# STANFORD PASCAL/360
# Implementation Guide

David L. Russell
Jeffrey Y. Sue

Computer Science Department
Stanford University

October 1974

--PRELIMINARY DRAFT--

Authors' current addresses:

David Russell, Digital Systems Laboratory,
Stanford University, Stanford, California 94305
Jeffrey Sue, UCLA School of Medicine,
Los Angeles, California 90025

This work was supported in part by the National Science Foundation under Grant No. GJ41644 and by the Atomic Energy Commission at the Stanford Linear Accelerator Center.

## INTRODUCTION

This guide describes an implementation of a PASCAL [1] compiler for IBM 360/370 series computers. This project is in a sense incomplete. Some features of the full language have been omitted, because they were not necessary for the completion of the bootstrap (from the CDC 6000 series computers [2]). Furthermore, there exist some known bugs and deficiencies in the system at the time of this writing. This manual is intended to act as a guide in modifying the compiler and correcting these errors. Since the compiler is written in PASCAL, and since the implementation is fairly efficient, this partial system will be a good starting point for the implementation of a compiler for the complete language.

The PASCAL compiler described here is a modification of the January 1972 version of the PASCAL/6600 compiler. The basic structure of that compiler is described in [3], and should be well understood before reading this report. Only those modifications of the compiler that were necessary in converting the system to a new computer will be detailed here. Thus, emphasis will be placed on the run-time environment, the addressing mechanisms, the interfaces to the operating system, and similar items.

## PROJECT HISTORY

The work described in this report was started in the fall of 1971, when Professor Niklaus Wirth was visiting at Stanford. At that time the authors, James Peterson, and Reinhard Wilhelm, all graduate students at Stanford, became involved. Work progressed slowly, due to other commitments, and by the summer of 1972, when Wirth returned to Zurich, the compiler consisted of much undebugged code in both PL/I and PASCAL. Many of the statement handlers, as well as the scanning routines and the declaration processing, had been completed.

At this point, Peterson and Wilhelm left the project, and no work was done during the summer. In the fall, the authors resumed work, and were able to debug most of the PL/I code that then existed. A simple monitor was written, and the first PASCAL programs were correctly translated and run in December 1972.

In January 1973 the first author left Stanford for six months, and work was suspended until the spring of 1974, at which time Ruby Lee joined the project. During the spring and summer, the remaining features necessary for bootstrapping the compiler were added and both the PASCAL and PL/I versions were completed and debugged. The compiler was successfully bootstrapped in July 1974.

Subsequently, Ruby Lee and the second author left the project, and the details of the compiler described in this report correspond to the version in existence at that time.

## LANGUAGE DIFFERENCES

The language implemented is the language described in the original definition of PASCAL [1], with the following exceptions:

- hexadecimal quantities may be specified
- braces { } to define comments are replaced by quotation marks " "
- brackets [ ] to define powersets are replaced by the notation SET(...)
- brackets [ ] to define array subscript expressions are replaced by parentheses ( )
- procedures and functions as formal parameters are not implemented
- real arithmetic is not completely implemented
- powersets are not implemented
- character I/O is not implemented
- no EOL character is defined
- not all arithmetic functions (SIN, COS, etc.) are implemented
- PACKED records are not implemented

Some of these changes clearly represent minor transliteration differences, and some represent features which were not implemented because they were not needed in order to complete the bootstrap to the 360/370.

## IMPLEMENTATION DETAILS

In this section, the details of the implementation of the October 1974 version of the compiler are described. Since it is structurally similar to the PASCAL/6600 compiler, only those parts of the compiler in which substantial differences exist will be described.

Many of the differences that do exist are due to the fact that one of the most important goals of this project was to maintain compatibility with the existing IBM operating systems. This implied that, as much as possible, the following requirements were met:

- OS procedure call conventions were followed
- OS input/output conventions were followed
- OS object modules were generated
- reentrant code was generated
- seperate compilation of global level procedures was allowed
- easy access to external program libraries was assured

With these goals met, a PASCAL program could automatically take advantage of the many services already offered by the operating system. In particular,

- the FORTRAN library would be available for the mathematical functions
- assembly language subroutines could be included
- input/output routines would be easily accessible
- the programs would run under ORVYL, a Stanford time-sharing service

While not all of the goals listed above have been completed in the first version, the design of the compiler is such that their addition is expected to be quite easy. Much of the following discussion will be centered around the facilities for interfacing to the operating system.

### Monitor

The monitor is a short assembly language program whose sole duty is to set up the environment for the PASCAL code, and to receive error conditions from the object code. The monitor performs the following functions:

1. Receives initial control from the operating system
2. Does a GETMAIN to get core for the run-time data area stack
3. Releases some of this core back to the operating system, to be used for buffers, DCBs, etc.
4. Sets up the registers properly

5. Does a BALR to the main program of the PASCAL code

Runtime environment and register assignment

A run-time stack is maintained in core. When a procedure (or function) is entered, a data area for that procedure is allocated from the stack area. The first 72 bytes of this data area are used for the OS save area, which is chained both forward and backward in standard OS style. The first word of the save area (which is unused in OS, except for PL/I programs), contains the static link, i.e. the address of the data area of the lexically enclosing procedure.

Parameters, if any, are placed into the data area starting at location 72 (after the save area). All other variables are assigned storage locations following the parameter list. No attempt is made to optimize the location of variables; they are assigned addresses in the same order that they are declared.

(Storage assignment at the global level is slightly different. There are no parameters, and space for fixed-to-float conversions and for the predefined input and output files is reserved.)

The compiler simulates a stack machine. Thus, the floating-point registers and some of the general purpose registers are used during execution to hold arithmetic quantities. The other general purpose registers are used to maintain the PASCAL environment.

R0,1 - temporary registers; contents may not be saved across a procedure call
R2 - points to the top of the run-time stack (first free doubleword)
R3,4 - base registers for the program segment
R5-11 - execution stack for temporary arithmetic and address quantities
R12 - points to the global data area
R13 - points to the local data area
R14,15- used for procedure calls, standard OS conventions

A diagram of the run-time stack appears in Figure 1.

Procedure entry and exit

The procedure entry and exit code follows standard OS conventions, and in addition, does the necessary work to maintain the internal PASCAL environment. (Procedure and function calls are described elsewhere.)

Procedure entry:

```
B     20(15)             Branch around entry id
DC    AL1(10)            Length of entry id
DC    CL10'proc name'    Entry id
DS    X                  Align to instruction boundary
DC    A(0)               Will contain length of data area
STM   14,12,12(13)       Save registers
LR    3,15               Program base register
LA    4,4095(0,3)        Second program base register
LA    4,1(0,4)
ST    13,4(0,2)          Dynamic back link
ST    2,8(0,13)          Dynamic forward link
LR    13,2               Address local data area
A     2,16(0,3)          Update pointer to top of stack
```

In addition,

Check for stack overflow, and exit if necessary

If at global level, perform data initializations, if any

Initialize files, if any

Initialize classes, if any

Procedure exit:

Close files, if any

```
L     13,4(0,13)         Load dynamic back link
LM    14,12,12(13)       Restore registers
MVI   12(13),X'FF'       Return flag
BR    14                 Return
```

Data objects

Data objects in the compiler have two attributes that affect the assignment of storage to them: length and alignment. The length and alignment of a variable are determined from the length and alignment of the corresponding type. For simple types, the alignment is equal to the length and represents the minimum space in bytes required to hold the item. Thus, for simple types, the following lengths (and alignments) are used:

char - byte aligned
tagfield, subrange, constant, integer - either halfword or fullword aligned, depentding on the range of the quantity
pointer - fullword aligned
real - doubleword aligned
Boolean - halfword aligned

The length and alignment of structured types is determined from the length and alignment of the component types. Procedure UPJUSTIFY is used to perform the alignment function when required. (Note that for the IBM 370 series, data alignment is not necessary, and thus procedure UPJUSTIFY may be replaced by a null procedure). Structured types are assigned the largest alignment of any of their component types. Classes and files are doubleword aligned. Records are assigned the length of their largest variant part; PACKED records are not implemented.

## Data addressing

### Constants

Constants appearing in the PASCAL source may be any length allowed for simple types. Thus there are four separate constant tables which are maintained, for halfword, fullword, doubleword, and alfa constants. In addition, a table for external references is kept. To help in using these constants, a definition of a 'constant' type was added:

```
TYPE CONSTKIND = (INTEGERS, REALS, ALFAS, CHARS,
                    SYMBOLICS, EXTREF);
     CONST = RECORD CASE KONSTKIND: CONSTKIND OF
               INTEGERS: CHARS: SYMBOLICS: (IVALUE: INTEGER);
               REALS: (RVALUE: REAL);
               ALFAS: (AVALUE: ALFA);
               EXTREF: (EVALUE: ALFA);
             END;
```

In addition, two procedures were defined:

```
PROCEDURE LDCST2(FVAL: CONSTANT; FRP:SHRTINT);
PROCEDURE LDCST(FVAL: CONSTANT);
```

LDCST2 loads a constant into a given register, and LDCST loads a constant onto the top of the register stack. The actual constant values are placed in the program segment, after the exit code for that procedure. V-type external references may be loaded by loading a constant where KONSTKIND=EXTREF, and the name of the external reference has been assigned to EVALUE. Appropriate ESD and RLD entries will then be made in the object deck.

External names and label processing

Each procedure is written out as a separate CSECT. However, because of the PASCAL scope rules, the name of a procedure is not sufficient to identify it to the system at link edit time. Therefore, every procedure, function, and exit label is assigned an external name which is unique to the program being compiled. This name is constructed by appending characters from a given sequence to a root, which is obtained from the name of the enclosing global level procedure. This allows global level procedures to be separately compiled, and linkedited with the rest of the object (or load) module. (Of course, the global declarations must remain constant). A new field SDNAME in the contexttable for procedures and functions contains the external name that has been constructed for the procedure or function.

Since exit labels must be declared, there is no problem in assigning them an external name. These external names are entered into array EXTAB of exit labels. Jumps to exit labels generate the adjustment of the runtime stack to maintain the environment of the target location, followed by a load of the external name (using LDCST2), and finally a branch. Ordinary jumps generate a simple branch instruction; undefined label references are chained through the code until defined by the occurrence of the label.

Global data

At the global level only, initial values may be declared for global variables. In order to maintain reentrancy, this data is compiled into a separate CSECT with external name $GBLDAT. It is then copied by the global level procedure entry code into the runtime stack, before proceding with execution.

Input/output

A file is described by a 16 byte file control block (FCB). The FCB is defined as follows:

```
FCB       DSECT                 File Control Block
FCBBUFF   DS     A              Pointer to buffer
*                                 in PASCAL data area
FCBDDNAM  DS     CL10           File name
FCBLRECL  DS     H              LRECL according to PASCAL
FCBBUFF   EQU    *              Buffer follows FCB
FCBOFLGS  EQU    FCBLRECL       Open flag
          ORG    FCBDDNAM       Overwritten in open code
FCBWORK   DS     A              Pointer to save area and DCB
FCBCOUNT  DS     F              Number of char left in buffer
*                                  (Character files only)
*
WORKAREA  DSECT                 Save area and DCB
WORKSA    DS     18F
WORKDCB   DCB    DSORG=PS,(PM)    Or (GM), if get
```

For the very simple-minded I/O modules currently implemented, the following must be kept in mind:

- The buffer must follow the FCB, and location 0 in the FCB must point at it. The length of a file is thus (16 + the length of the buffer). The file, i.e. the FCB, is doubleword aligned.

- The filename must be at location 4 of the FCB. The first 8 characters are the OS ddname.

- The buffer length must be in location 14 of the FCB, and must equal the lrecl defined in the JCL. Further, the high-order bit of this field must be a 0.

- When the I/O routine is first entered, a work area is obtained, and an OS OPEN macro is executed. If the open is successful, the address of the workarea will be stored into the FCB, overwriting the first four bytes of the filename. In addition, the high-order bit of the lrecl field will be set to indicate that the file has been opened.

- The I/O routines expect to gain control with the address of the FCB in R1. LDCST2 is used to load the proper module entry point, which, for record files, is either PSCLGETR or PSCLPUTR.

Character files are not currently implemented, but may be simulated in the following manner:

```
TYPE CARD = ARRAY (1..80) OF CHAR;
VAR SYSIN: FILE OF CARD; I: INTEGER; C: CHAR;
GET(SYSIN);
C := SYSIN (I);  "ITH CHARACTER OF THE INPUT CARD"
```

## Procedure and function calls

The argument list is constructed starting at location 72 past register 2. The arguments will then be in the data area of the called procedure. VAR parameters and structured parameters are passed by reference. CONST parameters of simple type are passed by value.

The returned value of a function is stored in the data area of the calling procedure. The first parameter of the function is implicitly defined to be a pointer to the returned value. In the case of nested function calls, the value of the stacktop pointer in R2 is adjusted so that the partially constructed parameter list will not be overwritten by a function call within the argument list.

After the parameters have been set up, the static link is fixed, and the procedure is then called, using the external name stored in field SDNAME of the contexttable.

## Classes

Classes have an 8 byte descriptor which is initialized at run time. The first word of the descriptor points to the first free location in the area reserved for the class. The second word points to the first location after the class. The code generated for ALLOC tests if all the space for the class has been used, and if so, a null pointer is returned; otherwise, a pointer to the allocated record is returned.

records off end of input file'.

## FUTURE WORK

In order to expedite the actual bootstrapping of the compiler, the development and testing of many features was deferred. Also, some non-critical bugs were found. (Non-critical in the sense that bootstrapping was not affected). These features and bugs range from items which are very important for the practical use of the compiler, to items which can only be considered as 'bells and whistles'. A list of the known problems follows, in approximately their order of importance. In some cases, hints have been given as to potential solutions.

1. The I/O routines need improvement. Better checking of DCB parameters on the DD statement should be done, and similar sections of the I/O routines should be combined. If the user forgets the period at the end of a program, or makes some other mistake which causes the compiler to read past the end of the input text, the compiler ABENDS. An EODAD exit should be defined for all input files, and the EOF feature, not currently used, should be added. Also, CLOSEs are now done implicitly by the operating system; explicit CLOSEs should be done when leaving the scope of a declared file.

2. The monitor also needs improvement. The monitor should be able to trap program checks and ABENDs and format a simple message, instead of just creating a dump. This is especially true for common problems like using a NIL pointer, running out of stack space, or dividing by zero.

3. The compiler should keep a count of the syntax errors, and print a message at the end of compilation. A return code should be returned to OS. Any line that is in error should be printed, even if the listing has been turned off.

4. Real arithmetic needs to be debugged. There are known problems in the comparison code for real numbers, when automatic fixed-to-float conversions are required. This arises because there are two stacks, and moving an item from one stack to the other potentially changes the relative order of the operands; consider, for instance, the expression (I<X), with I integer, X real. Perhaps automatic conversions should be provided only for assignments; other coercions would have to be made explicitly.

5. A means to link to the routines in the FORTRAN library should be implemented. A scheme similar to ALGOLW might be possible, and then common functions (SQRT, SIN, etc.) would be predefined to be FORTRAN type. Since real OS object modules are generated, this modification would greatly extend the capabilities of the compiler. To do this requires that

the FORTRAN I/O package (ERRMON, etc.) be bypassed somehow.

6. Powersets currently do not work. The contexttable entries are set up, but the wrong code is generated.

7. Boolean expressions and jumps generate extremely inefficient code., since a true or false value is always generated. Some arithmetic operations generate condition codes themselves; the LCOND variant of the ATTR records may be used to record this information, and some jump sequences would be shorter. This improvement would help FOR, WHILE, and REPEAT statements, as well as IF-THEN-ELSE statements.

8. Routines to READ an item, WRITE an item, and to convert from floating to fixed should be written. Currently, users are required to format their output, and decode their input, by writing the proper routines in PASCAL.

9. Preliminary tests with PROGLOOK indicate that about 44 per cent of the time spent in the PASCAL compiler is used in searching the symbol table. Some form of hashing would be a great help, but probable difficult to fit into the general contexttable organization.

10. INSYMBOL has an inaccurate algorithm for building up the value of a real number. It should be made smarter.

11. Array addressing uses shifts to calculate some element addresses. In some cases, it would require less code, but more time, to use a MH instruction. The tradeoffs should be examined to determine which method is better.

12. Using shifts in TERM for multiplication by certain constants (also divide, DIV, and MOD) should be considered.

13. All fixed-point multiplication and division is now done by using R0-R1 as the double length pair. Since the registers are used in decresing order, some of the time a load of R0 or R1 could be avoided.

14. Keep some addresses in the registers. An address stack (in the general purpose registers) may be defined which occupies the same space as, but grows in a direction opposite to, the expression stack. Addresses and 4K constants may be put in these registers. As long as a branch is not made which invalidates these addresses, they may be used in later instructions. A descriptor (as yet undefined) for these quantities may reside in array ASTK. This address mechanism needs to be developed (or redesigned). Avoiding repeated loads of the same base addresses or 4K constants is expected

to have a very beneficial effect on the WITH construct, and on some array accessing.

15. Multiple errors at the same input position are marked only once in the listing. If errors occur at two or more different points in the input line, and more error numbers are listed than errors are marked, then it is difficult to know which marks go with which errors.

16. Procedures and functions as parameters need to be implemented.

17. Object decks currently have only one ESD or one RLD entry per card; this should be fixed. The deck should also be sequence numbered.

18. Nested function calls are allowed, but may run out of registers if the complexity is too great. This restriction could be eliminated, or made optional.

19. Parameter lists are limited to 4K in length.

20. Currently, a stack type machine is simulated. All operations are done between operands in the registers. It should be possible to use RX type instructions in some cases to reduce the size of the generated code.

21. (Single) character files are not currently implemented.

22. Some arrays may be directly addressable, while others are assigned addresses with a displacement into the data area of more than 4K. It might be possible to decide at compile-time which arrays should be directly addressable, (on the basis, say, of frequency of access), and assign the actual location of the arrays accordingly.

23. ALFA types should be removed and replaced by ARRAY OF CHAR. A generalized MOVE built-in procedure could replace the UNPACK and PACK procedures, if desired.

24. The concept of CLASS could be removed, as in the revised PASCAL language. This might require considerable run-time support and major changes to the compiler.

25. Records of a class are allocated the minimum amount of storage necessary, depending on the variants specified. Because of this, assignment of one record to another is not always valid, even if they are ostensible the same type, unless the length of the source is less than the length of the destination. Records which are not allocated from a

class can be assigned to each other (if they are the same type), since they are always allocated the maximum space of any variant.

Record comparisons are dangerous because unpacked records can have gaps due to alignment which may contain trash at run time. It is probably best to just disallow such assignments and comparisons.

26. The compiler options which generate code to perform run-time index checking, overflow checking, etc. have not been implemented. This would necessitate changes in the monitor as well.

ACKNOWLEDGEMENTS

The authors would like to thank Professor N. Wirth for his help in starting this project and his interest throughout its development, Professor T. H. Bredt for his continued encouragement, and those who at one time or another contributed to the project: Reinhard Wilhelm, Ruby Lee, James Peterson, and John Montague.

The authors are deeply indebted to the Computer Science Department of Stanford University and the Stanford Center for Information Processing for contributing the machine time and resources for this work.

REFERENCES

1. N. Wirth. 'The programming language PASCAL', Acta Informatica, 1, 35-63(1971).

2. D. Russell and J. Sue. 'Implementation of a PASCAL compiler for the IBM/360', Technical Note, Digital Systems Laboratory, Stanford, California, in preparation.

3. N. Wirth. 'The design of a PASCAL compiler', Software - Practice and Experience, 1, 309-333(1971).

## APPENDIX A. Compiler options

Compiler options may be specified by placing option switches inside a comment. The first character of the comment must be '$', followed by a sequence of option switches, separated by commas. Each option switch takes the form Ax, where A is a given option that is turned on if x='+' and turned off otherwise.

There are currently only three options that are implemented. L controls listing of the source text, P controls printing of the compiled machine code and constants at the end of a procedure, and C controls printing of compiled code during its actual generation (fixups will not have been made).

Thus, for example, the option string "$L-,P+" would turn off the source listing but print the generated code for the remaining procedures.

The default value for L is to produce a listing; the default value for P and C is not to print the generated instructions.

## APPENDIX B. Sample program and compiler output

A sample program is given below, along with the JCL necessary to run it on the system at SLAC. The exact JCL used at a particular installation will of course vary, but this example can act as a model. Although this job uses the linkage editor, the loader could also have been used.

```
//        JOB ,CLASS=E
//PASCAL EXEC PGM=PASCAL
//STEPLIB  DD DSN=WYL.SF.PAS.PASCALIB,DISP=SHR
//SYSPRINT DD SYSOUT=A,
//            DCB=(RECFM=FBA,LRECL=121,BLKSIZE=1452,BUFNO=2)
//SYSUDUMP DD SYSOUT=A
//PASCALGO DD DSN=&LOADSET,SPACE=(TRK,(5,1),RLSE),UNIT=SYSDA,
//            DISP=(,PASS),DCB=(BLKSIZE=400,RECFM=FB,LRECL=80)
//SYSIN    DD *
TYPE XX=ARRAY(1..8)OF INTEGER;
VAR A:XX;
    SYSPRINT:FILE OF ARRAY(0..120)OF CHAR;
   SYSIN:FILE OF ARRAY(0..79) OF CHAR;
    I,J:INTEGER;
PROCEDURE CLEAR;
   VAR I:INTEGER;
   BEGIN
      FOR I:=0 TO 120 DO SYSPRINT (I):=' ';
   END "CLEAR";
   "$P+"
PROCEDURE PRINT(VAL,PLACE:INTEGER);
   VAR I,J:INTEGER;
   BEGIN I:=VAL;   J:=PLACE;
      REPEAT
         SYSPRINT (J):=CHR(240+(I MOD 10));
         I:=I DIV 10;
         J:=J-1;
      UNTIL I<=0;
   END "PRINT";
    "$P-"
PROCEDURE PRINTC(A:XX; N:INTEGER);
   BEGIN
      PRINT(A(N),N*4);
   END "PRINTC";
BEGIN "PASCAL TRIANGLE"
   GET(SYSIN);
   CLEAR;
   FOR I:=0 TO 79 DO SYSPRINT (I):=SYSIN (I);
   PUT(SYSPRINT);
   FOR I:=1 TO 8 DO
      BEGIN IF I=1 THEN A(1):=1 ELSE A(I):=0;
```

```
         CLEAR;
         FOR J:=I DOWNTO 2 DO
            A(J):=A(J)+A(J-1);
         FOR J:=1 TO I DO
            PRINTC(A,J);
         PUT(SYSPRINT);
      END;
END.
//LKED    EXEC PGM=IEWL,REGION=150K,TIME=(,30),PARM='LET,LIST,MAP'
//SYSLIB    DD DSN=WYL.SF.PAS.PSCLLIB,DISP=SHR
//SYSLIN    DD DSN=&&LOADSET,UNIT=SYSDA,DISP=(MOD,DELETE,DELETE),
//             SPACE=(TRK,(0))
//SYSLMOD   DD DSN=&&GOSET(MAIN),UNIT=SYSDA,DISP=(NEW,PASS,DELETE),
//             SPACE=(TRK,(100,20,1),RLSE,,ROUND)
//SYSPRINT DD SYSOUT=A,DCB=(RECFM=FBM,LRECL=121,BLKSIZE=1573),
//             SPACE=(CYL,(1,1),RLSE)
//SYSUT1    DD UNIT=SYSDA,SPACE=(TRK,(100,20))
//GO      EXEC PGM=*.LKED.SYSLMOD,REGION=150K,COND=(5,LT,LKED)
//SYSPRINT DD SYSOUT=A,
//             DCB=(RECFM=FBA,BLKSIZE=1452,LRECL=121,BUFNO=2)
//SYSUDUMP DD SYSOUT=A
//SYSIN     DD *
 HELLO GEORGE
```

The compiler output for this program appears below:

```
000000 TYPE XX=ARRAY(1..8)OF INTEGER;
00007C VAR A:XX;
00009C     SYSPRINT:FILE OF ARRAY(0..120)OF CHAR;
000125    SYSIN:FILE OF ARRAY(0..79)  OF CHAR;
000188     I,J:INTEGER;
000190 PROCEDURE CLEAR;
000000    VAR I:INTEGER;
00004C    BEGIN
000040       FOR I:=0 TO 120 DO SYSPRINT (I):=' ';
000070    END "CLEAR";
000084    "$P+"
000084 PROCEDURE PRINT(VAL,PLACE:INTEGER);
000084    VAR I,J:INTEGER;
000058    BEGIN I:=VAL;   J:=PLACE;
000050       REPEAT
000050          SYSPRINT (J):=CHR(240+(I MOD 10));
000076          I:=I DIV 10;
00008C          J:=J-1;
000096       UNTIL I<=0;
0000AC    END "PRINT";
```

```
CONSTANTS
   0000C0  DC X' 00 00 00 00'

 000000     0     BC        15,  20( 0, 15)
 000014    20     STM       14, 12,  12( 13)
 000018    24     LR        3, 15
 00001A    26     LA        4, 4095( 0, 3)
 00001E    30     LA        4,    1( 0, 4)
 000022    34     ST        13,   4( 0, 2)
 000026    38     ST        2,    8( 0, 13)
 00002A    42     LR        13, 2
 00002C    44     A         2,   16( 0, 3)
 000030    48     C         2,   80( 0, 12)
 000034    52     BC        12,  64( 0, 3)
 000038    56     SR        1, 1
 00003A    58     L         15, 192( 0, 3)
 00003E    62     BCR       15, 15
 000040    64     L         11,  72( 0, 13)
 000044    68     ST        11,  80( 0, 13)
 000048    72     L         11,  76( 0, 13)
 00004C    76     ST        11,  84( 0, 13)
 000050    80     L         11, 156( 0, 12)
 000054    84     L         10,  84( 0, 13)
 000058    88     AR        11, 10
 00005A    90     LA        10, 240( 0, 0)
 00005E    94     L         9,   80( 0, 13)
 000062    98     LA        8,   10( 0, 0)
 000066   102     LR        0, 9
 000068   104     SRDA      0, 0,  32( 0)
 00006C   108     DR        0, 8
 00006E   110     LR        9, 0
 000070   112     AR        10, 9
 000072   114     STC       10,    0( 0, 11)
 000076   118     L         11,  80( 0, 13)
 00007A   122     LA        10,  10( 0, 0)
 00007E   126     LR        0, 11
 000080   128     SRDA      0, 0,  32( 0)
 000084   132     DR        0, 10
 000086   134     LR        11, 1
 000088   136     ST        11,  80( 0, 13)
 00008C   140     L         11,  84( 0, 13)
 000090   144     BCTR      11, 0
 000092   146     ST        11,  84( 0, 13)
 000096   150     L         11,  80( 0, 13)
 00009A   154     LTR       11, 11
 00009C   156     LA        11,   1( 0, 0)
 0000A0   160     BC        12, 166( 0, 3)
 0000A4   164     SR        11, 11
 0000A6   166     LTR       11, 11
 0000A8   168     BC        8,   80( 0, 3)
```

```
0000AC   172    L         13,   4( 0, 13)
0000B0   176    LM        14, 12,  12( 13)
0000B4   180    MVI        12( 13), 255
0000B8   184    BCR       15, 14
0000C4     "$P-"
0000C4 PROCEDURE PRINTC(A:XX; N:INTEGER);
0000C4    BEGIN
000040       PRINT(A(N),N*4);
000078    END "PRINTC";
000092 BEGIN "PASCAL TRIANGLE"
0000A8    GET(SYSIN);
0000B2    CLEAR;
0000BC    FOR I:=0 TO 79 DO SYSPRINT (I):=SYSIN (I);
0000FA    PUT(SYSPRINT);
000104    FOR I:=1 TO 8 DO
000116       BEGIN IF I=1 THEN A(1):=1 ELSE A(I):=0;
00014A          CLEAR;
000154          FOR J:=I DOWNTO 2 DO
000166             A(J):=A(J)+A(J-1);
000196          FOR J:=1 TO I DO
0001AC             PRINTC(A,J);
0001D8          PUT(SYSPRINT);
0001E2       END;
0001F0 END.
```

The linkage editor output is:

```
 F64-LEVEL LINKAGE EDITOR OPTIONS SPECIFIED LET,LIST,MAP
         DEFAULT OPTION(S) USED -  SIZE=(307200,61440)
IEW0000     INCLUDE SYSLIB(PSCLMON)
IEW0000     ENTRY PSCLMON

                                                MODULE MAP

  CONTROL SECTION                          ENTRY
    NAME    ORIGIN   LENGTH                  NAME   LOCATION     NAME
  CLEAR$$        00       84
  PRINT$$        88       C4
  PRINTC$       150       92
  $$$MAIN       1E8      23C
  PSCLMON       428       CE
                                           PSCLERR      4C6
  PSCLGETR*     4F8      124
  PSCLPUTR*     620      12C
```

```
 ENTRY ADDRESS       428
 TOTAL LENGTH        750
****MAIN       DOES NOT EXIST BUT HAS BEEN ADDED TO DATA SET
AUTHORIZATION CODE IS          0.
```

The output from the execution of this job is:

```
HELLO GEORGE
    1
    1    1
    1    2    1
    1    3    3    1
    1    4    6    4    1
    1    5   10   10    5    1
    1    6   15   20   15    6    1
    1    7   21   35   35   21    7    1
```

