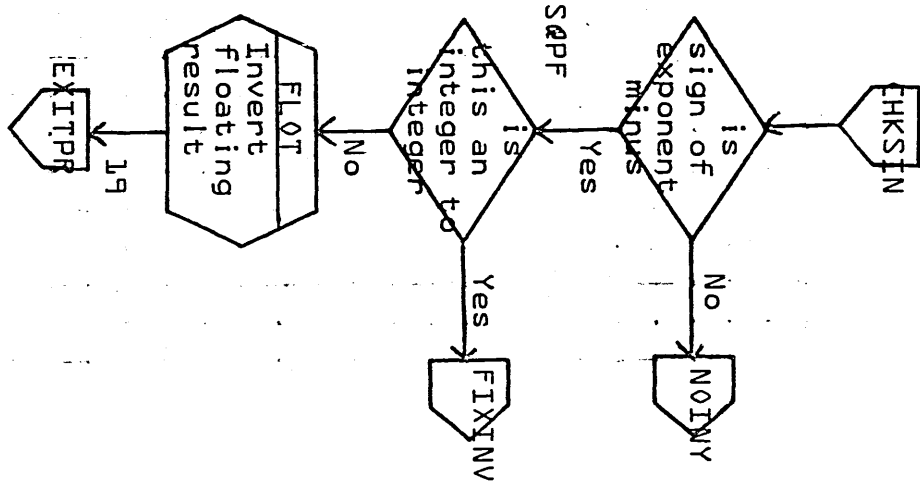


<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	Q8EXPN			PROJECT MGR.			
		PAGE 14 OF 18			PROJECT NAME			
	NUMBER	ISSUE DATE			TASK NO.			
	DRAWN BY	DATE			TASK NAME			

B-24



19. Exit



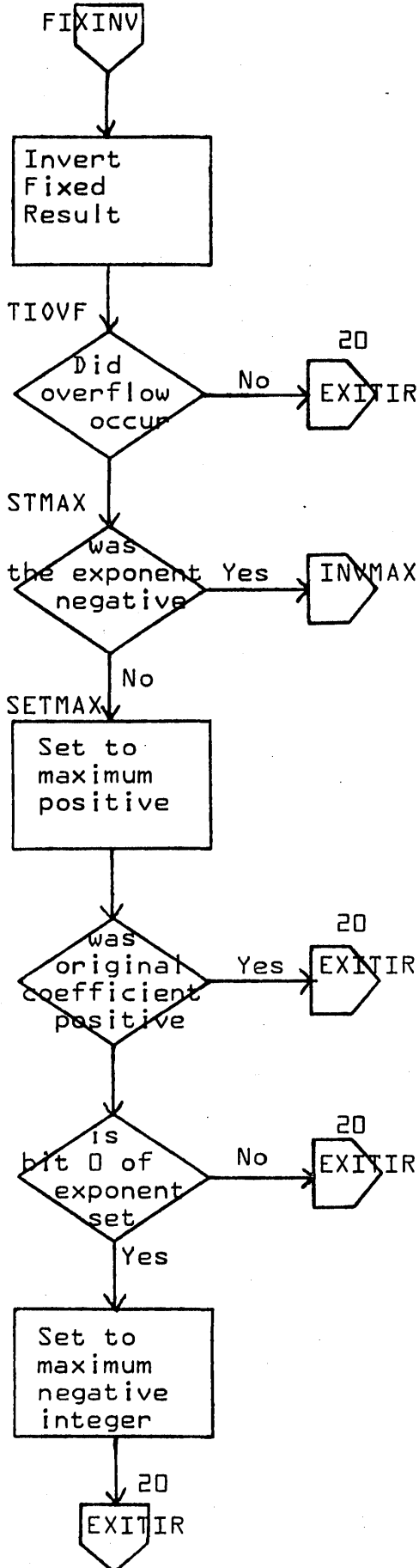
CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV.	APPROVED	DATE
DOCUMENT TITLE	QBEXPN			PROJECT MGR.			
		ISSUE DATE	PAGE 15 OF 18	PROJECT NAME			
NUMBER		DATE		TASK NO.			
DRAWN BY		DATE		TASK NAME			

8-25

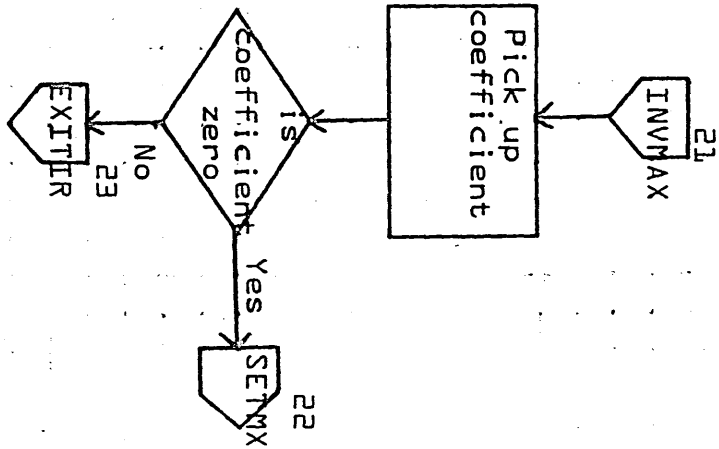
20. Exit



CONTROL DATA CORPORATION SOFTWARE DOCUMENT	DOCUMENT CLASS IMS	MACH. TYPE 1700	PROJECT NO.	REV	APPROVED	DATE
SAMPLE CODE <input type="checkbox"/>	DOCUMENT TITLE QBEXPN	PAGE 1 OF 1A	PROJECT MGR.			
FLOWCHART <input type="checkbox"/>	NUMBER	ISSUE DATE	PROJECT NAME			
DECISION TABLE <input type="checkbox"/>	DRAWN BY	DATE	TASK NO.			
OTHER <input type="checkbox"/>			TASK NAME			

PRINTED IN U.S.A.

- 21. Overflow with negative exponent, return zero.
- 22. Set to maximum value.
- 23. Exit.

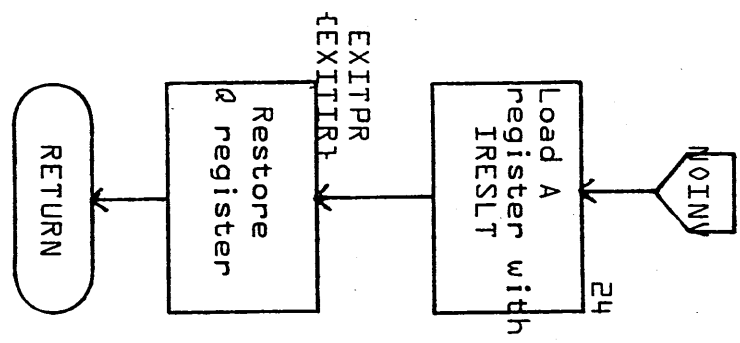


**CONTROL DATA CORPORATION**  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	QBEXPN		PAGE 176 F 18	PROJECT MGR.			
NUMBER		ISSUE DATE		TASK NO.			
DRAWN BY		DATE		TASK NAME			

24. Returns garbage in A register if F2I.



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	QBEXPN		PAGE 18 OF 18	PROJECT MGR.			
	NUMBER		ISSUE DATE		TASK NO.			
	DRAWN BY		DATE		TASK NAME			

DOCUMENT CLASS IMS PAGE NO. 8-29  
 PRODUCT NAME 1700 Mass Storage FORTRAN  
 PRODUCT NO. C005 /ERSION 2.0 MACHINE SERIES 1700

## APPENDIX A

## 1700 Mathematical Approximations

## 1.0 Introduction

This paper will derive approximations for object time 1700 Fortran programs which will allow fast evaluations with necessary accuracy and uncomplicated range reduction.

## 1.1 Accuracy

Since all approximations are to be in floating-point, it shall be required that

$$R < 2^{-23} = 1.192E-07$$

which is the basic round off due to the size of significant digits in 1700.

## 2.0 General Description of Method

The method will combine truncated Taylor-Mclaurin Power Series and Chebyshev Polynomials.

## 2.1 Truncated Taylor-Mclaurin Power Series

A function will be expressed as a truncated Taylor-Mclaurin series such that the error produced is less than R. i.e.,

$$f(N) = \sum_{n=0}^m \frac{f^{(n)}(0)}{n!} x^n, \quad |x| \leq a$$

and m is chosen such that

$$R < \left| \frac{f^{(m+1)}}{(m+1)!} x^{m+1} \right| = R_T$$

## 2.2 Chebyshev Expansions

Denote by  $T_n(x) = \cos(nt)$  the Chebyshev polynomial of degree n in  $x = \cos t$ . Note that  $|x| \leq 1$  and that  $|T_n(x)| \leq 1$ . Writing

$$T_n(x) = \sum_{i=0}^n C_n^i x^i$$

the coefficients  $C_n^i$  are computed from

$$C_n^i = 0 \text{ if } (i + n) \text{ is odd}$$

$$C_n^i = 2^{i-1} \left[ 2^{\binom{(n+i)/2}{(n-i)/2}} - \binom{(n+i-2)/2}{(n-i)/2} \right] (-1)^{\frac{n-i}{2}}$$

if  $(i + n)$  is even.

$$\text{Thus } x^n = T_n(x) - \sum_{i=0}^{n-1} (C_n^i x^i) \cdot \frac{1}{2^{n-1}}$$

Table of  $C_n^i$  is given at end of paper.

### 2.3

Use of Chebyshev Expansion

$$\begin{aligned} f(x) &= \sum_{n=0}^{m-1} \frac{f^{(n)}(x)}{n!} x^n + \frac{f^{(m)}(x)}{m!} x^m \\ &= \sum_{n=0}^{m-1} \frac{f^{(n)}(x)}{n!} x^n + \frac{T_m(x)}{2^{m-1}} - \sum_{i=0}^{m-1} \frac{C_m^i x^i}{2^{m-1}} \cdot \frac{f^{(m)}(x)}{m!} \end{aligned}$$

$$\text{If: } R \leq \left| \frac{T_m(x)}{m! 2^{m-1}} \cdot \frac{f^{(m)}(x)}{m!} \right| + R_T$$

then term involving  $T_m(x)$  can also be neglected and series now becomes

$$f(x) = \sum_{n=0}^{m-1} \left( \frac{f^{(n)}(x)}{n!} - \frac{f^{(m)}(x)}{m! 2^{m-1}} \cdot C_m^n \right) x^n$$

This process can then be repeated as long as the total error does not exceed  $R$ .

3.0 SIN (X) for  $X \leq \pi/2$

The Taylor-Mclaurin series truncated at  $n = 6$  is

$$\text{SIN}(X) = \sum_{n=0}^5 \frac{(-1)^n}{(2n+1)!} x^{2n+1}$$

with Relative Error at  $x = \pi/2 \leq \left| \frac{(\pi/2)^{13}}{13!} \right| = 5.692\text{E-}8$

Let  $x = (\pi/2)y$ , with  $y \leq 1$

$$\text{SIN}(X) = \sum_{n=0}^4 \frac{(-1)^n}{(2n+1)!} y^{2n+1} (\pi/2)^{2n+1} - (\pi/2) \frac{11}{11!} y^{11}$$

Now  $y^{11} = \frac{T_{11}(y)}{2^{10}} - \frac{C_{11}^{2n+1}}{2^{10}} y^{2n+1} \quad n=0, 1, \dots, 4$

Error occasioned in omitting  $T(y)$  term is

$$R_c \leq (\pi/2)^{11} \left| \frac{T_{11}(y)}{11! 2^{10}} \right| = 3.511\text{E-}9$$

Total error so far is  $6.043\text{E-}8$  and series becomes

$$\text{SIN}(X) = \sum_{n=0}^4 \left[ \frac{(-1)^n}{(2n+1)!} \cdot (\pi/2)^{2n+1} + (\pi/2)^{11} \cdot \frac{C_{11}^{2n+1}}{11! 2^{10}} \right] \cdot y^{2n+1} \quad (1)$$

or  $\text{SIN}(X) = \sum_{n=0}^3 \left[ \frac{(-1)^n}{(2n+1)!} \cdot (\pi/2)^{2n+1} + (\pi/2)^{11} \cdot \frac{C_{11}^{2n+1}}{11! 2^{10}} \right] \cdot y^{2n+1}$

$$+ \left( \frac{1}{9!} (\pi/2)^9 + (\pi/2)^{11} \cdot \frac{C_{11}^9}{11! 2^{10}} \right) \cdot y^9$$

$$y^9 = \frac{T_9(y)}{2^8} - \frac{C_9^{2n+1}}{2^8} \cdot y^{2n+1} \quad , n = 0, 1, 2, 3$$

Error caused in omitting  $T_9(y)$  is  $6.6\text{E-}7$  and therefore cannot be neglected.

Thus using (1) and  $y = (2/\pi)X$  we get

$$\sin(X) = \sum_{n=0}^4 \frac{(-1)^n}{(2n+1)!} + \frac{C_{11}^{2n+1}}{11! 2^{10}} (\pi/2)^{10-2n} \cdot x^{2n+1}$$

Therefore:  $\sin(X) = \sum_{n=0}^4 A_n X^{2n+1}$

Define  $B = (\pi/4)^2$  then

$$A_0 = 1 - \frac{B^5}{10!} = 0.99999998$$

$$A_1 = -1/3! + \frac{5B^4}{10!} = -.166666468$$

$$A_2 = 1/5! - \frac{7B^3}{10!} = .83328836E-2$$

$$A_3 = -1/7! + \frac{4B^2}{10!} = -.19799328E-3$$

$$A_4 = 1/9! - \frac{B}{10!} = .25857445E-5$$



4.0  $\text{TANH}(X)$  for  $x \leq \pi/8$

Taylor-Mclaurin series for  $\text{TANH}(X)$  is

$$\text{TANH}(X) = \sum_{n=1}^{\infty} \frac{(2)^{2n} (2^{2n}-1)}{(2n)!} \cdot B_{2n}(0) X^{2n-1}$$

where  $B_{2n}(0)$  is the Bernoulli-Euler number  $B_{2n}(X)$  evaluated at  $X = 0$ .

$$B_{2n}(x) = \frac{(-1)^{n-1} 2 (2n)!}{(2\pi)^{2n}} \cdot \sum_{N=1}^{\infty} \frac{\cos 2N\pi X}{N^{2n}}$$

$$B_{2n}(0) = \frac{(-1)^{n-1} 2 (2n)!}{(2\pi)^{2n}} \cdot R_{2n}$$

where  $R_{2n} = \sum_{N=1}^{\infty} N^{-2n}$  is the Riemann-Zeta function.

2n	$R_{2n}$
2	1.64493407
4	1.08232323
6	1.01734306
8	1.00407736
10	1.00099458
12	1.00024609
14	1.00006125

so that series can be written as:

$$\text{TANH}(X) = \sum_{n=1}^{\infty} \frac{2(2^{2n}-1)}{\pi^{2n}} \cdot R_{2n} X^{2n-1} \cdot (-1)^{n-1}$$

DOCUMENT CLASS 128 PAGE NO             
 PRODUCT NAME Mass Storage FORTRAN  
 PRODUCT NO. (1115) VERSION 1.0 MACHINE SERIES 1/00

Let  $X = (\pi/8) \cdot y$  and truncate series at  $n = 7$

$$\text{TANH}(X) = \sum_{n=1}^6 \frac{2(2^{2n}-1)}{2^{6n-3}} \cdot R_{2n} y^{2n-1}$$

with error  $\leq \left| \frac{2^{14}-1}{2^{38}} \cdot R_{14} \right| = 1.897E-8$

$$\text{TANH}(X) = \sum_{n=1}^5 \frac{2(2^{2n}-1)}{2^{6n-3}} \cdot R_{2n} \cdot y^{2n-1} + \frac{(2^{12}-1)}{(2)^{32}} \cdot R_{12} \cdot y^{11}$$

The Chebyshev expansion for  $y^{11}$  is

$$y^{11} = \frac{T_{11}(y)}{2^{10}} - \frac{C_{11}^{2n-1}}{2^{10}} \cdot y^{2n-1}$$

and error in omitting  $T_{11}(y)$  term is

$$\left| \frac{2^{12}-1}{2^{42}} \cdot R_{12} \right| = 2.965E-10$$

and total error so far  $1.927E-8$  and series is

$$\text{TAN}(X) = \sum_{n=1}^5 \left[ \frac{(2^{2n}-1)}{2^{6n-4}} \cdot R_{2n} - \frac{(2^{12}-1)}{(2)^{42}} \cdot R_{12} \cdot C_{11}^{2n-1} \right] \cdot y^{2n-1} \quad (1)$$

Even though similar expansion for  $y^9$  only gives a total error  $3.84E-8$  the value of

$$\tan(y) = \frac{(\sqrt{2} + 1) + \tan(x)}{1 - (\sqrt{2} + 1) \tan(x)}$$

becomes very critical as  $y \rightarrow \pi/2$  and  $\tan(x) \rightarrow \sqrt{2} - 1$ .

Therefore series will not be reduced any further.

Substituting  $y = X 8/\pi$  in (1) we get

$$\text{TANH}(X) = \sum_{n=1}^5 \left[ 2(2^{2n}-1)R_{2n} - \frac{E \cdot C_{11}^{2n-1}}{11(2)^{30-6n}} \cdot \frac{X^{2n-1}}{2^n} \right]$$

where  $E = \frac{(2^{12} - 1) \cdot 11}{(2)^{15}} \cdot R_{12}$

Define  $D = \pi^{-2}$  then

$$A_1 = D \cdot [6R_2 + E \cdot 2^{-24}] = 1.0000001$$

$$A_2 = D^2 \cdot [30R_4 - 5E \cdot 2^{-16}] = -.33333227$$

$$A_3 = D^3 \cdot [126R_6 + 7E \cdot 2^{-8}] = .13337246$$

$$A_4 = D^4 \cdot [510R_8 - 4E] = -.053388615$$

$$A_5 = D^5 \cdot [2046R_{10} + E \cdot 2^8] = .025628253$$

DOCUMENT CLASS IM PAGE NO 8-36  
 PRODUCT NAME 1700 Mass Storage FORTRAN  
 PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

5.0 Arctan(X) for  $X \leq \tan(\pi/16)$

Taylor-Mclaurin series is

$$\arctan(X) = \sum_{n=0}^{\infty} \frac{(-1)^n X^{2n+1}}{2n+1}$$

Since  $X = \tan(\pi/16) < .2$  then error caused by replacing  $X = .2y$  is greater than true error. True value of  $\tan(\pi/16) = .19891237$ .

$$\text{Then } \arctan(X) = \sum_{n=0}^{\infty} \frac{(-1)^n 2^{2n+1}}{(2n+1)(10)^{2n+1}} \cdot y^{2n+1}$$

truncation at  $n = 4$  causes error

$$\leq \left| \frac{2^9}{9 \cdot 10^9} \right| = 5.6888E-8$$

$$\text{Using } y^7 = \frac{T_7(y)}{2^6} - \frac{C_7^{2n+1}}{2^6} y^{2n+1} \quad \text{in}$$

$$\arctan(X) = \sum_{n=0}^2 \frac{(-1)^n 2^{2n+1}}{(2n+1)(10)^{2n+1}} \cdot y^{2n+1} - \frac{2^7}{(7)10^7} y^7$$

will give error  $\leq \left| \frac{2}{7 \times 10^7} \right| = 1.857E-8$  if  $T_7$  term is omitted. Then

total error so far is  $\leq 7.545E-8$  and series becomes

$$\arctan(X) = \sum_{n=0}^2 \left[ \frac{(-1)^n 2^{2n+1}}{(2n+1)(10)^{2n+1}} + \frac{2 \cdot C_7^{2n+1}}{(7)10^7} \right] \cdot y^{2n+1}$$

Substituting  $y = \frac{10X}{2}$  gives

DOCUMENT CLASS MS PAGE NO 8-37  
 PRODUCT NAME 1700 Mass Storage FORTRAN  
 PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

$$\arctan(x) = \sum_{n=0}^2 \left[ \frac{(-1)^n}{(2n+1)} + \frac{C_7}{7} \frac{10^{2n-6}}{2^{2n}} \right] \cdot x^{2n+1}$$

or

$$\text{Arctan}(X) = \sum_{n=0}^2 A_n X^{2n+1}$$

and

$$A_0 = 1 - 10^{-6} = .99999900$$

$$A_1 = -1/3 + 2 \cdot (10^{-4}) = -.33313333$$

$$A_2 = 1/5 - 10^{-2} = .19000000$$

6.0 Exponential,  $e^x$  for  $x \leq \frac{\ln 2}{2}$

Taylor-Mclaurin series is

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

at  $x = \frac{\ln 2}{2}$  and  $n = 7$  error is

$$\leq \left| \left( \frac{\ln 2}{2} \right)^7 \right| \cdot 1/7! = 1.1916E-7$$

Therefore series is normal series truncated after  $n = 6$ , i.e.,

$$e^x = \sum_{n=0}^6 \frac{x^n}{n!}$$

Now  $X$  and  $N$  can always be found such that  $e^y = 2^N e^X$  where  $X \leq \frac{\ln 2}{2}$

If  $X$  and  $N$  are defined as follows

$$\text{Let } t = y/\ln 2$$

$$N = t + \frac{1}{2} \quad \text{integral part}$$

$$w = t - N$$

$$X = w \ln 2 .$$

then

$$X = (t-N) \cdot \ln 2 = t \cdot \ln 2 - \ln(2)^N$$

$$e^X = e^y \cdot 2^{-N}$$

$$\text{and } e^y = 2^N e^X$$

7.0 Natural Logarithm X for  $|X| \leq (3 - 2\sqrt{2})$

The Taylor-Maclaurin series for  $\ln$  is

$$\ln \left( \frac{1+X}{1-X} \right) = 2 \sum_{n=0}^{\infty} \frac{X^{2n+1}}{2n+1}$$

truncating series at  $n = 4$  causes error

$$\left| \frac{2 \cdot (.17157287)^9}{9} \right| = 2.86E-8$$

and  $\ln \left( \frac{1+X}{1-X} \right) = \sum_{n=0}^3 \frac{2X^{2n+1}}{(2n+1)} \quad \dots \quad \text{Let } X = aY$

$$= \sum_{n=0}^3 \frac{2(aY)^{2n+1}}{2n+1}$$

$$Y^7 = \frac{T_7(Y)}{2^6} - \frac{C_7}{2^6} Y^{2n+1}$$

error in omitting  $T_7(x)$  term is

$$\leq \left| \frac{(.17157287)^7}{7 \cdot 2^5} \right| = 1.954E-8$$

Total error so far  $\leq 4.81E-8$

Attempt at further Chebyshev reduction will cause too great an error. Therefore final series is after substitution of  $Y = X/a$

$$\ln \frac{1+X}{1-X} = \sum_{n=0}^2 \left[ \frac{2}{2n+1} - \frac{a^{6-2n} C_7^{2n+1}}{7 \cdot 2^5} \right] X^{2n+1}$$

DOCUMENT CLASS IMS PAGE NO. 8-40  
 PRODUCT NAME 1700 Mass Storage FORTRAN  
 PRODUCT NO. C005 VERSION 2.0 MACHINE SERIES 1700

$$\ln \frac{1+X}{1-X} = \sum_{n=0}^{\infty} A_n X^{2n+1}$$

$$A_0 = 2 + 2^{-5} \cdot 6 = 2.0000008$$

$$A_1 = 2/3 - 2^{-2} \cdot 4 = 0.66645003$$

$$A_2 = 2/5 + a^2/2 = 0.41471862$$

Given  $Y = 2^N Z$  where  $1/2 \leq Z < 1$

Then  $\ln(Y) = N \cdot \ln 2 + \ln Z$ .

If X is defined as  $X = \frac{Z - \sqrt{2}/2}{Z + \sqrt{2}/2}$

then  $|X| \leq 3 - 2\sqrt{2}$

and  $\ln Z = -\frac{\ln 2}{2} + \ln \frac{1+X}{1-X}$

Finally  $\ln Y = (N - 1/2) \cdot \ln 2 + \ln \frac{1+X}{1-X}$



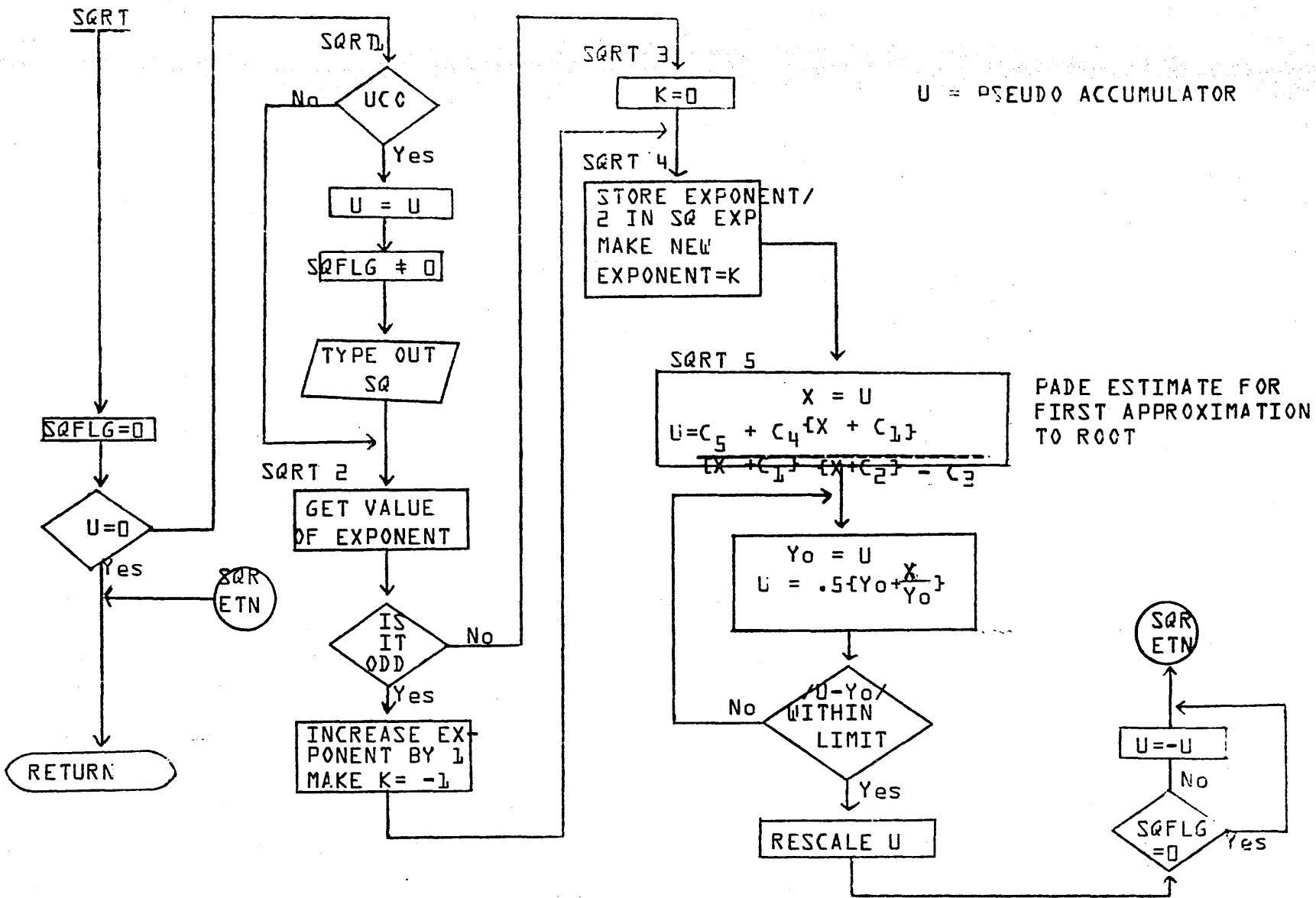
CONTROL DATA CORPORATION • DEVELOPMENT DIV • SOFTWARE DOCUMENT

DOCUMENT CLASS IMS PAGE NO B-41  
 PRODUCT NAME L700 Mass Storage FORTRAN  
 PRODUCT NO. CDD5 VERSION 2.0 MACHINE SERIES 1700

TABLE OF  $C_m^{2n+1}$

n \ m	5	7	9	11	13	15	17
1	$2^0$	5	-7	9	-11	13	-15
3	$2^2$	-5	14	-30	55	-91	140
5	$2^4$	1	-7	27	-77	182	-378
7	$2^6$		1	-9	44	-156	450
9	$2^8$			1	-11	65	-275
11	$2^{10}$				1	-13	90
13	$2^{12}$					1	-15
15	$2^{14}$						1
17	$2^{16}$						

Ex.  $C_{13}^9 = 65 \cdot 2^8$  and  $y^{13} = \frac{T_{13}}{2^{12}} - \frac{C_{13}^{2n+1}}{2^{12}} y^{2n+1}$



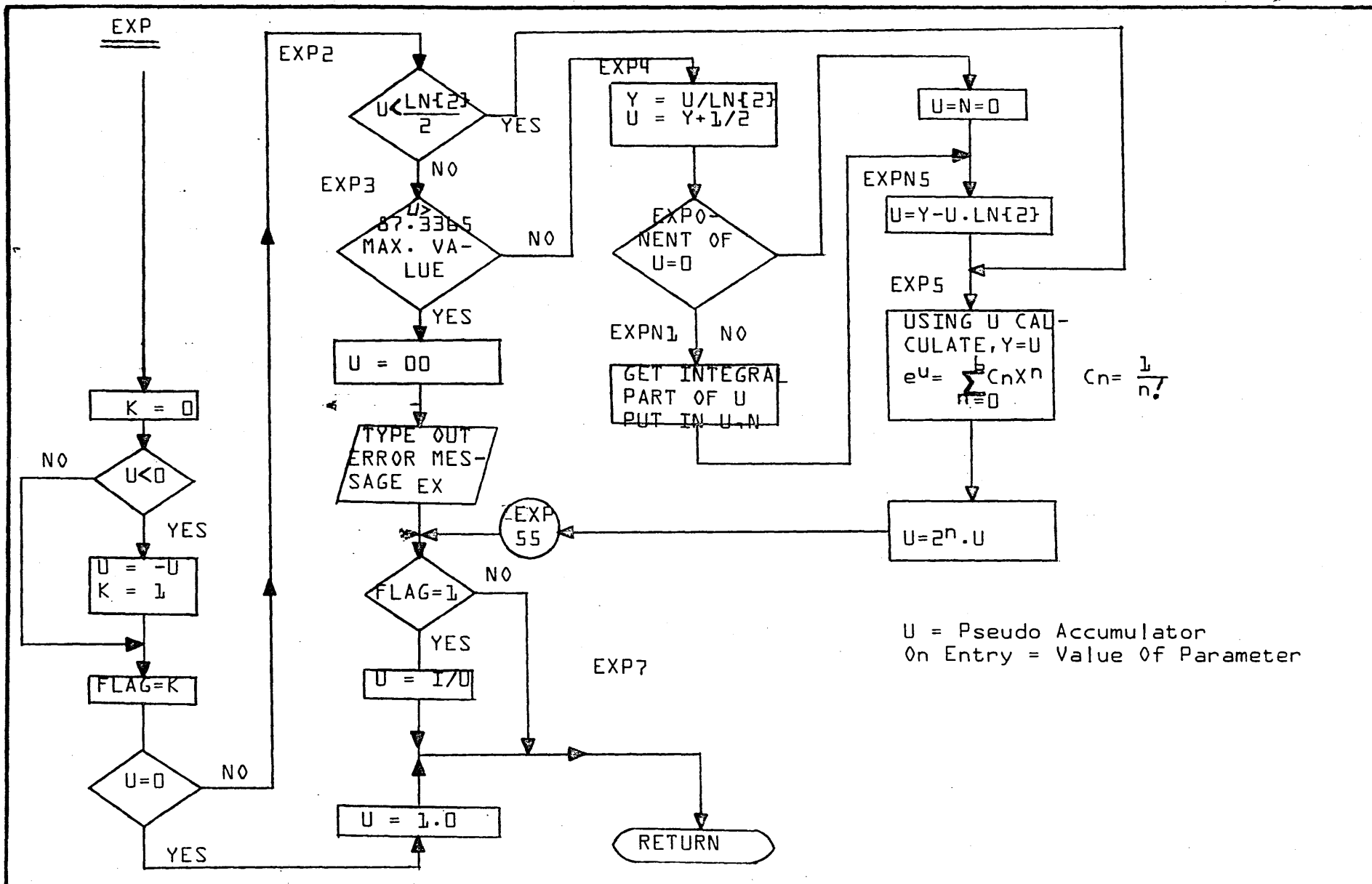
CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE   
FLOWCHART   
DECISION TABLE   
OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700
DOCUMENT TITLE	SQRT	PAGE 1 OF 1	
NUMBER	ISSUE DATE		
DRAWN BY	DATE		

PROJECT NO.	
PROJECT MGR.	
PROJECT NAME	
TASK NO.	
TASK NAME	

REV	APPROVED	DATE

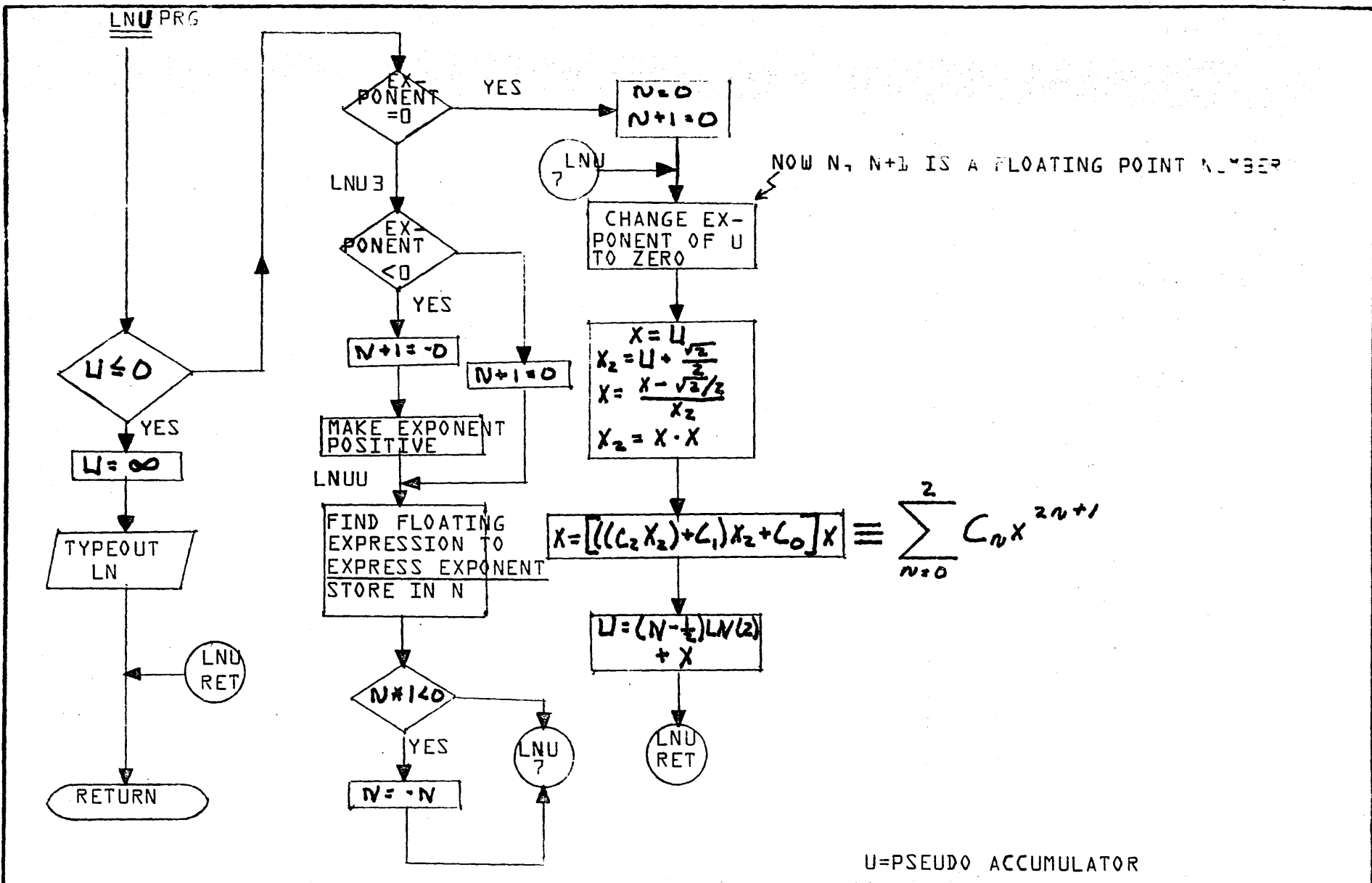


CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	EXP	PAGE 1 OF 1		PROJECT MGR.			
NUMBER	ISSUE DATE	TASK NO.					
DRAWN BY	DATE	TASK NAME					

Eh-9



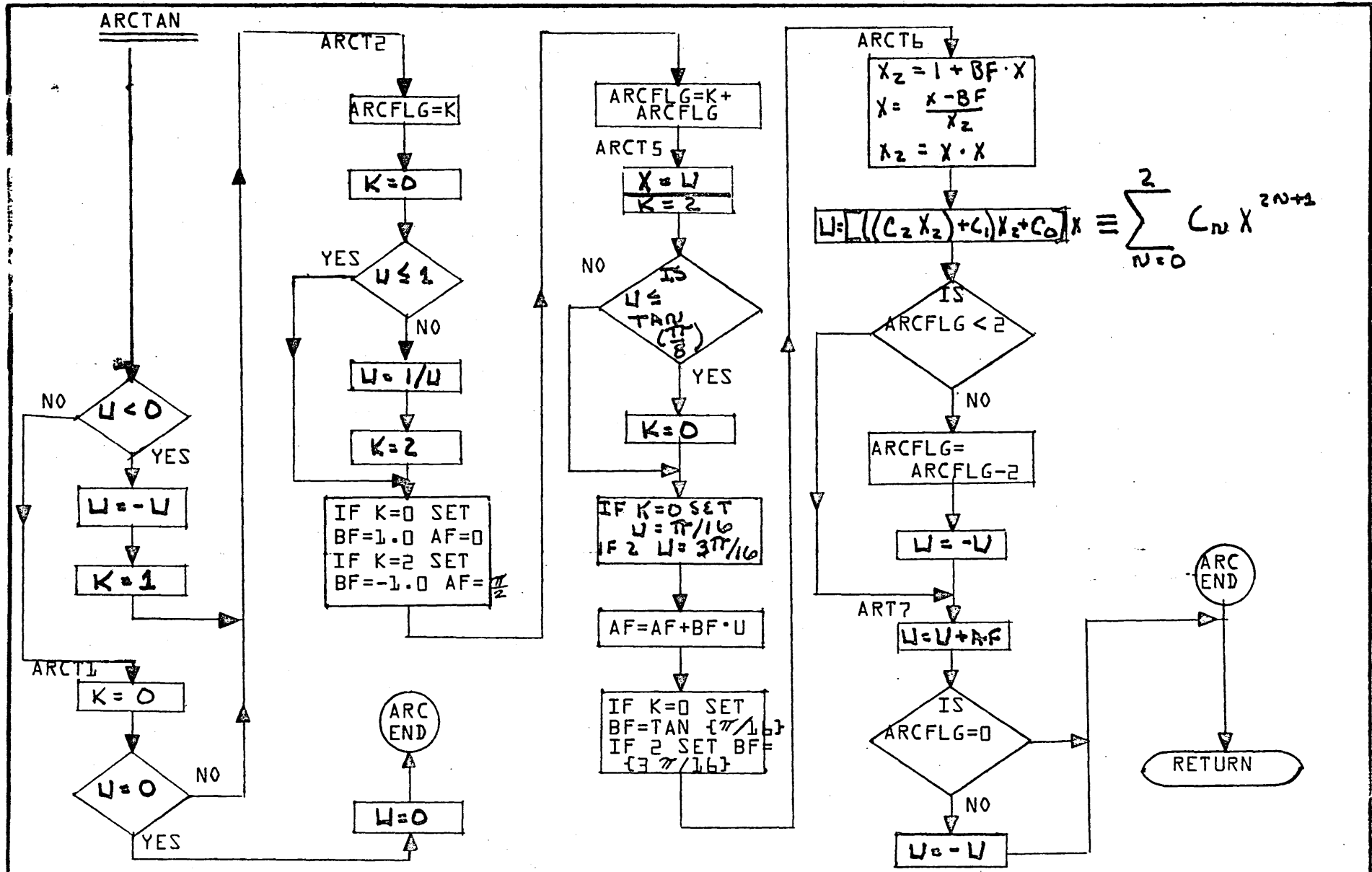
U=PSEUDO ACCUMULATOR

CONTROL DATA CORPORATION SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	LNU PRG	PAGE 1 OF 1		PROJECT MGR.			
	NUMBER		ISSUE DATE		PROJECT NAME			
	DRAWN BY		DATE		TASK NO.			
					TASK NAME			

B-4-B

ARCTAN

A  
B  
C  
D



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

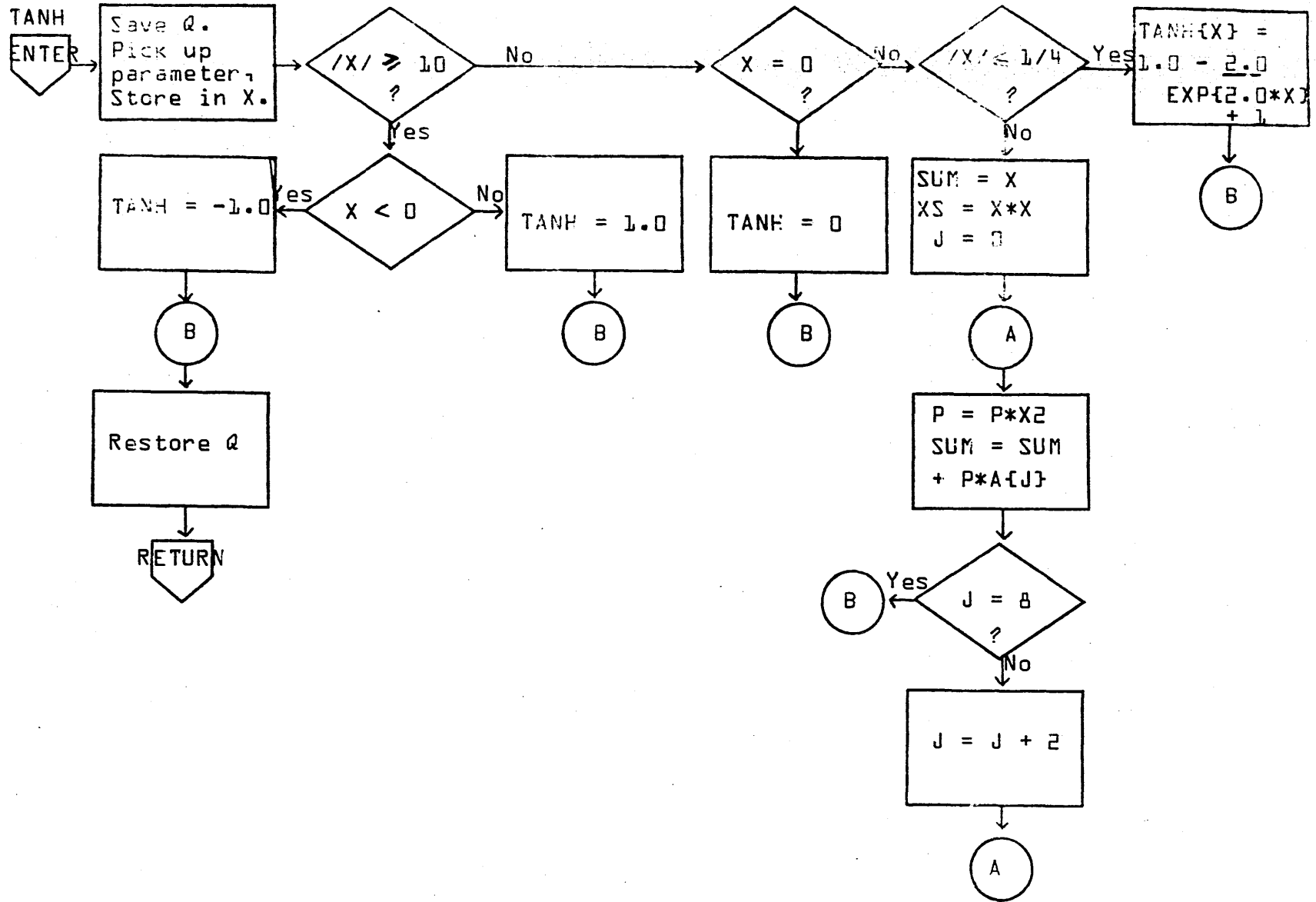
SAMPLE CODE

FLOWCHART

DECISION TABLE

OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	L700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	ARCTAN	PAGE 1 OF 1		PROJECT MGR.			
NUMBER	ISSUE DATE			PROJECT NAME			
DRAWN BY	DATE			TASK NO.			
				TASK NAME			



CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE

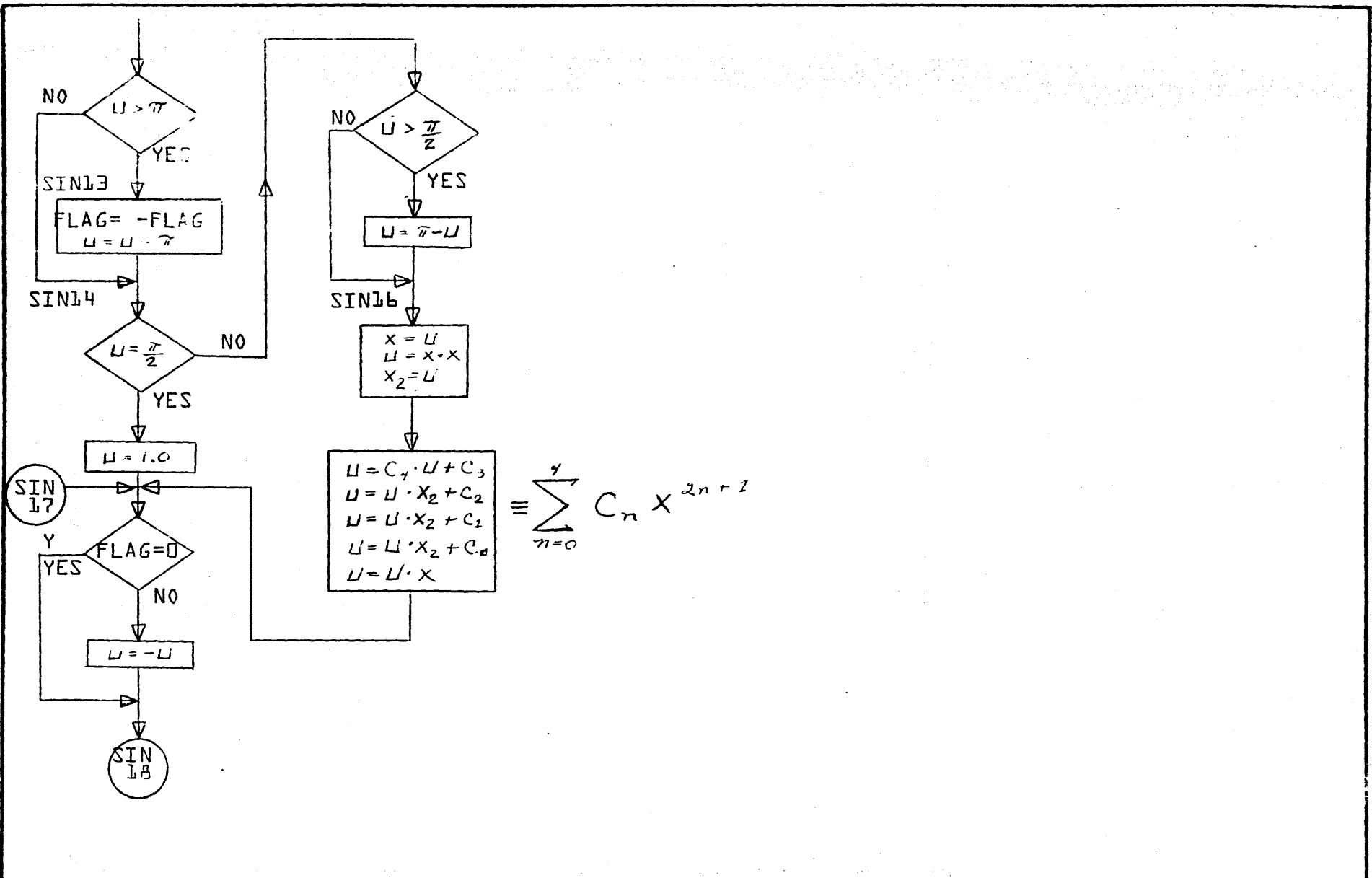
FLOWCHART

DECISION TABLE

OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	TANH			PROJECT MGR.			
	PAGE 1 OF 1			PROJECT NAME			
NUMBER	ISSUE DATE			TASK NO.			
DRAWN BY	DATE			TASK NAME			

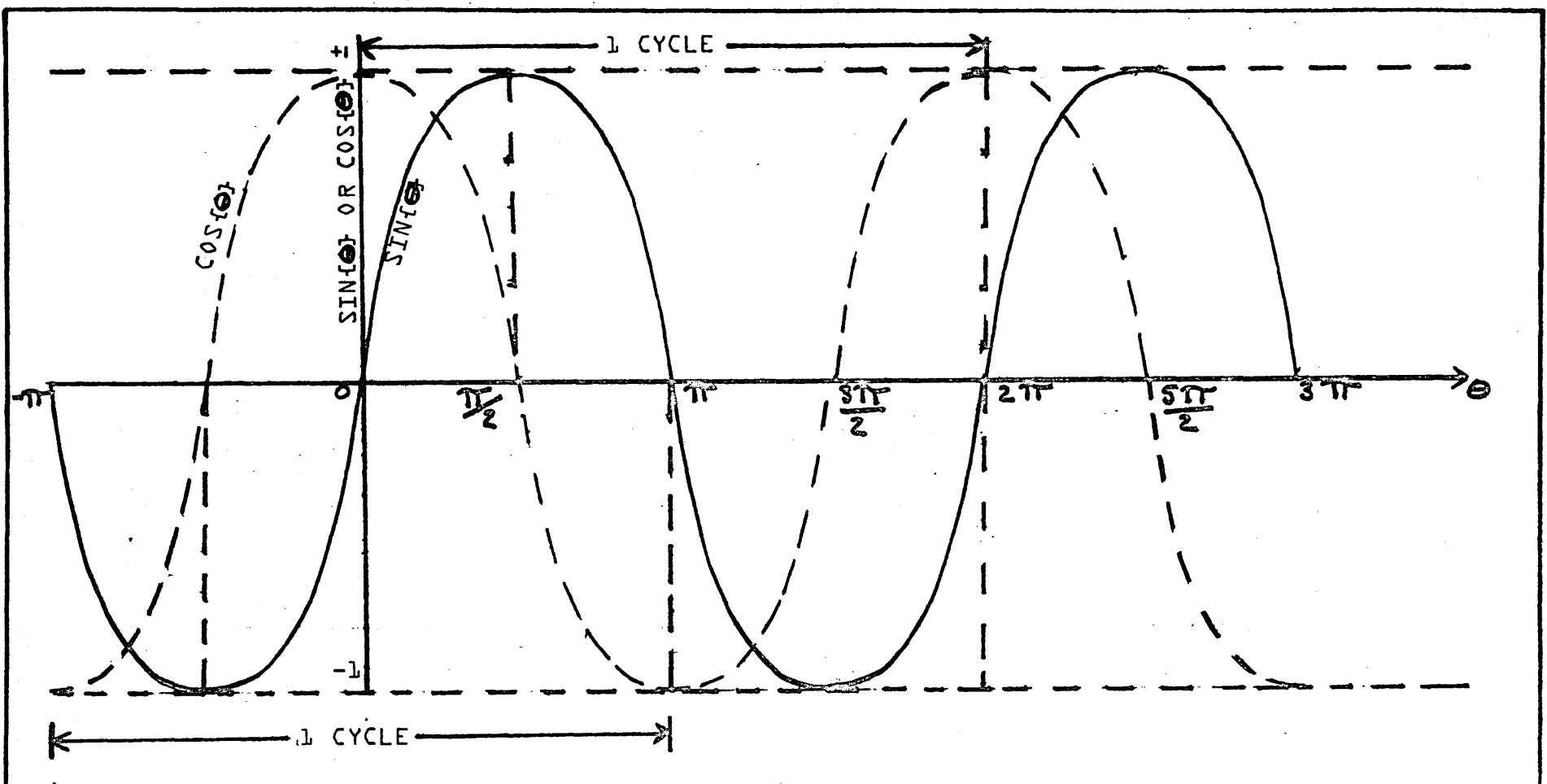




<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	COS&SIN		PROJECT MGR.				
	PAGE 2 OF 3				PROJECT NAME			
	NUMBER	ISSUE DATE		TASK NO.				
	DRAWN BY		DATE		TASK NAME			

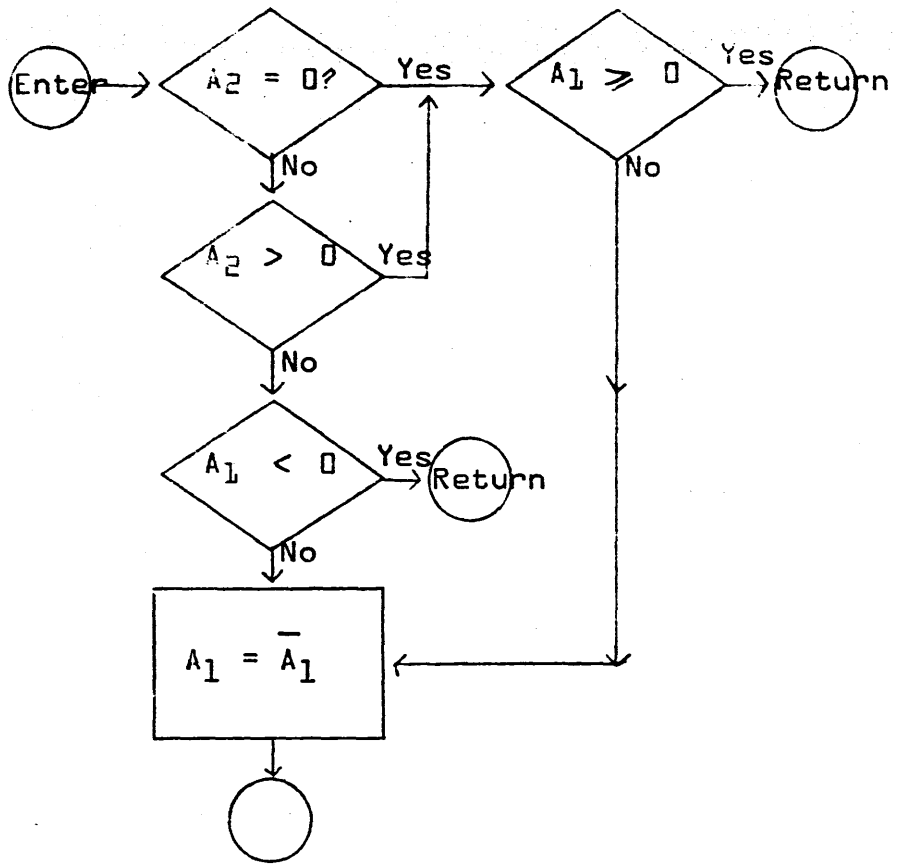
B-14B





$\cos\{\theta\} = \sin\{\theta + \frac{\pi}{2}\}$   
 LET  $\theta$  BE INPUT VALUE. IF  $\theta < 0$ ,  $\sin\{\theta\} = -\sin\{|\theta|\}$ .  
 LET I BE INTEGRAL PART OF  $\theta / 2\pi$ . THEN  $\beta = \theta - I \cdot 2\pi$  IS  $< 2\pi$ .  
 IF  $\beta > \pi$  THEN  $\sin\{\beta\} = -\sin\{2\pi - \beta\}$ , LET  $\beta = 2\pi - \beta$   
 IF  $\beta > \pi/2$  THEN  $\sin\{\beta\} = \sin\{\pi - \beta\}$ .

