

3 1 0 0
3 2 0 0
3 3 0 0
3 5 0 0

COMPUTER SYSTEMS
COMPASS
PROGRAMMING TRAINING MANUAL

CONTROL DATA
CORPORATION

INSTRUCTION INDEX

BY OCTAL OPERATION CODE			BY MNEMONIC OPERATION CODE			BY MNEMONIC OPERATION CODE			
OCTAL OPERATION CODE	MNEMONIC OPERATION CODE	SECTION NUMBER	OCTAL OPERATION CODE	MNEMONIC OPERATION CODE	SECTION NUMBER	MNEMONIC OPERATION CODE	SECTION NUMBER	MNEMONIC OPERATION CODE	SECTION NUMBER
00.0	HLT	3.5.5	40	STA,I	3.2.1	ADA,I	3.3.1	LDA,I	3.1.1
00.1	S-1	3.5.6	41	STQ,I	3.2.2	ADAAQ,I	6.1	LDAQ,I	5.1
00.2	S-2	3.5.6	42	SACH	8.2.3	ADE	14.2.8	LDE	14.2.6
00.3	S-3	3.5.6	43	SQCH	8.2.4	AEU	13.2.2	LDI,I	3.1.3
00.4	S-4	3.5.6	44	SWA,I	3.7	ATA	9.5.2	LDL,I	7.2
00.5	S-5	3.5.6	45	STAQ,I	5.2	ANA	7.4.2	LDQ,I	3.1.2
00.6	S-6	3.5.6	46	SCHA,I	8.2.5	ANA,S	7.4.2	LPA,I	7.4.1
00.7	RTJ	3.5.4	47	STI,I	3.2.3	ANI	7.4.4	LQGH	8.2.2
01	UJP,I	3.5.1	50	MUA,I	3.3.3	ANQ	7.4.3	MEQ	11.1
02.0	No Operation		51	DVA,I	3.3.4	ANQ,S	7.4.3	MOVE,INT	10.3
02.1-3	IJI	3.5.7	52	CPR,I	11.4	AQA	9.5.1	MTH	11.2
02.4	No Operation		53.01	TMQ	9.3.1	AQE	13.4.2	MUA,I	3.3.3
02.5-7	IJD	3.5.7	53.02	TMA	9.2.1	AQJ,EQ	3.5.3	MUAQ,I	6.3
03.0	AZJ,EQ	3.5.2	53.04	AQA	9.5.1	AQJ,GE	3.5.3	NOP	
03.1	AZJ,NE	3.5.2	53.(0+b)0	TIA	9.1.1	AQJ,LT	3.5.3	OTAC,INT	20.8.3
03.2	AZJ,GE	3.5.2	53.(0+b)3	TMI	9.4.1	AQJ,NE	3.5.3	OTAW,INT	20.8.4
03.3	AZJ,LT	3.5.2	53.(0+b)4	AIA	9.5.2	ASE	3.6.1	OUTC,INT,	20.7
03.4	AQJ,EQ	3.5.3	53.41	TQM	9.3.2	ASE,S	3.6.1	B,H	
03.5	AQJ,NE	3.5.3	53.42	TAM	9.2.2	ASG	3.6.2	OUTW,INT,	20.5
03.6	AQJ,GE	3.5.3	53.(4+b)0	TAI	9.1.2	ASG,S	3.6.2	B,H	
03.7	AQJ,LT	3.5.3	53.(4+b)3	TIM	9.4.2	AZJ,EQ	3.5.2	PAUS	10.4 &
04.0	ISE	3.6.1	53.(4+b)4	IAI	9.5.3	AZJ,GE	3.5.2	20.10.5.3	
04.1-3	ISE	3.6.1	54	LDI,I	3.1.3	AZJ,LT	3.5.2	QEL	13.3.2
04.4	ASE,S	3.6.1	55.0	No Operation		AZJ,NE	3.5.2	QSE	3.6.1
04.5	QSE,S	3.6.1	55.1	ELQ	13.3.1	CINS	20.9.5	QSE,S	3.6.1
04.6	ASE	3.6.1	55.2	EUA	13.2.1	CON	20.2	QSG	3.6.2
04.7	QSE	3.6.1	55.3	EAQ	13.4.1	COPY	20.9.2	QSG,S	3.6.2
05.0	ISG	3.6.2	55.4	No Operation		CPR,I	11.4	RAD,I	3.3.5
05.1-3	ISG	3.6.2	55.5	QEL	13.3.2	CTI	20.10.5.1	RTJ	3.5.4
05.4	ASG,S	3.6.2	55.6	AEU	13.2.2	CTO	20.10.5.2	SACH	8.2.3
05.5	QSG,S	3.6.2	55.7	AQE	13.4.2	DINT		SBA,I	3.3.2
05.6	ASG	3.6.2	56	MUAQ,I	6.3	DVA,I	3.3.4	SBAQ,I	6.2
05.7	QSG	3.6.2	57	DVAQ,I	6.4	DVAQ,I	6.4	SBCD	
06.0-7	MEQ	11.1	60	FAD,I	12.3.1	EAQ	13.4.1	SBE	14.2.9
07.0-7	MTH	11.2	61	FSB,I	12.3.2	ECHA	8.2.6	SCA,I	7.5.1
10.0	SSH	11.3	62	FMU,I	12.3.3	ECHA,S	8.2.6	SCAQ	13.5
10.1-3	ISI	3.6.3	63	FDV,I	12.3.4	EINT	21.1-2	SCHA,I	8.2.5
10.4	ISD	3.6.3	64	STQ	14.2.6	ELQ	13.3.1	SCIM	
10.5-7	ISD	3.6.3	65	STQ	14.2.7	ENA	3.4.4	SEL	20.3
11.0	ECHA	8.2.6	66	ADE	14.2.8	ENA,S	3.4.4	SET	14.2.5
11.4	ECHA,S	8.2.6	67	SBE	14.2.9	ENI	3.4.4	SFE	14.2.1
12.0-3	SHA	3.8.2	70.0-3	SFE	14.2.1	ENQ	3.4.4	SFFF	
12.4-7	SHQ	3.8.2	70.4	EZJ,EQ	14.2.2	ENQ,S	3.4.4	SHA	3.8.2
13.0-3	SHAQ	5.3	70.5	EZJ,LT	14.2.3	EOJ	14.2.4	SHAQ	5.3
13.4-7	SCAQ	13.5	70.6	EQJ	14.2.4	EUA	13.2.1	SHQ	3.8.2
14.0	NOP		70.7	SHT	14.2.5	EXS	20.9.1	SJI	3.5.6
14.1-3	ENI	3.4.4	71	SRCE,INT	10.2.1	EZJ,EQ	14.2.2	SJ2	3.5.6
14.4	ENA,S	3.4.4	71	SRCN,INT	10.2.2	EZJ,LT	14.2.3	SJ3	3.5.6
14.5	ENQ,S	3.4.4	72	MOVE,INT	10.3	FAD,I	12.3.1	SJ4	3.5.6
14.6	ENA	3.4.4	73	INPC,INT,	20.6	FDV,I	12.3.4	SJ5	3.5.6
14.7	ENQ	3.4.4		B,H		FMU,I	12.3.3	SJ6	3.5.6
15.0	No Operation		73	INAC,INT	20.8.1	FSB,I	12.3.2	SLS	
15.1-3	INI	3.4.3	74	INAW,INT	20.4	HLT	3.5.5	SQCH	8.2.4
15.4	INA,S	3.4.1		B,N		IAI	9.5.3	SRCE,INT	10.2.1
15.5	INQ,S	3.4.2	74	INAW,INT	20.8.2	IAPR		SRCN,INT	10.2.2
15.6	INA	3.4.1	75	OUTC,INT,	20.7	IJD	3.5.7	SSA,I	7.6
15.7	INQ	3.4.2		B,H		IJI	3.5.7	SSH	11.3
16.0	No Operation		75	OTAC,INT	20.8.3	INA	3.4.1	SSIM	21.1
16.1-3	XOI	7.5.4	76	OUTW,INT,	20.5	INA,S	3.4.1	STA,I	3.2.1
16.4	XOA,S	7.5.2		B,N		INAC,INT	20.8.1	STAQ,I	5.2
16.5	XOQ,S	7.5.3	76	OTAW,INT	20.8.4	INAW,INT	20.8.2	STE	14.2.7
16.6	XOA	7.5.2	77.0	CON	20.2	INCL	21.2	STI,I	3.2.3
16.7	XOQ	7.5.3	77.1	SEI	20.3	INI	3.4.3	STQ,I	3.2.2
17.0	No Operation		77.2	EXS	20.9.1	INPC,INT,	20.6	SWA,I	3.7
17.1-3	ANI	7.4.4	77.2	COPY	20.9.2	B,H		TAI	9.1.2
17.4	ANA,S	7.4.2	77.3	INS	20.9.4	INPW,INT,	20.4	TAM	9.2.2
17.5	ANQ,S	7.4.3	77.3	CINS	20.9.5	B,N		TIA	9.1.1
17.6	ANA	7.4.2	77.4	INTS	20.9.3	INQ	3.4.2	TIM	9.4.2
17.7	ANQ	7.4.3	77.50	INCL	21.2	INQ,S	3.4.2	TMA	9.2.1
20	LDA,I	3.1.1	77.51	IOCL		INS	20.9.4	TMI	9.4.1
21	LDQ,I	3.1.2	77.52	SSIM	21.1	INTS	20.9.3	TMQ	9.3.1
22	LACH	8.2.1	77.53	SCIM		IOCL		TQM	9.3.2
23	LQCH	8.2.2	77.54-56	No Operation		ISD	3.6.3	UGS	
24	LCA,I	7.3.1	77.57	IAPR		ISE	3.6.1	UJP,I	3.5.1
25	LDAQ,I	5.1	77.6	PABS	10.4 &	ISG	3.6.2	XOA	7.5.2
26	LCAQ,I	7.3.2			20.10.5.3	ISI	3.6.3	XOA,S	7.5.2
27	LDL,I	7.2	77.70	SLS		LACH	8.2.1	XOI	7.5.4
30	ADA,I	3.3.1	77.71	SFFF		LCA,I	7.3.1	XOQ	7.5.3
31	SBA,I	3.3.2	77.72	SBCD		LCAQ,I	7.3.2	XOQ,S	7.5.3
32	ADAAQ,I	6.1	77.73	DINT					
33	SBAQ,I	6.2	77.74	STQ	21.1-2				
34	RAD,I	3.3.5	77.75	CTI	20.10.5.1				
35	SSA,I	7.6	77.76	CTO	20.10.5.2				
36	SCA,I	7.5.1	77.77	USE					
37	LPA,I	7.4.1							