## APPENDIX C. Separate compilation of global procedures

## APPENDIX D. Description of distribution tape

```
1.
2.              "*************************************************
3.               *                                               *
4.               *          P A S C A L   C O M P I L E R        *
5.               *          *****************************        *
6.               *                                               *
7.               *                ( JANUARY 1972 )               *
8.               *                                               *
9.               *  AUTHORS:  U. AMMANN,  R. SCHILD              *
10.              *                                               *
11.              *            FACHGRUPPE COMPUTERWISSENSCHAFTEN  *
12.              *            EIDG. TECHNISCHE HOCHSCHULE        *
13.              *     CH-8006 ZUERICH                           *
14.              *                                               *
15.              *                                               *
16.              *           IBM 360 VERSION  SEPTEMBER 1974     *
17.              *                                               *
18.              *  AUTHORS:  D. RUSSELL, J. SUE, R. LEE,        *
19.              *            R. WILHELM                         *
20.              *                                               *
21.              *            COMPUTER SCIENCE DEPARTMENT        *
22.              *            STANFORD UNIVERSITY                *
23.              *            STANFORD, CALIFORNIA  94305        *
24.              *                                               *
25.              *************************************************"
26.
27.
28.          CONST MAX10 = 2147483647, MAXHALF = 32767, TWOTO12 = 4096,
29.             MAXISTK = 7, MAXRSTK = 4, MAXBASEREG = 2,
30.             MAXLABS = 30, MAXEXLABS = 15, CASMAX = 30,
31.             CSTMAX = 40, CODMAX = 8191,
32.             DISPLIM = 20, MAXLEVEL = 10, PTLIMIT = 10, FILLIMIT = 10,
33.             DISPLIMIT = 4095, WORDLENGTH = 32, ALFALENG = 10,
34.             JMPMAX = 49, UNDMAX = 70,
35.
36.             INPT = 84;  OUTPT = 104; "ADDRESSES OF FCBS FOR INPUT,OUTPUT"
37.
38.             GLOBALREG = 12, LOCALREG = 13;
39.
40.          TYPE CTP = @CONTEXTTABLE;
41.             CWORD = ARRAY (1..10) OF CHAR;
42.             PRINTLINE = ARRAY(0..120) OF CHAR;
43.             CARDIMAGE = ARRAY(1..80) OF CHAR;
44.             ADDRESS = 0 .. 16777216;
45.             SHRTINT = -MAXHALF .. MAXHALF;
46.             BITRANGE = 0 .. 80;  RG3 = 0 .. MAXLEVEL;
47.             BYTE = 0..255;
48.             ATTRKIND = (VARBL,SVAL,LVAL,LCOND);
49.             ATTR = RECORD TYPTR : CTP ;
50.                       CASE KIND : ATTRKIND OF
51.                          VARBL: (ACCESS : (DRCT,INDRCT,INXD) ;
52.                                  BREG:RG3 ; DPLMT:ADDRESS; ALIGNMENT:SHRTINT ;
53.                                  CASE PCKD : BOOLEAN OF
54.                                     FALSE: ;
55.                                     TRUE: (BITADR, BITSZ : BITRANGE)) ;
56.                          SVAL:  (VAL : SHRTINT) ;
57.                          LVAL:  (CTERM : INTEGER) ;
58.                          LCOND: (JMP : 0 .. 3 ; ARITH : BOOLEAN)
59.                    END ;
```

```
60.          "DESCRIBES EXPRESSIONS TO BE COMPILED.    EXAMPLES:
61.              VARBL:        DRCT:                   I                **)
62.                            INDRCT:                 INPUT@            *)
63.                            INXD:                   A(K)              *)
64.              SVAL:                                 4                **)
65.              LVAL:                                 2*I - 1           *)
66.              LCOND:        ARITH:                  X >= 4.1           *)
67.                           ¬ARITH:                  B < TRUE          *)
68.
69.              *) RESULT OF CODEGENERATION IN REGISTER X-RP
70.             **) NO CODE GENERATION UNTIL NOW"
71.
72.
73.          OPTPWR = (NOOPT,PUREP,POSP,NEGP);
74.          IDCLASS = (TYPES,KONST,PROC,VARS,FIELD,TAGFIELD,DUMMYCLASS);
75.          TYPFORM = (NUMERIC,SYMBOLIC,POINTER,POWER,
76.                     ARRAYS,RECORDS,CLASSS,FILES);
77.          IDKINDS = (ACTUAL,FORMAL);
78.          WHERE = (BLOCK,CWITH,VWITH);
79.          CONSTKIND = (INTEGERS, REALS, ALFAS, CHARS, SYMBOLICS,EXTREF);
80.          CONSTANT = "PACKED" RECORD CASE KONSTKIND : CONSTKIND OF
81.             INTEGERS : CHARS : SYMBOLICS : (IVALUE : INTEGER);
82.             REALS : (RVALUE : REAL);
83.             ALFAS : (AVALUE : ALFA);
84.             EXTREF : (EVALUE : ALFA);
85.          END;
86.
87.       VAR
88.          ONE, TEN, TENTH : REAL;  "CONSTANTS FOR INSYMBOL"
89.          ISTKLIM : SHRTINT;  NILVAL : CONSTANT;
90.          JMPTAB : ARRAY (0..JMPMAX) OF INTEGER;
91.          JMPIX : SHRTINT;
92.          "TRANSFER VECTOR FOR CALLS OF FORWARD DECLARED PROCEDURES AND
93.           GOTO STATEMENTS LEADING OUT OF PROCEDURES"
94.
95.          PILEV,FILEV : ARRAY (0..MAXLEVEL) OF SHRTINT;
96.          PFL,PEND : ARRAY(0..FILLIMIT) OF ADDRESS;
97.          FILPTS : ARRAY (0..FILLIMIT) OF CTP;
98.          XFILPT : CTP;
99.          PFTOP,FILTOP : SHRTINT;
100.         "PFL CONTAINS THE ADDRESSES OF LOCAL CLASSES,
101.            PILEV CONTAINS POINTERS INTO PFL AND PEND, PFTOP = TOP OF PFL.
102.          FILPTS CONTAINS POINTERS TO ENTRIES FOR LOCAL FILES,
103.            FILEV CONTAINS POINTERS INTO FILPTS, FILTOP = TOP OF FILPTS"
104.
105.         EXTAB : ARRAY (1..MAXEXLABS) OF
106.            RECORD EXVAL,EXLEVEL : SHRTINT; EXTNAME : ALFA   END;
107.         CEXTABIX : SHRTINT;
108.         FSTIXG : SHRTINT;
109.         "CONTAINS THE EXPLICITLY DECLARED LABELS OF ALL PROCEDURES
110.          NOT YET CLOSED, TOGETHER WITH THEIR CORRESPONDING INDEX
111.          INTO JMPTAB"
112.
113.         LABTAB : ARRAY (1..MAXLABS) OF
114.            "PACKED" RECORD LABVAL,LABLOC,CHAIN : SHRTINT  END;
115.         CLABIX : SHRTINT;
116.         "CONTAINS ALL LABELS MET SO FAR IN THE BODY OF THE PROCEDURE
117.          ACTUALLY BEING COMPILED, TOGETHER WITH INFORMATION WHETHER
118.          LABEL DEFINITION (<LABEL>:) ALREADY FOUND OR NOT (FLD2).
119.          IN THE FORMER CASE FLD3 CONTAINS THE CORRESPONDING ADDRESS,
```

```
 WHERE IN THE LATTER CASE FLD3 CONTAINS AN INDEX INTO UNDLAB
 WHERE THE OCCURRENCES ARE CHAINED"

UNDLAB : ARRAY (1..UNDMAX) OF
   "PACKED" RECORD SUCC, PLACE : SHRTINT ;
             LFTSH : BITRANGE ;
          END ;
CHNIX : SHRTINT;
"ACTS AS LISTSTRUCTURE, CHAINING OCCURRENCES OF CONSTANTS AND
 JUMPS TO NOT YET REACHED LABELS IN THE CODE OF THE PROCEDURE
 ACTUALLY BEING COMPILED.
 CHNIX = HEAD OF FREE LIST"

PTLIST : ARRAY (0..PTLIMIT) OF
   RECORD HNAME : ALFA;  PPTR : CTP  END;
PTX : SHRTINT;
"PTLIST CONTAINS NAMES OF YET UNDECLARED CLASSES AND
   POINTERS TO THE CORRESPONDING POINTER-TYPE ENTRIES
 PTX = TOP OF PTLIST "

ERRLIST : ARRAY (1..10) OF
   "PACKED" RECORD POS,NMR : SHRTINT  END;
ERRNRS : ARRAY (0..5) OF POWERSET 0..30 ;
ERRINX,POS1,CHCNT : SHRTINT;  EOLFLAG,ERR,COMPILING : BOOLEAN;
CHPTRX,PTOUT : SHRTINT;
"ERRLIST CONTAINS THE POSITIONS AND NUMBERS OF THE ERRORS
   ON ONE LINE
   ERRINX = TOP OF ERRLIST
   POS1 = POSITION OF LAST PRINTED ERRORMARK ( @ )
   CHCNT = POSITION OF LAST READ CHARACTER
 ERRLIST,ERRINX,POS1,CHCNT,EOLFLAG ARE USED BY NEXTCH
   AND ERROR.
 ERR IS SET WHENEVER ERROR HAS BEEN CALLED AND IS TESTED
   (AND RESET) BY SEVERAL PROCEDURES "

DISPLAY : ARRAY (0..DISPLIM) OF
"PACKED" RECORD FNAME : CTP;
      CASE OCCUR : WHERE OF
         BLOCK : ;
         CWITH : (CDSPL : ADDRESS; CLEV : RG3);
         VWITH : (VDSPL : ADDRESS);
END;
TOP,DISX : SHRTINT;
"TOP = TOP OF DISPLAY, DISX IS RETURNED BY SEARCH"

CA : SHRTINT;

LC,IC : ADDRESS;
"LOCATION COUNTER AND INSTRUCTION COUNTER"

LEVEL,RP,RP1,RRP : SHRTINT;
"RP,RP1 = REGISTER-POINTERS (STACK OF X-REGS)"
DP,ENDFLAG : BOOLEAN ;
"DP = TRUE : DECLARATION PART (USED BY NEXTCH)"
"ENDFLAG : TO TEST FOR END. "

PRCODE,ASSCHECK,INXCHECK,DIVCHECK,STOFLCHECK,
TRACECHECK,PCODE,LISTING : BOOLEAN ;   "COMPILER OPTIONS"
VDATA : BOOLEAN ;   "SET TRUE IF VALUE INITIALIZATIONS"
ASSERR: SHRTINT;
```

```
180.          GLOBALNAME,UNIQUENAME : ALFA;
181.
182.          GATTR : ATTR;   "GLOBAL ATTRIBUTE RECORD "
183.
184.          SSADDR : RECORD  B2,D2 : SHRTINT  END;
185.          RXADDR : RECORD  B2,D2,X2 : SHRTINT  END;
186.
187.          INTPTR,REALPTR,ALFAPTR,CHARPTR,BOOLPTR,NILPTR,
188.             UNDECPTR,PNUMPTR,EXTPTR,LAMPTR : CTP; "CONSTANT POINTERS"
189.
190.          NEXT,CTPTR : CTP;
191.          "VARIABLES POINTING INTO CONTEXTTABLE"
192.
193.          CH,CHVAL : CHAR; AVAL : ALFA; IVAL : INTEGER; RVAL : REAL;
194.          A : CWORD;
195.          NO,CL : SHRTINT;
196.          "CH IS OUTPUT BY NEXTCH AND USED BY INSYMBOL.
197.             AVAL/IVAL, NO, CL  ARE OUTPUT BY INSYMBOL"
198.
199.          TCT,TMAX,B6DPL,KK : ADDRESS ;
200.          PT : CTP; IT,IT1 : INTEGER;
201.       "AUXILIARY VARIABLES, USED IN MAIN PGM AND SEVERAL PROCEDURES"
202.
203.
204.          "CONSTANTS AND CONSTANT TABLES: "
205.          SPLITSTAT : ARRAY (0..48) OF SHRTINT;
206.          ERRCL,TERRCL : ARRAY (0..48) OF (IRRELSY,BEGSY,ENDSY);
207.          BLANK,BLANKALFA : ALFA;
208.          BLANKLINE : PRINTLINE; BLANKCARD : CARDIMAGE;
209.          WD : ARRAY (0..33) OF ALFA;
210.          WNO,WCL : ARRAY (0..33) OF SHRTINT;
211.          WL : ARRAY (0..11) OF SHRTINT;
212.          " WD = WORD DELIMITERS
213.           WNO,WCL = CORRESPONDING VALUES OF  NO  AND  CL
214.           WORD DELIMITERS OF LENGTH  I  START IN WD(WL(I))  "
215.          CHARTYPE,SYMNO,SYMCL : ARRAY (CHAR) OF BYTE;
216.          "NO AND CL OF EACH CHARACTER"
217.          MNEM : ARRAY(0..255) OF ARRAY(0..3) OF CHAR;
218.          INITNAM : ARRAY (-1..30) OF ALFA;
219.          "CONTAINS THE PREDEFINED IDENTIFIERS,
220.             USED FOR INITIALIZING THE CONTEXT TABLE"
221.          BASEREGNO, BASE : SHRTINT ;
222.          BASEREG : ARRAY(1..MAXBASEREG) OF SHRTINT;
223.          STK : ARRAY(1..MAXISTK) OF SHRTINT;
224.          RSTK : ARRAY(1..MAXRSTK) OF SHRTINT;
225.          ASTK : ARRAY(0..MAXISTK) OF INTEGER;
226.          MASKARRAY : ARRAY(1..7) OF SHRTINT;
227.          UNIQINDEX : SHRTINT;
228.          UNIQCH : ARRAY(0..54) OF CHAR;
229.          HEXCHAR: ARRAY(0..15) OF CHAR;
230.          SYSPRINT : FILE OF PRINTLINE;
231.          SYSIN : FILE OF CARDIMAGE;
232.          PASCALGO(OUT) : FILE OF CARDIMAGE;
233.          CSTTB : ARRAY (1..CSTMAX) OF
234.             RECORD VALU : CONSTANT; INX,CNEXT : SHRTINT  END;
235.          FLCX,HLCX,ALCX,RLCX,ELCX : SHRTINT;
236.          "CONTAINS ALL CONSTANTS C OCCURRING IN THE
237.           PROCEDURE ACTUALLY BEING COMPILED TOGETHER WITH AN INDEX INTO
238.           UNDLAB WHERE THEIR OCCURRENCES IN THE CODE OF THIS PROCEDURE
239.           ARE CHAINED"
```

```
240.
241.          CODE : ARRAY (0..CODMAX) OF BYTE;
242.
243.          CONTEXTTABLE : CLASS 550 OF
244.          "PACKED" RECORD
245.          NAME : ALFA;  NXTEL : CTP; ALIGN : SHRTINT;
246.             CASE KLASS : IDCLASS OF
247.             TYPES : (SIZE : INTEGER;
248.                CASE FORM : TYPFORM OF
249.                NUMERIC : (BITS : BITRANGE; MIN,MAX : INTEGER);
250.                SYMBOLIC : (FCONST,PWSET : CTP; BITSIZE : BITRANGE);
251.                POINTER : (DOMAIN,ELTYPE : CTP);
252.                POWER : (ELSET : CTP; PWBITS : BITRANGE);
253.                ARRAYS : (AELTYPE,INXTYPE : CTP;  LO,HI : INTEGER;
254.                   OPTTYP : OPTPWR; EXP1,EXP2 : BITRANGE);
255.                RECORDS : (FSTFLD,RECVAR : CTP);
256.                CLASSS : (PELTYPE : CTP);
257.                FILES : (FELTYPE : CTP)   );
258.             KONST : (CONTYPE : CTP;
259.                CASE CONKIND : IDKINDS OF
260.                ACTUAL : (SUCC : CTP; VALUES : CONSTANT);
261.                FORMAL : (CADDR : ADDRESS; CLEVEL : RG3)   );
262.             PROC : (PROCTYPE,FORMALS,SURRPROC : CTP;
263.                PROCKIND : IDKINDS;
264.                PROCADDR : ADDRESS; PROCLEVEL : RG3;
265.                SEGSIZE : SHRTINT; SDNAME : ALFA;
266.                PREDEF : BOOLEAN);
267.             VARS : (VTYPE : CTP; VKIND : IDKINDS;
268.                VADDR : ADDRESS; VLEVEL : RG3; SSIZE: INTEGER "ADDRESS");
269.             FIELD : (FLDTYPE : CTP; FLDADDR : ADDRESS;
270.                BITDISPL,BITWIDTH : BITRANGE);
271.             TAGFIELD : (CASESIZE : SHRTINT; VARIANTS : CTP;
272.                CASE TAGVAL : BOOLEAN OF
273.                FALSE : (CASETYPE : CTP);
274.                TRUE : (CASEVAL : SHRTINT)   );
275.          END;
276.
277.       " VARIABLE INITIALIZATIONS  "
278.          VALUE
279.             SPLITSTAT = (1,2,19*1,3,1,4,1,1,5,1,6,1,7,1,8,2*1,9,12*1,10) ;
280.             ERRCL = (16*IRRELSY,ENDSY,4*IRRELSY,BEGSY,ENDSY,BEGSY,IRRELSY,
281.                      ENDSY,BEGSY,IRRELSY,BEGSY,ENDSY,BEGSY,IRRELSY,BEGSY,
282.                      2*IRRELSY,BEGSY,IRRELSY,ENDSY,2*IRRELSY,2*ENDSY,
283.                      IRRELSY,3*ENDSY,2*IRRELSY,BEGSY);
284.             TERRCL = (9*IRRELSY,BEGSY,ENDSY,IRRELSY,ENDSY,3*IRRELSY,
285.                ENDSY,IRRELSY,BEGSY,2*IRRELSY,3*ENDSY,2*IRRELSY,ENDSY,
286.                IRRELSY,ENDSY,IRRELSY,ENDSY,IRRELSY,ENDSY,2*IRRELSY,
287.                ENDSY,IRRELSY,ENDSY,BEGSY,IRRELSY,2*ENDSY,IRRELSY,
288.                3*ENDSY,2*IRRELSY,ENDSY);
289.             BLANK = '  ' ;  BLANKALFA = '  ';  "MAY NOT WORK"
290.             BLANKLINE = (121*' ');
291.             BLANKCARD = (80*' ');
292.             WD = ('IF','DO','TO','OF','IN',
293.                'END','NIL','FOR','DIV','MOD','VAR','SET',
294.                'THEN','ELSE','GOTO','CASE','WITH','TYPE','FILE',
295.                'BEGIN','UNTIL','WHILE','ARRAY','VALUE','CLASS',
296.                   'CONST','LABEL',
297.                'REPEAT','DOWNTO','RECORD','PACKED',
298.                'FUNCTION','POWERSET','PROCEDURE');
299.             WNO = (23,31,33,27,8,
```

```
            22,36,32,6,6,43,11,
            24,25,35,26,48,37,38,
            21,29,30,38,47,38,41,40,
            28,33,38,42,
            44,38,45);
        WCL  = (0,0,1,0,7,
            0,0,0,4,5,0,0,
            6*0,3,
            0,0,0,1,0,4,0,0,
            0,2,2,0,
            0,5,0);
        WL = (0,0,0,5,12,19,27,31,31,33,34,34);
        CHARTYPE=(75*0,7,5,4*11,11*0,3*11,4,2*11,9*0,11,2*0,6,11*0,
                  8,3,11,10,11,9,65*0,9*1,7*0,9*1,8*0,8*1,6*0,10*2,6*0);
        SYMNO=(75*0,17,8,9,7,7,6,11*0,6,10,16,5,7,6,9*0,15,2*0,8,11*0,
               19,2,18,0,8,0,65*0,9*1,7*0,9*1,8*0,8*1,6*0,10*2,6*0);
        SYMCL=(75*0,0,1,0,1,3,3,11*0,1,0,0,1,2,2,9*0,0,2*0,4,11*0,
               0,1,0,0,6,0,65*0,9*0,7*0,9*0,8*0,8*0,6*0,10*1,6*0);
    MNEM =  (4*'    ','SPM ','BALR','BCTR',
      'BCR ','SSK ','ISK ','SVC ',2*'    ','BASR',2*'    ','LPR ','LNR ',
      'LTR ','LCR ','NR  ','CLR ','OR  ','XR  ',
      'LR  ','CR  ','AR  ','SR  ','MR  ',
      'DR  ','ALR ','SLR ','LPDR','LNDR','LTDR','LCDR','HDR ','LRDR',
      'MXR ','MXDR','LDR ','CDR ','ADR ','SDR ','MDR ','DDR ','AWR ','SWR '
      ,'LPER','LNER','LTER','LCER','HER ','LRER','LAXR','SXR ','LER ',
    'CER ','AER ','SER ','MER ','DER ','AUR ','SUR ','STH ','LA  ','STC ',
      'IC  ','EX  ','BAL ','BCT ',
      'BC  ','LH  ','CH  ','AH  ','SH  ','MH  ','BAS ',
      'CVD ','CVB ','ST  ',3*'    ','N   ','CL  ',
      'O   ','X   ','L   ','C   ','A   ','S   ','M   ',
      'D   ','AL  ','SL  ','STD ',6*'    ','MXD ',
      'LD  ','CD  ','AD  ','SD  ','MD  ',
      'DD  ','AW  ','SW  ','STE ',7*'    ',
      'LE  ','CE  ','AE  ','SE  ','ME  ','DE  ',
      'AU  ','SU  ','SSM ','    ','LPSW',
      '    ','WRD ','RDD ','BXH ','BXLE',
      'SRL ','SLL ','SRA ','SLA ','SRDL','SLDL','SRDA','SLDA','STM ',
      'TM  ','MVI ','TS  ','NI  ','CLI ','OI  ',
      'XI  ','LM  ',3*'    ','SIO ','TIO '
      ,'HIO ','TCH ',16*'    ','STMC','LRA ',5*'    ','LMC ',25*'    '
      ,'MVN ','MVC ','MVZ ','NC  ','CLC ',
      'OC  ','XC  ',4*'    ','TR  ','TRT ',
      'ED  ','EDMK',17*'    ','MVO ','PACK','UNPK',4*'    ','ZAP ','CP  ',
      'AP  ','SP  ','MP  ','DP  ',2*'    ');
        INITNAM = ('TEXT','WEOR','GET','PUT','RESET','ALLOC',
            'PACK','UNPACK','INSERT','APPEND',
            'READ','WRITE',
            'ODD','INT','CHR','EOF','ABS','SQR',
            'TRUNC','PRED','SUCC','EOL','ALFALENG',
            'INPUT','OUTPUT','ALFA','REAL','CHAR','BOOLEAN',
            'FALSE','TRUE','INTEGER');
        BASEREG = (3,4);
        STK = (11,10,9,8,7,6,5);
        RSTK = (0,2,4,6);
        MASKARRAY = (4,12,10,2,6,8,0);
           UNIQCH = ('.','<','(','+','&','*',')',';',
             '¬','/',',','%','_','>','?',':','#','@','=',
             '0','1','2','3','4','5','6','7','8','9',
             'A','B','C','D','E','F','G','H','I','J','K','L','M',
             'N','O','P','Q','R','S','T','U','V','W','X','Y','Z');
```

```
360.          HEXCHAR = ('0','1','2','3','4','5','6','7','8','9','A',
361.             'B','C','D','E','F');
362.
363.        "TWO STANDARD PROCEDURES  INSERT  AND  APPEND  ARE PROVIDED FOR
364.           CODE GENERATION.
365.         APPEND(A,B,C) SHIFTS THE CONTENTS OF A LEFT B BITS AND
366.           'OR'S C INTO IT. RESULT IN A. B,C UNCHANGED.
367.         INSERT(A,B,C) SHIFTS THE CONTENTS OF A LEFT B BITS AND
368.           'OR'S THEM INTO C. RESULT IN C. A,B UNCHANGED
369.         A,B,C ARE OF TYPE INTEGER  "
370.
371.        PROCEDURE OUTCH(C:CHAR) ; FORWARD ;
372.        PROCEDURE OUTALF(A : ALFA; WIDTH : SHRTINT)  ; FORWARD ;
373.        PROCEDURE OUTI(VAL : INTEGER ; LGTH : SHRTINT); FORWARD;
374.        PROCEDURE OUTHEX(FVAL : INTEGER ; WIDTH : INTEGER); FORWARD;
375.        PROCEDURE PRTERR ; FORWARD ;
376.        PROCEDURE NEXTCH ; FORWARD ;
377.        PROCEDURE ERROR(I:SHRTINT) ; FORWARD ;
378.        PROCEDURE ERRMESSAGE(ALF : ALFA ; VAL : SHRTINT) ; FORWARD ;
379.        PROCEDURE INSYMBOL; FORWARD;
380.        PROCEDURE SRCHREC(P:CTP); FORWARD;
381.        PROCEDURE SEARCH; FORWARD;
382.        PROCEDURE INCONST (VAR V : CONSTKIND;  P : CTP;
383.            CONST NXT : CTP) ; FORWARD;
384.        PROCEDURE GENJP(FJPADDR : ADDRESS); FORWARD;
385.        PROCEDURE GENRR(OP,R1,R2 : SHRTINT); FORWARD;
386.        PROCEDURE GENRS(OP,R1,R3,XD2,XB2 : SHRTINT); FORWARD;
387.        PROCEDURE GENRX(OP,R1,XD2,XX2,XB2 : SHRTINT); FORWARD;
388.        PROCEDURE GENSI(OP,XD1,XB1,XI2 : SHRTINT); FORWARD;
389.        PROCEDURE GENSS(OP,XD1,XL,XB1,XD2,XB2 : SHRTINT); FORWARD;
390.        PROCEDURE CHECKBNDS(FRP,FMIN,FMAX,FADR : INTEGER); FORWARD;
391.        PROCEDURE INS(FADR,FCA : ADDRESS); FORWARD;
392.        PROCEDURE MULOPT(VAL1:INTEGER;
393.           VAR EXP1,EXP2:INTEGER;  OPT:OPTPWR); FORWARD;
394.        PROCEDURE SKIP(FNO : SHRTINT) ; FORWARD;
395.        PROCEDURE PRINTC(INC : BOOLEAN); FORWARD;
396.        PROCEDURE PRTCOMP; FORWARD;
397.        PROCEDURE PRINTCT; FORWARD;
398.        FUNCTION LOG2(VAL1:INTEGER) : SHRTINT; FORWARD;
399.        PROCEDURE STKERR; FORWARD;
400.        PROCEDURE GENCONST(FVAL:CONSTANT; FCA:ADDRESS); FORWARD;
401.        PROCEDURE LDCST2(FVAL:CONSTANT; FRP:SHRTINT); FORWARD;
402.        PROCEDURE LDCST(FVAL:CONSTANT); FORWARD;
403.        PROCEDURE GENADDR(FATTR:ATTR; I:SHRTINT); FORWARD;
404.        PROCEDURE GENSADDR(FATTR:ATTR); FORWARD;
405.        PROCEDURE LOADBASE(FRP, L : SHRTINT) ; FORWARD;
406.        PROCEDURE LOAD(VAR FATTR:ATTR); FORWARD;
407.        PROCEDURE STORE(FATTR:ATTR); FORWARD;
408.        PROCEDURE LOADADR(VAR FATTR:ATTR); FORWARD;
409.        PROCEDURE ADDRESSVAR(FCTP : CTP ; VAR FATTR : ATTR) ; FORWARD;
410.        PROCEDURE UPALIGN (VAR DISPL:INTEGER; CONST ALIGN:INTEGER);  FORWARD;
411.        PROCEDURE VARIABLE; FORWARD;
412.        PROCEDURE FACTOR; FORWARD;
413.        PROCEDURE TERM; FORWARD;
414.        PROCEDURE SIMPLEEXP; FORWARD;
415.        PROCEDURE EXPRESSION; FORWARD;
416.        PROCEDURE PASSPARAMS; FORWARD;
417.        PROCEDURE ASSIGN; FORWARD;
418.        PROCEDURE VARIAB; FORWARD;
419.        PROCEDURE ALLC; FORWARD;
```

```
420.        PROCEDURE FIXEDTOFLOAT; FORWARD;
421.        PROCEDURE FLOATTOFIXED; FORWARD;
422.        PROCEDURE GETPUT(LPSW:SHRTINT); FORWARD;
423.        PROCEDURE INSAPP(LPSW:SHRTINT); FORWARD;
424.        PROCEDURE PCK; FORWARD;
425.        PROCEDURE READIR; FORWARD;
426.        PROCEDURE TITLE; FORWARD;
427.        PROCEDURE UNPCK; FORWARD;
428.        PROCEDURE WRITEIR; FORWARD;
429.        PROCEDURE WRITOUT(CNAME:ALFA); FORWARD;
430.        PROCEDURE IFSTAT; FORWARD;
431.        PROCEDURE CASESTAT;FORWARD;
432.        PROCEDURE REPEATSTAT;FORWARD;
433.        PROCEDURE WHILESTAT;FORWARD;
434.        PROCEDURE FORSTAT;FORWARD;
435.        PROCEDURE GOTOSTAT;FORWARD;
436.        PROCEDURE WITHSTAT;FORWARD;
437.        PROCEDURE COMPSTAT;FORWARD;
438.        PROCEDURE STATEMENT;FORWARD;
439.        PROCEDURE SETSDNAME; FORWARD;
440.        PROCEDURE TYPEDECL(VAR TL:INTEGER; P1:CTP); FORWARD;
441.        PROCEDURE BODY(SURRPTR,FIRSTENTRY:CTP); FORWARD;
442.        PROCEDURE OUTCH;
443.           BEGIN
444.              IF C=EOL THEN
445.                 BEGIN PUT(SYSPRINT);
446.                   SYSPRINT@:=BLANKLINE; PTOUT:=0 END
447.              ELSE BEGIN PTOUT := PTOUT+1;
448.                 SYSPRINT@(PTOUT) := C
449.              END
450.           END "OUTCH";
451.
452.        PROCEDURE OUTALF;
453.           VAR AA : CWORD ; I : SHRTINT ;
454.        BEGIN UNPACK(A,AA,1) ;
455.           FOR I := 1 TO WIDTH DO
456.              SYSPRINT@(PTOUT+I) := AA(I) ;
457.           PTOUT := PTOUT+WIDTH;
458.        END " OUTALF " ;
459.
460.        PROCEDURE OUTI;
461.         " PRINT VAL DECIMAL IN LGTH PLACES. LEADING ZEROES ARE SUPPRESSED "
462.           VAR S,S1 : INTEGER;
463.              I,J : SHRTINT;
464.              DIGIT : ARRAY (0..17) OF SHRTINT;
465.        BEGIN I := 0;  J := LGTH - 1;  S := ABS(VAL);
466.           REPEAT S1 := S DIV 10;  DIGIT(I) := S - S1*10 ;
467.              S := S1;  I := I + 1;
468.           UNTIL S = 0;
469.           WHILE J > I DO
470.           BEGIN J := J - 1;  OUTCH(' ')  END;
471.           IF VAL < 0 THEN OUTCH('-') ELSE OUTCH(' ');
472.           REPEAT I := I - 1;
473.              OUTCH(HEXCHAR(DIGIT(I)));
474.           UNTIL I = 0;
475.        END  " OUTI " ;
476.
477.        PROCEDURE OUTHEX;
478.         " PRINT VAL IN WIDTH HEXADECIMAL PLACES, PRECEDED BY A SPACE"
479.           VAR D : ARRAY (1..8) OF BYTE;
```

```
480.              I,VAL : INTEGER;
481.          BEGIN OUTCH(' ');     VAL :=FVAL;
482.             FOR I := WIDTH DOWNTO 1 DO
483.                BEGIN
484.                   D(I) := VAL MOD 16;
485.                   VAL := VAL DIV 16;
486.                END;
487.             FOR I := 1 TO WIDTH DO OUTCH(HEXCHAR(D(I)));
488.          END "OUTHEX";
489.
490.          PROCEDURE PRTERR;
491.             VAR I : SHRTINT;
492.          BEGIN  SYSPRINT@:=BLANKLINE; PTOUT:=0; OUTCH(' ');
493.             FOR I := 1 TO 4 DO OUTCH('*');
494.             FOR I:=1 TO ERRINX DO
495.                SYSPRINT@(ERRLIST(I).POS+7) := '|';
496.             PTOUT := 80;
497.             FOR I := 1 TO ERRINX DO OUTI(ERRLIST(I).NMR,4);
498.             OUTCH(EOL); ERRINX := 0; POS1 := 0;
499.          END "PRTERR";
500.
501.          PROCEDURE NEXTCH;
502.             "READS NEXT CHAR OF FILE INPUT.
503.                CONVERTS EOL TO BLANK,
504.                PRINTS ERROR SUMMARY AFTER EACH LINE (IF ERRORS PRESENT)
505.                AND PRINTS ADDRESS AT BEGINNING OF LINE "
506.          BEGIN
507.             IF EOLFLAG THEN
508.             BEGIN IF ERRINX > 0 THEN "PRINT ERRORS" PRTERR;
509.                EOLFLAG := FALSE;  CHCNT := 0;
510.                IF LISTING THEN
511.                BEGIN
512.                  IF DP THEN OUTHEX(LC,6) ELSE OUTHEX(IC,6);
513.                  OUTCH(' ');
514.                END;
515.                GET(SYSIN);  CHPTRX:=72;
516.                WHILE (CHPTRX>0) & (SYSIN@(CHPTRX)=' ') DO
517.                   CHPTRX:=CHPTRX-1;
518.                IF CHPTRX<=0 THEN CHPTRX:=0;
519.                SYSIN@(CHPTRX+1):=EOL;
520.                CHPTRX:=1;
521.             END "EOLFLAG";
522.             CH:=SYSIN@(CHPTRX);  CHPTRX:=CHPTRX+1;
523.             IF LISTING THEN OUTCH(CH); CHCNT := CHCNT + 1;
524.             IF CH = EOL THEN
525.             BEGIN CH := ' ';  EOLFLAG := TRUE  END;
526.          END "NEXTCH";
527.
528.          PROCEDURE ERROR;
529.          BEGIN "MEM(41B) := INT(TRUE);   ERRFLAG"
530.             ERR := TRUE;
531.            "ERRNRS(I DIV 30) := ERRNRS(I DIV 30)!(I MOD 30) ; "
532.             IF ERRINX >= 9 THEN
533.                WITH ERRLIST(10) DO
534.                BEGIN POS := 0 ; NMR := 104 ; ERRINX := 10 ;
535.                  "ERRNRS(NMR DIV 30) := ERRNRS(NMR DIV 30)!(NMR MOD 30) ;"
536.                END ELSE
537.             BEGIN IF CHCNT > POS1 THEN POS1 := CHCNT ;
538.                ERRINX := ERRINX + 1;
539.                WITH ERRLIST(ERRINX) DO
```

```
540.              BEGIN POS := POS1;  NMR := I END;
541.              POS1 := POS1 + 1;
542.           END;
543.        END "ERROR";
544.
545.        PROCEDURE ERRMESSAGE;
546.           VAR IT3 : SHRTINT ;
547.        BEGIN ERROR(43) ; OUTCH(EOL) ;
548.           FOR IT3 := 1 TO 9 DO OUTALF(BLANK,10) ;
549.           OUTALF(ALF,10) ; OUTI(VAL,7) ; OUTCH(EOL) ;
550.           IF ¬EOLFLAG THEN
551.           FOR IT3 := 1 TO CHCNT+8 DO OUTCH(' ') ;
552.        END "ERRMESSAGE";
553.
554.     "  I N S Y M B O L                READS ONE PASCAL-SYMBOL  "
555.        " LIST OF SYMBOL NUMBERS
556.        NO   CL   SYMBOL              NO   CL   SYMBOL
557.
558.         1        ID                  20        :=
559.         2    1   INTEGER CONST       21        BEGIN
560.         2    2   REAL    CONST       22        END
561.         2    3   ALFA    CONST       23        IF
562.         2    4   CHAR    CONST       24        THEN
563.                                      25        ELSE
564.                                      26        CASE
565.         5    1   ¬                   27        OF
566.         6    1   *                   28        REPEAT
567.         6    2   /                   29        UNTIL
568.         6    3   &                   30        WHILE
569.         6    4   DIV                 31        DO
570.         6    5   MOD                 32        FOR
571.         7    1   +                   33   1    TO
572.         7    2   -                   33   2    DOWNTO
573.         7    3   |                   35        GOTO
574.         8    1   <                   36        NIL
575.         8    2   <=                  37        TYPE
576.         8    3   >=                  38   1    ARRAY
577.         8    4   >                   38   2    RECORD
578.         8    5   ¬=                  38   3    FILE
579.         8    6   =                   38   4    CLASS
580.         8    7   IN                  38   5    POWERSET
581.         9        (                   40        LABEL
582.        10        )                   41        CONST
583.        11        SET                 42        PACKED
584.                                      43        VAR
585.        15        ,                   44        FUNCTION
586.        16        ;                   45        PROCEDURE
587.        17        .                   47        VALUE
588.        18        @                   48        WITH
589.        19        :
590.
591.           NOT PASCAL SYMBOLS : BLANK  '  $  "
592.
593.        PROCEDURE INSYMBOL;
594.           VAR  SCALE,EXP : SHRTINT;  R,FAC : REAL;
595.              I,J,K : SHRTINT;  BT1,SIGN,SIZEOK : BOOLEAN;
596.              CHTYPE : BYTE;
597.
598.           PROCEDURE OPTION;
599.              VAR CH1 : CHAR;
```

```
600.          BEGIN
601.             REPEAT NEXTCH; CH1 := CH; NEXTCH;
602.                IF CH1 = 'A' THEN ASSCHECK := CH = '+' ELSE
603.                IF CH1 = 'C' THEN PRCODE := CH = '+' ELSE
604.                IF CH1 = 'D' THEN DIVCHECK := CH = '+' ELSE
605.                IF CH1 = 'L' THEN LISTING := CH = '+' ELSE
606.                IF CH1 = 'O' THEN STOFLCHECK := CH ='+' ELSE
607.                IF CH1 = 'P' THEN PCODE := CH = '+' ELSE
608.                IF CH1 = 'X' THEN INXCHECK := CH = '+' ;
609.                NEXTCH;
610.             UNTIL CH¬=','
611.          END " OPTION ";
612.
613.       BEGIN
614.       1: WHILE CHARTYPE(CH) = 0 DO NEXTCH; "BLANKS AND ILLEGAL CHARACTERS"
615.          NO := SYMNO(CH);
616.          CL := SYMCL(CH);
617.          CASE CHARTYPE(CH) OF
618.            1:  "A LETTER - MUST BE THE START OF AN IDENTIFIER OR
619.                     RESERVED WORD"
620.                BEGIN   UNPACK(BLANKALFA,A,1);   I := 0;
621.                   REPEAT IF I < ALFALENG THEN
622.                     BEGIN I := I+1;   A(I) := CH;   END;
623.                     NEXTCH;  CHTYPE := CHARTYPE(CH)
624.                   UNTIL ¬((CHTYPE=1)|(CHTYPE=2));
625.                   PACK(A,1,AVAL);
626.                   FOR J := WL(I) TO WL(I+1)-1 DO
627.                      IF AVAL = WD(J) THEN
628.                         BEGIN NO := WNO(J);   CL := WCL(J);    END;
629.                END;
630.
631.            2:  "A DIGIT - MUST BE A NUMBER CONSTANT"
632.                BEGIN   SIZEOK := TRUE;   IVAL := 0;
633.                   REPEAT
634.                      BEGIN IF IVAL>MAX10 THEN SIZEOK := FALSE;
635.                         IF SIZEOK THEN
636.                           IVAL := IVAL*10 + (INT(CH) - INT('0'));
637.                         NEXTCH
638.                      END;
639.                   UNTIL CHARTYPE(CH) ¬= 2;
640.                   IF ¬SIZEOK THEN BEGIN IVAL := 0; ERROR(2) END;
641.                   SCALE := 0;
642.                   IF CH = '.' THEN
643.                   BEGIN NEXTCH; IF CH = '.' THEN CH := ':' ELSE
644.                      BEGIN RVAL := IVAL; CL := 2;
645.                         IF CHARTYPE(CH)¬=2 THEN ERROR(3) ELSE
646.                         REPEAT R := INT(CH) - INT('0');
647.                            RVAL := TEN*RVAL + R;
648.                            SCALE := SCALE - 1; NEXTCH
649.                         UNTIL CHARTYPE(CH)¬=2;
650.                      END;
651.                   END;
652.                   IF CH = 'E' THEN