<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	COS&SIN		PAGE 7 OF 8	PROJECT MGR.			
	NUMBER	ISSUE DATE	PROJECT NAME	TASK NO.				
	DRAWN BY	DATE	TASK NAME					



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	SIGN		PROJECT MGR.				
	PAGE 1 OF 1				PROJECT NAME			
	NUMBER	ISSUE DATE		TASK NO.				
	DRAWN BY		DATE		TASK NAME			

DOCUMENT CLASS IMS PAGE NO. 9-1  
PRODUCT NAME 1700 Mass Storage FORTRAN  
PRODUCT MODEL NO. COO5 V2.0 MACHINE SERIES 1700

## TABLE OF CONTENTS

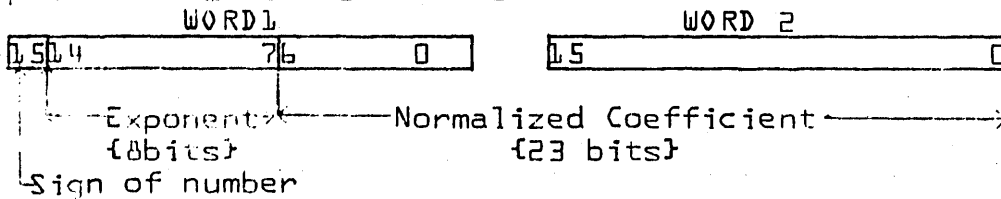
- 9.0 Floating Point Package
- 9.1 General Description
  - 9.1.1 Exponent
  - 9.1.2 Coefficient
  - 9.1.3 Calling Sequence
  - 9.1.4 Operations
  - 9.1.5 Operands
    - 9.1.5.1 Absolute Addressing
    - 9.1.5.2 Relative Addressing
    - 9.1.5.3 Example
  - 9.1.6 Fault Condition
- 9.2 Floating Point Arithmetic with 23 bit numbers
  - 9.2.1 Introduction
  - 9.2.2 The Four Arithmetic Operations
- 9.3 Low Core Storage Location
- 9.4 Entry Points and Externals
- 9.5 Flow chart of FLOAT

DOCUMENT CLASS IMS PAGE NO. 9-2  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005 V2.0 MACHINE SERIES 1700

9.0 FLOATING POINT PACKAGE

9.1 General Description

Each floating point number requires two consecutive 1700 Computer words. The first word, containing the most significant bits of the number, is the one addressed. Normalized floating point format is as follows:



A floating point number X is in the range given below and is significant to one part in  $8 \times 10^6$ .

$$-2^{127} \{1 - 2^{-23}\} \leq X \leq 2^{127} \{1 - 2^{-23}\}$$

If the most significant word is zero (16 bits of zero or one), a floating point zero is assumed.

9.1.1 Exponent

The floating point exponent is an 8-bit quantity with a value ranging from 00 to FF<sub>16</sub>. Through biasing by 80<sub>16</sub>, this range expresses both positive and negative exponents.

9.1.2 Coefficient

The coefficient consists of a 23 bit number n,  $1 - 2^{-23} \geq |n| \geq 0$ . The high order bit position of the first word is the coefficient sign bit.

A zero denotes a positive coefficient and a one denotes a negative coefficient. When the coefficient is negative, the entire floating point number is stored in complement form.

9.1.3 Calling Sequence

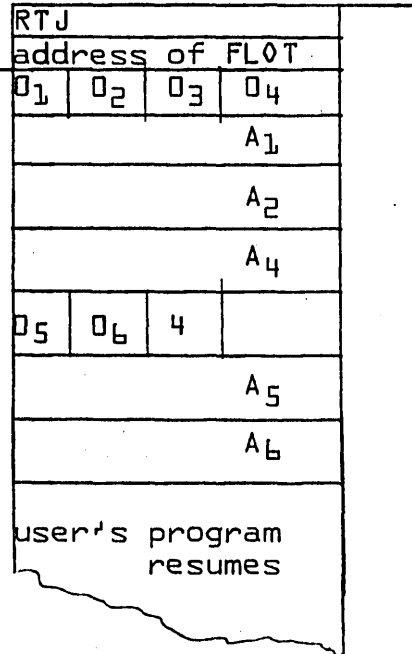
FLOT uses an interpretive calling sequence. It does not save Q or FF; it uses the communications locations #C5 and #C6 as a pseudo accumulator. The pseudo accumulator is retained between calls to FLOT.

The interpretation of the calling sequence is based on a string of 4-bit bytes representing the operations, followed by the operand addresses. The leftmost 4-bit byte is the first operation; the operand addresses, if they exist, follow in the same order as the operation bytes, one word

DOCUMENT CLASS IMS PAGE NO. \_\_\_\_\_  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005 V2.0 MACHINE SERIES 1700

per byte. As many bytes may exist as desired, but the terminating byte must be 4, the operation FEND.

Example:



The O<sub>i</sub>'s are the operation codes; the A<sub>i</sub>'s are their operand addresses. An operand address is not needed by all operations, for instance, operation code O<sub>3</sub> does not have a corresponding A<sub>3</sub>.

#### 9.1.4 Operations

The following operations are used by the floating point package.

<u>Operation</u>	<u>4-bit Code</u>	<u>Meaning</u>
CHMD	5	Change of mode operation. All operand addresses following this operation code in the calling sequence are made relative if the preceding addresses were absolute, or absolute if the preceding addresses were relative. Addresses are initially absolute. No operand is needed.
INDX	F <sub>16</sub>	Index. The operand corresponding to INDX is used to increment the operand of the following operations: FLDD, FLST, FADD, FSUB, FMPY, and FDIV.

CONTROL DATA CORPORATION  
3000/1700 DEVELOPMENT

DIVISION

DOCUMENT CLASS. IMS PAGE NO. 9-4  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005 V2.0 MACHINE SERIES 1700

Each succeeding **INDX** supersedes the last. No index is initially assumed.

<b>NIDX</b>	6	No index. The succeeding operands do not have indexing increments. <b>NIDX</b> supersedes any preceding <b>INDX</b> , and is superseded by any following <b>INDX</b> . <b>NIDX</b> is assumed initially. No operand is needed.
<b>FLDD</b>	B <sub>16</sub>	Floating load. The floating point number in the corresponding effective operand address is transferred to the pseudo accumulator.
<b>FLST</b>	D <sub>16</sub>	Floating store. The floating point number is transferred from the pseudo accumulator to the corresponding effective operand address.
<b>FCOM</b>	7	Floating complement. The pseudo accumulator is complemented. No operand is needed.
<b>FADD</b>	E <sub>16</sub>	Floating add. The contents of the effective operand address is added to the pseudo accumulator, and the sum is left in the pseudo accumulator.
<b>FSUB</b>	8	Floating subtract. The contents of the effective operand address is subtracted from the pseudo accumulator, and the result is left in the pseudo accumulator.
<b>FMPY</b>	9	Floating multiply. The pseudo accumulator is multiplied by the contents of the effective operand address, and the result is left in the pseudo accumulator.
<b>FDIV</b>	A <sub>16</sub>	Floating divide. The pseudo accumulator is divided by the contents of the effective operand address, and the result is left in the pseudo accumulator.
<b>FEND</b>	4 {also 1, 2,3}	End of calling sequence. This operation terminates the calling sequence. No operand needed.

DOCUMENT CLASS IMS PAGE NO. 9-5  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005 V2.0 MACHINE SERIES 1700

9.1.5 Operands

All operand addresses, absolute or relative, are placed in bits 14-0. Bit 15 is a flag set to distinguish between direct and indirect addresses.

9.1.5.1 Absolute Addressing

An absolute address may be either direct or indirect. All indirect addressing is executed prior to indexing. Only one level of indirect addressing is allowed.

9.1.5.2 Relative Addressing

A relative address may be indirect to one level of indirectness. The relative address is computed by subtracting the location of the address in the calling sequence from the address contained in that location. For instance, if the relative address,  $A_1$ , is contained in location  $L+3$ , where  $L$  is the first word of the calling sequence, the relative address =  $A_1 - L + 3$ .

9.1.5.3 Example

Compute the floating point arithmetic statement:

$$X = -(A\{I\} + B\{I\}) * C\{I\} + D\{J\} * E\{J\}$$

Assuming that  $X$ ,  $J$ ,  $D$ ,  $E$ , and the temporary location TEMP are absolutely addressed, and the other operands relatively addressed, the generated calling sequence would be:

RTJ		FLOT	
F16	B16	9	6
absolute address J			
absolute address D			
absolute address E			
D16	5	F16	B16
absolute address TEMP			
relative address I			

DOCUMENT CLASS IMS  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005 V2.0 MACHINE SERIES 1700

relative address A			
E <sub>16</sub>	9	7	6
relative address B			
relative address C			
5	E <sub>16</sub>	D <sub>16</sub>	4
absolute address TEMP			
absolute address X			

9.1.6 Fault Conditions

Fault conditions are flagged in a communication location {#CB} whenever they occur during execution. The following bits are set to one:

<u>BIT</u>	<u>Fault Condition</u>
15	Exponent overflow; the exponent in an arithmetic operation exceeds the maximum range limit.
14	Divide fault; division by zero is encountered.
13	Exponent underflow; the exponent in an arithmetic operation is less than the minimum range limit.

If exponent underflow occurs, a floating point zero results. If exponent overflow occurs, the largest word of the appropriate sign results. A divide fault is treated as overflow. These error conditions may be tested with the IFALT function.

9.2 FLOATING-POINT ARITHMETIC WITH 23-BIT NUMBERS

9.2.1 Introduction

Consider the double precision floating-point number:

$$F = f \cdot 2^b \quad \{1\}$$

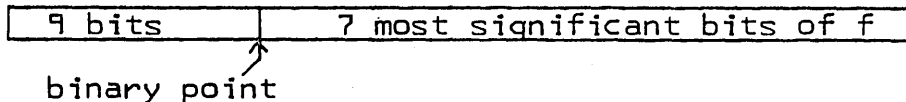
where |f| lies in the range



DOCUMENT CLASS IMS PAGE NO. 9-7  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. COO5 V2.0 MACHINE SERIES 1700

$$1/2 \leq |f| \leq 1 - 2^{-23} \quad \{2\}$$

We have a machine with a word length of 16 bits and that the 32 bits in the double length word are divided in the following way:



16 least significant bits of f

The leftmost block of nine bits is divided into three parts:

1. The first {leftmost} bit represents the sign of F.
  2. The second bit represents the sign of  $\beta$ .
  3. The next 7 bits represent the magnitude of  $\beta$ .
- This allows 23 bits for the representation of f. We shall assume that the binary point lies at the left and of the 23 bits representing f so that the 7 most significant bits of f are stored in the first word of the pair, and the 16 least significant bits of f are stored in the second word of the pair.

If we write

$$|f| = c + d \cdot 2^{-7} \quad \{3\}$$

where c lies in the range

$$1/2 \leq c \leq 1 - 2^{-7} \quad \{4\}$$

and where d lies in the range

$$1/2 \leq d \leq 1 - 2^{-16} \quad \{5\}$$

then c represents the 7 most significant bits of f and d represents the 16 least significant bits of f.

We wish to consider the four basic arithmetic operations using floating point numbers of the form discussed. Consequently, in order to have notation for two operands, let us consider a second floating point number.

$$G = g \cdot 2^s \quad \{6\}$$

where |g| lies in the range

$$1/2 \leq |g| \leq 1 - 2^{-23} \quad \{7\}$$

DOCUMENT CLASS IMS PAGE NO. 9-8  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005 V2.0 MACHINE SERIES 1700

If we write

$$|g| = a + b \cdot 2^{-7} \quad \{8\}$$

where a lies in the range

$$1/2 \leq a \leq 1 \quad -2^{-7} \quad \{9\}$$

where b lies in the range

$$1/2 \leq b \leq 1 \quad -2^{-16} \quad \{10\}$$

then a represents the 7 most significant bits of |g| and b represents the 16 least significant bits of |g|.

The machine represents negative numbers using a "one's-complement" system. In fact the procedure for changing the sign of a floating point number is to perform a bit-by-bit complement of the entire 32 bits (including the 9 bits representing the sign and exponent).

### 9.2.2 THE FOUR ARITHMETIC OPERATIONS

#### A. Multiplication

$$F \cdot G = \{f \cdot 2^B\} \{g \cdot 2^S\} \\ = \{\text{sign of } F \cdot G\} |f| \cdot |g| \cdot 2^{B+S} \quad \{11\}$$

The computational procedure is primarily concerned with the formation of  $|f| \cdot |g| \cdot 2^{B+S}$  since  $\{\text{sign } F \cdot G\}$  can be recorded in advance and used later to apply the correct sign to the product. In addition to recording  $\{\text{sign } F \cdot G\}$  we record the exponents  $B$  and  $S$  until after the product  $|f| \cdot |g|$  is formed. In fact, we propose the following algorithm for multiplying  $F$  by  $G$ .

1. Determine and record  $\{\text{sign } F \cdot G\}$
2. Form  $|F|$  and  $|G|$ .
3. Record the leftmost nine bits of  $|F|$  and  $|G|$ . This, in effect, records  $B$  and  $S$ .
4. Shift the 23 bits of  $|f|$  and  $|g|$  left until each has the bit pattern:

+	15 most significant bits	C
---	--------------------------	---

0	8 least significant bits   7 zeros	D
---	------------------------------------	---

If this procedure is followed,  $|f|$  is no longer represented by  $\{3\}$  during the computation is step 5, below, but has the form

DOCUMENT CLASS IMS PAGE NO. 9-9  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005 V2.0 MACHINE SERIES 1700

$$|f| = C + D \cdot 2^{-15} \quad \{12\}$$

where C lies in the range

$$1/2 \leq C \leq 1 \cdot 2^{-15} \quad \{13\}$$

where D lies in the range

$$0 \leq D \leq 1 \cdot 2^{-8} \quad \{14\}$$

likewise |g| has the form

$$|g| = A + B \cdot 2^{-15} \quad \{15\}$$

and where A lies in the range

$$1/2 \leq A \leq 1 \cdot 2^{-15} \quad \{16\}$$

and where B lies in the range

$$0 \leq B \leq 1 \cdot 2^{-8} \quad \{17\}$$

5. Use fixed point operations in forming the product

$$\begin{aligned} |f| \cdot |g| &= \{C+D \cdot 2^{-15}\} \{A+B \cdot 2^{-15}\} \\ &= CA + \{CB + DA\} \cdot 2^{-15} + DB \cdot 2^{-30} \\ &= CA + \{CB + DA\} \cdot 2^{-15} \quad \{18\} \end{aligned}$$

Notice that the term  $DB \cdot 2^{-30}$  may be ignored because once the product is placed back in standard form only 23 bits are retained. Notice, also, that {18} is written in such a way that it exhibits the efficiency of the following choice of computational steps:

- a. Form CA giving a double length product.
  - b. Form CB and retain the most significant half of the double length product.
  - c. Form DA.
  - d. Add the most significant half of DA to the most significant half of CB.
  - e. Add the least significant half of CA to the sum obtained in {d}. This result is the second half of the double length product. The first half of the double length product is the most significant half of CA which was formed above in a.
- b. Next, round and normalize the product obtained using {18} in Step 5. Any adjustment in the

DOCUMENT CLASS: IMS PAGE NO. 9-10  
 PRODUCT NAME: 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO.: C005 V2.0 MACHINE SERIES: 1700

exponent  $\beta + 5$  which is necessary because of the normalization of  $|f| \cdot |g|$  must be performed.

7. Finally, pack the 23 bits of the normalized product and the 9 bits representing the sign and the adjusted exponent into two 16 bit words. If {sign F.G} is negative, the two words must then be complemented to give the correct sign to the product.

B. Division

$$\frac{G}{F} = \frac{g \cdot 2^S}{f \cdot 2^\beta} \quad \{19\}$$

$$= \{\text{sign of } G \cdot F\} \left| \frac{g}{f} \right| \cdot 2^{S-\beta}$$

as a matter of fact, since we want

$$\left| \frac{g}{f} \right| < 1, \quad \{20\}$$

we scale the numerator and write

$$\frac{G}{F} = \{\text{sign } G \cdot F\} \left| \frac{g}{f} \right| \cdot 2^{S-\beta+1} \quad \{21\}$$

Thus we propose the following algorithm for dividing G by F.

1. Determine the record {sign F.G}
2. Form  $|F|$  and  $|G|$ .
3. Record the leftmost nine bits of  $|F|$  and  $|G|$ . this, in effect, records  $\beta$  and  $5$ .
4. Arrange the 23 bits of  $|f|$  to give the bit pattern:

+ | 15 most significant bits of |f| C

0 | 8 least significant bits | 7 zeros D

and the 23 bits of  $|g|$  to give the bit pattern:

+ | 0 | 14 most significant bits of |g| A

0 | 9 least significant bits | 6 zeros B

Thus,  $|f|$  is represented by {12}, {13}, and {14} as in the case of multiplication. However, in this case we have:

DOCUMENT CLASS IMS PAGE NO. 9-11  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005 V2.0 MACHINE SERIES 1700

$$\left| \frac{g}{2} \right| = A + B \cdot 2^{-15} \quad \{22\}$$

where A lies in the range

$$1/4 \leq A \leq 1/2 - 2^{-15} \quad \{23\}$$

and where B lies in the range

$$0 \leq B \leq 1 - 2^{-9} \quad \{24\}$$

5. Use fixed point operations in forming the quotient

$$\left| \frac{g}{2} \right| = \frac{A + B \cdot 2^{-15}}{C + D \cdot 2^{-15}}$$

$$\begin{aligned} &= \left[ \frac{A + B \cdot 2^{-15}}{C} \right] \left[ \frac{1}{1 + \frac{D \cdot 2^{-15}}{C}} \right] \\ &= \left[ \frac{A}{C} + \frac{B}{C} \cdot 2^{-15} \right] \left[ 1 - \frac{D}{C} \cdot 2^{-15} + \frac{D^2}{C^2} \cdot 2^{-30} \dots \right] \\ &= \frac{A}{C} + \frac{B}{C} \cdot 2^{-15} - \frac{AD}{C^2} \cdot 2^{-15} - \frac{BD}{C^2} \cdot 2^{-30} + \frac{AD^2}{C^3} \cdot 2^{-30} + \dots \\ &= \frac{A}{C} + \left[ \frac{B}{C} - \frac{AD}{C^2} \right] \cdot 2^{-15} + \left[ \frac{AD^2}{C^3} - \frac{BD}{C^2} \right] \cdot 2^{-30} + \dots \\ &= A + \left[ \frac{B}{C} - \frac{AD}{C} \right] \cdot 2^{-15} \quad \{25\} \end{aligned}$$

Notice that the terms beginning with

$$\left[ \frac{AD^2}{C^3} - \frac{BD}{C^2} \right] \cdot 2^{-30}$$

may be ignored because only 23 bits of the quotient are retained. Notice, also, that {25} is written in such a way that it exhibits the efficiency of the following choice of computational steps:

- Form  $-AD$  giving a double length product.
- Divide  $-AD$  {as a double length dividend} by  $C$ .
- Form  $B - \frac{AD}{C}$  {rounded to single length}.

DOCUMENT CLASS IMS  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005 V2.0  
 PAGE NO. 9-17  
 MACHINE SERIES 1700

- d. Form the double length dividend

$$A + \left[ B - \frac{AD}{C} \right] \cdot 2^{-15}$$

{The sign of the second term makes this tricky.}

- e. Divide this double length dividend by C.  
 f. To obtain the second half of the double length quotient, the remainder resulting from the division in the previous step must not be divided by C.

The class of machines for which this procedure is efficient must have the feature that fixed point multiplication leaves a double length product which can be used as a double length dividend for fixed point division.

- b. Next, round and normalize the quotient obtained using {25} and the procedure of step 5. Any adjustment in the exponent which is necessary because of the normalization of  $\left| \frac{q}{f} \right|$  must be performed.

7. Finally, pack the 23 bits of the normalized quotient and the 9 bits representing the sign and the adjusted exponent into two 16 bit words. If {sign F.G} is negative, the two words must then be complemented to give the correct sign to the quotient.

C. Addition

$$F + G = f \cdot 2^B + g \cdot 2^S \quad \{26\}$$

The basic problem in floating point addition is to adjust the exponent of F {or G} so that the binary points are lined up before the addition takes place.

Let us introduce the notation L to represent a pair of cells containing the larger of the two numbers F and G, and the notation M to represent a pair of cells containing the smaller of the two numbers. We shall say that F is larger than G if

$$B \geq S \quad \{27\}$$

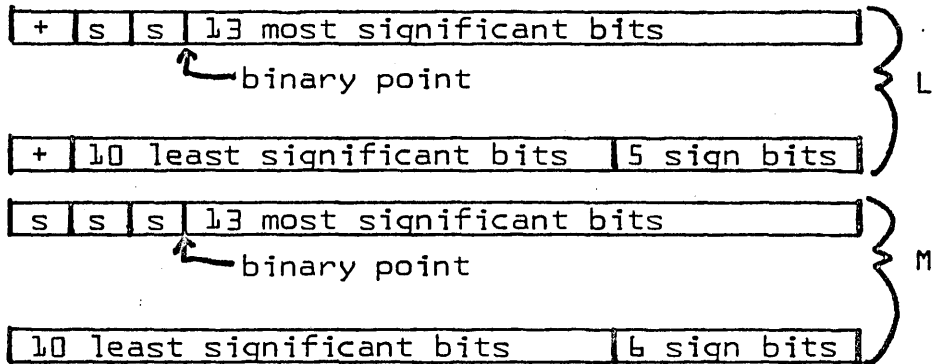
and F is smaller than G if

$$B < S \quad \{28\}$$

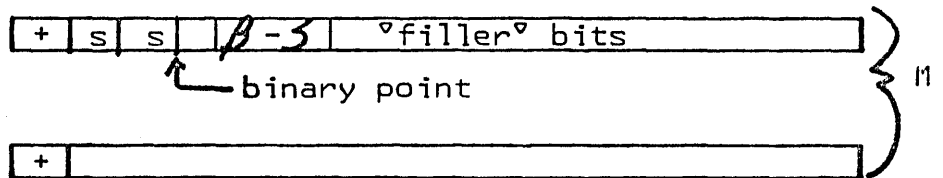
DOCUMENT CLASS IMS PAGE NO. \_\_\_\_\_  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005 V2.0 MACHINE SERIES 1700

We are not concerned about the relative magnitudes of  $f$  and  $g$  in case the exponents are equal. Using this convention, we propose the following algorithm for forming  $F + G$ .

1. Record the leftmost nine bits of  $F$  and  $G$ . This, in effect, records  $\beta$  and  $5$ .
2. Determine the sign of  $\{\beta - 5\}$  and thus determine whether  $F$  is smaller or larger than  $G$  according to  $\{27\}$  &  $\{28\}$ .
3. Place  $f$  and  $g$  in  $L$  and  $M$ . If  $F$  is larger than  $G$ , then  $f$  goes to  $L$ ; otherwise  $f$  goes into  $M$  and  $g$  goes into  $L$ . The following bit patterns should be formed (here  $s$  means "sign bit"):

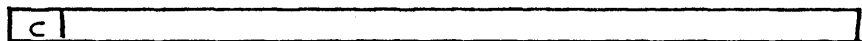


4. Shift  $M$  right  $|\beta - 5|$  places and put a "0" bit at the beginning of each of the two words. If  $|\beta - 5| \geq 23$  then there is no need to continue since all significant bits in  $M$  will be lost.



Notice that the  $|\beta - 5|$  "filler" bits, between the binary point in  $M$  and the most significant bit of the fraction, are sign bits. This is mathematically correct in a one's complement representation of negative numbers.

5. Add the second halves of  $L$  and  $M$ .



We call the first bit of this sum  $c$ . If it is a one we actually have a carry. If it is a zero we do not

DOCUMENT CLASS IMS PAGE NO. 9-14  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005 V2.0 MACHINE SERIES 1700

have a carry. However, it usually is easier to add  $c$  (see step 6) than to test to see whether or not it needs to be added as a carry bit in forming the sum of the first halves of  $L$  and  $M$ .

- b. Add the first halves of  $L$  and  $M$  and add the carry bit obtained from step 5.

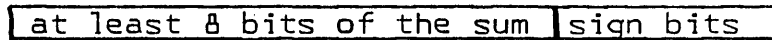
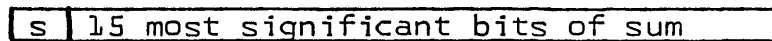


If  $e = 1$ , then an end-around-carry must be performed. This means that a one is added at the right end of the word produced in step 5. Since this might also produce a carry bit, the  $c$  in the diagram (see step 5) must be cleared to zero before the end-around-carry. If a carry bit is again produced, then a one must be added at the right end of the word above. It can be shown that this last operation can never produce another  $e = 1$ .

If  $v = s$ , then  $v$  is a sign bit. However, if  $v \neq s$  then there has been overflow during the addition, and  $v$  is the most significant bit of the sum. In the latter case an adjustment of the exponent will be necessary to give the correct answer.

7. Shift of second half of the sum left one place to clear out the carry bit  $c$ . Then shift the double length sum left (a) one place if  $v \neq s$ ; (b) two places if  $v = s$ .

This leaves the sum in the following form:



8. If the double length sum was shifted one place left in step 7 ( $v \neq s$ ) then the exponent must be adjusted to take care of the overflow. This means adding one to the exponent  $\beta$  or  $\gamma$ , whichever is larger. (This will be the exponent of the sum.) If the double length sum was shifted two places left in step 7, no adjustment of exponent is necessary.
9. The form of the sum given by step 8 must be checked for normalization since it is possible that several of the leading bits of the sum may be zero. (Cancell-



DOCUMENT CLASS IMS PAGE NO. 9-15  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005 V2.0 MACHINE SERIES 1700

ation occurs when two numbers of opposite sign but nearly equal magnitude are added.} If the sum is not normalized then it should be normalized at this point and appropriate adjustments in the exponent should be made. If 23 left shifts are not sufficient for normalization then the sum should be made zero.

10. At this point the normalized sum may be rounded although the extra coding involved may not be worth the gain. If rounding is desired then there are two cases to be considered depending on the sign of the sum. These cases require that care be taken in handling any carry bit produced by the rounding operation.
11. Now pack the 23 most significant bits of the sum, along with 9 bits representing the sign and exponent, into two 16 bit words. If the sign of the sum is negative, then the first 9 bits must be complemented before the packing takes place.

#### D. Subtraction

No special subroutine is necessary since

$$F - G = F + \{-G\}$$

and one merely complements G before entering the addition subroutine.

#### REFERENCES:

Gregory, Robert T. and Raney, James L.; Floating Point Arithmetic With 84-bit Numbers, Communications of the ACM, Volume 1/ Number 1/ January 1964.

#### 9.3 Low Core Storage Location

FLOAT uses the low core location #C5 through #D7; they are assigned as follows:

CONTROL DATA CORPORATION  
3000/1700 DEVELOPMENT

DIVISION

DOCUMENT CLASS IMS PAGE NO. 9-16  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005 V2.0 MACHINE SERIES 1700

Location Name	Location	Use
SIGN	‡C7	Sign register-sign of floating point number.
ERRORS	‡C8	Error bits to indicate exponent overflow, divide fault, exponent underflow-which are used by IFALT to check for these floating point errors.
SHIFCT	‡D1	Shift counter.
P	‡D2	Pseudo program counter which is used by FLOAT to know where it is in the user's program when it wants to pick up more parameters or return.
INDEX	‡D3	Effective address of parameter
RELADR	‡D4	Absolute/relative indication RELADR { =1 relative address =0 absolute address
OPCNT	‡D5	Op counter - keeps track of which op code is being processed.
OPCODE	‡D6	Op code holder - holds op codes in the call.
QS	‡D7	Saves Q register upon entry into FLOAT.
G G+1	‡C5 ‡C6	Pseudo accumulator - the two words of the floating point number are placed here. The pseudo accumulator is broken up into C, D and DELTA.
C	‡CD	Most significant bits of the coefficient of the floating point number.
D	‡CE	Least significant bits of coefficient of floating point number.

DOCUMENT CLASS IMS PAGE NO. 9-17  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005 V2.0 MACHINE SERIES 1700

Location Name	Location	Use
DELTA	#D0	Exponent of floating point number.
F	#C9	1. Address of parameter or 2. First word of a {two-word} floating point number - used in the same manner as G of the pseudo accumulator is used when two operands are needed, e.g. for an add. In this case, F and F+1 are broken up into A, B and BETA.
F+1	#CA	The second word of a {two-word} floating point number.
A	#CB	The most significant bits of the coefficient of the floating point number.
B	#CC	The least significant bits of the coefficient of the floating point number.
BETA	#CF	Exponent of floating point number.

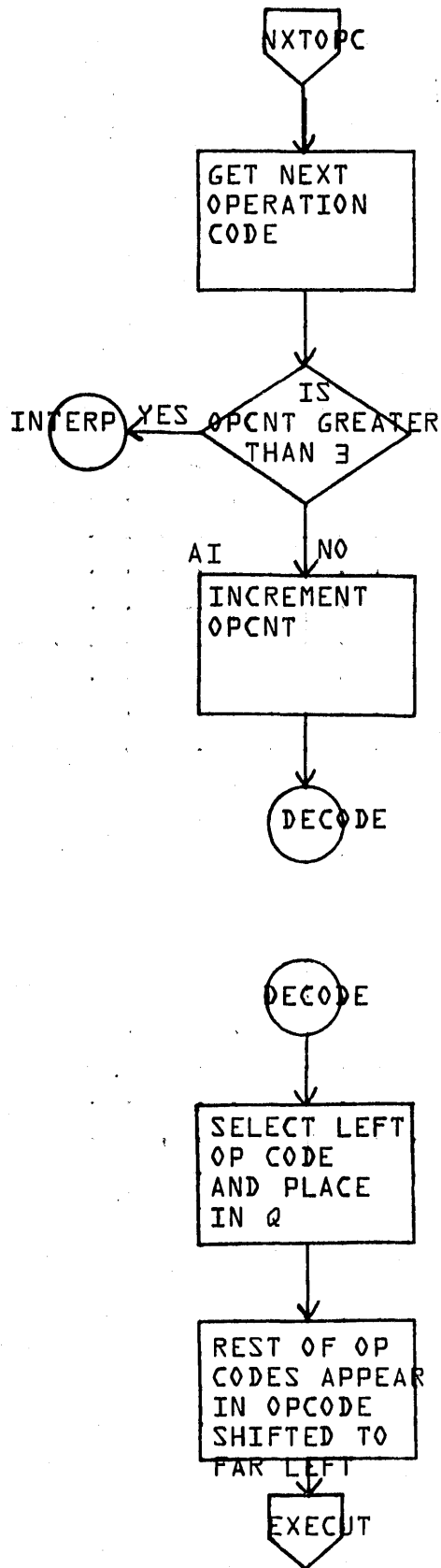
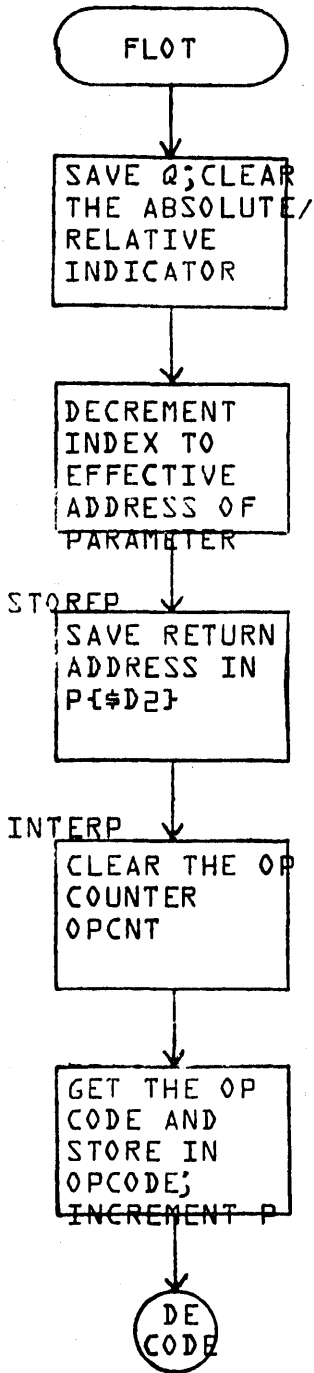
#### 9.4 Entry Points and Externals

The routine FLOAT has one entry point, FLOT, and no externals.

#### 9.5 Flowchart of FLOAT.

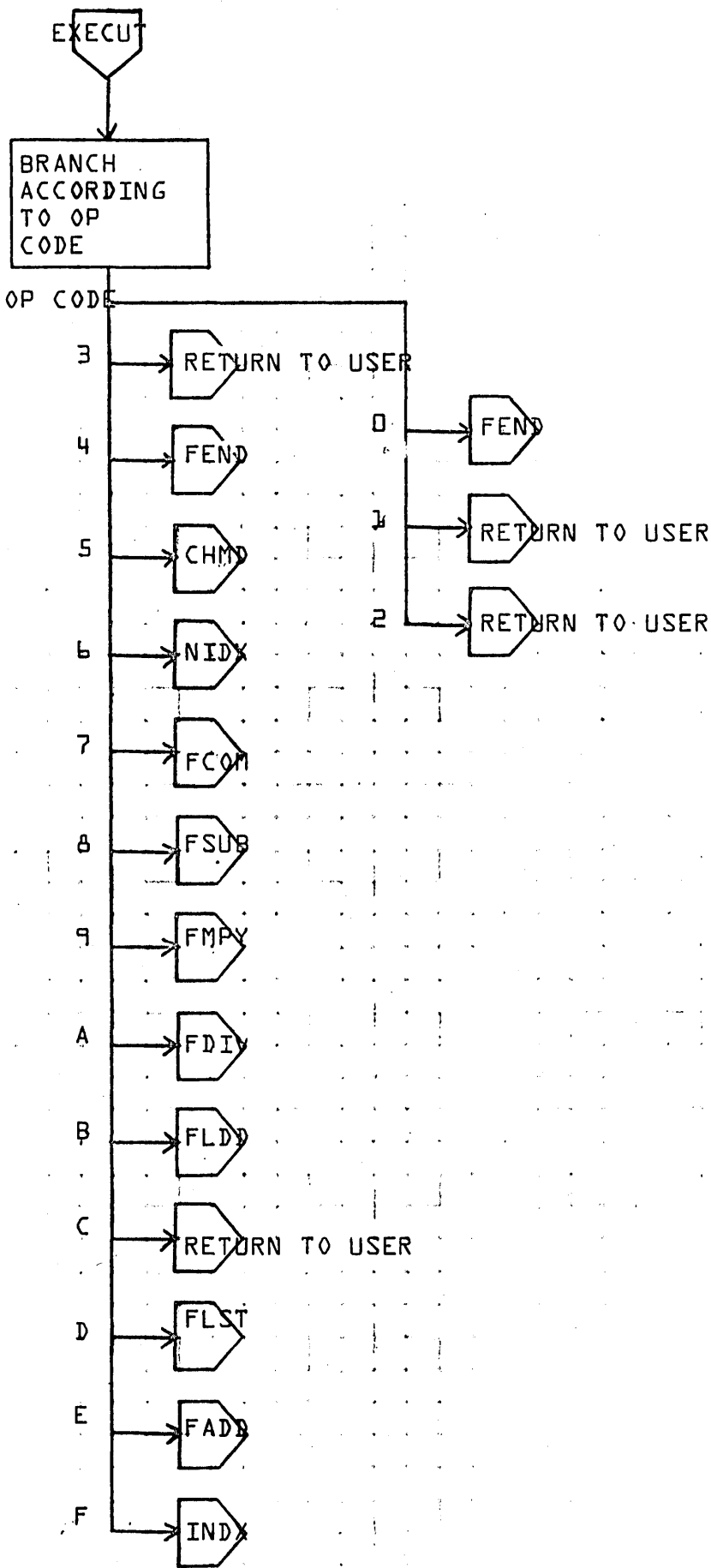
The following abbreviations are used:

MSB = most significant bits  
 LSB = least significant bits



CONTROL DATA CORPORATION		DATE
SOFTWARE DOCUMENT	APPROVED	REV
SAMPLE CODE	PROJECT NO.	PROJECT MGR.
FLOWCHART	PROJECT NAME	TASK NO.
DECISION TABLE	TASK NAME	
OTHER		
DOCUMENT CLASS	MACH. TYPE	DATE
IMS	1700	
DOCUMENT TITLE	PAGE	ISSUE DATE
FLOAT	187	
NUMBER	DRAWN BY	

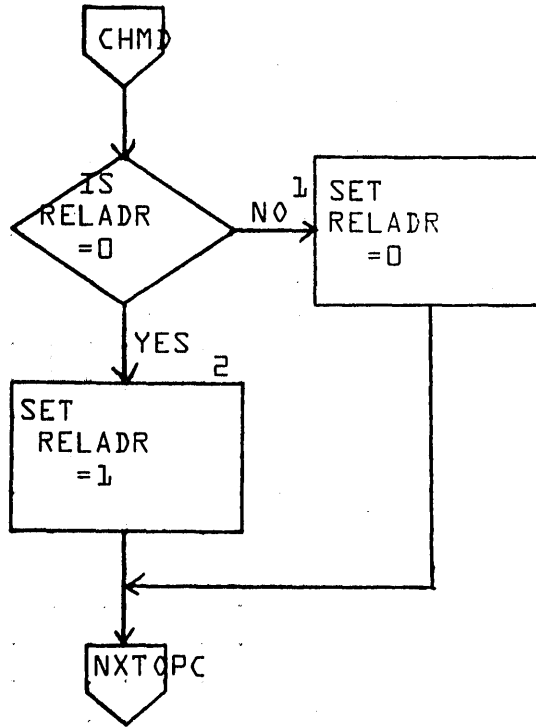
PRINTED IN USA



CONTROL DATA CORPORATION SOFTWARE DOCUMENT	DOCUMENT CLASS IMS	MACH. TYPE 1700	PROJECT NO.	REV	APPROVED	DATE
SAMPLE CODE	DOCUMENT TITLE FLOAT	PAGE 2 of 39	PROJECT MGR.			
FLOWCHART	NUMBER	ISSUE DATE	PROJECT NAME			
DECISION TABLE	DRAWN BY	DATE	TASK NO.			
OTHER			TASK NAME			

OPERATION CODE : 5  
 CHANGL MODE OF OPERATION [CHMD]

- 1. RELATIVE ADDRESS
- 2. ABSOLUTE ADDRESS



CONTROL DATA CORPORATION SOFTWARE DOCUMENT		DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.		REV		APPROVED		DATE	
SAMPLE CODE: <input type="checkbox"/>		DOCUMENT TITLE	FLOAT	PAGE 3 OF 39		PROJECT MGR.							
FLOWCHART <input checked="" type="checkbox"/>		NUMBER		ISSUE DATE		PROJECT NAME							
DECISION TABLE <input type="checkbox"/>		DRAWN BY		DATE		TASK NO.							
OTHER <input type="checkbox"/>						TASK NAME							

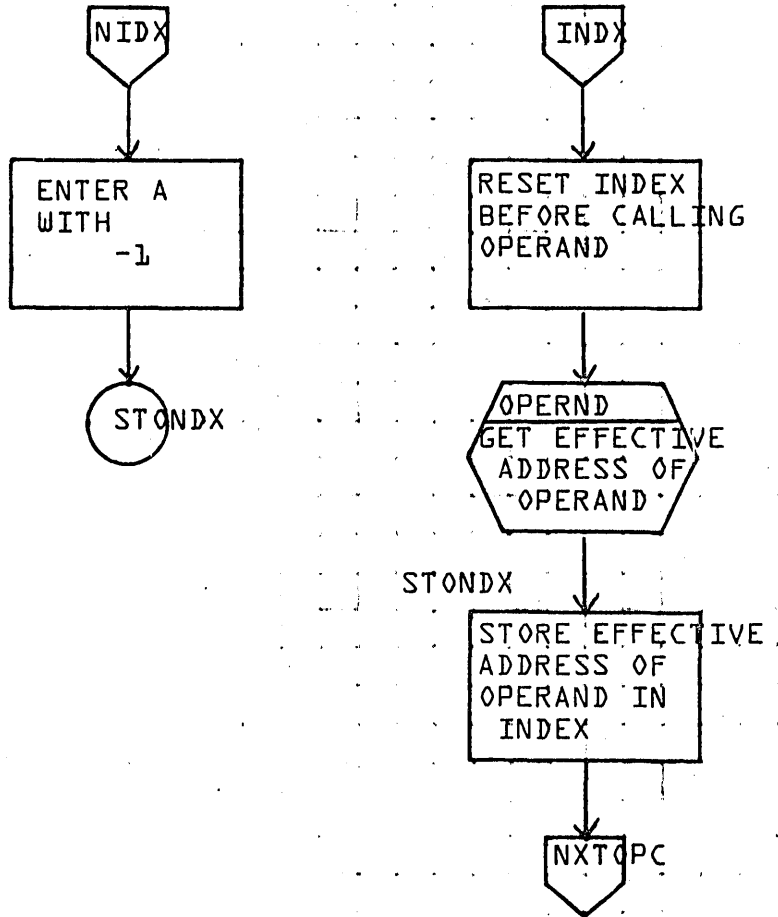
PRINTED IN USA

OPERATION CODE = b

NO INDEX {NIDX}

OPERATION CODE = F

INDEX {INDX}

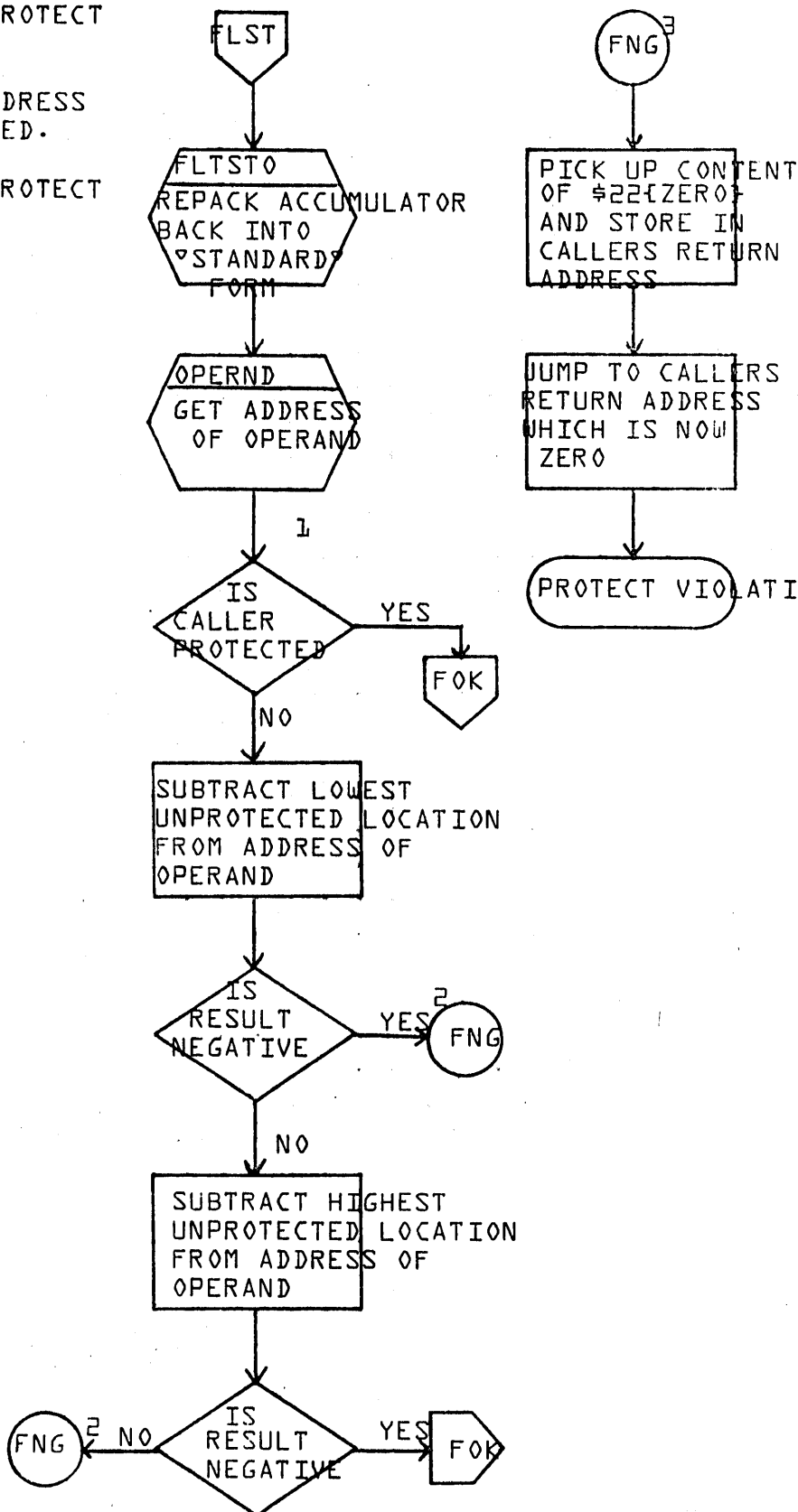


<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT		MACH. TYPE IMS	PROJECT NO. 1700	DATE
SAMPLE CODE	NUMBER 4	DOCUMENT TITLE FLOAT	PROJECT MGR. 039	APPROVED
FLOWCHART	ISSUE DATE	PROJECT NAME	TASK NO.	REV
DECISION TABLE	DATE	TASK NAME	TASK NAME	DATE
OTHER	DRAWN BY	TASK NAME	TASK NAME	DATE

OPERATION CODE = D

STORE ACCUMULATOR {FLST}

1. TEST FOR PROTECT VIOLATION.
2. STORAGE ADDRESS IS PROTECTED.
3. SIMULATE PROTECT VIOLATION.



DATE	
APPROVED	
REV	
PROJECT NO.	
PROJECT MGR.	
PROJECT NAME	
TASK NO.	
TASK NAME	
MACH. TYPE	1700
DOCUMENT CLASS	IMS
DOCUMENT TITLE	FLOAT
ISSUE DATE	PAGE 5 OF 34
NUMBER	
DRAWN BY	
DATE	

CONTROL DATA CORPORATION  
 SOFTWARE DOCUMENT

SAMPLE CODE

FLOWCHART

DECISION TABLE

OTHER

PRINTED IN U.S.A.



FOK

STORE 1ST WORD  
OF FLOATING POINT  
NUMBER IN CALLER'S  
RETURN ADDRESS +1

STORE 2ND WORD  
OF FLOATING POINT  
NUMBER IN CALLER'S  
RETURN ADDRESS +2

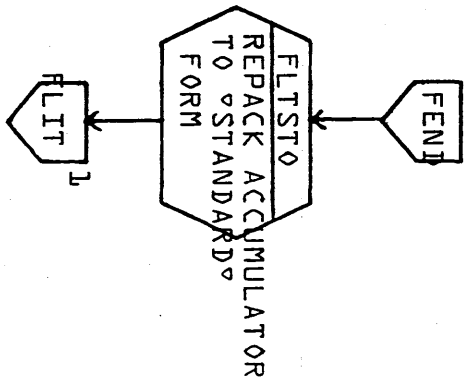
ANXT

NXT PC

CONTROL DATA CORPORATION SOFTWARE DOCUMENT		DOCUMENT CLASS IMS	MACH. TYPE 1700	PROJECT NO.	REV	APPROVED	DATE
SAMPLE CODE <input type="checkbox"/>		DOCUMENT TITLE FLOAT	ISSUE DATE	PROJECT MGR.			
FLOWCHART <input checked="" type="checkbox"/>		NUMBER	PAGE 6 OF 39	PROJECT NAME			
DECISION TABLE <input type="checkbox"/>		DRAWN BY	DATE	TASK NO.			
OTHER <input type="checkbox"/>				TASK NAME			

OPERATION CODE 11,4  
 END OF CALLING SEQUENCE (FEND)

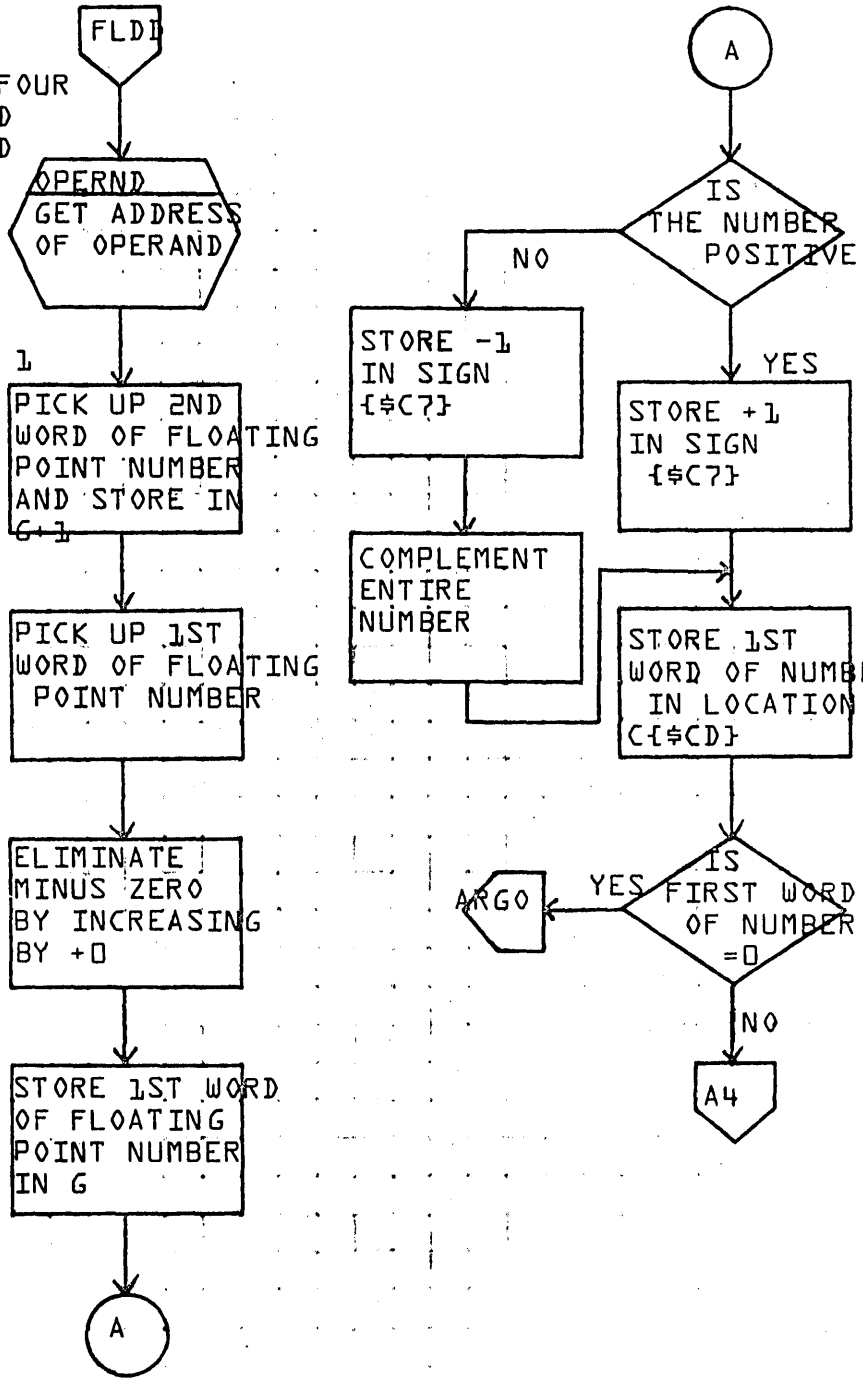
J. RESTORE Q AND  
 RETURN TO USER.



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	FLOAT	PAGE 7 OF 39		PROJECT MGR.			
	NUMBER	ISSUE DATE			PROJECT NAME			
	DRAWN BY	DATE			TASK NO.			
					TASK NAME			

OPERATION CODE = B  
 FLOATING LOAD {FLDD}

1. FOLLOWING FOUR  
 BOXES LOAD  
 THE OPERAND  
 INTO G  
 AND G+1.



DATE	
APPROVED	
REV	
PROJECT NO.	1700
PROJECT MGR.	
PROJECT NAME	
TASK NO.	
TASK NAME	
MACH. TYPE	FLOAT
PAGE NO. OF FIG.	39
ISSUE DATE	
NUMBER	
DRAWN BY	
DATE	
DOCUMENT CLASS	IMS
DOCUMENT TITLE	FLOAT

CONTROL DATA CORPORATION  
 SOFTWARE DOCUMENT

SAMPLE CODE

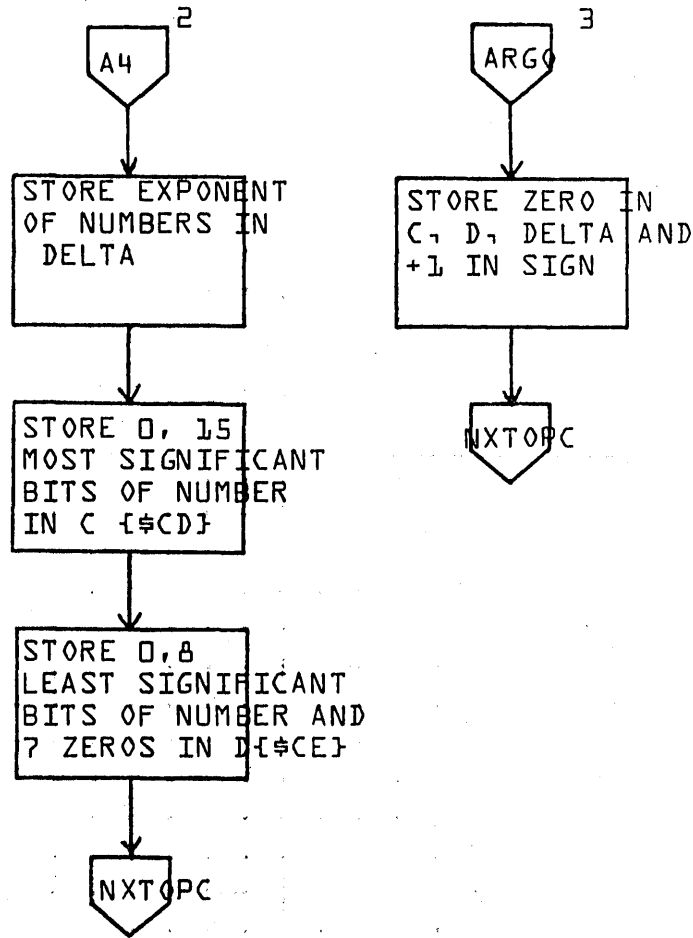
FLOWCHART

DECISION TABLE

OTHER

2. UNPACK  
OPERAND INTO  
C, D, AND DELTA.

3. FLOATING  
POINT NUMBER  
IS ASSUMED TO  
BE ZERO SINCE  
FIRST WORD IS  
ZERO

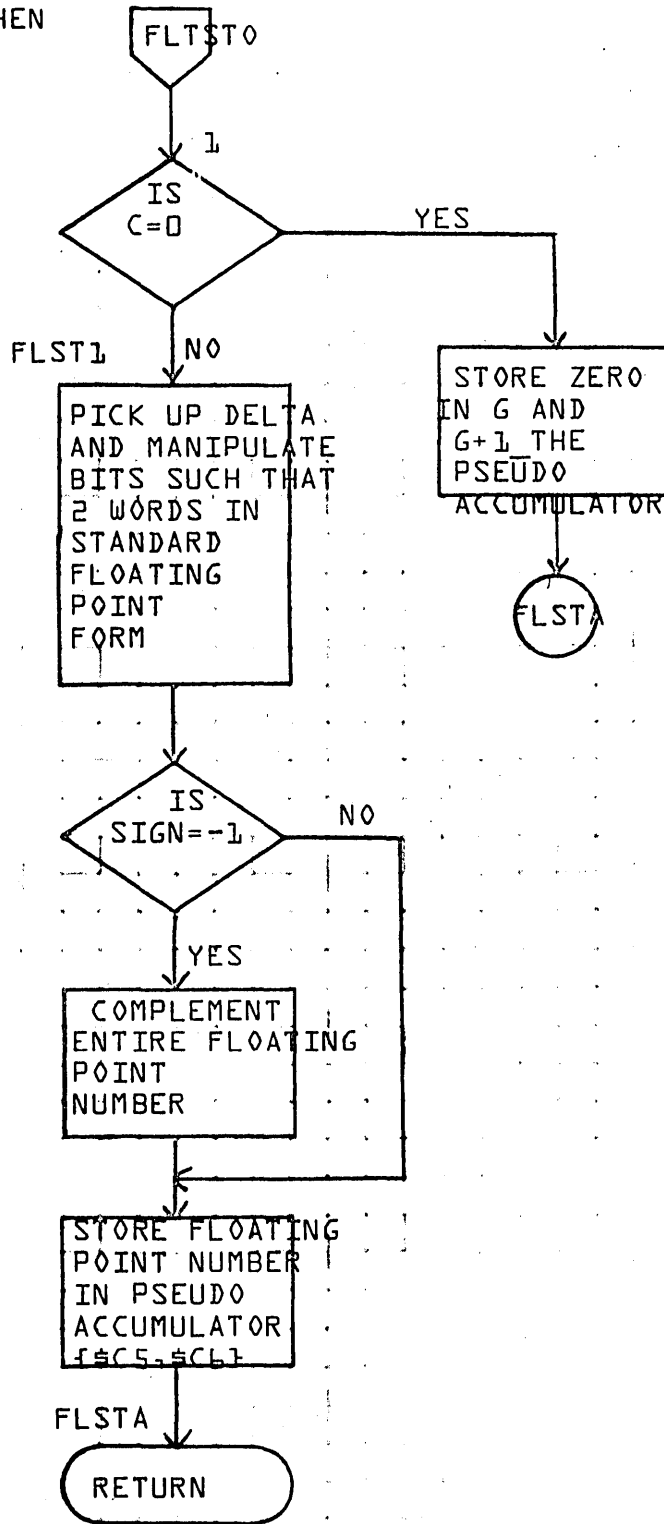


CONTROL DATA CORPORATION		DATE
SOFTWARE DOCUMENT	APPROVED	DATE
SAMPLE CODE <input type="checkbox"/>	REV	
FLOWCHART <input checked="" type="checkbox"/>	PROJECT NO.	
DECISION TABLE <input type="checkbox"/>	PROJECT MGR.	
OTHER <input type="checkbox"/>	PROJECT NAME	
	TASK NO.	
	TASK NAME	
DOCUMENT CLASS	MACH. TYPE	1700
DOCUMENT TITLE	ISSUE DATE	PAGE 4 OF 39
NUMBER	DATE	
DRAWN BY	DATE	

PRINTED IN U.S.A.