	Introduction to 3100/3200/3300/3500	1
	Computer Hardware	2
	Introduction to SCOPE/COMPASS	3
	COMPASS Instructions	4
	Instruction Modification	5
3100	48-bit Operations	6
3200	48-bit, Fixed Point, Arithmetic	7
3300	Logical Operations	8
3500	Character Mode of Operation	9
	Inter-register Transfers	10
	Search and Move Operations	11
	Storage Tests	12
COMPASS	Floating-point Operations	13
PROGRAMMING	48-bit Register Operations	14
TRAINING	B C D Digit Operations	15
MANUAL	COMPASS Pseudo Instructions	16
	SCOPE Organization of I/O	17
	SCOPE Control Cards	18
	SCOPE Debugging Aids	19
	COMPASS Assembly of Constants	20
	Input/Output Without CIO	21
	Interrupts	A
	Sample COMPASS Programs	B
	Additional Exercises	

May 1967
Pub. No. 60184200

Copyright 1967, Control Data Corporation
Printed in the United States of America

Aknowledgement

With the exception of an index, three appendices, two additional chapters, a few new sections and extensive editing, the very excellent programming training manual that follows has been written by Mr. H. D. Pridmore and his staff at Commonwealth Bureau of Census and Statistics in Australia. We are grateful to Mr. Pridmore and the Bureau for permission to modify and print their manual so that all 3100/3200/3300/3500 users may benefit from their efforts.