653.                   BEGIN IF SCALE = 0 THEN
654.                      BEGIN RVAL := IVAL; CL := 2  END;
655.                      SIGN := FALSE; NEXTCH;
656.                      IF CH = '+' THEN NEXTCH ELSE
657.                      IF CH = '-' THEN
658.                         BEGIN NEXTCH; SIGN := TRUE  END;
659.                      EXP := 0;
```

```
660.                  WHILE CHARTYPE(CH)=2 DO
661.                  BEGIN EXP := 10*EXP + (INT(CH) - INT('0'));
662.                     NEXTCH
663.                  END;
664.                  IF SIGN THEN EXP := -EXP;
665.                  SCALE := SCALE + EXP
666.               END;
667.               IF SCALE ¬= 0 THEN
668.               BEGIN R := ONE;
669.                  IF SCALE < 0 THEN
670.                  BEGIN FAC := TENTH; SCALE := -SCALE END
671.                  ELSE FAC := TEN;
672.                  REPEAT IF ODD(SCALE) THEN R := R*FAC;
673.                     FAC := SQR(FAC); SCALE := SCALE DIV 2
674.                  UNTIL SCALE = 0; "R = 10 ** SCALE"
675.                  RVAL := RVAL*R
676.               END;
677.            END;
678.
679.         3: "  #  - MUST BE A HEXADECIMAL CONSTANT"
680.            BEGIN IVAL := 0;   NEXTCH;   CHTYPE := CHARTYPE(CH);
681.               WHILE (CHTYPE=2)|(CHTYPE=1) DO
682.                  BEGIN CASE CHTYPE OF
683.                     1:   J := INT(CH) - INT('A') + 10;
684.                     2:   J := INT(CH) - INT('0');    END;
685.                     IVAL := IVAL*16+J;
686.                     NEXTCH;  CHTYPE:=CHARTYPE(CH);
687.                  END;
688.            END;
689.
690.         4: "   ¬  - CHECK FOR ¬=    "
691.            BEGIN NEXTCH;
692.               IF CH = '=' THEN BEGIN NO := 8; CL := 5;
693.                                      NEXTCH   END;
694.            END;
695.
696.         5: "   <  - CHECK FOR <=    "
697.            BEGIN NEXTCH;
698.               IF CH = '=' THEN BEGIN CL := 2;  NEXTCH  END;
699.            END;
700.
701.         6: "   >  - CHECK FOR >=    "
702.            BEGIN NEXTCH;
703.               IF CH = '=' THEN BEGIN CL := 3;  NEXTCH  END;
704.            END;
705.
706.         7: "   .  - CHECK FOR ..    "
707.            BEGIN NEXTCH;
708.               IF CH = '.' THEN BEGIN NO := 19; NEXTCH  END;
709.            END;
710.
711.         8: "   :  - CHECK FOR :=    "
712.            BEGIN NEXTCH;
713.               IF CH = '=' THEN BEGIN NO := 20; NEXTCH  END;
714.            END;
715.
716.         9: "  ""  - CHECK INSIDE COMMENT FOR OPTION SWITCHES  "
717.            BEGIN NEXTCH;
718.               IF CH = '$' THEN OPTION;
719.               WHILE CH ¬= '"' DO NEXTCH;
```

```
720.                  NEXTCH; GOTO 1;
721.               END;
722.
723.           10: "  '   - AN ALFA OR CHAR CONSTANT  "
724.               BEGIN  UNPACK(BLANKALFA,A,1); NO := 2; K := 0; BT1 := FALSE;
725.                  REPEAT NEXTCH;
726.                     IF CH = '''' THEN
727.                     BEGIN NEXTCH; BT1 := CH ¬= '''' END;
728.                     IF ¬BT1 THEN
729.                        IF K = ALFALENG THEN
730.                        BEGIN ERROR(94); BT1 := TRUE END ELSE
731.                        BEGIN K := K + 1; A(K) := CH END;
732.                  UNTIL BT1;
733.                  IF K =1 THEN  " CHAR CONST "
734.                  BEGIN CL:=4; CHVAL := A(1) END
735.                  ELSE  " ALFA CONST "
736.                  BEGIN PACK (A,1,AVAL);  CL := 3  END;
737.               END;
738.
739.           11: "OTHER SYMBOLS - DO NOTHING ELSE - NO,CL ALREADY SET"
740.               NEXTCH;
741.
742.        END;
743.     END  "INSYMBOL" ;
744.
745.     PROCEDURE SRCHREC;
746.         " SEARCHES ONE BLOCK, RETURNS CTPTR "
747.     BEGIN CTPTR := P;
748.        WHILE CTPTR ¬= NIL DO
749.           IF CTPTR@.NAME = AVAL THEN GOTO 1
750.           ELSE CTPTR := CTPTR@.NXTEL;
751.   1:END  " SRCHREC " ;
752.
753.     PROCEDURE SEARCH;
754.        " SEARCHES CONTEXTTABLE, RETURNS CTPTR AND DISX = INDEX TO
755.           DISPLAY "
756.        VAR I : SHRTINT;
757.     BEGIN FOR I := TOP DOWNTO 0 DO
758.        BEGIN CTPTR := DISPLAY(I).FNAME;
759.           WHILE CTPTR ¬= NIL DO
760.              IF CTPTR@.NAME = AVAL THEN GOTO 1
761.              ELSE CTPTR := CTPTR@.NXTEL;
762.        END;
763.   1:   DISX := I;
764.     END  " SEARCH " ;
765.
766.     PROCEDURE INCONST;
767.        " INPUT PARAMETER : NXT. SEARCHING IS TO BEGIN AT NXT
768.          OUTPUT PARAMETER : V = KIND OF CONSTANT
769.                             P = TYPE POINTER , P = NIL IF ERROR "
770.        VAR SIGN : BOOLEAN;  PT : CTP;
771.     BEGIN SIGN := FALSE;  P := NIL;
772.        IF NO = 7 THEN
773.        BEGIN SIGN := CL = 2;
774.           IF CL <= 2 THEN INSYMBOL;
775.        END;
776.        IF NO = 2 THEN
777.        BEGIN CASE CL OF
778.   1:      BEGIN  P := INTPTR;  V := INTEGERS;  END;
779.   2:      BEGIN  P := REALPTR;  V := REALS;    END;
```

```
780.       3:       BEGIN  P := ALFAPTR;  V := ALFAS;     END;
781.       4:       BEGIN  P := CHARPTR;  V := CHARS; IVAL:=INT(CHVAL);     END;
782.              END;
783.           END ELSE
784.           IF NO = 1 THEN
785.           BEGIN PT := CTPTR;
786.              SRCHREC(NXT);
787.              IF CTPTR = NIL THEN SEARCH;
788.              IF CTPTR = NIL THEN ERROR(12) ELSE
789.              WITH CTPTR@ DO
790.              BEGIN IF (KLASS ¬= KONST)|(CONKIND = FORMAL) THEN ERROR(63)
791.                 ELSE
792.                 WITH VALUES DO
793.                 BEGIN P := CONTYPE;  V := KONSTKIND;
794.                    CASE V OF
795.              INTEGERS, CHARS, SYMBOLICS:  IVAL := IVALUE;
796.              REALS:  RVAL := RVALUE;
797.              ALFAS:  AVAL := AVALUE;
798.                    END "CASE V" ;
799.                 END;
800.                 CTPTR := PT;  "ORIGINAL PASCAL DIFFERS HERE"
801.              END;
802.           END ELSE ERROR(3);
803.           IF SIGN THEN
804.           BEGIN  IVAL := -IVAL;  RVAL := -RVAL  END;
805.        END " INCONST " ;
806.
807.        PROCEDURE GENJP;
808.        BEGIN
809.        "  IF GATTR.KIND ¬= LCOND THEN " .
810.           IF GATTR.TYPTR¬=BOOLPTR THEN ERROR(57)
811.           ELSE  BEGIN   LOAD(GATTR);
812.              GENRR(#12,STK(RP),STK(RP)); "LTR"
813.              GENRX(#47,8,0,0,0); "BC FALSE "
814.              IF FJPADDR¬=0 THEN INS(FJPADDR,IC-2)
815.           END
816.         "  ELSE BEGIN ""STUFF FOR LCOND GOES HERE"" END "
817.        END "GENJP";
818.
819.        PROCEDURE GENRR;
820.        BEGIN
821.           IF IC>=CODMAX-2 THEN BEGIN ERROR(90); IC:=0 END;
822.           CODE(IC):=OP;  CODE(IC+1):=R1*16+R2;
823.           IF PRCODE THEN PRINTC(FALSE);
824.           IC:=IC+2
825.        END "GENRR";
826.
827.        PROCEDURE GENRS;
828.        BEGIN
829.           IF IC>=CODMAX-4 THEN BEGIN ERROR(90); IC:=0 END;
830.           CODE(IC):=OP;  CODE(IC+1):=R1*16+R3;
831.           CODE(IC+2):=XB2*16+XD2 DIV 256;  CODE(IC+3):=XD2 MOD 256;
832.           IF PRCODE THEN PRINTC(FALSE);
833.           IC:=IC+4
834.        END "GENRS";
835.
836.        PROCEDURE GENRX;
837.        BEGIN
838.           IF IC>=CODMAX-4 THEN BEGIN ERROR(90); IC:=0 END;
839.           CODE(IC):=OP;  CODE(IC+1):=R1*16+XX2;
```

```
840.          CODE(IC+2):=XB2*16+XD2 DIV 256;
841.          CODE(IC+3):=XD2 MOD 256;
842.          IF PRCODE THEN PRINTC(FALSE);
843.          IC:=IC+4
844.       END "GENRX";
845.
846.       PROCEDURE GENSI;
847.       BEGIN
848.          IF IC>=CODMAX-4 THEN BEGIN ERROR(90); IC:=0 END;
849.          CODE(IC):=OP;  CODE(IC+1):=XI2;
850.          CODE(IC+2):=XB1*16+XD1 DIV 256;
851.          CODE(IC+3):=XD1 MOD 256;
852.          IF PRCODE THEN PRINTC(FALSE);
853.          IC:=IC+4
854.       END "GENSI";
855.
856.       PROCEDURE GENSS;
857.       BEGIN
858.          IF IC>=CODMAX-6 THEN BEGIN ERROR(90); IC:=0 END;
859.          CODE(IC):=OP;  CODE(IC+1):=XL;  CODE(IC+2):=XB1*16+XD1 DIV 256;
860.          CODE(IC+3):=XD1 MOD 256;  CODE(IC+4):=XB2*16+XD2 DIV 256;
861.          CODE(IC+5):=XD2 MOD 256;
862.          IF PRCODE THEN PRINTC(FALSE);
863.          IC:=IC+6
864.       END "GENSS";
865.
866.       PROCEDURE CHECKBNDS;
867.       BEGIN
868.       END "CHECKBNDS";
869.
870.       PROCEDURE INS;
871.       VAR LADR : ADDRESS;  B : SHRTINT;
872.       BEGIN
873.          LADR := FADR;
874.          IF LADR >= 4095 THEN BEGIN B:=BASEREG(2); LADR := LADR-4096 END
875.          ELSE B := BASEREG(1);
876.          CODE(FCA):=B*16+LADR DIV 256;
877.          CODE(FCA+1):=LADR MOD 256
878.       END "INS";
879.
880.       PROCEDURE MULOPT;
881.         "MULOPT DECIDES IF VAL = 2@EXP1              : OPT = PUREP
882.                           = 2@EXP1*(2@EXP2+1) : OPT = POSP
883.                           = 2@EXP1*(2@EXP2-1) : OPT = NEGP
884.                           OTHER                      : OPT = NOOPT "
885.          VAR E1,E2 : SHRTINT;
886.              VAL : SHRTINT;
887.       BEGIN IF VAL1 > 0 THEN
888.          BEGIN E1 := 0;
889.              VAL := VAL1 ;
890.              WHILE ¬ODD(VAL) DO
891.              BEGIN VAL := VAL DIV 2;  E1 := E1 + 1  END;
892.              IF VAL = 1 THEN
893.              BEGIN OPT := PUREP;  EXP1 := E1  END ELSE
894.              BEGIN VAL := VAL DIV 2;  E2 := 1;
895.                 IF ODD(VAL) THEN
896.                 BEGIN
897.                    REPEAT VAL := VAL DIV 2;  E2 := E2 + 1;
898.                    UNTIL ¬ODD(VAL);
899.                    IF VAL > 0 THEN OPT := NOOPT ELSE
```

```
900.                  BEGIN OPT := NEGP;
901.                     EXP2 := E2;  EXP1 := E1;
902.                  END;
903.               END ELSE
904.               BEGIN
905.                  REPEAT VAL := VAL DIV 2;  E2 := E2 + 1;
906.                  UNTIL ODD(VAL);
907.                  IF VAL > 1 THEN OPT := NOOPT  ELSE
908.                  BEGIN OPT := POSP;
909.                     EXP2 := E2;  EXP1 := E1;
910.                  END;
911.               END;
912.            END;
913.         END ELSE OPT := NOOPT;
914.      END "MULOPT";
915.
916.      FUNCTION LOG2;
917.         VAR E : SHRTINT;  VAL : INTEGER;
918.      BEGIN E := 1;
919.         VAL := VAL1;
920.         WHILE VAL > 0 DO
921.         BEGIN VAL := VAL DIV 2;  E := E + 1  END;
922.         LOG2 := E;
923.      END "LOG2";
924.
925.
926.      PROCEDURE SKIP;
927.      BEGIN
928.         WHILE (ERRCL(NO) = IRRELSY)&(FNO ¬= NO) DO
929.         IF (NO = 38)&(CL = 2) THEN  "RECORD"
930.         BEGIN REPEAT INSYMBOL; SKIP(49);
931.           "UNTIL ¬(NO IN SET(16,26)); "
932.            UNTIL ¬((NO=16)|(NO=26));
933.            IF NO = 22 THEN INSYMBOL ;
934.         END ELSE INSYMBOL;
935.      END  " SKIP "  ;
936.
937.      PROCEDURE PRINTC;
938.      VAR A:ALFA; LINE:PRINTLINE; LPTOUT: SHRTINT;
939.      BEGIN
940.         LINE:=SYSPRINT@; LPTOUT:=PTOUT; SYSPRINT@:=BLANKLINE; PTOUT:=0;
941.         OUTHEX(IC,6);  OUTALF(BLANKALFA,2);  OUTI(IC,4);
942.         OUTALF(BLANKALFA,4);
943.         IF CODE(IC)>=256 THEN
944.         BEGIN OUTALF('     ERROR',10); OUTALF(BLANKALFA,5);
945.            OUTI(CODE(IC),10); OUTI(CODE(IC+1),10);
946.            IF INC THEN IC:=IC+2
947.         END
948.         ELSE BEGIN
949.           PACK(MNEM(CODE(IC)),0,A);  OUTALF(A,4);
950.           OUTALF(BLANKALFA,4);
951.           IF CODE(IC)<=63 THEN
952.           BEGIN    "RR-FORMAT"
953.              OUTI(CODE(IC+1) DIV 16,2); OUTCH(',');
954.              OUTI(CODE(IC+1) MOD 16,2);
955.              IF INC THEN IC:=IC+2;
956.           END
957.           ELSE IF CODE(IC)<=127 THEN
958.           BEGIN  "RX-FORMAT"
959.              OUTI(CODE(IC+1) DIV 16,2); OUTCH(',');
```

```
960.              OUTI(CODE(IC+3)+256*(CODE(IC+2) MOD 16),4);
961.              OUTCH('('); OUTI(CODE(IC+1) MOD 16,2);
962.              OUTCH(','); OUTI(CODE(IC+2) DIV 16,2);
963.              OUTCH(')');
964.              IF INC THEN IC:=IC+4
965.           END
966.           ELSE IF ((134<=CODE(IC))&(CODE(IC)<=144))|(CODE(IC)=152) THEN
967.           BEGIN  "RS-FORMAT"
968.              OUTI(CODE(IC+1) DIV 16,2); OUTCH(',');
969.              OUTI(CODE(IC+1) MOD 16,2); OUTCH(',');
970.              OUTI(CODE(IC+3)+256*(CODE(IC+2) MOD 16),4);
971.              OUTCH('('); OUTI(CODE(IC+2) DIV 16,2); OUTCH(')');
972.              IF INC THEN IC:=IC+4
973.           END
974.           ELSE IF CODE(IC)>191 THEN
975.           BEGIN   "SS-FORMAT"
976.              OUTI(CODE(IC+3)+256*(CODE(IC+2) MOD 16),4);
977.              OUTCH('('); OUTI(CODE(IC+1),3); OUTCH(',');
978.              OUTI(CODE(IC+2) DIV 16,2); OUTCH(')'); OUTCH(',');
979.              OUTI(CODE(IC+5)+256*(CODE(IC+4) MOD 16),4); OUTCH('(');
980.              OUTI(CODE(IC+4) DIV 16,2); OUTCH(')');
981.              IF INC THEN IC:=IC+6
982.           END
983.           ELSE
984.           BEGIN   "SI-FORMAT"
985.              OUTI(CODE(IC+3)+256*(CODE(IC+2) MOD 16),4); OUTCH('(');
986.              OUTI(CODE(IC+2) DIV 16,2); OUTCH(')'); OUTCH(',');
987.              OUTI(CODE(IC+1),3);
988.              IF INC THEN IC:=IC+4
989.            END
990.        END;
991.        PUT(SYSPRINT); SYSPRINT@:=LINE; PTOUT:=LPTOUT;
992.     END "PRINTC";
993.
994.     PROCEDURE PRTCOMP;
995.     VAR ICSAVE : ADDRESS;
996.     BEGIN
997.        OUTCH(EOL);
998.        ICSAVE:=IC;
999.        IC:=0;
1000.       PRINTC(TRUE);
1001.       IC:=IC+16;  "SKIP AROUND CONSTANTS"
1002.       WHILE IC<ICSAVE DO PRINTC(TRUE);
1003.    END "PRTCOMP";
1004.
1005.    PROCEDURE PRINTCT;
1006.    BEGIN
1007.       OUTCH(EOL);  OUTALF('PRINTCT NO',10);
1008.       OUTALF('T IMPLEMEN',10); OUTALF('TED',3);
1009.       OUTCH(EOL);
1010.    END "PRTCOMP";
1011.
1012.    PROCEDURE STKERR;
1013.    BEGIN
1014.       IF ISTKLIM<MAXISTK THEN
1015.       BEGIN  ISTKLIM:=ISTKLIM+1;  RP:=RP+1;  ASTK(RP):=0  END
1016.       ELSE ERROR(33);
1017.    END "STKERR";
1018.
1019.    PROCEDURE GENCONST;
```

```
1020.       VAR X:REAL; AA,E:ALFA; I:INTEGER; LAST,NT:SHRTINT;
1021.       BEGIN
1022.          WITH FVAL DO
1023.          CASE KONSTKIND OF
1024.          REALS:
1025.            BEGIN X:=RVALUE; NT:=RLCX;
1026.              WHILE NT¬=0 DO WITH CSTTB(NT) DO
1027.                IF X=VALU.RVALUE THEN GOTO 1
1028.                ELSE NT:=CNEXT;
1029.              LAST:=RLCX;
1030.              RLCX:=CHNIX
1031.            END;
1032.          INTEGERS:
1033.            BEGIN I:=IVALUE; IF ABS(I)<=32767 THEN NT:=HLCX ELSE NT:=FLCX;
1034.              LAST:=NT;
1035.              WHILE NT¬=0 DO WITH CSTTB(NT) DO
1036.                IF I=VALU.IVALUE THEN GOTO 1
1037.                ELSE NT:=CNEXT;
1038.              IF ABS(I)<=32767 THEN HLCX:=CHNIX ELSE FLCX:=CHNIX
1039.            END;
1040.          ALFAS:
1041.            BEGIN AA:=AVALUE; NT:=ALCX;
1042.              WHILE NT¬=0 DO WITH CSTTB(NT) DO
1043.                IF AA=VALU.AVALUE THEN GOTO 1
1044.                ELSE NT:=CNEXT;
1045.              LAST:=ALCX;
1046.              ALCX:=CHNIX
1047.            END;
1048.          EXTREF:
1049.            BEGIN E:=EVALUE; NT:=ELCX;
1050.              WHILE NT¬=0 DO WITH CSTTB(NT) DO
1051.                IF E=VALU.EVALUE THEN GOTO 1
1052.                ELSE NT:=CNEXT;
1053.              LAST:=ELCX;
1054.              ELCX:=CHNIX
1055.            END;
1056.          END; "CASE KONSTKIND"
1057.          WITH CSTTB(CHNIX) DO
1058.          BEGIN CNEXT:=LAST; VALU:=FVAL; INX:=0 END;
1059.          NT:=CHNIX;
1060.          IF CHNIX<CSTMAX THEN CHNIX:=CHNIX+1 ELSE ERROR(84);
1061.       1: WITH CSTTB(NT) DO
1062.          BEGIN CODE(FCA):=INX DIV 256; CODE(FCA+1):=INX MOD 256;
1063.            INX:=FCA END
1064.       END "GENCONST";
1065.
1066.       PROCEDURE LDCST2;
1067.       VAR OP:SHRTINT;
1068.       BEGIN
1069.          OP:=0;
1070.          WITH FVAL DO
1071.          CASE KONSTKIND OF
1072.          INTEGERS:
1073.            IF (IVALUE>=0)&(IVALUE<=4095) THEN
1074.              GENRX(#41,FRP,IVALUE,0,0) "LA"
1075.            ELSE IF ABS(IVALUE)<=32767 THEN OP:=#48 "LH"
1076.            ELSE OP:=#58; "L"
1077.          REALS: OP:=#68;
1078.          EXTREF: OP:=#58;
1079.          ALFAS: ERROR(85);
```

```
1080.          END; "CASE KONSTKIND"
1081.          IF OP¬=0 THEN
1082.            BEGIN GENRX(OP,ERP,0,0,0);
1083.                  GENCONST(FVAL,IC-2) END
1084.        END "LDCST2";
1085.
1086.        PROCEDURE LDCST;
1087.        VAR RREG:SHRTINT;
1088.        BEGIN
1089.          RREG:=0;
1090.          WITH FVAL DO
1091.          CASE KONSTKIND OF
1092.            INTEGERS, EXTREF:
1093.              BEGIN IF RP<ISTKLIM THEN RP:=RP+1 ELSE STKERR;
1094.                    RREG:=STK(RP)  END;
1095.            REALS:
1096.              BEGIN IF RRP<MAXRSTK THEN RRP:=RRP+1 ELSE ERROR(33);
1097.                    RREG:=RSTK(RRP)  END;
1098.            ALFAS: ERROR(84);
1099.          END; "CASE KONSTKIND"
1100.          LDCST2(FVAL,RREG)
1101.        END "LDCST";
1102.
1103.        PROCEDURE GENADDR;
1104.        VAR XCONS:CONSTANT; RPD,LX:SHRTINT;  D:ADDRESS;
1105.        BEGIN
1106.          WITH FATTR,RXADDR DO
1107.          BEGIN
1108.            B2:=0;  X2:=0;  D:=DPLMT;
1109.            IF ACCESS¬=DRCT THEN LX:=STK(RP-1); RPD:=1; RP:=RP+1;
1110.            IF TYPTR¬=NIL THEN
1111.            IF PCKD THEN ERROR(87)
1112.            ELSE BEGIN
1113.              IF ACCESS=INDRCT THEN BEGIN RPD:=RPD+1; B2:=LX END
1114.              ELSE IF BREG=0 THEN B2:=GLOBALREG
1115.              ELSE IF BREG=LEVEL THEN B2:=LOCALREG
1116.              ELSE BEGIN
1117.                IF RP<ISTKLIM THEN RP:=RP+1  ELSE STKERR;
1118.                RPD:=RPD+1;  B2:=STK(RP);  LOADBASE(B2,BREG)
1119.              END;
1120.              IF ACCESS=INXD THEN BEGIN RPD:=RPD+1; X2:=LX END;
1121.              IF (D<0)|(D>4095) THEN
1122.                WITH XCONS DO
1123.                BEGIN KONSTKIND:=INTEGERS;
1124.                  IF D<0 THEN BEGIN IVALUE:=D; D:=0 END
1125.                  ELSE BEGIN IVALUE:=D-D MOD 4096; D:=D MOD 4096 END;
1126.                  LDCST(XCONS);  RPD:=RPD+1;
1127.                  IF X2=0 THEN X2:=STK(RP) ELSE GENRR(#1A,X2,STK(RP)) "AR"
1128.                END;
1129.              D2:=D;
1130.              RP:=RP-RPD
1131.            END   "TYPTR¬=NIL & ¬PCKD"
1132.          END  "WITH FATTR,RXADDR"
1133.        END "GENADDR";
1134.
1135.        PROCEDURE GENSADDR;
1136.        BEGIN
1137.          WITH FATTR,RXADDR DO
1138.          BEGIN
1139.            IF TYPTR@.SIZE>256 THEN
```

```
1140.            BEGIN LOADADR(FATTR);
1141.              SSADDR.D2:=0;  SSADDR.B2:=STK(RP) END
1142.            ELSE BEGIN GENADDR(FATTR,0);
1143.              IF X2¬=0 THEN
1144.                BEGIN "B2 WILL NEVER(?) BE 0"
1145.                  IF RP<ISTKLIM THEN RP:=RP+1 ELSE STKERR;
1146.                  GENRX(#41,STK(RP),D2,X2,B2);
1147.                  WITH SSADDR DO BEGIN D2:=0; B2:=STK(RP) END
1148.                END
1149.              ELSE BEGIN SSADDR.D2:=D2; SSADDR.B2:=B2 END
1150.            END
1151.        END
1152.      END "GENSADDR";
1153.
1154.      PROCEDURE LOADBASE;
1155.        VAR I : SHRTINT ;
1156.      BEGIN GENRX(#58,FRP,0,0,13); "L"
1157.        FOR I := 1 TO LEVEL - L - 1 DO GENRX(#58,FRP,0,0,FRP);
1158.      END "LOADBASE" ;
1159.
1160.      PROCEDURE LOAD;
1161.        "THIS PROCEDURE GENERATES CODE FOR LOADING A QUANTITY DESCRIBED
1162.         BY FATTR INTO A REGISTER."
1163.
1164.      VAR LOADTYP,OP,RREG:SHRTINT;
1165.          LCONST:CONSTANT;
1166.
1167.        BEGIN WITH FATTR,RXADDR DO
1168.          IF KIND=VARBL THEN
1169.           BEGIN "***KIND=VARBL***"
1170.          LOADTYP:=ALIGNMENT;
1171.        "DETERMINE PROPER OP CODE"
1172.          IF LOADTYP=1 THEN OP:=#43  "IC"
1173.          ELSE IF LOADTYP=2 THEN OP:=#48  "LH"
1174.          ELSE IF LOADTYP=4 THEN OP:=#58  "L"
1175.          ELSE IF LOADTYP=8 THEN OP:=#68  "LD"
1176.          ELSE ERROR(120);
1177.          GENADDR(FATTR,0);
1178.
1179.        "DETERMINE RESULT REGISTER, RREG"
1180.          IF LOADTYP=8 THEN
1181.            BEGIN IF RRP<MAXRSTK THEN "NO REAL STACK OVERFLOW"
1182.              BEGIN RRP:=RRP+1;  RREG:=RSTK(RRP) END ELSE ERROR(33) END
1183.          ELSE BEGIN
1184.            IF RP<ISTKLIM THEN RP := RP+1  ELSE STKERR;
1185.            RREG := STK(RP);
1186.          END;
1187.
1188.        "GENERATE CODE.  SPECIAL CASE IF <CHAR>.
1189.           ALFA(ILLEGAL) ALSO WILL FALL HERE"
1190.          IF LOADTYP=1 THEN "CHAR"
1191.            BEGIN  IF (ACCESS=DRCT)&((BREG=0)|(BREG=LEVEL)) THEN
1192.               BEGIN GENRR(#1B,RREG,RREG);  "SR - CLEAR JUNK IN RREG"
1193.                     GENRX(#43,RREG,D2,0,B2) "IC"   END
1194.               ELSE BEGIN GENRR(#1B,0,0);  "SR - CLEAR R0"
1195.                      GENRX(#43,0,D2,X2,B2);  "IC - LOAD INTO R0"
1196.                      GENRR(#18,RREG,0)  "LR-MOVE TO RREG" END END "CHAR"
1197.            ELSE "REAL OR INT"  GENRX(OP,RREG,D2,X2,B2);  "ACTJAL LOAD"
1198.           END  "***KIND=VARBL***"
1199.
```

```
1200.              ELSE IF KIND=SVAL THEN
1201.              BEGIN "***KIND=SVAL***"
1202.                IF RP<ISTKLIM THEN RP:=RP+1 ELSE STKERR;
1203.                IF VAL=0 THEN GENRR(#1B,STK(RP),STK(RP))  "SR"
1204.                  ELSE GENRX(#41,STK(RP),VAL,0,0);
1205.              END "***KIND=SVAL***"
1206.
1207.              ELSE IF KIND=LVAL THEN
1208.              BEGIN "***KIND=LVAL***"
1209.                RREG:=STK(RP);
1210.                IF CTERM=-1 THEN GENRR(#6,STK(RP),0) "BCTR"
1211.                ELSE IF CTERM¬=0 THEN
1212.                WITH LCONST DO
1213.                BEGIN
1214.                  KONSTKIND:=INTEGERS; IVALUE:=CTERM;
1215.                  LOCST(LCONST); RP:=RP-1;
1216.                  GENRR(#1A,STK(RP),STK(RP+1));
1217.                  CTERM:=0
1218.                END
1219.              END "***KIND=LVAL***"
1220.
1221.              ELSE IF KIND=LCOND THEN
1222.              BEGIN "***KIND=LCOND***"
1223.                "NOT YET IMPLEMENTED"
1224.              END "***KIND=LCOND***"
1225.
1226.              ELSE ERROR(507)
1227.
1228.          END "LOAD";
1229.
1230.        PROCEDURE STORE;
1231.          "THIS PROCEDURE GENERATES CODE FOR STORING THE QUANTITY DESCRIBED
1232.           BY FATTR INTO A REGISTER."
1233.
1234.        VAR LOADTYP,OP,RREG,I:SHRTINT;
1235.
1236.          BEGIN WITH FATTR,RXADDR DO
1237.            IF KIND=VARBL THEN
1238.             BEGIN "***KIND=VARBL***"
1239.            LOADTYP:=ALIGNMENT;
1240.
1241.          "DETERMINE PROPER OP CODE"
1242.            IF LOADTYP=1 THEN OP:=#42  "STC"
1243.            ELSE IF LOADTYP=2 THEN OP:=#40  "STH"
1244.            ELSE IF LOADTYP=4 THEN OP:=#50  "ST"
1245.            ELSE IF LOADTYP=8 THEN OP:=#60  "STD"
1246.            ELSE ERROR(120);
1247.
1248.          "DETERMINE REGISTER TO BE STORED, RREG"
1249.            IF LOADTYP=8 THEN
1250.              BEGIN I:=0; RREG:=RSTK(RRP); RRP:=RRP-1 END
1251.            ELSE BEGIN RREG:=STK(RP); I:=1 END;
1252.
1253.          "GENERATE CODE"
1254.            GENADDR(FATTR,I);
1255.            GENRX(OP,RREG,D2,X2,B2);
1256.            RP:=RP-I
1257.
1258.             END  "***KIND=VARBL***"
1259.
```

```
1260.              ELSE ERROR(507)
1261.
1262.          END "STORE";
1263.
1264.         PROCEDURE LOADADR;
1265.           "THIS PROCEDURE GENERATES CODE FOR LOADING THE ADDRESS OF THE
1266.            VARIABLE DESCRIBED BY FATTR INTO THE TOP OF THE STACK."
1267.
1268.         VAR RREG:SHRTINT;
1269.
1270.            BEGIN WITH FATTR,RXADDR DO
1271.              IF TYPTR¬=NIL THEN
1272.              IF PCKD THEN ERROR(87) ELSE
1273.                BEGIN
1274.
1275.            "DETERMINE RESULT REGISTER, RREG"
1276.              GENADDR(FATTR,0);
1277.              IF RP<ISTKLIM THEN RP := RP+1  ELSE STKERR;
1278.              RREG := STK(RP);
1279.
1280.            "GENERATE CODE"
1281.              GENRX(#41,RREG,D2,X2,B2);     "LA"
1282.              ACCESS:=INDRCT;  DPLMT:=0
1283.
1284.                END "NOT PACKED"
1285.
1286.          END "LOADADR";
1287.
1288.         PROCEDURE ADDRESSVAR;
1289.         "GENERATES CODE TO ADDRESS THE QUANTITY WITH CTPTR = FCTP AND
1290.          BUILDS UP ITS ATTRIBUTES IN FATTR"
1291.         VAR TATTR:ATTR;
1292.         BEGIN WITH FCTP@, FATTR DO
1293.           BEGIN KIND := VARBL ;
1294.             IF KLASS=VARS THEN
1295.             BEGIN
1296.               IF VTYPE@.FORM>=CLASSS THEN ALIGNMENT:=4
1297.               ELSE ALIGNMENT:=ALIGN
1298.             END
1299.             ELSE ALIGNMENT:=ALIGN;
1300.
1301.             IF KLASS = VARS THEN
1302.             BEGIN TYPTR := VTYPE ; PCKD := FALSE ;
1303.               IF VKIND = ACTUAL THEN
1304.               BEGIN ACCESS := DRCT ; DPLMT := VADDR ; BREG := VLEVEL ;
1305.               END ELSE
1306.               BEGIN IF RP < ISTKLIM THEN RP := RP + 1 ELSE STKERR;
1307.                 IF VLEVEL = LEVEL
1308.                    THEN GENRX(#58,STK(RP),VADDR,0,LOCALREG) "L"
1309.                 ELSE BEGIN LOADBASE(STK(RP),VLEVEL) ;
1310.                    GENRX(#58,STK(RP),VADDR,0,STK(RP));  "L"
1311.                 END;
1312.                 ACCESS := INDRCT ; DPLMT := 0 ;  ALIGNMENT:=TYPTR@.ALIGN;
1313.               END ;
1314.             END ELSE
1315.             IF KLASS = FIELD THEN
1316.             BEGIN TYPTR := FLDTYPE ;
1317.               WITH DISPLAY(DISX) DO
1318.               BEGIN IF OCCUR = CWITH THEN
1319.                 BEGIN ACCESS := DRCT; BREG := CLEV; DPLMT := FLDADDR + CDSPL
```

```
1320.                 END ELSE
1321.                 BEGIN
1322.                   WITH TATTR DO
1323.                   BEGIN ALIGNMENT:=4; KIND:=VARBL; TYPTR:=INTPTR;
1324.                     BREG:=LEVEL; DPLMT:=VDSPL; ACCESS:=DRCT;
1325.                     PCKD:=FALSE; LOAD(TATTR)
1326.                   END;
1327.                   ACCESS:=INDRCT; DPLMT:=FLDADDR; ALIGNMENT:=TYPTR@.ALIGN;
1328.                 END;
1329.                 IF BITWIDTH ¬= 0 THEN
1330.                 BEGIN PCKD := TRUE ; BITADR := BITDISPL ; BITSZ := BITWIDTH
1331.                 END ELSE PCKD := FALSE
1332.               END
1333.             END ELSE
1334.             BEGIN TYPTR := PROCTYPE; ACCESS := DRCT; BREG := PROCLEVEL + 1;
1335.               DPLMT := 72 ; PCKD := FALSE ;
1336.               ALIGNMENT:=4;
1337.               IF PROCKIND = FORMAL THEN ERROR(81) ELSE
1338.               IF LEVEL ¬= BREG THEN ERROR(85)
1339.               ELSE BEGIN LOAD(FATTR); ACCESS:=INDRCT; DPLMT:=0;
1340.                     ALIGNMENT:=TYPTR@.ALIGN END
1341.             END
1342.           END "WITH FCTP@, FATTR" ;
1343.         END "ADDRESSVAR" ;
1344.
1345.         PROCEDURE UPALIGN;
1346.           " INCREASES DISPL UNTIL IT IS A MULTIPLE OF ALIGN "
1347.           VAR I : INTEGER;
1348.         BEGIN
1349.           IF ALIGN > 0 THEN
1350.           BEGIN  I := DISPL MOD ALIGN;
1351.             IF I > 0 THEN DISPL := DISPL + ALIGN - I;
1352.           END;
1353.         END "UPALIGN";
1354.
1355.         PROCEDURE VARIABLE ;
1356.           VAR LATTR : ATTR ;    XCONS : CONSTANT;
1357.             XSIZE:INTEGER;
1358.         BEGIN ADDRESSVAR(CTPTR,LATTR) ;
1359.           INSYMBOL ;
1360.          "WHILE NO IN SET(9,17,18) DO""(,.,@"
1361.           WHILE (NO=9)|(NO=17)|(NO=18) DO "(,.,@"
1362.           IF NO = 9 THEN "("
1363.           BEGIN
1364.             REPEAT WITH LATTR DO
1365.               IF TYPTR ¬= NIL THEN
1366.               IF TYPTR@.FORM ¬= ARRAYS THEN
1367.               BEGIN ERROR(34) ; TYPTR := NIL END ;
1368.               INSYMBOL ; EXPRESSION ;
1369.               IF GATTR.TYPTR ¬= NIL THEN
1370.               BEGIN IF GATTR.TYPTR@.FORM > SYMBOLIC THEN ERROR(35) ;
1371.                 IF LATTR.TYPTR ¬= NIL THEN
1372.                 BEGIN PT := LATTR.TYPTR@.INXTYPE ;
1373.                   IF ((GATTR.TYPTR@.FORM = SYMBOLIC)|(PT@.FORM = SYMBOLIC))&
1374.                   (GATTR.TYPTR ¬= PT) THEN ERROR(36) ;
1375.                   WITH GATTR DO
1376.                   IF KIND = SVAL THEN
1377.                   BEGIN IF (LATTR.TYPTR@.LO > VAL)|(LATTR.TYPTR@.HI < VAL)
1378.                     THEN ERROR(99) ;
1379.                     IT := VAL
```

```
1380.                    END ELSE
1381.                    IF KIND = LVAL THEN IT := CTERM ELSE
1382.                    BEGIN LOAD(GATTR) ; IT := 0 END ;
1383.                    WITH LATTR, TYPTR@ DO
1384.                    BEGIN XSIZE:=AELTYPE@.SIZE;
1385.                      UPALIGN(XSIZE,AELTYPE@.ALIGN);
1386.                      DPLMT := DPLMT + (IT - LO)*XSIZE ;
1387.                      IF GATTR.KIND ¬= SVAL THEN
1388.                      BEGIN IF INXCHECK THEN CHECKBNDS(STK(RP),LO-IT,HI-IT,0);
1389.                        IF XSIZE¬=1 THEN
1390.                        IF OPTTYP = NOOPT THEN
1391.                        WITH XCONS DO
1392.                        BEGIN
1393.                          KONSTKIND:=INTEGERS;
1394.                          IVALUE:=XSIZE;
1395.                          LDCST2(XCONS,1);
1396.                          GENRR(#1C,0,STK(RP)); "MR"
1397.                          GENRR(#18,STK(RP),1); "LR"
1398.                        END
1399.                        ELSE BEGIN
1400.                           IF EXP1 ¬= 0 THEN GENRS(#89,STK(RP),0,EXP1,0); "SLL"
1401.                          IF OPTTYP ¬= PUREP THEN
1402.                          BEGIN GENRR(#18,0,STK(RP)); "LR"
1403.                             GENRS(#89,STK(RP),0,EXP2,0); "SLL"
1404.                            IF OPTTYP = POSP THEN GENRR(#1A,STK(RP),0) "AR"
1405.                                             ELSE GENRR(#1B,STK(RP),0); "SR"
1406.                          END
1407.                        END ;
1408.                        IF ACCESS = DRCT THEN ACCESS := INXD ELSE
1409.                        BEGIN RP := RP - 1 ;
1410.                           GENRR(#1A,STK(RP),STK(RP+1)) "AR" END
1411.                      END
1412.                    END ;
1413.                  END "IF LATTR.TYPTR ¬= NIL"
1414.                END "IF GATTR.TYPTR ¬= NIL" ;
1415.                IF LATTR.TYPTR¬=NIL THEN LATTR.TYPTR:=LATTR.TYPTR@.AELTYPE ;
1416.              UNTIL NO ¬= 15 "," ;
1417.              IF NO=10 ")" THEN INSYMBOL ELSE ERROR(37)
1418.            END "IF NO = 9" ELSE
1419.            IF NO = 17 THEN "."
1420.            BEGIN INSYMBOL ;