REPACKING ACCUMULATOR

1. IF C IS ZERO, THEN THE PSEUDO ACCUMULATOR IS ZERO

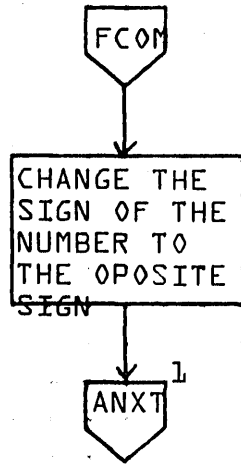


APPROVED		DATE	
REV	PROJECT NO.	PROJECT MGR.	PROJECT NAME
	PROJECT NO.	PROJECT MGR.	PROJECT NAME
	ISSUE DATE	PAGE 106	39
NUMBER	ISSUE DATE	TASK NO.	TASK NAME
DRAWN BY	DATE		
CONTROL DATA CORPORATION SOFTWARE DOCUMENT SAMPLE CODE <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> FLOWCHART DECISION TABLE OTHER			

OPERATION CODE = 7

FLOATING COMPLEMENT {FCOM}

- 1. A JUMP TO THE LABEL ANXT TAKES YOU TO A JUMP TO THE LABEL NXTOPC-NEXT OP CODE

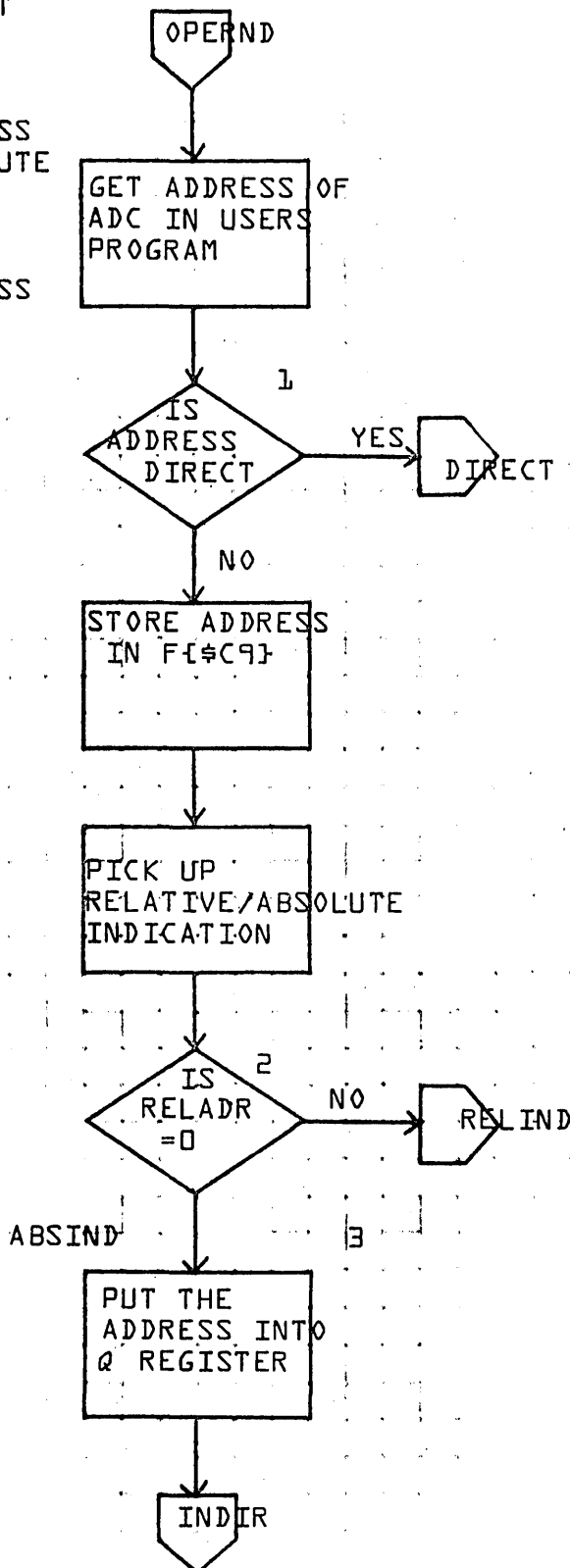


CONTROL DATA CORPORATION SOFTWARE DOCUMENT		DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.		REV		APPROVED		DATE	
SAMPLE CODE		DOCUMENT TITLE	1700	PROJECT MGR.		PROJECT NAME							
FLOWCHART			1700	PAGE	1700	TASK NO.							
DECISION TABLE		NUMBER		ISSUE DATE		TASK NAME							
OTHER		DRAWN BY		DATE									

PRINTED IN USA

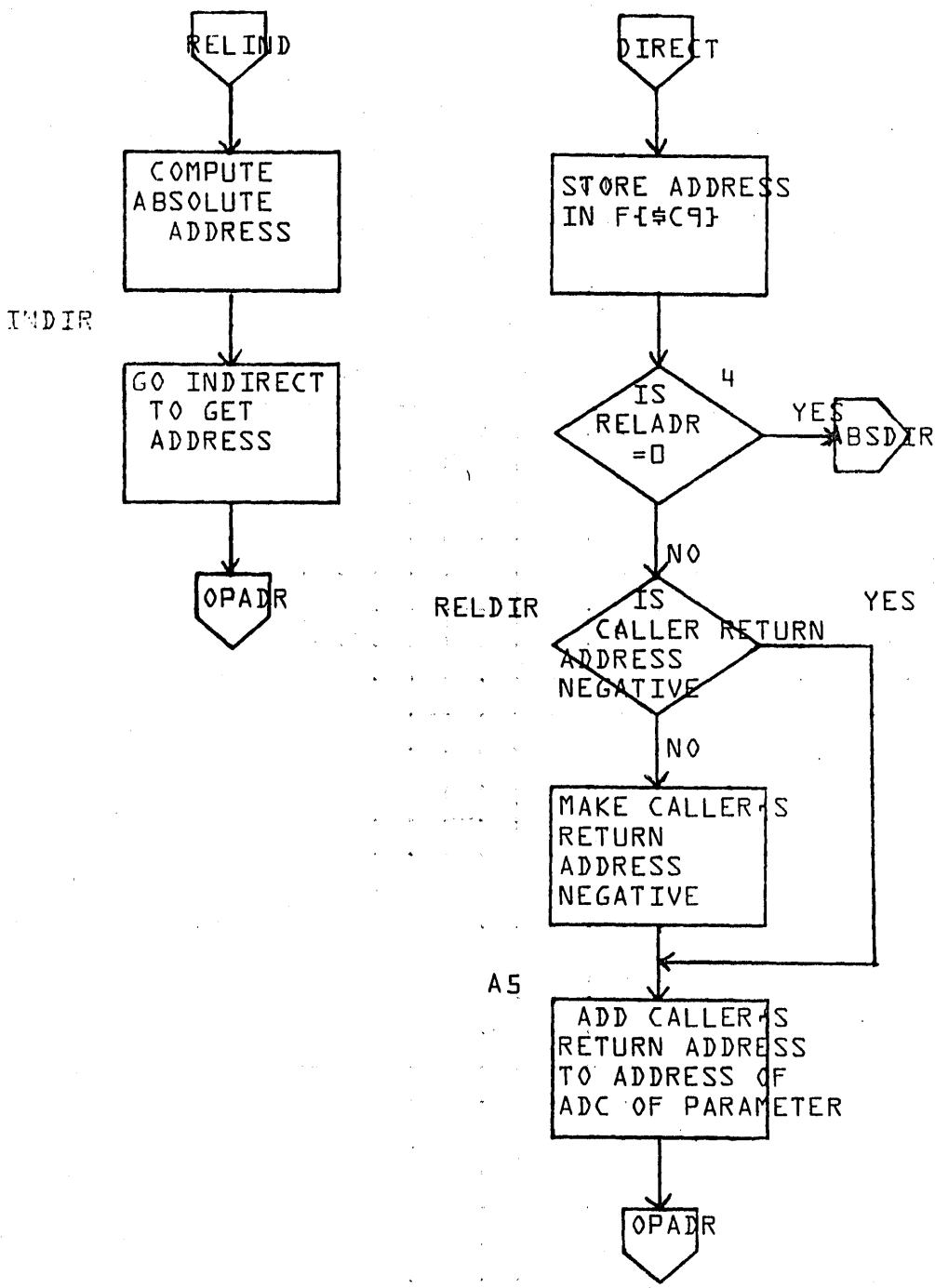
GET EFFECTIVE ADDRESS OF NEXT OPERAND

1. ADDRESS IS DIRECT  
IF INDIRECT BIT  
{BIT 15} IS NOT  
SET.
2. RELADR=1 IF  
RELATIVE ADDRESS  
AND=0 IF ABSOLUTE  
ADDRESS.
3. ABSOLUTE AND  
INDIRECT ADDRESS



CONTROL DATA CORPORATION SOFTWARE DOCUMENT	DOCUMENT CLASS IMS	MACH. TYPE 1700	PROJECT NO.	REV	APPROVED	DATE
SAMPLE CODE FLOWCHART	DOCUMENT TITLE FLOAT	PAGE 12 OF 39	PROJECT MGR.			
DECISION TABLE	NUMBER	ISSUE DATE	PROJECT NAME			
OTHER	DRAWN BY	DATE	TASK NO.			
			TASK NAME			

4. IS ADDRESS ABSOLUTE?



DATE		APPROVED		REV	
PROJECT NO.	1700	PROJECT MGR.			
PROJECT NAME		PAGE	13	OF	34
TASK NO.		ISSUE DATE			
TASK NAME		DATE			
DOCUMENT CLASS	IMS	MACH. TYPE	FLOAT		
DOCUMENT TITLE					
NUMBER					
DRAWN BY					

CONTROL DATA CORPORATION  
SOFTWARE DOCUMENT

SAMPLE CODE

FLOWCHART

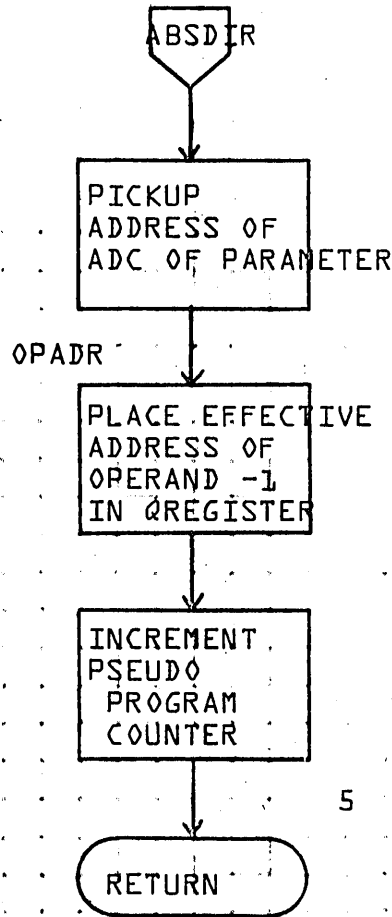
DECISION TABLE

OTHER

PRINTED IN USA



5. RETURN WITH EFFECTIVE ADDRESS OF OPERAND-1 IN QREGISTER.



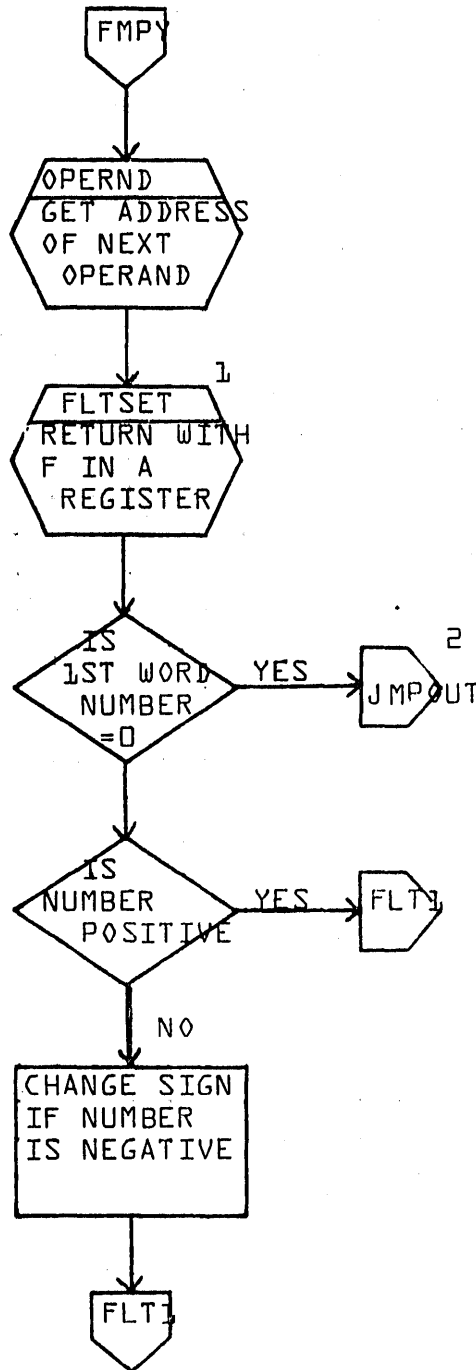
CONTROL DATA CORPORATION		DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.		REV		APPROVED	DATE
SOFTWARE DOCUMENT		DOCUMENT TITLE	FLOAT			PROJECT MGR.					
SAMPLE CODE		NUMBER		PAGE	119F	PROJECT NAME					
FLOWCHART		DRAWN BY		ISSUE DATE		TASK NO.					
DECISION TABLE				DATE		TASK NAME					
OTHER											

OPERATION CODE = 9

FLOATING MULTIPLY {FMPY}

1. GETS THE FLOATING POINT NUMBER AND STORES IN F AND F+1; THEN UNPACKS THE NUMBER INTO  
 A=0, 15MSB  
 B=0, 8LSB, 7 ZEROS  
 BETA=EXPONENT

2. A JUMP TO JMPOUT CAUSES A JUMP TO ARG0.



DATE		APPROVED	REV
PROJECT NO.		PROJECT MGR.	PROJECT NAME
MACH. TYPE 1700		PAGE 5 OF 37	TASK NO.
DOCUMENT CLASS IMS	DOCUMENT TITLE FLOAT	ISSUE DATE	TASK NAME
NUMBER	DRAWN BY	DATE	

CONTROL DATA CORPORATION  
 SOFTWARE DOCUMENT

SAMPLE CODE

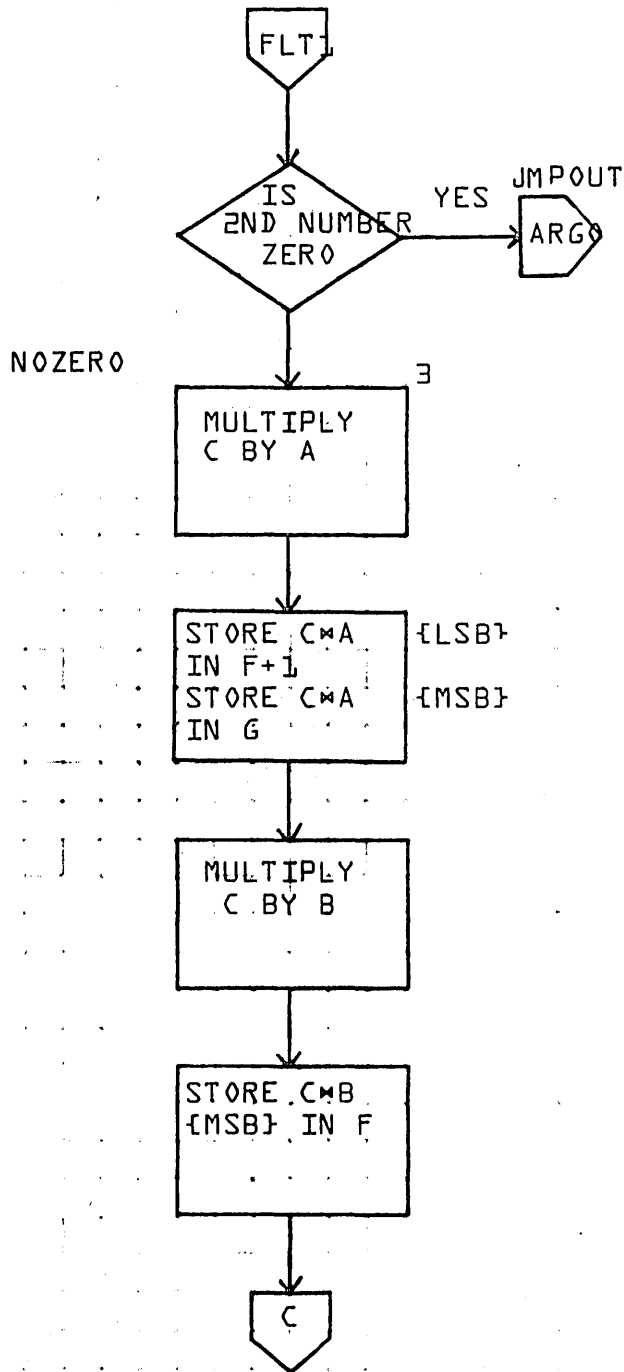
FLOWCHART

DECISION TABLE

OTHER

PRINTED IN USA

3. MULTIPLY THE MSB OF ONE NUMBER BY THE MSB OF THE OTHER NUMBER



CONTROL DATA CORPORATION SOFTWARE DOCUMENT	DOCUMENT CLASS IMS	MACH. TYPE 1700	PROJECT NO.	REV	APPROVED	DATE
SAMPLE CODE	DOCUMENT TITLE FLOAT	ISSUE DATE	PROJECT MGR.			
FLOWCHART	PAGE 16 OF 39		PROJECT NAME			
DECISION TABLE			TASK NO.			
OTHER			TASK NAME			

4. UPON ENTRY THE A AND Q REGISTERS ARE:

$$A = \{A \times D\{MSB\}\} + \{C \times B\{MSB\}\}$$

$$Q = C \times A\{MSB\}$$

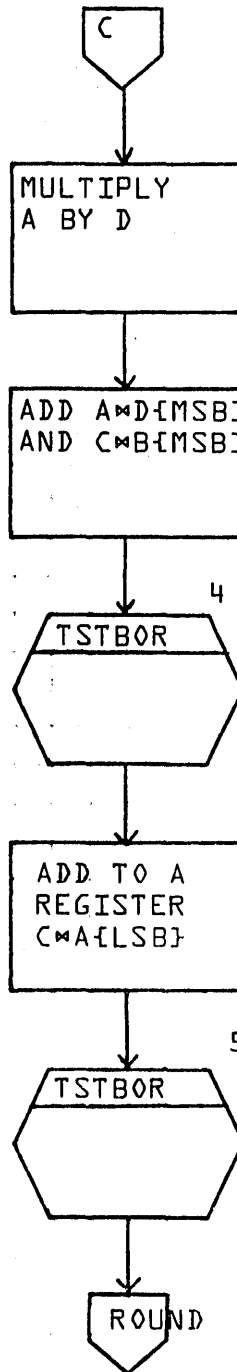
IN THIS SECTION OF CODE, IF A IS NEGATIVE, CARRY INTO G; IF Q IS NEGATIVE, END AROUND.

5. SAME AS #4 ABOVE EXCEPT THAT

$$A = A \times D\{MSB\}$$

$$+ C \times B\{MSB\}$$

$$+ C \times A\{LSB\}$$



CONTROL DATA CORPORATION		DATE
SOFTWARE DOCUMENT		APPROVED
DOCUMENT CLASS	MACH. TYPE	REV
IMS	1700	
DOCUMENT TITLE	PROJECT NO.	
FLOAT		
NUMBER	PROJECT MGR.	
	PROJECT NAME	
DRAWN BY	TASK NO.	
	TASK NAME	
	ISSUE DATE	
	DATE	

P-34

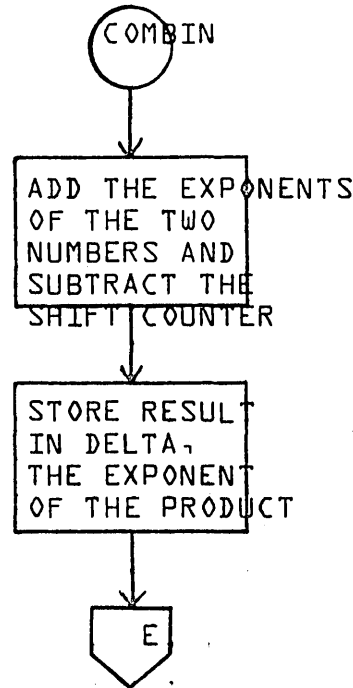
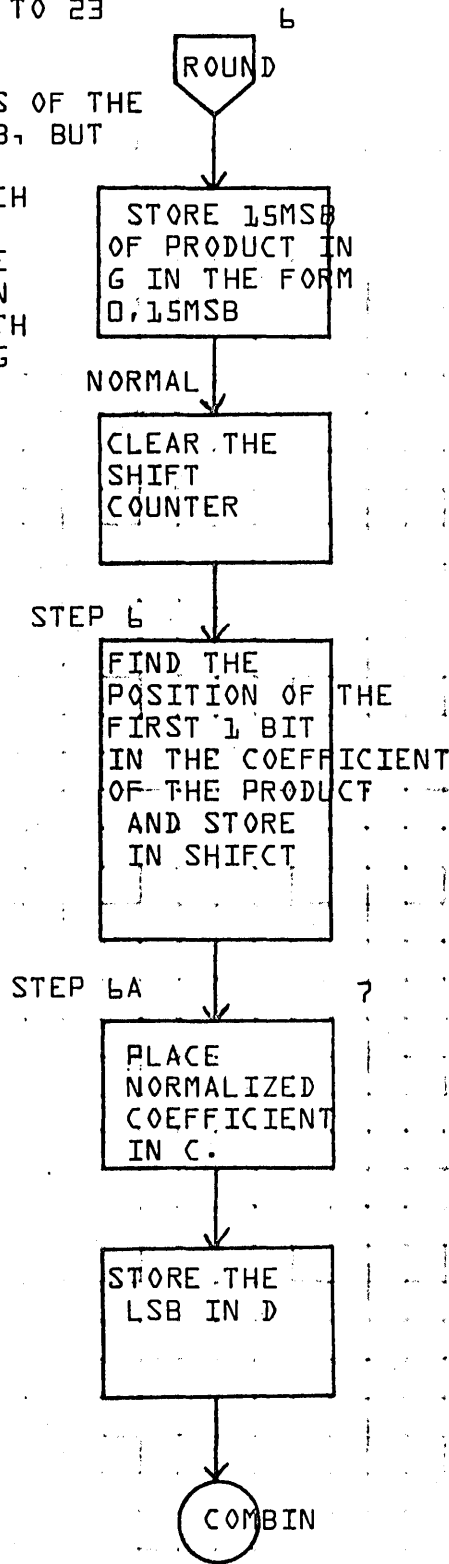
PRINTED IN USA

AA1365 (FORMERLY CA127-1)

A B C D

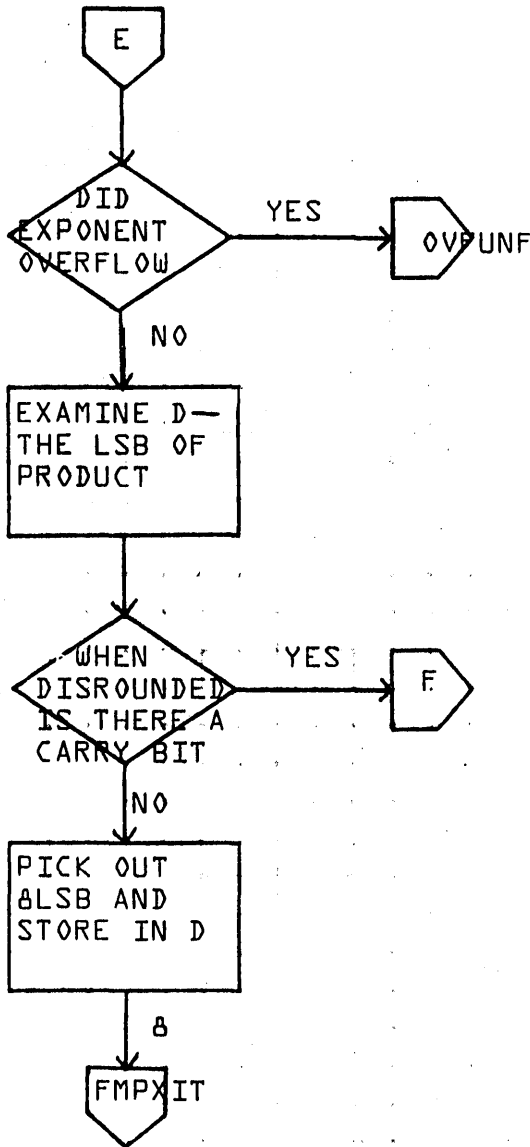
6. THIS SECTION ROUNDS AND TRUNCATES TO 23 BITS.

7. COEFFICIENT IS OF THE FORM  $D \cdot 15MSB$ , BUT THE 15MSB ARE NORMALIZED SUCH THAT A ONE APPEARS IN THE FIRST LOCATION OF THE MSB WITH SHIFCT KEEPING TRACK OF HOW MANY PLACES HAVE BEEN SHIFTED.



CONTROL DATA CORPORATION SOFTWARE DOCUMENT		DATE	APPROVED	REV
DOCUMENT CLASS	MACH. TYPE	PROJECT NO.		
DOCUMENT TITLE	ISSUE DATE	PROJECT MGR.		
SAMPLE CODE	DATE	PROJECT NAME		
FLOWCHART	ISSUE DATE	TASK NO.		
DECISION TABLE	DATE	TASK NAME		
OTHER				

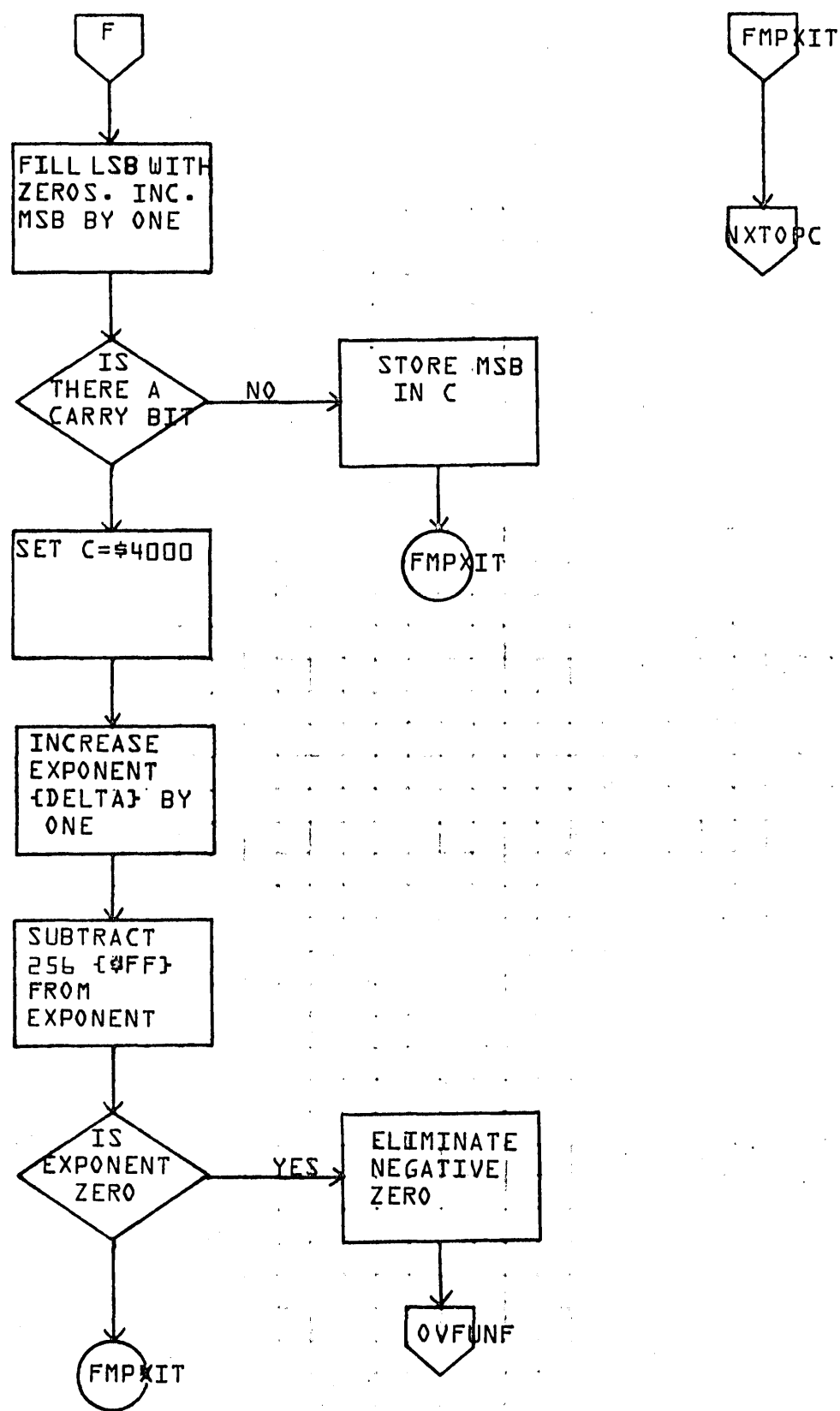
B. A JUMP TO FMPXIT THEN CAUSES A JUMP TO NXTOPC {NEXT OP CODE}.



CONTROL DATA CORPORATION SOFTWARE DOCUMENT		DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.		REV		APPROVED		DATE	
SAMPLE CODE		DOCUMENT TITLE	FLOAT	PAGE 19 OF 39		PROJECT MGR.							
FLOWCHART		NUMBER		ISSUE DATE		PROJECT NAME							
DECISION TABLE		DRAWN BY		DATE		TASK NO.							
OTHER						TASK NAME							

PRINTED IN USA

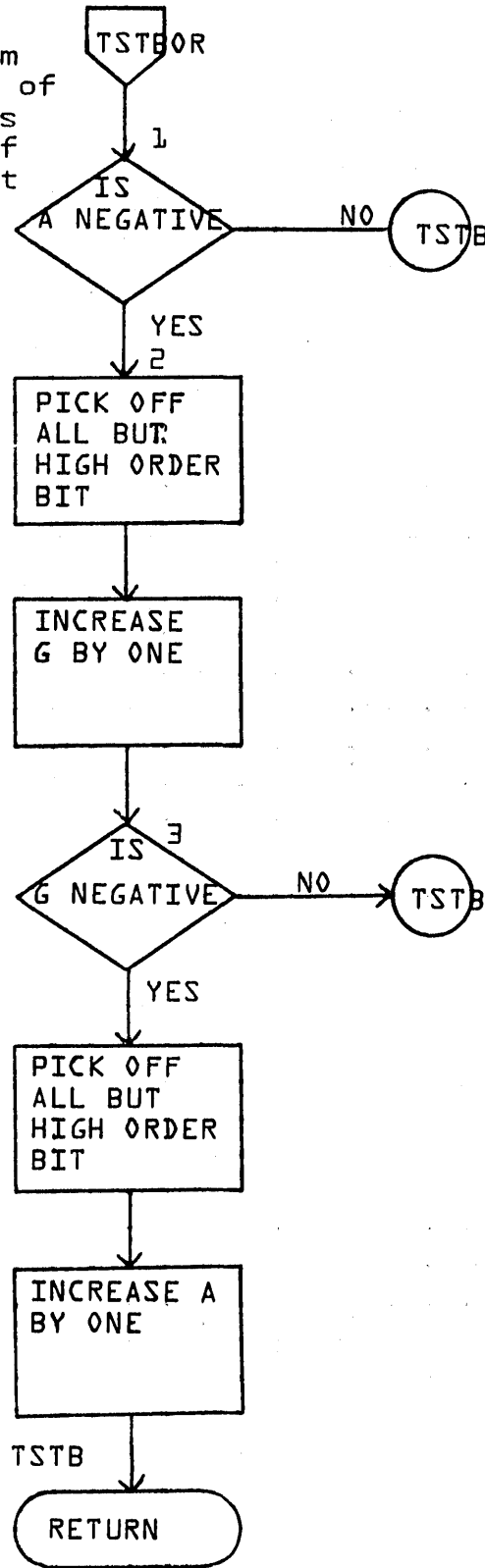
1 2 3 4 5



CONTROL DATA CORPORATION SOFTWARE DOCUMENT		DOCUMENT CLASS IMS	MACH. TYPE 1700	PROJECT NO.	REV	APPROVED	DATE
		DOCUMENT TITLE FLOAT	PAGE NO. 2029	PROJECT MGR.			
		NUMBER	ISSUE DATE	PROJECT NAME			
		DRAWN BY	DATE	TASK NO.			
				TASK NAME			

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

1. A contains the sum of the first word of one argument times the second word of the other argument and vice versa.
2. If A is negative, carry into G.
3. If G is negative, end around.



DATE	APPROVED	REV	PROJECT NO.	MACH. TYPE	DOCUMENT CLASS
			J700	IMS	
			PROJECT MGR.	DOCUMENT TITLE	
			PROJECT NAME	FLOAT	
			TASK NO.	PAGE 1 of 79	
			TASK NAME	ISSUE DATE	
				NUMBER	
				DRAWN BY	
				DATE	

CONTROL DATA CORPORATION  
 SOFTWARE DOCUMENT

SAMPLE CODE

FLOWCHART

DECISION TABLE

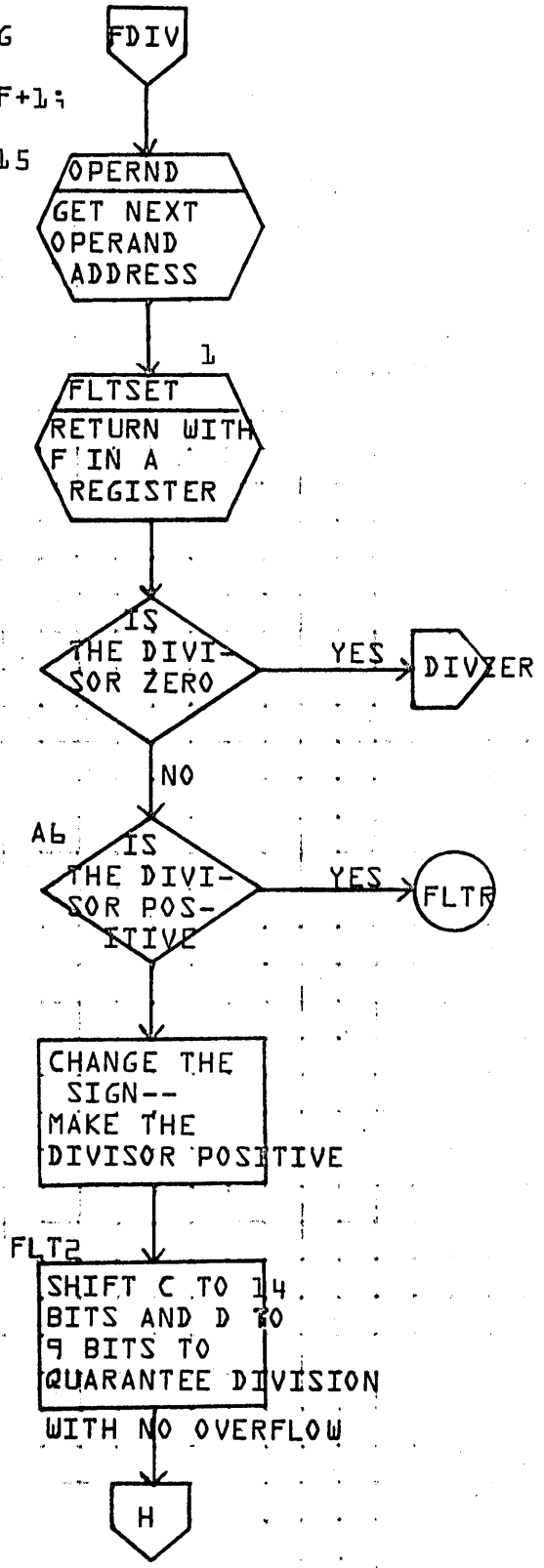
OTHER

PRINTED IN USA



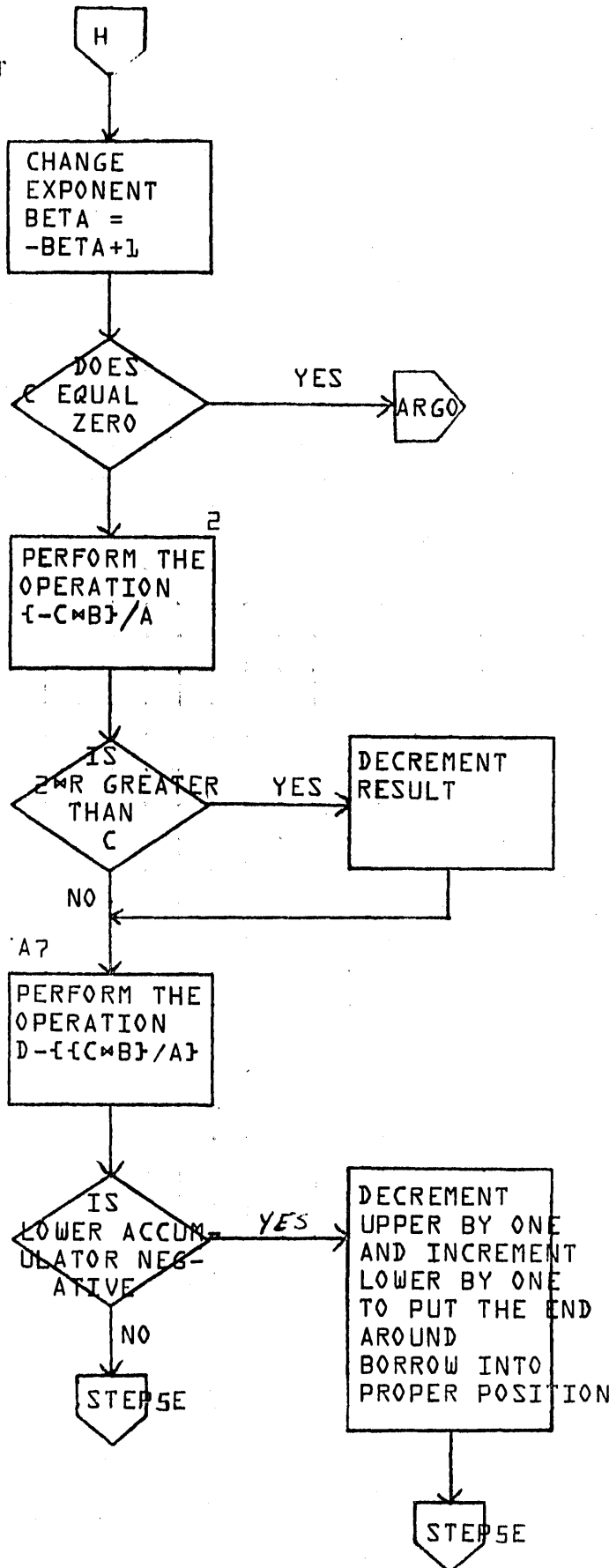
Operation Code = A  
Floating Divide {FDIV}

- GETS THE FLOATING POINT NUMBER AND STORES IN F AND F+1; THEN UNPACKS THE NUMBER INTO A=0.15 MSB B=0.8LSB, 7 zeroes BETA=EXPONENT



CONTROL DATA CORPORATION SOFTWARE DOCUMENT		DOCUMENT CLASS IMS	MACH. TYPE 1700	PROJECT NO.	APPROVED	DATE
SAMPLE CODE <input type="checkbox"/>		DOCUMENT TITLE FLOAT	ISSUE DATE PAGE 22 OF 39	PROJECT MGR.	REV	
FLOWCHART <input checked="" type="checkbox"/>		NUMBER	DATE	PROJECT NAME		
DECISION TABLE <input type="checkbox"/>		DRAWN BY		TASK NO.		
OTHER <input type="checkbox"/>				TASK NAME		

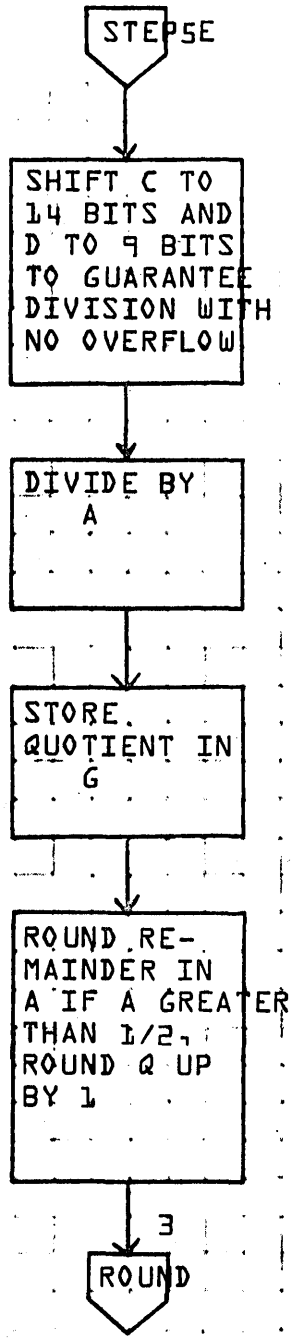
2. THE REMAINDER MUST BE POSITIVE



REV	APPROVE::	DATE
PROJECT NO.	PROJECT MGR.	PROJECT NAME
TASK NO.	TASK NAME	
DOCUMENT CLASS	MACH. TYPE	DATE
IMS	L700	
DOCUMENT TITLE	PAGE 23 of 39	
NUMBER	ISSUE DATE	
DRAWN BY		
CONTROL DATA CORPORATION	SOFTWARE DOCUMENT	
SAMPLE CODE	<input type="checkbox"/>	<input checked="" type="checkbox"/>
FLOWCHART	<input type="checkbox"/>	<input type="checkbox"/>
DECISION TABLE	<input type="checkbox"/>	<input type="checkbox"/>
OTHER	<input type="checkbox"/>	<input type="checkbox"/>

PRINTED IN USA

3. ROUND, TRUNCATE AND NORMALIZE THE RESULTS.



CONTROL DATA CORPORATION SOFTWARE DOCUMENT		DOCUMENT CLASS	IMS	MACH. TYPE	J700	PROJECT NO.	REV	APPROVED	DATE
SAMPLE CODE		DOCUMENT TITLE	FLOAT	ISSUE DATE	PAGE 24 of 39	PROJECT MGR.			
FLOWCHART		NUMBER		DATE		PROJECT NAME			
DECISION TABLE		DRAWN BY				TASK NO.			
OTHER						TASK NAME			

5

4

3

2

1

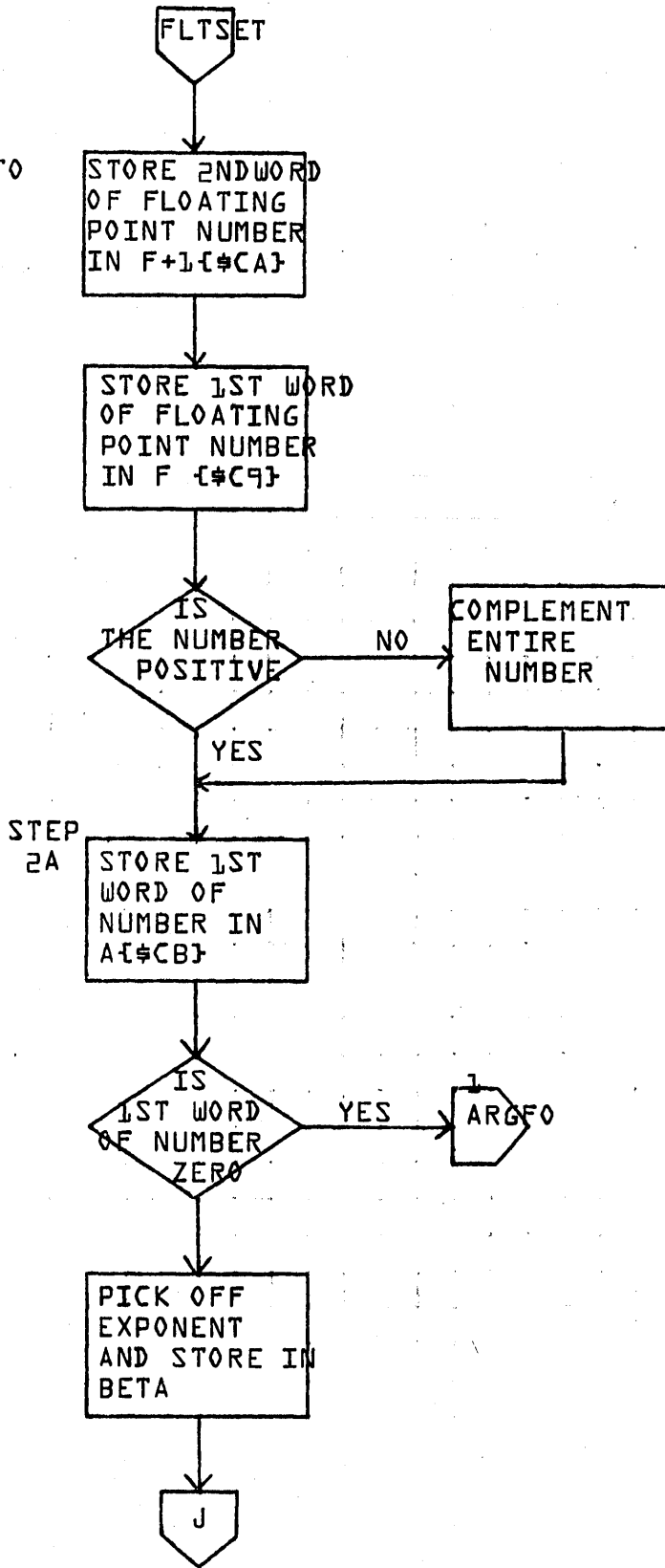
A

B

C

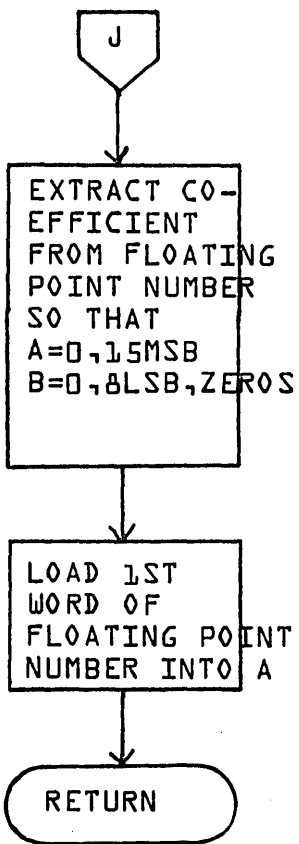
D

- AT ARGFO, THE ENTIRE NUMBER IS SET EQUAL TO +0  
 $A = +0$   
 $B = +0$   
 $BETA = +1$



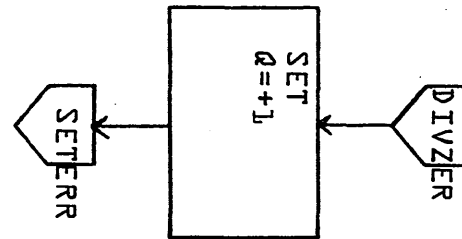
CONTROL DATA CORPORATION		DATE
SOFTWARE DOCUMENT	APPROVED	
SAMPLE CODE <input type="checkbox"/>	REV	
FLOWCHART <input checked="" type="checkbox"/>	PROJECT NO.	
DECISION TABLE <input type="checkbox"/>	PROJECT MGR.	
OTHER <input type="checkbox"/>	PROJECT NAME	
	TASK NO.	
	TASK NAME	
DOCUMENT CLASS IMS	MACH. TYPE 1700	ISSUE DATE
DOCUMENT TITLE	PAGE 25F39	DATE
NUMBER		
DRAWN BY		

PRINTED IN U.S.A.



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>		DOCUMENT CLASS <b>IMS</b>	MACH. TYPE <b>1700</b>	PROJECT NO. 	REV. 	APPROVED 	DATE 
DOCUMENT TITLE <b>FLOAT</b>		ISSUE DATE 	PAGE# OF # <b>26 of 39</b>	PROJECT MGR. 			
NUMBER 	DRAWN BY 	DATE 	PROJECT NAME 	TASK NO. 			

DIVIDE CHECK WHEN DIVIDING BY ZERO

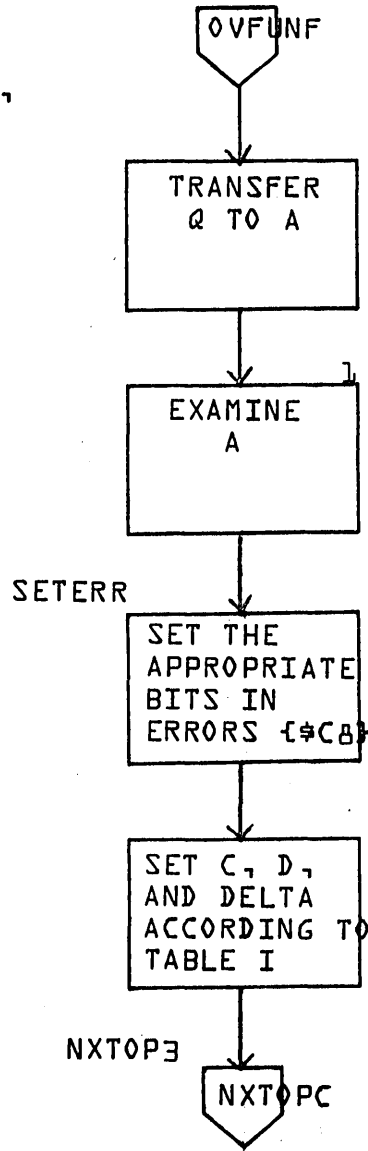


<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	- FLOAT		PAGE 27 OF 39	PROJECT MGR.			
	NUMBER		ISSUE DATE		PROJECT NAME			
	DRAWN BY		DATE		TASK NO.			
					TASK NAME			

9-44-6

### OVERFLOW OR UNDERFLOW

- 1. IF A IS POSITIVE,  
AN OVERFLOW  
OCCURRED.
- IF A IS NOT POSITIVE,  
AN UNDERFLOW  
OCCURRED.



CONTROL DATA CORPORATION SOFTWARE DOCUMENT		DOCUMENT CLASS	MACH. TYPE	PROJECT NO.	REV	APPROVED	DATE
SAMPLE CODE		DOCUMENT TITLE	1700				
FLOWCHART		NUMBER	PAGE	PROJECT MGR.			
DECISION TABLE		ISSUE DATE	28 OF 39	PROJECT NAME			
OTHER		DRAWN BY		TASK NO.			
				TASK NAME			

	C	D	DELTA
EXPONENT OVERFLOW	⊕7FFF	⊕7F80	⊕7F
DIVIDE FAULT	⊕7FFF	⊕7F80	⊕7F
EXPONENT UNDERFLOW	⊕4000	⊕0000	-⊕7F

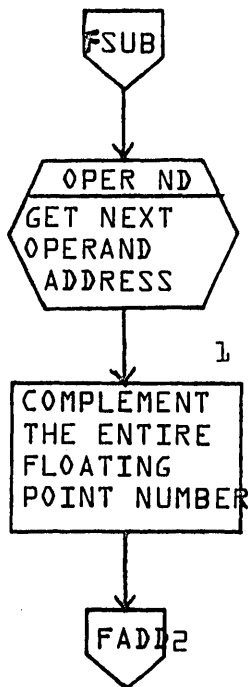
TABLE I

<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	FLOAT	PAGE 29 OF 39		PROJECT MGR.			
	NUMBER	ISSUE DATE			PROJECT NAME			
	DRAWN BY		DATE		TASK NO.			
					TASK NAME			



OPERATION CODE = 8  
 FLOATING SUBTRACT {FSUB}

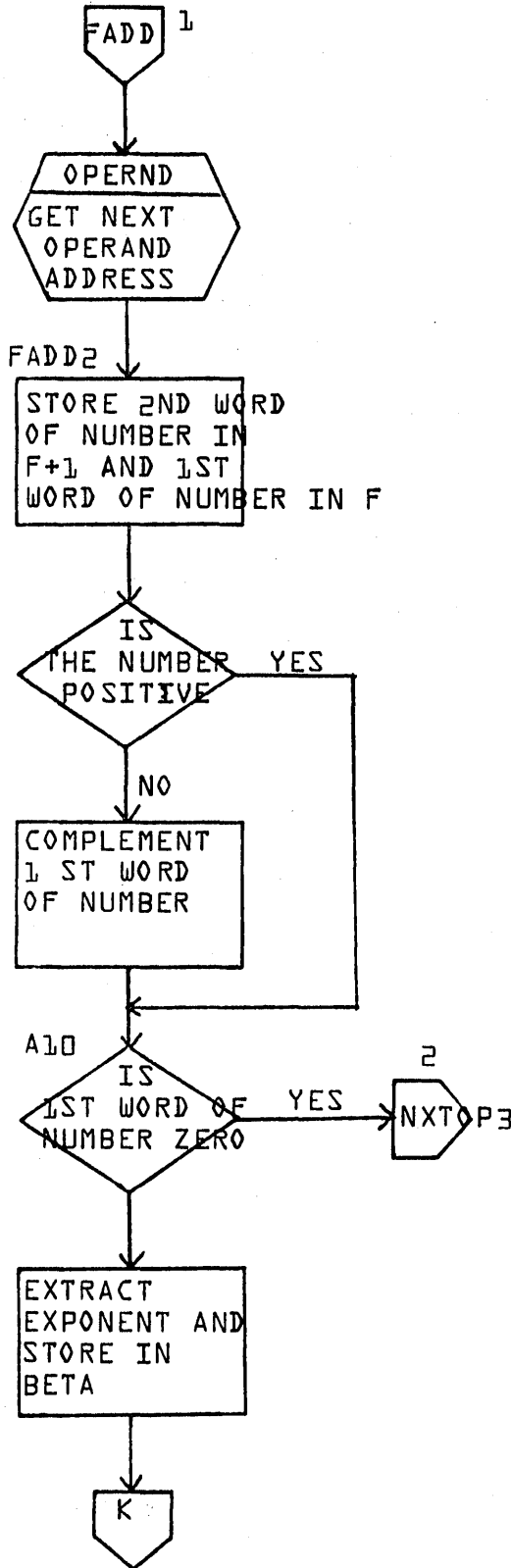
- 1. CHANGE THE SIGN BEFORE ENTERING THE ADD ROUTINE.



CONTROL DATA CORPORATION SOFTWARE DOCUMENT		DOCUMENT CLASS	MACH. TYPE	PROJECT NO.	REV	APPROVED	DATE
SAMPLE CODE		DOCUMENT TITLE	1700	PROJECT MGR.			
FLOWCHART		NUMBER	PAGE 30 of 39	PROJECT NAME			
DECISION TABLE		DRAWN BY	ISSUE DATE	TASK NO.			
OTHER			DATE	TASK NAME			

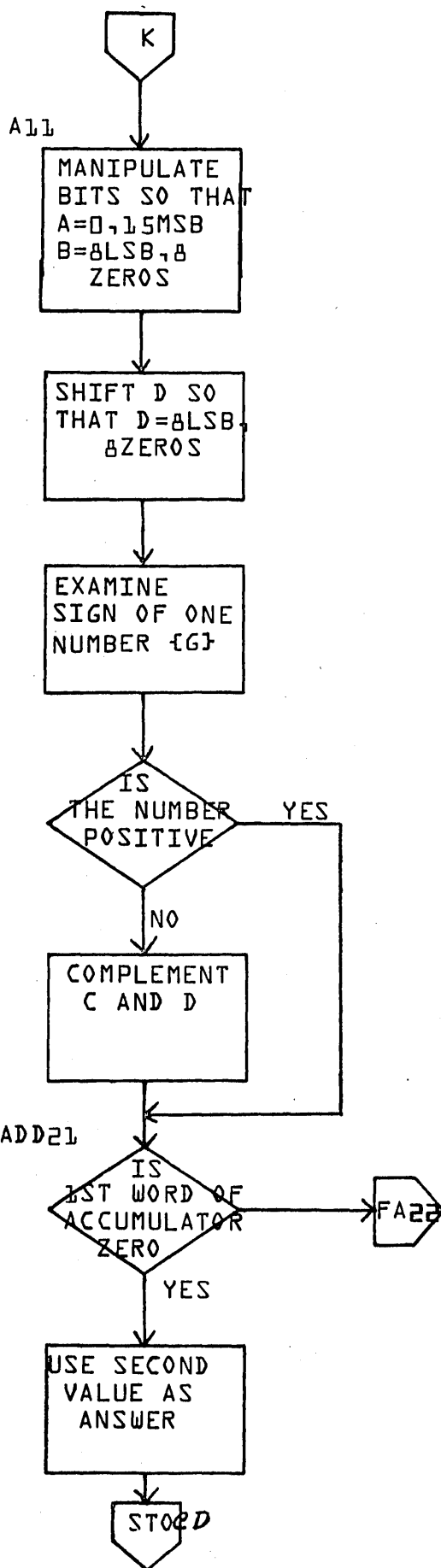
OPERATION CODE = E  
 FLOATING ADD {FADD}

1. THIS SECTION ADDS THE FLOATING POINT NUMBERS IN G AND F AND PLACES THE RESULT IN G.
2. JUMP OUT - BACK TO NXTOPC, GET NEXT OP CODE.