EDP Education Services Department
Corporate Marketing
Control Data Corporation
May 8, 1967

CONTENTS

CHAPTER 1 - INTRODUCTION TO 3100/3200/3300/3500 COMPUTER HARDWARE

- 1.1 3200 HARDWARE
 - 1.1.1 Diagram of 3200 Computer
 - 1.1.2 Data Bus
 - 1.1.3 Arithmetic Section
 - 1.1.4 Program Control
 - 1.1.5 Block Control
 - 1.1.6 Diagram of 3200 Console Register Display
- 1.2 3200 CORE STORAGE
 - 1.2.1 Storage Word
 - 1.2.2 Word Addressing

CHAPTER 2 - INTRODUCTION TO SCOPE/COMPASS

- 2.1 INTRODUCTION TO THE SCOPE MONITOR
 - 2.1.1 SCOPE Library Tape
 - 2.1.2 SCOPE System Terms
 - 2.1.3 SCOPE Control Card
 - 2.1.4 SCOPE/COMPASS Run
 - 2.1.5 SCOPE/Program Execution Run
- 2.2 INTRODUCTION TO THE COMPASS ASSEMBLY SYSTEM
 - 2.2.1 Assembly Process
 - 2.2.2 General Word Addressing Instruction Format
 - 2.2.3 COMPASS Source Program - Coding
 - 2.2.3.1 LOCATION Field
 - 2.2.3.2 OPERATION Field
 - 2.2.3.3 ADDRESS Field
 - 2.2.3.4 COMMENTS Field
 - 2.2.4 Coding Simple COMPASS Programs
 - 2.2.4.1 Beginning the Program
 - 2.2.4.2 Ending the Program
 - 2.2.4.3 SCOPE Entry
 - 2.2.4.4 Deck Structure for COMPASS Course Exercises

CONTENTS (cont.)

CHAPTER 3 - COMPASS INSTRUCTIONS

- 3.1 LOAD INSTRUCTIONS
 - 3.1.1 Load A
 - 3.1.2 Load Q
 - 3.1.3 Load Index
- 3.2 STORE INSTRUCTIONS
 - 3.2.1 Store A
 - 3.2.2 Store Q
 - 3.2.3 Store Index
- 3.3 ARITHMETIC, FIXED POINT, 24-BIT PRECISION
 - 3.3.1 Add to A
 - 3.3.2 Subtract from A
 - 3.3.3 Multiply A
 - 3.3.4 Divide A
 - 3.3.5 Replace Add
- 3.4 REGISTER OPERATIONS WITHOUT STORAGE REFERENCE
 - 3.4.1 Increase A
 - 3.4.2 Increase Q
 - 3.4.3 Increase Index
 - 3.4.4 Enter Register
- 3.5 JUMP INSTRUCTIONS
 - 3.5.1 Unconditional Jump
 - 3.5.2 Compare A with Zero, Jump
 - 3.5.3 Compare A with Q, Jump
 - 3.5.4 Return Jump
 - 3.5.5 Unconditional Halt
 - 3.5.6 Selective Jump
 - 3.5.7 Index Jump (Incremental/Decremental)
- 3.6 SKIP INSTRUCTIONS
 - 3.6.1 Skip if Equal
 - 3.6.2 Skip if Greater Than or Equal
 - 3.6.3 Index Skip Incremental/Decremental

CONTENTS (cont.)

CHAPTER 3 -

- 3.7 STORE WORD ADDRESS
- 3.8 SHIFT INSTRUCTIONS
 - 3.8.1 Shift Instruction Format
 - 3.8.2 Shift A and Shift Q

CHAPTER 4 - INSTRUCTION MODIFICATION

- 4.1 SIGN EXTENSION
- 4.2 ADDRESS MODES
- 4.3 INDEX MODIFICATION OF WORD ADDRESSING INSTRUCTIONS

CHAPTER 5 - 48-BIT OPERATIONS

- 5.1 LOAD AQ
- 5.2 STORE AQ
- 5.3 SHIFT AQ

CHAPTER 6 - 48-BIT, FIXED POINT, ARITHMETIC

- 6.1 ADD TO AQ
- 6.2 SUBTRACT FROM AQ
- 6.3 MULTIPLY AQ
- 6.4 DIVIDE AQ

CHAPTER 7 - LOGICAL OPERATIONS

- 7.1 LOGIC TABLES
 - 7.1.1 Logical "AND"
 - 7.1.2 Inclusive "OR"
 - 7.1.3 Exclusive "OR"
 - 7.1.4 Examples of Logical Operations Using Octal Numbers
- 7.2 LOAD A LOGICAL
- 7.3 LOAD COMPLEMENTS
 - 7.3.1 Load A Complement
 - 7.3.2 Load AQ Complement

CONTENTS (cont.)

CHAPTER 7 -

- 7.4 LOGICAL "AND" OPERATIONS
 - 7.4.1 Logical Product A
 - 7.4.2 AND of A and y
 - 7.4.3 AND of Q and y
 - 7.4.4 AND of Index Register B^b and y
- 7.5 EXCLUSIVE "OR" OPERATIONS
 - 7.5.1 Selectively Complement A
 - 7.5.2 Exclusive OR of A and y
 - 7.5.3 Exclusive OR of Q and y
 - 7.5.4 Exclusive OR of Index Register B^b and y
- 7.6 SELECTIVELY SET A

CHAPTER 8 - CHARACTER MODE OF OPERATION

- 8.1 INTRODUCTION
- 8.2 CHARACTER ADDRESS INSTRUCTIONS
 - 8.2.1 Load A Character
 - 8.2.2 Load Q Character
 - 8.2.3 Store A Character
 - 8.2.4 Store Q Character
 - 8.2.5 Store Character Address
 - 8.2.6 Enter Character Address into A
- 8.3 INDEX MODIFICATION OF CHARACTER ADDRESSING INSTRUCTIONS

CHAPTER 9 - INTER-REGISTER TRANSFERS

- 9.1 TRANSFERS BETWEEN THE A REGISTER AND INDEX REGISTERS
 - 9.1.1 Index Register to A Register
 - 9.1.2 A Register to Index Register
- 9.2 TRANSFERS BETWEEN THE A REGISTER AND THE REGISTER FILE
 - 9.2.1 Register File to A Register
 - 9.2.2 A Register to Register File
- 9.3 TRANSFERS BETWEEN THE Q REGISTER AND THE REGISTER FILE

CONTENTS (cont.)