1421.              IF NO = 1 THEN "ID"
1422.              BEGIN IF LATTR.TYPTR ¬= NIL THEN
1423.                BEGIN IF LATTR.TYPTR@.FORM = RECORDS THEN
1424.                  WITH LATTR DO
1425.                  BEGIN SRCHREC(TYPTR@.FSTFLD) ;
1426.                    IF CTPTR = NIL THEN
1427.                    BEGIN ERROR(39) ; CTPTR := UNDECPTR END ;
1428.                    WITH CTPTR@ DO
1429.                    BEGIN TYPTR := FLDTYPE ; DPLMT := DPLMT + FLDADDR ;
1430.                      ALIGNMENT := ALIGN;
1431.                      IF BITWIDTH ¬= 0 THEN
1432.                      BEGIN BITADR := BITDISPL ; BITSZ := BITWIDTH ;
1433.                        PCKD := TRUE
1434.                      END ELSE PCKD := FALSE
1435.                    END
1436.                  END ELSE
1437.                  BEGIN ERROR(38) ; LATTR.TYPTR := NIL END ;
1438.                END ;
1439.                INSYMBOL
```

```
1440.            END ELSE
1441.            BEGIN ERROR(41) ; LATTR.TYPTR := NIL END
1442.          END "IF NO = 17" ELSE "@"
1443.          WITH LATTR DO
1444.          BEGIN
1445.            IF TYPTR ¬= NIL THEN
1446.            BEGIN LOAD(LATTR) ; ACCESS := INDRCT ; DPLMT := 0 ;
1447.              IF TYPTR@.FORM = FILES THEN TYPTR := TYPTR@.FELTYPE ELSE
1448.              IF TYPTR@.FORM = POINTER THEN TYPTR:=TYPTR@.ELTYPE
1449.              ELSE TYPTR:=NIL
1450.            END ;
1451.            IF TYPTR¬=NIL THEN ALIGNMENT:=TYPTR@.ALIGN ELSE ERROR(40);
1452.            INSYMBOL
1453.          END "@" ;
1454.          GATTR := LATTR
1455.        END "VARIABLE" ;
1456.
1457.        PROCEDURE FACTOR ;
1458.          VAR LATTR,TATTR : ATTR ; LPTR : CTP ; LRP, LRRP : SHRTINT ;
1459.              AT : ADDRESS;  LPSVAL, IT2 : INTEGER;
1460.              XCONS : CONSTANT;  LTCT,TCTX,LB6,LB6A : ADDRESS;
1461.              OP,REG: SHRTINT;
1462.
1463.          PROCEDURE ELEMENT ;
1464.          "WORKS UP THE NEXT ELEMENT IN THE SET BEING ANALYSED"
1465.          BEGIN
1466.            EXPRESSION ;
1467.            IF GATTR.TYPTR ¬= NIL THEN
1468.            IF GATTR.TYPTR@.FORM > SYMBOLIC THEN ERROR(35) ELSE
1469.            IF (GATTR.TYPTR = REALPTR)|(GATTR.TYPTR = ALFAPTR) THEN
1470.            ERROR(62) ELSE
1471.            BEGIN IF LATTR.TYPTR ¬= NIL THEN
1472.              IF ((LATTR.TYPTR@.FORM ¬= NUMERIC)|(GATTR.TYPTR@.FORM ¬=
1473.              NUMERIC))&(LATTR.TYPTR ¬= GATTR.TYPTR) THEN ERROR(73) ;
1474.              LATTR.TYPTR := GATTR.TYPTR ;
1475.              IF GATTR.KIND = SVAL THEN INSERT(1,GATTR.VAL,LPSVAL) ELSE
1476.              BEGIN LOAD(GATTR) ; GENRX(#41,0,1,0,0); "LA"
1477.                GENRS(#89,0,0,0,STK(RP)); "SLL"
1478.                IF LATTR.KIND = SVAL THEN
1479.                BEGIN GENRR(#18,STK(RP),0); "LR"
1480.                  LATTR.KIND := LVAL ; LATTR.CTERM := 0
1481.                END ELSE
1482.                BEGIN
1483.                  RP := RP - 1 ; GENRR(#16,STK(RP),0); "OR"
1484.                END
1485.              END
1486.            END ;
1487.          END "ELEMENT" ;
1488.
1489.         BEGIN IF NO = 1 THEN                                    "IDENTIFIER"
1490.           BEGIN SEARCH ;
1491.             IF CTPTR = NIL THEN
1492.             BEGIN ERROR(31) ; CTPTR := UNDECPTR END ;
1493.             CASE CTPTR@.KLASS OF
1494.        TYPES: BEGIN ERROR(45) ; GATTR.TYPTR := NIL ; INSYMBOL END ;
1495.        KONST: WITH GATTR, CTPTR@ DO
1496.               BEGIN TYPTR := CONTYPE ;
1497.                 IF CONKIND = ACTUAL THEN
1498.                 WITH VALUES DO
1499.                   CASE KONSTKIND OF
```

```
                    INTEGERS, SYMBOLICS:
                      IF IVALUE>=TWOTO12 THEN
                      BEGIN LDCST(VALUES); KIND:=LVAL; CTERM:=0 END
                      ELSE BEGIN KIND:=SVAL; VAL:=IVALUE END;
                    REALS:
                      BEGIN LDCST(VALUES); KIND:=LVAL; CTERM:=0 END;
                    ALFAS:
                      BEGIN IF RP<ISTKLIM THEN RP:=RP+1 ELSE STKERR;
                        GENRX(#41,STK(RP),0,0,0); "LA"
                        GENCONST(VALUES,IC-2);  PCKD:=FALSE;
                        KIND:=VARBL; ACCESS:=INDRCT; DPLMT:=0
                      END;
                    CHARS:
                      BEGIN KIND:=SVAL; VAL:=IVALUE END;
                  END "CASE KONSTKIND, WITH VALUES"
                ELSE "FORMAL"
                BEGIN KIND := VARBL ; ACCESS := DRCT ;
                  BREG := CLEVEL ; DPLMT := CADDR ; PCKD := FALSE ;
                  ALIGNMENT:=ALIGN
                END ;
                INSYMBOL
              END ;
PROC : BEGIN INSYMBOL ;
         IF (CTPTR@.PROCTYPE = NIL)|(CTPTR@.PROCTYPE = CTPTR) THEN
         BEGIN ERROR(46) ; GATTR.TYPTR := NIL END ELSE
         IF NO ¬= 9 THEN "("
         BEGIN ERROR(79) ; GATTR.TYPTR := NIL END ELSE
         IF CTPTR@.PREDEF THEN                          "**INLINE FCT**"
         BEGIN LPTR := CTPTR ; INSYMBOL ; EXPRESSION ;
           IF GATTR.TYPTR ¬= NIL THEN
           CASE LPTR@.SEGSIZE OF
    "ODD" 1: BEGIN IF GATTR.TYPTR@.FORM ¬= NUMERIC THEN ERROR(47) ;
               LOAD(GATTR) ;
               GENRS(#89,STK(RP),0,31,0); "SLL"
               GENRS(#88,STK(RP),0,31,0); "SRL"
               WITH GATTR DO
               BEGIN TYPTR := BOOLPTR ; "KIND := LCOND ; JMP := 2 ;
                 ARITH := TRUE " KIND:=LVAL; CTERM:=0
               END
             END ;
    "INT" 2: WITH GATTR DO
             BEGIN IF TYPTR@.FORM>POWER THEN ERROR(44);
               LOAD(GATTR);  KIND:=LVAL; CTERM:=0; TYPTR:=INTPTR
             END;
    "CHR" 3: BEGIN IF GATTR.TYPTR@.FORM ¬= NUMERIC THEN ERROR(47) ;
               IF ASSCHECK THEN
               BEGIN LOAD(GATTR) ;
                 WITH GATTR DO
                 BEGIN KIND := LVAL ; CTERM := 0 END ;
                 CHECKBNDS(STK(RP),0,255,ASSERR)
               END ;
               GATTR.TYPTR := CHARPTR
             END ;
    "EOF" 4: BEGIN IF GATTR.TYPTR@.FORM ¬= FILES THEN ERROR(44) ;
               WITH GATTR DO
               BEGIN DPLMT := DPLMT + 1 ; LOAD(GATTR) ;
                 TYPTR := BOOLPTR ; KIND := LCOND ; JMP := 2 ;
                 ARITH := TRUE  "WHAT IS HAPPENING???"
               END
             END ;
```

```
1560.          "ABS" 5: BEGIN IF (GATTR.TYPTR ¬= REALPTR)&(GATTR.TYPTR@.FORM ¬=
1561.                     NUMERIC) THEN ERROR(44) ;
1562.                     LOAD(GATTR) ;
1563.                     IF GATTR.TYPTR@.ALIGN=8 THEN
1564.                     BEGIN OP := #20; "LPDR" REG := RSTK(RRP) END
1565.                     ELSE BEGIN OP := #10; "LPR" REG := STK(RP) END;
1566.                     GENRR(OP,REG,REG);
1567.                     WITH GATTR DO
1568.                     BEGIN IF TYPTR ¬= REALPTR THEN TYPTR := INTPTR ;
1569.                       KIND := LVAL ; CTERM := 0
1570.                     END
1571.                   END ;
1572.          "SQR" 6: BEGIN IF (GATTR.TYPTR ¬= REALPTR)&(GATTR.TYPTR@.FORM ¬=
1573.                     NUMERIC) THEN ERROR(44) ;
1574.                     LOAD(GATTR) ;
1575.                     WITH GATTR DO
1576.                     BEGIN IF TYPTR = REALPTR THEN
1577.                       GENRR(#2C,RSTK(RRP),RSTK(RRP)) "MDR"
1578.                       ELSE BEGIN GENRR(#18,1,STK(RP)); "LR"
1579.                                  GENRR(#1C,0,STK(RP)); "MR"
1580.                                  GENRR(#18,STK(RP),1); "LR"
1581.                                  TYPTR := INTPTR END;
1582.                       KIND := LVAL ; CTERM := 0 ;
1583.                     END
1584.                   END ;
1585.          "TRC" 7: BEGIN IF GATTR.TYPTR ¬= REALPTR THEN ERROR(44) ;
1586.                     LOAD(GATTR) ;
1587.                     FLOATTOFIXED;
1588.                     WITH GATTR DO
1589.                     BEGIN TYPTR := INTPTR ; KIND := LVAL ; CTERM := 0 END
1590.                   END ;
1591.          "PRE" 8,
1592.          "SUC" 9: IF (GATTR.TYPTR@.FORM > SYMBOLIC)|(GATTR.TYPTR =
1593.                   REALPTR)|(GATTR.TYPTR = ALFAPTR) THEN
1594.                   BEGIN ERROR(44) ; GATTR.TYPTR := NIL END ELSE
1595.                   BEGIN LOAD(GATTR) ;
1596.                     WITH GATTR.TYPTR@ DO
1597.                     IF FORM = NUMERIC THEN
1598.                     BEGIN IT1 := MIN ; IT2 := MAX END ELSE
1599.                     BEGIN IT1 := 0 ; IT2 := FCONST@.VALUES.IVALUE END ;
1600.                     IT := LPTR@.SEGSIZE ;
1601.                     IF IT=8 THEN GENRR(#06,STK(RP),0) "BCTR"
1602.                     ELSE BEGIN GENRX(#41,0,1,0,0); "LA"
1603.                                GENRR(#1A,STK(RP),0) "AR" END;
1604.                     IF ASSCHECK&(GATTR.TYPTR ¬= INTPTR) THEN
1605.                     CHECKBNDS(STK(RP),IT1,IT2,ASSERR) ;
1606.                     WITH GATTR DO
1607.                     BEGIN KIND := LVAL ; CTERM := 0 END
1608.                   END
1609.                 END "CASE LPTR@.SEGSIZE OF" ;
1610.                 IF NO ¬= 10 THEN ")"
1611.                 BEGIN ERROR(48) ; GATTR.TYPTR := NIL END ELSE INSYMBOL
1612.               END "IF CTPTR@.PREDEF" ELSE              "**EXTERNAL FCT**"
1613.               WITH TATTR,RXADDR DO
1614.               BEGIN  LTCT:=TCT;   LRRP:=RRP;
1615.                 IF RRP>0 THEN
1616.                 BEGIN UPALIGN(TCT,8);
1617.                   KIND:=VARBL; ACCESS:=DRCT; BREG:=LEVEL;
1618.                   PCKD:=FALSE; ALIGNMENT:=8;  TYPTR:=REALPTR;
1619.                   DPLMT:=TCT;
```

```
1620.                   FOR IT := 1 TO LRRP DO
1621.                     BEGIN
1622.                       STORE(TATTR);
1623.                       TCT := TCT + 8; IF TCT > TMAX THEN TMAX := TCT;
1624.                       DPLMT:=TCT;
1625.                     END;
1626.                   TCTX:=DPLMT;
1627.                 END;
1628.                 LB6:=B6DPL;
1629.                 IF LB6¬=72 THEN
1630.                 WITH XCONS DO
1631.                 BEGIN
1632.                   KONSTKIND:=INTEGERS;
1633.                   LB6A:=LB6; UPALIGN(LB6A,8);
1634.                   IVALUE:=LB6A; LDCST(XCONS);
1635.                   GENRR(#1A,2,STK(RP)) "AR"
1636.                 END;
1637.                 LRP := RP ; " LPTR := CTPTR@.PROCTYPE ; "
1638.                 RRP := 0 ; "RESTRICTS # OF NESTED FUNCTION CALLS"
1639.               "BECAUSE IT LEAVES LB6A IN STK(RP)"
1640.                 PASSPARAMS ;
1641.                 RRP:=0;  RP := LRP ;
1642.                 IF LB6¬=72 THEN
1643.                 BEGIN GENRR(#1B,2,STK(RP));  "SR"  RP:=RP-1 END;
1644.                 B6DPL:=LB6;
1645.                 IF LRRP ¬= 0 THEN
1646.                 BEGIN
1647.                   DPLMT:=TCTX;
1648.                   FOR IT := LRRP DOWNTO 1 DO
1649.                     BEGIN DPLMT:=DPLMT-8;
1650.                       LOAD(TATTR);
1651.                     END;
1652.                 END;
1653.                 TCT:=LTCT;
1654.              "  WITH GATTR DO
1655.                 BEGIN TYPTR := LPTR ; KIND := LVAL ; CTERM := 0 END "
1656.               END
1657.             END ;
1658.         VARS ,
1659.         FIELD : VARIABLE ;
1660.           END "CASE CTPTR@.KLASS OF" ;
1661.       END "IF NO = 1" ELSE
1662.       IF NO = 2 THEN                                          "CONSTANT"
1663.       BEGIN WITH GATTR,XCONS DO
1664.         CASE CL OF
1665.          1: BEGIN TYPTR := INTPTR ;
1666.               IF ABS(IVAL) >= TWOTO12 THEN
1667.                 BEGIN KONSTKIND := INTEGERS ;
1668.                   IVALUE := IVAL ;
1669.                   LDCST(XCONS) ;
1670.                   KIND := LVAL ; CTERM := 0 ; END
1671.                 ELSE BEGIN KIND := SVAL ; VAL := IVAL END;
1672.             END;
1673.          2: BEGIN TYPTR := REALPTR ;
1674.               KONSTKIND := REALS ;
1675.               RVALUE := RVAL ;
1676.               LDCST(XCONS) ;
1677.               KIND := LVAL ; CTERM := 0
1678.             END;
1679.          3: BEGIN TYPTR := ALFAPTR ;
```

```
1680.                IF RP<ISTKLIM THEN RP:=RP+1 ELSE STKERR;
1681.                GENRX(#41,STK(RP),0,0,0);  "LA"
1682.                KONSTKIND:=ALFAS;  AVALUE:=AVAL;
1683.                GENCONST(XCONS,IC-2);  PCKD:=FALSE;
1684.                KIND:=VARBL; ACCESS:=INDRCT; DPLMT:=0
1685.              END;
1686.           4: BEGIN TYPTR := CHARPTR ;
1687.                KIND := SVAL ;
1688.                VAL := INT(CHVAL) ;
1689.              END;
1690.         END ;
1691.         INSYMBOL ;
1692.       END "IF NO = 2" ELSE
1693.       IF NO = 36 THEN                                              "NIL"
1694.       BEGIN WITH GATTR DO
1695.         BEGIN TYPTR := NILPTR ; KIND := SVAL ; VAL := NILVAL.IVALUE END ;
1696.         INSYMBOL
1697.       END ELSE
1698.       IF NO = 5 THEN                                               " ¬ "
1699.       BEGIN INSYMBOL ; FACTOR ;
1700.         WITH GATTR DO
1701.         IF TYPTR ¬= NIL THEN
1702.         IF TYPTR = BOOLPTR THEN
1703.         BEGIN
1704.           LOAD(GATTR) ;
1705.           GENRX(#41,0,1,0,0) ; "LA"
1706.           GENRR(#17,STK(RP),0) ; "XR"
1707.           KIND := LVAL ; CTERM := 0 ;
1708.         END ELSE ERROR(50)
1709.       END "IF NO = 5" ELSE
1710.       IF NO = 9 THEN                                               " ( "
1711.       BEGIN INSYMBOL ; EXPRESSION ;
1712.         IF NO ¬= 10 THEN ")"
1713.         BEGIN ERROR(48) ; GATTR.TYPTR := NIL END ELSE INSYMBOL
1714.       END ELSE
1715.       IF NO = 11 THEN                                                " SET "
1716.       BEGIN INSYMBOL ;
1717.         IF NO = 9 THEN
1718.         BEGIN INSYMBOL ;
1719.           IF NO = 10 THEN ")"
1720.           WITH GATTR DO
1721.           BEGIN TYPTR := LAMPTR ; KIND := SVAL ;
1722.              VAL := 0 ; INSYMBOL
1723.           END ELSE
1724.           BEGIN WITH LATTR DO
1725.             BEGIN TYPTR := NIL ; KIND := SVAL END ; LPSVAL := 0 ;
1726.             ELEMENT ;
1727.             WHILE NO = 15 DO ","
1728.             BEGIN INSYMBOL ; ELEMENT END ;
1729.             IF LPSVAL ¬= 0 THEN
1730.             IF LPSVAL >= TWOTO12 THEN
1731.             BEGIN XCONS.KONSTKIND := INTEGERS ;
1732.               XCONS.IVALUE := LPSVAL ;
1733.               LDCST(XCONS) ;
1734.               IF LATTR.KIND = LVAL THEN
1735.               BEGIN RP:=RP-1 ; GENRR(#16,STK(RP),STK(RP+1)) "OR" END ELSE
1736.               LATTR.KIND := LVAL ; LATTR.CTERM := 0
1737.             END ELSE
1738.             IF LATTR.KIND = SVAL THEN LATTR.VAL := LPSVAL ELSE
1739.             BEGIN GENRX(#41,0,LPSVAL,0,0) ; "LA"
```

```
1740.                    GENRR(#16,STK(RP),0) "OR" END ;
1741.                IF LATTR.TYPTR ¬= NIL THEN
1742.                IF LATTR.TYPTR@.FORM = NUMERIC THEN LATTR.TYPTR := PNUMPTR
1743.                ELSE LATTR.TYPTR := LATTR.TYPTR@.PWSET ;
1744.                IF NO ¬= 10 THEN ")"
1745.                BEGIN ERROR(37) ; LATTR.TYPTR := NIL END ELSE INSYMBOL ;
1746.                GATTR := LATTR
1747.              END ;
1748.            END "IF NO = 9" ELSE
1749.            BEGIN ERROR(42) ; GATTR.TYPTR := NIL END
1750.          END "IF NO = 11" ELSE
1751.          BEGIN ERROR(42) ; GATTR.TYPTR := NIL END
1752.       END "FACTOR" ;
1753.
1754.    PROCEDURE TERM ;
1755.    VAR LATTR : ATTR; BT1,BT2,BT3,BT4 : BOOLEAN ;
1756.        LMOPCL,OPC : SHRTINT ;
1757.    PROCEDURE CHECKDIV(FRP : RG3 );
1758.       BEGIN IF GATTR.KIND = SVAL THEN
1759.          BEGIN IF GATTR.VAL = 0 THEN ERROR(102);
1760.          END ELSE
1761.          IF DIVCHECK THEN
1762.          BEGIN " JUMP ZERODIVIDE    " END;
1763.       END "DIVCHECK";
1764.
1765.    BEGIN FACTOR;
1766.       IF NO = 6 THEN "MULOP"
1767.       BEGIN
1768.          LOAD(GATTR);
1769.          LATTR.TYPTR:=GATTR.TYPTR; LATTR.KIND:=LVAL;
1770.          LATTR.CTERM:=0;
1771.       REPEAT LMOPCL:=CL; INSYMBOL; FACTOR;
1772.          IF (LATTR.TYPTR¬=NIL)&(GATTR.TYPTR¬=NIL) THEN
1773.          BEGIN IF LMOPCL = 3 THEN "&"
1774.          BEGIN LOAD(GATTR); RP:=RP-1;
1775.             IF (LATTR.TYPTR=GATTR.TYPTR)&((LATTR.TYPTR@.FORM=POWER)|
1776.             (LATTR.TYPTR=BOOLPTR)) THEN
1777.             GENRR(#14,STK(RP),STK(RP+1)) ELSE ERROR(50)
1778.          END ELSE
1779.          BEGIN
1780.             BT1:=LATTR.TYPTR@.FORM=NUMERIC;
1781.             BT2:=GATTR.TYPTR@.FORM=NUMERIC;
1782.             BT3:=LATTR.TYPTR=REALPTR;
1783.             BT4:=GATTR.TYPTR=REALPTR;
1784.             IF (BT1&BT4)|(BT2&BT3) THEN "MIXED EXPRESSIONS " ERROR(50)
1785.             ELSE
1786.             IF (BT3&BT4) THEN
1787.             BEGIN LOAD(GATTR);
1788.                IF LMOPCL=1 THEN OPC:=44 ELSE
1789.                IF LMOPCL=2 THEN OPC:=45 ELSE ERROR(50);
1790.                RRP:=RRP-1;
1791.                GENRR(OPC,RSTK(RRP),RSTK(RRP+1))
1792.             END
1793.             ELSE
1794.             BEGIN
1795.                IF (BT1&BT2) THEN
1796.                BEGIN LOAD(GATTR);
1797.                   IF LMOPCL = 1 THEN " * "
1798.                     BEGIN OPC:=28; GENRR(#18,1,STK(RP-1)) END
1799.                   ELSE " / DIV MOD "
```

```
1800.                    BEGIN OPC:=29;
1801.                      GENRR(#18,0,STK(RP-1));
1802.                      GENRS(#8E,0,0,32,0)
1803.                    END;
1804.                  GENRR(OPC,#0,STK(RP)); RP:=RP-1;
1805.                  CASE LMOPCL OF
1806.            "*" 1: BEGIN
1807.                     " CHECK FOR OVERFLOW "
1808.                     GENRR(#18,STK(RP),#1);
1809.                   END;
1810.       "/ DIV " 2,4: GENRR(#18,STK(RP),#1);
1811.          "MOD" 5: GENRR(#18,STK(RP),#0)
1812.                   END
1813.                END ELSE ERROR(50)
1814.             END
1815.          END
1816.       END
1817.       UNTIL NO¬=6;
1818.       GATTR:=LATTR;
1819.       END "IF NO = 6 "
1820.    END "TERM";
1821.
1822.    PROCEDURE SIMPLEEXP;
1823.    VAR LATTR : ATTR ; LADOPCL : SHRTINT ; LFG,BT1,BT2 : BOOLEAN;
1824.    BEGIN
1825.       LFG:=FALSE;
1826.       IF NO=7 THEN "ADDOP"
1827.       BEGIN
1828.          IF CL = 2 THEN LFG:=TRUE ELSE IF CL = 3 THEN ERROR(51);
1829.          INSYMBOL
1830.       END;
1831.       TERM;
1832.       IF LFG|(NO=7) THEN
1833.       BEGIN WITH LATTR DO
1834.          BEGIN
1835.             TYPTR:=GATTR.TYPTR; KIND:=LVAL; CTERM:=0;
1836.             IF TYPTR¬=NIL THEN
1837.             BEGIN LOAD(GATTR);
1838.                IF LFG THEN
1839.                BEGIN
1840.                   IF TYPTR@.FORM = NUMERIC THEN GENRR(#13,STK(RP),STK(RP))
1841.                   ELSE IF TYPTR=REALPTR THEN GENRR(#23,RSTK(RRP),RSTK(RRP))
1842.                   ELSE ERROR(50);
1843.                END
1844.             END
1845.          END;
1846.          WHILE NO=7 DO
1847.          BEGIN LADOPCL:=CL; INSYMBOL; TERM;
1848.             IF (LATTR.TYPTR¬=NIL)&(GATTR.TYPTR¬=NIL) THEN
1849.             BEGIN BT1:=LATTR.TYPTR@.FORM=NUMERIC;
1850.                   BT2:=GATTR.TYPTR@.FORM=NUMERIC;
1851.                IF BT1&BT2&(GATTR.KIND=SVAL) THEN
1852.                BEGIN WITH LATTR DO
1853.                   BEGIN TYPTR:=INTPTR;
1854.                      CASE LADOPCL OF
1855.                 "+" 1: CTERM:=CTERM+GATTR.VAL;
1856.                 "-" 2: CTERM:=CTERM-GATTR.VAL;
1857.                      3: ERROR(50)
1858.                      END
1859.                END
```

```
1860.                  END ELSE
1861.                  BEGIN LOAD(GATTR);
1862.                     IF BT1&BT2 THEN
1863.                     BEGIN LATTR.TYPTR:=INTPTR; RP:=RP-1;
1864.                        CASE LADOPCL OF
1865.                    1:  GENRR(#1A,STK(RP),STK(RP+1));
1866.                    2:  GENRR(#1B,STK(RP),STK(RP+1));
1867.                    3:  ERROR(50)
1868.                        END
1869.                     END ELSE
1870.                     IF (LATTR.TYPTR=REALPTR)&(GATTR.TYPTR=REALPTR) THEN
1871.                     BEGIN
1872.                        RRP:=RRP-1;
1873.                        CASE LADOPCL OF
1874.                    1:  GENRR(#2A,RSTK(RRP),RSTK(RRP+1));
1875.                    2:  GENRR(#2B,RSTK(RRP),RSTK(RRP+1));
1876.                    3:  ERROR(50)
1877.                        END
1878.                     END ELSE
1879.                     IF LATTR.TYPTR=GATTR.TYPTR THEN
1880.                     BEGIN RP:=RP-1;
1881.                        IF (LATTR.TYPTR=BOOLPTR)&(LADOPCL=3) THEN
1882.                        GENRR(#16,STK(RP),STK(RP+1)) ELSE
1883.                        IF LATTR.TYPTR@.FORM = POWER THEN
1884.                        CASE LADOPCL OF
1885.                     1: BEGIN
1886.                           GENRR(#17,STK(RP+1),STK(RP));
1887.                           GENRR(#14,STK(RP),STK(RP+1))
1888.                        END;
1889.                        2: ERROR(50);
1890.                        3: GENRR(#16,STK(RP),STK(RP+1))
1891.                        END
1892.                     END ELSE ERROR(50)
1893.                  END
1894.               END
1895.            END "WHILE NO=7 " ;
1896.            GATTR:=LATTR;
1897.         END "IF LFG|(NO=7)"
1898.      END "SIMPLEEXP";
1899.
1900.         PROCEDURE EXPRESSION;
1901.         VAR LATTR:ATTR; LOF,BT1,BT2: BOOLEAN;
1902.                  XSIZE: INTEGER;
1903.                  LRELOPCL,OP,RREG1,RREG2,D1,R1,D2,R2,AT,AT2,SAVERP,
1904.                  MASK,COUNT,LCST: INTEGER;
1905.
1906.          BEGIN
1907.          SAVERP:=RP;  AT2:=0;
1908.          SIMPLEEXP;
1909.          IF NO=8 THEN " RELOP "
1910.            BEGIN LATTR:=GATTR;  LOF:=TRUE;
1911.              IF LATTR.TYPTR¬=NIL THEN
1912.                IF(LATTR.TYPTR@.FORM>=ARRAYS)|(LATTR.TYPTR=ALFAPTR) THEN
1913.                  BEGIN  GENSADDR(LATTR);
1914.                    D1:=SSADDR.D2;  R1:=SSADDR.B2;  RP:=RP+1;
1915.                  END
1916.                ELSE IF(LATTR.KIND=SVAL)&((CL=7)|((LATTR.VAL=0)&
1917.                        (LATTR.TYPTR¬=BOOLPTR))) THEN
1918.                  BEGIN  LOF:=FALSE;  LCST:=LATTR.VAL;
1919.                    IF CL<=4 THEN CL:=5-CL;
1920.                  END
```

```
1921.                 ELSE LOAD(LATTR);
1922.               LRELOPCL:=CL;  INSYMBOL;  SIMPLEEXP;
1923.               IF (LATTR.TYPTR¬=NIL)&(GATTR.TYPTR¬=NIL) THEN
1924.                 BEGIN IF(GATTR.TYPTR@.FORM>=ARRAYS)|(GATTR.TYPTR=ALFAPTR) THEN
1925.                   BEGIN  GENSADDR(GATTR);
1926.                     D2:=SSADDR.D2;  R2:=SSADDR.B2;
1927.                   END
1928.                   ELSE IF(GATTR.KIND=SVAL)&(GATTR.VAL=0)&
1929.                        (GATTR.TYPTR¬=BOOLPTR)&LOF
1930.                   THEN LOF:=FALSE
1931.                   ELSE LOAD(GATTR);
1932.
1933.                   IF LRELOPCL=7 THEN " IN "
1934.                     BEGIN  IF(GATTR.TYPTR@.FORM=POWER)&
1935.                              (GATTR.TYPTR@.ELSET=LATTR.TYPTR)
1936.                            |(GATTR.TYPTR=PNUMPTR)&(LATTR.TYPTR@.FORM=NUMERIC)
1937.                       THEN BEGIN  IF LOF THEN
1938.                         BEGIN  GENRS(#88,STK(RP),0,0,STK(RP-1)); " SRL "
1939.                           GENRX(#41,STK(RP-1),1,0,0);  " LA "
1940.                           GENRR(#14,STK(RP-1),STK(RP)); " NR "
1941.                           RP:=RP-1;
1942.                         END
1943.                         ELSE BEGIN  GENRS(137,STK(RP),31-LCST,0,0); "SLL"
1944.                           GENRS(#88,STK(RP),31,0,0);  " SRL "
1945.                         END
1946.                       END ELSE ERROR(50);
1947.                     END"  LRELOPCL=7  "
1948.                   ELSE BEGIN BT1:=LATTR.TYPTR@.FORM=NUMERIC;
1949.                     BT2:=GATTR.TYPTR@.FORM=NUMERIC;
1950.                     IF BT1&(GATTR.TYPTR=REALPTR)THEN
1951.                       BEGIN  IF LOF THEN FIXEDTOFLOAT;
1952.                         IF LRELOPCL<=4 THEN LRELOPCL:=5-LRELOPCL;
1953.                         LATTR.TYPTR:=REALPTR;
1954.                       END;
1955.                     IF BT2&(LATTR.TYPTR=REALPTR) THEN
1956.                       BEGIN  IF LOF THEN FIXEDTOFLOAT;
1957.                         GATTR.TYPTR:=REALPTR;
1958.                       END;
1959.                     IF(BT1&BT2)|(LATTR.TYPTR=GATTR.TYPTR)&
1960.                       (LATTR.TYPTR@.FORM<=POINTER)&(LATTR.TYPTR¬=ALFAPTR)
1961.                       |(LATTR.TYPTR@.FORM=POINTER)&(GATTR.TYPTR=NILPTR)
1962.                       |(GATTR.TYPTR@.FORM=POINTER)&(LATTR.TYPTR=NILPTR)
1963.                       THEN BEGIN
1964.                       IF LATTR.TYPTR=REALPTR THEN
1965.                         IF LOF THEN
1966.                           BEGIN OP:=#29; " CDR "
1967.                             RREG2:=RSTK(RRP);  RREG1:=RSTK(RRP-1);
1968.                             RRP:=RRP-2;
1969.                           END
1970.                         ELSE BEGIN  OP:=#22;  " LTDR  "
1971.                             RREG2:=RSTK(RRP);  RREG1:=RSTK(RRP);
1972.                             RRP:=RRP-1;
1973.                           END
1974.                       ELSE IF LOF THEN
1975.                         BEGIN  OP:=#19;  " CR "
1976.                           RREG2:=STK(RP);  RREG1:=STK(RP-1);
1977.                           RP:=RP-2;
1978.                         END
1979.                       ELSE BEGIN OP:=#12; " LTR "
1980.                           RREG2:=STK(RP);  RREG1:=STK(RP);  RP:=RP-1;
```

```
1981.                 END
1982.             END  "  IF(BT1&BT2)|...   "
1983.           ELSE IF (LATTR.TYPTR@.FORM=POWER)&
1984.                   (GATTR.TYPTR@.FORM=POWER) THEN
1985.             IF(LATTR.TYPTR=GATTR.TYPTR)|(LATTR.TYPTR=LAMPTR)
1986.                 |(GATTR.TYPTR=LAMPTR) THEN
1987.               BEGIN  IF(LRELOPCL=1)|(LRELOPCL=4)THEN ERROR(88)
1988.                 ELSE IF LRELOPCL>=5 THEN IF LOF THEN OP:=#19 "CR "
1989.                                          ELSE OP:=#12 "LTR "
1990.                 ELSE IF ¬LOF THEN ERROR(89)
1991.                 ELSE BEGIN  OP:=#14; " NR "
1992.                   IF LRELOPCL=2 THEN
1993.                     BEGIN  RREG2:=STK(RP);  RREG1:=STK(RP-1); END
1994.                   ELSE BEGIN RREG2:=STK(RP-1); RREG1:=STK(RP); END;
1995.                   GENRR(#13,RREG2,RREG2);  " LCR "
1996.                   GENRR(#6,RREG2,0);  " BCTR "
1997.                   LRELOPCL:=6;
1998.                 END;
1999.                 IF LOF THEN RP:=RP-2 ELSE RP:=RP-1;
2000.               END;  " POWERSETS "
2001.             MASK:=MASKARRAY(LRELOPCL);
2002.           END; "BT1:=LATTR.TYPTR@.FORM=NUMERIC"
2003.           IF LATTR.TYPTR@.FORM>=CLASSS THEN ERROR(50)
2004.           ELSE IF(GATTR.TYPTR=LATTR.TYPTR)&
2005.                   ((LATTR.TYPTR@.FORM>=ARRAYS)|(LATTR.TYPTR=ALFAPTR))
2006.           THEN BEGIN  XSIZE:=LATTR.TYPTR@.SIZE;
2007.             COUNT:=(XSIZE-1) DIV 256;
2008.             IF COUNT>0 THEN
2009.               BEGIN  XSIZE:=XSIZE MOD 256;
2010.                 GENRX(#41,0,COUNT,0,0); " LA "
2011.                 AT:=IC;
2012.                 GENSS(#D5,D1,255,R1,D2,R2);  " CLC "
2013.                 GENRX(#47,15-MASK,0,0,0);  AT2:=IC-2;
2014.                 GENRX(#41,R1,256,0,R1);
2015.                 GENRX(#41,R2,256,0,R2);
2016.                 GENRX(#46,0,0,0,0);  " BCT "
2017.                 INS(AT,IC-2);
2018.               END;
2019.             GENSS(#D5,D1,XSIZE-1,R1,D2,R2);
2020.             RP:=SAVERP;
2021.           END
2022.           ELSE GENRR(OP,RREG1,RREG2);  " SPECIFIC TEST "
2023.
2024.           IF LRELOPCL¬=7 THEN
2025.             BEGIN  IF RP<ISTKLIM THEN RP:=RP+1 ELSE STKERR;
2026.               GENRX(#41,STK(RP),1,0,0);  " LA - TRUE "
2027.               GENRX(#47,MASK,0,0,0);  " BRANCH IF TRUE "
2028.               AT:=IC-2;
2029.               IF AT2¬=0 THEN INS(IC,AT2); " FIX LOOP EXIT "
2030.               GENRR(27,STK(RP),STK(RP)); " SR - FALSE "
2031.               INS(IC,AT);  " JUMPS HERE IF TRUE "
2032.             END
2033.         END;  " TYPTRS ¬ NIL "
2034.       LATTR.TYPTR:=BOOLPTR;  LATTR.KIND:=LVAL;  LATTR.CTERM:=0;
2035.       GATTR:=LATTR;
2036.     END  "  IF NO:=8  "
2037.   END "EXPRESSION";
2038.
2039.   PROCEDURE PASSPARAMS;
2040.   VAR LPL,LPC,LDSP,IT2,LPA,RPSAVE,RRPSAVE,LTCT: INTEGER;
```

```
2041.          LPK,BT: BOOLEAN; TP:CTP; ISIZE:INTEGER;
2042.           LFP:CTP; TATTR:ATTR; XCONS:CONSTANT;
2043.       BEGIN
2044.         LPL:=CTPTR@.PROCLEVEL;  LFP:=CTPTR@.FORMALS;
2045.         LPK:=CTPTR@.PROCKIND=ACTUAL;
2046.         XCONS.KONSTKIND:=EXTREF;  XCONS.EVALUE:=CTPTR@.SDNAME;
2047.         LDSP:=72;  LPC:=0;  RPSAVE:=RP;  RRPSAVE:=RRP;  LTCT:=TCT;
2048.         IF CTPTR@.PROCTYPE¬=CTPTR THEN " FUNCTION "
2049.           BEGIN  TP:=CTPTR@.PROCTYPE;
2050.             UPALIGN(TCT,TP@.ALIGN);
2051.             TATTR.ALIGNMENT:=TP@.ALIGN;  TATTR.KIND:=VARBL;
2052.             TATTR.TYPTR:=TP;  TATTR.BREG:=LEVEL;
2053.             TATTR.DPLMT:=TCT;  TATTR.ACCESS:=DRCT;  TATTR.PCKD:=FALSE;
2054.             IT2:=TCT;  TCT:=TCT+TP@.SIZE;  IF TCT>TMAX THEN TMAX:=TCT;
2055.             LOADADR(TATTR);  TATTR.DPLMT:=IT2;  TATTR.ACCESS:=DRCT;
2056.             GENRX(#50,STK(RP),72,0,2);  " ST "
2057.             RP:=RPSAVE;
2058.             LDSP:=76;
2059.           END;
2060.         IF NO=9 THEN " ( "
2061.           BEGIN
2062.             REPEAT
2063.               IF LFP=NIL THEN
2064.                 BEGIN  IF LPK THEN
2065.                   BEGIN  ERROR(72);  SKIP(49);  GOTO 1;  END;
2066.                   BT:=FALSE;
2067.                 END
2068.               ELSE BT:=LFP@.KLASS=PROC;
2069.               INSYMBOL;
2070.               IF BT THEN "  PROC FCT TO BE PASSED "
2071.                 BEGIN  "  NOT IMPLEMENTED  "
2072.                   ERROR(11);  INSYMBOL;
2073.                 END
2074.               ELSE "  EXPRESSION TO BE PASSED  "
2075.                 BEGIN  B6DPL:=LDSP;  EXPRESSION;
2076.                   IF GATTR.TYPTR¬=NIL THEN
2077.                     BEGIN  IF LPK THEN
2078.                       BEGIN  IF(LFP@.KLASS=KONST)|(LFP@.VKIND=ACTUAL) THEN
2079.                         BEGIN IF GATTR.TYPTR¬=ALFAPTR THEN LOAD(GATTR)
2080.                             ELSE GENSADDR(GATTR);
2081.                           IF LFP@.KLASS=KONST THEN
2082.                             BEGIN  PT:=LFP@.CONTYPE; LDSP:=LFP@.CADDR; END
2083.                           ELSE BEGIN PT:=LFP@.VTYPE; LDSP:=LFP@.VADDR END;
2084.                           IF(GATTR.TYPTR@.FORM=NUMERIC)&(PT=REALPTR) THEN