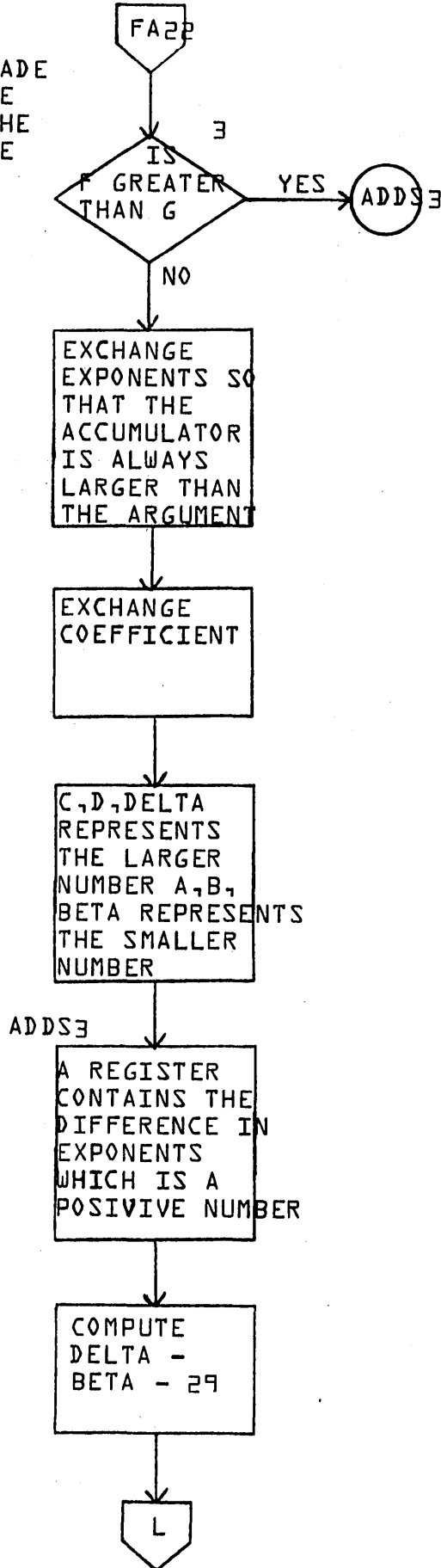
DATE		APPROVED	
REV		DATE	
PROJECT NO.	PROJECT MGR.	PROJECT NAME	TASK NO.
MACH. TYPE 1700	PAGE 1 OF 39	ISSUE DATE	DATE
DOCUMENT CLASS IMS	DOCUMENT TITLE FLOAT	NUMBER	DRAWN BY
CONTROL DATA CORPORATION SOFTWARE DOCUMENT			
<input type="checkbox"/> SAMPLE CODE <input checked="" type="checkbox"/> FLOWCHART <input type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER			

PRINTED IN USA



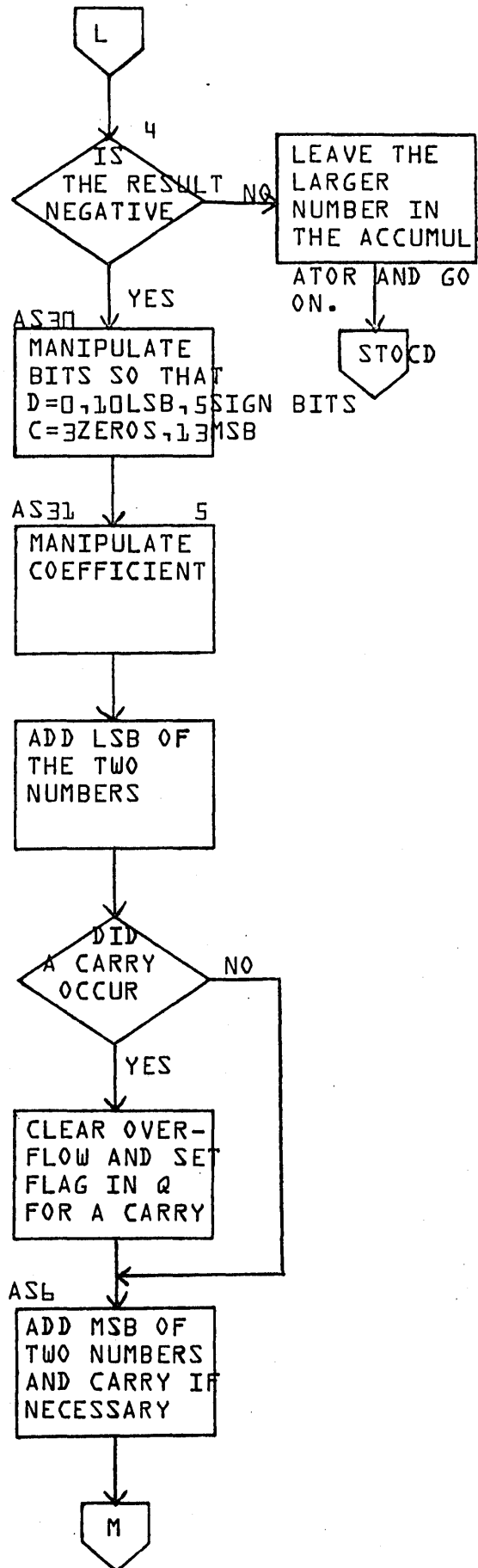
CONTROL DATA CORPORATION SOFTWARE DOCUMENT		DOCUMENT CLASS IMS	MACH. TYPE 1700	PROJECT NO.	REV	APPROVED	DATE
SAMPLE CODE FLOWCHART DECISION TABLE OTHER		DOCUMENT TITLE FLOAT	PAGE 22 OF 39	PROJECT MGR.			
		NUMBER	ISSUE DATE	PROJECT NAME			
		DRAWN BY	DATE	TASK NO.			
				TASK NAME			

3. THIS CHECK IS MADE BY EXAMINING THE DIFFERENCE OF THE EXPONENTS OF THE TWO NUMBERS.



DATE	APPROVED	REV	PROJECT NO.	MACH. TYPE	DOCUMENT CLASS	ISSUE DATE	
			PROJECT MGR.	1700	IMS	39	
			PROJECT NAME		FLOAT	PAGES OF	
			TASK NO.			39	
			TASK NAME			DATE	
CONTROL DATA CORPORATION		SOFTWARE DOCUMENT		SAMPLE CODE		DRAWN BY	
				<input type="checkbox"/>			
				<input checked="" type="checkbox"/>			
				<input type="checkbox"/>			
				<input type="checkbox"/>			

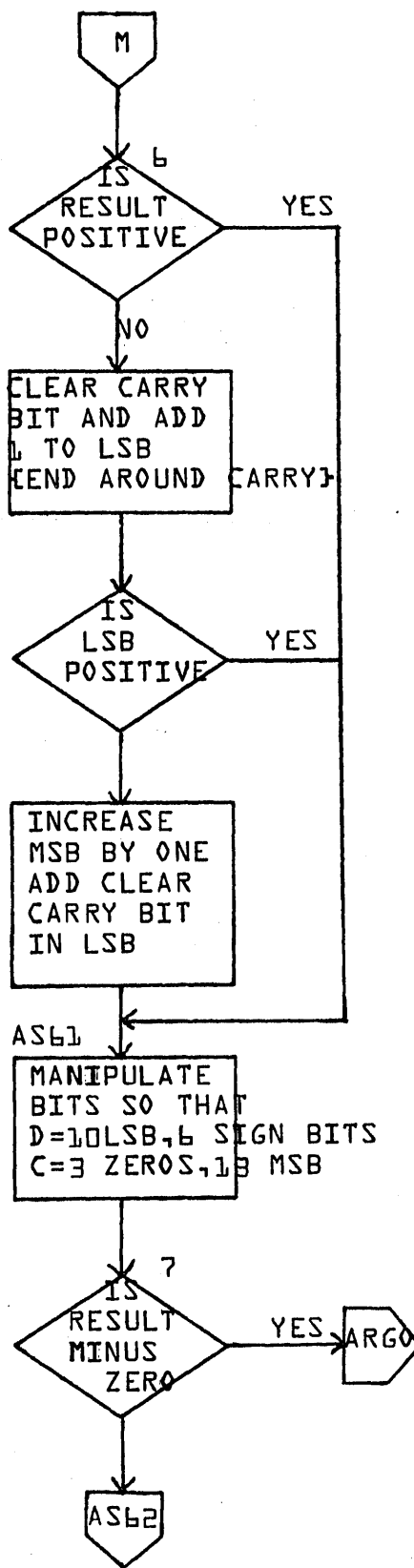
4. RESULT IS RESULT OF DELTA-BETA-29; IS THERE "TOO" GREAT A DIFFERENCE BETWEEN THE NUMBERS.
5. SHIFT SMALLER NUMBER RIGHT {DELTA-BETA} AND SET THE SIGN OF MSB TO POSITIVE. SHIFT LSB RIGHT 1 AND SET THE SIGN OF LSB POSITIVE. CLEAR THE CARRY TO BIT 15 AND ADD 1 TO MSB POSITION THE SMALLER NUMBER.



DATE		APPROVED	
REV		PROJECT NO.	
		PROJECT MGR.	
		PROJECT NAME	
		TASK NO.	
		TASK NAME	
DOCUMENT CLASS	IMS	MACH. TYPE	1700
DOCUMENT TITLE	FLOAT	PAGE	4 OF 39
NUMBER		ISSUE DATE	
DRAWN BY		DATE	
CONTROL DATA CORPORATION			
SOFTWARE DOCUMENT			
SAMPLE CODE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
FLOWCHART	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
DECISION TABLE	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
OTHER	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

6. RESULT IS RESULT OF ADDITION OF MSB AND CARRY BIT.

7. MINUS ZERO IS  
Q=7FFF  
A=FFFE

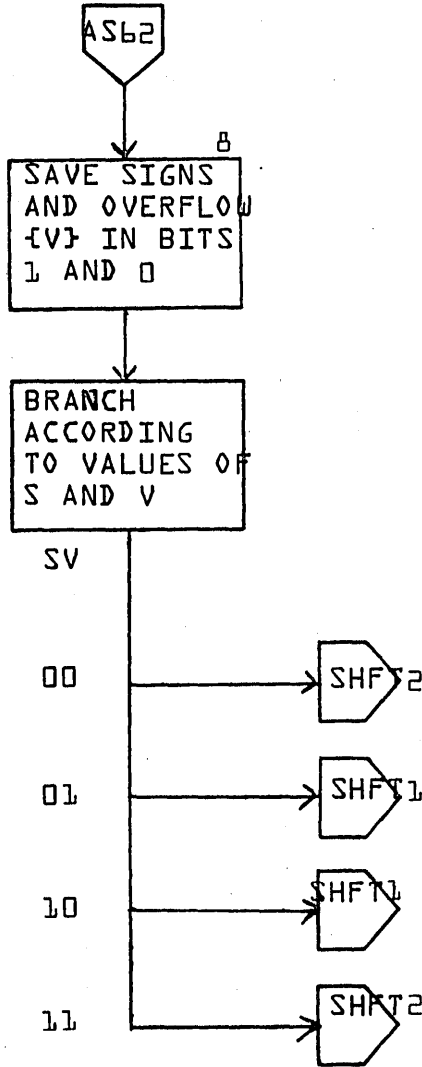


CONTROL DATA CORPORATION SOFTWARE DOCUMENT	DOCUMENT CLASS IMS	MACH. TYPE 1700	PROJECT NO.	DATE
SAMPLE CODE	DOCUMENT TITLE FLOAT	PAGE 35 OF 39	PROJECT MGR.	APPROVED
FLOWCHART	ISSUE DATE	PROJECT NAME	TASK NO.	REV
DECISION TABLE	DRAWN BY	TASK NAME		
OTHER				

PRINTED IN U.S.A.

CA 127-1

B. IF S.EQ.V, THEN V IS  
SIGN EXTENSION.  
IF S.NE.V, THEN RESULT  
OVERFLOWED INTO V.



CONTROL DATA CORPORATION SOFTWARE DOCUMENT	DOCUMENT CLASS IMS	MACH. TYPE J700	PROJECT NO.	APPROVED	DATE
SAMPLE CODE FLOWCHART	DOCUMENT TITLE FLOAT	PAGE 36 OF 38	PROJECT MGR.	REV	
DECISION TABLE	NUMBER	ISSUE DATE	PROJECT NAME		
OTHER	DRAWN BY	DATE	TASK NO.		
			TASK NAME		

A

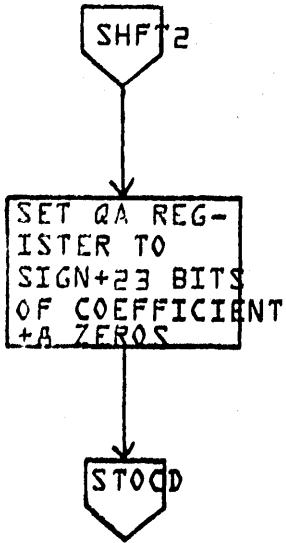
B

C

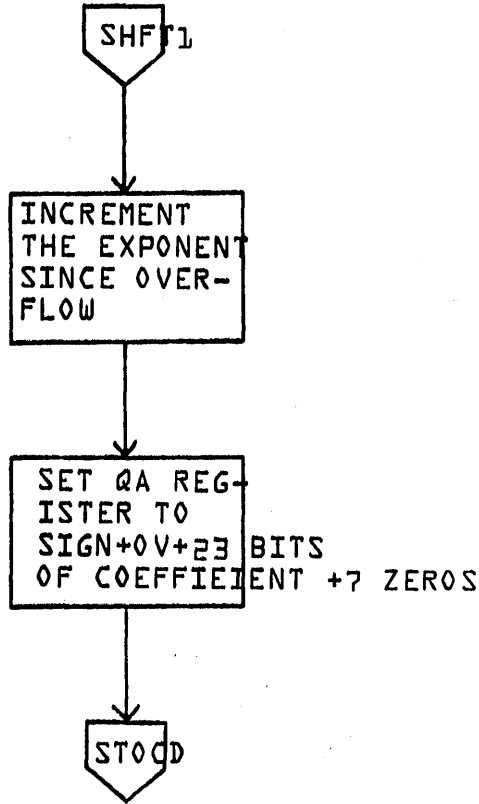
D

CONTROL DATA CORPORATION

NO OVERFLOW



OVERFLOW OCCURRED

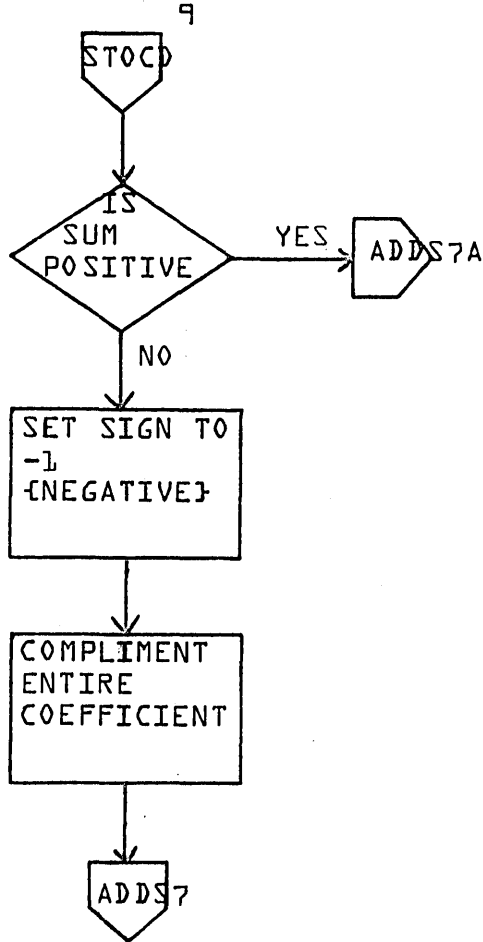


CONTROL DATA CORPORATION SOFTWARE DOCUMENT		DOCUMENT CLASS IMS	MACH. TYPE J700	PROJECT NO.	REV	APPROVED	DATE
SAMPLE CODE FLOWCHART <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>		DOCUMENT TITLE FLOAT	PAGE 27 OF 38	PROJECT MGR.			
DECISION TABLE <input type="checkbox"/>		NUMBER	ISSUE DATE	PROJECT NAME			
OTHER <input type="checkbox"/>		DRAWN BY	DATE	TASK NO.			
				TASK NAME			

PRINTED IN U.S.A.



9. SAVE THE SIGN AND  
MAGNITUDE OF THE  
RESULTS.  
ON ENTRY  
C=SIGN+15MSB  
D=8LSB+8ZEROS



CONTROL DATA CORPORATION SOFTWARE DOCUMENT		DOCUMENT CLASS	IMS	MACH. TYPE	L700	PROJECT NO.		REV		APPROVED		DATE	
SAMPLE CODE		DOCUMENT TITLE	FLOAT			PROJECT MGR.							
FLOWCHART		NUMBER			PAGE 28 OF 39	PROJECT NAME							
DECISION TABLE		DRAWN BY			ISSUE DATE	TASK NO.							
OTHER					DATE	TASK NAME							

A

B

C

D

5

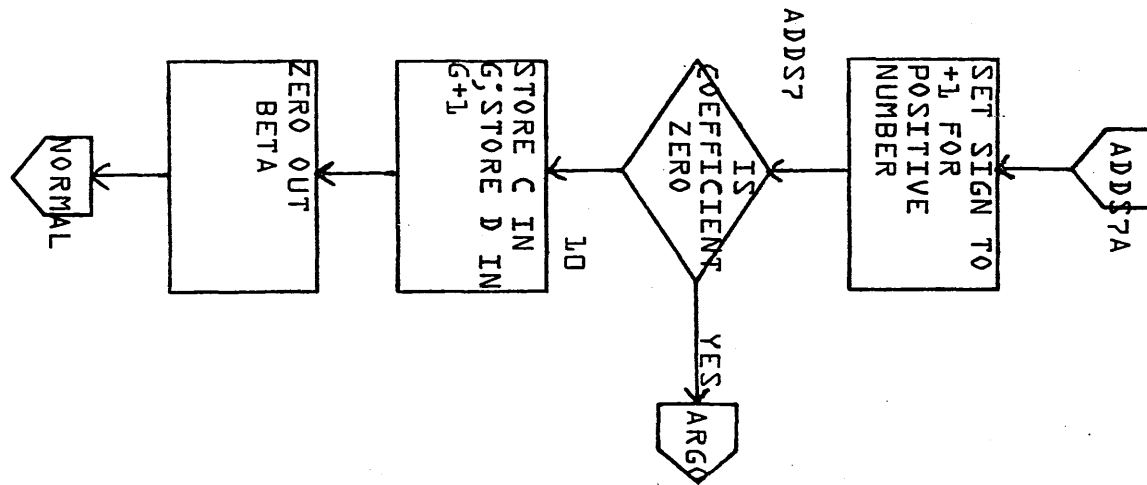
4

3

2

1

10. C=0+15MSB  
D=8LSB+8ZEROS



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input checked="" type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	FLOAT	PAGE 29 OF 39		PROJECT MGR.			
	NUMBER		ISSUE DATE		PROJECT NAME			
	DRAWN BY		DATE		TASK NO.			
					TASK NAME			

DOCUMENT CLASS IMSPAGE NO. 10-1PRODUCT NAME 1700 Mass Storage FORTRANPRODUCT MODEL NO. C005 V2.0MACHINE SERIES 1700

## CHAPTER 10

## TABLE OF CONTENTS

- 10.0 Object Time Intrinsic Functions
- 10.1 General Description
- 10.2 Q8AB Routine
- 10.3 SIGN Routine
- 10.4 FXFL Routine

DOCUMENT CLASS IMS PAGE NO. 10-2  
 PRODUCT NAME 1700 Mass Storage FORTRAN  
 PRODUCT MODEL NO. C005 V2.0 MACHINE SERIES 1700

## 10.0 Object Time Intrinsic Functions

### 10.1 General Description

The following Object Time Intrinsic Functions are included in this chapter:

Intrinsic Functions	I/O Routine
ABS{X}	Q8AB
SIGN{X, Y}	SIGN
IFIX{X}	FXFL
FLOAT{I}	FXFL

### 10.2 Q8AB Routine

This routine, which is written in 1700 Assembly Language, computer the absolute value of a floating point number and leaves the result in the pseudo accumulation.

The calling sequence is:

RTJ ABS

1. address of argument

The entry points are:

Q8AB

ABS {which is equated to Q8AB}

The external declared is:

FLOT

The low core locations used by this routine are:

ABSPRM {#E3, #E4} for the parameter of the absolute value function

FLACC {#C5, #C6} pseudo accumulator

### 10.3 SIGN Routine

This routine, which is written in 1700 Assembly Language, computer the sign of the second argument times the absolute value of the first argument and leaves the result in the pseudo accumulator.

DOCUMENT CLASS IMS PAGE NO. 10-3  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

The calling sequence is:

RTJ SIGN

1. address of first argument
2. address of second argument

The entry points are:

QBSG  
SIGN {which is equated to QBSG}

The declared external is:

FL0T

The low core locations used by this routine are:

A1 {#D8}  
A2 {#D9}  
FF {#E1} used to save I register  
QS {#E2} used to save Q register

10.4

#### FXFL Routine

This routine, which is written in 1700 Assembly Language, performs the conversion of a floating point number into an integer {Intrinsic Function IFIX} and of an integer into a floating point number {Intrinsic Function FLOAT}; it is also called by programs which have conversion cross equal signs.

The entry points declared by this routine are:

QBQFIX  
QBFX  
QBQFLT  
QBFL0T  
FLOAT {which is equated to QBFL0T}  
IFIX {which is equated to QBFX}  
DFIX {which is equated to IFIX}

The declared external is:

FL0T

## CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 10-4  
 PRODUCT NAME 1700 Mass Storage FORTRAN  
 PRODUCT MODEL NO. C005 V2.0 MACHINE SERIES 1700

The low core locations used by this routine are:

CELL2        {#D8}  
 CELL1        {#DA}  
 QS            {#E2} used to save Q register

The calling sequences for FXFL for the Intrinsic Functions are:

for IFIX{X}:

RTJ IFIX

1. address of argument

The resultant integer is in the A register upon return to the user program.

for FLOAT{I}:

RTJ FLOAT

1. address of argument

The calling sequence for the conversion across the equal sign of a floating point number into an integer is:

RTJ QBQFIX

In this case the argument is in the pseudo accumulator upon entry and the resultant integer is in the A register upon exit.

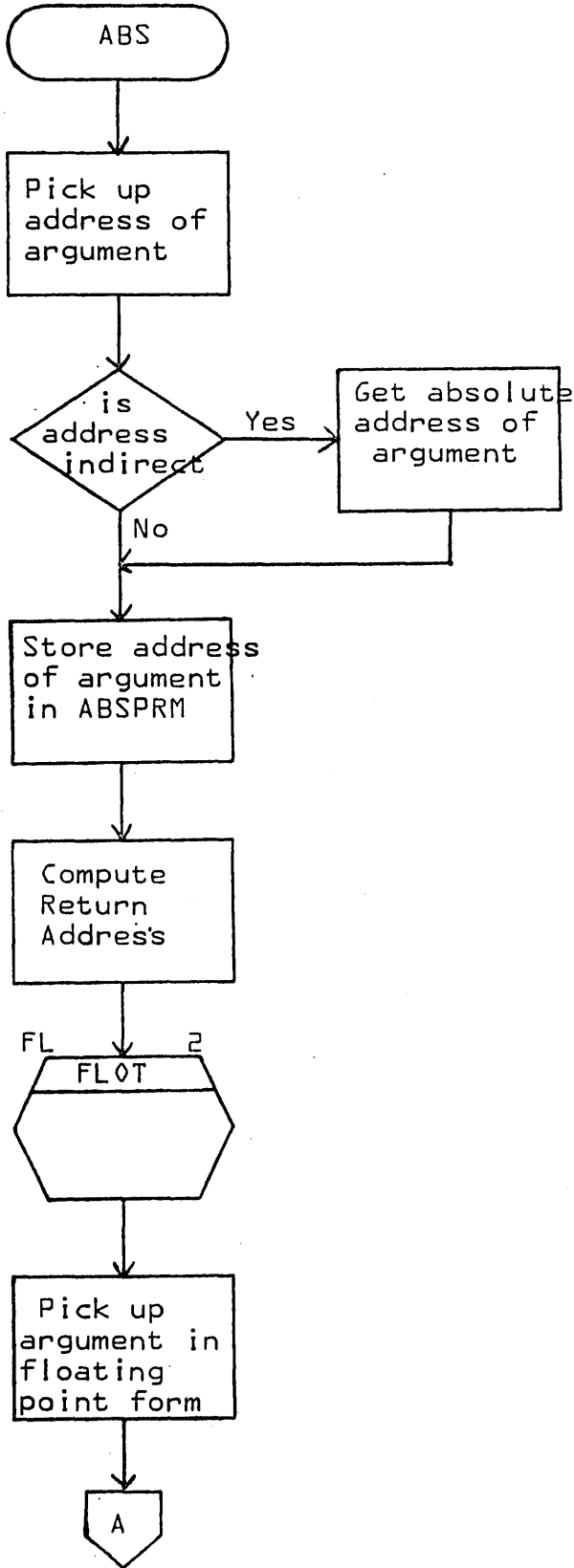
The calling sequence for the conversion across the equal sign of an integer into a floating point number is:

RTJ QBQFLT

In this case the argument is in the A register upon entry.

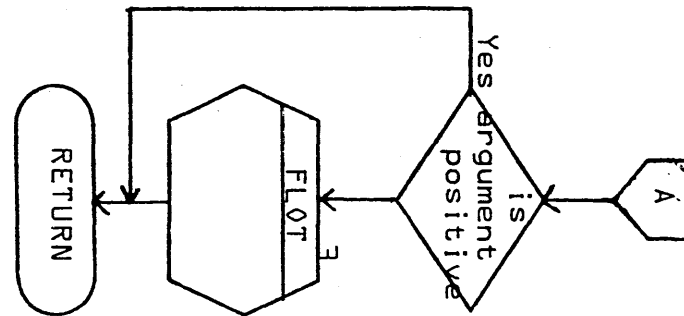
1. Is address negative?

2. Do a floating point load of argument into pseudo accumulator.



CONTROL DATA CORPORATION	DATE
SOFTWARE DOCUMENT	APPROVED
SAMPLE CODE <input type="checkbox"/>	REV
FLOWCHART <input type="checkbox"/>	PROJECT NO.
DECISION TABLE <input type="checkbox"/>	PROJECT MGR.
OTHER <input type="checkbox"/>	PROJECT NAME
	TASK NO.
	TASK NAME
	DATE
	ISSUE DATE
	DATE
	DATE

3. Do a floating point complement of the argument of ABS.



**CONTROL DATA CORPORATION**

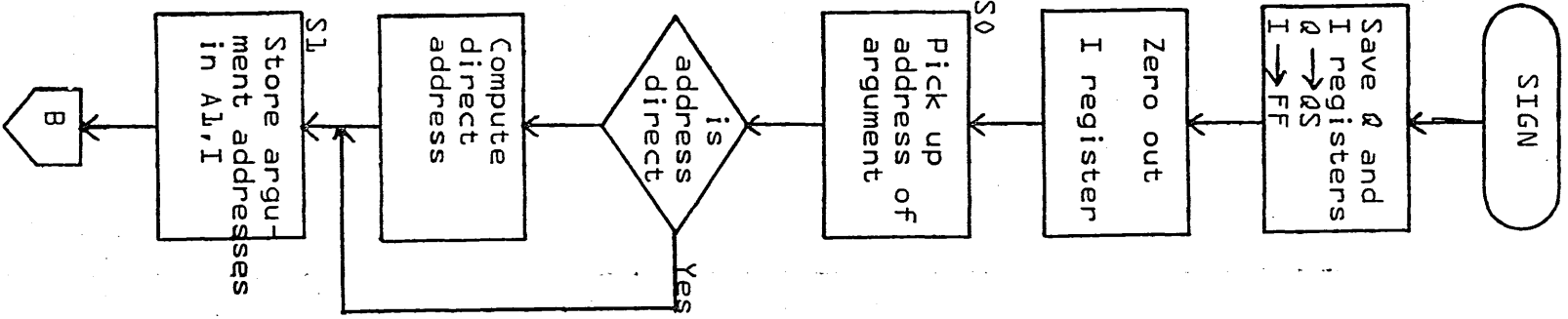
SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE				PROJECT MGR.			
NUMBER	Q8AB	PAGE	2 OF 2	PROJECT NAME			
		ISSUE DATE		TASK NO.			
DRAWN BY		DATE		TASK NAME			

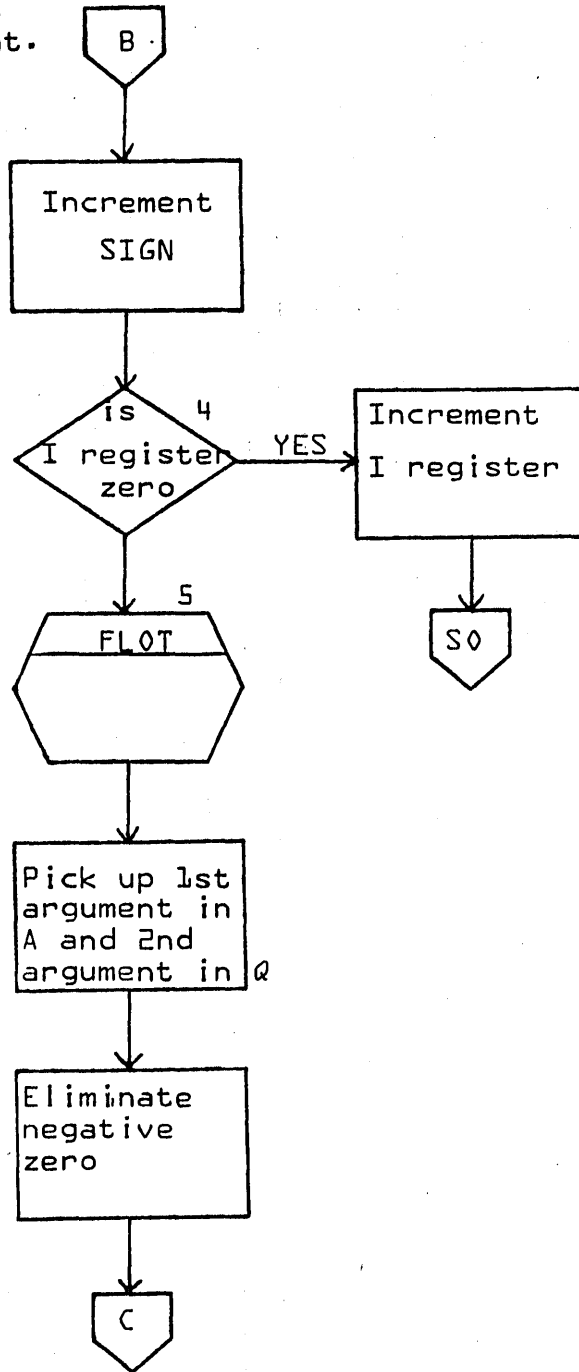


A  
B  
C  
D



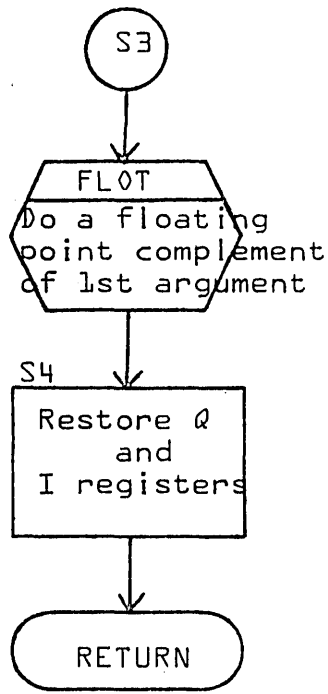
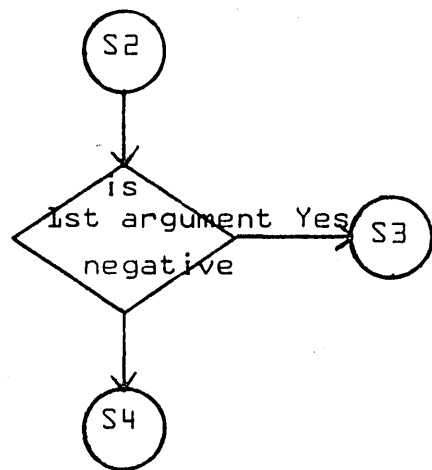
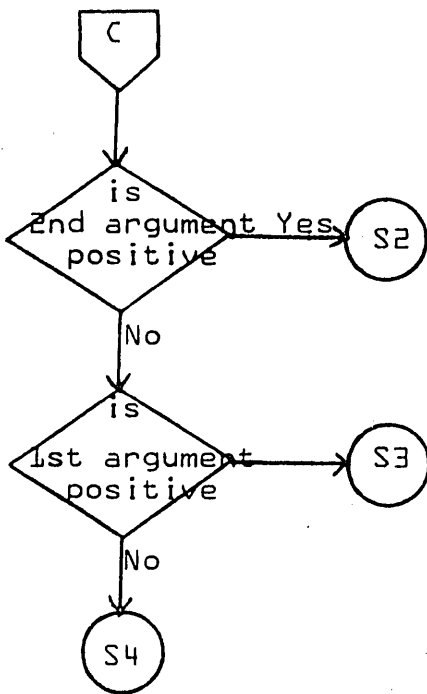
<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
	DOCUMENT TITLE	SIGN			PROJECT MGR.			
		PAGE 1 OF 3			PROJECT NAME			
	NUMBER	ISSUE DATE			TASK NO.			
	DRAWN BY	DATE			TASK NAME			

- 4. Have we just processed the first argument's address.
- 5. Do a floating point load of 1st argument.



CONTROL DATA CORPORATION SOFTWARE DOCUMENT	DOCUMENT CLASS TITLE	IMS	MACH. TYPE	1700	PROJECT NO.		REV	APPROVED	DATE
SAMPLE CODE	DOCUMENT NUMBER	SIGN	PAGE	2 of 3	PROJECT MGR.				
FLOWCHART	ISSUE DATE				PROJECT NAME				
DECISION TABLE	DRAWN BY				TASK NO.				
OTHER					TASK NAME				

PRINTED IN U.S.A.



CONTROL DATA CORPORATION		DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.		APPROVED	DATE
SOFTWARE DOCUMENT		DOCUMENT TITLE	SIGN	PAGE	3 of 3	PROJECT MGR.		REV	
SAMPLE CODE <input type="checkbox"/>		NUMBER		ISSUE DATE		PROJECT NAME			
FLOWCHART <input type="checkbox"/>		DRAWN BY		DATE		TASK NO.			
DECISION TABLE <input type="checkbox"/>						TASK NAME			
OTHER <input type="checkbox"/>									

A

B

C

D

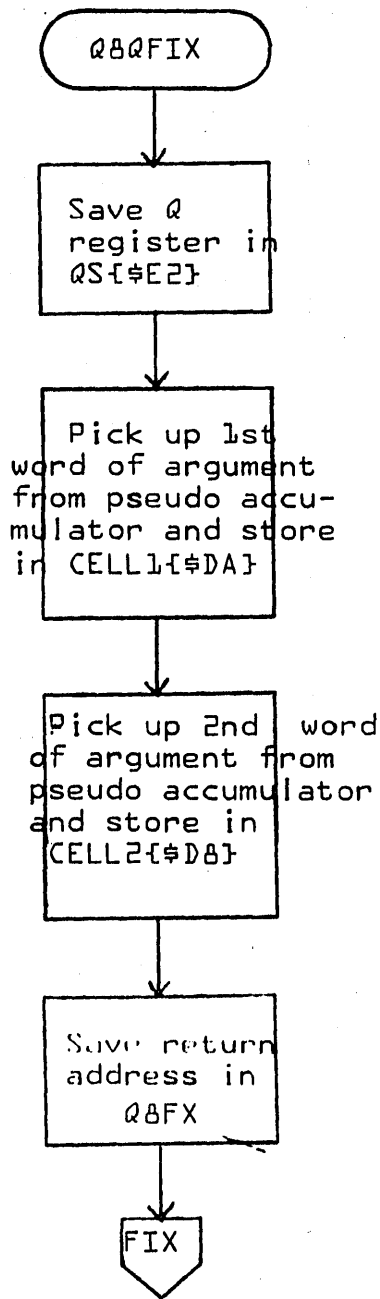
5

4

3

2

1



CONTROL DATA CORPORATION		DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	APPROVED	DATE
SOFTWARE DOCUMENT		DOCUMENT TITLE	FXFL			PROJECT MGR.		
SAMPLE CODE <input type="checkbox"/>		NUMBER		PAGE	1 of 9	PROJECT NAME		
FLOWCHART <input type="checkbox"/>		DRAWN BY		ISSUE DATE		TASK NO.		
DECISION TABLE <input type="checkbox"/>				DATE		TASK NAME		
OTHER <input type="checkbox"/>								

PRINTED IN U.S.A.

1 2 3 4 5

Q&FX  
{IFIX}

Save Q  
register  
in QS

Pick up  
address of  
argument

is  
address  
direct

Yes

No

Compute  
Direct  
address

FX5  
Pick up 1st  
word of argu-  
ment and store  
in CELL1{#DA}

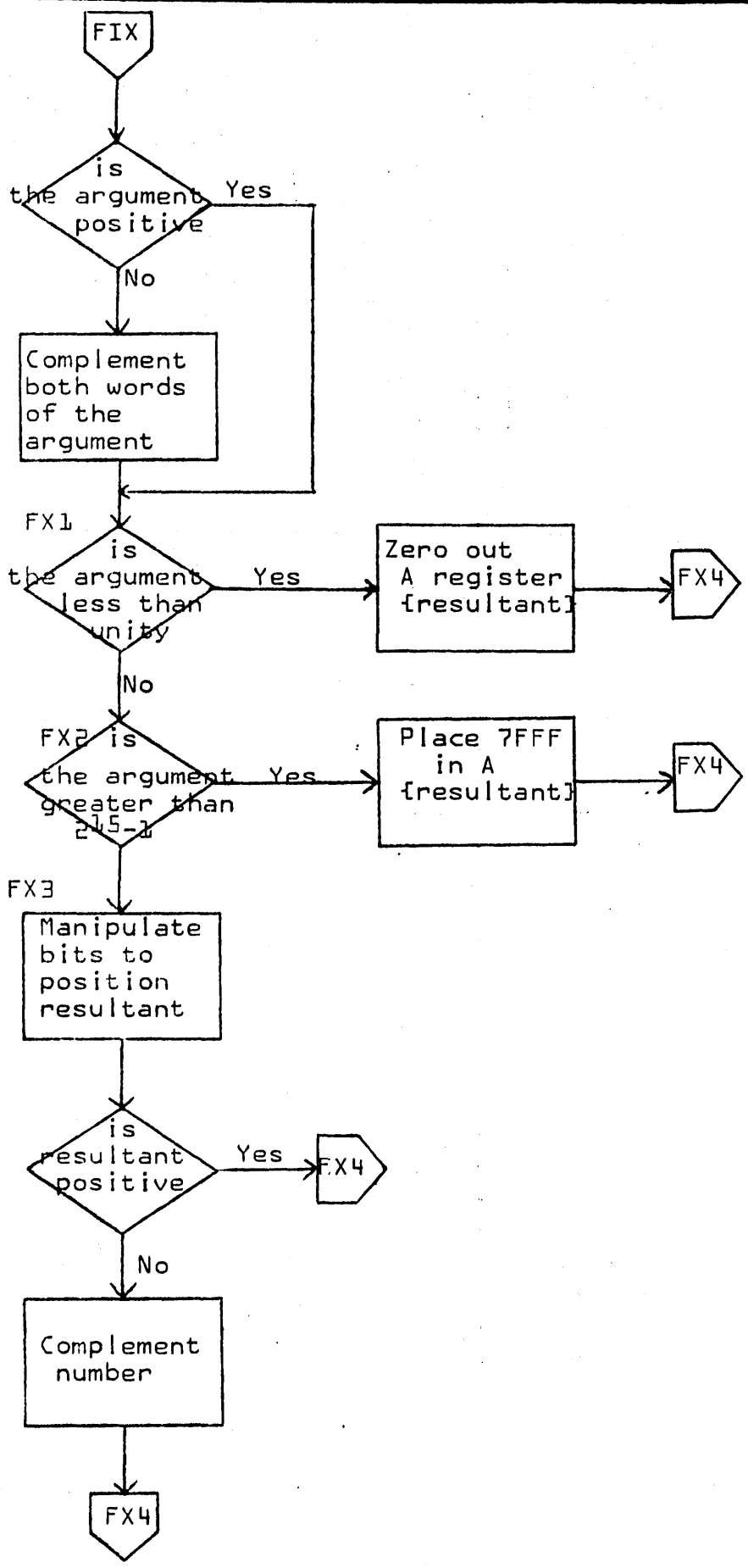
Pick up 2nd  
word of argument  
and store in  
CELL2{#DB }

Compute  
return  
address

FIX

CONTROL DATA CORPORATION SOFTWARE DOCUMENT		DOCUMENT CLASS	TMS	MACH. TYPE	1700	PROJECT NO.		REV	APPROVED	DATE
SAMPLE CODE <input type="checkbox"/>		DOCUMENT TITLE	FXFL			PROJECT MGR.				
FLOWCHART <input type="checkbox"/>		NUMBER		ISSUE DATE	PAGE 2 OF 9	PROJECT NAME				
DECISION TABLE <input type="checkbox"/>		DRAWN BY		DATE		TASK NO.				
OTHER <input type="checkbox"/>						TASK NAME				

A B C D



CONTROL DATA CORPORATION SOFTWARE DOCUMENT		DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.		REV		APPROVED		DATE	
SAMPLE CODE <input type="checkbox"/>		DOCUMENT TITLE	FXFL			PROJECT MGR.							
FLOWCHART <input type="checkbox"/>		NUMBER			PAGE 3 of 9	PROJECT NAME							
DECISION TABLE <input type="checkbox"/>		DRAWN BY			ISSUE DATE	TASK NO.							
OTHER <input type="checkbox"/>					DATE	TASK NAME							

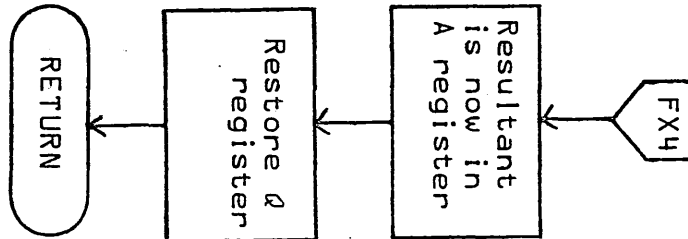
PRINTED IN U.S.A.

A

B

C

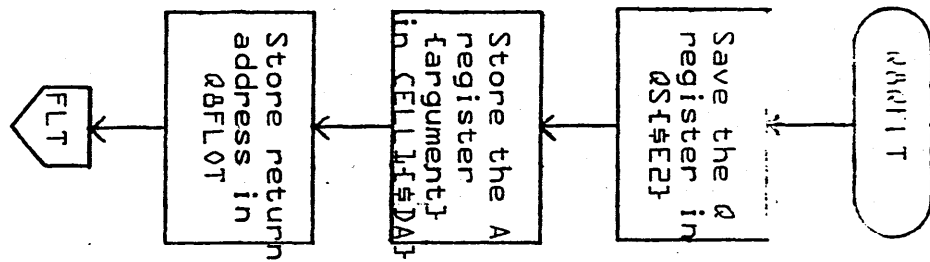
D



**CONTROL DATA CORPORATION**  
**SOFTWARE DOCUMENT**

SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	FXFL PAGE 4 of 9			PROJECT MGR.			
NUMBER	ISSUE DATE	TASK NO.		PROJECT NAME			
DRAWN BY	DATE	TASK NAME					



<b>CONTROL DATA CORPORATION</b> SOFTWARE DOCUMENT  SAMPLE CODE <input type="checkbox"/> FLOWCHART <input type="checkbox"/> DECISION TABLE <input type="checkbox"/> OTHER <input type="checkbox"/>	DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	18005125	DATE
	DOCUMENT TITLE	FXFL		PAGE 5 OF 9	PROJECT MGR.			
	NUMBER		ISSUE DATE		PROJECT NAME			
	DRAWN BY		DATE		TASK NO.			
					TASK NAME			



5

4

3

2

1

Q&FLOT  
{FLOAT}

Store Q  
register in  
Q\$E2}

Pick up  
address of  
argument

is  
address  
direct

Yes

Compute  
direct  
address

No

Pick up  
argument and  
store in  
CELL1{DA}

Compute  
return  
address

FLT

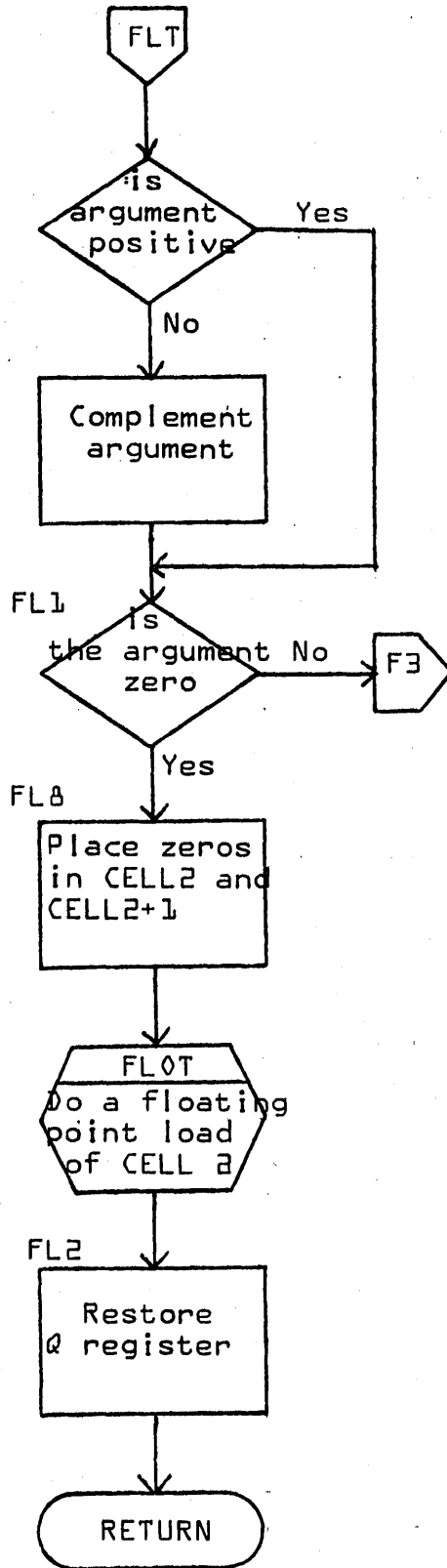
CONTROL DATA CORPORATION SOFTWARE DOCUMENT		DOCUMENT CLASS IMS	MACH. TYPE 1700	PROJECT NO.	APPROVED	DATE
SAMPLE CODE <input type="checkbox"/>		DOCUMENT TITLE FXFL	PAGE 6 OF 9	PROJECT MGR.		
FLOWCHART <input type="checkbox"/>		NUMBER	ISSUE DATE	PROJECT NAME		
DECISION TABLE <input type="checkbox"/>		DRAWN BY	DATE	TASK NO.		
OTHER <input type="checkbox"/>				TASK NAME		

A

B

C

D



CONTROL DATA CORPORATION SOFTWARE DOCUMENT	DOCUMENT CLASS IMS	MACH. TYPE J700	PROJECT NO.	REV	APPROVED	DATE
SAMPLE CODE <input type="checkbox"/>	DOCUMENT TITLE FXFL	PAGE 7 OF 9	PROJECT MGR.			
FLOWCHART <input type="checkbox"/>			PROJECT NAME			
DECISION TABLE <input type="checkbox"/>	NUMBER	ISSUE DATE	TASK NO.			
OTHER <input type="checkbox"/>	DRAWN BY	DATE	TASK NAME			

PRINTED IN U.S.A.

A B C D

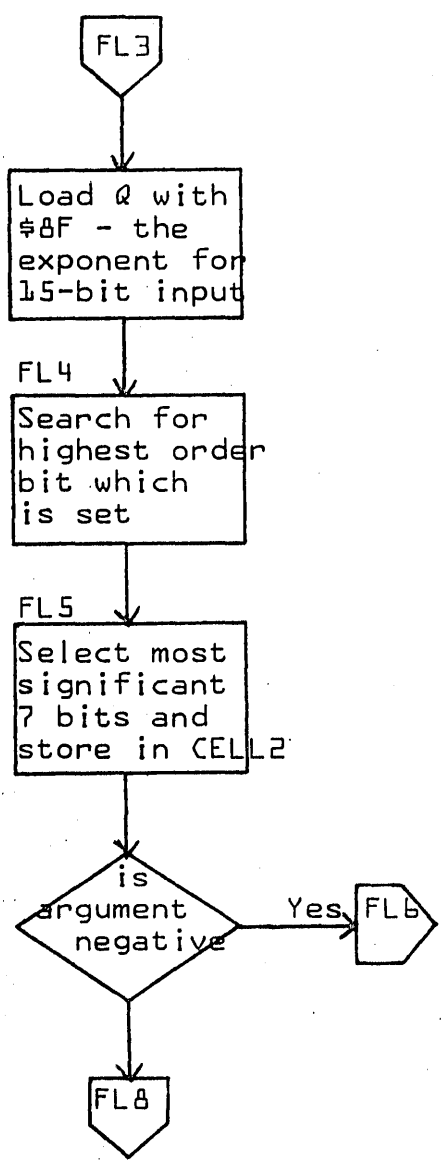
5

4

3

2

1



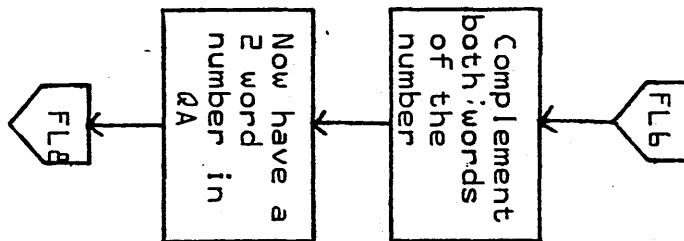
CONTROL DATA CORPORATION SOFTWARE DOCUMENT		DOCUMENT CLASS IMS	MACH. TYPE 1700	PROJECT NO.	APPROVED	DATE
SAMPLE CODE <input type="checkbox"/>		DOCUMENT TITLE FXFL	PAGE 8 OF 9	PROJECT MGR.	REV	
FLOWCHART <input type="checkbox"/>		NUMBER	ISSUE DATE	PROJECT NAME		
DECISION TABLE <input type="checkbox"/>		DRAWN BY	DATE	TASK NO.		
OTHER <input type="checkbox"/>				TASK NAME		

A

B

C

D



**CONTROL DATA CORPORATION**

SOFTWARE DOCUMENT

- SAMPLE CODE
- FLOWCHART
- DECISION TABLE
- OTHER

DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.	REV	APPROVED	DATE
DOCUMENT TITLE	FXFL			PROJECT MGR.			
		PAGE	9 OF 9	PROJECT NAME			
NUMBER		ISSUE DATE		TASK NO.			
DRAWN BY		DATE		TASK NAME			

DOCUMENT CLASS IMS PAGE NO. 11-1  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. CO05\*3.1 A/B MACHINE SERIES 1700

CHAPTER 11

T A B L E O F C O N T E N T S

11.0 DOUBLE PRECISION OBJECT TIME ARITHMETIC ROUTINES

11.1 Introduction

11.2 General Description

- 11.2.1 Temporary Storage
- 11.2.2 Calling Sequence
- 11.2.3 Returning Sequence

11.3 Function Description

11.3.1 Trigonometric Sine {DSIN}

- 11.3.1.1 Range Reduction
- 11.3.1.2 Series for SIN{X} with  $X \leq \pi/2$

11.3.2 Trigonometric Cosin {DCOS}

- 11.3.2.1 Temporary Storage for DSIN and DCOS

11.3.3 Arctangent {DATAN}

- 11.3.3.1 Range Reduction
- 11.3.3.2 Series for Arctan{X},  $X \leq \tan\{\pi/16\}$
- 11.3.3.3 Temporary Storage for DATAN

11.3.4 Exponential {DEXP}

- 11.3.4.1 Range Reduction
- 11.3.4.2 Series for  $e^X$  for  $X \leq \ln 2/2$
- 11.3.4.3 Temporary Storage for DEXP

11.3.5 Natural Logarithm {DLOG}

- 11.3.5.1 Range Reduction
- 11.3.5.2 Series for  $\ln \frac{1+X}{1-X}$  for  $|X| \leq 3 - 2\sqrt{2}$
- 11.3.5.3 Temporary Storage for DLOG

11.3.6 Square Root {DSQRT}

- 11.3.6.1 Range Reduction
- 11.3.6.2 Iterative Method
- 11.3.6.3 Temporary Storage for DSQRT

11.4 Type Out Message

- 11.4.1 Routine Argument Answer

11.5 References

11.6 Double Precision Exponentiation - Subroutine DEXPN

DOCUMENT CLASS IMS PAGE NO. 11-2  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

11.0 DOUBLE PRECISION OBJECT TIME ARITHMETIC ROUTINES

11.1 Introduction

This chapter deals with the following object time routines:

External Function	Symbolic Name
Trigonometric sine	DSIN
Trigonometric cosine	DCOS
Exponential	DEXP
Natural logarithm	DLOG
Arctangent	DATAN
Square Root	DSQRT

The above routines are designed to operate under the 1700 Mass Storage FORTRAN. They are written in 1700 Assembly Language for optimization of code and time of execution. However, they can be used by any other program working under the 1700 Operating System as long as the proper calling sequence is generated, and cells #C5 and #E5 are reserved for the floating point package and temporary storage.

11.2 General Description

The argument to any of the functions is a real value number written in the 1700 Mass Storage FORTRAN floating point notation. The result is also a real value number in floating point notation.

11.2.1 Temporary Storage

These routines use the unprotected communications area from #C9 to #E3 as temporary storage area.

11.2.2 Calling Sequence

Each object time routine is entered via an RTJ instruction to its symbolic name, followed by a cell containing the address of core where the value of the argument can be found in floating point notation.

DOCUMENT CLASS IMS PAGE NO. 11-3  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. CO05\*3.1 A/B MACHINE SERIES 1700

11.2.3 Returning Sequence

Each object time routine will return control to the user at the program location immediately following the parameter address with the calculated value of the function in the floating point pseudo-accumulator.

11.3 Function Description

The mathematical method used to approximate the functions will be found in Appendix A of this document. Basically it is a truncated Taylor-Mclaurin series.

11.3.1 Trigonometric Sine {DSIN}

11.3.1.1 Range Reduction

Let Z be the value of the argument.

If  $|Z| > 2^{21}$  then the answer will be assumed zero.

If:  $Z < 0$  let  $U = -Z$  and  $\sin \{Z\} = -\sin \{U\}$   
 $Z \geq 0$  let  $U = Z$  and  $\sin \{Z\} = \sin \{U\}$

Let I be integral part of  $U/2$  and let

$Y = U - I \cdot 2\pi$ , then  $Y \leq 2\pi$ , and  $\sin \{U\} = \sin \{Y\}$

If:  $Y > \pi$ , let  $T = Y - \pi$  then  $\sin \{Y\} = -\sin \{T\}$   
 $Y \leq \pi$ , let  $T = Y$  then  $\sin \{Y\} = \sin \{T\}$

If:  $T > \pi/2$ , let  $X = \pi - T$   
 $T \leq \pi/2$ , let  $X = T$   
 and  $\sin \{T\} = \sin \{X\}$ , with  $X \leq \pi/2$

11.3.1.2 Series for SIN{X} with  $X \leq \pi/2$

Series is a normal truncated Taylor-Mclaurin series:

$$\text{SIN}\{X\} = \sum_{n=0}^{\infty} \frac{\{-1\}^n X^{2n+1}}{\{2n+1\}!}$$

DOCUMENT CLASS IMS PAGE NO. 11-4  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. CO05\*3.1 A/B MACHINE SERIES 1700

11.3.2 Trigonometric Cosine {DCOS}

Since  $\cos\{Z\} = \sin\{Z + \pi/2\}$  then object time routine DCOS will simply be a second entry to the DSIN routine.

11.3.2.1 Temporary Storage for DSIN and DCOS

Nine cells of temporary storage are required by either DSIN or DCOS. They are:

RETADD	for return address $\$D9$
FLAG	for sign indicator $\$D8$
X and X <sub>2</sub>	for temporary floating point numbers, $\$DA$ to $\$DF$
QS	for saving Q $\$E2$

11.3.3 Arctangent {DATAN}

11.3.3.1 Range Reduction

Let Z be value of input argument.