CHAPTER 9 -

- 9.3.1 Register File to Q Register
- 9.3.2 Q Register to Register File
- 9.4 TRANSFERS BETWEEN INDEX REGISTERS AND THE REGISTER FILE
 - 9.4.1 Register File to Index Register
 - 9.4.2 Index Register to Register File
- 9.5 INTER-REGISTER ADDITION
 - 9.5.1 Add Contents of Q to Contents of A
 - 9.5.2 Add Contents of Index Register to Contents of A
 - 9.5.3 Add Contents of A to Contents of Index Register

CHAPTER 10 - SEARCH AND MOVE OPERATIONS

- 10.1 BLOCK CONTROL
- 10.2 SEARCH OPERATIONS
 - 10.2.1 Search for Character Equality
 - 10.2.2 Search for Character Inequality
- 10.3 MOVE INSTRUCTION
- 10.4 PAUSE INSTRUCTION
(as used with SEARCH/MOVE Instructions)

CHAPTER 11 - STORAGE TESTS

- 11.1 MASKED EQUALITY SEARCH
- 11.2 MASKED THRESHOLD SEARCH
- 11.3 STORAGE SHIFT
- 11.4 COMPARE (WITHIN LIMITS TEST)

CHAPTER 12 - FLOATING POINT OPERATIONS

- 12.1 INTRODUCTION
 - 12.1.1 Storage of Floating Point Numbers
 - 12.1.2 Normalizing the Coefficient
 - 12.1.3 Exponent
 - 12.1.4 Conversion Procedures

CONTENTS (cont.)

CHAPTER 12 -

- 12.1.5 Unpacking Floating Point Numbers
- 12.2 EXECUTION OF FLOATING POINT OPERATIONS
 - 12.2.1 Addition
 - 12.2.2 Subtraction
 - 12.2.3 Rounding of Floating Point Numbers
 - 12.2.4 Multiplication
 - 12.2.5 Division
- 12.3 FLOATING POINT INSTRUCTIONS
 - 12.3.1 Floating Point ADD
 - 12.3.2 Floating Point SUBTRACT
 - 12.3.3 Floating Point MULTIPLY
 - 12.3.4 Floating Point DIVIDE

CHAPTER 13 - 48-BIT REGISTER OPERATIONS

- 13.1 48-BIT E REGISTER
 - 13.1.1 Introduction
 - 13.1.2 Trapped Instructions for the E Register
- 13.2 TRANSFERS BETWEEN A AND E_U
 - 13.2.1 Transfer E_U to A
 - 13.2.2 Transfer A to E_U
- 13.3 TRANSFERS BETWEEN Q AND E_L
 - 13.3.1 Transfer E_L to Q
 - 13.3.2 Transfer Q to E_L
- 13.4 TRANSFERS BETWEEN AQ AND E
 - 13.4.1 Transfer E to AQ
 - 13.4.2 Transfer AQ to E
- 13.5 SCALE AQ
- 13.6 USE OF THE SCALE AQ INSTRUCTION

CHAPTER 14 - BCD DIGIT OPERATIONS

- 14.1 INTRODUCTION

CONTENTS (cont.)

CHAPTER 14 -

- 14.1.1 BCD Digits
- 14.1.2 Field
- 14.1.3 Sign Bits
- 14.1.4 E_D Register (In Machine)
- 14.1.5 E_D Register (On Console)
- 14.1.6 BCD Fault
- 14.2 BCD INSTRUCTIONS
 - 14.2.1 Shift E_D Register
 - 14.2.2 E_D Equal to ZERO Jump
 - 14.2.3 E_D Less Than ZERO Jump
 - 14.2.4 E_D Overflow Jump
 - 14.2.5 Setting Field Length in D Register
 - 14.2.6 Load E_D
 - 14.2.7 Store E_D
 - 14.2.8 Add to E_D
 - 14.2.9 Subtract From E_D
- 14.3 BCD TRAPPED INSTRUCTIONS

CHAPTER 15 - COMPASS PSEUDO INSTRUCTIONS

- 15.1 CONCEPTS OF PSEUDO INSTRUCTIONS
- 15.2 PROGRAM DEFINITION
 - 15.2.1 IDENT Instruction
 - 15.2.2 END Instruction
 - 15.2.3 FINIS Instruction
- 15.3 ASSEMBLY AREAS
 - 15.3.1 Introduction
 - 15.3.2 DATA Area
 - 15.3.3 Return Assembly Control to Subprogram PRG Area
 - 15.3.4 COMMON Area
 - 15.3.5 ORGR Instruction
- 15.4 STORAGE RESERVATIONS

CONTENTS (cont.)

CHAPTER 15 -

- 15.4.1 Word Block
- 15.4.2 Character Block
- 15.5 ENTRY AND EXTERNAL INSTRUCTIONS
 - 15.5.1 ENTRY Pseudo Instruction
 - 15.5.2 EXTERNAL Pseudo Instruction
 - 15.5.3 SCOPE Loading of Subprogram
- 15.6 SYMBOL DEFINITION BY EQUIVALENCING
 - 15.6.1 Introduction
 - 15.6.2 Word Equating
 - 15.6.3 Character Equating
- 15.7 COMPASS OUTPUT LISTING CONTROL
 - 15.7.1 REMarks
 - 15.7.2 NO LIST Instruction
 - 15.7.3 Resume LISTing Instruction
 - 15.7.4 SPACE Instruction
 - 15.7.5 New Page EJECT Instruction
 - 15.7.6 TITLE Instruction
 - 15.7.7 Comments

CHAPTER 16 - SCOPE ORGANIZATION OF INPUT/OUTPUT

- 16.1 INTRODUCTION
 - 16.1.1 Programmer Units
 - 16.1.2 Scratch Units
 - 16.1.3 Systems Units
- 16.2 CENTRAL INPUT/OUTPUT ROUTINE
 - 16.2.1 Introduction
 - 16.2.2 Calling Sequences
 - 16.2.3 Input/Output Operations
 - 16.2.4 Tape Control Operations
 - 16.2.5 Unit Status Requests
 - 16.2.6 Format Selection
 - 16.2.7 Page Control of the Line Printer

CONTENTS (cont.)