2085.                             BEGIN  FIXEDTOFLOAT;
2086.                               GATTR.TYPTR:=REALPTR;
2087.                             END
2088.                         END
2089.                         ELSE BEGIN
2090.                           IF GATTR.KIND=VARBL THEN LOADADR(GATTR)
2091.                           ELSE ERROR(86);
2092.                           PT:=LFP@.VTYPE;
2093.                           LDSP:=LFP@.VADDR;
2094.                           IF PT¬=NIL THEN
2095.                             IF GATTR.TYPTR@.SIZE¬=PT@.SIZE THEN
2096.                               ERROR(60);
2097.                         END;
2098.                         IF PT¬=NIL THEN
2099.                           IF GATTR.TYPTR¬=PT THEN
2100.                           IF(GATTR.TYPTR@.FORM¬=NUMERIC)|
```

```
                    (PT@.FORM¬=NUMERIC) THEN
                  IF(GATTR.TYPTR¬=NILPTR)|
                    (PT@.FORM¬=POINTER) THEN
                  IF(GATTR.TYPTR¬=LAMPTR)|
                    (PT@.FORM¬=POWER) THEN
                  ERROR(60);
              END  "  IF LPK  "
              ELSE BEGIN  "  FORMAL PROCEDURE STUFF  "
              END
            END  " IF GATTR.TYPTR¬:=NIL "
        END;  "  EXPRESSION TO BE PASSED  "
      "  STORE INTO PARAMETER AREA  "
      IF LPK THEN
        BEGIN  ISIZE:=PT@.SIZE;
          IF(LFP@.KLASS=KONST)|(LFP@.VKIND=ACTUAL) THEN
            IF GATTR.TYPTR=ALFAPTR THEN   " MVC "
              GENSS(#D2,LDSP,9,2,SSADDR.D2,SSADDR.B2)
            ELSE IF ISIZE=2 THEN " STH "
              GENRX(#40,STK(RP),LDSP,0,2)
            ELSE IF ISIZE=1 THEN " STC "
              GENRX(#42,STK(RP),LDSP,0,2)
            ELSE IF ISIZE=8 THEN " STD "
              GENRX(#60,RSTK(RRP),LDSP,0,2)
            ELSE GENRX(#50,STK(RP),LDSP,0,2) " ST "
          ELSE BEGIN GENRX(#50,STK(RP),LDSP,0,2); " ST "
                     ISIZE:=4;  END;
          LDSP:=LDSP+ISIZE;  LFP:=LFP@.NXTEL;
        END
      ELSE BEGIN  " NOT LPK "   END;
      LPC:=LPC+1;  RP:=RPSAVE;  RRP:=RRPSAVE;
    UNTIL NO¬=15;
    IF NO¬=10 THEN " ) "
      BEGIN  ERROR(48);  SKIP(49);  END
    ELSE INSYMBOL;
  END;  " IF NO:=9 "
  IF LFP¬=NIL THEN ERROR(72);
  IF LPK THEN
    BEGIN  IF LPL=LEVEL THEN GENRX(#50,13,0,0,2) " ST "
      ELSE IF LPL=LEVEL-1 THEN GENSS(#D2,0,3,2,0,13) " MVC "
      ELSE IF LPL<LEVEL-1 THEN
        BEGIN  LOADBASE(1,LPL+1);  GENSS(#D2,0,3,2,0,1); END
      ELSE ERROR(99);
      LDCST2(XCONS,15);
    END
  ELSE BEGIN  " NOT IMPLEMENTED  "  END;
  GENRR(#5,14,15);  "  FINALLY DO BALR "
  GATTR:=TATTR;
 1:B6DPL:=72; RP:=RPSAVE;  RRP:=RRPSAVE;  TCT:=LTCT;
END "PASSPARAMS";

PROCEDURE ASSIGN;
  VAR LATTR:ATTR; PTR:CTP;
    XSIZE,I:INTEGER;
    AT,COUNT,DS,RS,DD,RD: INTEGER;
 BEGIN
  VARIABLE;  LATTR:=GATTR;
  IF(LATTR.TYPTR@.FORM>=ARRAYS) | (LATTR.TYPTR=ALFAPTR) THEN
    BEGIN  GENSADDR(LATTR);
        DD:=SSADDR.D2;  RD:=SSADDR.B2;   RP:=RP+1;  END;
  IF NO¬=20  " = " THEN
```

```
2161.              BEGIN  IF GATTR.TYPTR ¬= NIL THEN ERROR(52);
2162.                   SKIP(20);
2163.                 IF NO¬=20 THEN
2164.                   BEGIN IF GATTR.TYPTR=NIL THEN ERROR(52); GOTO 1 END
2165.              END;
2166.            INSYMBOL;  EXPRESSION;
2167.            IF GATTR.TYPTR=NIL THEN SKIP(49)  ELSE
2168.            IF LATTR.TYPTR ¬=NIL THEN
2169.              BEGIN  " TYPE CHECKING - FIRST PART  "
2170.                 IF (LATTR.TYPTR=REALPTR) & (GATTR.TYPTR@.FORM=NUMERIC) THEN
2171.                 BEGIN LOAD(GATTR); GATTR.KIND:=LVAL; GATTR.CTERM:=0;
2172.                    FIXEDTOFLOAT; GATTR.TYPTR:=REALPTR;
2173.                 END
2174.                 ELSE IF LATTR.TYPTR¬=GATTR.TYPTR THEN
2175.                 IF (LATTR.TYPTR@.FORM¬=NUMERIC)|(GATTR.TYPTR@.FORM¬=NUMERIC)
2176.                 THEN IF(LATTR.TYPTR@.FORM¬=POINTER)|(GATTR.TYPTR¬=NILPTR)
2177.                 THEN IF(LATTR.TYPTR@.FORM¬=POWER)|(GATTR.TYPTR¬=LAMPTR)
2178.                 THEN ERROR(53);
2179.
2180.                 "  INTEGER AND SYMBOLIC BOUNDS CHECKING  "
2181.                 IF(LATTR.TYPTR@.FORM=NUMERIC)&(LATTR.TYPTR¬=INTPTR) THEN
2182.                  BEGIN
2183.                    IF GATTR.KIND=SVAL THEN
2184.                       BEGIN IF(LATTR.TYPTR@.MIN>GATTR.VAL)|
2185.                             (LATTR.TYPTR@.MAX<GATTR.VAL) THEN ERROR(101);
2186.                       END
2187.                    ELSE IF ASSCHECK THEN
2188.                       BEGIN  LOAD(GATTR); GATTR.KIND:=LVAL;  GATTR.CTERM:=0;
2189.                            CHECKBNDS(STK(RP),LATTR.TYPTR@.MIN,
2190.                                      LATTR.TYPTR@.MAX,ASSERR); END
2191.                  END
2192.                ELSE IF(LATTR.TYPTR@.FORM=SYMBOLIC)&(LATTR.TYPTR¬=REALPTR)
2193.                  &(LATTR.TYPTR¬=ALFAPTR)&(GATTR.KIND¬=SVAL)&ASSCHECK
2194.                  THEN BEGIN PTR:=LATTR.TYPTR@.FCONST;  LOAD(GATTR);
2195.                     GATTR.KIND:=LVAL;  GATTR.CTERM:=0;
2196.                     CHECKBNDS(STK(RP),0,PTR@.VALUES.IVALUE,ASSERR);
2197.                  END;
2198.
2199.            "  CODE IS FINALLY PRODUCED  "
2200.            IF(LATTR.TYPTR@.FORM>=CLASSS) " CLASS OR FILES " THEN
2201.               ERROR(53)
2202.            ELSE IF (LATTR.TYPTR@.FORM>POWER)|(LATTR.TYPTR=ALFAPTR) THEN
2203.               BEGIN  GENSADDR(GATTR);
2204.                  DS:=SSADDR.D2;  RS:=SSADDR.B2;
2205.                  XSIZE:=LATTR.TYPTR@.SIZE;
2206.                  IF XSIZE > 256 THEN
2207.                  BEGIN  COUNT := (XSIZE -1) DIV 256;
2208.                     I := 1;
2209.                     WHILE (I<RP)&(STK(I)¬=RS) DO I:=I+1;
2210.                     IF STK(I) ¬= RS THEN
2211.                     BEGIN
2212.                        IF RP < ISTKLIM THEN RP:=RP+1 ELSE STKERR;
2213.                        GENRR(#18,STK(RP),RS); "LR"  RS := STK(RP);
2214.                     END;
2215.                     I := 1;
2216.                     WHILE (I<RP)&(STK(I)¬=RD) DO I:=I+1;
2217.                     IF STK(I) ¬= RD THEN
2218.                     BEGIN
2219.                        IF RP < ISTKLIM THEN RP:=RP+1 ELSE STKERR;
2220.                        GENRR(#18,STK(RP),RD); "LR"  RD := STK(RP);
```

```
                        END;
                        XSIZE:=XSIZE MOD 256;
                        GENRX(#41,0,COUNT,0,0);  " LA "
                        AT:=IC;
                        GENSS(#D2,DD,255,RD,DS,RS);  "  MVC "
                        GENRX(#41,RD,256,0,RD);  " LA "
                        GENRX(#41,RS,256,0,RS);  " LA "
                        GENRX(#46,0,0,0,0);  " BCT "
                           INS(AT,IC-2);
                     END;
                     GENSS(#D2,DD,XSIZE-1,RD,DS,RS); " MVC - FINISH "
                  END  " FORM>POWER OR ALFA "
               ELSE BEGIN  LOAD(GATTR);  STORE(LATTR);  END
            END;
      1:END "ASSIGN";

        PROCEDURE VARIAB;
         BEGIN
           IF NO¬=1 THEN BEGIN ERROR(49);  GATTR.TYPTR:=NIL;  END
           ELSE BEGIN  SEARCH;
             IF CTPTR=NIL THEN BEGIN ERROR(31);  CTPTR:=UNDECPTR; END;
             IF CTPTR@.KLASS <= PROC THEN
               BEGIN  ERROR(32);  INSYMBOL;  GATTR.TYPTR:=NIL;  END
             ELSE VARIABLE;
           END
        END "VARIAB";

        PROCEDURE ALLC;
        VAR LATTR:ATTR; LCA,RREG,RREG1,LALIGN:SHRTINT;
          BT1:BOOLEAN; XCONS:CONSTANT; PT1:CTP;
        BEGIN
          IF NO¬=9 "(" THEN BEGIN ERROR(79); SKIP(49); GOTO 10 END;
          INSYMBOL; VARIAB; PT:=NIL;
          IF GATTR.TYPTR¬=NIL THEN WITH GATTR.TYPTR@ DO
          IF FORM¬=POINTER THEN ERROR(44)
          ELSE BEGIN ADDRESSVAR(DOMAIN,LATTR); LOADADR(LATTR);
            PT:=ELTYPE END;
          IF PT=NIL THEN BEGIN SKIP(49); GOTO 10 END;
          LALIGN:=PT@.ALIGN;
          IF NO=15 THEN ","
          BEGIN PT:=PT@.RECVAR;
            REPEAT
              IF PT=NIL THEN BEGIN ERROR(66); SKIP(49); GOTO 10 END;
              INSYMBOL;
              IF NO=1 THEN
              BEGIN
                SEARCH;
                IF CTPTR=NIL THEN BEGIN ERROR(31); SKIP(49); GOTO 10 END;
                IF CTPTR@.KLASS¬=KONST THEN
                  BEGIN ERROR(63); SKIP(49); GOTO 10 END;
                PT1:=PT@.CASETYPE;
                IF((PT1@.FORM=SYMBOLIC)&(CTPTR@.CONTYPE¬=PT1))|
                  ((PT1@.FORM=NUMERIC)&(CTPTR@.CONTYPE¬=INTPTR))
                  THEN BEGIN ERROR(73); SKIP(49); GOTO 10 END;
                IVAL:=CTPTR@.VALUES.IVALUE
              END
              ELSE IF (NO¬=2)|(CL=2)|(CL=3) THEN
                BEGIN ERROR(63); SKIP(49); GOTO 10 END
              ELSE
              BEGIN PT1:=PT@.CASETYPE;
```

```
2281.                IF ((CL=1)&(PT1@.FORM¬=NUMERIC))|
2282.                   ((CL=4)&(PT1¬=CHARPTR)) THEN
2283.                  BEGIN ERROR(73); SKIP(49); GOTO 10 END
2284.              END;
2285.              CTPTR:=PT@.VARIANTS; BT1:=FALSE;
2286.              WHILE(CTPTR¬=NIL)&¬BT1 DO
2287.              IF CTPTR@.CASEVAL=IVAL THEN BT1:=TRUE ELSE CTPTR:=CTPTR@.NXTEL;
2288.              IF CTPTR=NIL THEN BEGIN IT:=PT@.CASESIZE; PT:=NIL END
2289.              ELSE BEGIN IT:=CTPTR@.CASESIZE; PT:=CTPTR@.VARIANTS END;
2290.              INSYMBOL;
2291.            UNTIL NO¬=15
2292.          END
2293.          ELSE IT:=PT@.SIZE;
2294.          UPALIGN(IT,LALIGN);
2295.          RREG1:=STK(RP); IF RP<ISTKLIM THEN RP:=RP+1 ELSE STKERR;
2296.          RREG:=STK(RP);
2297.          WITH XCONS DO BEGIN KONSTKIND:=INTEGERS; IVALUE:=IT END;
2298.          GENRX(#58,RREG,0,0,RREG1); "L - GET FREE POINTER"
2299.          LDCST2(XCONS,0);
2300.          GENRR(#1A,0,RREG); "AR - NEW FREE POINTER"
2301.          GENRX(#50,0,0,0,RREG1); "ST - SAVE NEW FREE POINTER"
2302.          GENRX(#59,0,4,0,RREG1); "C - SPACE LEFT IN CLASS?"
2303.          GENRX(#47,12,0,0,0); LCA:=IC-2; "BLE - YES"
2304.          GENRX(#50,RREG,0,0,RREG1); "RESTORE OLD FREE POINTER"
2305.          GENRR(#1B,RREG,RREG); "SR - NILVAL"
2306.          INS(IC,LCA);  "FIX UP JUMP"
2307.          STORE(GATTR); "STORE POINTER TO NEW SPACE"
2308.          IF NO¬=10 ")" THEN BEGIN ERROR(48); SKIP(49) END ELSE INSYMBOL;
2309.      10:END "ALLC";
2310.
2311.          PROCEDURE FIXEDTOFLOAT;
2312.          BEGIN
2313.            GENRR(#18,0,STK(RP)); "LR" RP:=Rp-1;
2314.            GENRR(#10,1,0); "LPR"
2315.            GENRS(#88,0,0,31,0); "SRL"
2316.            GENRS(#89,0,0,31,0); "SLL"
2317.            GENRS(#90,0,1,72,12); "STM INTO TEMPORARY LOCATION"
2318.            GENSI(#96,72,12,#4E); "OI #4E - PSEUDOEXPONENT"
2319.            IF RRP<MAXRSTK THEN RRP:=RRP+1 ELSE ERROR(33);
2320.            GENRR(#2B,RSTK(RRP),RSTK(RRP)); "SDR"
2321.            GENRX(#6A,RSTK(RRP),72,0,12); "AD - NORMALIZE"
2322.          END "FIXEDTOFLOAT";
2323.
2324.          PROCEDURE FLOATTOFIXED;
2325.          BEGIN
2326.            RP:=RP+1;
2327.            RRP:=RRP-1;
2328.          END "FLOATTOFIXED";
2329.
2330.          PROCEDURE GETPUT;
2331.          VAR VCON:CONSTANT;
2332.          BEGIN
2333.            IF NO¬=9 "(" THEN BEGIN ERROR(79); SKIP(49); GOTO 10 END;
2334.            INSYMBOL; VARIAB;
2335.            IF GATTR.TYPTR¬= NIL THEN
2336.            WITH GATTR.TYPTR@,VCON,RXADDR DO
2337.            BEGIN
2338.              IF FORM=FILES THEN
2339.              BEGIN "LOAD CORRECT EXTERNAL REFERENCE"
2340.                CASE LPSW OF
```

```
2341.              1 "EOR": BEGIN END;
2342.              2 "GET": IF FELTYPE=CHARPTR THEN EVALUE:='PSCLGETC'
2343.                       ELSE EVALUE:='PSCLGETR';
2344.              3 "PUT": IF FELTYPE=CHARPTR THEN EVALUE:='PSCLPUTC'
2345.                       ELSE EVALUE:='PSCLPUTR';
2346.              4 "RESET": BEGIN END;
2347.              END "CASE";
2348.              GENADDR(GATTR,0); "ADDRESS OF FCB"
2349.              GENRX(#41,1,D2,X2,B2); "LA INTO R1"
2350.              KONSTKIND:=EXTREF;
2351.              LOCST2(VCON,15);
2352.              GENRR(#5,14,15); "BALR TO I/O ROUTINE"
2353.            END
2354.            ELSE IF (FORM=POINTER)&(LPSW=4) THEN
2355.            BEGIN LOAD(GATTR); "VALUE TO RESET"
2356.              ADDRESSVAR(DOMAIN,GATTR);
2357.              STORE(GATTR);
2358.            END
2359.            ELSE ERROR(44);
2360.          END;
2361.          IF NO¬=10 ")" THEN BEGIN ERROR(48); SKIP(49) END ELSE INSYMBOL;
2362.     10:END "GETPUT";
2363.
2364.        PROCEDURE INSAPP;
2365.        VAR LATTR:ATTR;
2366.        BEGIN
2367.          IF NO¬=9 "(" THEN BEGIN ERROR(79); SKIP(49); GOTO 10 END;
2368.          INSYMBOL;
2369.          IF LPSW=8 THEN EXPRESSION ELSE VARIAB;
2370.          IF GATTR.TYPTR¬=NIL THEN
2371.            IF GATTR.TYPTR@.FORM¬=NUMERIC THEN ERROR(44) ELSE LOAD(GATTR);
2372.          LATTR:=GATTR;
2373.          IF NO¬=15 THEN BEGIN ERROR(80); SKIP(49); GOTO 10 END;
2374.          INSYMBOL; EXPRESSION;
2375.          WITH GATTR DO
2376.          IF TYPTR¬=NIL THEN
2377.            IF TYPTR@.FORM¬=NUMERIC THEN ERROR(44)
2378.            ELSE IF KIND=SVAL THEN GENRS(#89,STK(RP),VAL,0,0) "SLL"
2379.            ELSE BEGIN LOAD(GATTR); RP:=RP-1;
2380.              GENRS(#89,STK(RP),0,0,STK(RP+1)) "SLL"
2381.            END;
2382.          IF NO¬=15 THEN BEGIN ERROR(80); SKIP(49); GOTO 10 END;
2383.          INSYMBOL;
2384.          IF LPSW=8 THEN VARIAB ELSE EXPRESSION;
2385.          WITH GATTR DO
2386.          IF TYPTR¬=NIL THEN
2387.            IF TYPTR@.FORM¬=NUMERIC THEN ERROR(44)
2388.            ELSE BEGIN LOAD(GATTR); RP:=RP-1;
2389.              GENRR(#16,STK(RP),STK(RP+1));
2390.              IF LPSW=8 THEN STORE(GATTR) ELSE STORE(LATTR)
2391.            END;
2392.          IF NO¬=10 ")" THEN BEGIN ERROR(48); SKIP(49) END ELSE INSYMBOL;
2393.     10:END "INSAPP";
2394.
2395.        PROCEDURE PCK;
2396.        VAR LPT:CTP; SREG,D,L:SHRTINT;
2397.        BEGIN
2398.          IF NO¬=9 "(" THEN BEGIN ERROR(79); SKIP(49); GOTO 10 END;
2399.          INSYMBOL; VARIAB;
2400.          WITH GATTR DO
```

```
2401.          IF TYPTR¬=NIL THEN
2402.          WITH TYPTR@ DO
2403.          BEGIN
2404.            IF FORM¬=ARRAYS THEN ERROR(44)
2405.            ELSE IF AELTYPE¬=CHARPTR THEN ERROR(44)
2406.            ELSE BEGIN LPT:=INXTYPE; DPLMT:=DPLMT-LO END;
2407.            L := SIZE;  LOADADR(GATTR)
2408.          END
2409.          ELSE BEGIN  LPT:=INTPTR; L:=1; "ERROR CONDX"  END;
2410.          IF L>10 THEN L:=10;
2411.          IF NO¬=15 THEN BEGIN ERROR(80); SKIP(49); GOTO 10 END;
2412.          INSYMBOL; EXPRESSION;
2413.          WITH GATTR DO
2414.          IF TYPTR¬=NIL THEN
2415.            IF (TYPTR@.FORM¬=NUMERIC)|(LPT@.FORM¬=NUMERIC) THEN
2416.              BEGIN IF TYPTR¬=LPT THEN ERROR(44) END
2417.            ELSE IF KIND=SVAL THEN D:=VAL
2418.          ELSE
2419.          BEGIN
2420.            LOAD(GATTR); RP:=RP-1;  D:=0;
2421.            GENRR(#1A,STK(RP),STK(RP+1)); "AR"
2422.          END;
2423.          SREG:=RP;
2424.          IF NO¬=15 THEN BEGIN ERROR(80); SKIP(49); GOTO 10 END;
2425.          INSYMBOL; VARIAB;
2426.          WITH GATTR,SSADDR DO
2427.          IF TYPTR¬=NIL THEN
2428.            IF TYPTR¬=ALFAPTR THEN ERROR(44)
2429.            ELSE
2430.            BEGIN GENSADDR(GATTR);
2431.              GENSS(#D2,D2,L-1,B2,D,STK(SREG)); "MVC"
2432.            END;
2433.          IF NO¬=10 ")" THEN BEGIN ERROR(48); SKIP(49) END ELSE INSYMBOL;
2434.      10:END "PCK";
2435.
2436.        PROCEDURE READIR;
2437.        BEGIN
2438.        END "READIR";
2439.
2440.        PROCEDURE TITLE;
2441.        BEGIN
2442.        END "TITLE";
2443.
2444.        PROCEDURE UNPCK;
2445.        VAR LPT:CTP; D,LD,LB:SHRTINT;
2446.        BEGIN
2447.          IF NO¬=9 "(" THEN BEGIN ERROR(79); SKIP(49); GOTO 10 END;
2448.          INSYMBOL; EXPRESSION;
2449.          WITH GATTR,SSADDR DO
2450.          IF TYPTR¬=NIL THEN
2451.            IF TYPTR¬=ALFAPTR THEN ERROR(44)
2452.            ELSE BEGIN GENSADDR(GATTR); LD:=D2; LB:=B2; RP:=RP+1 END;
2453.          IF NO¬=15 THEN BEGIN ERROR(80); SKIP(49); GOTO 10 END;
2454.          INSYMBOL; VARIAB;
2455.          WITH GATTR DO
2456.          IF TYPTR¬=NIL THEN
2457.          WITH TYPTR@ DO
2458.          BEGIN
2459.            IF FORM¬=ARRAYS THEN ERROR(44)
2460.            ELSE IF AELTYPE¬=CHARPTR THEN ERROR(44)
```

```
2461.            ELSE BEGIN LPT:=INXTYPE;
2462.              DPLMT:=DPLMT-LO
2463.            END;
2464.            LOADADR(GATTR)
2465.          END
2466.          ELSE LPT:=INTPTR;
2467.          IF NO¬=15 THEN BEGIN ERROR(80); SKIP(49); GOTO 10 END;
2468.          INSYMBOL; EXPRESSION;
2469.          WITH GATTR DO
2470.          IF TYPTR¬=NIL THEN
2471.            IF (TYPTR@.FORM¬=NUMERIC)!(LPT@.FORM¬=NUMERIC) THEN
2472.             BEGIN IF TYPTR¬=LPT THEN ERROR(44) END
2473.            ELSE IF KIND=SVAL THEN D:=VAL
2474.            ELSE BEGIN LOAD(GATTR); RP:=RP-1; D:=0;
2475.              GENRR(#1A,STK(RP),STK(RP+1)) "AR"
2476.            END;
2477.          GENSS(#D2,D,9,STK(RP),LD,LB); "MVC"
2478.          IF NO¬=10 ")" THEN BEGIN ERROR(48); SKIP(49) END ELSE INSYMBOL;
2479.      10:END "UNPCK";
2480.
2481.        PROCEDURE WRITEIR;
2482.        BEGIN
2483.        END "WRITEIR";
2484.
2485.        PROCEDURE WRITOUT;
2486.        VAR I,ESDID,J,LIC,AMT:SHRTINT;
2487.          K:INTEGER;
2488.        BEGIN
2489.          PASCALGO@:=BLANKCARD;
2490.          "FIRST ESD ITEM IS CSECT NAME"
2491.          PASCALGO@(1):=CHR(2);  UNPACK('ESD',PASCALGO@,2);
2492.          PASCALGO@(11):=CHR(0); PASCALGO@(12):=CHR(16);
2493.          UNPACK(BLANKALFA,PASCALGO@,13);
2494.          PASCALGO@(15):=CHR(0); PASCALGO@(16):=CHR(1);
2495.          UNPACK(CNAME,PASCALGO@,17);
2496.          FOR I:=25 TO 28 DO PASCALGO@(I):=CHR(0);
2497.          PASCALGO@(29):=' '; PASCALGO@(30):=CHR(IC DIV 65536);
2498.          PASCALGO@(31):=CHR(IC MOD 65536 DIV 256);
2499.          PASCALGO@(32):=CHR(IC MOD 256);
2500.          PUT(PASCALGO);
2501.          "WRITE OUT ESD ENTRIES FOR EXTERNAL REFERENCES"
2502.          ESDID:=1; I:=ELCX; UNPACK(BLANKALFA,PASCALGO@,29);
2503.          WHILE I¬=0 DO
2504.          BEGIN
2505.            ESDID:=ESDID+1; PASCALGO@(15):=CHR(ESDID DIV 256);
2506.            PASCALGO@(16):=CHR(ESDID MOD 256);
2507.            UNPACK(CSTTB(I).VALU.EVALUE,PASCALGO@,17);
2508.            PASCALGO@(25):=CHR(2);
2509.            FOR J:=26 TO 28 DO PASCALGO@(J):=CHR(0);
2510.            PUT(PASCALGO);
2511.            I:=CSTTB(I).CNEXT;
2512.          END;
2513.          "WRITE OUT ESD ENTRIES FOR EXTERNAL LABEL REFERENCES"
2514.          PASCALGO@(15):=' '; PASCALGO@(16):=' ';
2515.          PASCALGO@(29):=' '; PASCALGO@(30):=CHR(0);
2516.          PASCALGO@(31):=CHR(0); PASCALGO@(32):=CHR(1);
2517.          FOR IT:=FSTIXG TO CEXTABIX DO
2518.          BEGIN
2519.            FOR IT1:=1 TO CLABIX DO
2520.              IF EXTAB(IT).EXVAL=LABTAB(IT1).LABVAL THEN
```

```
2521.              BEGIN
2522.                UNPACK(EXTAB(IT).EXTNAME,PASCALGO@,17);
2523.                PASCALGO@(25):=CHR(1); K:=LABTAB(IT1).LABLOC;
2524.                PASCALGO@(26):=CHR(K DIV 65536);
2525.                PASCALGO@(27):=CHR(K MOD 65536 DIV 256);
2526.                PASCALGO@(28):=CHR(K MOD 256);
2527.                PUT(PASCALGO);
2528.                GOTO 1
2529.              END;
2530.            ERRMESSAGE('EXIT:',EXTAB(IT).EXVAL);
2531.          1:END;
2532.            "WRITE OUT TEXT CARDS"
2533.            UNPACK('TXT',PASCALGO@,2); PASCALGO@(11):=CHR(0);
2534.            PASCALGO@(13):=' '; PASCALGO@(14):=' ';
2535.            PASCALGO@(15):=CHR(0); PASCALGO@(16):=CHR(1);
2536.            LIC:=0;
2537.            WHILE LIC<IC DO
2538.            BEGIN
2539.              IF IC-LIC>=56 THEN AMT:=56 ELSE AMT:=IC-LIC;
2540.              PASCALGO@(6):=CHR(LIC DIV 65536);
2541.              PASCALGO@(7):=CHR(LIC MOD 65536 DIV 256);
2542.              PASCALGO@(8):=CHR(LIC MOD 256);
2543.              PASCALGO@(12):=CHR(AMT);
2544.              FOR I:=LIC TO AMT+LIC-1 DO PASCALGO@(I-LIC+17):=CHR(CODE(I));
2545.              FOR I:=AMT+17 TO 72 DO PASCALGO@(I):=' ';
2546.              LIC:=LIC+AMT;
2547.              PUT(PASCALGO)
2548.            END;
2549.            "WRITE OUT RLD CARDS"
2550.            PASCALGO@:=BLANKCARD; PASCALGO@(1):=CHR(2);
2551.            UNPACK('RLD',PASCALGO@,2);
2552.            PASCALGO@(11):=CHR(0); PASCALGO@(12):=CHR(8);
2553.            PASCALGO@(19):=CHR(0); PASCALGO@(20):=CHR(1);
2554.            PASCALGO@(21):=CHR(28);
2555.            ESDID:=1; I:=ELCX;
2556.            WHILE I¬=0 DO
2557.            BEGIN ESDID:=ESDID+1;
2558.              PASCALGO@(17):=CHR(ESDID DIV 256);
2559.              PASCALGO@(18):=CHR(ESDID MOD 256);
2560.              K:=CSTTB(I).INX; PASCALGO@(22):=CHR(K DIV 65536);
2561.              PASCALGO@(23):=CHR(K MOD 65536 DIV 256);
2562.              PASCALGO@(24):=CHR(K MOD 256);
2563.              PUT(PASCALGO);
2564.              I:=CSTTB(I).CNEXT
2565.            END;
2566.            "WRITE OUT END CARD"
2567.            PASCALGO@:=BLANKCARD; PASCALGO@(1):=CHR(2);
2568.            UNPACK('END',PASCALGO@,2);
2569.            PUT(PASCALGO);
2570.            IF(LEVEL=0)&(CNAME='$$$MAIN') THEN
2571.            BEGIN
2572.              UNPACK(' INCLUDE S',PASCALGO@,1);
2573.              UNPACK('YSLIB(PSCL',PASCALGO@,11);
2574.              UNPACK('MON)      ',PASCALGO@,21);
2575.              PUT(PASCALGO); PASCALGO@:=BLANKCARD;
2576.              UNPACK(' ENTRY PSC',PASCALGO@,1);
2577.              UNPACK('LMON      ',PASCALGO@,11);
2578.              PUT(PASCALGO);
2579.            END
2580.          END "WRITOUT";
2581.
```

```
2582.            PROCEDURE IFSTAT ;
2583.            VAR LCA1, LCA2: ADDRESS;
2584.            BEGIN
2585.               INSYMBOL ; EXPRESSION ;
2586.               IF GATTR.TYPTR ¬= NIL THEN GENJP(0) ;
2587.               LCA1 := IC-2;
2588.               IF NO ¬= 24 THEN  " SY ¬= THEN "
2589.               BEGIN
2590.                  IF GATTR.TYPTR ¬= NIL THEN ERROR(56) ; SKIP(24) ;
2591.                  IF NO ¬= 24 THEN
2592.                  BEGIN
2593.                     IF GATTR.TYPTR = NIL THEN ERROR(56) ;
2594.                     IF ERRCL(NO) = ENDSY THEN GOTO 30 ; GOTO 20 ;
2595.                  END
2596.               END ;
2597.               INSYMBOL ;
2598.      20:      STATEMENT ;
2599.      30:      ISTKLIM:=MAXISTK;  ASTK(MAXISTK):=0;
2600.               IF NO ¬= 25 THEN  " SY ¬= ELSE "
2601.               INS(IC,LCA1)  ELSE
2602.               BEGIN
2603.                  GENRX(#47,#F,0,0,0);
2604.                  LCA2 := IC-2;
2605.                  INS(IC,LCA1);
2606.                  INSYMBOL ; STATEMENT ;
2607.                  INS(IC,LCA2);
2608.                  ISTKLIM:=MAXISTK;  ASTK(MAXISTK):=0
2609.               END
2610.            END  " IFSTAT "  ;
2611.
2612.
2613.         PROCEDURE CASESTAT ;
2614.         TYPE PTR = @LCSLABS ;
2615.         VAR LCSLABS : CLASS 30 OF "PACKED" RECORD NEXT : PTR ;
2616.                                                  CSLAB : SHRTINT ;
2617.                                                  ADDR : ADDRESS ;
2618.                                                END ;
2619.             CSJMP, BASEJMP, OUTCASE, LMIN, LMAX : SHRTINT ;
2620.               LPTR, PT1, PT3 : PTR ; LCTP : CTP ;
2621.               BJ: CONSTANT;  LERR: BOOLEAN;
2622.
2623.           PROCEDURE ACASE ;
2624.           VAR IT,FIXUP : SHRTINT ; LERRFG : BOOLEAN ; PT1,PT2 : PTR ;
2625.           BEGIN LERRFG := TRUE ;
2626.             REPEAT IF (NO=15)|(NO=27) THEN ", OF" INSYMBOL ;
2627.             IT:=4097;  "GREATER THAN TWOTO12"
2628.             IF NO = 1 THEN "ID"
2629.             BEGIN SEARCH ;
2630.               IF CTPTR = NIL THEN
2631.               BEGIN ERROR(31) ; CTPTR := UNDECPTR END ;
2632.               WITH CTPTR@ DO
2633.               BEGIN IF KLASS ¬= KONST THEN
2634.                 BEGIN IF LERRFG THEN ERROR(61) ; GOTO 1 END ;
2635.                 IF CONTYPE ¬= NIL THEN
2636.                 BEGIN IF CONKIND = FORMAL THEN
2637.                   BEGIN ERROR(61) ; GOTO 1 END ;
2638.                   IF LCTP = NIL THEN LCTP := CONTYPE ELSE
2639.                   IF LCTP ¬= CONTYPE THEN ERROR(73) ;
2640.                   IT := VALUES.IVALUE ;
2641.                 END
```

```
2642.              END
2643.            END "IF NO = 1" ELSE
2644.            IF NO = 2 THEN "CONST"
2645.            BEGIN CASE CL OF
2646.              1: BEGIN PT := INTPTR ; IT:=IVAL END;
2647.              2: PT := REALPTR ;
2648.              3: PT := ALFAPTR ;
2649.              4: BEGIN  PT := CHARPTR ; IT:=INT(CHVAL)  END;
2650.              END ;
2651.              IF LCTP = NIL THEN LCTP := PT ELSE
2652.              IF LCTP ¬= PT THEN ERROR(73) ;
2653.            END "IF NO = 2" ELSE
2654.            BEGIN IF LERRFG THEN ERROR(61) ; GOTO 1 END ;
2655.            IF IT >= TWOTO12 THEN ERROR(100) ;
2656.            LERRFG := FALSE ; PT1 := LPTR ; PT2 := NIL ;
2657.            IF ¬ERR THEN
2658.            WHILE PT1 ¬= NIL DO
2659.            BEGIN IF PT1@.CSLAB >= IT THEN
2660.              BEGIN IF PT1@.CSLAB = IT THEN ERROR(77) ; GOTO 3 END ;
2661.              PT2 := PT1 ; PT1 := PT1@.NEXT ;
2662.            END ;
2663.            LMAX := IT ;    "IT>ALL PT1@.CSLAB'S"
2664.      3:    ALLOC(PT3) ;    "NEW LABEL-DESCRIPTOR"
2665.            IF PT3 ¬= NIL THEN
2666.            BEGIN WITH PT3@ DO
2667.              BEGIN NEXT := PT1 ; CSLAB := IT ; ADDR := IC END ;
2668.              IF PT2 = NIL THEN
2669.              BEGIN LPTR := PT3 ; LMIN := IT END ELSE PT2@.NEXT := PT3 ;
2670.            END ELSE ERROR(71) ;
2671.            INSYMBOL ;
2672.            UNTIL NO ¬= 15 ; ","
2673.            IF NO ¬= 19 THEN ":" ERROR(64) ELSE INSYMBOL ;
2674.      1:    STATEMENT ; GENRX(#47,15,0,0,0); "B" FIXUP:=IC-2;
2675.            IF NO = 16 THEN ";"
2676.            BEGIN INSYMBOL ; IF NO ¬= 22 THEN ACASE END ELSE
2677.            IF ERRCL(NO) = BEGSY THEN
2678.            BEGIN ERROR(58) ; GOTO 1 END ELSE
2679.            IF NO ¬= 22 THEN ERROR(68) ;
2680.            BASEJMP:=IC-LMIN*2;
2681.            OUTCASE:=BASEJMP+(LMAX+1)*2;
2682.            INS(OUTCASE,FIXUP)
2683.          END "ACASE" ;
2684.
2685.        BEGIN "CASESTAT"
2686.          LMIN := 0 ; LMAX := 0 ; LPTR := NIL ; LERR := ERR ; ERR := FALSE ;
2687.          INSYMBOL ; EXPRESSION ;
2688.          LCTP := GATTR.TYPTR ;
2689.          IF LCTP ¬= NIL THEN
2690.          BEGIN IF LCTP@.FORM = NUMERIC THEN LCTP := INTPTR ELSE
2691.            IF (LCTP@.FORM > SYMBOLIC)|(LCTP = REALPTR)|(LCTP = ALFAPTR) THEN
2692.            BEGIN ERROR(62) ; LCTP := NIL END ;
2693.            LOAD(GATTR);
2694.          END ;
2695.         "IF INXCHECK THEN
2696.          BEGIN GEN30(71B,7,0,IC) ; GEN30(71B,2,0,0) ;
2697.            LCA1 := CA ; LCP1 := CP ; GEN15(37B,0,1,2) ;
2698.            GEN30(71B,2,0,0) ; LCA2 := CA ; LCP2 := CP ;
2699.            GEN15(37B,2,2,1) ; GEN15(12B,2,2,0) ;
2700.            GEN30(03B,3,2,INXERR)
2701.          END ;     "
```

```
2702.          IF NO ¬= 27 THEN "OF"
2703.          BEGIN IF LCTP ¬= NIL THEN ERROR(65) ; SKIP(27) ;
2704.            IF NO ¬= 27 THEN
2705.            IF LCTP = NIL THEN ERROR(65) ;
2706.          END ;
2707.          GENRR(#1A,STK(RP),STK(RP)); "AR"
2708.          GENRX(#4A,STK(RP),0,0,0); "AH"
2709.          CSJMP:=IC-2;
2710.          GENRX(#48,1,0,STK(RP),BASEREG(1)); "LH"
2711.          GENRX(#47,15,0,1,BASEREG(1)); "B"
2712.          ACASE ;
2713.          WITH BJ DO BEGIN KONSTKIND:=INTEGERS; IVALUE:=BASEJMP END;
2714.          GENCONST(BJ,CSJMP);
2715.         "IF INXCHECK THEN
2716.          BEGIN INS(LMIN,LCP1,LCA1) ; INS(LMAX,LCP2,LCA2) ; END ;"
2717.          IT := LMIN ; PT1:=LPTR;
2718.          IF ¬ERR THEN
2719.          WHILE PT1 ¬= NIL DO
2720.          BEGIN
2721.            WHILE(IT<PT1@.CSLAB)DO
2722.              BEGIN
2723.                CODE(IC):=OUTCASE DIV 256;
2724.                CODE(IC+1):=OUTCASE MOD 256;
2725.                IC:=IC+2;
2726.                IT:=IT+1
2727.              END;
2728.            CODE(IC):=PT1@.ADDR DIV 256;
2729.            CODE(IC+1):=PT1@.ADDR MOD 256;
2730.            IC:=IC+2;
2731.            IT:=IT+1;
2732.            PT1:=PT1@.NEXT
2733.          END;
2734.          IF NO = 22 THEN "END" INSYMBOL ;
2735.          ERR:=ERR|LERR
2736.        END "CASESTAT" ;
2737.
2738.            PROCEDURE REPEATSTAT ;
2739.            VAR LJPADDR: ADDRESS ;
2740.            BEGIN
2741.               LJPADDR := IC ;
2742.               ISTKLIM:=MAXISTK;  ASTK(MAXISTK):=0;
2743.               REPEAT
2744.               BEGIN
2745.                  INSYMBOL ;
2746.      20:         STATEMENT ;
2747.                  IF ERRCL(NO) = BEGSY THEN BEGIN ERROR(58) ; GOTO 20 END ;
2748.                  IF NO = 25 THEN  "ELSE"
2749.                  BEGIN ERROR(54) ; INSYMBOL ; GOTO 20 END
2750.               END
2751.               UNTIL NO ¬= 16 ;  " SY ¬= ; "
2752.               IF NO ¬= 29 THEN  " SY ¬= UNTIL "  ERROR(67) ELSE
2753.               BEGIN
2754.                  INSYMBOL ; EXPRESSION ;
2755.                  IF GATTR.TYPTR ¬= NIL THEN GENJP(LJPADDR) ELSE SKIP(49)
2756.               END
2757.            END  " REPEATSTAT "  ;
2758.
2759.            PROCEDURE WHILESTAT ;
2760.            VAR LJPADDR : ADDRESS;  LCA : SHRTINT;
2761.            BEGIN
```

```
2762.              ISTKLIM:=MAXISTK;  ASTK(MAXISTK):=0;
2763.              LJPADDR := IC ;
2764.              INSYMBOL ; EXPRESSION ;
2765.              IF GATTR.TYPTR ¬= NIL THEN GENJP(0) ;
2766.              LCA := IC-2;
2767.              IF NO ¬= 31 THEN  " SY ¬= DO "
2768.              BEGIN
2769.                 IF GATTR.TYPTR ¬= NIL THEN ERROR(59) ; SKIP(31) ;
2770.                 IF NO ¬= 31 THEN
2771.                 BEGIN
2772.                    IF GATTR.TYPTR = NIL THEN ERROR(59) ;
2773.                    IF ERRCL(NO) = BEGSY THEN GOTO 20 ; GOTO 10
2774.                 END
2775.              END ;
2776.              INSYMBOL ;
2777.      20:     STATEMENT ;
2778.              GENRX(#47,15,0,0,0); INS(LJPADDR,IC-2);
2779.      10:     INS(IC,LCA);
2780.              ISTKLIM:=MAXISTK;  ASTK(MAXISTK):=0
2781.           END  " WHILESTAT "  ;
2782.
2783.           PROCEDURE FORSTAT ;
2784.           VAR LATTR,TATTR : ATTR ; LCLASS, LCA, I : SHRTINT ;
2785.              LJPADDR : ADDRESS ; LOF : BOOLEAN ;
2786.              XTCT:ADDRESS; PTR:CTP;
2787.