If:  $Z < 0$ , let  $U = -Z$ , and  $\arctan\{Z\} = -\arctan\{U\}$   
 $Z \geq 0$ , let  $U = Z$ , and  $\arctan\{Z\} = \arctan\{U\}$

If:  $U < 1$ , let  $Y = U$ ,  $A = 0$ , and  $B = 1$   
 $U > 1$ , let  $Y = 1/U$ ,  $A = \pi/2$ , and  $B = -1$

If:  $Y \leq \tan\{\pi/8\}$  let  $T = \pi/16$   
 $Y > \tan\{\pi/8\}$  let  $T = 3\pi/16$

and  $\arctan\{U\} = A + B\{T + \arctan\{X\}\}$

where  $X = \frac{Y - \tan\{T\}}{1 + Y \tan\{T\}}$

11.3.3.2 Series for  $\arctan\{X\}$ ,  $X \leq \tan\{\pi/16\}$

Series is a normal truncated Taylor-Mclaurin series:

$$\text{Arctan}\{X\} = \sum_{n=0}^{\infty} \frac{\{-1\}^n X^{2n+1}}{2n+1}$$



DOCUMENT CLASS IMS PAGE NO. 11-5  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.J A/B MACHINE SERIES 1700

11.3.3.3 Temporary Storage for DATAN

Twelve cells of temporary storage are required.

They are:

ARCRET	for return address #D9
ARCFLG	for path decision #D8
X <sub>1</sub> X <sub>2</sub> ,AF,BF	for floating point numbers, #DA to #E2
QS	for saving Q #E3

11.3.4 Exponential {DEXP}

11.3.4.1 Range Reduction

Let Z be input value of argument

If: Z < 0 let Y = - Z, and e<sup>Z</sup> = 1/e<sup>Y</sup>  
 Z > 0 let Y = Z, and e<sup>Z</sup> = e<sup>Y</sup>  
 Z = 0 put answer equal to 1 and exit.

Now e<sup>Y</sup> = 2<sup>N</sup>e<sup>X</sup> if X and N are defined as follows:

Let T = Y/ln2  
 N = T + 1/2 take integral part  
 W = T - N  
 X = W ln 2

Therefore the maximum absolute value for Y is equal to 88.02968 which will generate the maximum floating point number. If input value is greater than this maximum value, then a message will be typed out and the answer is given as the maximum floating number permissible in 1700 FORTRAN.

11.3.4.2 Series for e<sup>X</sup> for X ≤ ln 2/2

Series is normal truncated Taylor-Mclaurin series:

$$e^X = \sum_{n=0}^9 \frac{X^n}{n!}$$

DOCUMENT CLASS IMS PAGE NO. 11-6  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

11.3.4.3 Temporary Storage for DEXP

Seven cells of temporary storage are required. They are:

RETEXP	for return address	⊘DB
FLAG,N	for indicator	⊘DC, DD
Y	for floating point number	⊘DB, ⊘D9, ⊘DA
QS	for saving	⊘E1

11.3.5 Natural Logarithm {DL0G}

11.3.5.1 Range Reduction

Let Z be input value of argument.

If  $Z \leq 0$  then the answer will be the maximum floating point number possible in 1700.

Now  $Z = 2^N Y$

and  $\ln\{Z\} = \{N - 1/2\} \ln 2 + \ln \left\{ \frac{1+X}{1-X} \right\}$

where  $X = \frac{Y - \sqrt{2/2}}{Y + \sqrt{2/2}}$

Thus  $|X| \leq 3 - 2\sqrt{2}$

11.3.5.2 Series for  $\ln \left\{ \frac{1+X}{1-X} \right\}$  for  $|X| \leq 3 - 2\sqrt{2}$

Series is a normal truncated Taylor-Mclaurin series:

$$\ln \left\{ \frac{1+X}{1-X} \right\} = 2 \sum_{n=0}^{\infty} \frac{X^{2n+1}}{2n+1}$$

11.3.5.3 Temporary Storage for DL0G

Eleven cells of temporary storage are required. They are:

LNRET	for return address	⊘DB
N,X,X2	for floating point numbers	⊘D9 to ⊘E1
QS	for saving	⊘E3

DOCUMENT CLASS IMS PAGE NO. 11-7  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

11.3.6 Square Root {DSQRT}

11.3.6.1 Range Reduction

Let Z be input value of argument.

If  $Z < 0$  let  $Y = -Z$  and  $\sqrt{Z} = -\sqrt{Y}$

also type out error message.

If  $Z \geq 0$  let  $Y = Z$

Now  $Y = 2^{2N} X$  and  $\sqrt{Y} = 2^N \sqrt{X}$  where  $1/4 \leq X < 1$

The Padé approximation for a first estimate is:

$$y_i = \frac{25}{7} - \frac{\left[ \frac{5000}{343} \left\{ X + \frac{15}{49} \right\} \right]}{\left\{ X + \frac{15}{49} \right\} \left\{ X + \frac{235}{49} \right\} - \frac{400}{2401}}$$

11.3.6.2 Iterative Method

The square root is computed via a Newton-Raphson iteration starting with a first approximation,  $y_i$  above. Successive approximations are found from:

$$y_{i+1} = 1/2 \left\{ y_i + \frac{X}{y_i} \right\}$$

$$\text{until } \left| y_{i+1} - y_i \right| \leq 2^{-38}$$

11.3.6.3 Temporary Storage for DSQRT

Twelve cells of temporary storage are required. They are:

QSAVE	for saving Q #E2
SQRRET	for return address #C9
SQREXP	for value of N #CA
ARG, Y1,	for floating point numbers,
TEMP,	#DB to #E1
SQRFLG	

DOCUMENT CLASS IMS PAGE NO. 11-8  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

11.4 Type Out Message

Any illegal input value will cause a predetermined value to be given as an answer and the program will exit normally. It is left to the user to take corrective action.

11.4.1

Routine	Argument	Answer
DSIN or DCOS	$ Z  > 2^{21}$	0
DEXP	$ Z  > 88.02968$	$\infty$
DLOG	$Z \leq 0$	$\infty$
DSQRT	$Z < 0$	$-\sqrt{ Z }$

11.5

References

CONTROL DATA 6600 Computer Systems, Programming System/Library Functions, A Study of Mathematical Approximations, Pub. No. 60114500.

HANDBOOK OF MATHEMATICAL FUNCTIONS, National Bureau of Standards Applied Mathematics Series-55, Library of Congress catalog number 64-60036.

11.6

DEXPN Subroutine - Double Precision Exponentiation {\*\*}

This subroutine is called by the FORTRAN compiler for programs that use the exponentiation operator, i.e., the symbol \*\*, in performing double precision exponentiation. This subroutine is written in 1700 Assembly Language. It contains the following entry points:

- QBQD2I Double precision floating number to integer power
- QBQD2F Double precision floating number to single precision floating power
- QBQD2D Double precision floating number to double precision floating power

DOCUMENT CLASS IMS PAGE NO. 11-9  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

The following externals are declared:

DFLOT  
DLOG  
DEXP

The cells required for temporary storage are:

DFLOFG	⊥E5	Floating/Fixed Flag
DFRSLT	⊥DB	Intermediate DP Results ⊥DB, ⊥DC, ⊥DD
SIGN	⊥E1	Sign of Exponent
COEFF	⊥E2	Address of Coefficient
EXPO	⊥D7	Address of Exponent
MLTPR	⊥DE	Powers of Coefficient {double precision floating point number - 3 words}
DFLOVF	⊥CB	Floating Point Package Overflow Flag
DFLACC	⊥C5	Double Precision Floating Point Pseudo Accumulation {⊥C5, ⊥Cb, ⊥C7}
CZERO	⊥22	Low Core Cell containing Zero

The calling sequence is:

RTJ Q8QXXX Where XXX is D2I, D2F, or D2D

1. address of coefficient
2. address of exponent

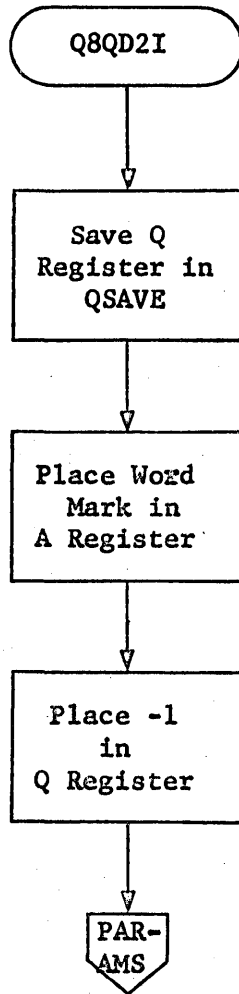
DOCUMENT CLASS IMS PAGE NO. 11-10  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

Double Floating to Single Floating



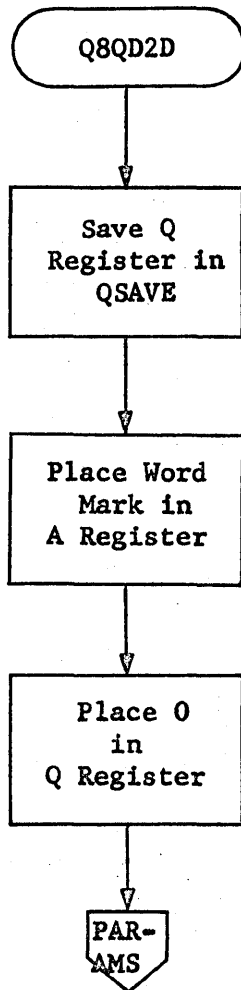
DOCUMENT CLASS IMS PAGE NO. 11-11  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

Double Floating to Integer



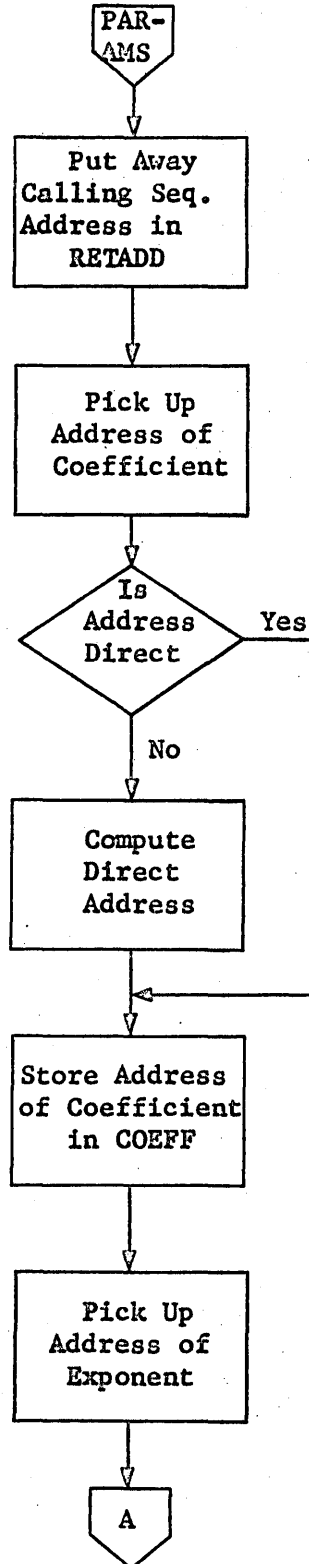
DOCUMENT CLASS IMS PAGE NO. 11-12  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

Double Floating to Double Floating



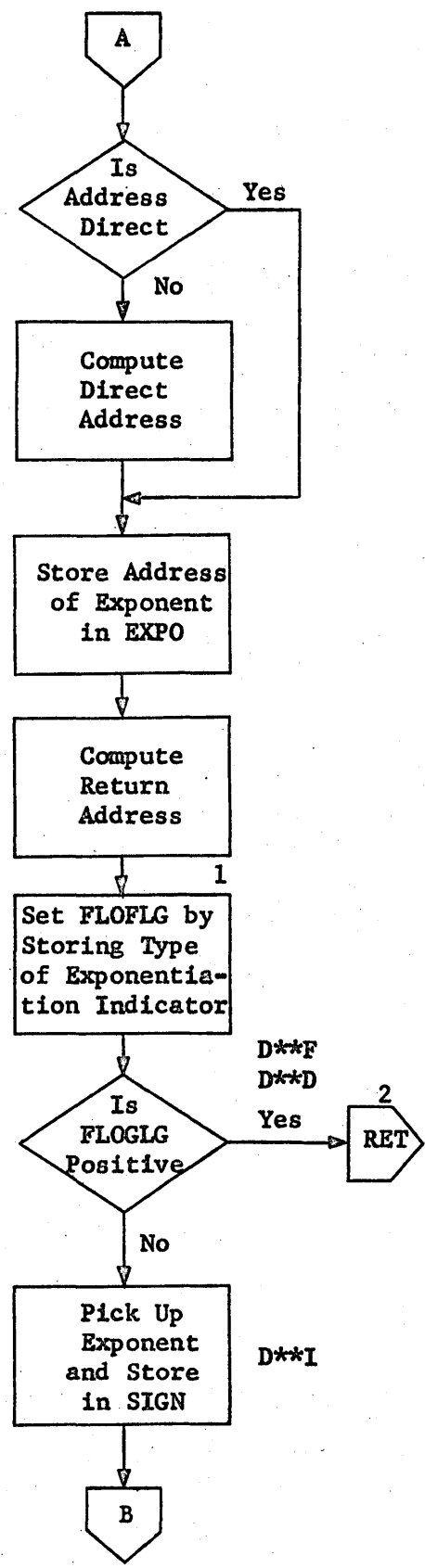


DOCUMENT CLASS IMS PAGE NO. 11-13  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. CO05\*3.1 A/B MACHINE SERIES 1700



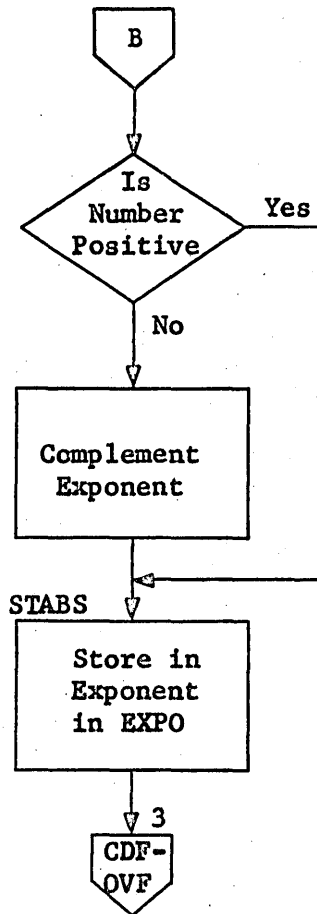
DOCUMENT CLASS IMS PAGE NO. 11-14  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

1. DFLOFG equals
  - 1 D\*\*I
  - 0 D\*\*F
  - +1 D\*\*D
2. From this place a jump is made to DLOGEX



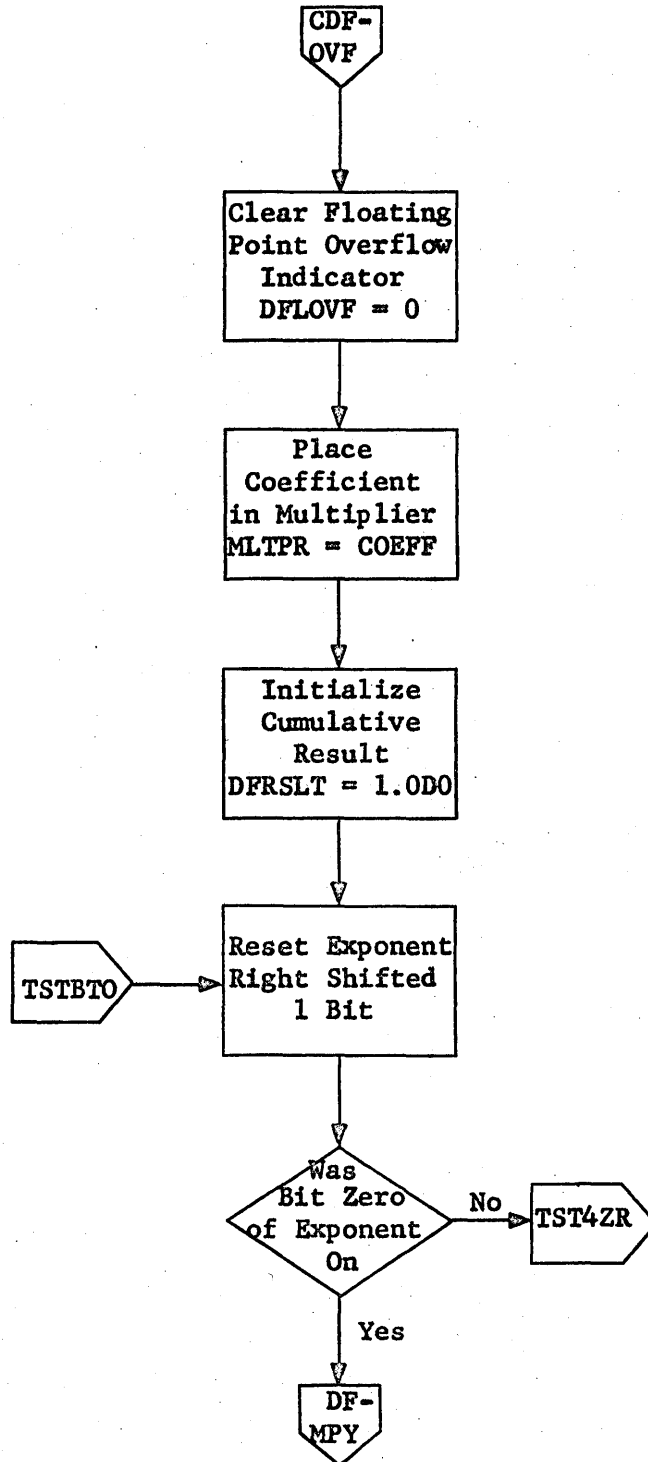
DOCUMENT CLASS IMS PAGE NO. 11-15  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

- 3. Process D\*\*I
- 4. Process D\*\*F  
or D\*\*D



DOCUMENT CLASS IMS PAGE NO. 11-16  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

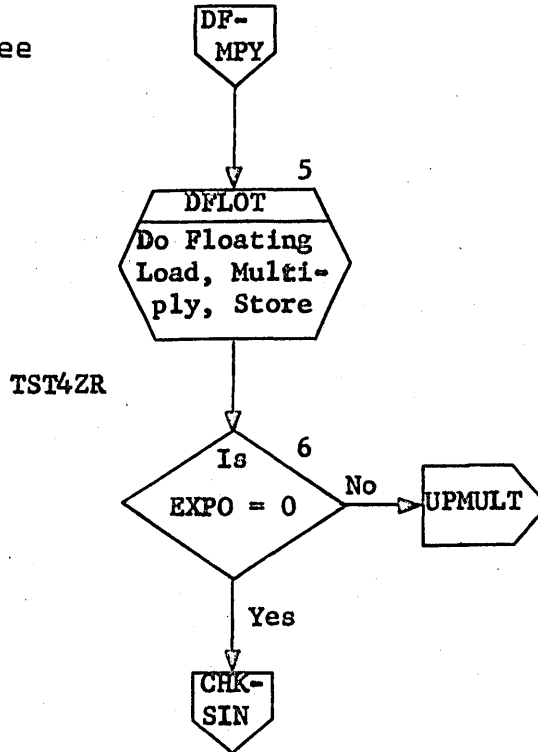
Process Double Floating to Integer



DOCUMENT CLASS IMS PAGE NO. 11-17  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

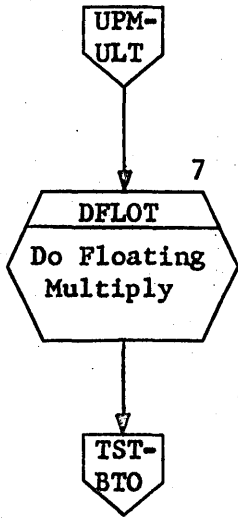
5. Floating  
DFRSLT =  
DFRSLT\*MLTPR

6. Test exponent to see  
if all significant  
bits have been  
shifted out.



DOCUMENT CLASS IMS PAGE NO. 11-18  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

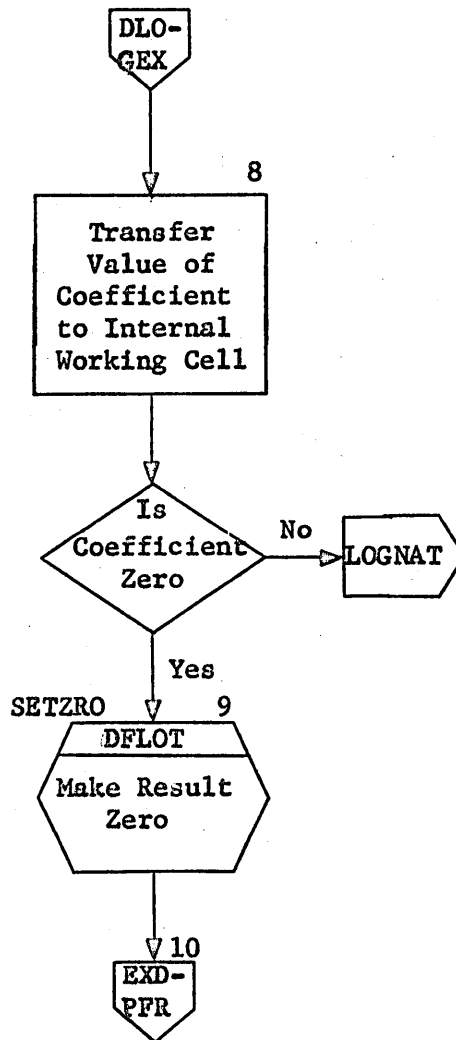
7. Square floating multiplier  
MLTPR = MLTPR\*MLTPR



DOCUMENT CLASS IMS PAGE NO. 11-19  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

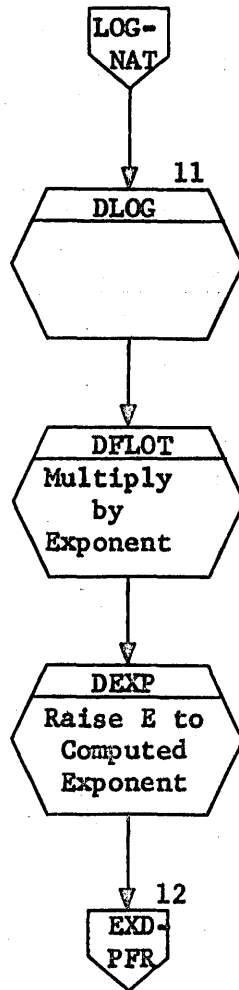
Process Double Floating to Floating or Single Floating to Floating

- 8. First word into  
DFRSLT;  
Second word into  
DFRSLT+1
- 9. Make result zero  
to avoid a fault  
in the natural  
log routine.
- 10. Exit



DOCUMENT CLASS IMS PAGE NO. 11-20  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

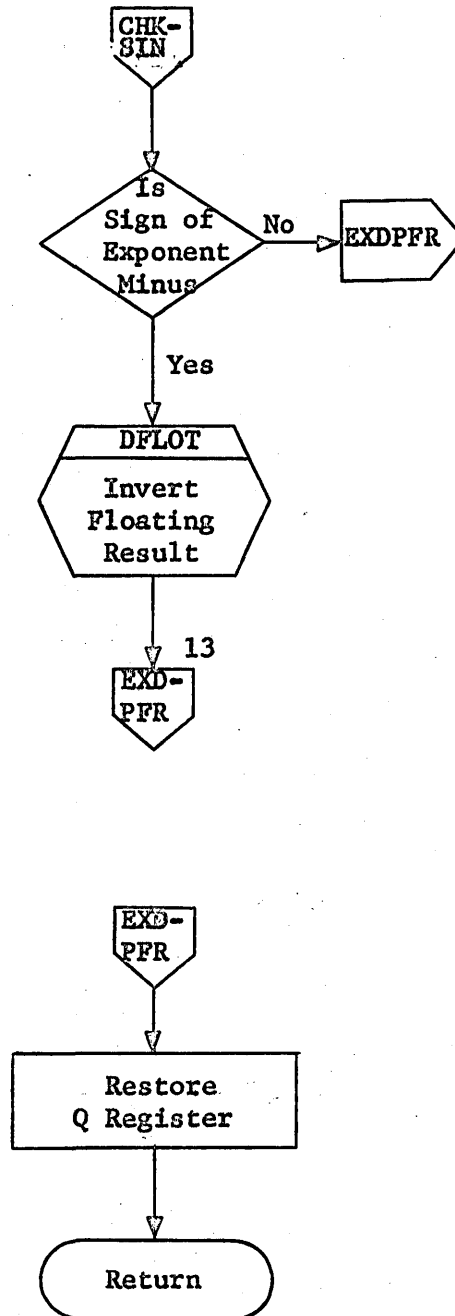
- 11. Take natural log of coefficient.
- 12. Exit





DOCUMENT CLASS IMS PAGE NO. 11-21  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

13. Exit



DOCUMENT CLASS IMS PAGE NO. 11-22  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

APPENDIX A

1700 Double Precision Mathematical Approximations

1.0 INTRODUCTION

This paper will derive approximations for object time 1700 FORTRAN programs which will allow fast evaluations with necessary accuracy and uncomplicated range reduction.

1.1 Accuracy

Since all approximations are to be in floating-point, it shall be required that:

$$R < 2^{-39} = 1.819E-12$$

which is the basic round off due to the size of significant digits in 1700.

2.0 GENERAL DESCRIPTION OF METHOD

The method will use normal truncated Taylor-Mclaurin Power Series.

2.1 Truncated Taylor-Mclaurin Power Series

A function will be expressed as a truncated Taylor-Mclaurin series such that the error produced is less than R. That is,

$$f\{N\} = \sum_{n=0}^m \frac{f\{n\}\{0\}}{n!} x^n, \quad /n/ \leq a$$

and m is chosen such that

$$R < \left| \frac{f\{m+1\}}{(m+1)!} x^{m+1} \right| = R_T$$

DOCUMENT CLASS IMS PAGE NO. 11-23  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

3.0 SIN{X} for  $X < \pi/2$

The Taylor-Mclaurin series is:

$$\text{SIN}\{X\} = \sum_{n=0}^{\infty} \frac{\{-1\}^n X^{2n+1}}{(2n+1)!}$$

Which expands to:

$$\text{SIN}\{X\} = X - \frac{X^3}{3!} + \frac{X^5}{5!} - \frac{X^7}{7!} + \frac{X^9}{9!} - \frac{X^{11}}{11!} + \frac{X^{13}}{13!} - \frac{X^{15}}{15!} + \frac{X^{17}}{17!}$$

4.0 Arctan{X} for  $X \leq \tan \{\pi/16\}$

The Taylor-Mclaurin series is:

$$\text{Arctan}\{X\} = \sum_{n=0}^{\infty} \frac{\{-1\}^n X^{2n+1}}{2n+1}$$

Which expands to:

$$\text{Arctan}\{X\} = X - \frac{X^3}{3} + \frac{X^5}{5} - \frac{X^7}{7} + \frac{X^9}{9} - \frac{X^{11}}{11} + \frac{X^{13}}{13} - \frac{X^{15}}{15} + \frac{X^{17}}{17}$$

5.0 Exponential,  $e^x$  for  $x \leq \frac{\ln 2}{2}$

Taylor-Mclaurin series is a normal series truncated after  $n = 9$ , that is,

$$e^x = \sum_{n=0}^9 \frac{x^n}{n!}$$

Which expanded, gives:

$$e^x = x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} + \frac{x^6}{6!} + \frac{x^7}{7!} + \frac{x^8}{8!} + \frac{x^9}{9!}$$

DOCUMENT CLASS IMS PAGE NO. 11-24  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

Now X and N can always be found such that  $e^y = 2^N e^X$   
 where  $X \leq \frac{\ln 2}{2}$

If X and N are defined as follows

$$\begin{aligned} \text{Let } t &= y/\ln 2 \\ N &= t + 1/2: \text{ integral part} \\ w &= t - N \\ X &= w \ln 2 . \end{aligned}$$

then

$$\begin{aligned} X &= \{t-N\} \cdot \ln 2 = t \cdot \ln 2 - \ln\{2^N\} \\ e^X &= e^y \cdot 2^{-N} \\ \text{and } e^y &= 2^N e^X \end{aligned}$$

6.0

Natural Logarithm X for  $|X| \leq \{3-2\sqrt{2}\}$

The Taylor-Mclaurin series is:

$$\ln\left\{\frac{1+X}{1-X}\right\} = 2 \sum_{n=0}^{\infty} \frac{X^{2n+1}}{2n+1}$$

Which expands to:

$$\ln\left\{\frac{1+X}{1-X}\right\} = 2X + \frac{2X^3}{3} + \frac{2X^5}{5} + \frac{2X^7}{7} + \frac{2X^9}{9} + \frac{2X^{11}}{11} + \frac{2X^{13}}{13} + \frac{2X^{15}}{15} + \frac{2X^{17}}{17}$$

Given  $Y = 2^N Z$  where  $1/2 \leq Z < 1$

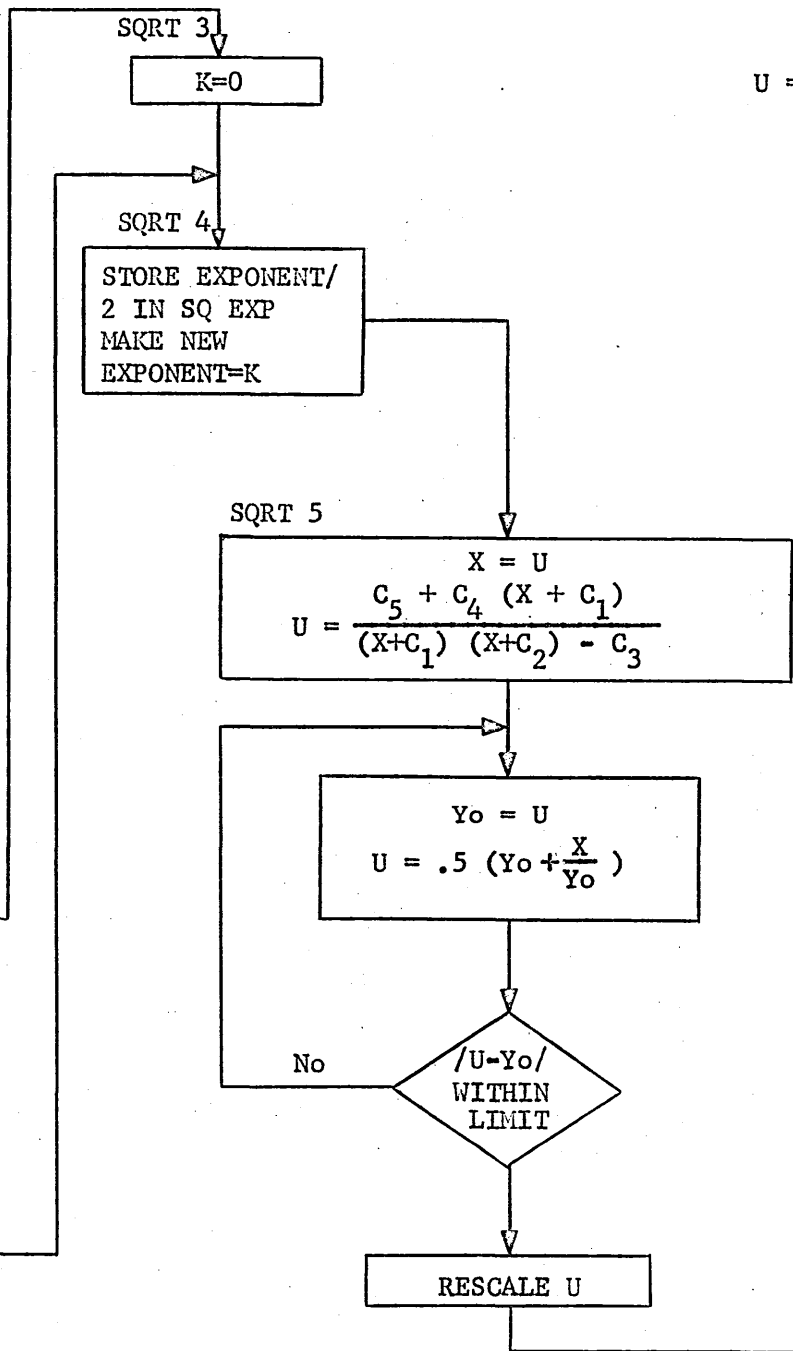
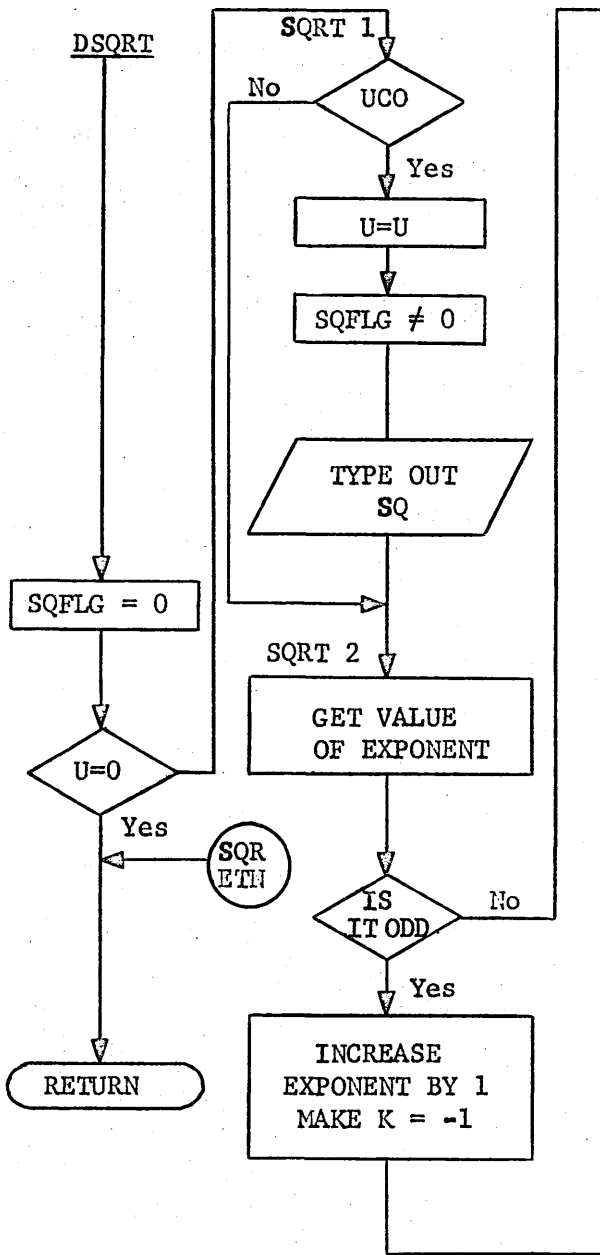
Then  $\ln\{Y\} = N \cdot \ln 2 + \ln Z$

If X is defined as  $X = \frac{Z - \sqrt{2/2}}{Z + \sqrt{2/2}}$

then  $|X| \leq 3 - 2\sqrt{2}$

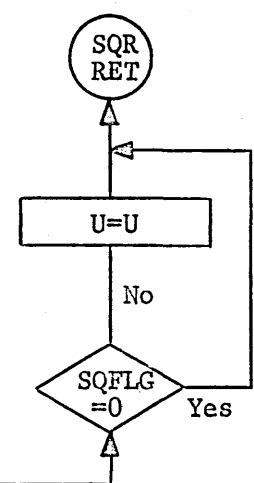
and  $\ln Z = -\frac{\ln 2}{2} + \ln \frac{1+X}{1-X}$

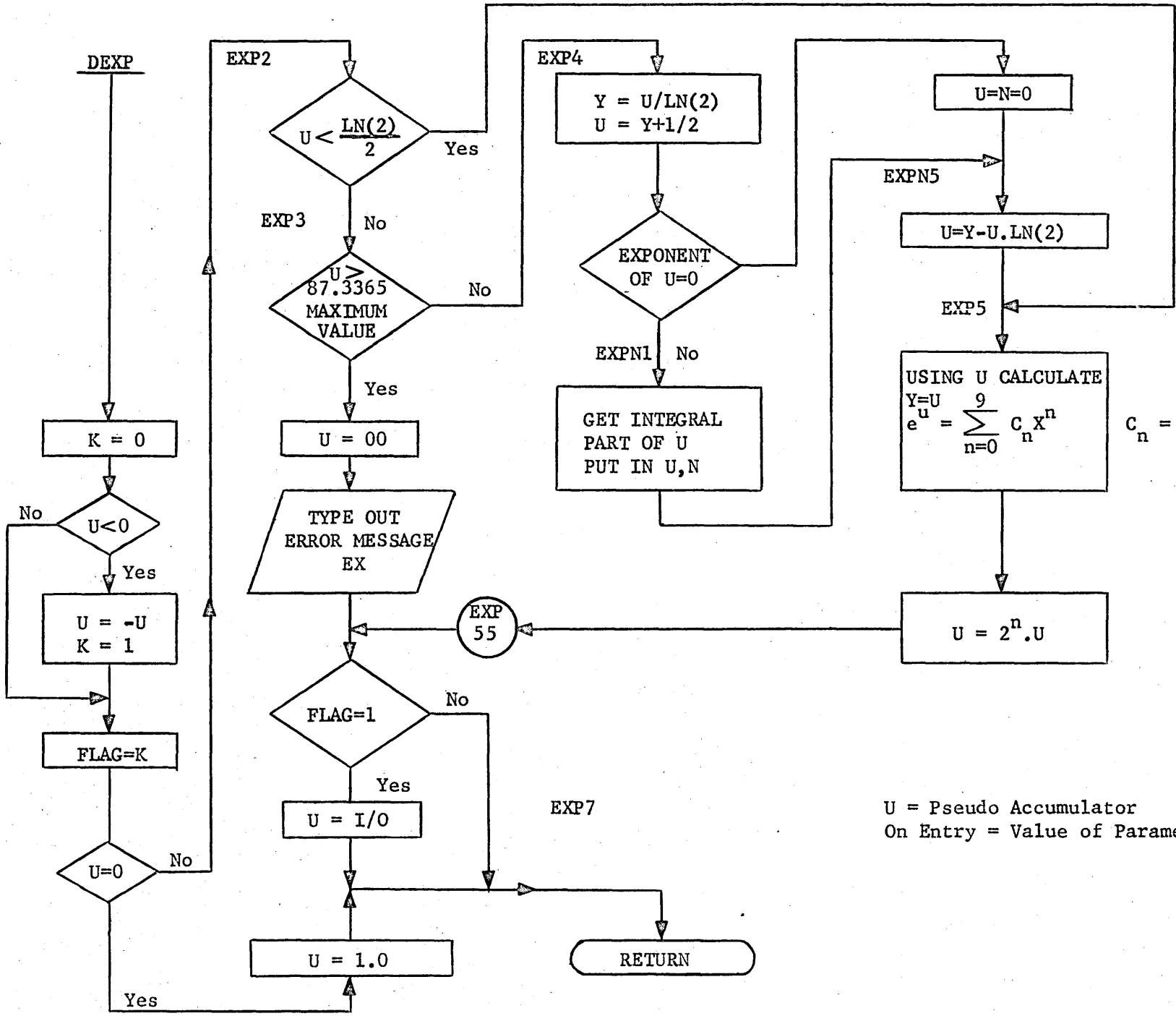
Finally,  $\ln Y = \{N - 1/2\} \cdot \ln 2 + \ln \frac{1+X}{1-X}$



U = PSEUDO ACCUMULATOR

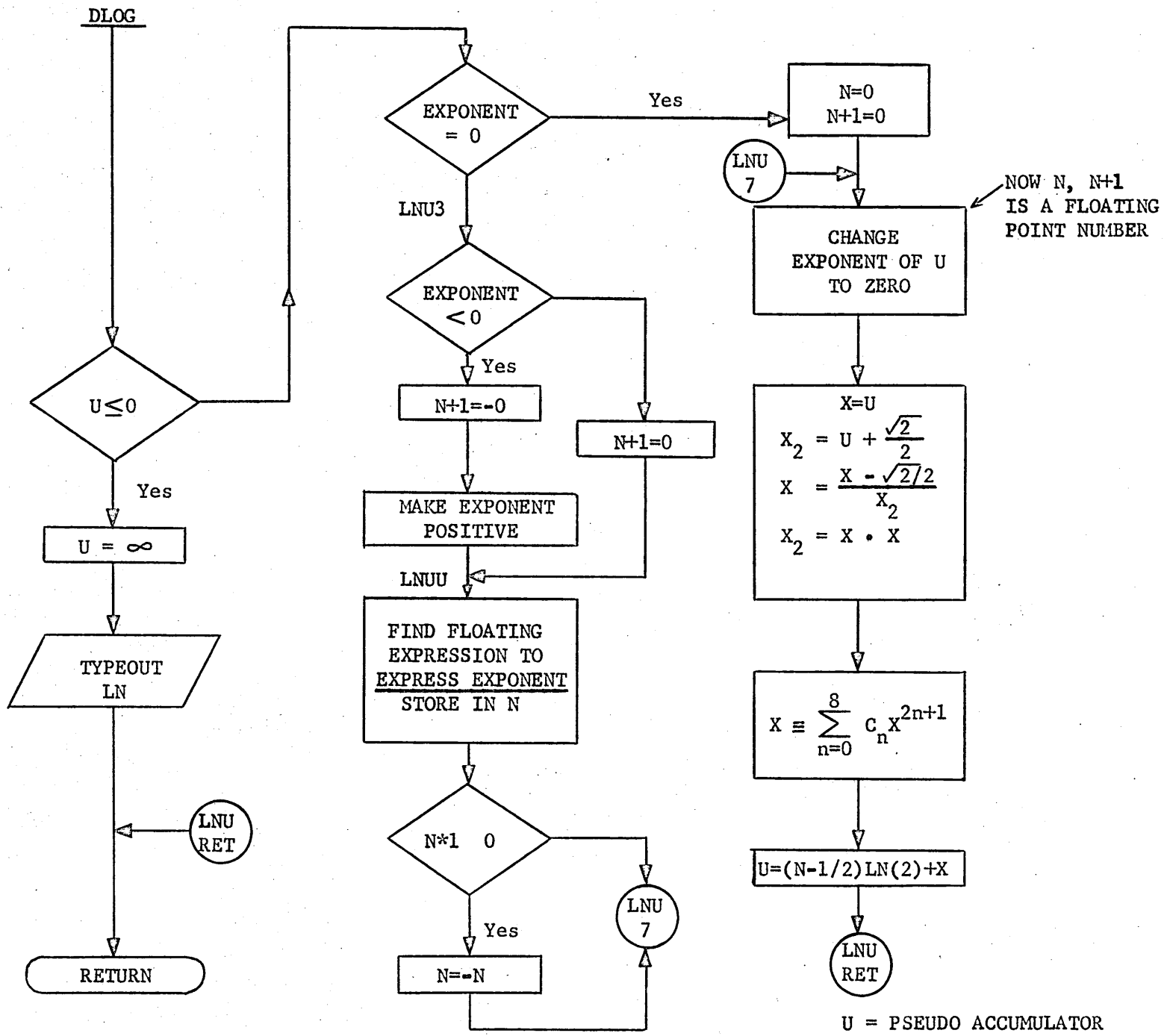
PADE ESTIMATE  
FOR FIRST  
APPROXIMATION  
TO ROOT

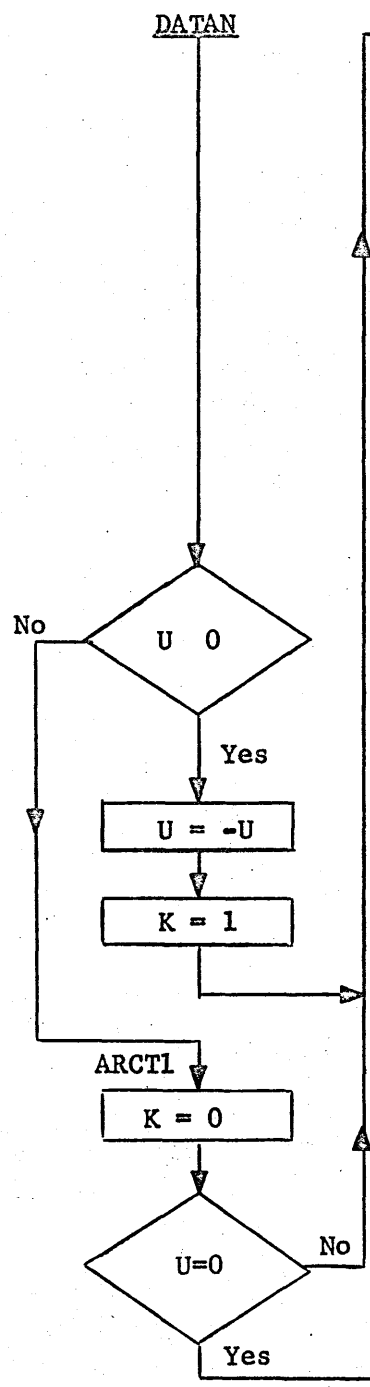
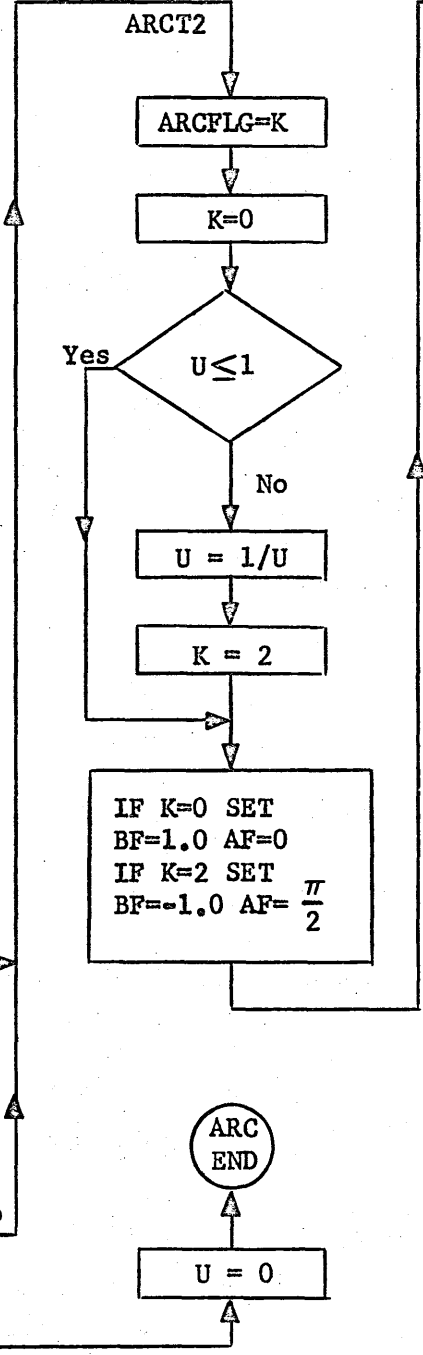
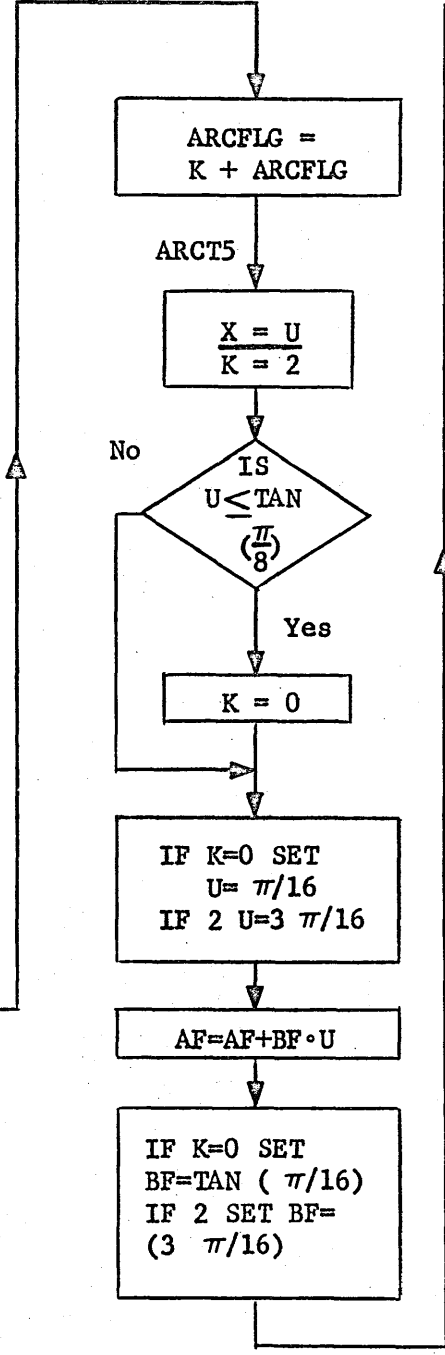
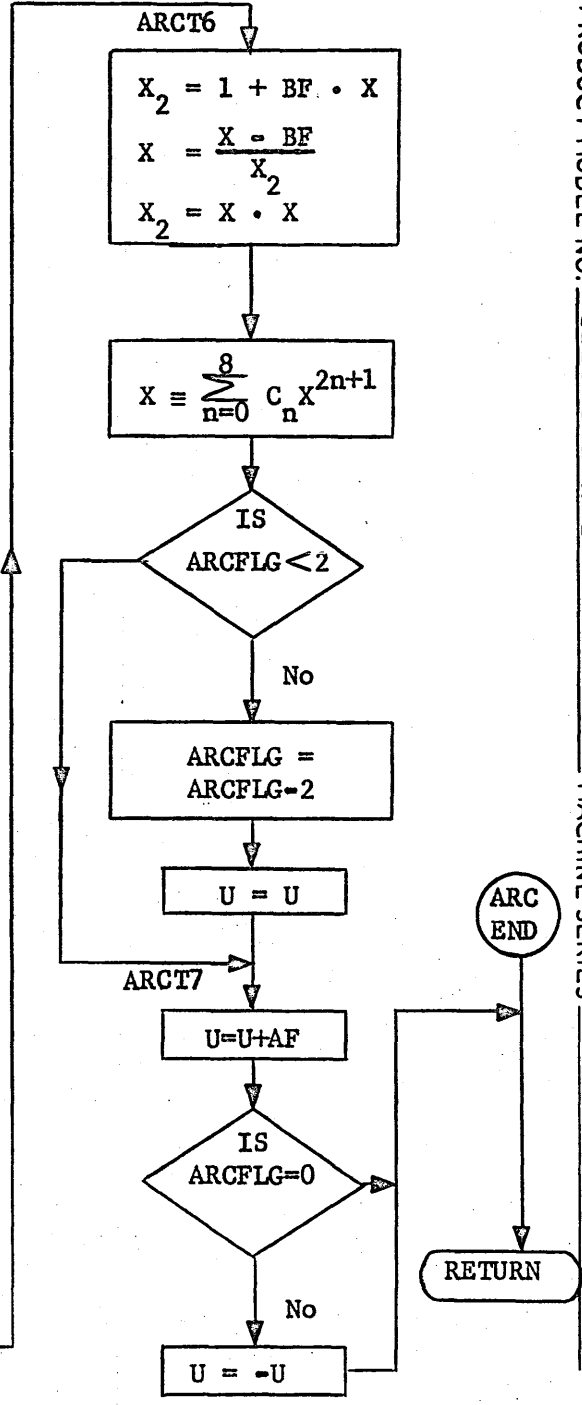




U = Pseudo Accumulator  
On Entry = Value of Parameter

$$C_n = \frac{1}{n!}$$

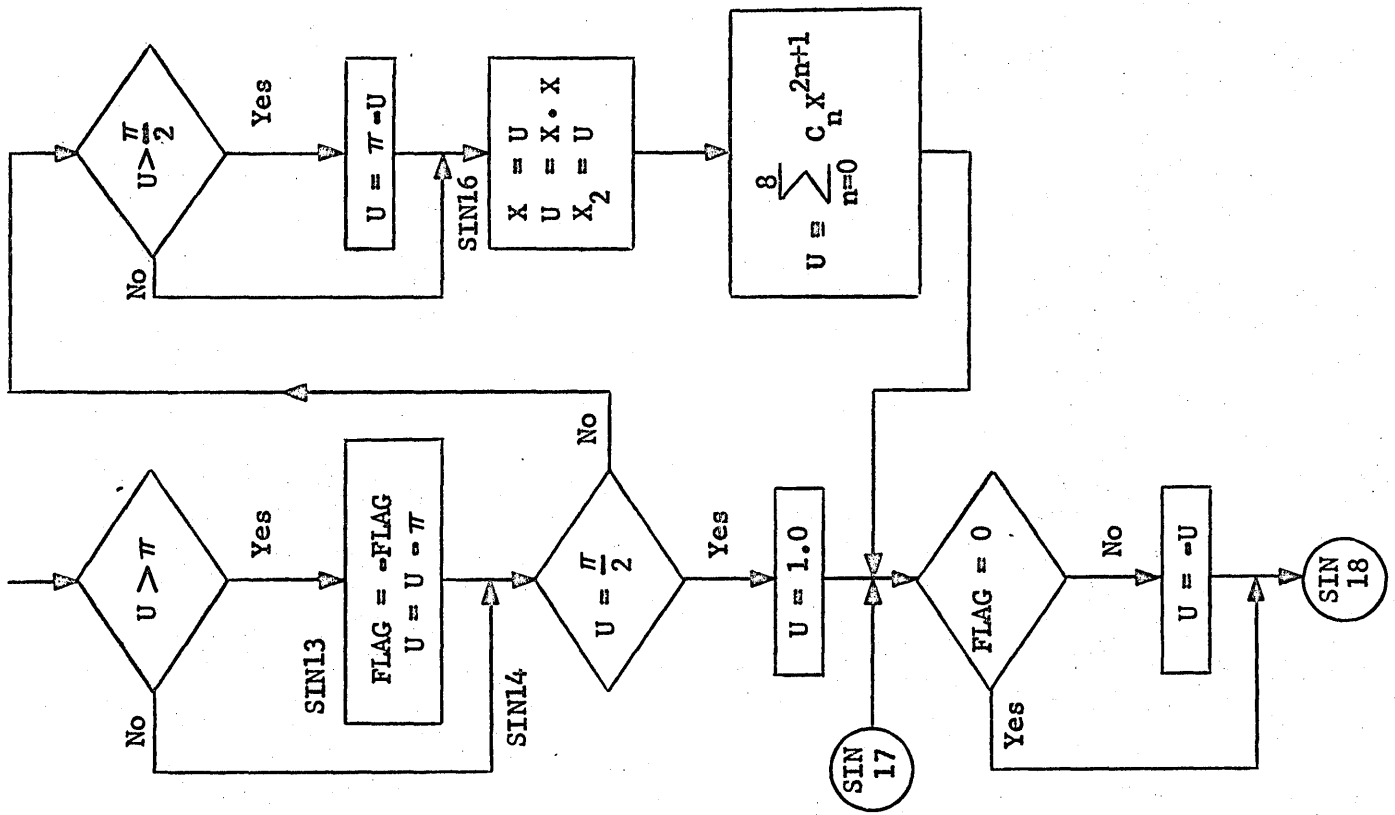




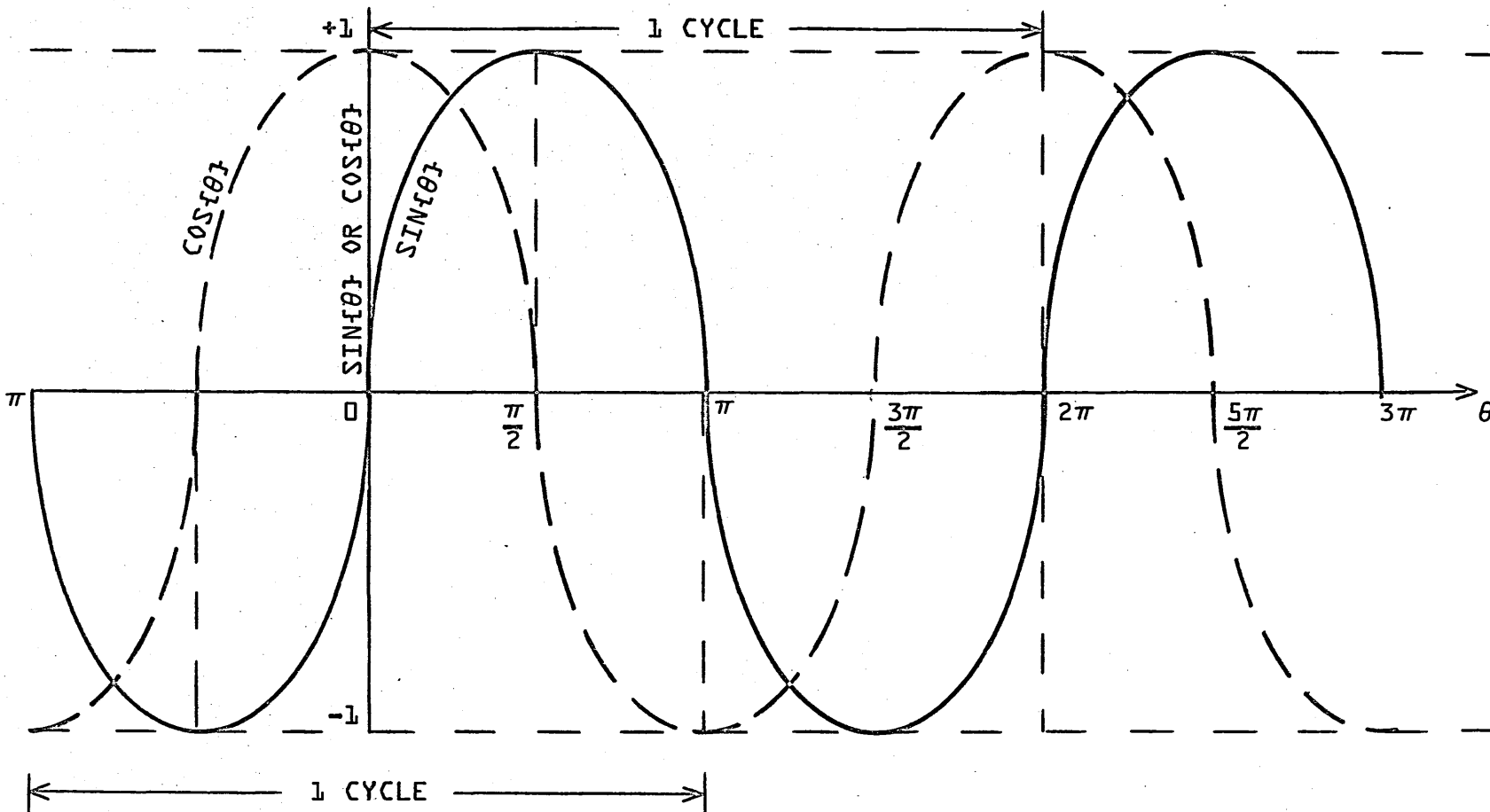




DOCUMENT CLASS IMS PAGE NO. 11-30  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700



DOCUMENT CLASS IMS PAGE NO. 11-31  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. CD05\*3.1 A/B MACHINE SERIES 1700



$$\cos\{\theta\} = \sin\{\theta + \frac{\pi}{2}\}$$

LET  $\theta$  BE INPUT VALUE. IF  $\theta < 0$ ,  $\sin\{\theta\} = -\sin\{|\theta|\}$ .  
 LET  $I$  BE INTEGRAL PART OF  $\theta/2\pi$ . THEN  $\beta = \theta - I \cdot 2\pi$  IS  $< 2\pi$ .  
 IF  $\beta > \pi$  THEN  $\sin\{\beta\} = -\sin\{2\pi - \beta\}$ , LET  $\beta = 2\pi - \beta$   
 IF  $\beta > \pi/2$  THEN  $\sin\{\beta\} = \sin\{\pi - \beta\}$ .