CHAPTER 17 - SCOPE CONTROL CARDS

- 17.1 INTRODUCTION
- 17.2 SEQUENCE CARD
- 17.3 JOB CARD
- 17.4 ENDSCOPE STATEMENT
- 17.5 ENDREEL STATEMENT
- 17.6 CTO STATEMENT
- 17.7 REWIND STATEMENT
- 17.8 UNLOAD STATEMENT
- 17.9 EQUIP STATEMENTS
 - 17.9.1 Hardware Definition
 - 17.9.2 Equating Logical Units
 - 17.9.3 Physical Unit Assignment
- 17.10 TRANSFER STATEMENT
- 17.11 LOAD STATEMENT
- 17.12 COMPASS LIBRARY CALLING STATEMENT
- 17.13 RUN STATEMENT
- 17.14 DIAGRAMMATIC DECK

CHAPTER 18 - SCOPE DEBUGGING AIDS

- 18.1 OCTAL CORRECTION CARDS
 - 18.1.1 Location Symbols
 - 18.1.2 Octal Corrections
 - 18.1.3 Relocation Factors
 - 18.1.4 Error Indicators
- 18.2 SNAP DUMPS
 - 18.2.1 Errors
 - 18.2.2 Location
 - 18.2.3 First and Last Word Addresses
 - 18.2.4 Mode
 - 18.2.5 Identification
 - 18.2.6 Note

CONTENTS (cont.)

CHAPTER 18 -

- 18.2.7 Example
- 18.2.8 Rules for Using SNAP
- 18.3 OTHER DEBUGGING AIDS
 - 18.3.1 Memory Map
 - 18.3.2 Abnormal Termination Dump
- 18.4 COMPASS ERROR CODES

CHAPTER 19 - COMPASS ASSEMBLY OF CONSTANTS

- 19.1 OCTAL CONSTANT PSEUDO INSTRUCTIONS
- 19.2 DECIMAL CONSTANTS, FIXED POINT
- 19.3 DOUBLE PRECISION AND/OR FLOATING POINT CONSTANTS
- 19.4 BCD CONSTANTS
- 19.5 BCD CHARACTER CONSTANTS
- 19.6 VARIABLE FIELD CONSTANTS
 - 19.6.1 Introduction
 - 19.6.2 Octal Mode
 - 19.6.3 Hollerith Mode
 - 19.6.4 Word Address Arithmetic Mode
 - 19.6.5 Character Address Mode
 - 19.6.6 Example of VFD Instruction

CHAPTER 20 - INPUT/OUTPUT WITHOUT CIO

- 20.1 INPUT/OUTPUT CHARACTERISTICS
 - 20.1.1 Introduction
 - 20.1.2 Interface Signals
 - 20.1.3 System Configuration
 - 20.1.4 Logical Sequence of Events for Initiating
INPUT/OUTPUT Operations
- 20.2 CONNECT
- 20.3 SELECT
- 20.4 WORD ADDRESSED INPUT TO STORAGE

CONTENTS (cont.)

CHAPTER 20 -

- 20.5 WORD ADDRESSED OUTPUT FROM STORAGE
 - 20.6 CHARACTER ADDRESSED INPUT TO STORAGE
 - 20.7 CHARACTER ADDRESSED OUTPUT FROM STORAGE
 - 20.8 INPUT/OUTPUT TO AND FROM THE A REGISTER
 - 20.8.1 Input Character to A
 - 20.8.2 Input Word to A
 - 20.8.3 Output Character from A
 - 20.8.4 Output Word from A
 - 20.9 SENSING INSTRUCTIONS
 - 20.9.1 Sense External Status
 - 20.9.2 Copy External Status and Interrupt Mask Register
 - 20.9.3 Sense Interrupt
 - 20.9.4 Sense Internal Status
 - 20.9.5 Copy Internal Status and Interrupt Mask Register
 - 20.9.6 Comments on Internal Status and the Interrupt
Mask Register
 - 20.10 CONSOLE TYPEWRITER INPUT/OUTPUT
 - 20.10.1 General Description
 - 20.10.2 Operation
 - 20.10.2.1 Set Tabs, Margins and Spacing
 - 20.10.2.2 Clear
 - 20.10.2.3 Status Checking
 - 20.10.2.4 Type In and Type Load
 - 20.10.2.5 Type Out and Type Dump
 - 20.10.3 Typewriter Console Switches and Indicators
 - 20.10.4 Character Codes
 - 20.10.5 Input/Output Instructions
 - 20.10.5.1 Set Console Typewriter Input
 - 20.10.5.2 Set Console Typewriter Output
 - 20.10.5.3 Pause Instruction
- (as used with console typewriter instructions)

CONTENTS (cont.)