2788.           PROCEDURE CHTYPE ;
2789.           BEGIN
2790.              WITH GATTR DO
2791.              IF TYPTR ¬= NIL THEN
2792.              IF (TYPTR@.FORM > SYMBOLIC)|(TYPTR = REALPTR)|
2793.              (TYPTR = ALFAPTR) THEN
2794.              BEGIN ERROR(62) ; TYPTR := NIL END
2795.           END ;
2796.
2797.           PROCEDURE CHTYPES ;
2798.           BEGIN
2799.              WITH GATTR DO
2800.              IF (TYPTR ¬= NIL)&(LATTR.TYPTR ¬= NIL) THEN
2801.              IF((TYPTR@.FORM = SYMBOLIC)|(LATTR.TYPTR@.FORM = SYMBOLIC))&
2802.              (TYPTR ¬= LATTR.TYPTR) THEN
2803.              BEGIN ERROR(73) ; TYPTR := NIL END
2804.           END ;
2805.
2806.           BEGIN  ISTKLIM := MAXISTK;  ASTK(MAXISTK) := 0 ;
2807.              LCA := 0 ;
2808.              INSYMBOL ;
2809.              IF NO ¬= 1 THEN
2810.              BEGIN ERROR(49) ; GATTR.TYPTR := NIL END ELSE
2811.              BEGIN SEARCH ;
2812.                 IF CTPTR = NIL THEN
2813.                 BEGIN ERROR(31) ; CTPTR := UNDECPTR END ;
2814.                 IF CTPTR@.KLASS <= PROC THEN
2815.                 BEGIN ERROR(32) ; INSYMBOL END ELSE VARIABLE
2816.              END ;
2817.              CHTYPE ;
2818.              IF GATTR.TYPTR ¬= NIL THEN
2819.              WITH GATTR DO
2820.              IF ACCESS ¬= DRCT THEN
2821.              BEGIN ERROR(69) ; TYPTR := NIL END ELSE
```

```
2822.              IF (BREG¬=0)&(BREG¬=LEVEL) THEN ERROR(69) ;
2823.              LATTR := GATTR ;
2824.              IF NO ¬= 20 THEN  " SY ¬= := "
2825.              BEGIN
2826.                 IF GATTR.TYPTR ¬= NIL THEN ERROR(52) ; SKIP(20);
2827.                 IF NO ¬= 20 THEN
2828.                 BEGIN
2829.                    IF GATTR.TYPTR = NIL THEN ERROR(52) ;
2830.                    IF ERRCL(NO) = BEGSY THEN GOTO 20 ; GOTO 10
2831.                 END
2832.              END ;
2833.              INSYMBOL ; EXPRESSION ;
2834.              CHTYPE ; CHTYPES ;
2835.              LOAD(GATTR);
2836.              IF NO ¬= 33 THEN  " SY ¬= TO/DOWNTO "
2837.              BEGIN
2838.                 IF GATTR.TYPTR ¬= NIL THEN ERROR(70) ; SKIP(33) ;
2839.                 IF NO ¬= 33 THEN
2840.                 BEGIN
2841.                    IF GATTR.TYPTR = NIL THEN ERROR(70) ;
2842.                    IF ERRCL(NO) = BEGSY THEN GOTO 20 ; GOTO 10
2843.                 END
2844.              END ;
2845.              LCLASS := CL ;
2846.              INSYMBOL ; EXPRESSION ;
2847.              CHTYPE ; CHTYPES ;
2848.              XTCT:=TCT;
2849.              IF (GATTR.TYPTR ¬= NIL)&(LATTR.TYPTR ¬= NIL) THEN
2850.              BEGIN
2851.                 IF GATTR.KIND=SVAL THEN
2852.                 BEGIN TATTR:=GATTR; LJPADDR:=IC; LOAD(TATTR); RP:=RP-1 END
2853.                 ELSE WITH TATTR DO
2854.                 BEGIN
2855.                   LOAD(GATTR); UPALIGN(TCT,4);
2856.                   ALIGNMENT:=4; KIND:=VARBL; TYPTR:=INTPTR;
2857.                   BREG:=LEVEL; DPLMT:=TCT; ACCESS:=DRCT;
2858.                   PCKD:=FALSE; STORE(TATTR); TCT:=TCT+4;
2859.                   IF TCT>TMAX THEN TMAX:=TCT; LOF:=FALSE;
2860.                   LJPADDR:=IC
2861.                 END;
2862.                 IF LCLASS = 1 THEN "STEP +1" GENRR(#19,STK(RP),STK(RP+1))
2863.                               ELSE "STEP -1" GENRR(#19,STK(RP+1),STK(RP));
2864.                 GENRX(#47,#2,#0,#0,#0); LCA:= IC-2;
2865.                 IF ASSCHECK THEN
2866.                 WITH LATTR.TYPTR@ DO
2867.                 IF FORM = NUMERIC THEN
2868.                 BEGIN IF LATTR.TYPTR ¬= INTPTR THEN
2869.                    CHECKBNDS(STK(RP),MIN,MAX,0)
2870.                 END ELSE CHECKBNDS(STK(RP),0,FCONST@.VALUES.IVALUE,0) ;
2871.                 STORE(LATTR);
2872.              END ;
2873.              IF NO ¬= 31 THEN  " SY ¬= DO "
2874.              BEGIN
2875.                 IF GATTR.TYPTR ¬= NIL THEN ERROR(59) ; SKIP(31) ;
2876.                 IF NO ¬= 31 THEN
2877.                 BEGIN
2878.                    IF GATTR.TYPTR = NIL THEN ERROR(59) ;
2879.                    IF ERRCL(NO) = BEGSY THEN GOTO 20 ; GOTO 10
2880.                 END
2881.              END ;
```

```
2882.              INSYMBOL ;
2883.      20:     STATEMENT ;
2884.              IF LATTR.TYPTR ¬= NIL THEN
2885.              BEGIN LOAD(LATTR);
2886.                 IF LCLASS=1 THEN
2887.                   BEGIN GENRX(#41,0,1,0,0); "LA"
2888.                         GENRR(#1A,STK(RP),0) "AR"
2889.                   END
2890.                 ELSE GENRR(#6,STK(RP),0); "BCTR"
2891.                 IF TATTR.KIND¬=SVAL THEN
2892.                   BEGIN LOAD(TATTR); RP:=RP-2 END
2893.                 ELSE RP:=RP-1;
2894.                 GENRX(#47,15,0,0,0); "B"
2895.                 INS(LJPADDR,IC-2); INS(IC,LCA)
2896.              END ;
2897.              TCT:=XTCT;
2898.              ISTKLIM:=MAXISTK;  ASTK(MAXISTK):=0;
2899.      10:  END  " FORSTAT "  ;
2900.
2901.           PROCEDURE GOTOSTAT ;
2902.           VAR XCONS: CONSTANT;
2903.           BEGIN INSYMBOL ;
2904.                 IF (NO = 1)&(AVAL = 'EXIT') THEN
2905.                 BEGIN INSYMBOL ;
2906.                    IF (NO ¬= 2)|(CL ¬= 1) THEN
2907.                    BEGIN ERROR(61) ; SKIP(49) END ELSE
2908.                    BEGIN FOR IT := CEXTABIX DOWNTO 1 DO
2909.                       IF EXTAB(IT).EXVAL = IVAL THEN
2910.                       BEGIN IT1 := EXTAB(IT).EXLEVEL ;
2911.                         IF IT1=0 THEN GENRR(#18,13,12) "LR"
2912.                         ELSE IF IT1¬=LEVEL THEN LOADBASE(13,IT1)
2913.                         ELSE ERROR(61);
2914.                         WITH XCONS DO
2915.                           BEGIN KONSTKIND:=EXTREF;
2916.                             EVALUE:=EXTAB(IT).EXTNAME
2917.                           END;
2918.                         LDCST2(XCONS,STK(1)); "GET V-CON"
2919.                         GENRS(#98,14,4,12,13); "LM"
2920.                         GENRR(#7,15,STK(1)); "BRANCH"
2921.                         GOTO 1
2922.                       END ;
2923.                       ERROR(43) ;
2924.      1:               INSYMBOL
2925.                    END
2926.                 END ELSE
2927.                 IF (NO ¬= 2)|(CL ¬= 1) THEN
2928.                 BEGIN ERROR(61) ; SKIP(49) END ELSE
2929.                 BEGIN IF IVAL >= TWOTO12 THEN ERROR(100) ;
2930.                     " SEARCH THROUGH LABELTABLE OF CURRENT BLOCK "
2931.                   GENRX(#47,15,0,0,0); "BRANCH"
2932.                    FOR IT := 1 TO CLABIX DO
2933.                    WITH LABTAB(IT) DO
2934.                    BEGIN
2935.                       IF LABVAL = IVAL THEN  " LABEL ALREADY OCCURED "
2936.                       BEGIN
2937.                         IF LABLOC¬=0 THEN INS(LABLOC,IC-2)
2938.                         ELSE IF CHAIN=0 THEN BEGIN ERROR(75); GOTO 20 END
2939.                         ELSE BEGIN
2940.                           CODE(IC-2):=CHAIN DIV 256;
2941.                           CODE(IC-1):=CHAIN MOD 256;
```

```
                        CHAIN:=IC-2
                      END;
                      GOTO 20
                    END
                  END ;
                   " LABEL NOT YET MET, ENTER IT INTO LABELTABLE "
                  IF CLABIX = MAXLABS THEN BEGIN ERROR(74) ; GOTO 20 END ;
                  CLABIX := CLABIX + 1 ;
                  WITH LABTAB(CLABIX) DO
                  BEGIN
                    LABVAL:=IVAL;
                    LABLOC:=0;
                    CHAIN:=IC-2
                  END ;
20:               INSYMBOL ;
              END ;
        END  " GOTOSTAT "  ;

        PROCEDURE WITHSTAT ;
        VAR SAVETOP,SAVETCT: ADDRESS; TATTR:ATTR;
        BEGIN
           SAVETOP:=TOP; SAVETCT:=TCT;
           UPALIGN(TCT,4);
           WITH TATTR DO
           BEGIN
             ALIGNMENT:=4; KIND:=VARBL; TYPTR:=INTPTR;
             BREG:=LEVEL; ACCESS:=DRCT; PCKD:=FALSE
           END;
           REPEAT
              INSYMBOL ;
              VARIAB ;
              WITH GATTR DO
              IF TYPTR ¬= NIL THEN
              IF TYPTR@.FORM ¬= RECORDS THEN ERROR(38) ELSE
              BEGIN
                 TOP := TOP + 1 ;
                 IF TOP > DISPLIMIT THEN ERROR(82) ELSE
                 WITH DISPLAY(TOP) DO
                 BEGIN
                    FNAME := TYPTR@.FSTFLD ;
                    IF (ACCESS = DRCT)&((BREG=0)|(BREG=LEVEL)) THEN
                    BEGIN OCCUR := CWITH;
                       CDSPL := DPLMT;  CLEV := BREG;
                    END ELSE
                    BEGIN  TATTR.DPLMT:=TCT;
                       LOADADR(GATTR); STORE(TATTR);
                       OCCUR := VWITH ; VDSPL := TCT;
                       TCT := TCT + 4 ;
                       IF TCT > TMAX THEN TMAX := TCT ;
                    END
                 END
              END
           UNTIL NO ¬= 15 ;  " SY ¬= , "
           IF NO ¬= 31 THEN
           BEGIN
              IF GATTR.TYPTR ¬= NIL THEN ERROR(59) ; SKIP(31) ;
              IF NO ¬= 31 THEN
              BEGIN
                 IF GATTR.TYPTR = NIL THEN ERROR(59) ;
                 IF ERRCL(NO) = BEGSY THEN GOTO 20 ; GOTO 10
```

```
3002.                 END
3003.               END ;
3004.               INSYMBOL ;
3005.     20:       STATEMENT ;
3006.     10:       TOP := SAVETOP; TCT := SAVETCT;
3007.           END  " WITHSTAT "  ;
3008.
3009.       PROCEDURE COMPSTAT ;
3010.       BEGIN REPEAT BEGIN
3011.             INSYMBOL ;
3012.     1:      STATEMENT ;
3013.             IF ERRCL(NO) = BEGSY THEN
3014.           BEGIN ERROR(58) ; GOTO 1 END ;
3015.           IF NO = 25 THEN  "ELSE"
3016.           BEGIN ERROR(54) ; INSYMBOL ; GOTO 1 END
3017.           END
3018.         UNTIL NO ¬= 16 ";" ;
3019.         IF NO ¬= 22 THEN "END" ERROR(68) ELSE INSYMBOL ;
3020.       END "COMPSTAT" ;
3021.
3022.       PROCEDURE STATEMENT ;
3023.          VAR LPSW,TCHAIN : SHRTINT;
3024.       BEGIN
3025.         IF (NO = 2)&(CL = 1) THEN "LABEL"
3026.         BEGIN  ISTKLIM:=MAXISTK;  ASTK(MAXISTK):=0;
3027.           IF IVAL >= TWOTO12 THEN ERROR(100) ;
3028.           FOR IT := 1 TO CLABIX DO
3029.           WITH LABTAB(IT) DO
3030.           IF LABVAL = IVAL THEN "FOUND"
3031.           BEGIN
3032.             IF LABLOC¬=0 THEN "MULTIDEF" ERROR(77)
3033.             ELSE "FIXUP"
3034.             BEGIN TCHAIN:=CHAIN;
3035.               WHILE TCHAIN¬=0 DO
3036.                 BEGIN
3037.                   IT1:=CODE(TCHAIN)*256+CODE(TCHAIN+1);
3038.                   INS(IC,TCHAIN);
3039.                   TCHAIN:=IT1
3040.                 END;
3041.               CHAIN:=TCHAIN;
3042.               LABLOC:=IC
3043.             END;
3044.             GOTO 1
3045.           END "IF, WITH, FOR" ;
3046.           "NEW LABEL"
3047.           IF CLABIX = MAXLABS THEN ERROR(74) ELSE
3048.           BEGIN CLABIX := CLABIX + 1 ;
3049.             WITH LABTAB(CLABIX) DO
3050.             BEGIN LABVAL := IVAL ; LABLOC:=IC; CHAIN:=0 END
3051.           END ;
3052.     1:    INSYMBOL ;
3053.           IF NO ¬= 19 THEN ":"
3054.           BEGIN ERROR(64) ; SKIP(49) END ELSE INSYMBOL ;
3055.       END "IF (NO=2)&(CL=1)" ;
3056.       RP := 0 ;   RRP:=0;
3057.       CASE SPLITSTAT(NO) OF
3058.       "PASS  "  1:  "ENDSY OR IRRELSY" ;
3059.       "IDENT "  2:  BEGIN SEARCH ;
3060.                       IF CTPTR = NIL THEN
3061.                       BEGIN ERROR(31) ; CTPTR := UNDECPTR END ;
```

```
                        WITH CTPTR@ DO
                        IF KLASS <= KONST THEN ERROR(55) ELSE
                        IF (KLASS = PROC)&
                        ((PROCTYPE = CTPTR)|(PROCTYPE = NIL)) THEN
                        BEGIN " PROCCALL "
                          IF PROCTYPE = CTPTR THEN
                          BEGIN INSYMBOL ; IF CTPTR@.PREDEF THEN
                            BEGIN LPSW := CTPTR@.SEGSIZE ;
                             CASE LPSW OF
                               0:     TITLE ;
                         1,2,3,4:     GETPUT(LPSW) ;
                               5:     ALLC ;
                               6:     PCK ;
                               7:     UNPCK ;
                             8,9:     INSAPP(LPSW) ;
                              10:     READIR ;
                              11:     WRITEIR ;
                                END ;
                                END ELSE  PASSPARAMS ;
                          END ELSE SKIP(49)
                        END "PROCCALL" ELSE  ASSIGN ;
                      END ;
          "BEGIN "  3:  COMPSTAT ;
          "IF    "  4:  IFSTAT ;
          "CASE  "  5:  CASESTAT ;
          "REPEAT"  6:  REPEATSTAT ;
          "WHILE "  7:  WHILESTAT ;
          "FOR   "  8:  FORSTAT ;
          "GOTO  "  9:  GOTOSTAT ;
          "WITH  " 10:  WITHSTAT ;
          END ;
          IF ERRCL(NO) = IRRELSY THEN
          BEGIN ERROR(54) ; SKIP(49) END ;
          RP := 0;  RRP:=0
        END "STATEMENT" ;

         PROCEDURE SETSDNAME;
         VAR I:INTEGER; A:ARRAY(1..10) OF CHAR;
         BEGIN
           IF UNIQINDEX=0 THEN
           BEGIN
             UNPACK(GLOBALNAME,A,1);
             A(8):=' '; A(9):=' '; A(10):=' ';
             I:=7;
             WHILE A(I)=' ' DO
             BEGIN A(I):='$'; I:=I-1 END
           END
           ELSE
           BEGIN
             UNPACK(UNIQUENAME,A,1);
             A(8):=UNIQCH(UNIQINDEX MOD 55);
             I:=UNIQINDEX DIV 55;
             IF I¬=0 THEN
             BEGIN
               A(7):=UNIQCH(I MOD 19);
               I:=I DIV 19;
               IF I¬=0 THEN A(6):=UNIQCH(I)
             END
           END;
           PACK(A,1,UNIQUENAME);
```

```
3122.          UNIQINDEX:=UNIQINDEX+1
3123.        END "SETSDNAME";
3124.
3125.        PROCEDURE TYPEDECL;
3126.          "RETURNS  TL = SIZE OF ITEM, P1 POINTS TO TYPE RECORD"
3127.        VAR I,L,LL,J,CV,E1,E2,BDISPL : INTEGER;
3128.           OPT : OPTPWR;  DISPL : ADDRESS;  PACKFLAG,LERR : BOOLEAN;
3129.           LASTFLD,PP,P,NXTF,NXTC,NXTA,RTYP : CTP;
3130.           LALIGN:INTEGER;
3131.
3132.           PROCEDURE SKIPT(FNO : INTEGER) ;
3133.           BEGIN
3134.              WHILE (TERRCL(NO) = IRRELSY) & (FNO ¬= NO) DO INSYMBOL;
3135.           END "SKIPT";
3136.
3137.           PROCEDURE TYPERR(I : INTEGER);
3138.           BEGIN TL := 0;  P1 := NIL;
3139.              ERROR(I);  SKIPT(49);
3140.           END "TYPERR";
3141.
3142.           PROCEDURE SUBRANGE(VAR VAL1,VAL2 : INTEGER; N1 : CTP;
3143.              CONST P : CTP) ;
3144.              "THE FIRST SYMBOL OF SUBRANGE HAS BEEN READ.
3145.                 THE PROCEDURE RETURNS THE TWO BOUND-VALUES IN VAL1,VAL2,
3146.                 AND RETURNS  N1 = POINTER TO TYPE OF CONSTANTS.
3147.                 ERRORS : TYPES DO NOT AGREE, TYPE IS NOT INTEGER,CHAR,
3148.                    OR SYMBOLIC,  VAL1 > VAL2.
3149.                 P INDICATES BEGINNING OF SEARCH FOR FIRST SYMBOL IF IT IS
3150.                 AN ID, SINCE SEARCH OFTEN ALREADY PERFORMED BY CALLER."
3151.              VAR N2 : CTP; CKIND: CONSTKIND;
3152.           BEGIN INCONST(CKIND,N1,P);
3153.              IF (N1=NIL) | (N1=REALPTR) | (N1=ALFAPTR) THEN
3154.                 ERR := TRUE ELSE
3155.              BEGIN  VAL1 := IVAL;  INSYMBOL;
3156.                 IF NO ¬= 19 ":" THEN ERROR(10) ELSE INSYMBOL ;
3157.                 INCONST(CKIND,N2,NEXT);
3158.                 IF N1 ¬= N2 THEN ERR := TRUE ELSE
3159.                 BEGIN  VAL2 := IVAL;
3160.                    IF VAL1 > VAL2 THEN ERROR(25);
3161.                 END "N1 = N2";
3162.                 INSYMBOL;
3163.              END;
3164.           END "SUBRANGE";
3165.
3166.           PROCEDURE SUBTYPE(VAR I,J : INTEGER; P : CTP);
3167.              "EITHER A TYPE-ID FOR A SUBRANGE OR AN EXPLICIT SUBRANGE
3168.                 ARE PROCESSED.
3169.                 RETURNS I = LOWBOUND, J = HIGHBOUND,
3170.                    P = POINTER TO TYPE OF CONSTANTS  "
3171.           BEGIN
3172.              IF NO = 1 THEN  "IDENTIFIER"
3173.              BEGIN SRCHREC(NEXT);
3174.                 IF CTPTR = NIL THEN SEARCH;
3175.                 IF CTPTR = NIL THEN ERROR(12) ELSE
3176.                 BEGIN IF CTPTR@.KLASS = TYPES THEN
3177.                    BEGIN IF CTPTR@.FORM > SYMBOLIC THEN ERROR(13) ELSE
3178.                       BEGIN
3179.                         CASE CTPTR@.FORM OF
3180.        NUMERIC:                BEGIN I := CTPTR@.MIN;  J := CTPTR@.MAX;
3181.                                   P := INTPTR;
```

```
                        END;
SYMBOLIC:               BEGIN IF (CTPTR = REALPTR)|(CTPTR = ALFAPTR) THEN
                          ERROR(6) ELSE
                          BEGIN I := 0 ;
                            J:=CTPTR@.FCONST@.VALUES.IVALUE;
                            P := CTPTR ;
                          END
                        END;
                      END "CASE";
                      INSYMBOL;
                    END;
                END "TYPES" ELSE
                IF CTPTR@.KLASS = KONST THEN
                   SUBRANGE(I,J,P,CTPTR) ELSE ERROR(63) ;
             END;
          END "ID" ELSE
          IF (NO=2)|(NO=7)
        "IF NO IN SET(2,7)" "CONST +-|"  THEN SUBRANGE(I,J,P,NIL) ELSE
             ERROR(1) ;
       END "SUBTYPE";

       PROCEDURE SCALDECL(N : CTP);
       BEGIN SUBRANGE(I,J,PP,N);
          IF ¬ERR THEN
          IF PP@.FORM = SYMBOLIC THEN
          BEGIN ERROR(28);  P1 := NIL  END ELSE
          BEGIN ALLOC(P,TYPES,NUMERIC);
             WITH P@ DO
             BEGIN NAME := BLANK;  NXTEL := NIL;  KLASS := TYPES;
                FORM := NUMERIC;  MIN := I;  MAX := J;
                IF ABS(I) > ABS(J) THEN BITS := LOG2(ABS(I)) ELSE
                                        BITS := LOG2(ABS(J)) ;
                IF BITS <= 16 THEN SIZE := 2  ELSE SIZE := 4;
                ALIGN := SIZE;
             END;
             TL := P@.SIZE;  P1 := P;
          END ELSE P1 := NIL
       END "SCALDECL";

    PROCEDURE FIELDLIST(VAR MAXSIZE:INTEGER; VARPTR,NXTF:CTP);
    "RETURNS MAXSIZE OF FIELD, VARPTR - POINTS TO CASE VARIABLE,
       AND NXTF - POINTS TO THE (CHAINED) FIELDS "
       VAR MXL,L,LL,MINSIZE,CASEBITS,I,J : INTEGER;
          P,PP,PP1,PP2,NXTC,NXT,CPTR : CTP;
          TAGFLAG : BOOLEAN;

       PROCEDURE RECERR(I : INTEGER);
       BEGIN ERROR(I);  SKIPT(49); END;

       PROCEDURE ADJUST;
          " MOVE LAST FIELD TO RIGHT. IF IT IS THE ONLY FIELD,
             CHANGE TO NONPACKED.
             IF LAST FIELD IS TAGFIELD THEN DO NOT MOVE.
             INCREASE DISPL, RESET BDISPL  "
       BEGIN IF ¬TAGFLAG THEN
          WITH LASTFLD@ DO
          BEGIN IF BITDISPL = 0 THEN "ONLY ONE FIELD IN WORD"
                BITWIDTH := 0 ELSE
                BITDISPL := WORDLENGTH - BITWIDTH;
          END;
```

```
3242.              DISPL := DISPL + 1;  BDISPL := 0;
3243.            END "ADJUST"; "MAY REQUIRE MODIFICATIONS LATER"
3244.
3245.          BEGIN   "FIELDLIST"
3246.            TAGFLAG := TRUE;    NXT := NXTF;
3247.            REPEAT   "UNTIL (TERRCL(NO) = ENDSY) & (NO ¬= 26)"
3248.              IF NO = 26 THEN "CASE" GOTO 2 ;
3249.              I := 0;
3250.        1:      IF NO ¬= 1 THEN
3251.                BEGIN RECERR(11);
3252.                  IF TERRCL(NO) = BEGSY THEN GOTO 11;  GOTO 12;
3253.                END;
3254.                SRCHREC(NXT);
3255.                IF CTPTR ¬= NIL THEN ERROR(5) ELSE
3256.                BEGIN ALLOC(P,FIELD); I := I + 1;
3257.                  WITH P@ DO
3258.                  BEGIN NAME := AVAL;  NXTEL := NXT;  KLASS := FIELD;
3259.                    FLDTYPE := NIL;
3260.                  END;  NXT := P;
3261.                END;
3262.                INSYMBOL;
3263.                IF NO = 15 THEN ","
3264.                BEGIN INSYMBOL;  GOTO 1 END;
3265.                IF NO ¬= 19 THEN "NOT :" ERROR(10) ELSE INSYMBOL;
3266.        11:     TYPEDECL(L,P);
3267.                IF P ¬= NIL THEN
3268.                IF P@.FORM > RECORDS THEN ERROR(30) ELSE
3269.                BEGIN IF PACKFLAG THEN "NO PACKED RECORDS AT THIS TIME"
3270.                  BEGIN"IF I > 1 THEN ""REVERSE POINTERS""
3271.                    BEGIN PP := NXT;
3272.                      FOR I := I DOWNTO 1 DO
3273.                      BEGIN PP1 := PP@.NXTEL;
3274.                        PP@.NXTEL := PP2;  PP2 := PP;
3275.                        PP := PP1;
3276.                      END;
3277.                      NXT@.NXTEL := PP;  NXT := PP2;
3278.                    END ""REVERSE"" ELSE PP := NXT@.NXTEL;
3279.                    PP1 := NXT;
3280.                    IF P@.FORM <= POWER THEN
3281.                    BEGIN
3282.                      CASE P@.FORM OF
3283.          NUMERIC:      LL := P@.BITS;
3284.          SYMBOLIC:     LL := P@.BITSIZE;
3285.          POINTER:      LL := 18 ;
3286.          POWER:        LL := P@.PWBITS;
3287.                      END;
3288.                      REPEAT
3289.                        IF BDISPL + LL > WORDLENGTH THEN ADJUST;
3290.                        IF LL = WORDLENGTH THEN
3291.                        BEGIN PP1@.BITWIDTH := 0;
3292.                          PP1@.FLDADDR := DISPL;  DISPL := DISPL + 1;
3293.                        END ELSE
3294.                        BEGIN PP1@.BITWIDTH := LL;
3295.                          PP1@.BITDISPL := BDISPL;
3296.                          PP1@.FLDADDR := DISPL;  BDISPL := BDISPL + LL;
3297.                        END;
3298.                        PP1@.FLDTYPE := P;
3299.                        LASTFLD := PP1;  TAGFLAG := FALSE;
3300.                        PP1 := PP1@.NXTEL;
3301.                      UNTIL PP1 = PP;
```

```
3302.                    END ""FORM <= POINTER"" ELSE
3303.                    BEGIN IF BDISPL ¬= 0 THEN ADJUST;
3304.                      TAGFLAG := FALSE;
3305.                      REPEAT
3306.                      PP1@.BITWIDTH := 0;  PP1@.FLDTYPE := P;
3307.                        PP1@.FLDADDR := DISPL;
3308.                        DISPL := DISPL + L;  LASTFLD := PP1;
3309.                        PP1 := PP1@.NXTEL;
3310.                      UNTIL PP1 = PP;
3311.                    END;  "
3312.                 END "PACKFLAG" ELSE
3313.                 BEGIN  LL := P@.ALIGN;
3314.                    IF LL > LALIGN THEN LALIGN := LL;
3315.                    UPALIGN(DISPL,LL);  UPALIGN(L,LL);
3316.                    LL := DISPL + I*L;  DISPL := LL;
3317.                    PP := NXT;
3318.                    FOR I := I DOWNTO 1 DO
3319.                    BEGIN PP@.FLDTYPE := P;  LL := LL - L;
3320.                      PP@.FLDADDR := LL;  PP@.BITWIDTH := 0;
3321.                      PP@.ALIGN:=P@.ALIGN;  PP := PP@.NXTEL;
3322.                    END;
3323.                 END;
3324.              END "FORM <= RECORDS";
3325.     12:      IF NO = 16 ";" THEN INSYMBOL;
3326.           UNTIL (TERRCL(NO) = ENDSY) & (NO ¬= 26 "CASE" );
3327.           IF BDISPL ¬= 0 THEN ADJUST;
3328.           MAXSIZE := DISPL;  VARPTR := NIL;
3329.           GOTO 9;
3330.     2:    "CASE"
3331.           INSYMBOL;
3332.           IF NO ¬= 1 THEN ERROR(11) ELSE
3333.           BEGIN SRCHREC(NXT);
3334.              IF CTPTR ¬= NIL THEN ERROR(5) ELSE
3335.              BEGIN ALLOC(P,FIELD);
3336.                 WITH P@ DO
3337.                 BEGIN NAME := AVAL;  NXTEL := NXT;  KLASS := FIELD;
3338.                    FLDTYPE := NIL;
3339.                 END;  NXT := P;
3340.              END;
3341.              INSYMBOL;
3342.           END "NO = 1";
3343.              IF NO ¬= 19 ":" THEN ERROR(10) ELSE INSYMBOL;
3344.              IF NO ¬= 1 THEN ERROR(11) ELSE
3345.              BEGIN SRCHREC(NEXT);
3346.                 IF CTPTR = NIL THEN SEARCH;
3347.                 IF CTPTR = NIL THEN ERROR(12) ELSE
3348.                 IF (CTPTR@.KLASS¬=TYPES) | (CTPTR@.FORM>SYMBOLIC) THEN
3349.                    "(¬NUMERIC) & (¬SYMBOLIC)"  ERROR(7);
3350.                 INSYMBOL;
3351.                    IF NO ¬= 27 "OF" THEN ERROR(14);
3352.                    CPTR := CTPTR;
3353.                    IF CPTR ¬= NIL THEN
3354.                    IF PACKFLAG THEN "NO PACKED RECORDS AT THIS TIME"
3355.                    BEGIN "
3356.                         CASE CPTR@.FORM OF
3357.           NUMERIC:                LL := CPTR@.BITS;
3358.           SYMBOLIC:               LL := CPTR@.BITSIZE;
3359.                         END;
3360.                         IF BDISPL + LL > WORDLENGTH THEN ADJUST;
3361.                         P@.BITDISPL := BDISPL;  P@.BITWIDTH := LL;
```

```
3362.                     P@.FLDADDR := DISPL;  BDISPL := BDISPL + LL;
3363.                   LASTFLD := P;  CASEBITS := BDISPL; "
3364.                 END "PACKFLAG" ELSE
3365.                 BEGIN IF CPTR@.ALIGN > LALIGN THEN LALIGN := CPTR@.ALIGN;
3366.                   UPALIGN(DISPL,CPTR@.ALIGN);
3367.                   P@.FLDADDR := DISPL;
3368.                   P@.BITWIDTH := 0;
3369.                   P@.ALIGN:=CPTR@.ALIGN;
3370.                 END;
3371.                 P@.FLDTYPE := CPTR;  DISPL := DISPL + CPTR@.SIZE;
3372.                 MINSIZE := DISPL;  MAXSIZE := DISPL;
3373.                 NXTC := NIL;  INSYMBOL;
3374.                 REPEAT  I := 0;
3375.                   REPEAT
3376.                   " IF (NO>2) | (NO=2) & ¬(CL IN SET(1,4)) THEN"
3377.                     IF (NO>2) | (NO=2) & ¬((CL=1)|(CL=4)) THEN
3378.                       "HAS TO BE NUMERIC OR CHAR CONSTANT OR ID"
3379.                     BEGIN RECERR(63);  GOTO 3 END ;
3380.                     IF CPTR ¬= NIL THEN
3381.                       IF NO = 1 THEN
3382.                       BEGIN SRCHREC(NEXT);
3383.                         IF CTPTR = NIL THEN SEARCH;
3384.                         IF CTPTR = NIL THEN ERROR(12) ELSE
3385.                         WITH CTPTR@ DO
3386.                           IF KLASS ¬= KONST THEN ERROR(63) ELSE
3387.                           IF (CPTR@.FORM = SYMBOLIC) &
3388.                               (CONTYPE ¬= CPTR)  |
3389.                             (CPTR@.FORM = NUMERIC) &
3390.                               (CONTYPE ¬= INTPTR) THEN ERROR(73) ELSE
3391.                           IT := VALUES.IVALUE;
3392.                       END "NO = 1" ELSE
3393.                       IF (CL = 1) & (CPTR@.FORM ¬= NUMERIC)
3394.                         | (CL = 4) & (CPTR ¬= CHARPTR) THEN
3395.                           ERROR(73) ELSE
3396.                       IT := IVAL;  "END CPTR ¬= NIL"
3397.                       ALLOC(P,TAGFIELD);
3398.                       WITH P@ DO
3399.                       BEGIN NAME := BLANK;  NXTEL := NXTC;
3400.                         KLASS := TAGFIELD;
3401.                         TAGVAL := TRUE;  CASEVAL := IT;
3402.                       END;  NXTC := P;  I := I + 1;
3403.                       INSYMBOL;
3404.           3:          IF NO ¬= 19 THEN ERROR(10) ELSE INSYMBOL;
3405.                 UNTIL NO > 2;
3406.                 IF NO = 9 THEN  " ( FIELDLIST ) "
3407.                 BEGIN IF PACKFLAG THEN "NO PACKED RECORDS NOW"
3408.                   BEGIN "DISPL := MINSIZE - 1;
3409.                     BDISPL := CASEBITS;  "
3410.                   END ELSE DISPL := MINSIZE;
3411.                   INSYMBOL;  FIELDLIST(MXL,PP,NXT);
3412.                   IF NO ¬= 10 THEN ERROR(9) ELSE INSYMBOL;
3413.                 END ELSE
3414.                 BEGIN PP := NIL;  MXL := MINSIZE END;
3415.                 P := NXTC;
3416.                 FOR I := I DOWNTO 1 DO
3417.                 WITH P@ DO
3418.                 BEGIN CASESIZE := MXL;  VARIANTS := PP;
3419.                   P := NXTEL;
3420.                 END;  IF MAXSIZE < MXL THEN MAXSIZE := MXL;
3421.                 IF NO = 16 THEN INSYMBOL;
```

```
                    UNTIL NO > 2;
                    ALLOC(P,TAGFIELD);
                    WITH P@ DO
                    BEGIN NAME := BLANK;  NXTEL := NIL;  KLASS := TAGFIELD;
                       CASESIZE := MINSIZE;  VARIANTS := NXTC;
                       TAGVAL := FALSE;  CASETYPE := CPTR;
                    END;
                    VARPTR := P;
              END "NO = 1";
        9:    NXTF := NXT;
          END " FIELDLIST ";

          "TYPEDECL"
          BEGIN PACKFLAG := FALSE;  LALIGN := 0;
             IF NO = 42 THEN  "PACKED" "NO PACKED RECORDS AT THIS TIME"
             BEGIN PACKFLAG := "TRUE"  FALSE;  INSYMBOL END;
             IF NO = 1 THEN
             BEGIN SRCHREC(NEXT);
                IF CTPTR = NIL THEN SEARCH;
                IF CTPTR = NIL THEN
                BEGIN ERROR(12);  P1 := NIL;  SKIPT(16) END ELSE
                BEGIN IF CTPTR@.KLASS = TYPES THEN "TYPE-ID"
                   BEGIN TL := CTPTR@.SIZE;  P1 := CTPTR;  INSYMBOL END ELSE
                   IF CTPTR@.KLASS = KONST THEN SCALDECL(CTPTR) ELSE
                   TYPERR(11);
                END;
             END "ID" ELSE
             IF NO = 9 THEN  " SYMBOLIC "
             BEGIN CV := 0; LERR := ERR ; ERR := FALSE;ALLOC(P,TYPES,SYMBOLIC);
                WITH P@ DO
                BEGIN NAME := BLANK;  NXTEL := NIL;  KLASS := TYPES;
                   FORM := SYMBOLIC;
                END;  RTYP := P;  NXTC := NIL;
                REPEAT INSYMBOL;
                   IF NO ¬= 1 THEN
                   BEGIN ERROR(11);  SKIPT(15);
                      GOTO 2;
                   END;
                   SRCHREC(NEXT);
                      IF CTPTR ¬= NIL THEN ERROR(8) ELSE
                      BEGIN ALLOC(P,KONST,ACTUAL);
                         WITH P@ DO
                         BEGIN NAME := AVAL;  NXTEL := NEXT;  KLASS := KONST;
                            CONTYPE := RTYP;  CONKIND := ACTUAL;
                            VALUES.KONSTKIND:=SYMBOLICS;
                            VALUES.IVALUE:=CV;  SUCC:=NXTC;
                         END;  CV := CV + 1;  NEXT := P;  NXTC := P;
                      END;
                      INSYMBOL;
        2:         UNTIL NO ¬= 15;
                  ALLOC(P,TYPES,POWER) ;
                   WITH P@ DO
                   BEGIN NAME := BLANK;  NXTEL := NIL;  KLASS := TYPES;
                      SIZE:=4;  ALIGN:=4;  "CHANGE HERE FOR DIFF SIZE PWSET"
                      FORM := POWER;  ELSET := RTYP;  PWBITS := CV + 1;
                   END;
                   WITH RTYP@ DO
                   BEGIN FCONST := NEXT;
                      BITSIZE := LOG2(CV - 1);  PWSET := P;