DOCUMENT CLASS IMS PAGE NO. 12-1  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. COO5\*3.1 A/B MACHINE SERIES 1700

TABLE OF CONTENTS

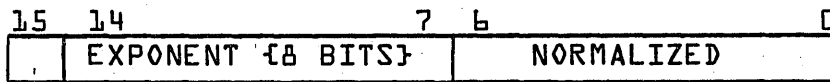
- 12.0 Double Precision Floating Point Package
- 12.1 General Description
  - 12.1.1 Exponent
  - 12.1.2 Coefficient
  - 12.1.3 Calling Sequence
  - 12.1.4 Operations
  - 12.1.5 Operands
    - 12.1.5.1 Absolute Addressing
    - 12.1.5.2 Relative Addressing
    - 12.1.5.3 Example
  - 12.1.6 Fault Condition
- 12.2 Double Precision Floating Point Arithmetic with 39 bit numbers
  - 12.2.1 Introduction
  - 12.2.2 The Four Arithmetic Operations
- 12.3 Low Core Storage Locations and Temporary or Volatile Storage
- 12.4 Entry Points and Externals
- 12.5 Flow chart of DFL0T

DOCUMENT CLASS IMS PAGE NO. 12-2  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. CO05\*3.1 A/B MACHINE SERIES 1700

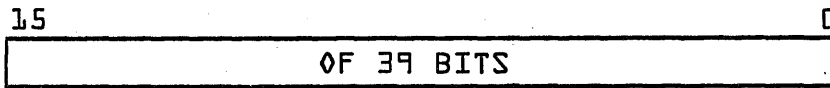
12.0 DOUBLE PRECISION FLOATING POINT PACKAGE

12.1 General Description

Each double precision floating point number requires three consecutive words of 1700 Computer storage. The first word, containing the most significant bits of the number, is the one that is addressed. Normalized floating point format is as follows:



↑  
SIGN OF NUMBER



Thus the numbers,  $X$ , expressible are of the range  $-2^{127}$   $\{1-2^{39}\} \leq X \leq 2^{127} \{1-2^{39}\}$  and are significant to one part in 549 billion. If the most significant word is zero (16 bits of zero or 1), a floating point zero is assumed.

12.1.1 Exponent

The floating point exponent is an 8-bit quantity with a value ranging from 00 to FF<sub>16</sub>. Through biasing by 80<sub>16</sub>, this range expresses both positive and negative exponents.

12.1.2 Coefficient

The coefficient consists of a 39 bit number  $n$ ,  $1-2^{-39} \geq |n| \geq 0$ . The high order bit position of the first word is the coefficient sign bit.

A zero denotes a positive coefficient and a one denotes a negative coefficient. When the coefficient is negative, the entire floating point number is stored in complement form.

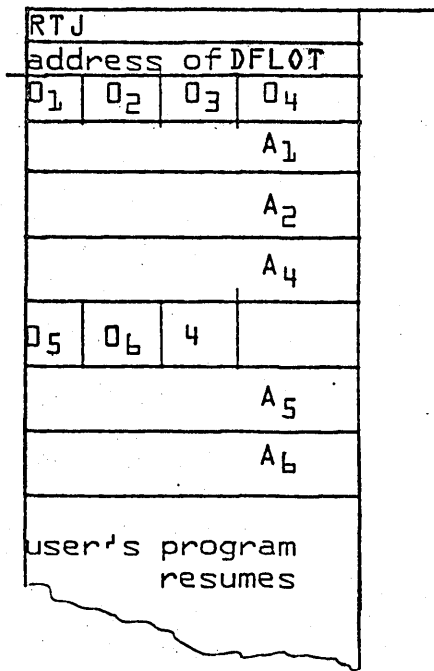
DOCUMENT CLASS IMS PAGE NO. 12-3  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

12.1.3 Calling Sequence

DFLOT uses an interpretive calling sequence. It saves the  $\alpha$ ,  $A_1$ , and  $I$  registers in Temporary Storage; it uses the communications locations  $\$C5$ ,  $\$C6$ , and  $\$C7$  as a pseudo accumulator. The pseudo accumulator is retained in volatile or temporary storage between calls to DFLOT.

The interpretation of the calling sequence is based on a string of 4-bit bytes representing the operations, followed by the operand addresses. The leftmost 4-bit byte is the first operation; the operand addresses, if they exist, follow in the same order as the operation bytes, one word per byte. As many bytes may exist as desired, but the terminating byte must be 4, the operation FEND.

Example:



The  $\alpha_i$ 's are the operation codes; the  $A_i$ 's are their operand addresses. An operand address is not needed by all operations, for instance, operation code  $\alpha_3$  does not have a corresponding  $A_3$ .

DOCUMENT CLASS IMS PAGE NO. 12-4  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. CO05\*3.1 A/B MACHINE SERIES 1700

12.1.4 Operations

The following operations are used by the floating point package.

<u>Operation</u>	<u>4-bit Code</u>	<u>Meaning</u>
CHMD	5	Change of mode operation. All operand addresses following this operation code in the calling sequence are made relative if the preceding addresses were absolute, or absolute if the preceding addresses were relative. Addresses are assumed initially absolute. No operand is needed.
INDX	F <sub>16</sub>	Index. The operand corresponding to INDX is used to increment the operand of the following operations: DFLDD, DFLST, DFADD, DFSUB, DFMPY, and DFDIV.  Each succeeding INDX supercedes the last. No index is initially assumed.
NIDX	6	No index. The succeeding operands do not have indexing increments. NIDX supercedes any preceding INDX, and is superceded by any following INDX. NIDX is assumed initially. No operand is needed.
DFLDD	B <sub>16</sub>	Double Floating load. The floating point number in the corresponding effective operand address is transferred to the pseudo accumulator located in temporary storage.
DFLST	D <sub>16</sub>	Double Floating store. The floating point number is transferred from the pseudo accumulator to the corresponding effective operand address.
DFCOM	7	Double Floating complement. The pseudo accumulator is complemented. No operand is needed.
DFADD	E <sub>16</sub>	Double Floating add. The contents of the effective operand address is added to the pseudo accumulator, and the sum is left in the pseudo accumulator.



CONTROL DATA CORPORATION  
LA JOLLA RESOURCE CENTER \_\_\_\_\_ DIVISION

DOCUMENT CLASS IMS PAGE NO. 12-5  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. CO05\*3.1 A/B MACHINE SERIES 1700

DFSUB	8	Double Floating subtract. The contents of the effective operand address is subtracted from the pseudo accumulator, and the result is left in the pseudo accumulator.
DFMPY	9	Double Floating multiply. The pseudo accumulator is multiplied by the contents of the effective operand address, and the result is left in the pseudo accumulator.
DFDIV	A <sub>16</sub>	Double Floating divide. The pseudo accumulator is divided by the contents of the effective operand address, and the result is left in the pseudo accumulator.
FEND	4 {also 1, 2,3}	End of calling sequence. This operation terminates the calling sequence. No operand needed.

12.1.5 Operands

All operand addresses, absolute or relative, are placed in bits 14-0. Bit 15 is a flag set to distinguish between direct and indirect addresses.

12.1.5.1 Absolute Addressing

An absolute address may be either direct or indirect. All indirect addressing is executed prior to indexing. Only one level of indirect addressing is allowed.

12.1.5.2 Relative Addressing

A relative address may be indirect to one level of indirectness. The relative address is computed by subtracting the location of the address in the calling sequence from the address contained in that location. For instance, if the relative address, A<sub>1</sub>, is contained in location L+3, where L is the first word of the calling sequence, the relative address = A<sub>1</sub> - L + 3.

12.1.5.3 Example

Compute the floating point arithmetic statement:

$$X = -(A\{I\} + B\{I\}) * C\{I\} + D\{J\} * E\{J\}$$

DOCUMENT CLASS IMS PAGE NO. 12-6  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. COO5\*3.1 A/B MACHINE SERIES 1700

Assuming that X, J, D, E, and the temporary location TEMP are absolutely addressed, and the other operands relatively addressed, the generated calling sequence would be:

RTJ		DFLOT	
F <sub>16</sub>	B <sub>16</sub>	9	(6)
absolute address J			
absolute address D			
absolute address E			
D <sub>16</sub>	(5)	F <sub>16</sub>	B <sub>16</sub>
absolute address TEMP			
relative address I			
relative address A			
E <sub>16</sub>	9	(7)	(6)
relative address B			
relative address C			
(5)	E <sub>16</sub>	D <sub>16</sub>	(4)
absolute address TEMP			
absolute address X			

A circle around the operator indicates that the operator has no operand.

**12.1.6 Fault Conditions**

Fault conditions are flagged in a communication location {#C&} whenever they occur during execution. The following bits are set to one:

<u>BIT</u>	<u>Fault Condition</u>
15	Exponent overflow; the exponent in an arithmetic operation exceeds the maximum range limit.

DOCUMENT CLASS IMS PAGE NO. 12-7  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

<u>BIT</u>	<u>Fault Condition</u>
14	Divide fault; division by zero is encountered.
13	Exponent underflow; the exponent in an arithmetic operation is less than the minimum range limit.

If exponent underflow occurs, a floating point zero results. If exponent overflow occurs, the largest word of the appropriate sign results. A divide fault is treated as overflow. These error conditions may be tested with the IFALT function.

12.2 FLOATING-POINT ARITHMETIC WITH 39-BIT NUMBERS

9.2.1 Introduction

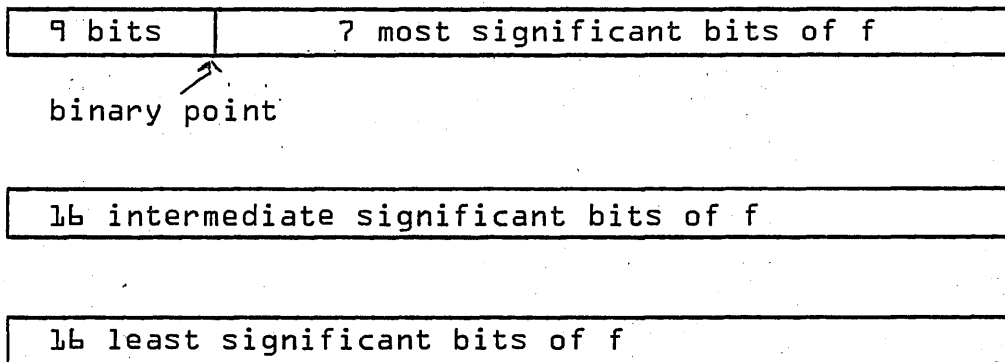
Consider the double precision floating-point number:

$$F = f \cdot 2^{\beta} \quad \{1\}$$

where  $|f|$  lies in the range

$$1/2 \leq |f| \leq 1 - 2^{-39} \quad \{2\}$$

We have a machine with a word length of 16 bits and that the 48 bits in the double length word are divided in the following way:



The leftmost block of nine bits is divided into three parts:

1. The first {leftmost} bit represents the sign of F.
2. The second bit represents the sign of  $\beta$ .

DOCUMENT CLASS IMS PAGE NO. 12-B  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

3. The next 7 bits represent the magnitude of f. This allows 39 bits for the representation of f. We shall assume that the binary point lies at the left and of the 39 bits representing f so that the 7 most significant bits of f are stored in the first word of the three, and the 16 least significant bits of f are stored in the third word of the three.

If we write

$$|f| = c + c_i \times 2^{-7} + d \times 2^{-23} \quad \{3\}$$

where c lies in the range

$$1/2 \leq c \leq 1 - 2^{-7} \quad \{4\}$$

$c_i$  lies in the range

$$0 \leq c_i \leq 1 - 2^{-16} \quad \{5\}$$

and where d lies in the range

$$0 \leq d \leq 1 - 2^{-16} \quad \{6\}$$

then c represents the 7 most significant bits of f,  $c_i$  represents the 16 intermediate significant bits of f, and d represents the 16 least significant bits of f.

We wish to consider the four basic arithmetic operations using double precision floating point numbers of the form discussed. Consequently, in order to have notation for two operands, let us consider a second double precision floating point number.

$$G = g \cdot 2^9, \quad \{7\}$$

where |g| lies in the range

$$1/2 \leq |g| \leq 1 - 2^{-39} \quad \{8\}$$

If we write

$$|g| = a + a_i \times 2^{-7} + b \times 2^{-23} \quad \{9\}$$

where a lies in the range

$$1/2 \leq a \leq 1 - 2^{-7} \quad \{10\}$$

DOCUMENT CLASS IMS PAGE NO. 12-8A  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

$a_i$  lies in the range

$$0 \leq a_i \leq 1 - 2^{-1b} \quad \{11\}$$

and  $b$  lies in the range

$$0 \leq b \leq 1 - 2^{-1b} \quad \{12\}$$

then  $a$  represents the 7 most significant bits of  $|g|$ ,  $a_i$  represents the intermediate significant bits of  $|g|$ , and  $b$  represents the  $1b$  least significant bits of  $|g|$ .

The machine represents negative numbers using a "one's-complement" system. In fact the procedure for changing the sign of a floating point number is to perform a bit-by-bit complement of the entire 48 bits (including the 9 bits representing the sign and exponent).

12.2.2

THE FOUR ARITHMETIC OPERATIONS

A. Addition

$$F + G = f \times 2^\beta + g \times 2^\delta \quad \{13\}$$

The basic problem in floating point addition is to adjust the exponent of  $F$  (or  $G$ ) so that the binary points are lined up before the addition takes place.

Let us introduce the notation  $L$  to represent three cells containing the larger of the two numbers  $F$  and  $G$ , and the notation  $S$  to represent three cells containing the smaller of the two numbers. We shall say that  $F$  is larger than  $G$  if

$$\beta \geq \delta \quad \{14\}$$

and  $F$  is smaller than  $G$  if

$$\beta < \delta \quad \{15\}$$

We are not concerned about the relative magnitudes of  $f$  and  $g$  in case the exponents are equal. Using this convention, we process the following algorithm for forming  $F + G$ :

1. Record the leftmost nine bits of  $F$  and  $G$ . This, in effect, records  $\beta$  and  $\delta$ .

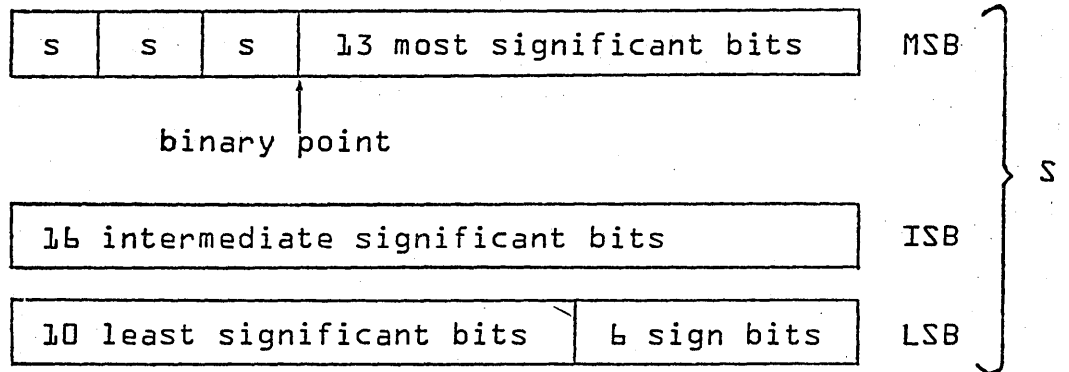
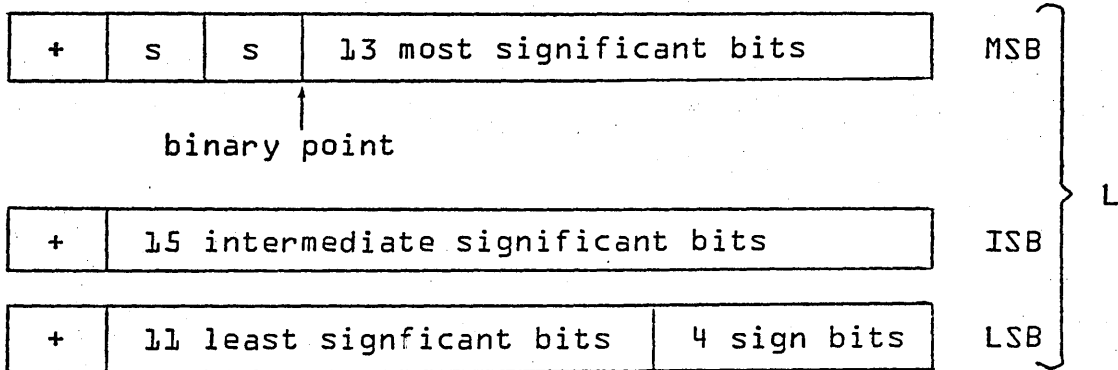
C

O

O

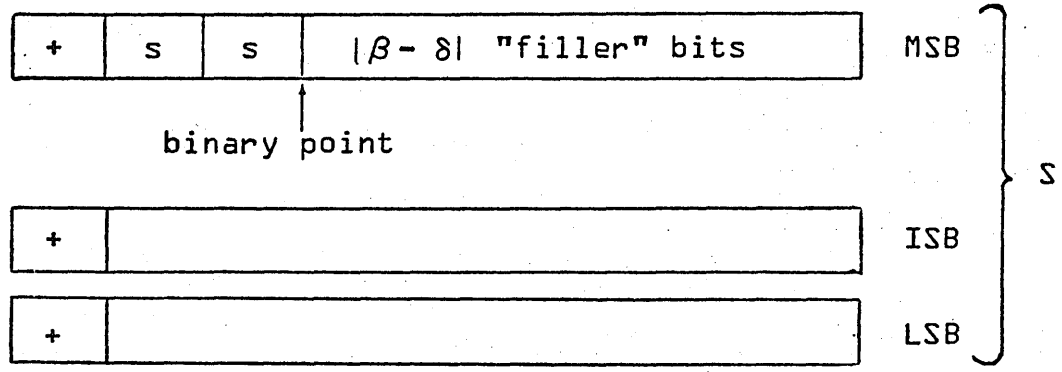
DOCUMENT CLASS IMS PAGE NO. 12-9  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

2. Determine the sign of  $\{\beta - \delta\}$  and thus determine whether F is smaller or larger than G according to  $\{14\}$  and  $\{15\}$ .
3. Place f and g in L and S. If F is larger than G, then f goes into L; otherwise, f goes into S and g goes into L. The following bit patterns should be formed (here s means "sign bit"):



4. Shift S right  $|\beta - \delta|$  places and put a "+" bit at the beginning of each of the three words. If  $|\beta - \delta| = 39$ , then there is no need to continue since all significant bits in S will be lost.

DOCUMENT CLASS IMS PAGE NO. 12-10  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. CD05\*3.1 A/B MACHINE SERIES 1700



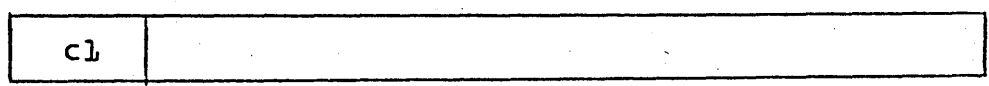
Notice that the  $|\beta - \delta|$  "filler" bits, between the binary point in S and the most significant bit of the function, are sign bits. This is mathematically correct in a one's complement representation of negative numbers.

- 5. Add the LSB portions of L and S.



We call the first bit of this sum c. If it is a one, we actually have a carry. However, it usually is easier to add c (see step 6) than to test to see whether or not it needs to be added as a carry bit in forming the sum of the ISB portions of L and S.

- 6. Add the ISB portions of L and S and the carry bit from step 5.



If c1 is set to one, we have a carry and we will add c1 to step 7 in forming the sum of the most significant bits of L and S.

- 7. Add the MSB portions of L and S and add the carry bit obtained from step 6.





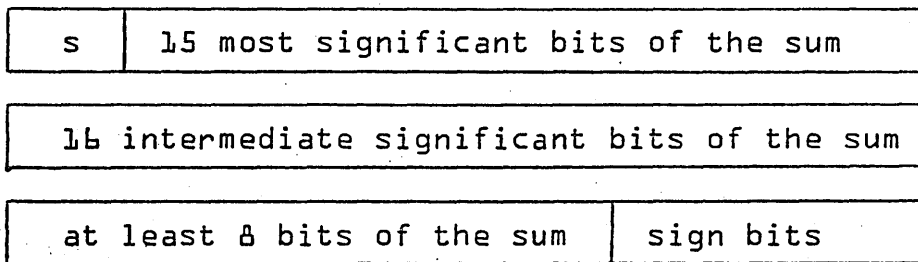
DOCUMENT CLASS IMS PAGE NO. 12-11  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 -A/B MACHINE SERIES 1700

If  $e = 1$  then an end-around-carry must be performed. This means that a one is added at the right end of the word produced in step 5. Since this might also produce a carry bit, the  $c$  in the diagram {see step 5} must be cleared to zero before the end-around-carry. If a carry bit is again produced, then a one must be added at the right end of the word produced in step 6. Since this might also produce a carry bit, the  $c_1$  in the diagram {see step 6} must be cleared to zero before the end-around-carry. If a carry bit is again produced, then a one must be added at the right end of the word above. It can be shown that this last operation can never produce another  $e = 1$ .

If  $v = s$  then  $v$  is a sign bit. However, if  $v \neq s$  then there has been overflow during the addition, and the  $v$  is the most significant bit of the sum. In the latter case an adjustment of the exponent will be necessary to give the correct answer.

8. Shift the LSB portion of the sum left one place to clear out the carry bit  $c$ . Then shift the ISB portion of the sum left one place to clear out the carry bit  $c_1$ . Then shift the LSB portion of the sum one place and put the bit shifted off into the right most bit of the ISB portion of the sum. Then shift the triple length sum left {a} one place if  $v \neq s$ ; {b} two places if  $v = s$ .

This leaves the sum in the following form:



9. If the triple length sum was shifted one place left in step 7 { $v \neq s$ } then the exponent must be adjusted to take care of the overflow. This means adding one to the exponent  $\beta$  or  $\delta$ , whichever is larger. {This will be the exponent of the sum.} If the double length sum was shifted two places left in step 7, no adjustment of exponent is necessary.

DOCUMENT CLASS IMS PAGE NO. 12-12  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. COO5\*3.1 A/B MACHINE SERIES 1700

10. The form of the sum given by step 8 must be checked for normalization since it is possible that several of the leading bits of the sum may be zero. {Cancellation occurs when two numbers of opposite sign but nearly equal magnitude are added.} If the sum is not normalized at this point, appropriate adjustments in the exponents should be made.

If 39 left shifts are not sufficient for normalization, then the sum should be made zero.

11. At this point the normalized sum may be rounded although the extra coding involved may not be worth the gain. If rounding is desired, then there are two cases to be considered depending on the sign of the sum. These cases require that care be taken in handling any carry bit produced by the rounding operations.

12. Now pack the 39 most significant bits of the sum, along with 9 bits representing the sign and exponent, into three 16-bit words {in the standard way}. If the sign of the sum is negative, then the first 9 bits must be complemented before the packing takes place.

#### B. Subtraction

No special subroutine is necessary since

$$F - G = F + \{-G\}$$

and one merely complements G before entering the addition subroutine.

#### C. Multiplication

$$F \times G = \{f \times 2^\beta\} \{g \times 2^\delta\} \quad \{16\}$$

$$= \{\text{sign } F \times G\} |f| \times |g| \times 2^{\beta+\delta}$$

The computational procedure is primarily concerned with the formation of  $|f| \times |g| \times 2^{\beta+\delta}$  since  $\{\text{sign } F \times G\}$  can be recorded in advance and used later to apply the correct sign to the product. In addition to recording  $\{\text{sign } F \times G\}$ , we record the exponents  $\beta$  and  $\delta$  after the product  $|f| \times |g|$  is formed. In fact, we propose the following algorithm for multiplying F by G:

DOCUMENT CLASS IMS PAGE NO. 12-13  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

1. Determine and record {sign F x G}
2. Form |F| and |G|
3. Record the leftmost nine bits of |F| and |G|. This, in effect, records  $\beta$  and  $\delta$ .
4. Shift the 39 bits of |f| and |g| left until each has the bit pattern.

+	15 most significant bits	C and A
0	15 intermediate significant bits	C <sub>i</sub> and A <sub>i</sub>
0	9 least significant bits	D and B
	6 zeros	

If this procedure is followed, f is no longer represented by {3} during the computation in step 5, below but has the form:

$$f = C + C_i \times 2^{-15} + D \times 2^{-30} \quad \{17\}$$

where C, C<sub>i</sub>, and D lie in the following range

$$2^{-1} \leq C \leq 1 - 2^{-15} \quad \{18\}$$

$$0 \leq C_i \leq 1 - 2^{-15} \quad \{19\}$$

$$0 \leq D \leq 1 - 2^{-9} \quad \{20\}$$

likewise |g| has the form

$$|g| = A + A_i \times 2^{-15} + B \times 2^{-30} \quad \{21\}$$

where A, A<sub>i</sub>, and B lie in the following range

$$2^{-1} \leq A \leq 1 - 2^{-15} \quad \{22\}$$

$$0 \leq A_i \leq 1 - 2^{-15} \quad \{23\}$$

$$0 \leq B \leq 1 - 2^{-9} \quad \{24\}$$

DOCUMENT CLASS IMS PAGE NO. 12-14  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

5. Use fixed point operations in forming the product.

$$\begin{aligned}
 f \times g &= \{C + C_i \times 2^{-15} + D \times 2^{-30}\} \{A + A_i \times 2^{-15} + B \times 2^{-30}\} \\
 &= CA + \{CA_i + C_iA\} \times 2^{-15} + \{DA + C_iA_i + CB\} \times 2^{-30} \\
 &\quad + \{DA_i + C_iB\} \times 2^{-45} + DB \times 2^{-60} \\
 &= CA + \{CA_i + C_iA\} \times 2^{-15} + \{DA + C_iA_i + CB\} \times 2^{-30} \\
 &\hspace{15em} \{25\}
 \end{aligned}$$

Notice that the terms  $\{DA_i + C_iB\} \times 2^{-45}$  and  $DB \times 2^{-60}$  may be ignored because once the product is placed back in standard form, only 39 bits are retained. The following computational steps are performed.

- a. Form CA giving a double length product.
- b. Form DA and retain the most significant half of the double length product.
- c. Form  $C_iA_i$  and retain the most significant half of the double length product.
- d. Form  $C_iA$  giving a double length product.
- e. Add the most significant half of DA to the most significant half of  $C_iA_i$ .
- f. Form CB and retain the most significant half of the double length product.
- g. Add the most significant half of DB to the sum obtained in {e}.
- h. Add the least significant half of  $C_iA$  to the sum obtained in {g}.
- i. Form  $CA_i$  giving a double length product.
- j. Add the least significant half of  $CA_i$  to the sum obtained in {h}. This result is the least significant portion of the triple length product.
- k. Add the most significant half of  $C_iA$  to  $CA_i$ .
- l. Add the least significant half of CA to the sum obtained in {k}. This result is the intermediate significant portion of the triple length product. The first half of the double length product is the most significant half of CA which was formed above in {a}.

DOCUMENT CLASS IMS PAGE NO. 12-15  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

- b. Next, round and normalize the product obtained using {25} in step 5. Any adjustment in the exponent  $\beta + \delta$  which is necessary because of the normalization of {f} x {g} must be performed.
- 7. Finally, pack the 39 bits of the normalized product and the 9 bits representing the sign and the adjusted exponent into three 16 bit words {in the standard way}. If the {sign F.G} is negative, the two words must then be complemented to give the correct sign to the product.

D. Division

$$\begin{aligned} \frac{G}{F} &= G \times \frac{1}{F} = \{g \times 2^\delta\} \times \left\{ \frac{1}{f \times 2^\beta} \right\} \\ &= \{\text{Sgn } G \times F\} \times |g| \times \left| \frac{1}{f} \right| \times 2^{\delta-\beta} \end{aligned} \quad \{26\}$$

As a matter of fact, since we want

$$\left| \frac{g}{f} \right| < 1, \quad \{27\}$$

We scale the numerator and write

$$\frac{G}{F} = \{\text{Sgn } G \times F\} \times |g| \times \left| \frac{1}{f} \right| \times 2^{\delta-\beta+1} \quad \{28\}$$

Thus we propose the following algorithm for dividing G by F:

1. Determine and record {Sgn F x G}.
2. Form {F} and {G}.
3. Record the leftmost eight bits of {F} and {G}. This, in effect, records  $\beta$  and  $\delta$ .
4. Arrange the 39 bits of {f} to give the bit pattern:

+	15 most significant bits of {f}	A
---	---------------------------------	---

0	15 intermediate significant bits of {f}	Ai
---	---	----

DOCUMENT CLASS IMS PAGE NO. 12-16  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

0	9 least significant bits	6 zeros	B
---	--------------------------	---------	---

and the 39 bits which represent the number 1.0 to give the bit pattern:

+	14 most significant bits of 1/2	\$2000	α
---	---------------------------------	--------	---

0	15 intermediate significant bits of 1/2	\$0000	γ
---	---	--------	---

0	10 least significant bits of 1/2	5 zeros	ε
---	----------------------------------	---------	---

5. Use fixed point operations in forming the quotient.

$$\frac{1}{A + A_i \times 2^{-15} + B \times 2^{-30}} = \frac{\alpha + \gamma \times 2^{-15} + \epsilon \times 2^{-30}}{A + A_i \times 2^{-15} + B \times 2^{-30}}$$

where: α = \$2000  
 γ = \$0000  
 ε = \$0000

$$= \frac{\alpha}{A} + \frac{1}{A} \left\{ \gamma - \alpha \frac{A_i}{A} \right\} \times 2^{-15} + \frac{1}{A} \left\{ \epsilon - \frac{\alpha B}{A} \right\} \times 2^{-30} - \frac{A_i}{A^2} \left\{ \gamma - \frac{\alpha A_i}{A} \right\} \times 2^{-30}$$

$$= \frac{1}{A} \left[ \alpha - \frac{\alpha}{A} [A_i \times 2^{-15} + \{B - \frac{A_i^2}{A}\} \times 2^{-30}] \right]$$

{29}

DOCUMENT CLASS IMS PAGE NO. 12-17  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. COO5\*3.1 A/B MACHINE SERIES 1700

Any terms beginning with  $2^{-45}$  are ignored because only 39 bits of the quotient are retained. The following computational steps are performed:

- a. Form  $- Ai^2$  giving a double length product.
- b. Divide  $- Ai^2$  {as a double length dividend} by A.
- c. Form  $B - \frac{Ai^2}{A}$  {rounded to single length}.
- d. Form the double length dividend
 
$$Ai \times 2^{-15} + \left[ \left\{ B - \frac{Ai^2}{A} \right\} \times 2^{-30} \right]$$
 {The sign of the second term makes this tricky}.
- e. Divide the double length dividend by A and multiply the result by  $\alpha$ . The multiply is accomplished by shifting the result of the divide.
- f. To obtain the second half of the double length quotient, the remainder resulting from the division in the previous step must now be divided by A.
- g. Form  $\alpha$  {the most significant bits of  $1/2$ } and the result obtained from step e as a double length dividend.
- h. Divide the double length dividend by A. The result is the most significant bits of the quotient.
- i. Form the remainder of step h and the result of step f as a double length dividend.
- j. Divide the double length dividend by A. The result is the intermediate significant bits of the quotient.
- k. Divide the remainder obtained in step j by A. The result is the least significant bits of the quotient.
- l. Next, round and normalize the 3 word quotient using {29} and the procedure of step 5. Any adjustment in the exponent which is necessary because of normalization of  $\left| \frac{1}{2} \right|$  must be performed.
- m. The 3 word quotient is then multiplied by  $|g|$ .

DOCUMENT CLASS IMS PAGE NO. 12-18  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

- n. Next, round and normalize the product. Any adjustment in the exponent  $\beta + \delta$  which is necessary due to the normalization must be performed.
- o. Finally pack the 39 bits of the normalized quotient and the 9 bits representing the sign and the exponent into three 16 bit words {in the standard way}. If  $\{Sgn F \times G\}$  is negative, the three words must then be complemented to give the correct sign to the quotient.

REFERENCES

Gregory, Robert T. and Raney, James L.: Floating Point Arithmetic with 84-Bit Numbers, Communications of the ACM, Volume 1/ Number 1/ January, 1964.

12.3

Low Core Storage Location and Temporary or Volatile Storage

DFLOT sets the low core locations  $\#C5$  through  $\#C8$ ; they are assigned as follows:

Location Name	Location	Use
G G+1 G+2	$\#C5$ $\#C6$ $\#C7$	Pseudo accumulator - the three words of the double precision floating point number are placed here.
ERRORS	$\#C8$	Error bits to indicate exponent overflow, divide fault, and exponent underflow. These error bits are used by IFALT to check for these floating point errors.

There are two versions of DFLOT. The two packages are called by the same name {DFLOT}, but one is reentrant and the other non-re-entrant. Both packages are usable by run-anywhere programs. The re-entrant package must operate in protected core. The non re-entrant package may operate anywhere.

The non-reentrant version of DFLOT utilizes temporary storage to perform its computations. The re-entrant version of DFLOT utilizes. The volatile storage concept for temporary storage



CONTROL DATA CORPORATION  
LA JOLLA RESOURCE CENTER

DIVISION

DOCUMENT CLASS IMS PAGE NO. 12-19  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. COO5\*3.1 A/B MACHINE SERIES 1700

The non-re-entrant version of DFLOT requires the deck labeled DUMVOL which contains entry points AVOLA and AVOLR. This deck provides the simulated volatile storage or temporary storage required for DFLOT in the background.

DFLOT uses 30 volatile or temporary storage locations; they are assigned as follows:

Location Name	Volatile or Temporary Storage Location	Use
	0,I	Contents of Q register
	1,I	Contents of A register
	2,I	Contents of I register
G	3,I	Pseudo Accumulator - the three words of the double precision floating point number are placed here. The pseudo accumulator is broken up into C, CI, D, and DELTA
G+1	4,I	
G+2	5,I	
SIGN	6,I	Sign register - sign double precision floating point number
ERRORS	7,I	Error bits Bit 15 indicates exponent overflow Bit 14 indicates divide fault {divide by zero} Bit 13 indicates exponent underflow. These are used by IFALT to check for these floating point errors.
F	8,I	1. Address of parameter or 2. First word of a {three word} double precision floating point number. It is used when two operands are needed, e.g. for an add. In this case F, F+1, and F+2 are broken up into A, AI, B, BETA.

CONTROL DATA CORPORATION  
LA JOLLA RESOURCE CENTER

DIVISION

DOCUMENT CLASS IMS PAGE NO. 12-20  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

Location Name	Volatile or Temporary Storage Location	Use
F+1	9.I	The second word of a {three word} double precision floating point number.
F+2	10.I	The third word of a {three word} double precision floating point number.
A	11.I	The most significant bits of the coefficient of the double precision floating point number.
AI	12.I	The intermediate significant bits of the coefficient of the double precision floating point number.
B	13.I	The least significant bits of the coefficient of the double precision floating point number.
BETA	14.I	Exponent of the double precision floating point number.
C	15.I	Most significant bits of the coefficient of the double precision floating point number.
CI	16.I	Intermediate significant bits of the double precision floating point number.
D	17.I	Least significant bits of the coefficient of the double precision floating point number.
DELTA	18.I	Exponent of double precision floating point number.
SHIFCT	19.I	Shift counter

CONTROL DATA CORPORATION  
LA JOLLA RESOURCE CENTER

DIVISION

DOCUMENT CLASS IMS PAGE NO. 12-21  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

Location Name	Volatile or Temporary Storage Location	Use
P	20,I	Pseudo program address counter which is used by DFL0T to know where it is in the users program when it wants to pick up more parameters from the variable calling sequence or return to the calling program.
RELADR	21,I	Absolute/relative indication RELADR { = 1 relative address = 0 absolute address
OPCNT	22,I	Operation counter - keeps track of which operation code is being processed.
INDEX	23,I	Effective address of parameter
OPCODE	24,I	Operation code holder - holds the operation {op} codes contained within the call.
QS	25,I	Saves Q register upon entry into DFL0T
TEMP0	26,I	Temporary Storage for many purposes
TEMP2	27,I	Temporary Storage for may purposes
T1		Temporary Storage for MSB of 1/Divisor
T2		Temporary Storage for ISB of 1/Divisor
T3		Temporary Storage for LSB of 1/Divisor
MULDIV		Multiply/Divide Flag; = 1 for Divide; = 0 for multiply.

DOCUMENT CLASS IMS PAGE NO. 12-22  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. COO5\*3.1 A/B MACHINE SERIES 1700

12.4 Entry Points and Externals

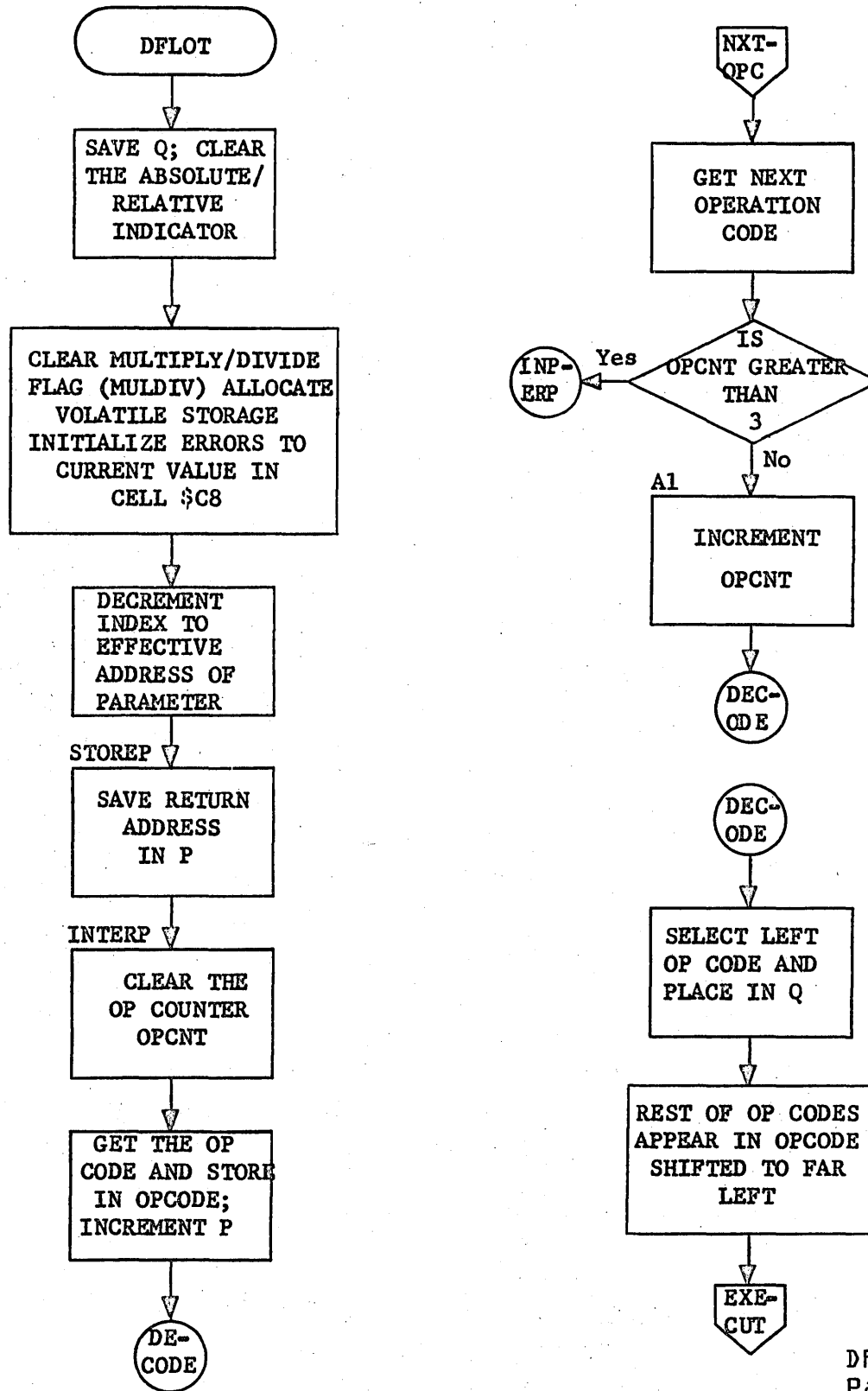
The routine DFL0T has one entry point, DFL0T. DFL0T has two externals: AV0LA and AV0LR.

12.5 Flowchart of DFL0T

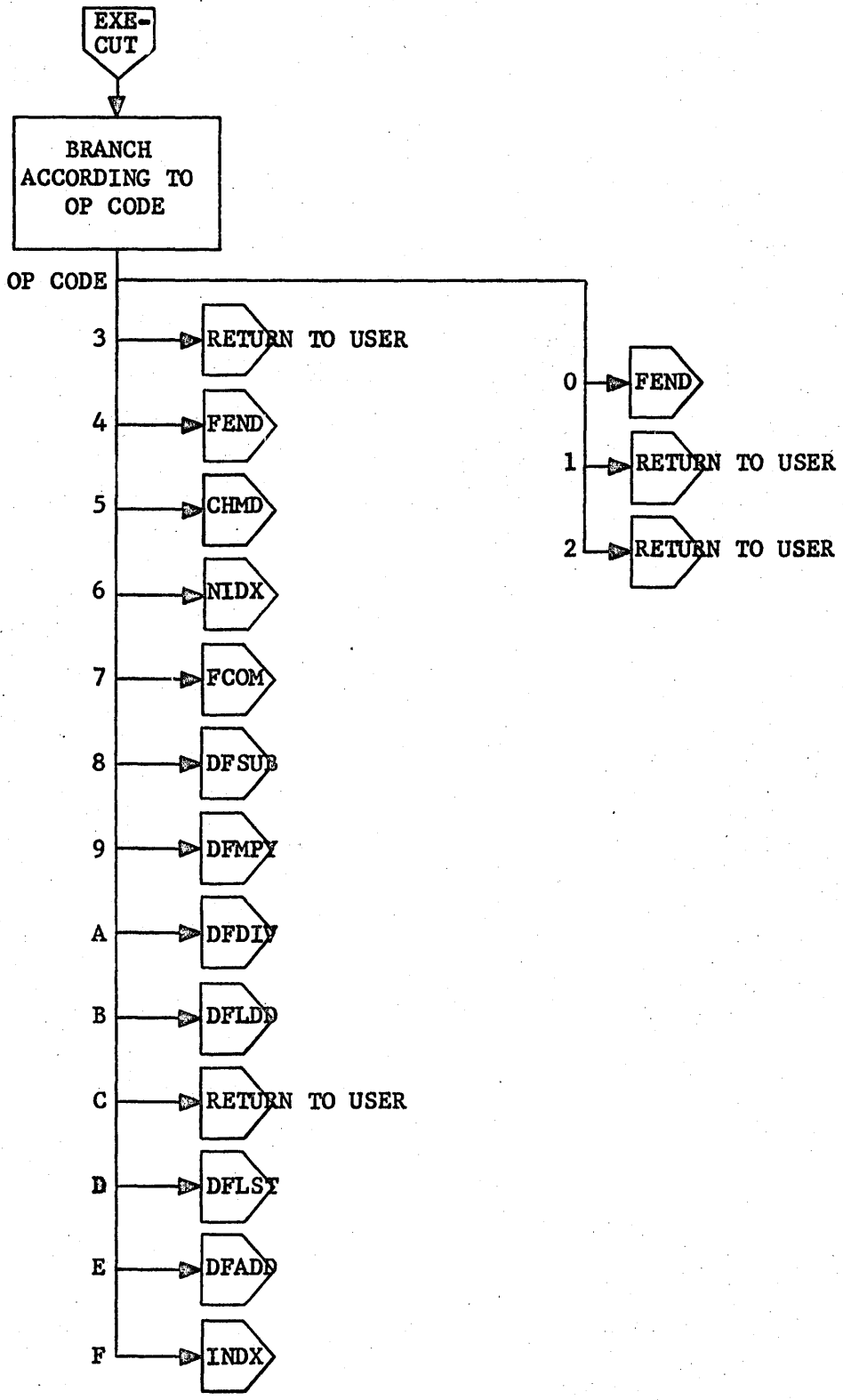
The following abbreviations are used:

MSB = most significant bits  
ISB = intermediate significant bits  
LSB = least significant bits

DOCUMENT CLASS IMS PAGE NO. 12-23  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. COO5\*3.1 A/B MACHINE SERIES 1700

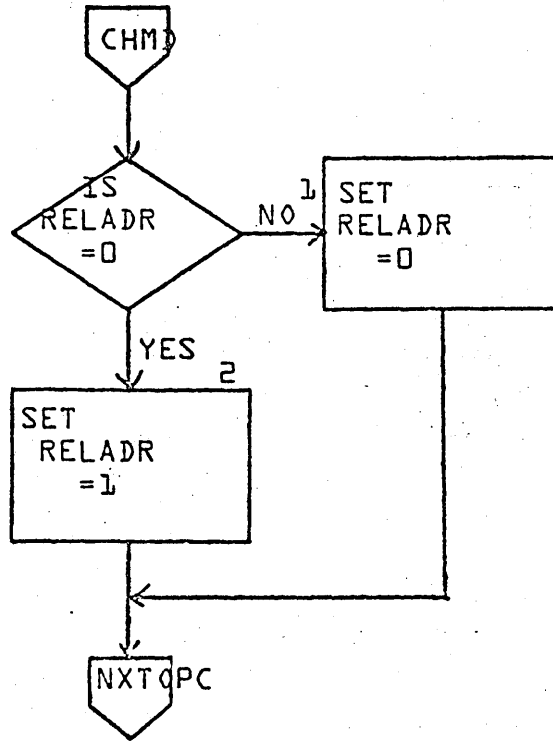


DOCUMENT CLASS IMS PAGE NO. 12-24  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700



OPERATION CODE =5  
 CHANGE MODE OF OPERATION {CHMD}

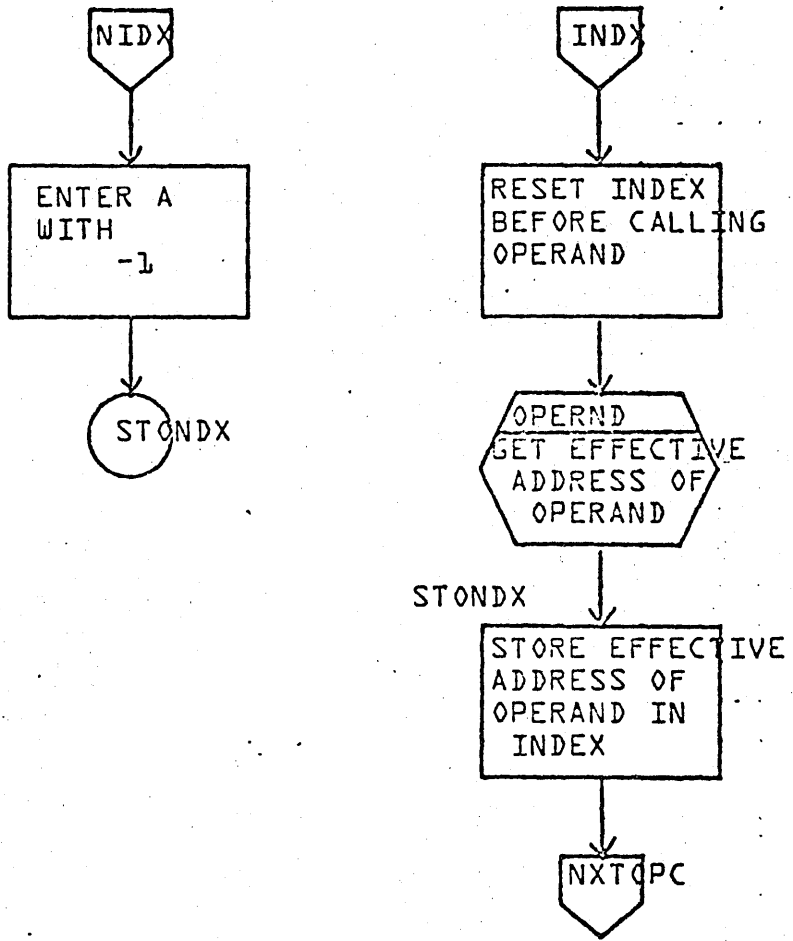
- 1. RELATIVE ADDRESS
- 2. ABSOLUTE ADDRESS



LA JOLLA RESOURCE CENTER  
 IMS Page 12-25  
 1700 MASS STORAGE FORTRAN  
 C005\*3.1 A/B

CONTROL DATA CORPORATION SOFTWARE DOCUMENT		DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.		REV		APPROVED		DATE	
SAMPLE CODE		DOCUMENT TITLE	DFLOT	PAGE 3 OF 34		PROJECT MGR.							
FLOWCHART		NUMBER		ISSUE DATE		PROJECT NAME							
DECISION TABLE		DRAWN BY		DATE		TASK NO.							
OTHER						TASK NAME							

OPERATION CODE = 6  
 NO INDEX {NIDX}  
 OPERATION CODE = F  
 INDEX {INDX}



LA JOLLA RESOURCE CENTER  
 IMS Page 12-26  
 1700 MASS STORAGE FORTRAN  
 C005\*3.1 A/B

CONTROL DATA CORPORATION SOFTWARE DOCUMENT		DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.		REV		APPROVED		DATE	
SAMPLE CODE		DOCUMENT TITLE	DFLOT			PROJECT MGR.							
FLOWCHART				PAGE 4	OF 9	PROJECT NAME							
DECISION TABLE				ISSUE DATE		TASK NO.							
OTHER				DATE		TASK NAME							

PRINTED

LA 27 11



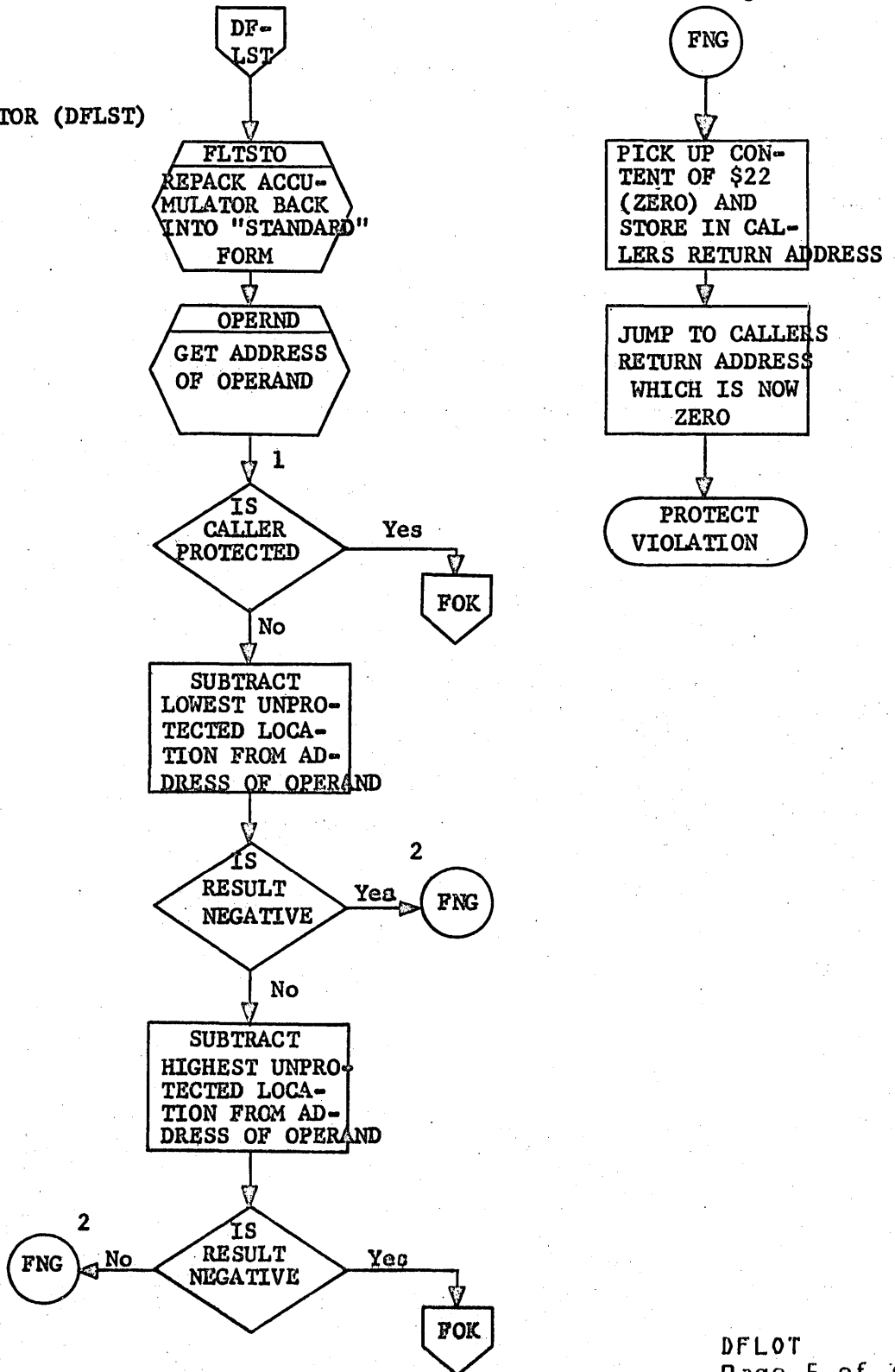
DOCUMENT CLASS IMS PAGE NO. 12-27  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

3

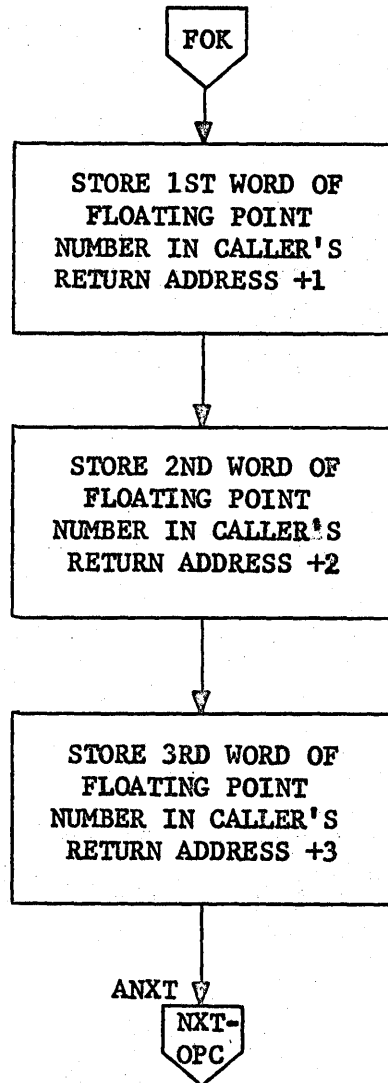
OPERATION CODE = D

DOUBLE STORE ACCUMULATOR (DFLST)

1. TEST FOR PROTECT VIOLATION.
2. STORAGE ADDRESS IS PROTECTED.
3. SIMULATE PROTECT VIOLATION.



DOCUMENT CLASS IMS PAGE NO. 12-28  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700