CHAPTER 21 - INTERRUPTS

- 21.1 INTERRUPTS USING CIC
- 21.2 INTERRUPTS WITHOUT USING CIC
- 21.3 I/O USING CIC
- 21.4 INTERRUPT MASK REGISTER BIT ASSIGNMENTS
- 21.5 CIT ASSIGNMENTS

INTRODUCTION TO 3100/3200/3300/3500 COMPUTER HARDWARE

1.1 3200 HARDWARE

1.1.1 Diagram of 3200 Computer

1.1.2 Data Bus

1.1.3 Arithmetic Section

1.1.4 Program Control

1.1.5 Block Control

1.1.6 Diagram of 3200 Console Register Display

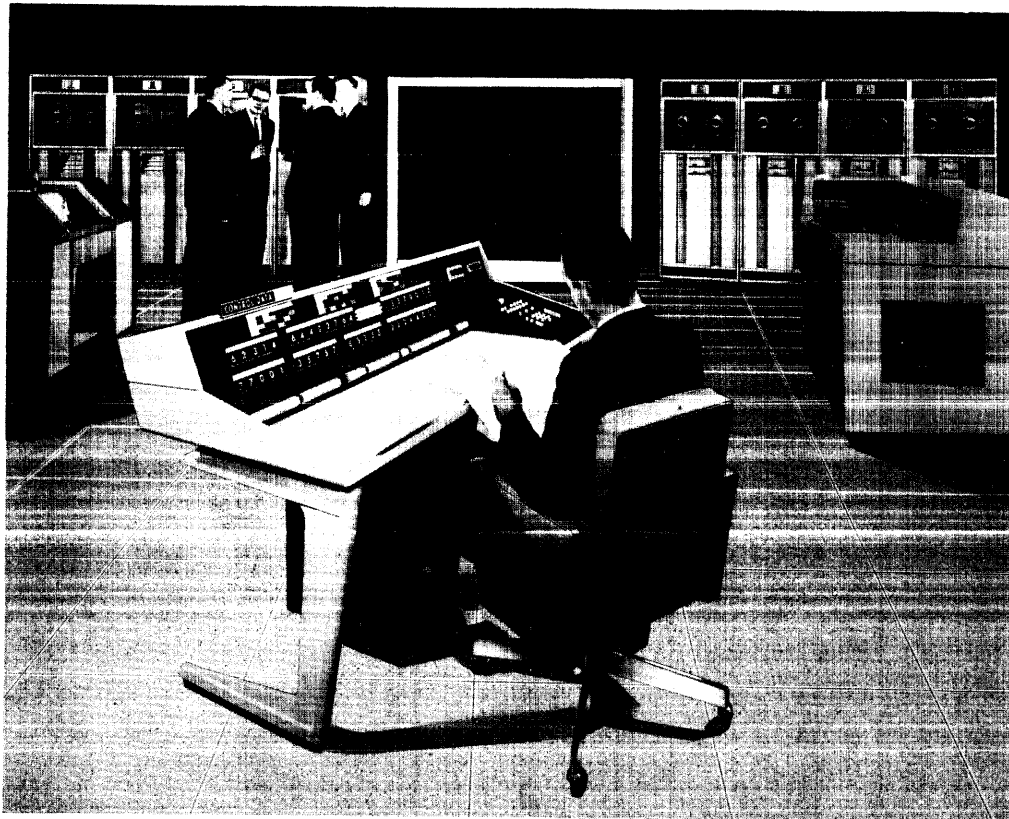
1.2 3200 CORE STORAGE

1.2.1 Storage Word

1.2.2 Word Addressing

CHAPTER 1 - INTRODUCTION TO 3100/3200/3300/3500 COMPUTER HARDWARE

The majority of the programming that will be done for the 3100/3200/3300/3500 computers will be done with a subset of the total instruction repertoire for the particular machine. The subset chosen for this manual applies primarily to the 3100/3200 computers because of the BCD instructions chosen. If this chapter were eliminated, what remains applies equally to all computers in the series. The 3300/3500 computers in addition to having different BCD instructions, have some additional features and instructions. These will not be discussed here. The student is referred to the appropriate machine reference manual for a description of those additions. The timing information included in this manual applies to the 3200/3300 computers. 3100 times are about forty percent higher; 3500 times are at least forty percent lower. Other than in the case of timing information and the BCD instructions, the manual points out those areas that are specifically 3200, which was the machine chosen as representative of those in the series.