                      IF CV-1 < MAXHALF THEN SIZE := 2  ELSE SIZE := 4;
```

```
3482.                     ALIGN := SIZE;
3483.                  END;
3484.               IF ERR THEN P1 := NIL ELSE
3485.               BEGIN ERR := LERR ; P1 := RTYP END ;
3486.               TL := RTYP@.SIZE ;
3487.               IF NO ¬= 10 THEN TYPERR(9) ELSE INSYMBOL ;
3488.         END  " SYMBOLIC "  ELSE
3489.        "IF NO IN SET(2,7) THEN"" SUBRANGE "
3490.         IF (NO=2)|(NO=7) THEN " SUBRANGE "
3491.         BEGIN LERR := ERR; ERR := FALSE;
3492.            SCALDECL(NEXT);
3493.            IF ERR THEN ERROR(6) ELSE ERR := LERR
3494.         END "SUBRANGE" ELSE
3495.         IF NO = 38 THEN   " STRUCTURED TYPES "
3496.         CASE CL OF
3497.      "ARRAY" 1:
3498.         BEGIN INSYMBOL;
3499.            IF NO ¬= 9 "(" THEN
3500.            BEGIN TYPERR(98);
3501.               IF TERRCL(NO) = BEGSY THEN TYPEDECL(TL,CTPTR); GOTO 19;
3502.            END;
3503.               NXTA := NIL;
3504.               REPEAT ALLOC(P,TYPES,ARRAYS);
3505.                  WITH P@ DO
3506.                  BEGIN NAME := BLANK;  NXTEL := NIL;  KLASS := TYPES;
3507.                     FORM := ARRAYS;  AELTYPE := NXTA;
3508.                     " AELTYPE TEMPORARILY LINKS SUBARRAYS "
3509.                  END;  NXTA := P;
3510.                  INSYMBOL;
3511.                  LERR := ERR;  ERR := FALSE;
3512.                  SUBTYPE(I,J,P);
3513.                  IF ERR THEN
3514.                  BEGIN ERROR(6);  SKIPT(15); I := 0; J := 0; P := NIL
3515.                  END ELSE
3516.                  ERR := LERR;
3517.                  WITH NXTA@ DO
3518.                  BEGIN LO := I;  HI := J;  INXTYPE := P END;
3519.               UNTIL NO ¬= 15;
3520.               IF NO ¬= 10 ")" THEN
3521.               BEGIN ERROR(37); SKIPT(27);
3522.                  IF TERRCL(NO) = BEGSY THEN GOTO 11;
3523.                  IF NO = 27 THEN
3524.                  BEGIN INSYMBOL; GOTO 11 END ;
3525.                  IF NO ¬= 10 ")" THEN
3526.                  BEGIN P1 := NIL ; TL := 0; GOTO 19 END;
3527.               END;
3528.               INSYMBOL;
3529.                  IF NO ¬= 27 THEN ERROR(14) ELSE INSYMBOL;
3530.      11:         TYPEDECL(TL,CTPTR);
3531.                  IF CTPTR ¬= NIL THEN
3532.                  IF CTPTR@.FORM > RECORDS THEN
3533.                  BEGIN ERROR(30); CTPTR := NIL END ELSE
3534.                  BEGIN  UPALIGN(TL,CTPTR@.ALIGN);
3535.                  "CHANGE HERE FOR PACKED ARRAYS"
3536.                     REPEAT
3537.                       WITH NXTA@ DO
3538.                       BEGIN MULOPT(TL,E1,E2,OPT);
3539.                         OPTTYP := OPT;  EXP1 := E1;  EXP2 := E2;
3540.                         TL := TL*(HI - LO + 1);
3541.                         SIZE := TL;
```

```
                        P := AELTYPE;  AELTYPE := CTPTR;
                        ALIGN := CTPTR@.ALIGN;
                      END;
                      CTPTR := NXTA;  NXTA := P;
                    UNTIL NXTA = NIL;
                    " NOW TL = SIZE OF ARRAY, CTPTR POINTS TO IT "
                END;  P1 := CTPTR;
19:  END "ARRAY";
"RECORD" 2:
     BEGIN ALLOC(P,TYPES,RECORDS);
        WITH P@ DO
        BEGIN NAME := BLANK;  NXTEL := NIL;  KLASS := TYPES;
           FORM := RECORDS;
        END;  RTYP := P;
        INSYMBOL;  NXTF := NIL;
        DISPL := 0;  BDISPL := 0;  LERR := ERR;  ERR := FALSE;
        FIELDLIST(TL,P,NXTF);
        IF NO ¬= 22 THEN ERROR(17);
        IF ERR THEN TYPERR(18) ELSE
        WITH RTYP@ DO
        BEGIN SIZE := TL;  FSTFLD := NXTF;  RECVAR := P;
           P1 := RTYP;  ERR := LERR;  INSYMBOL;
           ALIGN := LALIGN;
        END;
     END "RECORD";
"FILE" 3:
     BEGIN INSYMBOL;
        IF NO ¬= 27 THEN ERROR(14) ELSE INSYMBOL;
        TYPEDECL(TL,CTPTR);
        IF CTPTR ¬= NIL THEN
        IF CTPTR@.FORM > RECORDS THEN
        BEGIN ERROR(30); P1 := NIL END ELSE
        BEGIN ALLOC(P,TYPES,FILES);
           WITH P@ DO
           BEGIN NAME := BLANK;  NXTEL := NIL;  KLASS := TYPES;
              FORM:=FILES;  SIZE:=CTPTR@.SIZE+16;  FELTYPE:=CTPTR;
              IF CTPTR@.ALIGN<=4 THEN ALIGN:=4
              ELSE ALIGN:=CTPTR@.ALIGN;
              TL:=SIZE;
           END;
           P1 := P;
        END ELSE P1 := NIL;
     END "FILE";
"CLASS" 4:
     BEGIN ALLOC(P,TYPES,CLASSS);
        WITH P@ DO
        BEGIN NAME := BLANK;  NXTEL := NIL;  KLASS := TYPES;
           FORM := CLASSS;
        END;
        INSYMBOL ;
        IF (NO = 2)&(CL = 1) THEN
        BEGIN I := IVAL ; INSYMBOL END ELSE
           I := 100 ; "DEFAULT CLASS SIZE"
           IF NO ¬= 27 THEN ERROR(14) ELSE INSYMBOL;
           TYPEDECL(TL,CTPTR);
           IF CTPTR ¬= NIL THEN
              IF CTPTR@.FORM > RECORDS THEN
              BEGIN ERROR(30); P1 := NIL END
              ELSE WITH P@ DO
              BEGIN  UPALIGN(TL,CTPTR@.ALIGN);
```

```
                    "CHANGE HERE FOR PACKED CLASSES"
                    TL := TL*I+8; "8-BYTE CLASS DESCRIPTOR"
                    SIZE := TL;  PELTYPE := CTPTR;
                    IF CTPTR@.ALIGN<4 THEN ALIGN:=4
                    ELSE ALIGN:=CTPTR@.ALIGN;
                    P1 := P;
                 END
              ELSE P1:=NIL;
        END "CLASS";
     "POWERSET" 5:
        BEGIN INSYMBOL;  LERR := ERR;  ERR := FALSE;
           SUBTYPE(I,J,P);
           IF ERR THEN TYPERR(6) ELSE
           BEGIN ERR := LERR;
              CASE P@.FORM OF
  NUMERIC:       IF (I < 0) | (J > WORDLENGTH - 2) THEN
                 BEGIN ERROR(6);  P1 := NIL END ELSE
                    P1 := PNUMPTR;
  SYMBOLIC:      BEGIN IF P = CHARPTR THEN J := WORDLENGTH - 2  ; "??????"
                    IF J > WORDLENGTH - 2 THEN
                    BEGIN ERROR(6); P1 := NIL END ELSE P1 := P@.PWSET;
                 END;
              END "CASE";
              TL := 4;  "CHANGE FOR DIFFERENT SIZE PWRSET"
              IF ¬ERR THEN P1@.SIZE:=4;
           END "¬ERR";
        END "POWERSET";

        END "STRUCTURED TYPES" ELSE
        IF NO = 18 THEN "POINTER"
        BEGIN INSYMBOL;
           IF NO ¬= 1 THEN TYPERR(11) ELSE
           BEGIN TL := 4;  ALLOC(P,TYPES,POINTER);
              WITH P@ DO
              BEGIN NAME := BLANK;  NXTEL := NIL;  KLASS := TYPES;
                 FORM := POINTER;  SIZE := 4;  ALIGN := 4;
              END;
              SRCHREC(NEXT);
              IF CTPTR = NIL THEN SEARCH;
              IF CTPTR ¬= NIL THEN
              IF CTPTR@.VTYPE = NIL THEN CTPTR := NIL;
              IF CTPTR = NIL THEN "UNDECLARED CLASS"
              IF PTX > PTLIMIT THEN
              BEGIN ERROR(92); P1 := NIL; INSYMBOL END ELSE
              WITH PTLIST(PTX) , P@ DO
              BEGIN HNAME := AVAL;  PPTR := P;
                 DOMAIN := P;  ELTYPE := P;
                 PTX := PTX + 1;  P1 := P;  INSYMBOL;
              END ELSE
              WITH CTPTR@ DO
              IF (KLASS = VARS) & (VTYPE@.FORM = CLASSS) THEN
              BEGIN P@.DOMAIN := CTPTR;
                 P@.ELTYPE := VTYPE@.PELTYPE;
                 P1 := P;  INSYMBOL;
              END ELSE TYPERR(15);
           END;
        END "POINTER" ELSE TYPERR(18);
     END "TYPEDECL";

     PROCEDURE BODY;
```

```
3662.          "SURRPTR - SURROUNDING PROCEDURE, FIRSTENTRY - DUMMY USED
3663.             TO RESET CONTESTTABLE AND MAKE LOCAL VARIABLES DISAPPEAR"
3664.
3665.       LABEL 1;
3666.       VAR I,J : INTEGER;
3667.          OLDLEV : RG3;
3668.          LC1,TL,LL,FSTIX,LCA,LCP : INTEGER ;
3669.          N,P,PROCPTR,LFIRSTENTRY,SPTR : CTP;
3670.          TYPID : ALFA ; LCH : CWORD ;  KKIND: CONSTKIND;
3671.          VLC,MLC:ADDRESS;
3672.
3673.          PROCEDURE FINDSEMICOLON;
3674.          BEGIN IF NO ¬= 16 THEN
3675.             BEGIN ERROR(58);  SKIP(16);
3676.                IF NO ¬= 16 THEN GOTO EXIT 1;
3677.             END;  INSYMBOL;
3678.          END "FINDSEMICOLON";
3679.
3680.       PROCEDURE WRITVAL;
3681.       VAR VLEN,IT,IT1:INTEGER; VTYP,ETYP:CTP;
3682.           I,ESDID,J,LIC,AMT:SHRTINT; K:INTEGER;
3683.
3684.         PROCEDURE STCONS;
3685.         VAR I:SHRTINT; ATEMP: ARRAY(1..10) OF CHAR; IX:INTEGER;
3686.         BEGIN FOR I:=1 TO 10 DO ATEMP(I):=' ';
3687.           CASE KKIND OF
3688.           INTEGERS, SYMBOLICS, CHARS:
3689.             IF VLEN=10 THEN ATEMP(1):=CHR(IVAL)
3690.             ELSE BEGIN IX:=IVAL;
3691.               FOR I:=VLEN DOWNTO 1 DO
3692.               BEGIN ATEMP(I):=CHR(IX MOD 256);
3693.                 IX:=IX DIV 256
3694.               END
3695.             END;
3696.           REALS: BEGIN END; "HELP - WHAT DO WE DO???"
3697.           ALFAS: UNPACK(AVAL,ATEMP,1);
3698.           END; "CASE"
3699.           FOR I:=0 TO VLEN-1 DO CODE(IC+I):=INT(ATEMP(I+1));
3700.           IC:=IC+VLEN
3701.         END "STCONS";
3702.
3703.       BEGIN
3704.         VDATA:=TRUE;
3705.         PASCALGO@:=BLANKCARD; PASCALGO@(1):=CHR(2);
3706.         "ONLY ESD ITEM IS CSECT NAME"
3707.         UNPACK('ESD',PASCALGO@,2);
3708.         PASCALGO@(11):=CHR(0); PASCALGO@(12):=CHR(16);
3709.         PASCALGO@(15):=CHR(0); PASCALGO@(16):=CHR(1);
3710.         UNPACK('$GBLDAT',PASCALGO@,17);
3711.         FOR I:=25 TO 28 DO PASCALGO@(I):=CHR(0);
3712.         FOR I:=30 TO 32 DO PASCALGO@(I):=CHR(0);
3713.         PUT(PASCALGO);
3714.         INSYMBOL;
3715.         WHILE NO=1 DO
3716.         BEGIN
3717.           SRCHREC(NEXT);
3718.           IF CTPTR=NIL THEN SEARCH;
3719.           IF CTPTR=NIL THEN BEGIN ERROR(12); FINDSEMICOLON; GOTO 20 END;
3720.           IF CTPTR@.KLASS¬=VARS THEN
3721.             BEGIN ERROR(26); FINDSEMICOLON; GOTO 20 END;
```

```
3722.              IT:=CTPTR@.VADDR; IC:=0;
3723.              IF IT<MLC THEN MLC:=IT;
3724.              VTYP:=CTPTR@.VTYPE;  VLEN:=VTYP@.SIZE;
3725.              INSYMBOL;
3726.              IF(NO¬=8)|(CL¬=6) THEN ERROR(4) ELSE INSYMBOL;
3727.              IF NO=9 "(" THEN "LIST"
3728.              BEGIN
3729.                IF VTYP@.FORM=ARRAYS THEN
3730.                BEGIN ETYP:=VTYP@.AELTYPE; VLEN:=ETYP@.SIZE END
3731.                ELSE ERROR(27);
3732.                REPEAT INSYMBOL;
3733.                  INCONST(KKIND,PT,NEXT);
3734.                  INSYMBOL;
3735.                  IT1:=1;
3736.                  IF (NO=6)&(CL=1) THEN "*"
3737.                  BEGIN
3738.                    IF KKIND=INTEGERS THEN IT1:=IVAL ELSE ;
3739.                    INSYMBOL;
3740.                    INCONST(KKIND,PT,NEXT);
3741.                    INSYMBOL;
3742.                  END;
3743.                  IF ¬ERR THEN
3744.                  BEGIN
3745.                    STCONS;
3746.                    FOR J:=0 TO IT1-2 DO "0 TIMES IF IT1=1"
3747.                    BEGIN
3748.                      FOR I:=0 TO VLEN-1 DO CODE(IC+I):=CODE(IC-VLEN+I);
3749.                      IC:=IC+VLEN
3750.                    END
3751.                  END;
3752.                  IF (NO¬=15)&(NO¬=10) THEN
3753.                    BEGIN ERROR(20); FINDSEMICOLON; GOTO 20 END;
3754.                UNTIL NO=10;
3755.                INSYMBOL
3756.              END "NO=9"
3757.              ELSE
3758.              BEGIN
3759.                INCONST(KKIND,PT,NEXT);
3760.                INSYMBOL;
3761.                IF ¬ERR THEN STCONS
3762.              END;
3763.              IF IC+IT>VLC THEN VLC:=IC+IT;
3764.              "WRITE OUT TEXT CARDS"
3765.              UNPACK('TXT',PASCALGO@,2);
3766.              PASCALGO@(13):=' '; PASCALGO@(14):=' ';
3767.              PASCALGO@(15):=CHR(0); PASCALGO@(16):=CHR(1);
3768.              PASCALGO@(11):=CHR(0);
3769.              LIC:=0;
3770.              WHILE LIC<IC DO
3771.              BEGIN
3772.                IF IC-LIC>=56 THEN AMT:=56 ELSE AMT:=IC-LIC;
3773.                K:=IT+LIC;
3774.                PASCALGO@(6):=CHR(K DIV 65536);
3775.                PASCALGO@(7):=CHR(K MOD 65536 DIV 256);
3776.                PASCALGO@(8):=CHR(K MOD 256);
3777.                PASCALGO@(12):=CHR(AMT);
3778.                FOR I:=LIC TO LIC+AMT-1 DO PASCALGO@(I-LIC+17):=CHR(CODE(I));
3779.                FOR I:=AMT+17 TO 72 DO PASCALGO@(I):=' ';
3780.                LIC:=LIC+AMT;
3781.                PUT(PASCALGO);
```

```
3782.            END;
3783.            FINDSEMICOLON;
3784.       20: END;
3785.          PASCALGO@:=BLANKCARD; PASCALGO@(1):=CHR(2);
3786.          UNPACK('END',PASCALGO@,2);
3787.          PASCALGO@(29):=CHR(0); PASCALGO@(30):=CHR(VLC DIV 65536);
3788.          PASCALGO@(31):=CHR(VLC MOD 65536 DIV 256);
3789.          PASCALGO@(32):=CHR(VLC MOD 256);
3790.          PUT(PASCALGO)
3791.        END "WRITVAL";
3792.
3793.        PROCEDURE VARDECL;
3794.           LABEL 10;
3795.           VAR AT : ADDRESS; I,J : SHRTINT;
3796.
3797.           PROCEDURE FILERR;
3798.           BEGIN ERROR(95);  SKIP(12);  GOTO EXIT 10 END;
3799.
3800.        BEGIN INSYMBOL;  SPTR := NEXT;
3801.           WHILE NO = 1 DO
3802.           BEGIN I := 0;
3803.              REPEAT
3804.                 SRCHREC(NEXT);
3805.                 IF CTPTR ¬= NIL THEN ERROR(8) ELSE
3806.                 BEGIN ALLOC(P,VARS);
3807.                    WITH P@ DO
3808.                    BEGIN NAME := AVAL;  NXTEL := NEXT;  KLASS := VARS;
3809.                       VKIND := ACTUAL;  VTYPE := NIL;  ALIGN := 0;
3810.                       VLEVEL := 0 ;
3811.                    END;
3812.                    NEXT := P;  I := I + 1;
3813.                 END;
3814.              INSYMBOL;
3815.              IF NO = 9 "(" THEN " SET( IN,OUT,PRINT,PUNCH  ) "
3816.              BEGIN INSYMBOL;
3817.                 IF (NO = 8) & (CL = 7) THEN "IN"
3818.                    P@.VLEVEL := 4  ELSE
3819.                 IF NO = 1 THEN
3820.                    IF AVAL = 'OUT' THEN
3821.                      P@.VLEVEL := 1  ELSE
3822.                    IF AVAL = 'PRINT' THEN
3823.                      P@.VLEVEL := 2 ELSE
3824.                    IF AVAL = 'PUNCH' THEN
3825.                      P@.VLEVEL := 3 ELSE FILERR ELSE FILERR ;
3826.                 INSYMBOL;  IF NO ¬= 10 ")" THEN FILERR;
3827.                 INSYMBOL;
3828.              END;
3829.              ERR := FALSE;
3830.              IF NO = 15 "," THEN
3831.              BEGIN INSYMBOL;
3832.                 IF NO ¬= 1 THEN
3833.                 BEGIN ERROR(11);  GOTO 10 END;
3834.              END ELSE IF NO ¬= 19 ":" THEN ERROR(10);
3835.           UNTIL NO ¬= 1 ;
3836.           IF NO = 19 ":" THEN INSYMBOL
3837.              ELSE IF ¬ERR THEN ERROR(10) ;
3838.           N := NEXT;  ERR := FALSE;
3839.           TYPEDECL(TL,CTPTR);
3840.           IF ERR THEN GOTO 10;
3841.           IF CTPTR ¬= NIL THEN
```

```
3842.                BEGIN
3843.                   IF I>1 THEN UPALIGN(TL,CTPTR@.ALIGN);
3844.                   UPALIGN(LC,CTPTR@.ALIGN)
3845.                END;
3846.                LC := LC + I*TL;  LL := LC;
3847.                FOR I := I DOWNTO 1 DO
3848.                WITH N@ DO
3849.                BEGIN LL := LL - TL;
3850.                   VTYPE := CTPTR;
3851.                   IF CTPTR ¬= NIL THEN
3852.                   BEGIN  ALIGN := VTYPE@.ALIGN;
3853.                      VLEVEL:=LEVEL; VADDR:=LL; SSIZE:=CTPTR@.SIZE;
3854.                      IF CTPTR@.FORM = CLASSS THEN
3855.                      BEGIN  " CHECK FOR PREDECLARED CLASS "
3856.                         TYPID := NAME;  P := CTPTR@.PELTYPE;
3857.                         IF PFTOP = FILLIMIT THEN ERROR(92) ELSE
3858.                         BEGIN PFTOP := PFTOP + 1;
3859.                            PFL(PFTOP):=LL; PEND(PFTOP):=LL+TL
3860.                         END;
3861.                         FOR J := PTX - 1 DOWNTO 0 DO
3862.                         WITH PTLIST(J) DO
3863.                            IF HNAME = TYPID THEN
3864.                            BEGIN PPTR@.DOMAIN := N;
3865.                               PPTR@.ELTYPE := P;
3866.                               PTX := PTX - 1;
3867.                               HNAME := PTLIST(PTX).HNAME;
3868.                               PPTR := PTLIST(PTX).PPTR;
3869.                            END  " WITH, FOR " ;
3870.                      END  " CHECK CLASS "  ELSE
3871.                      IF CTPTR@.FORM = FILES THEN
3872.                         IF FILTOP = FILLIMIT THEN ERROR(92) ELSE
3873.                         BEGIN FILTOP := FILTOP + 1;  FILPTS(FILTOP) := N END
3874.                   END  "CTPTR ¬= NIL"  ;
3875.                   N := NXTEL;
3876.                END  " FOR I " ;
3877.          10:      FINDSEMICOLON;
3878.             END "WHILE NO = 1";
3879.             IF NO = 47 THEN   " VALUE "
3880.                BEGIN IF LEVEL ¬= 0 THEN ERROR(22);
3881.                   WRITVAL
3882.                END "VALUE";
3883.          END  " VARDECL " ;
3884.
3885.          PROCEDURE FORMPARM;
3886.             " ( HAS BEEN VERIFIED AND NEXT SYMBOL READ "
3887.             LABEL 3;
3888.             VAR SPEC : IDCLASS;  REP : BOOLEAN;
3889.                I:SHRTINT; LL:ADDRESS;
3890.
3891.             PROCEDURE FORMERR;
3892.             BEGIN ERROR(11);  SKIP(10);  GOTO EXIT 3 END;
3893.
3894.          BEGIN SPEC := KONST ;
3895.          1:   REPEAT  I:=0;
3896.                   IF NO = 45 THEN "PROCEDURE"
3897.                   REPEAT INSYMBOL;
3898.                      IF NO ¬= 1 THEN ERROR(11) ELSE
3899.                      BEGIN SRCHREC(NEXT);
3900.                         IF CTPTR ¬= NIL THEN ERROR(8) ELSE
3901.                         BEGIN ALLOC(P,PROC);
```

```
3902.                     WITH P@ DO
3903.                     BEGIN NAME := AVAL;  NXTEL := NEXT;  KLASS := PROC;
3904.                       PROCTYPE := P;  PROCKIND := FORMAL;
3905.                       PROCLEVEL := LEVEL;
3906.                       FORMALS := NIL;
3907.                     END;  LC := LC + 4;  NEXT := P;
3908.                  END "CTPTR = NIL";
3909.                  INSYMBOL;
3910.               END;
3911.            UNTIL NO ¬= 15
3912.            ELSE "NOT PROCEDURE"
3913.            BEGIN
3914.               IF NO ¬= 1 THEN
3915.               BEGIN IF NO = 43 THEN SPEC := VARS ELSE
3916.                     IF NO = 41 THEN SPEC := KONST ELSE
3917.                     IF NO = 44 THEN SPEC := PROC ELSE
3918.                        FORMERR;
3919.                  INSYMBOL;
3920.               END ELSE IF SPEC ¬= VARS THEN SPEC := KONST ;
3921.               IF NO ¬= 1 THEN FORMERR;
3922.               REPEAT
3923.                  SRCHREC(NEXT);
3924.                  IF CTPTR ¬= NIL THEN ERROR(8) ELSE
3925.                  BEGIN
3926.                     CASE SPEC OF
3927.     KONST:          BEGIN ALLOC(P,KONST,FORMAL);
3928.                       WITH P@ DO
3929.                       BEGIN CONTYPE := NIL;  CONKIND := FORMAL;
3930.                         I:=I+1;  CLEVEL := LEVEL;
3931.                       END;
3932.                     END;
3933.      PROC:          BEGIN ALLOC(P,PROC);
3934.                       WITH P@ DO
3935.                       BEGIN PROCTYPE := NIL;  PROCKIND := FORMAL;
3936.                         UPALIGN(LC,4);
3937.                         PROCADDR := LC;  PROCLEVEL := LEVEL;
3938.                         FORMALS := NIL;   ALIGN:=4;
3939.                       END;
3940.                       LC:=LC+4;
3941.                     END;
3942.      VARS:          BEGIN ALLOC(P,VARS);
3943.                       WITH P@ DO
3944.                       BEGIN VTYPE := NIL;  VKIND := FORMAL;
3945.                         UPALIGN(LC,4);  VLEVEL := LEVEL; VADDR := LC;
3946.                         ALIGN:=4;
3947.                       END;
3948.                       LC:=LC+4;
3949.                     END;
3950.                     END "CASE";
3951.                     P@.NAME := AVAL;  P@.NXTEL := NEXT;
3952.                     P@.KLASS := SPEC;
3953.                     NEXT := P;
3954.                  END "CTPTR = NIL";
3955.                  INSYMBOL;
3956.                  IF NO = 15 THEN
3957.                  BEGIN INSYMBOL;
3958.                     IF NO = 19 THEN ERROR(11);
3959.                  END ELSE IF NO ¬= 19 THEN GOTO 4;
3960.               UNTIL NO ¬= 1;
3961.      4:       IF NO ¬= 19 THEN ERROR(10) ELSE INSYMBOL;
```

```
3962.                 IF NO ¬= 1 THEN FORMERR;
3963.                 SEARCH;
3964.                 IF CTPTR = NIL THEN
3965.                 BEGIN ERROR(12);  GOTO 2 END;
3966.                 IF CTPTR@.KLASS ¬= TYPES THEN
3967.                 BEGIN ERROR(18) ; GOTO 2 END ;
3968.                 N := NEXT;
3969.                 WITH CTPTR@ DO
3970.                 IF SPEC=KONST THEN IF (FORM¬=NUMERIC)&(FORM¬=SYMBOLIC)
3971.                   &(FORM¬=POWER) THEN
3972.                 BEGIN TL:=4; UPALIGN(LC,4); LC:=LC+I*4; LL:=LC END
3973.                 ELSE BEGIN UPALIGN(LC,ALIGN); TL:=SIZE;
3974.                   UPALIGN(TL,ALIGN); LC:=LC+I*TL; LL:=LC END;
3975.                 REPEAT
3976.                    CASE SPEC OF
3977.         KONST:     IF N@.CONTYPE = NIL THEN
3978.                    BEGIN
3979.                      LL:=LL-TL;
3980.                      WITH CTPTR@ DO
3981.                      IF(FORM¬=NUMERIC)&(FORM¬=SYMBOLIC)&(FORM¬=POWER) THEN
3982.                      BEGIN
3983.                        N@.KLASS:=VARS;  N@.ALIGN:=4;
3984.                        IF FORM=POINTER THEN N@.VKIND:=ACTUAL
3985.                        ELSE N@.VKIND:=FORMAL;
3986.                        N@.VTYPE:=CTPTR; N@.VADDR:=LL;  N@.VLEVEL:=LEVEL;
3987.                      END
3988.                      ELSE
3989.                      BEGIN N@.ALIGN:=ALIGN;
3990.                        N@.CONTYPE:=CTPTR; N@.CADDR:=LL;
3991.                      END;
3992.                      N:=N@.NXTEL;
3993.                    END ELSE N := NIL;
3994.           PROC:    IF N@.PROCTYPE = NIL THEN
3995.                       IF CTPTR@.FORM > POWER THEN
3996.                       BEGIN ERROR(93);  N := NIL END ELSE
3997.                       BEGIN N@.PROCTYPE := CTPTR;  N := N@.NXTEL END
3998.                    ELSE N := NIL;
3999.           VARS:    IF N@.VTYPE = NIL THEN
4000.                    BEGIN N@.VTYPE := CTPTR;  N := N@.NXTEL;
4001.                    END ELSE N := NIL;
4002.                    END "CASE";
4003.                 UNTIL N = NIL;
4004.         2:      INSYMBOL;
4005.               END "NOT PROCEDURE";
4006.         " 3:    REP := NO IN SET(1,41,43,44,45);  "
4007.         3:    REP := (NO=1)|(NO=41)|(NO=43)|(NO=44)|(NO=45);
4008.               IF NO = 16 THEN
4009.               BEGIN INSYMBOL;  REP := NO ¬= 10  END;
4010.             UNTIL ¬REP;
4011.             IF NO ¬= 10 THEN
4012.             BEGIN ERROR(9);  SKIP(10);
4013.               "IF NO IN SET(41,43,44,45) THEN GOTO 1;"
4014.               IF (NO=41)|(NO=43)|(NO=44)|(NO=45) THEN GOTO 1;
4015.             END;
4016.             "REVERSE POINTERS"
4017.             N := NEXT;  NEXT := NIL;
4018.             WHILE N ¬= NIL DO
4019.             BEGIN P := N;  N := P@.NXTEL;
4020.                P@.NXTEL := NEXT;  NEXT := P;
4021.             END;
```

```
4022.         END "FORMPARM";
4023.
4024.         PROCEDURE ENTERBODY;
4025.         VAR I:INTEGER; TATTR:ATTR; ACON:CONSTANT;
4026.           ATEMP:ALFA; CTEMP:ARRAY(1..10) OF CHAR;
4027.         BEGIN  ISTKLIM:=MAXISTK;  ASTK(MAXISTK):=0;  RP:=0;
4028.           TCT:=LC; TMAX:=LC;
4029.           IC:=0; CLABIX:=0;
4030.           FLCX:=0; HLCX:=0; ALCX:=0; RLCX:=0; ELCX:=0; CHNIX:=1;
4031.           "PROCEDURE ENTRY CODE"
4032.           GENRX(#47,15,20,0,15); "B AROUND ENTRY ID"
4033.           CODE(IC):=10;   "LENGTH OF ENTRY POINT IDENTIFIER"  IC:=IC+1;
4034.           IF SURRPTR=NIL THEN ATEMP:='$$$MAIN'
4035.           ELSE ATEMP:=SURRPTR@.NAME;
4036.           UNPACK(ATEMP,CTEMP,1);
4037.           FOR I:=1 TO 10 DO CODE(IC+I-1):=INT(CTEMP(I));
4038.           IC:=IC+10;
4039.           CODE(IC):=0;  IC:=IC+1;  "REALIGN TO FULL-WORD"
4040.           IC:=IC+4; "LOCATION 16(15) WILL HOLD LENGTH OF DATA AREA"
4041.           GENRS(#90,14,12,12,13); "STM"
4042.           GENRR(#18,3,15); "LR"
4043.           GENRX(#41,4,4095,0,3); "LA"
4044.           GENRX(#41,4,1,0,4); "LA"
4045.           GENRX(#50,13,4,0,2); "ST"
4046.           GENRX(#50,2,8,0,13); "ST"
4047.           GENRR(#18,13,2); "LR"
4048.           GENRX(#5A,2,16,0,3); "A"
4049.           GENRX(#59,2,80,0,12);  "C 2,GETMAIN LIMIT"
4050.           GENRX(#47,12,0,0,0);  "BNH"
4051.           I := IC-2;
4052.           GENRR(#1B,1,1);  "SR"
4053.           WITH ACON DO
4054.           BEGIN  KONSTKIND := EXTREF;  EVALUE := 'PSCLERR';
4055.              LDCST2(ACON,15);  "=V(PSCLERR)"
4056.           END;
4057.           GENRR(#07,15,15);  "BR"
4058.           INS(IC,I);
4059.           IF (LEVEL=0)&VDATA THEN
4060.           WITH ACON DO
4061.           BEGIN "MOVE VALUE INITS INTO STACK FROM ADR=MLC TO VLC"
4062.             GENRX(#58,8,80,0,12); "L GETMAIN LIMIT"
4063.             KONSTKIND:=EXTREF; EVALUE:='$GBLDAT';
4064.             LDCST2(ACON,11); "=V($GBLDAT)"
4065.             KONSTKIND:=INTEGERS; IVALUE:=MLC;
4066.             LDCST2(ACON,9);
4067.             GENRX(#41,10,0,12,9); "LA - DESTINATION"
4068.             GENRX(#41,11,0,11,9); "LA - SOURCE"
4069.             IF VLC-MLC>256 THEN
4070.             BEGIN
4071.               GENRX(#41,6,256,0,0); "LA"
4072.               IVALUE:=VLC-256;
4073.               LDCST2(ACON,7); "=A(VLC-256)"
4074.               GENRX(#41,7,0,12,7); "LA - LIMIT"
4075.               I:=IC;
4076.               GENSS(#D2,0,255,10,0,11); "MVC"
4077.               GENRR(#1A,11,6); "AR"
4078.               GENRS(#87,10,6,0,0); "BXLE"
4079.               INS(I,IC-2);
4080.             END;
4081.             GENSS(#D2,0,(VLC-MLC-1) MOD 256,10,0,11);
```

```
4082.            GENRX(#50,8,80,0,12); "ST GETMAIN LIMIT"
4083.          END;
4084.          WITH TATTR DO
4085.          BEGIN
4086.            ALIGNMENT:=4; KIND:=VARBL; TYPTR:=INTPTR; BREG:=LEVEL;
4087.            ACCESS:=DRCT; PCKD:=FALSE;
4088.          END;
4089.          "FILE INITIALIZATION CODE"
4090.          FOR I:=FILEV(LEVEL) TO FILTOP DO
4091.          WITH ACON,FILPTS(I)@ DO
4092.          BEGIN
4093.            TATTR.DPLMT:=VADDR;
4094.            GENADDR(TATTR,0);
4095.            WITH RXADDR DO GENRX(#41,1,D2,X2,B2);
4096.            GENRX(#41,0,16,0,1); "LOAD ADR OF BUFFER"
4097.            GENRX(#50,0,0,0,1); "STORE BUFFER PTR IN FCB"
4098.            KONSTKIND:=ALFAS; AVALUE:=NAME;
4099.            GENSS(#D2,4,9,1,0,0); "MCV - FILENAME INTO FCB"
4100.            GENCONST(ACON,IC-2);
4101.            KONSTKIND:=INTEGERS; IVALUE:=VTYPE@.FELTYPE@.SIZE;
4102.            LDCST2(ACON,0);
4103.            GENRX(#40,0,14,0,1); "STH - RECORD SIZE INTO FCB"
4104.          END;
4105.          "CLASS INITIALIZATION CODE"
4106.          FOR I:=PILEV(LEVEL) TO PFTOP DO
4107.          WITH TATTR,RXADDR DO
4108.          BEGIN
4109.            DPLMT:=PFL(I); "ADDRESS OF CLASS"
4110.            GENADDR(TATTR,0);
4111.            GENRX(#41,1,D2,X2,B2);
4112.            GENRX(#41,0,8,0,1); "FREE POINTER"
4113.            GENRX(#50,0,0,0,1); "STORE IN CLASS DESCRIPTOR"
4114.            DPLMT:=PEND(I);
4115.            GENADDR(TATTR,0);
4116.            GENRX(#41,0,D2,X2,B2);
4117.            GENRX(#50,0,4,0,1); "STORE IN CLASS DESCRIPTOR"
4118.          END;
4119.        END "ENTERBODY";
4120.
4121.        PROCEDURE LEAVEBODY;
4122.        TYPE LTYP=(R,F,E,H,A,W);
4123.        VAR UNSPEC: RECORD CASE CTYP:LTYP OF
4124.          R:(RDUMMY:REAL; RVAL:REAL);
4125.          F:(FDUMMY:REAL; FVAL:INTEGER);
4126.          E:(EDUMMY:REAL; EVAL:INTEGER);
4127.          H:(HDUMMY:REAL; HVAL:SHRTINT);
4128.          A:(ADUMMY:REAL; AVAL:ALFA);
4129.          W:(WDUMMY:REAL; ATEMP:ARRAY(1..10)OF CHAR);
4130.          END;
4131.        "KLUDGED TO OVERLAY RVAL, FVAL, ETC."
4132.          LELCX,IT2,IT3,SAVEIC,SAVEIC2,TCHAIN,TINX: INTEGER;
4133.          NAME:ALFA;
4134.
4135.          PROCEDURE FIXCONST(LENG:INTEGER; FINX:SHRTINT);
4136.          VAR I,IT2,BYTE1,BYTE2,INDEX:INTEGER;
4137.          BEGIN  INDEX:=FINX;
4138.            IF IC+LENG<CODMAX THEN
4139.            BEGIN
4140.              FOR IT2:=1 TO LENG DO CODE(IC+IT2-1):=INT(UNSPEC.ATEMP(IT2));
4141.              BYTE1:=BASEREG(IC DIV 4096 +1)*16+IC MOD 4096 DIV 256;
```

```
4142.              BYTE2:=IC MOD 256;
4143.              WHILE INDEX¬=0 DO
4144.              BEGIN
4145.                IT2:=CODE(INDEX)*256+CODE(INDEX+1);
4146.                CODE(INDEX):=BYTE1;
4147.                CODE(INDEX+1):=BYTE2;
4148.                INDEX:=IT2
4149.              END;
4150.              IF PCODE THEN
4151.              BEGIN
4152.                OUTCH(' '); OUTCH(' '); OUTHEX(IC,6); OUTCH(' ');
4153.                OUTCH(' '); OUTCH('D'); OUTCH('C'); OUTCH(' ');
4154.                OUTCH('X'); OUTCH('''');
4155.                FOR I:=1 TO LENG DO OUTHEX(CODE(IC+I-1),2);
4156.                OUTCH(''''); OUTCH(EOL);
4157.              END;
4158.              IC:=IC+LENG;
4159.            END ELSE ERROR(107)
4160.          END "FIXCONST";
4161.
4162.        BEGIN
4163.          "MARK UNDEFINED LABELS AS ERRORS"
4164.          FOR IT:=1 TO CLABIX DO
4165.          WITH LABTAB(IT) DO
4166.          IF LABLOC=0 THEN "UNDECLARED LABEL"
4167.          BEGIN
4168.            ERRMESSAGE('LABEL:',LABVAL);
4169.            TCHAIN:=CHAIN;
4170.            WHILE TCHAIN¬=0 DO
4171.            BEGIN
4172.              IT1:=CODE(TCHAIN)*256+CODE(TCHAIN+1);
4173.              INS(IC,TCHAIN);
4174.              TCHAIN:=IT1
4175.            END;
4176.            CHAIN:=TCHAIN
4177.          END;
4178.          "CODE TO CALL CLOSE ROUTINES FOR FILES GOES HERE"
4179.          "EXIT CODE FOR PROCEDURES"
4180.          GENRX(#58,13,4,0,13); "L - DYNAMIC BACK LINK"
4181.          GENRS(#98,14,12,12,13); "LM - RESTORE REGISTERS"
4182.          GENSI(#92,12,13,255); "MVI - RETURN FLAG"
4183.          GENRR(#07,15,14); "BR 14"
4184.          UPALIGN(TMAX,8);
4185.          CODE(16):=TMAX DIV 16777216;
4186.          CODE(17):=TMAX DIV 65536 MOD 256;
4187.          CODE(18):=TMAX DIV 256 MOD 256;
4188.          CODE(19):=TMAX MOD 256;
4189.          SAVEIC:=IC;
4190.
4191.          "RESOLVE CONSTANTS, INSERT IN CODE"
4192.          IF PCODE THEN
4193.          BEGIN OUTCH(EOL); OUTALF('CONSTANTS',10); OUTCH(EOL) END;
4194.          UPALIGN(IC,8);
4195.          FOR IT2 := SAVEIC TO IC-1 DO  CODE(IT2) := 0;