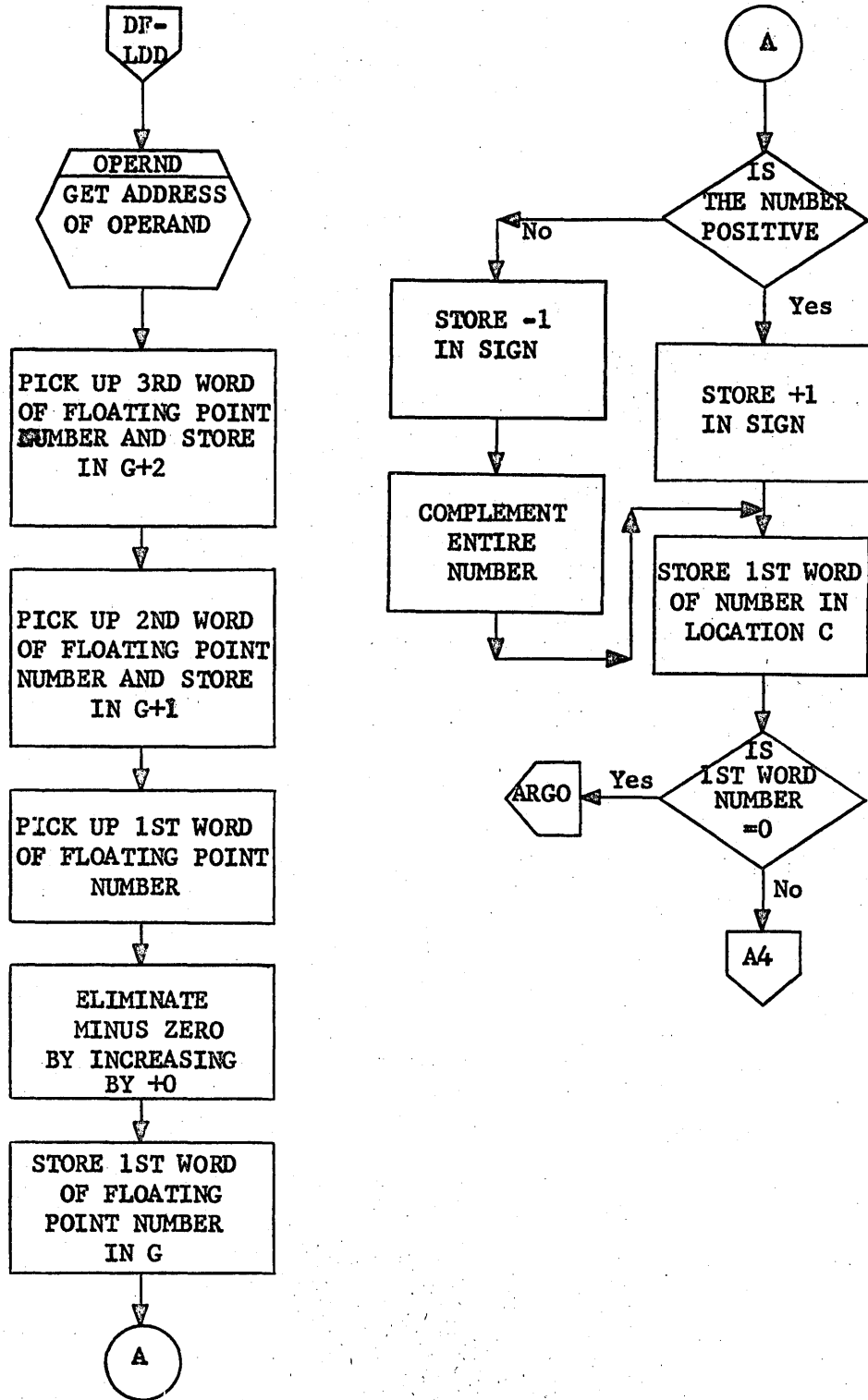


DOCUMENT CLASS IMS PAGE NO. 12-30  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

OPERATION CODE = B

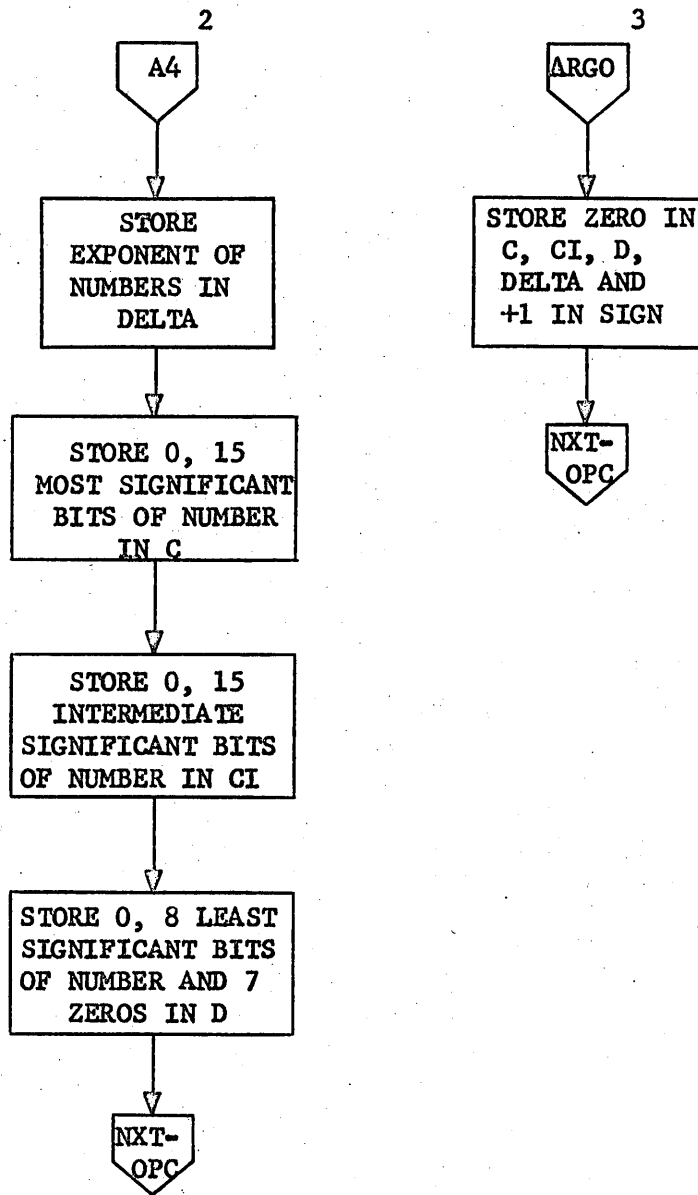
DOUBLE FLOATING LOAD  
(DFLOD)

1. FOLLOWING FIVE BOXES LOAD THE OPERAND INTO G, G+1, AND G+2.



DOCUMENT CLASS IMS PAGE NO. 12-31  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

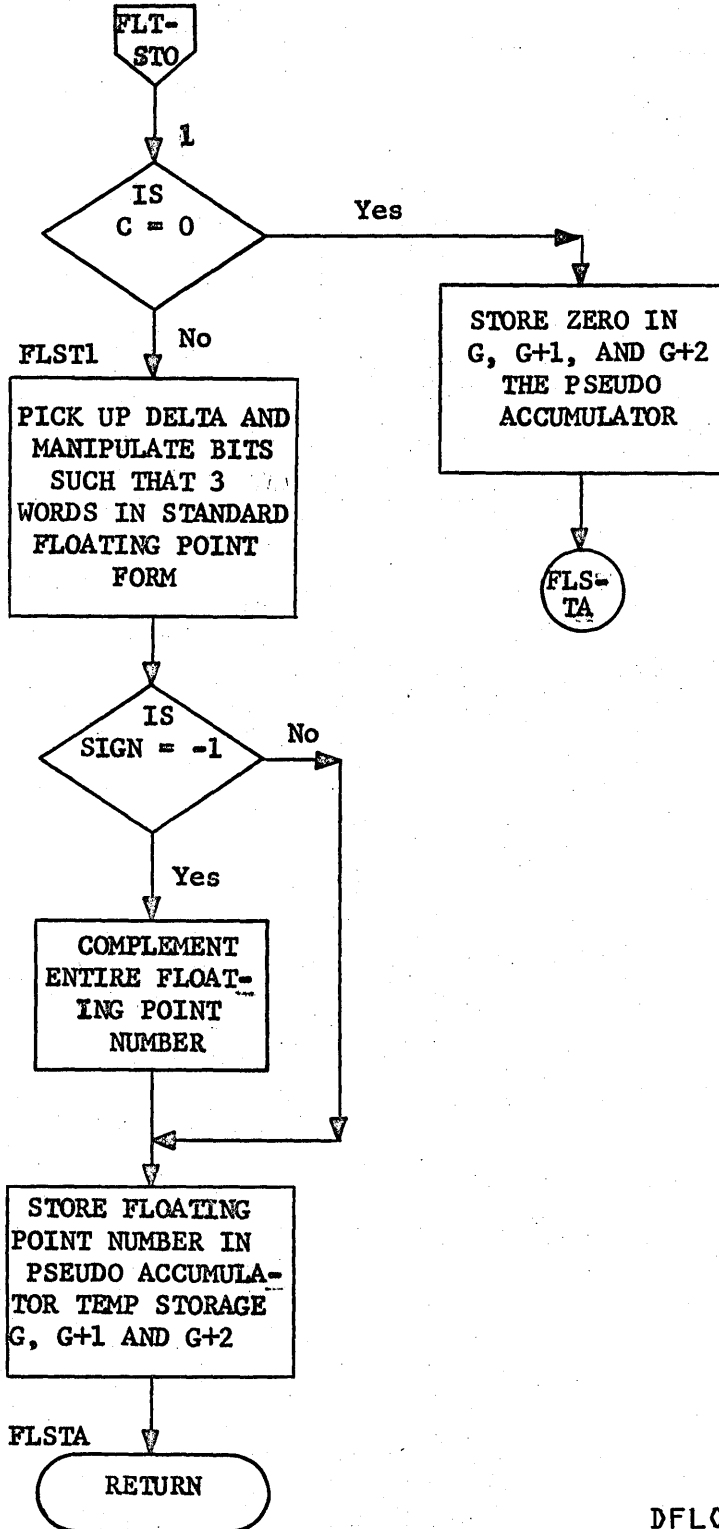
2. Unpack OPERAND into C, CI, D and DELTA.
3. Floating Point number is assumed to be zero since first word is zero.



DOCUMENT CLASS IMS PAGE NO. 12-32  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

REPACKING ACCUMULATOR

1. If C is zero, then the PSEUDO Accumulator is zero.

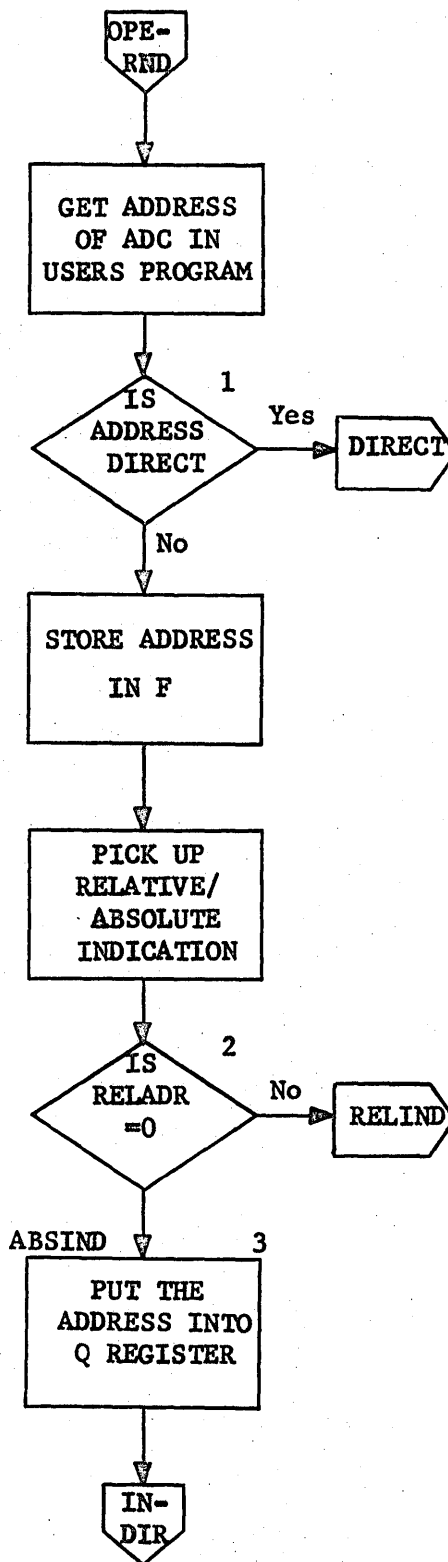




DOCUMENT CLASS IMS PAGE NO. 12-34  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

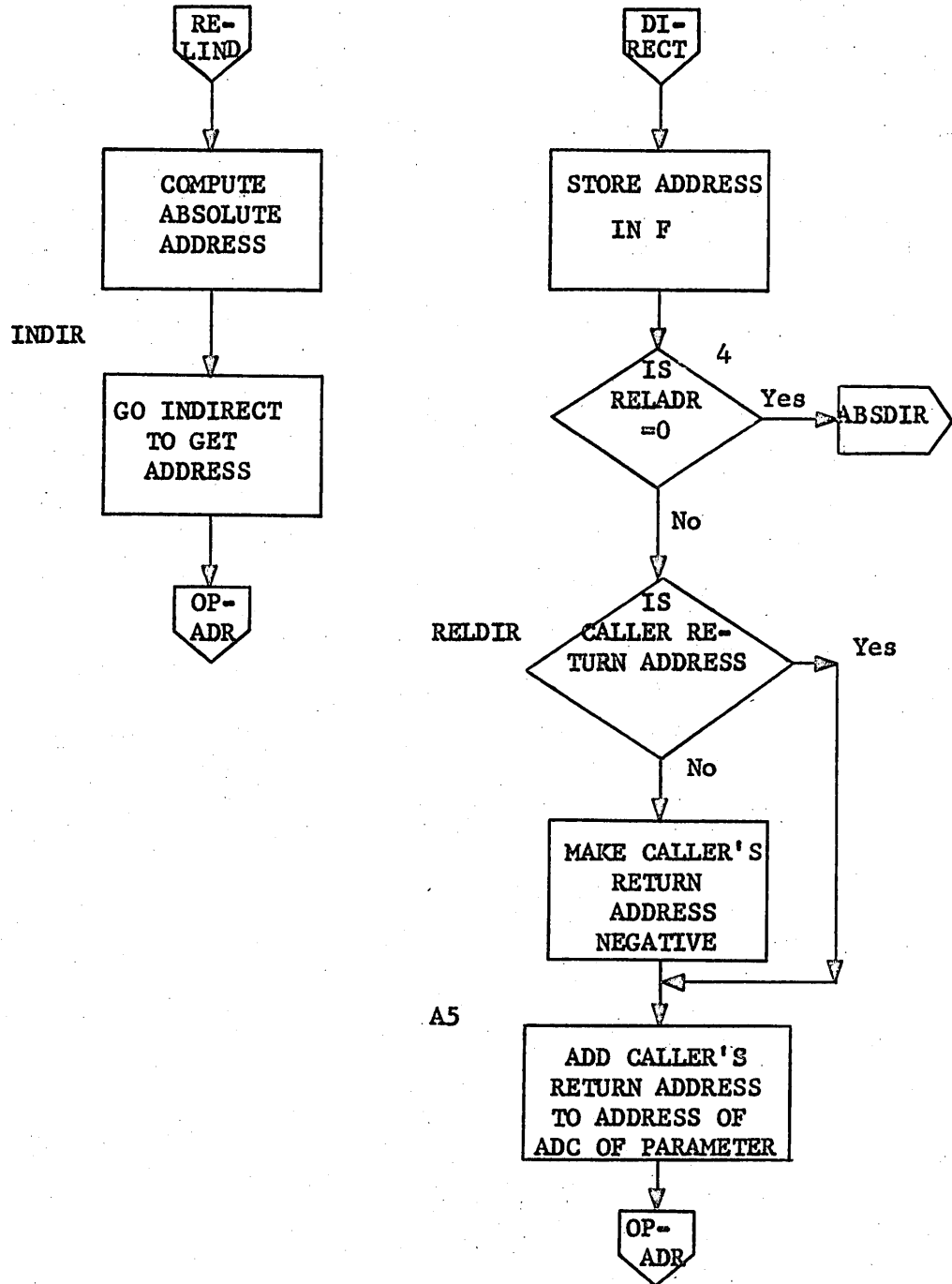
GET EFFECTIVE ADDRESS OF NEXT OPERAND

1. Address is direct if indirect bit (Bit 15) is not set.
2. RELADR=1 if relative address and=0 if absolute address.
3. Absolute and Indirect address.

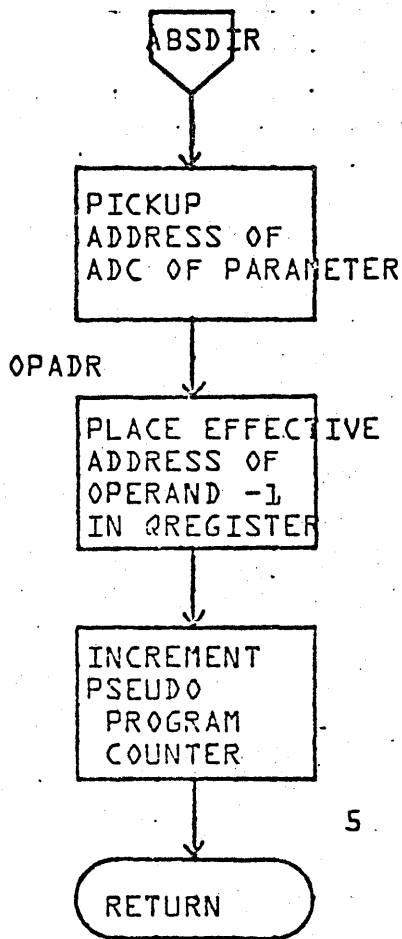




DOCUMENT CLASS IMS PAGE NO. 12-35  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3-1 A/B MACHINE SERIES 1700



5. RETURN WITH EFFECTIVE ADDRESS OF OPERAND-1 IN QREGISTER.



5

LA JOLLA RESOURCE CENTER  
 IMS Page 12-36  
 1700 MASS STORAGE FORTRAN  
 C005\*3.1 A/B

CONTROL DATA CORPORATION SOFTWARE DOCUMENT	DOCUMENT CLASS IMS	MACH. TYPE 1700	PROJECT NO.	APPROVED	DATE
SAMPLE CODE FLOWCHART DECISION TABLE OTHER	DOCUMENT TITLE DFLOT	PAGE 1 of 3	PROJECT MGR. PROJECT NAME TASK NO. TASK NAME		
	NUMBER	ISSUE DATE			
	DRAWN BY	DATE			

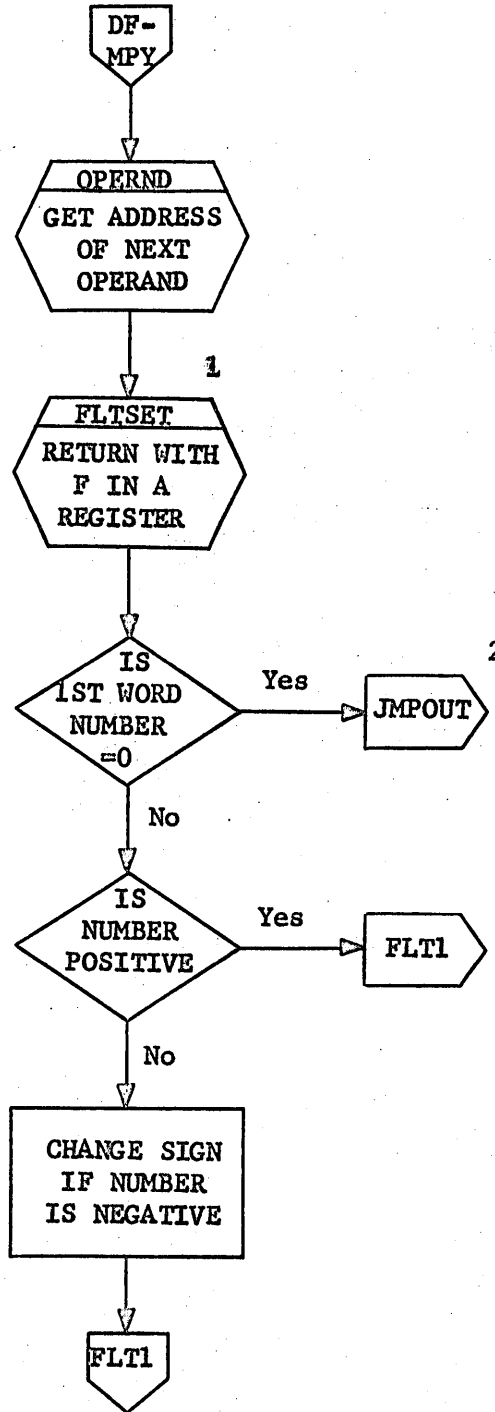
PRINTED IN

DOCUMENT CLASS IMS PAGE NO. 12-37  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

OPERATION CODE = 9

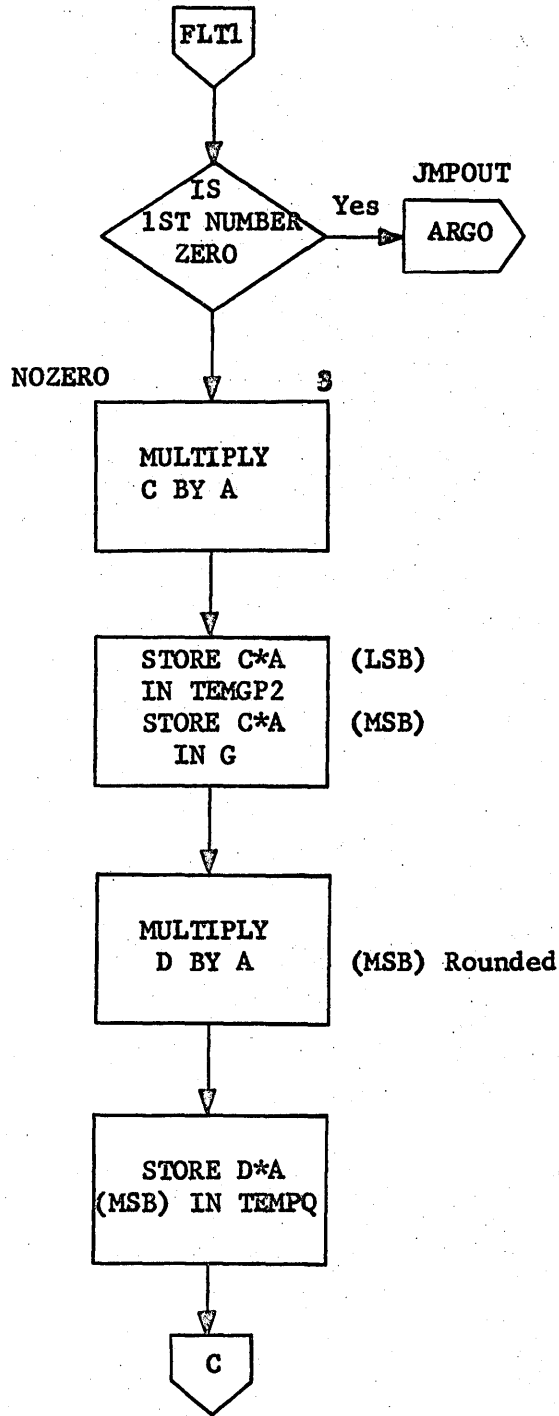
DOUBLE FLOATING MULTIPLY (DFMPY)

1. Gets the floating point number and stores in F, F+1 and F+2 then unpacks the number into  
 A=0, 15MSB  
 AI=0, 15ISB  
 B=0, 8LSB, 7 ZEROS  
 BETA=EXPONENT
2. A jump to JMPOUT causes a jump to ARGO.



DOCUMENT CLASS IMS PAGE NO. 12-38  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

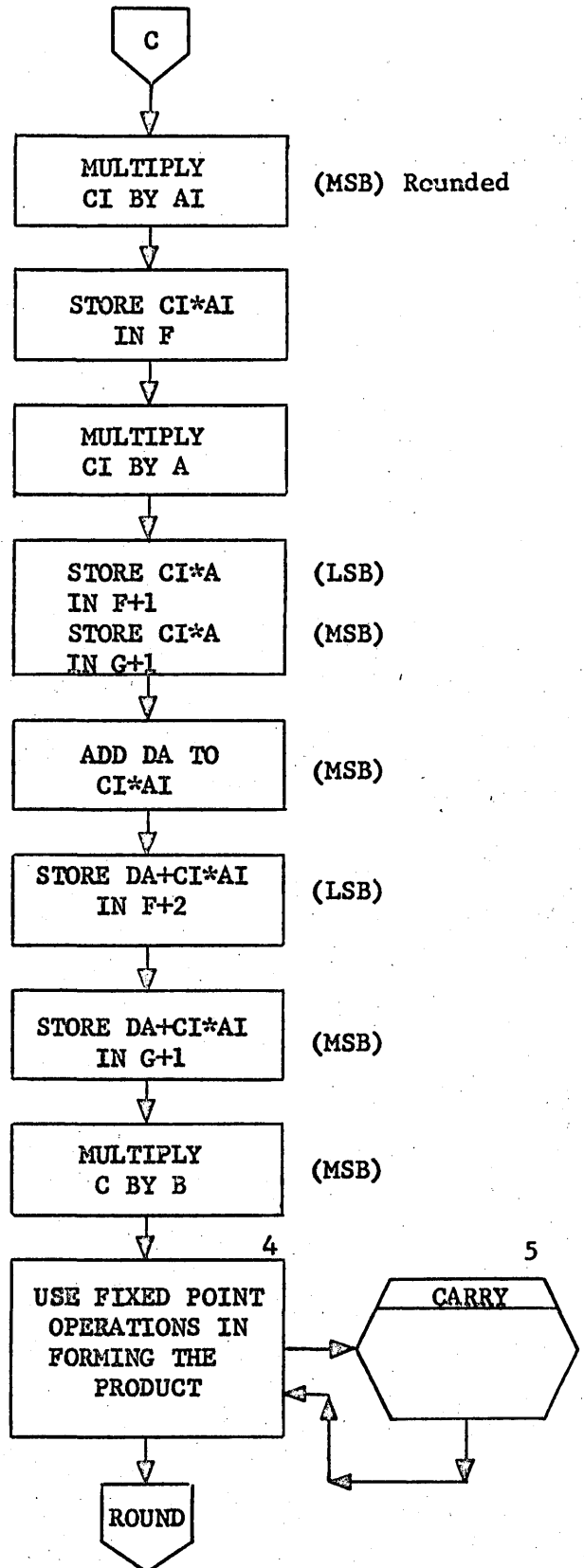
3. Multiply the MSB of one number by the MSB of the other number.



DOCUMENT CLASS IMS PAGE NO. 12-39  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. CO05\*3.1 A/B MACHINE SERIES 1700

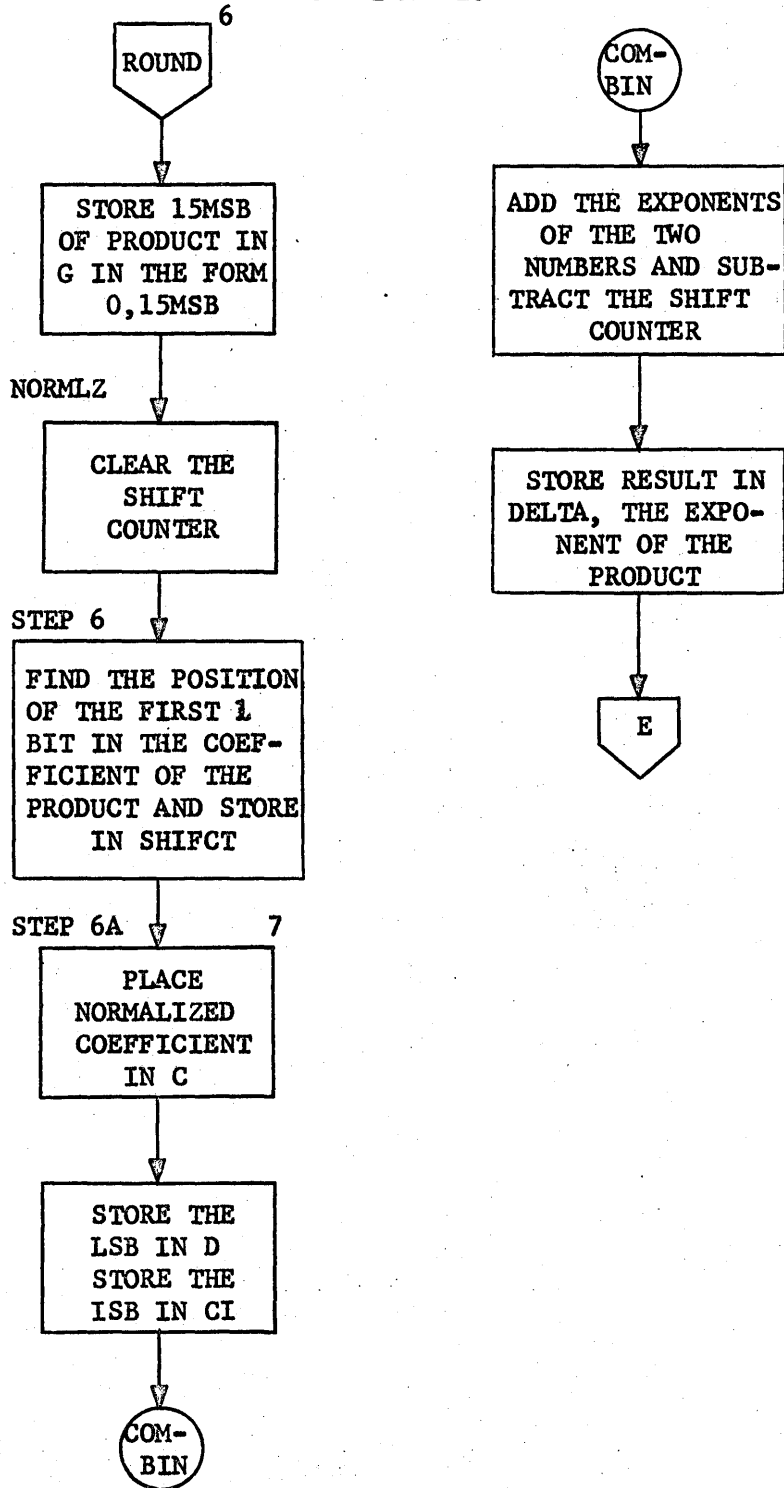
4.  $Product = C*A + (C*AI + CI*A)X2^{-15}$   
 $+ (D*A + CI*AI + C*B)X2^{-30}$

5. This routine checks for a carry in the A and Q registers. If A is negative, carry into Q. If Q is negative, end around.



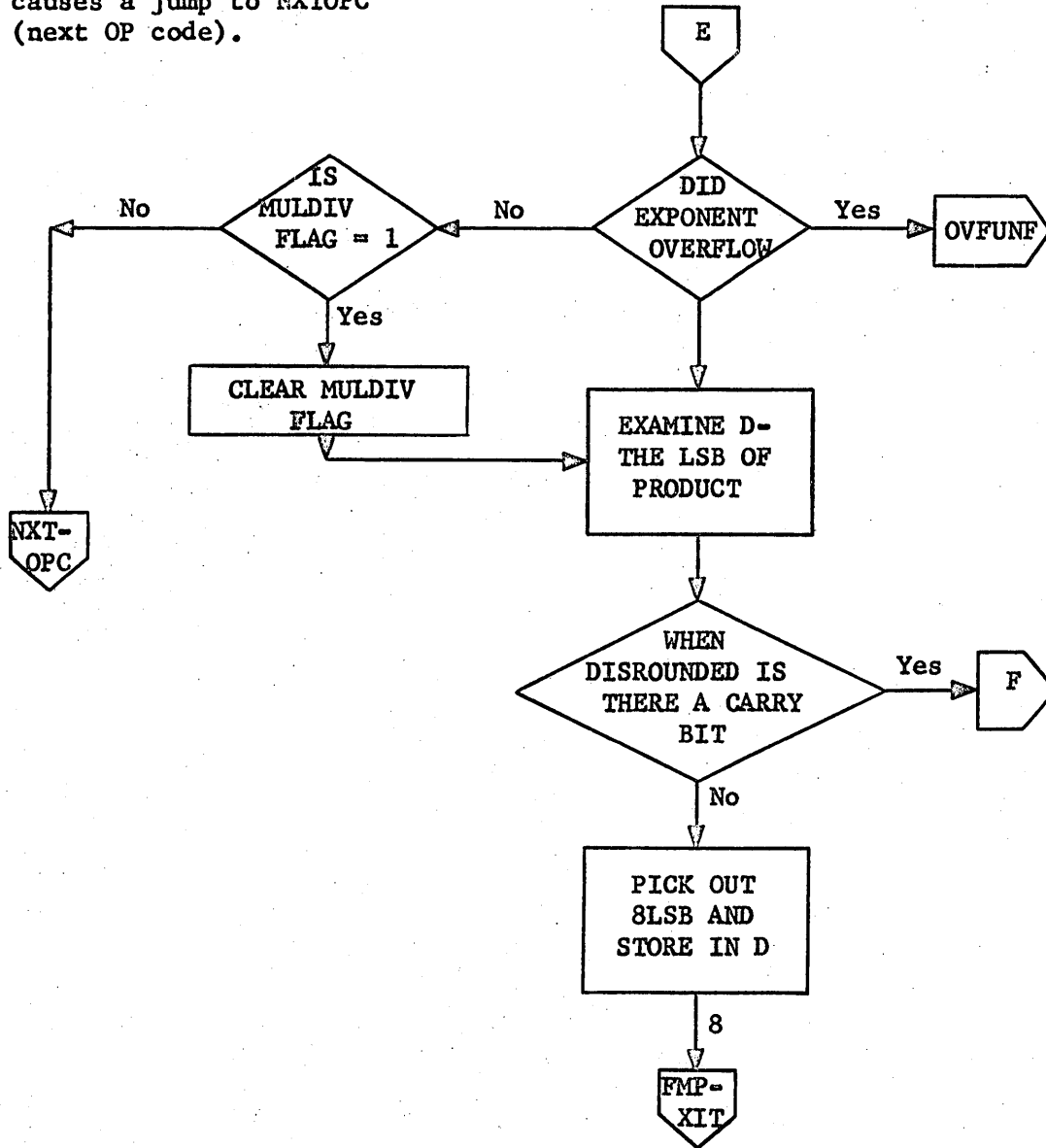
DOCUMENT CLASS IMS PAGE NO. 12-40  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. COO5\*3.1 A/B MACHINE SERIES 1700

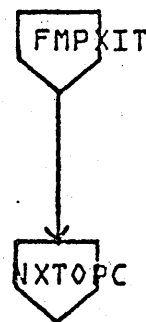
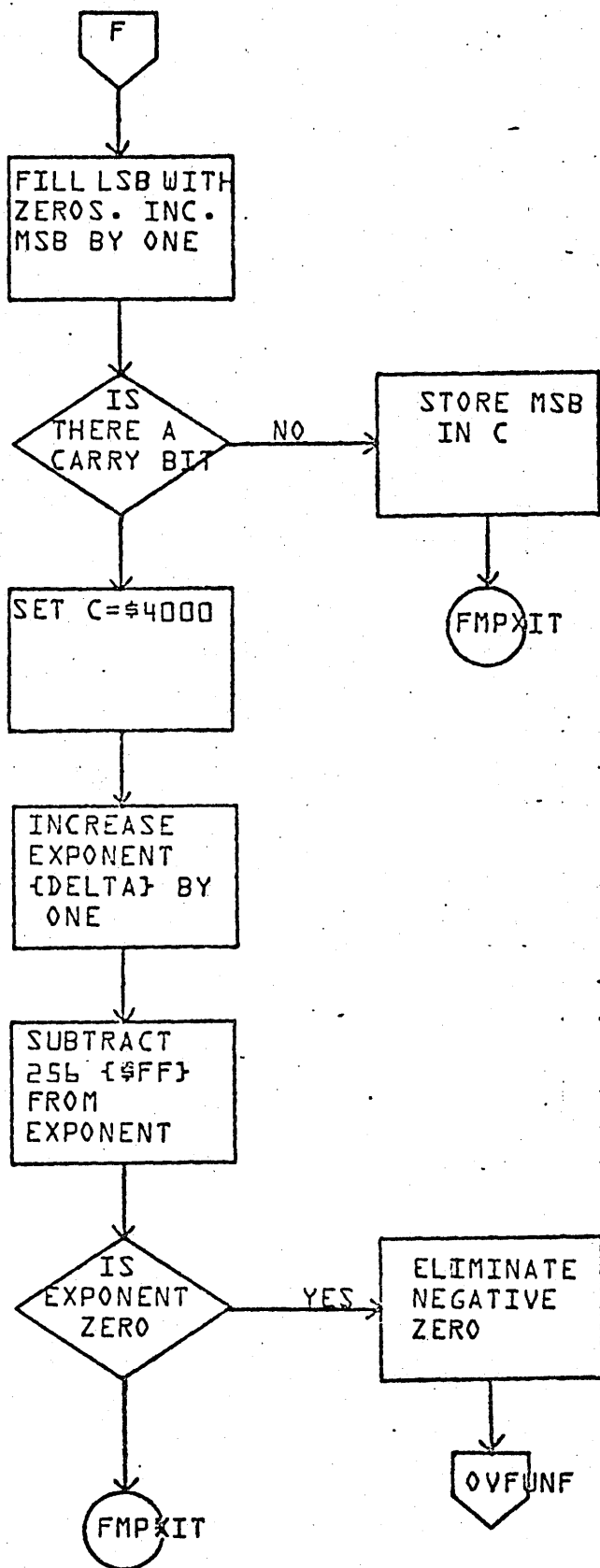
- 6. This section rounds and truncates to 39 bits.
- 7. Coefficient is of the form 0.15MSB, but the 15MSB are normalized such that a one appears in the first location of the MSB with SHIFCT keeping track of how many places have been shifted.



DOCUMENT CLASS IMS PAGE NO. 12-41  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

8. A jump to FMPXIT then causes a jump to NXTOPC (next OP code).





LA JOLLA RESOURCE CENTER  
 IMS Page 12-42  
 1700 MASS STORAGE FORTRAN  
 C005\*3.1 A/B

DATE		APPROVED		REV		PROJECT NO.		MACH. TYPE		DOCUMENT CLASS		ISSUE DATE		DRAWN BY	
							1700	IMS	DFLOT	0009					
PROJECT MGR.		PROJECT NAME		TASK NO.		TASK NAME		PAGE		NUMBER		ISSUE DATE		DATE	

CONTROL DATA CORPORATION  
 SOFTWARE DOCUMENT  
 SAMPLE CODE   
 FLOWCHART   
 DECISION TABLE   
 OTHER

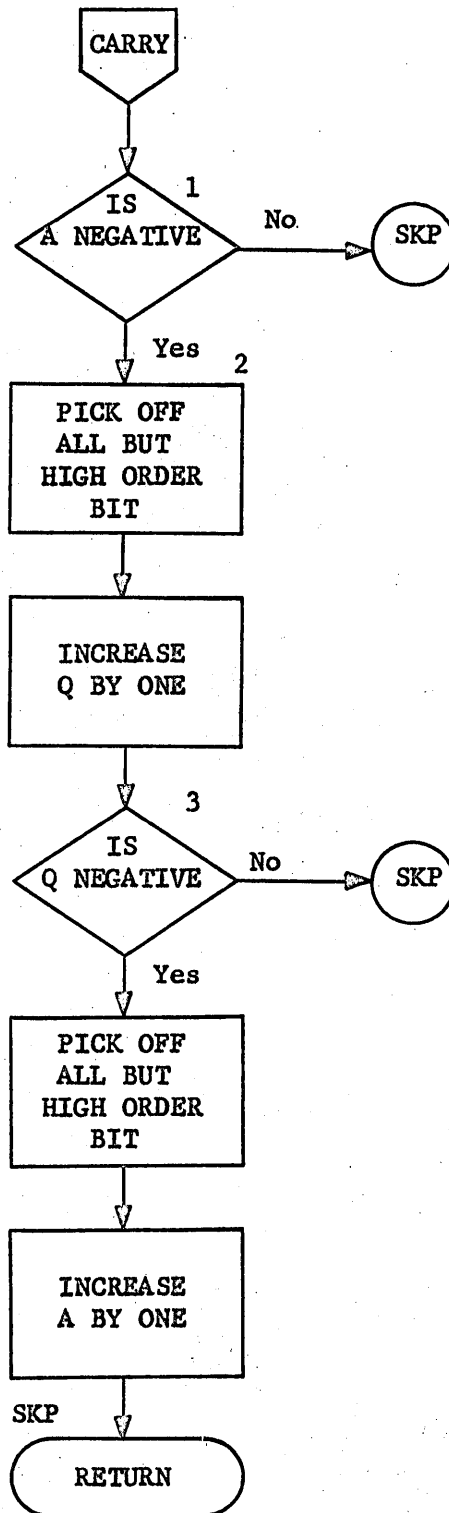
PRINTED IN

AA1386 IFOR CA127-11



DOCUMENT CLASS IMS PAGE NO. 12-43  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

1. A contains the sum of the first word of one argument times the second word of the other argument and vice versa.
2. If A is negative, carry into Q.
3. If Q is negative, end around.

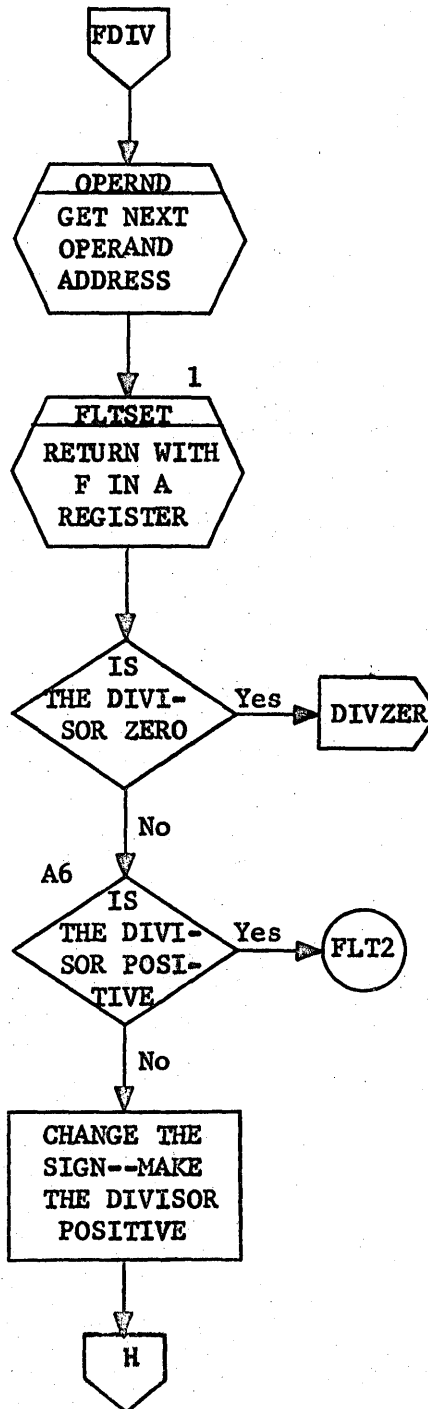


DOCUMENT CLASS IMS PAGE NO. 12-44  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

OPERATION CODE = A

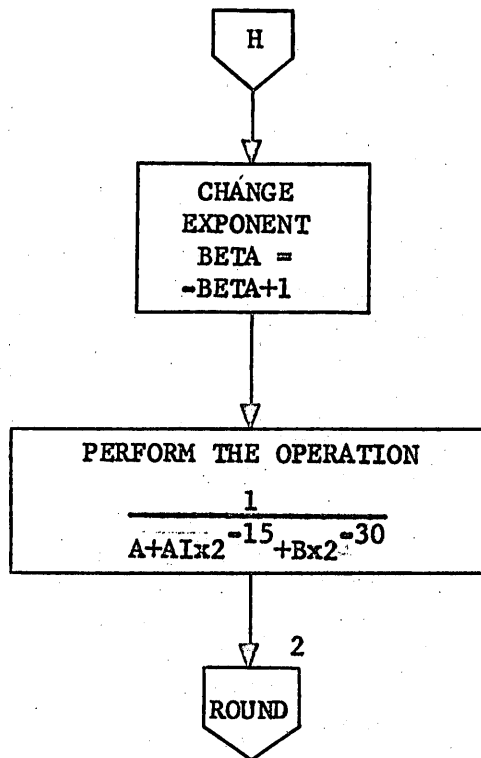
FLOATING DIVIDE (FDIV)

1. Gets the floating point number and stores in F, F+1 and F+2, then unpacks the number into A=0,15MSB  
 AI=0,15ISB  
 B=0,8LSB, 7 zeros  
 BETA=EXPONENT



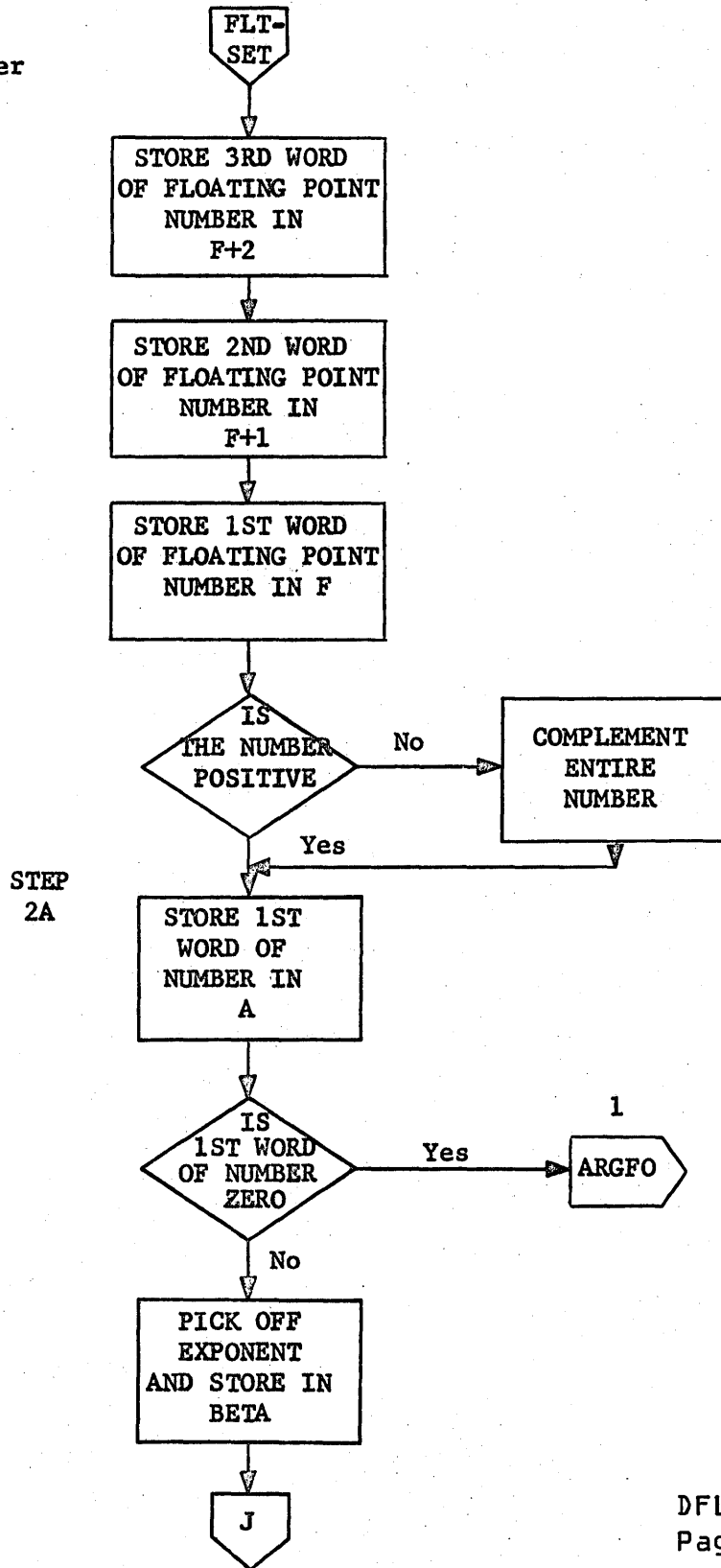
DOCUMENT CLASS IMS PAGE NO. 12-45  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

2. Round, Truncate and Normalize the results.

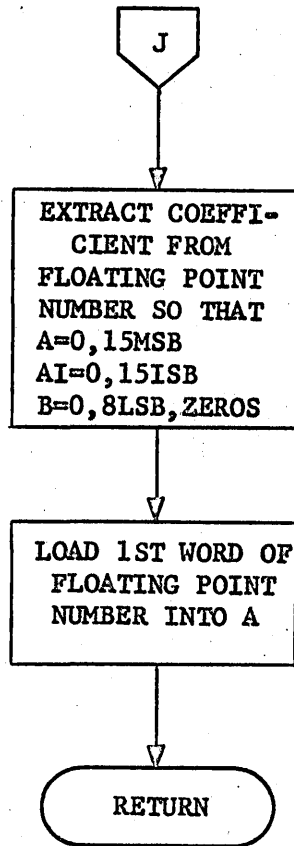


DOCUMENT CLASS IMS PAGE NO. 12-46  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

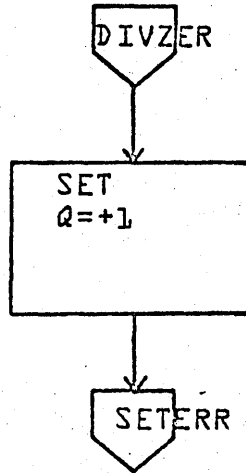
1. At ARGFO, the entire number is set equal to +0  
 $A \leftarrow +0$   
 $B \leftarrow +0$   
 $BETA \leftarrow +1$



DOCUMENT CLASS IMS PAGE NO. 12-47  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700



DIVIDE CHECK WHEN DIVIDING BY ZERO



LA JOLLA RESOURCE CENTER  
 IMS Page 12-48  
 1700 MASS STORAGE FORTRAN  
 C005\*3.1 A/B

CONTROL DATA CORPORATION SOFTWARE DOCUMENT		DOCUMENT CLASS	IMS	MACH. TYPE	1700	PROJECT NO.		REV		APPROVED		DATE	
SAMPLE CODE FLOWCHART DECISION TABLE		DOCUMENT TITLE	DFLOT	ISSUE DATE	PAGE 26 OF 37	PROJECT MGR.							
DRAWN BY		NUMBER		DATE		TASK NO.							
OTHER						TASK NAME							

5

4

2

1

A

B

C

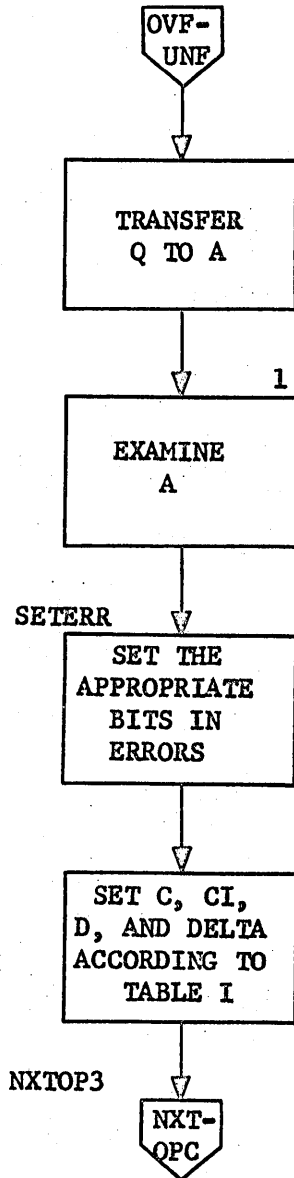
D

DOCUMENT CLASS IMS PAGE NO. 12-49  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

OVERFLOW OR UNDERFLOW

1. If A is positive, an overflow occurred.

If A is not positive, an underflow occurred.



CONTROL DATA CORPORATION  
 LA JOLLA RESOURCE CENTER \_\_\_\_\_ DIVISION

DOCUMENT CLASS IMS PAGE NO. 12-50  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

	C	CI	D	DELTA
EXPONENT OVERFLOW	\$7FFF	\$7FFF	\$7FC0	\$7F
DIVIDE FAULT	\$7FFF	\$7FFF	\$7FC0	\$7F
EXPONENT UNDERFLOW	\$4000	\$0000	\$0000	-\$7F

TABLE I

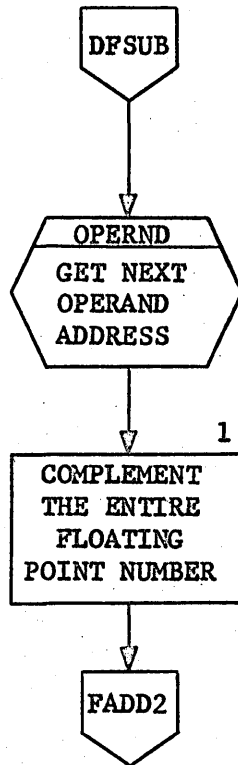


DOCUMENT CLASS IMS PAGE NO. 12-51  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

OPERATION CODE = 8

DOUBLE FLOATING SUBTRACT (DFSUB)

1. Change the sign before entering the Add routine.

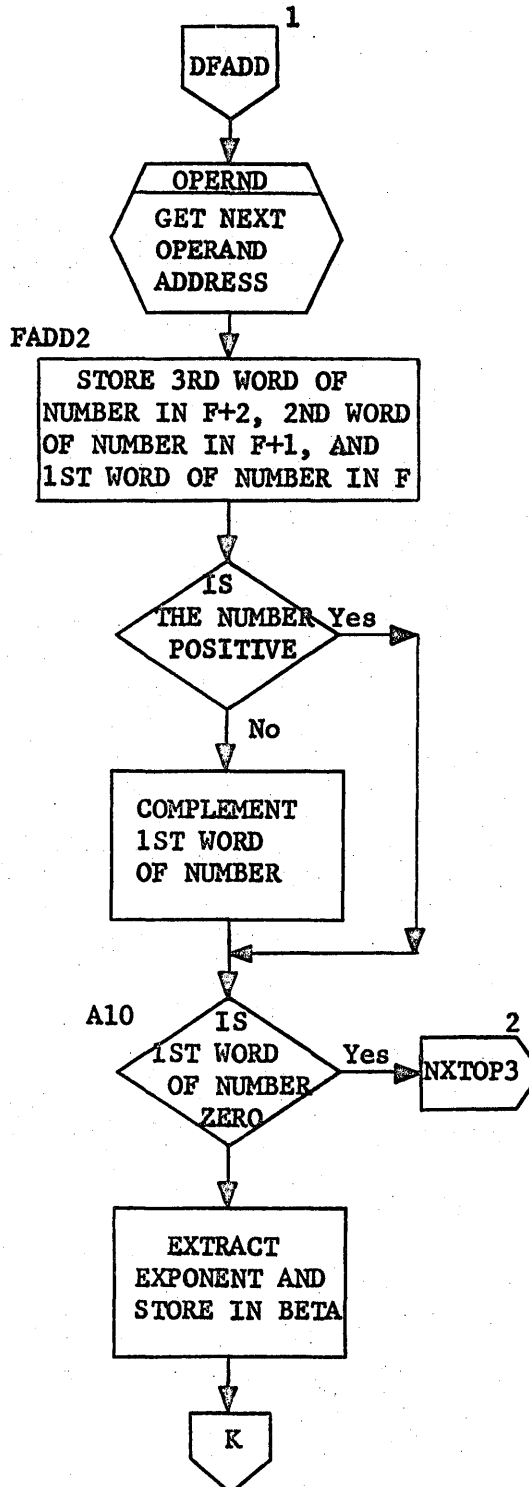


DOCUMENT CLASS IMS PAGE NO. 12-52  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

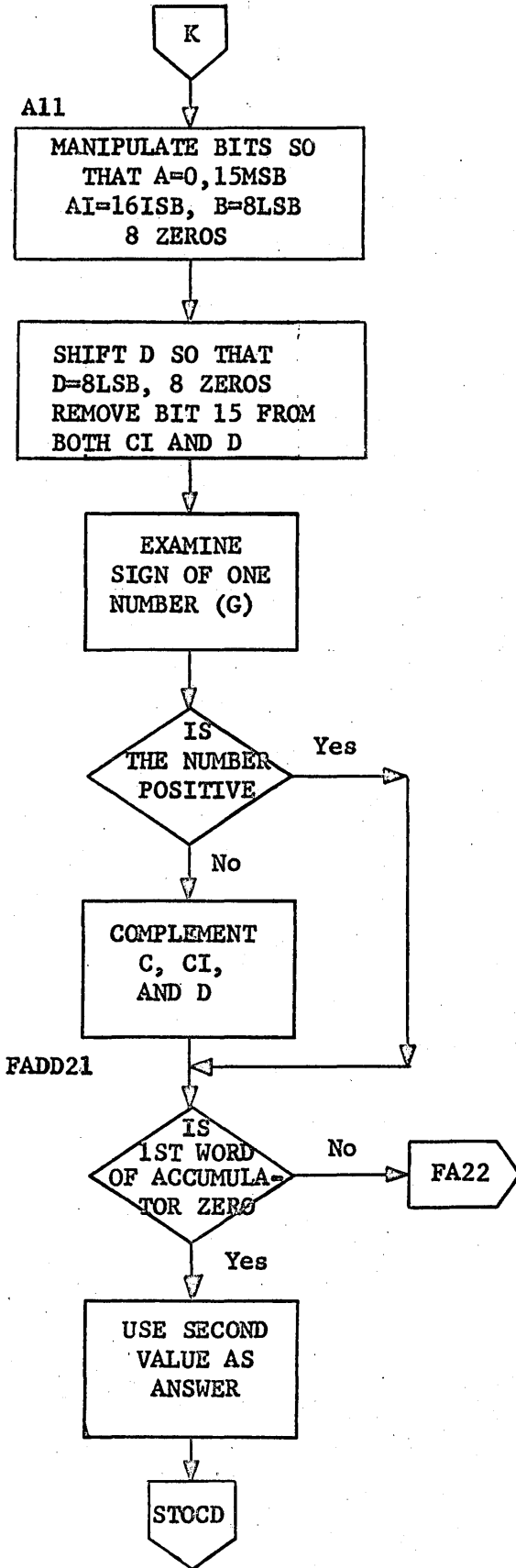
OPERATION CODE = E

DOUBLE FLOATING ADD (DFADD)

1. This section adds the floating point numbers in G and F and places the result in G.
2. Jump out - back to NXTOPC, get next OP code.