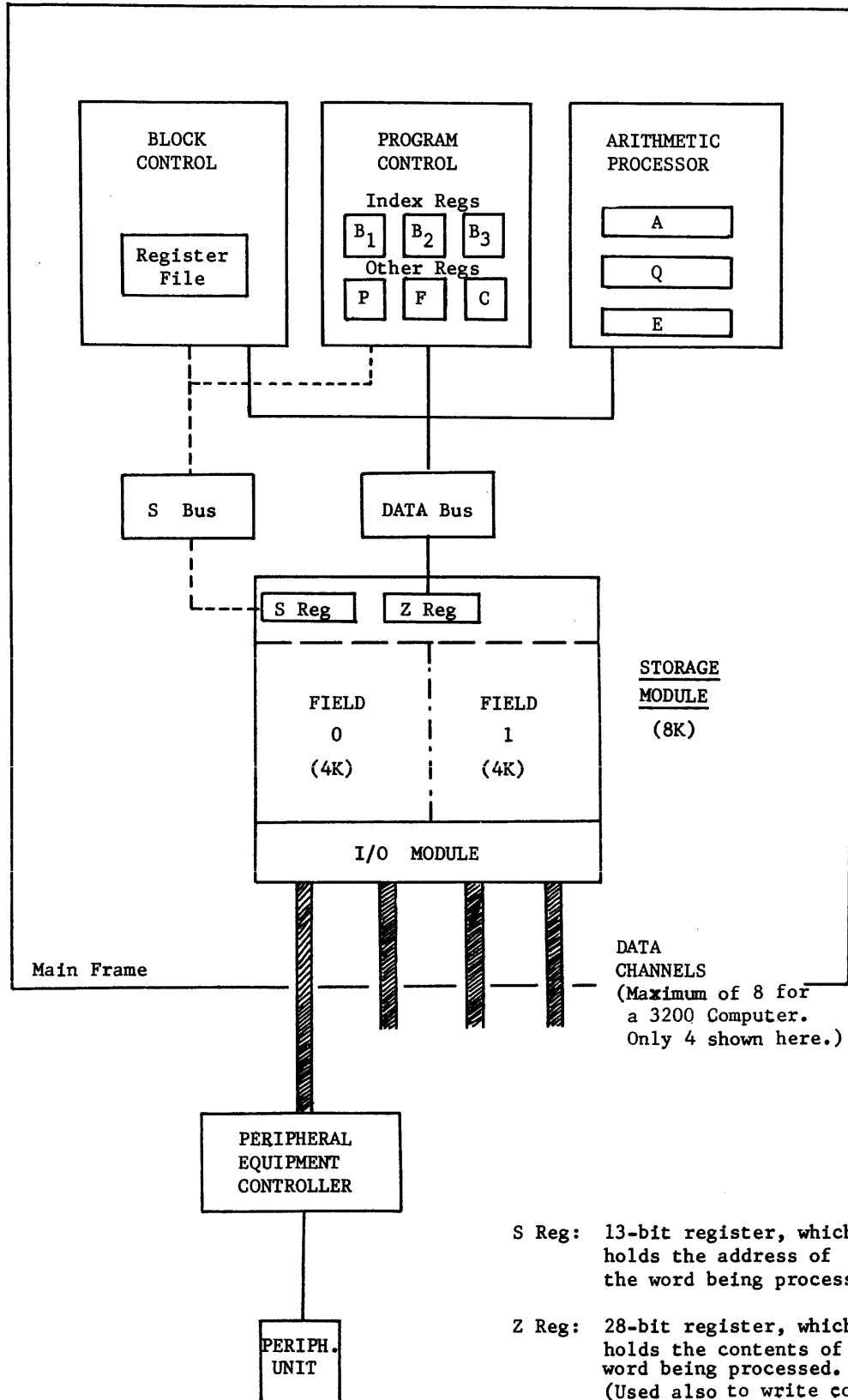


3200 COMPUTER SYSTEM

STUDENT NOTES

1.1 3200 HARDWARE

1.1.1 Diagram of 3200 computer



S Reg: 13-bit register, which holds the address of the word being processed.

Z Reg: 28-bit register, which holds the contents of the word being processed. (Used also to write contents of word back into store.)

1.1.2 Data Bus

The data bus is the path along which data flows between the various sections of the computer. The sections (storage, the arithmetic unit, the console typewriter, and the input/output section) are connected in parallel to the data bus. During execution of each instruction, program control determines which unique path is to be enabled so that the function called for in the program instruction can be carried out.

The data bus contains a 24-bit register called the Data Bus Register (DBR) which is used to hold data temporarily during a data transfer.

1.1.3 Arithmetic Section

The arithmetic section of the 3200 processor consists of three operational registers, namely -

- (i) A - arithmetic register
- (ii) Q - auxiliary arithmetic register
- (iii) E - optional arithmetic register.

(i) The A Register (Accumulator)

The A Register is the principal arithmetic register. It is a 24-bit register, whose contents can be displayed on the console of the 3200.

All arithmetic and logical operations use the A register in formulating a result. It is the only register with provision for adding its contents to the contents of a storage word or another register.

(ii) The Q Register (Quotient)

The Q register is an auxiliary register and is generally used in conjunction with the A register. It is a 24-bit register, whose contents can be displayed on the 3200 console. Combined with A it forms a 48-bit register, AQ. Most arithmetic operations possible with the A register are also possible with the AQ register. The Q register often is used to provide temporary storage for the contents of the A register while the A register is used for some other operation.

(iii) The E Register

This register acts as a supplement to the AQ register. It is a 48-bit register, whose contents can be displayed on the 3200 console, in the display sections usually occupied by A and Q. (The upper 24 bits of E - called E_U - are displayed in A, and the lower 24 bits of E - called E_L - are displayed in Q.) The register is used in 48-bit precision multiplication and division, and in floating point multiplication and division.

In BCD operations, the E register is designated the E_D register, and its size is extended to 53 bits, to enable it to handle thirteen 4-bit characters, plus a sign bit.

1.1.4 Program Control

The program control section contains six operational registers, all of which may be displayed on the 3200 console. They are -

- (i) F - program control register
- (ii) P - Program address counter
- (iii) C - Communication register
- (iv) B - Three index registers, B₁, B₂, and B₃

(i) The F Register

This 24-bit register is used to hold the instruction during the time it is being executed.

(ii) The P Register

This 15-bit register holds the address of the instruction currently being executed, and generates, in sequence, the storage addresses which contain the individual instructions. After execution of an instruction, P is altered to indicate the address of the next instruction to be read. The address is sent via the S (address) BUS to the specified storage module where the instruction is read.

(iii) The C Register

The 24-bit C register is used to enter quantities into Storage, A, Q, E, B or P registers via the console key board. The quantity is entered in C, and then transferred to the specified register when the "transfer" button on the keyboard is pushed.

(iv) The B Registers

The 15-bit B registers (Index registers) are used principally as counters and instruction modifiers.

1.1.5 Block Control

For a general description of the use of Block Control see Section 10.1.

(i) The Register File

The register file is a 64 word (24-bits per word) rapid access memory with a cycle time 0.5 microseconds. Although the programmer can access all registers in the file with the inter-register transfer instructions, certain registers are reserved for specific purposes. These are defined in the following table:

1.1.5 (cont.)

REGISTER NUMBER	RESERVED FOR
00-07	Modified I/O instruction word containing the current character address (Channel 0-7 control)
10-17	Modified I/O instruction word containing the last character address plus (or minus) one (Channel 0-7).
20	Current character address for search control
21	Source address for move control
22	Real time clock, current time
23	Current character address for typewriter control
24-27	Temporary storage
30	Last character address + 1 for search control
31	Destination address for move control
32	Real time clock, time at which to generate interrupt
33	Last character address + 1 for typewriter control
34-77	Temporary storage

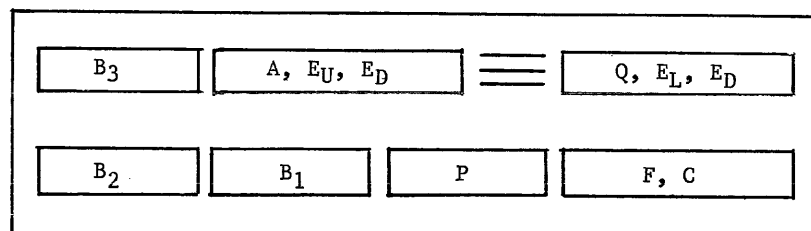
Because of the fast access time, use of Registers 24-27 and 34-77 as temporary storage will speed up program execution. Other registers may also be used for temporary storage if their use will not disrupt operations in progress.