4196.          WHILE RLCX¬=0 DO
4197.          WITH CSTTB(RLCX) DO
4198.          BEGIN
4199.            UNSPEC.RVAL:=VALU.RVALUE;
4200.            FIXCONST(8,INX);
4201.            RLCX:=CNEXT
```

```
4202.          END;
4203.          WHILE FLCX¬=0 DO
4204.          WITH CSTTB(FLCX) DO
4205.          BEGIN
4206.            UNSPEC.FVAL:=VALU.IVALUE;
4207.            FIXCONST(4,INX);
4208.            FLCX:=CNEXT
4209.          END;
4210.          LELCX:=ELCX; UNSPEC.EVAL:=0;
4211.          WHILE LELCX¬=0 DO
4212.          WITH CSTTB(LELCX) DO
4213.          BEGIN
4214.            TINX:=IC;
4215.            FIXCONST(4,INX);
4216.            INX:=TINX;
4217.            LELCX:=CNEXT
4218.          END;
4219.          WHILE HLCX¬=0 DO
4220.          WITH CSTTB(HLCX) DO
4221.          BEGIN
4222.            UNSPEC.HVAL:=VALU.IVALUE;
4223.            FIXCONST(2,INX);
4224.            HLCX:=CNEXT
4225.          END;
4226.          WHILE ALCX¬=0 DO
4227.          WITH CSTTB(ALCX) DO
4228.          BEGIN
4229.            UNSPEC.AVAL:=VALU.AVALUE;
4230.            FIXCONST(10,INX);
4231.            ALCX:=CNEXT
4232.          END;
4233.          IF SURRPTR¬=NIL THEN NAME:=SURRPTR@.SDNAME
4234.          ELSE NAME:='$$$MAIN';
4235.          FSTIXG:=FSTIX;
4236.          WRITOUT(NAME);
4237.          SAVEIC2:=IC; IC:=SAVEIC; IF PCODE THEN PRTCOMP;
4238.          IC:=SAVEIC2;
4239.          CEXTABIX:=FSTIX-1;
4240.        END "LEAVEBODY";
4241.
4242.        BEGIN  "BODY"
4243.           DP:=TRUE;
4244.           IC:=0; MLC:=16777216; VLC:=0;
4245.           PILEV(LEVEL):= PFTOP+1;
4246.           IF LEVEL=0 THEN
4247.           BEGIN FILEV(0):=0; LC:=72+8+4+20+20 END
4248.           ELSE FILEV(LEVEL):=FILTOP+1;
4249.           FSTIX := CEXTABIX + 1 ;
4250.    1:     IF NO = 40 THEN "LABEL"
4251.           BEGIN REPEAT INSYMBOL ;
4252.                 IF (NO = 2)&(CL = 1) THEN
4253.                 BEGIN FOR IT := FSTIX TO CEXTABIX DO
4254.                    IF EXTAB(IT).EXVAL = IVAL THEN
4255.                    BEGIN ERROR(77) ; GOTO 2 END ;
4256.                    IF CEXTABIX = MAXEXLABS THEN ERROR(78) ELSE
4257.                    BEGIN CEXTABIX := CEXTABIX + 1 ;
4258.                       SETSONAME;
4259.                       WITH EXTAB(CEXTABIX) DO
4260.                       BEGIN
4261.                          EXTNAME:=UNIQUENAME;
```

```
4262.                        EXLEVEL:=LEVEL;
4263.                        EXVAL:=IVAL
4264.                     END;
4265.                  END ;
4266.     2:           INSYMBOL
4267.               END "IF (NO=2)&(CL=1)" ELSE
4268.               BEGIN ERROR(61) ; GOTO 3 END
4269.            UNTIL NO ¬= 15 "," ;
4270.     3:     FINDSEMICOLON;
4271.         END ;
4272.         IF NO = 41 THEN " CONST "
4273.         BEGIN INSYMBOL ;
4274.               WHILE NO = 1 DO
4275.               BEGIN REPEAT SRCHREC(NEXT) ; IF CTPTR ¬= NIL THEN ERROR(8);
4276.                  ALLOC(P,KONST,ACTUAL);
4277.                  WITH P@ DO
4278.                  BEGIN NAME := AVAL;  NXTEL := NEXT;  KLASS := KONST;
4279.                    CONKIND := ACTUAL;
4280.                  END;  NEXT := P;
4281.                  INSYMBOL;
4282.                  IF (NO ¬= 8) | (CL ¬= 6) THEN ERROR(4) ELSE INSYMBOL;
4283.                  WITH P@ DO
4284.                  IF NO = 36 THEN  " NIL "
4285.                  BEGIN CONTYPE := NILPTR;
4286.                     VALUES := NILVAL;  INSYMBOL;
4287.                  END ELSE
4288.                  WITH VALUES DO
4289.                  BEGIN INCONST(KKIND,N,NEXT@.NXTEL);
4290.                     CONTYPE := N;  KONSTKIND := KKIND;
4291.                     CASE KKIND OF
4292.               INTEGERS, CHARS, SYMBOLICS:  IVALUE := IVAL;
4293.               REALS:  RVALUE := RVAL;
4294.               ALFAS:  AVALUE := AVAL;
4295.                     END "CASE KKIND";
4296.                     INSYMBOL;
4297.                  END;
4298.                     WHILE NO = 15 DO
4299.                     BEGIN INSYMBOL;
4300.                        IF NO ¬= 1 THEN
4301.                        BEGIN ERROR(11); SKIP(15) END;
4302.                     END;
4303.                  UNTIL NO ¬= 1;
4304.                  FINDSEMICOLON;
4305.               END " WHILE NO = 1 " ;
4306.         END "CONST";
4307.         IF NO =37 THEN  "TYPE"
4308.         BEGIN INSYMBOL;
4309.         WHILE NO = 1 DO
4310.         BEGIN SRCHREC(NEXT); IF CTPTR ¬= NIL THEN ERROR(8);
4311.            TYPID := AVAL;  INSYMBOL;
4312.            IF (NO ¬= 8) | (CL ¬= 6) THEN ERROR(4) ELSE INSYMBOL;
4313.            ERR := FALSE;  TYPEDECL(TL,P);
4314.            IF ¬ERR THEN
4315.            IF P@.NAME ¬= BLANK THEN ERROR(96) ELSE
4316.            BEGIN P@.NAME := TYPID;  P@.NXTEL := NEXT;
4317.               NEXT := P;
4318.            END;
4319.            FINDSEMICOLON;
4320.         END;
4321.         END  "TYPE" ;
```

```
4322.          IF NO=43 THEN  "VAR"  VARDECL;
4323.          IF PTX > 0 THEN
4324.          BEGIN ERROR(12);  OUTCH(EOL);
4325.             FOR PTX := PTX-1 DOWNTO 0 DO
4326.             BEGIN FOR I := 1 TO 9 DO OUTALF(BLANK,10);
4327.                OUTALF('CLASS-ID',10);
4328.                OUTALF(PTLIST(PTX).HNAME,10);  OUTCH(EOL);
4329.             END;
4330.             IF ¬EOLFLAG THEN
4331.             FOR I := 1 TO CHCNT+8 DO OUTCH(' ');
4332.          END;
4333.          DP := FALSE;
4334.         "IF NO IN SET(44,45) THEN""FUNCTION OR PROCEDURE"
4335.          IF (NO=44)|(NO=45) THEN "FUNCTION OR PROCEDURE"
4336.          BEGIN
4337.             REPEAT
4338.                LL := NO; ERR := FALSE;
4339.                OLDLEV := LEVEL;
4340.                IF LEVEL < MAXLEVEL THEN LEVEL := LEVEL + 1 ELSE ERROR(76);
4341.                INSYMBOL;
4342.                IF NO ¬= 1 THEN
4343.                BEGIN ERROR(11);  LEVEL := OLDLEV;  GOTO 1 END;
4344.                LC1 := LC;  SRCHREC(NEXT);
4345.                IF CTPTR ¬= NIL THEN
4346.                IF CTPTR@.KLASS ¬= PROC THEN
4347.                BEGIN ERROR(8); CTPTR := NIL END;
4348.                IF CTPTR = NIL THEN  " UNDECLARED PROCEDURE "
4349.                BEGIN ALLOC(PROCPTR,PROC);
4350.                   WITH PROCPTR@ DO
4351.                   BEGIN NAME := AVAL;  NXTEL := NEXT;  KLASS := PROC;
4352.                      PROCTYPE := PROCPTR;  PROCKIND := ACTUAL;
4353.                      PROCLEVEL := LEVEL - 1;  SURRPROC := SURRPTR;
4354.                      PREDEF:=FALSE;
4355.                      IF PROCLEVEL=0 THEN
4356.                        BEGIN GLOBALNAME:=AVAL; UNIQINDEX:=0 END;
4357.                      SETSDNAME;
4358.                      SDNAME:=UNIQUENAME;
4359.                END;
4360.                DISPLAY(TOP).FNAME := PROCPTR;
4361.                NEXT := NIL ;
4362.                INSYMBOL;
4363.                IF LL = 44 THEN  " FUNCTION "
4364.                BEGIN IF NO ¬= 9 THEN ERROR(29) ELSE
4365.                   BEGIN LC := 76; "SAVE SPACE FOR PTR TO RTN VAL(AT 72)"
4366.                     INSYMBOL; FORMPARM END;
4367.                   IF ERR THEN PROCPTR@.PROCTYPE := NIL;
4368.                   IF NO = 10 THEN
4369.                   BEGIN INSYMBOL;
4370.                     IF NO ¬= 19 THEN ERROR(10) ELSE INSYMBOL;
4371.                     IF NO ¬= 1 THEN
4372.                     BEGIN ERROR(11);  SKIP(49);
4373.                       PROCPTR@.PROCTYPE := NIL;
4374.                     END ELSE
4375.                     BEGIN SEARCH;
4376.                   IF CTPTR ¬= NIL THEN
4377.                   BEGIN IF CTPTR@.KLASS ¬= TYPES THEN
4378.                      BEGIN ERROR(93); CTPTR := NIL END ELSE
4379.                      IF CTPTR@.FORM > POWER THEN
4380.                      BEGIN ERROR(93) ; CTPTR := NIL END ;
4381.                   END ELSE ERROR(12) ;
```

```
4382.                     PROCPTR@.PROCTYPE := CTPTR ;
4383.                     INSYMBOL;
4384.                       END;
4385.                     END "NO = 10";
4386.                   END  " FUNCTION "  ELSE  " PROCEDURE "
4387.                   BEGIN LC := 72 ;
4388.                     IF NO = 9 THEN  " PARMLIST "
4389.                     BEGIN INSYMBOL; FORMPARM;
4390.                       IF ERR THEN PROCPTR@.PROCTYPE := NIL;
4391.                       IF NO = 10 THEN INSYMBOL;
4392.                     END "NO = 9";
4393.                   END "PROCEDURE";
4394.                   IF NO ¬= 16 THEN
4395.                   BEGIN PROCPTR@.PROCTYPE := NIL;
4396.                     ERROR(58);  SKIP(16);
4397.                   END ELSE INSYMBOL;
4398.                   PROCPTR@.FORMALS := NEXT;
4399.                   IF (NO=1) & (AVAL='FORWARD') THEN
4400.                   BEGIN NEXT := PROCPTR;
4401.                     PROCPTR@.SEGSIZE := -LC;
4402.                      " SEGSIZE < 0 SIGNIFIES FORWARD-DECLARATION "
4403.                     INSYMBOL;
4404.                   END ELSE
4405.                   BEGIN TOP := LEVEL + 1;
4406.                     WITH DISPLAY(TOP) DO
4407.                     BEGIN FNAME := NEXT;  OCCUR := BLOCK  END;
4408.                     ALLOC(LFIRSTENTRY,DUMMYCLASS) ;
4409.                     BODY(PROCPTR,LFIRSTENTRY);
4410.                   END;
4411.                 END  " NEW PROC "  ELSE  " PROC ID ALREADY DECLARED "
4412.                 BEGIN
4413.                   IF CTPTR@.SEGSIZE >= 0 THEN  " PREV. DECL NOT FORWARD "
4414.                     ERROR(16);
4415.                   INSYMBOL;
4416.                   IF NO = 9 THEN  " IGNORE PARM-LIST "
4417.                   BEGIN ERROR(23);
4418.                    " REPEAT SKIP(10)  UNTIL ¬(NO IN SET(16,41,43,44,45));"
4419.                     REPEAT INSYMBOL; SKIP(10)  UNTIL
4420.                         ¬((NO=16)|(NO=41)|(NO=43)|(NO=44)|(NO=45));
4421.                     IF NO ¬= 10 THEN ERROR(9) ELSE INSYMBOL;
4422.                   END;
4423.                   IF NO = 15 THEN SKIP(16);
4424.                   IF NO ¬= 16 THEN ERROR(58) ELSE INSYMBOL;
4425.                   IF (NO=1) & (AVAL='FORWARD') THEN "AGAIN FORWARD"
4426.                     INSYMBOL ELSE
4427.                  BEGIN
4428.                   PROCPTR := CTPTR;
4429.                   WITH PROCPTR@ DO
4430.                   BEGIN LC := -SEGSIZE;  NEXT := FORMALS;
4431.                     IF PROCLEVEL=0 THEN
4432.                     BEGIN GLOBALNAME:=NAME;
4433.                       UNIQINDEX:=0;
4434.                       SETSDNAME;
4435.                     END
4436.                   END;
4437.                   TOP := LEVEL + 1;
4438.                   WITH DISPLAY(TOP) DO
4439.                   BEGIN FNAME := NEXT;  OCCUR := BLOCK  END;
4440.                   ALLOC(LFIRSTENTRY,DUMMYCLASS);
4441.                   BODY(PROCPTR,LFIRSTENTRY);
```

```
4442.                 END "NOT FORWARD";
4443.               END  " OLD PROCEDURE " ;
4444.               LC := LC1;
4445.               LEVEL := OLDLEV;
4446.               FINDSEMICOLON;
4447.             "UNTIL ¬(NO IN SET(44,45)) ; "
4448.              UNTIL ¬((NO=44)|(NO=45)) ;
4449.           END  " FUNCTION OR PROCEDURE " ;
4450.           DISPLAY(TOP).FNAME := NEXT;
4451.           IF NO¬=17 THEN
4452.           BEGIN
4453.              IF NO¬=21 THEN
4454.              BEGIN ERROR(24); SKIP(49);
4455.               " WHILE NO IN SET(16,22) DO  "
4456.                 WHILE (NO=16)|(NO=22) DO
4457.                 BEGIN INSYMBOL;
4458.                    SKIP(49);
4459.                 END ;
4460.               "IF NO IN SET(37,40,41,43,44,45) THEN GOTO 1;  "
4461.                 IF (NO=37)|(NO=40)|(NO=41)|(NO=43)|(NO=44)|(NO=45)
4462.                    THEN GOTO 1;
4463.              END ELSE
4464.              ENTERBODY;
4465.              COMPSTAT ;
4466.              LEAVEBODY ;
4467.           END;
4468.           IF SURRPTR ¬= NIL THEN SURRPTR@.SEGSIZE := LC;
4469.           ALLOC(P,DUMMYCLASS);
4470.           RESET(FIRSTENTRY);
4471.           TOP := LEVEL;  NEXT := DISPLAY(TOP).FNAME;
4472.        END  " BODY " ;
4473.
4474.     BEGIN   " *** MAIN PROGRAM *** "
4475.
4476.        ONE:=1;  TEN:=10;  TENTH:=ONE/TEN;  "NO REAL CONSTANTS IN COMPILER"
4477.           " BUT FIXEDTOFLOAT HAS TO WORK "
4478.
4479.        ALLOC(NILPTR,TYPES,POINTER);
4480.        WITH NILPTR@ DO
4481.        BEGIN  NAME := BLANK;  NXTEL := NIL;  ALIGN := 4;
4482.           KLASS := TYPES;  SIZE := 0;  FORM := POINTER;  DOMAIN := NIL;
4483.           ELTYPE := NIL;
4484.        END;
4485.        NILVAL.KONSTKIND := INTEGERS;  NILVAL.IVALUE := 0;
4486.
4487.        NEXT := NIL;
4488.        FOR IT := 0 TO 11 DO
4489.        BEGIN  ALLOC(PT,PROC);  "TEXT .. WRITE"
4490.           WITH PT@ DO
4491.           BEGIN  NAME := INITNAM(IT-1);  NXTEL := NEXT;  ALIGN := 4;
4492.              KLASS := PROC;  PROCTYPE := PT;  FORMALS := NIL;
4493.              SURRPROC := NIL;  PROCKIND := ACTUAL;  SEGSIZE := IT;
4494.              PROCLEVEL := 0;  PREDEF := TRUE;
4495.           END;
4496.           NEXT := PT;
4497.        END;
4498.
4499.        FOR IT := 1 TO 9 DO
4500.        BEGIN  ALLOC(PT,PROC);  "ODD .. SUCC"
4501.           WITH PT@ DO
```

```
4502.          BEGIN  NAME := INITNAM(IT+10);  NXTEL := NEXT;  ALIGN := 4;
4503.             KLASS := PROC;  PROCTYPE := NILPTR;  FORMALS := NIL;
4504.             SURRPROC := NIL;  PROCKIND := ACTUAL;  SEGSIZE := IT;
4505.             PROCLEVEL := 0;  PREDEF := TRUE;
4506.          END;
4507.          NEXT := PT;
4508.       END;
4509.
4510.       ALLOC(CTPTR,TYPES,FILES);
4511.       WITH CTPTR@ DO
4512.       BEGIN  NAME := BLANK;  NXTEL := NIL;  ALIGN := 4;
4513.          KLASS := TYPES;  SIZE := 17;  FORM := FILES;
4514.       END;
4515.
4516.       "WAS THE ABOVE NECESSARY?  CHECK IT OUT"
4517.
4518.       XFILPT := CTPTR;
4519.       FOR IT := 22 TO 23 DO
4520.       BEGIN  ALLOC(PT,VARS);  "INPUT, OUTPUT"
4521.          WITH PT@ DO
4522.          BEGIN  NAME := INITNAM(IT);  NXTEL := NEXT;  ALIGN := 4;
4523.             KLASS := VARS;  VTYPE := CTPTR;  VKIND := ACTUAL;
4524.             IF IT = 22 THEN VADDR := INPT  ELSE VADDR := OUTPT;
4525.             VLEVEL := 0;  FILPTS(IT-22) := PT;  NEXT := PT;
4526.             SSIZE := VTYPE@.SIZE;
4527.          END;
4528.       END;
4529.
4530.       ALLOC(ALFAPTR,TYPES,SYMBOLIC);  "ALFA"
4531.       WITH ALFAPTR@ DO
4532.       BEGIN  NAME := INITNAM(24);  NXTEL := NEXT;  ALIGN := 1;
4533.          KLASS := TYPES;  SIZE := 10;  FORM := SYMBOLIC;
4534.          FCONST := NIL;  PWSET := NIL;  BITSIZE := 80;
4535.       END;
4536.
4537.       ALLOC(REALPTR,TYPES,SYMBOLIC);  "REAL"
4538.       WITH REALPTR@ DO
4539.       BEGIN  NAME := INITNAM(25);  NXTEL := ALFAPTR;  ALIGN := 8;
4540.          KLASS := TYPES;  SIZE := 8;  FORM := SYMBOLIC;  FCONST := NIL;
4541.          PWSET := NIL;  BITSIZE := 64;
4542.       END;
4543.
4544.       ALLOC(CHARPTR,TYPES,SYMBOLIC);  "CHAR"
4545.       WITH CHARPTR@ DO
4546.       BEGIN  NAME := INITNAM(26);  NXTEL := REALPTR;  ALIGN := 1;
4547.          KLASS := TYPES;  SIZE := 1;  FORM := SYMBOLIC;  BITSIZE := 8;
4548.          PWSET := NIL;  "PWSET OF CHARS NOT PRESENTLY ALLOWED "
4549.       END;
4550.
4551.       XFILPT@.FELTYPE := CHARPTR;
4552.
4553.       ALLOC(PT,KONST,ACTUAL);
4554.       WITH PT@ DO
4555.       BEGIN  NAME := BLANK;  NXTEL := NIL;  ALIGN := 1;
4556.          KLASS := KONST;  CONTYPE := CHARPTR;  CONKIND := ACTUAL;
4557.          SUCC := NIL;  VALUES.KONSTKIND := INTEGERS;  VALUES.IVALUE := 256;
4558.       END;
4559.
4560.       CHARPTR@.FCONST := PT;
4561.
```

```
4562.        ALLOC(BOOLPTR,TYPES,SYMBOLIC);  "BOOLEAN"
4563.        WITH BOOLPTR@ DO
4564.        BEGIN  NAME := INITNAM(27);  NXTEL := CHARPTR;  ALIGN := 2;
4565.           KLASS := TYPES;  SIZE := 2;  FORM := SYMBOLIC;  BITSIZE :=1;
4566.        END;
4567.
4568.        ALLOC(PT,KONST,ACTUAL);  "EOL"
4569.        WITH PT@ DO
4570.        BEGIN  NAME := INITNAM(20);  NXTEL := BOOLPTR; ALIGN := 1;
4571.           KLASS := KONST;  CONTYPE := CHARPTR;  SUCC := NIL;
4572.           CONKIND := ACTUAL;
4573.           VALUES.KONSTKIND := CHARS;   VALUES.IVALUE := 0;
4574.        END;
4575.
4576.        NEXT := PT;
4577.        CTPTR := NIL;
4578.        FOR IT := 0 TO 1 DO
4579.        BEGIN  ALLOC(PT,KONST,ACTUAL);  "FALSE,TRUE"
4580.           WITH PT@ DO
4581.           BEGIN  NAME := INITNAM(IT+28);  NXTEL := NEXT;  ALIGN := 2;
4582.              KLASS := KONST;  CONTYPE := BOOLPTR;  CONKIND := ACTUAL;
4583.              SUCC := CTPTR;  VALUES.KONSTKIND := INTEGERS;
4584.              VALUES.IVALUE := IT;  NEXT := PT;
4585.           END;
4586.        END;
4587.
4588.        BOOLPTR@.FCONST := NEXT;
4589.
4590.       ALLOC(INTPTR,TYPES,NUMERIC);  "INTEGER"
4591.        WITH INTPTR@ DO
4592.        BEGIN NAME := INITNAM(30);  NXTEL := NEXT;  ALIGN := 4;
4593.           KLASS := TYPES;  SIZE := 4;  FORM := NUMERIC;  BITS := 32;
4594.           MAX := 2147483647;  MIN := -MAX - 1;
4595.        END;
4596.
4597.
4598.        ALLOC(PT,TYPES,POWER);
4599.        WITH PT@ DO
4600.        BEGIN NAME := BLANK;  NXTEL := NIL;  ALIGN := 4;
4601.           KLASS := TYPES;  SIZE := 4;  FORM := POWER;
4602.           ELSET := BOOLPTR;  PWBITS := 3;
4603.        END;
4604.
4605.        BOOLPTR@.PWSET := PT;
4606.
4607.        ALLOC(UNDECPTR,VARS);
4608.        WITH UNDECPTR@ DO
4609.        BEGIN NAME := BLANK;  NXTEL := NIL;  ALIGN := 4;
4610.           KLASS := VARS;  VTYPE := NIL;  VKIND := ACTUAL;
4611.           VADDR := 0;  VLEVEL := 0; SSIZE := 4;
4612.        END;
4613.
4614.        ALLOC(PNUMPTR,TYPES,POWER);
4615.        WITH PNUMPTR@ DO
4616.        BEGIN NAME := BLANK;  NXTEL := NIL;  ALIGN := 4;
4617.           KLASS := TYPES;  SIZE := 4;  FORM := POWER;
4618.           ELSET := INTPTR;  PWBITS := 32;
4619.        END;
4620.
4621.        ALLOC(LAMPTR,TYPES,POWER);
```

```
4622.        WITH LAMPTR@ DO
4623.        BEGIN NAME := BLANK;  NXTEL := NIL;  ALIGN := 4;
4624.           KLASS := TYPES;  SIZE := 1;  FORM := POWER;
4625.           ELSET := NIL;  PWBITS := 0;
4626.        END;
4627.
4628.        EXTPTR := INTPTR;
4629.       "XFILPT := FILPTS(8);
4630.        WITH XFILPT@ DO
4631.        BEGIN AELTYPE := INTPTR;
4632.           INXTYPE := INTPTR;
4633.        END;   WHO KNOWS WHAT IS GOING ON HERE???"
4634.
4635.        " INITIALISATIONS "
4636.
4637.        ERRINX := 0;  POS1 := 0;  EOLFLAG := TRUE;  SYSPRINT@ := BLANKLINE;
4638.        PTOUT := 0;  DP := TRUE;  ERR := FALSE;
4639.        PRCODE := FALSE;  ASSCHECK := FALSE;  INXCHECK := FALSE;
4640.        DIVCHECK := FALSE;  STOFLCHECK := FALSE;  TRACECHECK := FALSE;
4641.        PCODE := FALSE;  VDATA := FALSE;  LISTING := TRUE;
4642.        PFTOP := -1;  FILTOP := 1;  CEXTABIX := 0;
4643.        PTX := 0;  CHNIX := 1;  B6DPL := 72;
4644.
4645.        IC := 0;  BASEREGNO := 1;  BASE := BASEREG(BASEREGNO);
4646.
4647.        DISPLAY(0).FNAME := EXTPTR;  DISPLAY(0).OCCUR := BLOCK;
4648.
4649.        NO := 0;  LC := IC;
4650.        NEXTCH;  INSYMBOL;
4651.        WHILE (NO ¬= 17 "." ) DO
4652.        BEGIN DISPLAY(1).FNAME := NIL;
4653.           DISPLAY(1).OCCUR := BLOCK;
4654.           TOP := 1;  LEVEL := 0;  NEXT := NIL;
4655.           GLOBALNAME := '$$$MAIN'; UNIQINDEX := 0;
4656.           SETSDNAME;  ISTKLIM := MAXISTK;
4657.           ALLOC(PT);
4658.        "********************
4659.         *                  *
4660.         *"  BODY(NIL,PT);"*  COMPILE USER PROGRAM
4661.         *                  *
4662.         ********************"
4663.        END;
4664.
4665.     END "PASCAL".
```

```
1.      //  JOB ,CLASS=E
2.      //ALLO  EXEC  PGM=IEFBR14
3.      //X  DD  DSN=WYL.SF.PAS.PSCLLIB,DISP=(NEW,KEEP),UNIT=DISK,
4.      //   VOL=SER=WYL010,SPACE=(TRK,(1,1,1),RLSE)
5.      //STEP1  EXEC ASMGCL,PARM.ASM='RENT',PARM.LKED='NCAL'
6.      //ASM.SYSIN DD *
7.      PSCLMON  CSECT
8.      *
9.      *  THIS MODULE IS THE ENTRY MODULE, AND IS RESPONSIBLE FOR
10.     *  OBTAINING CORE, AND PASSING CONTROL TO THE PASCAL PROGRAM
11.     *
12.     *  PROLOGUE
13.     *
14.             SAVE  (14,12),,*
15.             LR    10,15               BASE REGISTER
16.             USING PSCLMON,10
17.             GETMAIN R,LV=WORKSIZE
18.             ST    1,8(13)
19.             ST    13,4(1)
20.             LR    13,1
21.             USING WORKAREA,13
22.     *
23.     *  GET CORE FOR EXECUTION STACK
24.     *
25.             GETMAIN VU,LA=REQAMT,A=AMTOB,SP=1,MF=(E,GETLIST)
26.     *
27.     *  GIVE BACK 8K FOR BUFFERS,ETC.
28.     *
29.             L     0,RELAMT            AMOUNT TO RELEASE
30.             L     1,WHERE             FROM WHERE
31.             A     1,HOWMUCH           END OF AREA
32.             SR    1,0                 BEGINNING OF AREA TO GIVE UP
33.             N     1,MASK              ROUND DOWN TO DOUBLE WORD
34.             O     0,MASK2             INSERT SP NUMBER
35.             LA    3,0(0,1)            SAVE TOP OF GOTTEN AREA
36.             SH    3,=H'256'
37.             FREEMAIN R,LV=(0),A=(1)
38.     *
39.     *  SET UP REGISTERS A LA PASCAL CONVENTIONS
40.     *  AND EXECUTE PASCAL PROGRAM
41.     *
42.             L     2,WHERE             TOP OF STACK
43.             LR    12,2                ADDRESS OF GLOBAL DATA AREA
44.             ST    3,80(12)            SAVE GETMAIN LIMIT
45.             L     15,=V($$$MAIN)      ADDRESS OF MAIN PROGRAM
46.             BALR  14,15               BRANCH TO PASCAL PROGRAM
47.     *
48.     *  FREE ALL CORE
49.     *
50.             FREEMAIN R,SP=1           FREES STACK AREA
51.             LR    1,13                ADDRESS OF WORKAREA TO BE FREED
52.             L     13,SAVE+4           GET ADDRESS OF SAVED REGISTERS
53.     *       FREEMAIN R,LV=WORKSIZE,A=(1)
54.             RETURN (14,12),,RC=0      RETURN
55.     *
56.     PSCLERR  DS    0H                 (SIMPLE) RUN-TIME ERROR EXIT
57.             ENTRY PSCLERR
58.             USING *,15
59.             LA    1,1000(1)           ERROR CODE R1+1000
```

```
60.                ABEND (1),DUMP,STEP
61.          *
62.          *  DATA DEFINITIONS
63.          *
64.          REQAMT   DC    F'16384,131072'     PARM LIST FOR GETMAIN
65.          RELAMT   DC    F'12288'            AMOUNT TO BE RELEASED FOR BUFFERS
66.          MASK     DC    X'FFFFFFF8'         TO ROUND DOWN TO DOUBLE WORD
67.          MASK2    DC    X'01000000'         SUBPOOL NUMBER
68.                   LTORG
69.          WORKAREA DSECT
70.          SAVE     DS    18F                 SAVE AREA
71.          GETLIST  GETMAIN ,MF=L               GETMAIN PARM LIST
72.          AMTOB    DS    2F                  ANSWER RETURNED FROM GETMAIN
73.                   ORG   AMTOB
74.          WHERE    DS    F                   LOCATION OF OBTAINED AREA
75.          HOWMUCH  DS    F                   AMOUNT OBTAINED
76.          WORKSIZE EQU   *-WORKAREA          LENGTH OF WORKAREA
77.                   LTORG
78.                   END
79.          //LKED.SYSLMOD DD DSN=WYL.SF.PAS.PSCLLIB(PSCLMON),DISP=OLD,
80.          //    UNIT=DISK,VOL=SER=WYL010
81.          //STEP2  EXEC ASMGCL,PARM.ASM='RENT'
82.          //ASM.SYSIN DD *
83.          PSCLPUTR CSECT
84.                   SAVE  (14,12),,*
85.                   LR    10,15               BASE REGISTER
86.                   USING PSCLPUTR,10         TELL ASSEMBLER
87.                   LR    9,1                 FCB POINTER
88.                   USING FCB,9
89.                   TM    FCBOFLGS,X'80'      IS FILE ALREADY OPEN?
90.                   BO    OPENOK              YES
91.          *
92.          *  CODE TO OPEN FILE
93.          *
94.                   GETMAIN R,LV=WORKLENG     GET WORKAREA FOR THIS FILE
95.                   USING WORKAREA,13         INCLUDES SAVE AREA
96.                   ST    1,8(13)             LINK SAVE AREAS
97.                   ST    13,4(1)
98.                   LR    13,1
99.                   MVC   WORKDCB(WORKDCBL),DCB     MOVE IN COPY OF DCB
100.                  LA    7,WORKDCB           REMEMBER WHERE IT IS
101.                  USING IHADCB,7
102.                  MVC   DCBDDNAM,FCBDDNAM   MOVE IN FILENAME FROM FCB
103.                  OPEN  ((7),(OUTPUT)),MF=(E,WORKLIST)
104.                  TM    DCBOFLGS,X'10'      SUCCESSFUL OPEN?
105.                  BZ    ABENDO              NO-DIE
106.                  ST    13,FCBWORK@         SAVE WORK AREA ADDRESS IN FCB
107.                  LH    2,DCBLRECL          GET LRECL FROM DDCARD
108.                  LTR   2,2                 0-NOT PRESENT
109.                  BNZ   *+8
110.                  LH    2,DCBBLKSI          GET BLKSIZE INSTEAD
111.                  CH    2,FCBLRECL          SAME AS FILE RECORD SIZE?
112.                  BNE   ABENDL              NO-DIE
113.                  OI    FCBOFLGS,X'80'      EVERYTHING OK-SET OPEN FLAG IN FCB
114.                  B     PUT                 GO DO PUT
115.                  DROP  7
116.         *
117.         *  IF FILE WAS OPEN THEN MUST STILL DO ENTRY CODE
118.         *
119.         OPENOK   L     1,FCBWORK@
```

```
120.              ST    1,8(13)
121.              ST    13,4(1)
122.              LR    13,1
123.     PUT      PUT   WORKDCB,FCBBUFF     ACTUAL I/O
124.     *
125.     *  RETURN CODE
126.     *
127.              L     13,4(13)
128.              RETURN (14,12),,RC=0
129.     *
130.     *  ERROR EXITS
131.     *
132.     ABENDO   ABEND 2000
133.     ABENDL   ABEND 2001
134.              LTORG
135.     *
136.     *  DECLARATIONS
137.     *
138.     DCB      DCB   DSORG=PS,MACRF=(PM)
139.     WLIST    OPEN  (,),MF=L            LIST FORM OF OPEN MACRO(ALSO CLOSE)
140.     *
141.     FCB      DSECT                     FILE CONTROL BLOCK
142.     FCBBUFF@ DS    A                   POINTER TO BUFFER IN PASCAL DATA AREA
143.     FCBDDNAM DS    CL10                FILE NAME
144.     FCBLRECL DS    H                   LRECL ACCORDING TO PASCAL
145.     FCBBUFF  EQU   *                   BUFFER FOLLOWS FCB
146.     FCBOFLGS EQU   FCBLRECL            OPEN FLAG
147.              ORG   FCBDDNAM            OVERWRITTEN IN OPEN CODE
148.     FCBWORK@ DS    A                   POINTER TO SAVE AREA AND DCB
149.     FCBCOUNT DS    F                   NUMBER OF CHAR LEFT IN BUFFER
150.     *                                     (CHARACTER FILES ONLY)
151.     *
152.     WORKAREA DSECT                     SAVE AREA AND DCB
153.     WORKSA   DS    18F
154.     WORKDCB  DCB   DSORG=PS,MACRF=(PM)
155.     WORKLIST OPEN  (,),MF=L            LIST FORM OF OPEN MACRO(ALSO CLOSE)
156.     WORKDCBL EQU   *-WORKDCB           LENGTH OF DCB
157.     WORKLENG EQU   *-WORKAREA          LENGTH OF WORKAREA
158.     *
159.              DCBD  DSORG=PS
160.              END
161.     //LKED.SYSLMOD DD DSN=WYL.SF.PAS.PSCLLIB(PSCLPUTR),DISP=OLD,
162.     //  UNIT=DISK,VOL=SER=WYL010
163.     //STEP3  EXEC ASMGCL,PARM.ASM='RENT'
164.     //ASM.SYSIN DD *
165.     PSCLGETR CSECT
166.              SAVE  (14,12),,*
167.              LR    10,15               BASE REGISTER
168.              USING PSCLGETR,10         TELL ASSEMBLER
169.              LR    9,1                 FCB POINTER
170.              USING FCB,9
171.              TM    FCBOFLGS,X'80'      IS FILE ALREADY OPEN?
172.              BO    OPENOK              YES
173.     *
174.     *  CODE TO OPEN FILE
175.     *
176.              GETMAIN R,LV=WORKLENG     GET WORKAREA FOR THIS FILE
177.              USING WORKAREA,13         INCLUDES SAVE AREA
178.              ST    1,8(13)             LINK SAVE AREAS
179.              ST    13,4(1)
```

```
180.              LR     13,1
181.              MVC    WORKDCB(WORKDCBL),DCB     MOVE IN COPY OF DCB
182.              LA     7,WORKDCB                 REMEMBER WHERE IT IS
183.              USING  IHADCB,7
184.              MVC    DCBDDNAM,FCBDDNAM  MOVE IN FILENAME FROM FCB
185.              OPEN   ((7),(INPUT)),MF=(E,WORKLIST)
186.              TM     DCBOFLGS,X'10'     SUCCESSFUL OPEN?
187.              BZ     ABENDO             NO-DIE
188.              ST     13,FCBWORK@        SAVE WORK AREA ADDRESS IN FCB
189.              LH     2,DCBLRECL         GET LRECL FROM DDCARD
190.              LTR    2,2                0-NOT PRESENT
191.              BNZ    *+8
192.              LH     2,DCBBLKSI         GET BLKSIZE INSTEAD
193.              CH     2,FCBLRECL         SAME AS FILE RECORD SIZE?
194.              BNE    ABENDL             NO-DIE
195.              OI     FCBOFLGS,X'80'     EVERYTHING OK-SET OPEN FLAG IN FCB
196.              B      GET                GO DO GET
197.              DROP   7
198.     *
199.     *  IF FILE WAS OPEN THEN MUST STILL DO ENTRY CODE
200.     *
201.     OPENOK   L      1,FCBWORK@
202.              ST     1,8(13)
203.              ST     13,4(1)
204.              LR     13,1
205.     GET      GET    WORKDCB,FCBBUFF    ACTUAL I/O
206.     *
207.     *  RETURN CODE
208.     *
209.              L      13,4(13)
210.              RETURN (14,12),,RC=0
211.     *
212.     *  ERROR EXITS
213.     *
214.     ABENDO   ABEND 2000
215.     ABENDL   ABEND 2001
216.              LTORG
217.     *
218.     *  DECLARATIONS
219.     *
220.     DCB      DCB    DSORG=PS,MACRF=(GM)
221.     WLIST    OPEN   (,),MF=L           LIST FORM OF OPEN MACRO(ALSO CLOSE)
222.     *
223.     FCB      DSECT                     FILE CONTROL BLOCK
224.     FCBBUFF@ DS     A                 POINTER TO BUFFER IN PASCAL DATA AREA
225.     FCBDDNAM DS     CL10              FILE NAME
226.     FCBLRECL DS     H                 LRECL ACCORDING TO PASCAL
227.     FCBBUFF  EQU    *                 BUFFER FOLLOWS FCB
228.     FCBOFLGS EQU    FCBLRECL          OPEN FLAG
229.              ORG    FCBDDNAM          OVERWRITTEN IN OPEN CODE
230.     FCBWORK@ DS     A                 POINTER TO SAVE AREA AND DCB
231.     FCBCOUNT DS     F                 NUMBER OF CHAR LEFT IN BUFFER
232.     *                                    (CHARACTER FILES ONLY)
233.     *
234.     WORKAREA DSECT                    SAVE AREA AND DCB
235.     WORKSA   DS     18F
236.     WORKDCB  DCB    DSORG=PS,MACRF=(GM)
237.     WORKLIST OPEN   (,),MF=L          LIST FORM OF OPEN MACRO(ALSO CLOSE)
238.     WORKDCBL EQU    *-WORKDCB         LENGTH OF DCB
239.     WORKLENG EQU    *-WORKAREA        LENGTH OF WORKAREA
```

```
240.      *
241.                DCBD  DSORG=PS
242.                END
243.      //LKED.SYSLMOD DD DSN=WYL.SF.PAS.PSCLLIB(PSCLGETR),DISP=OLD,
244.      //  UNIT=DISK,VOL=SER=WYL010
```