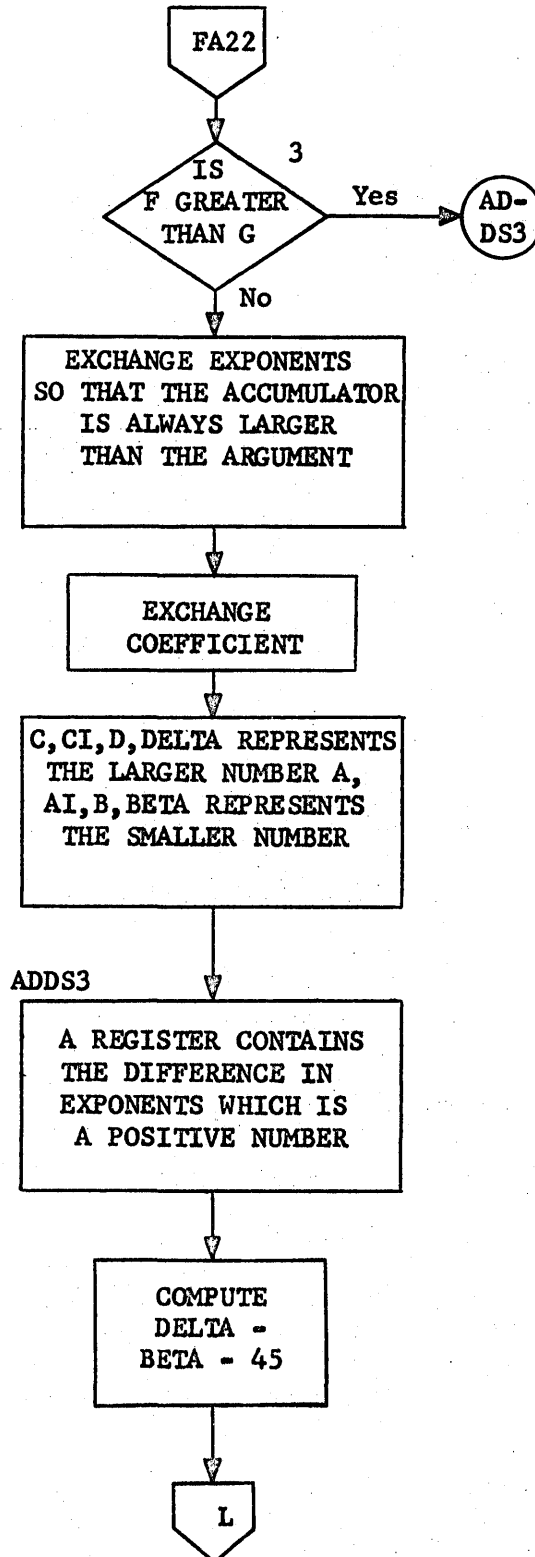


DOCUMENT CLASS IMS PAGE NO. 12-53  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700



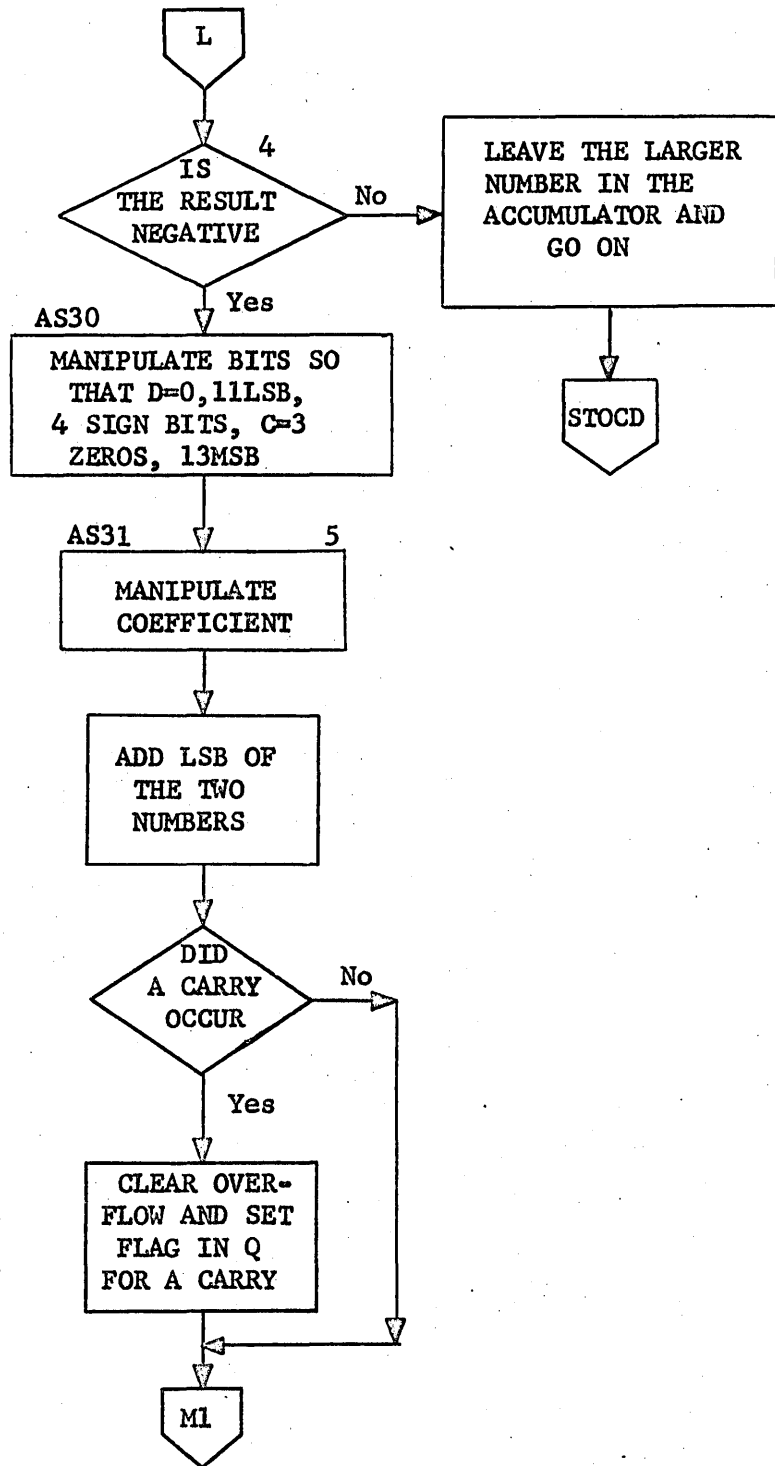
DOCUMENT CLASS IMS PAGE NO. 12-54  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. COO5\*3.1 A/B MACHINE SERIES 1700

3. This check is made by examining the difference of the exponents of the two numbers.

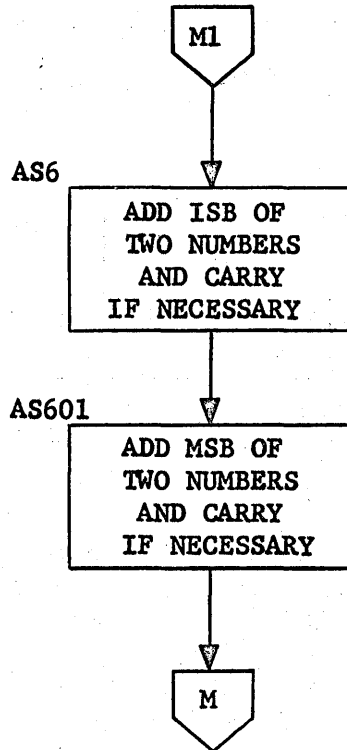


DOCUMENT CLASS IMS PAGE NO. 12-55  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

4. RESULT is result of DELTA-BETA-45; is there "TOO" great a difference between the numbers.
5. Shift smaller number right (DELTA-BETA) and set the sign of MSB to positive. Shift LSB right 1 and set the sign of LSB positive. Clear the carry to bit 15 and add 1 to MSB positive the smaller number.



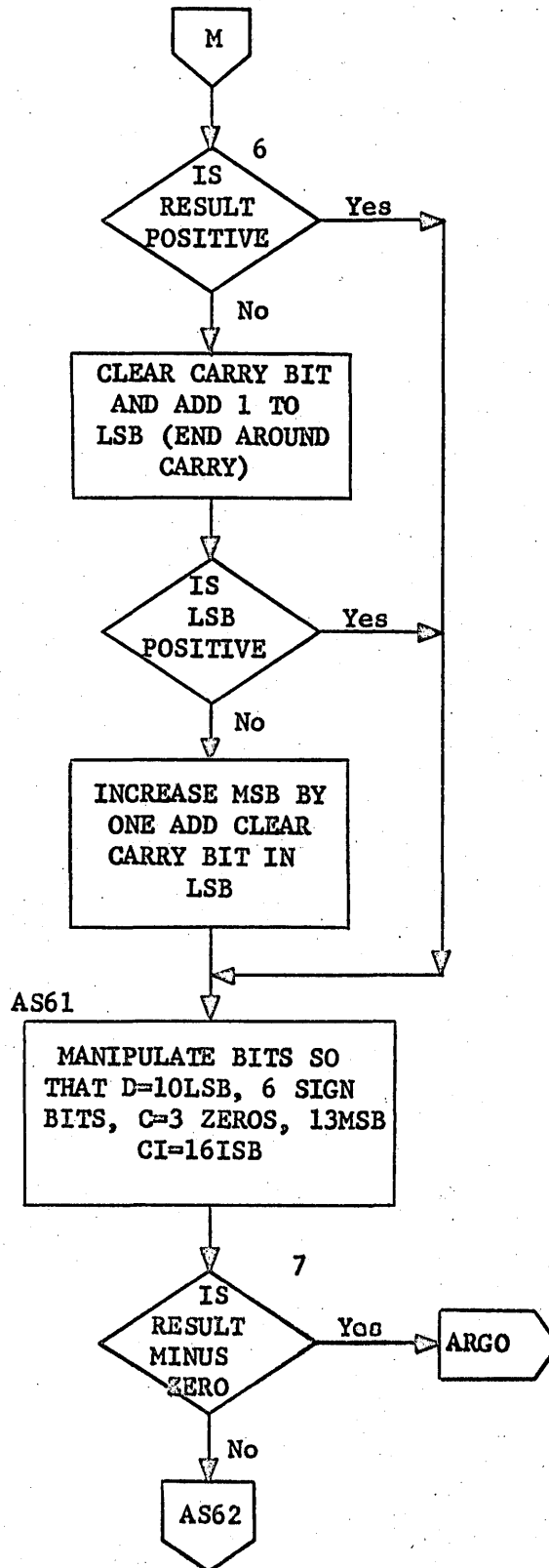
DOCUMENT CLASS IMS PAGE NO. 12-56  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700



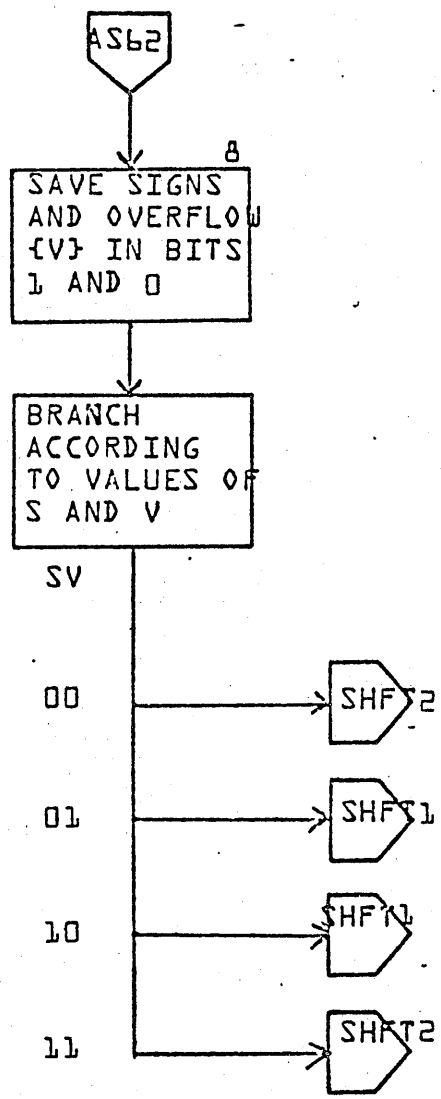
DOCUMENT CLASS IMS PAGE NO. 12-57  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

6. Result is result of addition of MSB and carry bit.

7. Minus zero is  
 Q=7FFF  
 A=FFFE



B. IF S.EQ.V, THEN V IS SIGN EXTENSION.  
IF S.NE.V, THEN RESULT OVERFLOWED INTO V.



LA JOLLA RESOURCE ENTER  
IMS Page 12-58  
1700 MASS STORAGE FORTRAN  
C005\*3.1 A/B

CONTROL DATA CORPORATION SOFTWARE DOCUMENT	DOCUMENT CLASS IMS	MACH. TYPE 1700	PROJECT NO.	APPROVED	DATE
SAMPLE CODE FLOWCHART DECISION TABLE OTHER	DOCUMENT TITLE DFLOT	PAGE 3 of 3	PROJECT MGR.	REV	
	NUMBER	ISSUE DATE	PROJECT NAME		
	DRAWN BY	DATE	TASK NO.		
			TASK NAME		

PRINTED

CA127-11

A

B

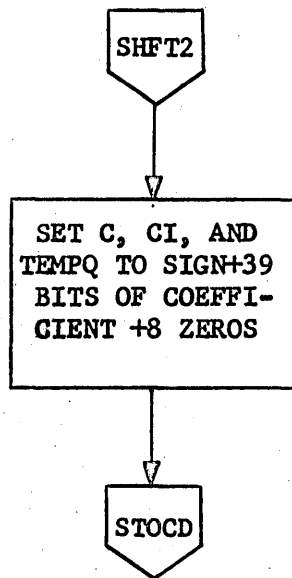
C

D

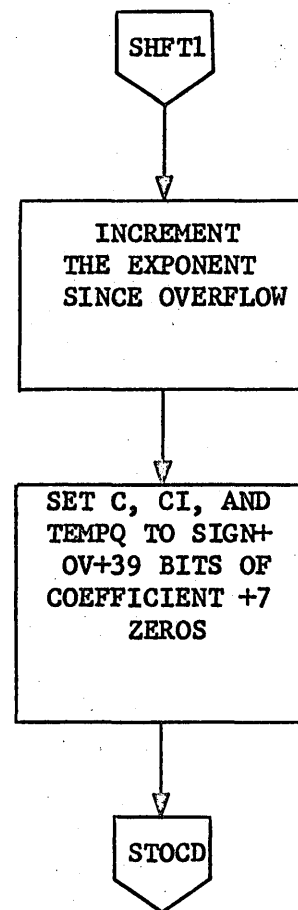


DOCUMENT CLASS IMS PAGE NO. 12-59  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

NO OVERFLOW

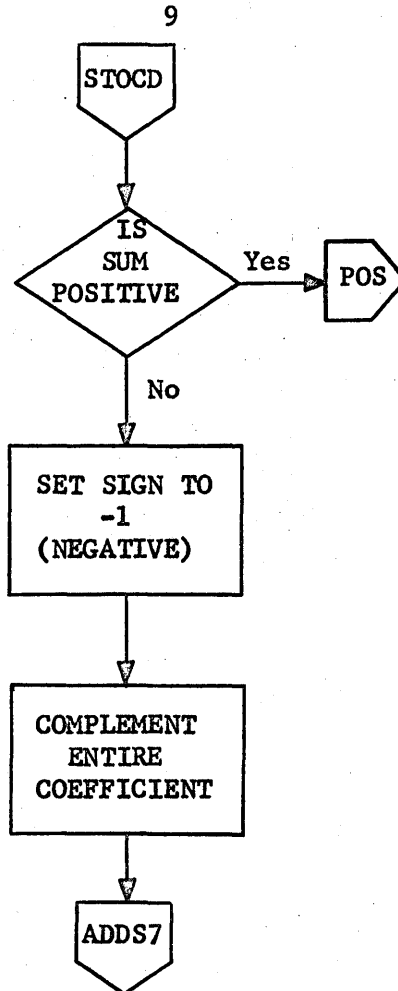


OVERFLOW OCCURRED



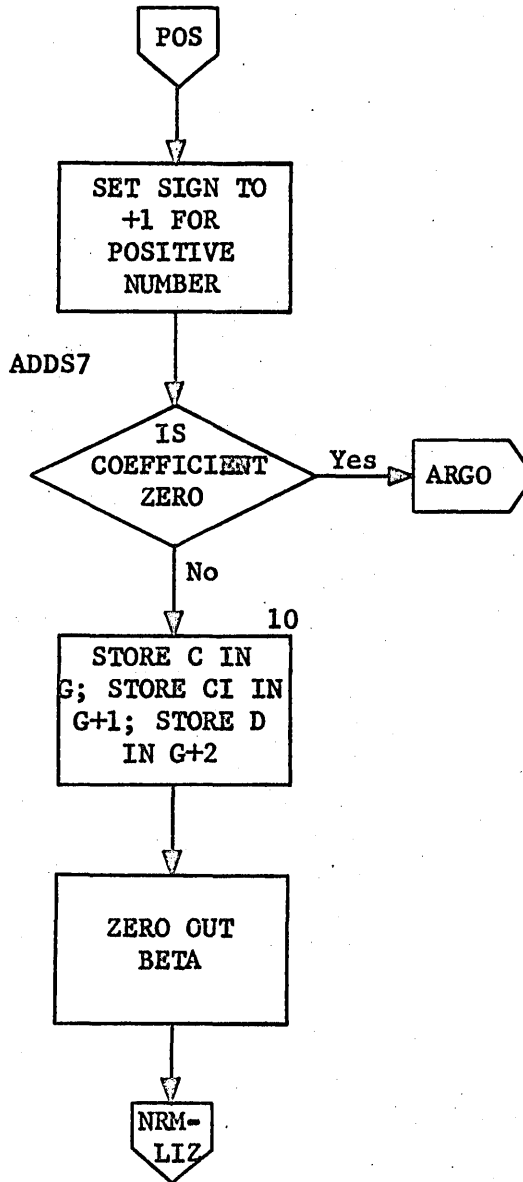
DOCUMENT CLASS IMS PAGE NO. 12-60  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. COO5\*3.1 A/B MACHINE SERIES 1700

9. Save the sign and magnitude of the results.  
On entry  
C=SIGN+15MSB  
CI=16ISB  
D=8LSB+8ZEROS



DOCUMENT CLASS IMS PAGE NO. 12-61  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

10. C=0+15MSB  
CI=16ISB  
D=8LSB+8ZEROS





DOCUMENT CLASS IMS PAGE NO. 13-1  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

CHAPTER 13

T A B L E O F C O N T E N T S

- 13.0 OBJECT TIME DOUBLE PRECISION INTRINSIC FUNCTIONS  
AND COMPILER SUPPORT ROUTINES
- 13.1 General Description
- 13.2 Q8DAB Routine
- 13.3 DSIGN Routine
- 13.4 Q8DFLT Routine
- 13.5 SNGL Routine
- 13.6 Q8DBLE Routine
- 13.7 DRSTOR Routine - Double Precision Compiler Support  
Object Time Routine

DOCUMENT CLASS TMS PAGE NO. 13-2  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

13.0 OBJECT TIME DOUBLE PRECISION INTRINSIC FUNCTIONS

13.1 General Description

The following Object Time Intrinsic Functions are included in this chapter:

Intrinsic Functions	I/O Routine
DABS{X}	QBDAB
DSIGN{X,Y}	DSIGN
DFIX{X}	FXFL {See Chapter 10 for Description}
DFLT{I}	QBDFLT
SNGL	SNGL
DBLE	QBDBLE

13.2 QBDAB Routine

This routine, which is written in 1700 Assembly Language, computes the absolute value of a double precision floating point number and leaves the result in the pseudo accumulator.

The calling sequence is:

RTJ DABS

1. address of argument

The entry points are:

QBDAB  
 DABS {which is equated to QBDAB}

The external declared is:

DFLOT

The low core locations used by this routine are:

DFLACC {#C5, #C6, #C7} Double Precision pseudo accumulator  
 MASKSB {#11} Mask #7FFF

DOCUMENT CLASS IMS PAGE NO. 13-3  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

13.3

DSIGN Routine

This routine, which is written in 1700 Assembly Language, computes the sign of the second argument times the absolute value of the first argument and leaves the result in the pseudo accumulator.

The calling sequence is:

RTJ DSIGN

1. address of first argument
2. address of second argument

The entry points are:

QBD SG  
DSIGN {which is equated to QBD SG}

The declared external is:

DFLOT

The low core locations used by this routine are:

A1 {D8}  
A2 {D9}  
FF {E1} used to save I register  
QS {E2} used to save Q register

13.4

QBDFLT Routine

Entry DFLT

This routine, which is written in 1700 Assembly Language, performs the conversion of an integer number into a double precision floating point number.

The calling sequence is:

RTJ DFLT

1. address of argument

The external declared is:

FLOAT

DOCUMENT CLASS IMS PAGE NO. 13-4  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3-1 A/B MACHINE SERIES 1700

The low core locations used by this routine are:

MASK { $\$11$ } MASK bit pattern  $\$7FFF$   
QS { $\$E2$ } used to save Q register

Entry Q8DFLT

This routine, which is written in Assembly Language, performs the conversion {across the equal sign} of an integer into a double precision Floating Point Number {DP=I}.

The calling sequence is:

LDA I {the integer value to be converted is in  
the A register upon entry}  
RTJ Q8DFLT

The external declared is:

Q8QFLT

The low core location used by this routine is:

QS { $\$E2$ } used to save Q register

13.5

SNGL Routine

This routine, which is written in 1700 Assembly Language, obtains the most significant 32 bits of a double precision argument.

The calling sequence is:

RTJ SNGL  
1. address of argument

The entry points are:

SNGL  
Q8SNGL {which is equated to SNGL}

The external declared is:

FL0T



DOCUMENT CLASS TMS PAGE NO. 13-5  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

The low core location used by this routine is:

MASK { $\$11$ } MASK word  $\$7FFF$

13.6 Q8DBLE Routine

This routine, which is written in 1700 Assembly Language, expresses a single precision number {the 2-word argument} in double precision form {3-word result}.

The calling sequence is:

RTJ DBLE

1. address of argument

The entry points are:

Q8DBLE  
DBLE {which is equated to Q8DBLE}

The low core locations used by this routine are:

DFLACC	{ $\$C5, \$C6, \$C7$ }	Double Precision Floating Point Pseudo Accumulator
MASK	{ $\$11$ }	Mask utilizing the following bit pattern $\$7FFF$
ZERO	{ $\$22$ }	Low Core cell containing Zero

13.7 DRSTOR Subroutine - Double Precision Compiler Support Object Time Routine

This subroutine is called by the FORTRAN compiler to perform double precision and single precision pseudo-accumulator stores which are necessary to provide efficient generated code. This subroutine is coded in 1700 Assembly Language. It contains the following entry points: DSTOR1, RSTOR1, and DSTOR2.

The low core locations used by this routine are:

CZERO	{ $\$22$ }	Low Core cell containing Zero
DFLACC	{ $\$C5, \$C6, \$C7$ }	Double precision floating point pseudo accumulator
MASK	{ $\$11$ }	MASK bit Pattern $\$7FFF$
QS	{ $\$E2$ }	Used to save Q register

DOCUMENT CLASS IMS PAGE NO. 13-6  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. 005\*3.1 A/B MACHINE SERIES 1700

Entry Point DSTOR1

This routine obtains values stored in  $\$C5$ ,  $\$C6$ , and  $\$C7$  and stores them in the parameter argument.

The calling sequence is:

RTJ DSTOR1

1. address of double precision argument

The input to the routine is in cells  $\$C5$ ,  $\$C6$  and  $\$C7$ . The output from the routine is the contents of cells  $\$C5$  and  $\$C6$  along with the value zero stored in the parameter argument.

Entry Point RSTOR1

This routine obtains values stored in  $\$C5$ ,  $\$C6$ , and  $\$C7$  and stores only  $\$C5$  and  $\$C6$  in the parameter argument which is single precision argument.

The calling sequence is:

RTJ RSTOR1

1. address of single precision argument

The input to the routine is stored in cells  $\$C5$ ,  $\$C6$  and  $\$C7$ .

The output from the routine is the contents of cells  $\$C5$  and  $\$C6$  into the parameter argument.

Entry Point DSTOR2

This routine obtains values stored in the pseudo floating accumulator {cells  $\$C5$  and  $\$C6$ } and stores them in the parameter argument. The routine also stores a zero into cell  $\$C7$  as well as into the third word of the parameter argument.

The calling sequence is:

RTJ DSTOR2

1. address of double precision argument

CONTROL DATA CORPORATION  
LA JOLLA RESOURCE CENTER DIVISION

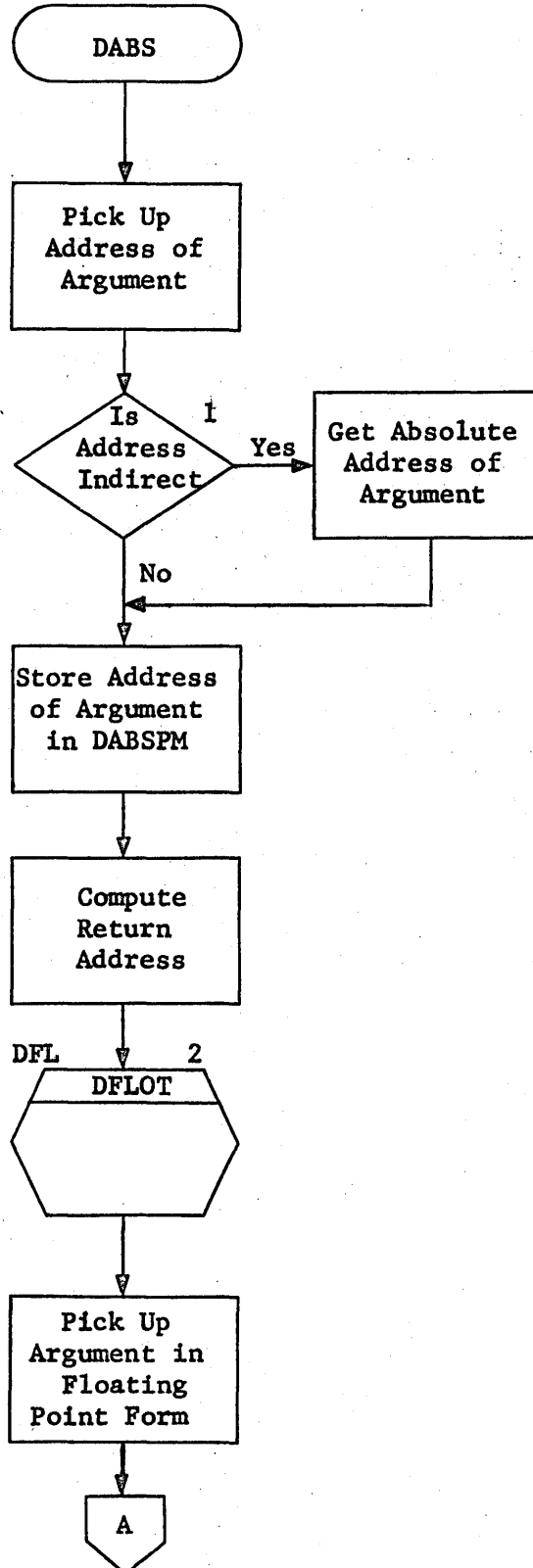
DOCUMENT CLASS IMS PAGE NO. 13-7  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. COO5\*3.1 A/B MACHINE SERIES 1700

The input to the routine is stored in cells  $\$C5$ ,  $\$C6$  and  $\$C7$ .

The output from the routine is the contents of cells  $\$C5$  and  $\$C6$  and the value zero all stored in the parameter argument. Cell  $\$C7$  is also set to zero.

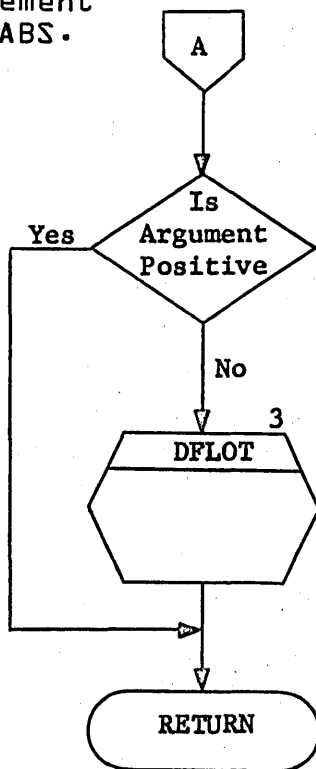
DOCUMENT CLASS TMS PAGE NO. 13-8  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. COO5\*3.1 A/B MACHINE SERIES 1700

1. Is address negative?
2. Do a double precision floating point load of argument into pseudo accumulator.

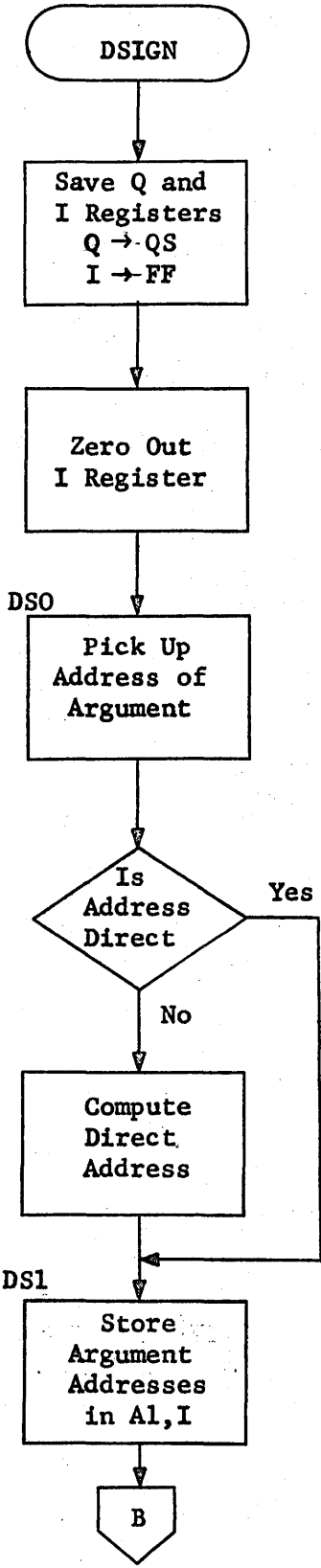


DOCUMENT CLASS IMS PAGE NO. 13-9  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. COO5\*3.1 A/B MACHINE SERIES 1700

3. Do a double precision floating point complement of the argument of DABS.

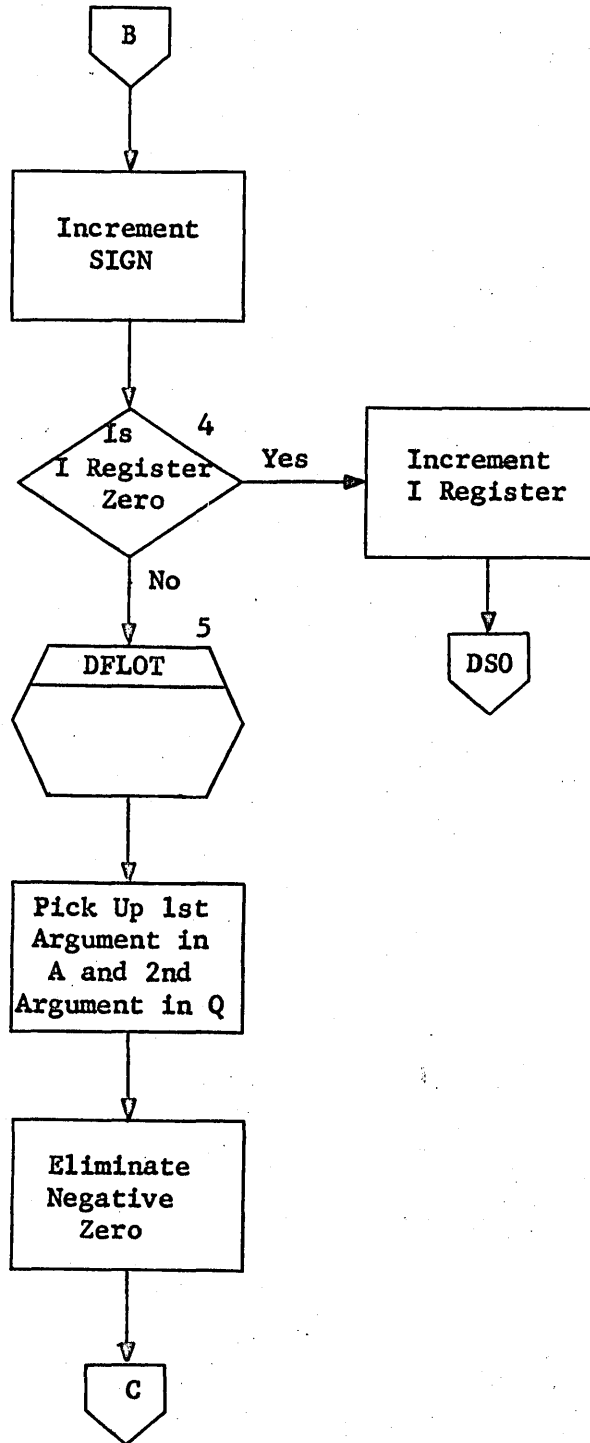


DOCUMENT CLASS IMS PAGE NO. 13-10  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

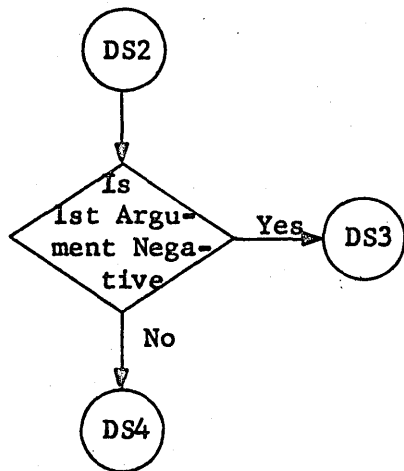
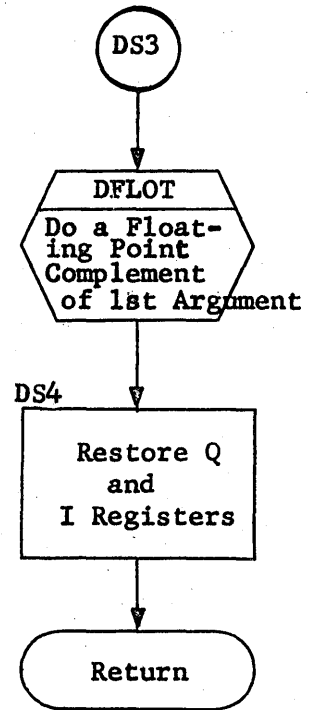
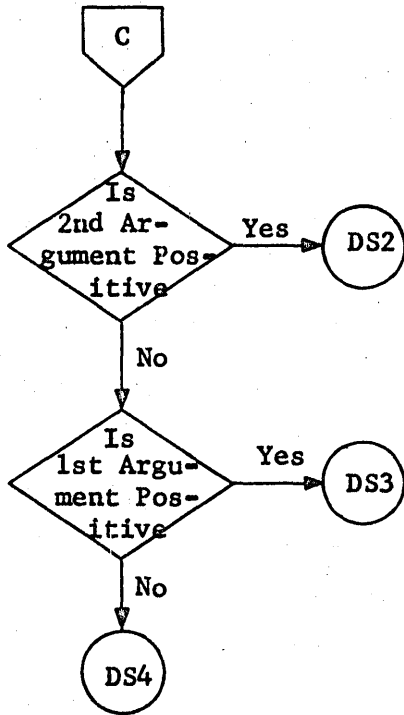


DOCUMENT CLASS IMS PAGE NO. 13-11  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. COO5\*3.1 A/B MACHINE SERIES 1700

- 4. Have we just processed the first argument's address.
- 5. Do a Double Precision floating point load of 1st argument.



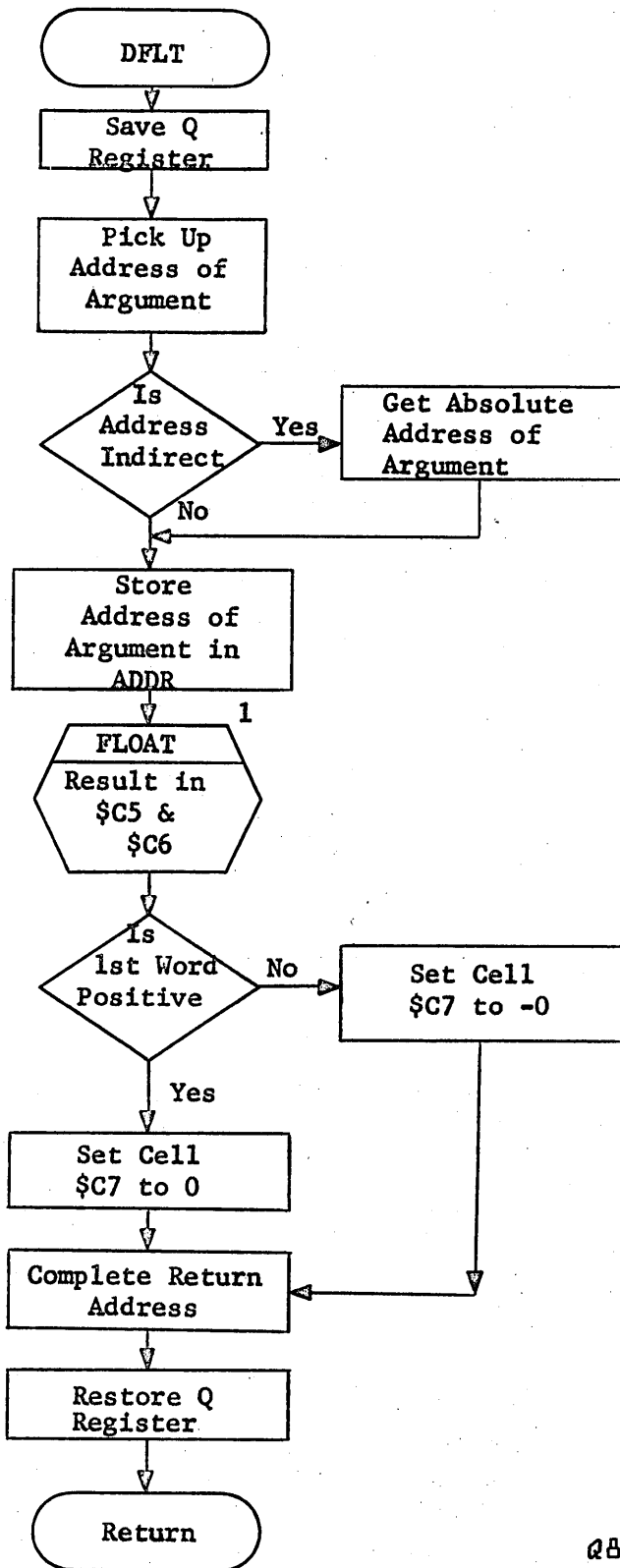
DOCUMENT CLASS IMS PAGE NO. 13-12  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. CO05\*3.1 A/B MACHINE SERIES 1700





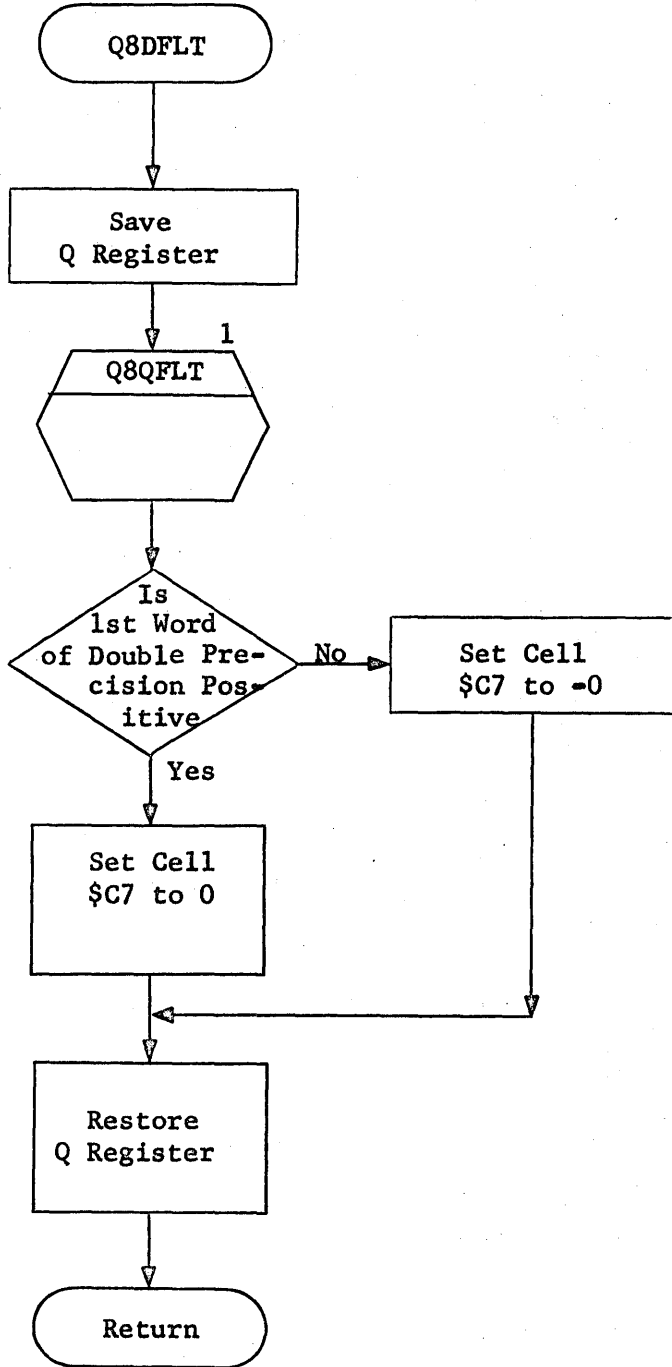
DOCUMENT CLASS IMS PAGE NO. 13-13  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

1. FLOAT the integer parameter argument.



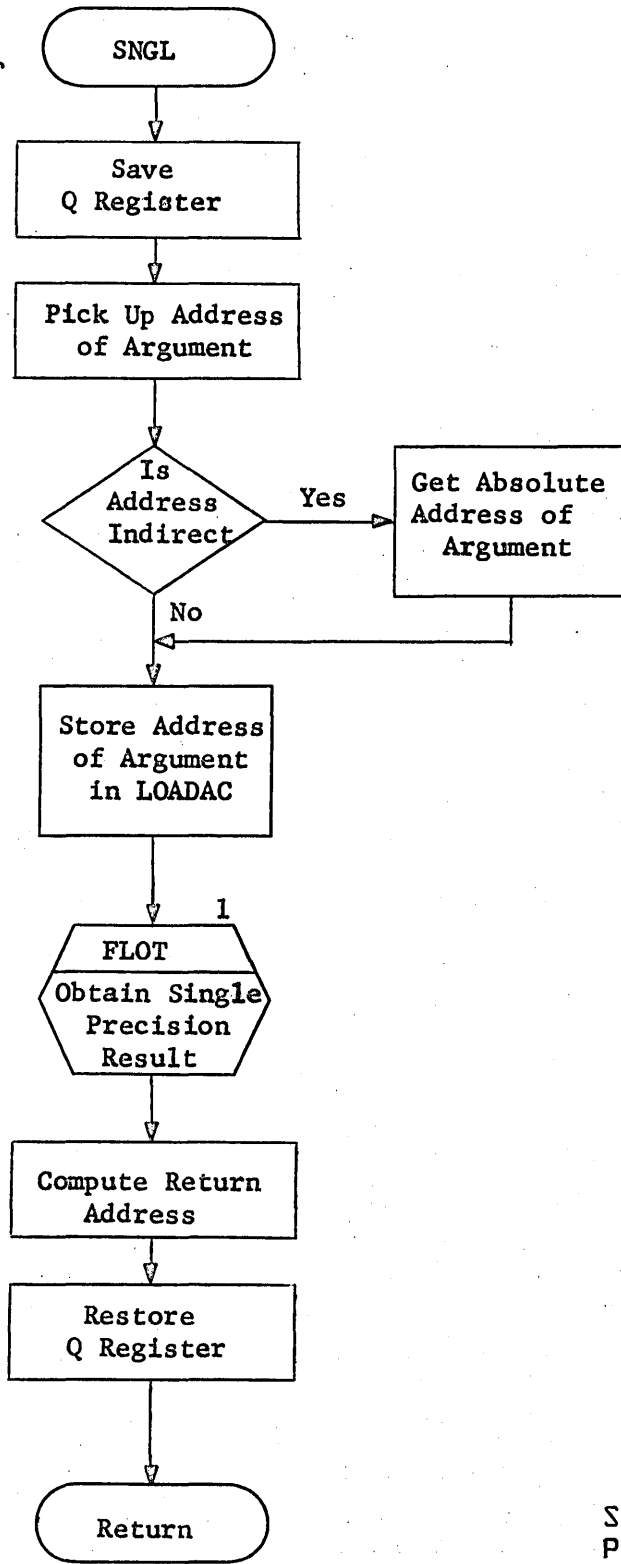
DOCUMENT CLASS IMS PAGE NO. 13-14  
 PRODUCT NAME 1700 MASS STORAGE FORTRAN  
 PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

1. Perform conversion of an integer to a floating point number. Input is in A register. Output is a floating point number contained in cells C5 & C6.

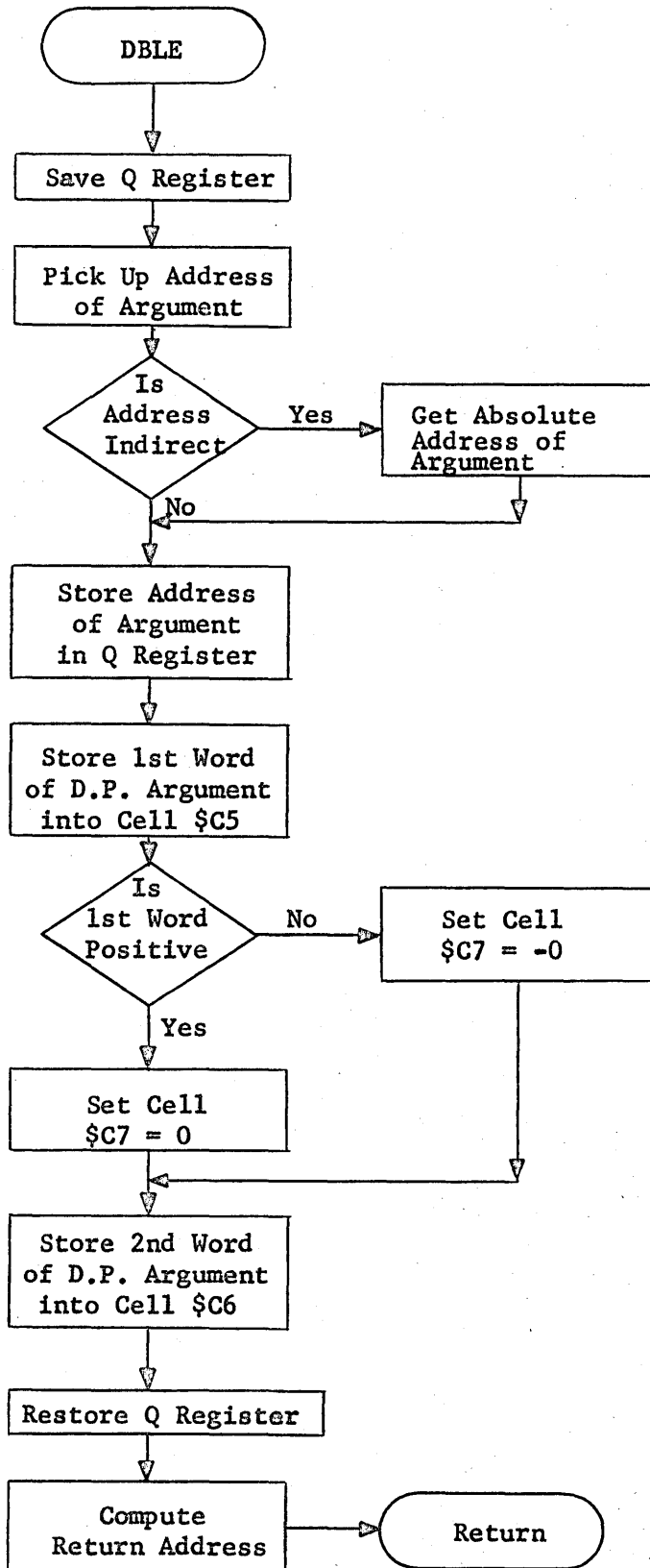


DOCUMENT CLASS IMS PAGE NO. 13-15  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

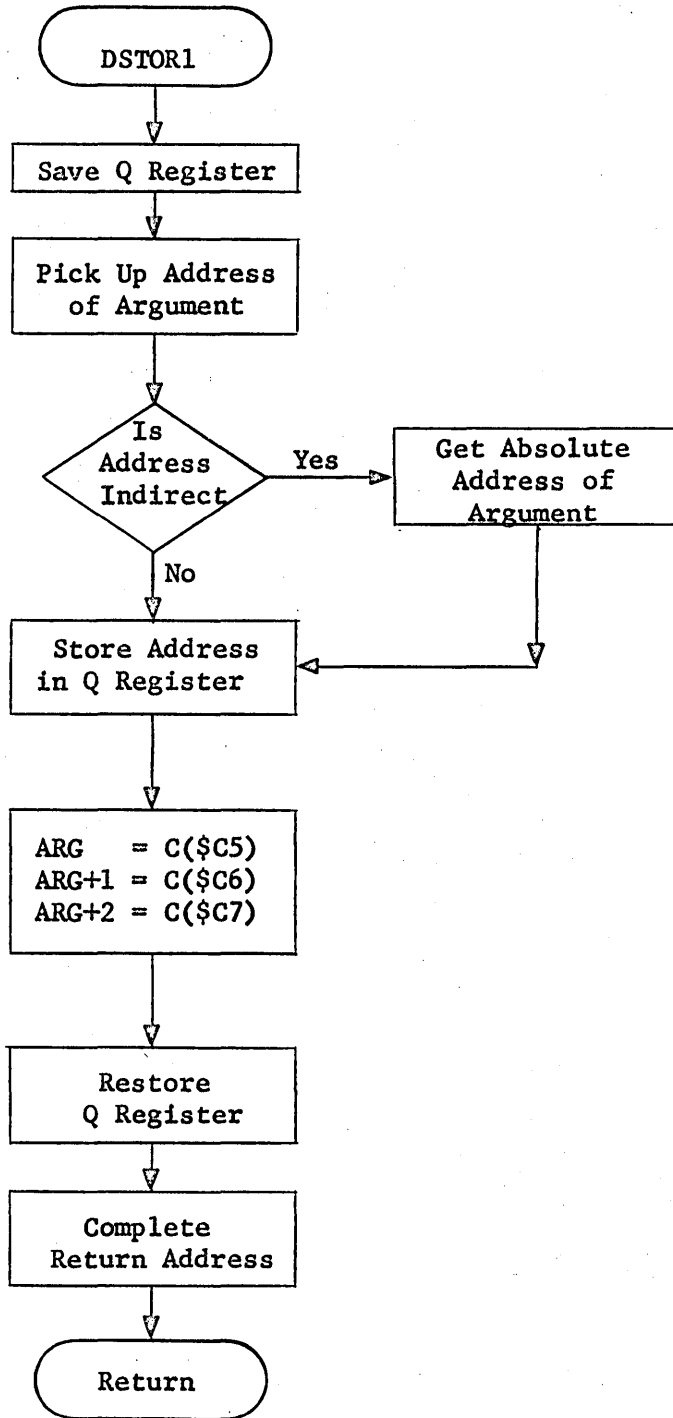
1. Do a floating point load of argument into pseudo accumulator {Cells #C5 & #C6}.



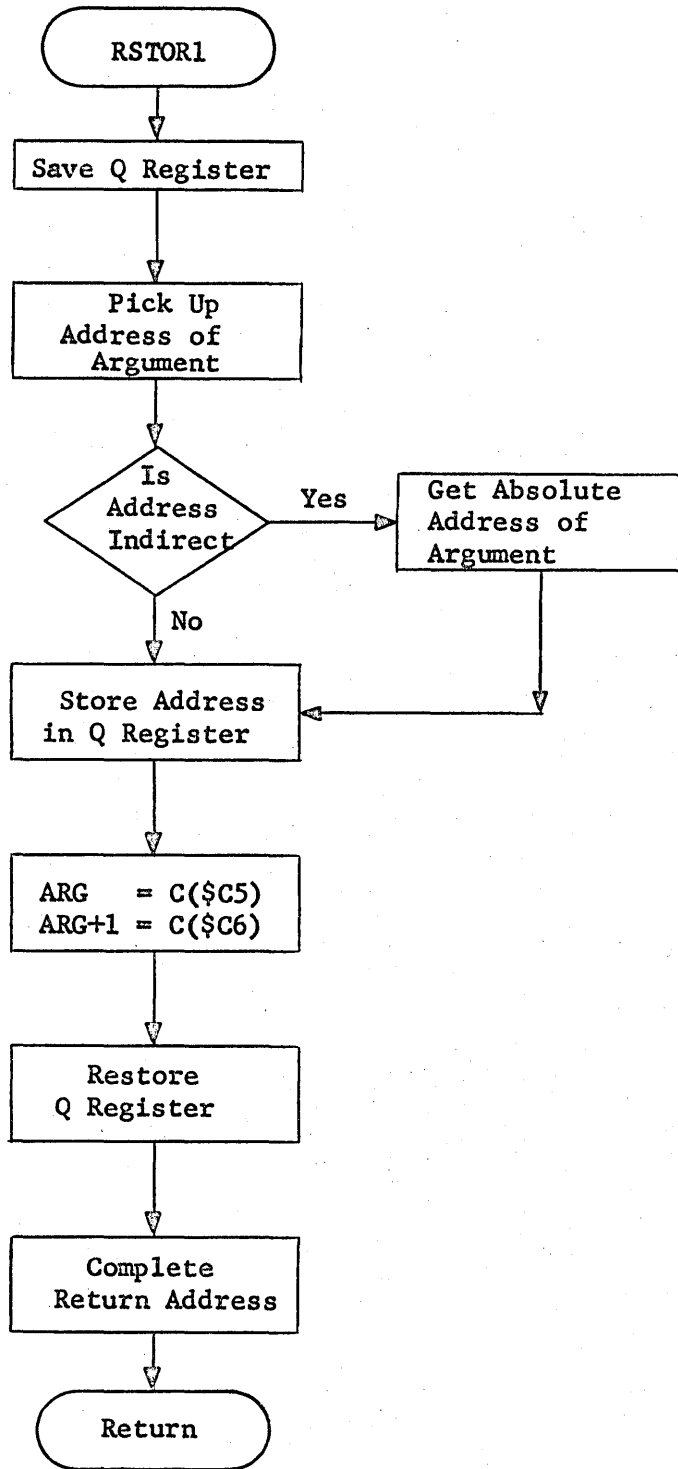
DOCUMENT CLASS IMS PAGE NO. 13-16  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700



DOCUMENT CLASS IMS PAGE NO. 13-17  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700



DOCUMENT CLASS IMS PAGE NO. 13-18  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700



DOCUMENT CLASS IMS PAGE NO. 13-19  
PRODUCT NAME 1700 MASS STORAGE FORTRAN  
PRODUCT MODEL NO. C005\*3.1 A/B MACHINE SERIES 1700