(ii) The Real Time Clock

The real time clock is a 24-bit counter that is incremented each millisecond. The current time is stored in Register 22. It is removed from storage each millisecond, updated, and compared with the contents of Register 32. When the two are equal, an interrupt condition occurs.

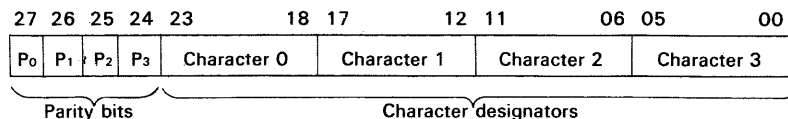
The clock has a period of 16,777,216 milliseconds (approximately 4 hours 40 minutes). It starts as soon as power is supplied to the computer. Its contents may be examined at any time by transferring them to the A register using an inter-register transfer statement. It may be reset to any 24 bit quantity (including zero) by loading A, and transferring the contents of A into Register 22.

1.1.6 Diagram of 3200 Console Register Display



1.2 3200 CORE STORAGE

1.2.1 Storage Word



The 3200 is a word machine with each word consisting of 28 bits. Each word may be regarded as four 6-bit characters as shown above, with parity bits for each character located in bits 24-27.

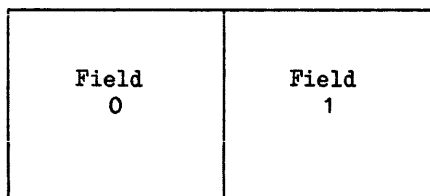
1.2.2 Word Addressing

Each word in the storage of the 3200 is addressable. The address of a particular word is its relative position in the storage. The first word is word 00000, the second is word 00001, the third 00002, etc. The address is specified as five octal digits, which may be broken down to indicate the actual location of the word in core.

The 3200 core is composed of modules of 8,192 words each. There may be up to four such modules attached to the computer, and the modules are numbered from 0 through 3. Thus an 8K 3200 would have only one module, numbered 0₈ (00₂), a 16K 3200 would have two modules, 00₂ and 01₂, and so on.

Each 8K module is made up of two 4,096 word fields, numbered field 0 and field 1.

Module 00



Within each field, words may be regarded as numbered from 0000 to 7777₈ (4096₁₀ words). The position of a word in a field is known as its Co-ordinate address in that field.

Thus a 15-bit address 01200₈ may be divided up to indicate its position in core as follows:

- bits 00 - 11 indicate the co-ordinate address (1200₈)
- bit 12 indicates the field (0)
- bits 13 - 14 indicate the module (0)

Examples:

Address 31040₈ = Module 1, field 1, address 1040₈

Address 77777₈ = Module 3, field 1, address 7777₈

2.1 INTRODUCTION TO THE SCOPE MONITOR

- 2.1.1 SCOPE Library Tape
- 2.1.2 SCOPE System terms
- 2.1.3 SCOPE Control Card
- 2.1.4 SCOPE/COMPASS Run
- 2.1.5 SCOPE/Program Execution Run

2.2 INTRODUCTION TO THE COMPASS ASSEMBLY SYSTEM

- 2.2.1 Assembly Process
- 2.2.2 General Word Addressing Instruction Format
- 2.2.3 COMPASS Source Program - Coding
 - 2.2.3.1 LOCATION Field
 - 2.2.3.2 OPERATION Field
 - 2.2.3.3 ADDRESS Field
 - 2.2.3.4 COMMENTS Field
- 2.2.4 Coding Simple COMPASS Programs
 - 2.2.4.1 Beginning the Program
 - 2.2.4.2 Ending the Program
 - 2.2.4.3 SCOPE Entry
 - 2.2.4.4 Deck Structure for COMPASS Course Exercises

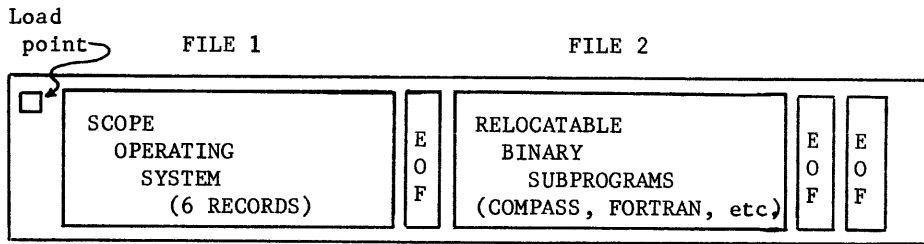
2.1 INTRODUCTION TO THE SCOPE MONITOR

The SCOPE monitor is a program that provides a system of operator and programmer aids to increase through-put and to simplify the operator's job.

Its purpose is to increase job processing efficiency by increasing information through-put, and to minimize operator errors, operator intervention and idle computer time.

2.1.1 SCOPE Library Tape

The library tape serves as the source for the SCOPE operating system as well as those library routines operating under control of SCOPE. The tape consists of two files:



LIBRARY TAPE FORMAT

The first file consists of absolute binary information which, from this point on, will be considered the SCOPE operating system. The second file consists of relocatable binary subprograms such as COMPASS, FORTRAN, COBOL, etc.

2.1.2 SCOPE System Terms

RUN

The complete execution of a subprogram under the control of SCOPE.

JOB

The sets of tasks assigned to SCOPE by the programmer. A job consists of one or more runs.

STACKED JOBS

A stack consists of one or more jobs. The termination of a job is signaled by the printout of the SEQUENCE card for the next job.

NON-STACKED JOBS

A stack which consists of only one job. The presence of the "NS" parameter on the job card indicates that this job is a non-stacked job.

The termination of a non-stacked job is signaled by the printout of "NORMAL END" or "ABNORMAL END" at which time processing will halt.

2.1.2 (cont.)

SUBPROGRAM

The smallest unit recognized by the SCOPE LOADER.

LUN

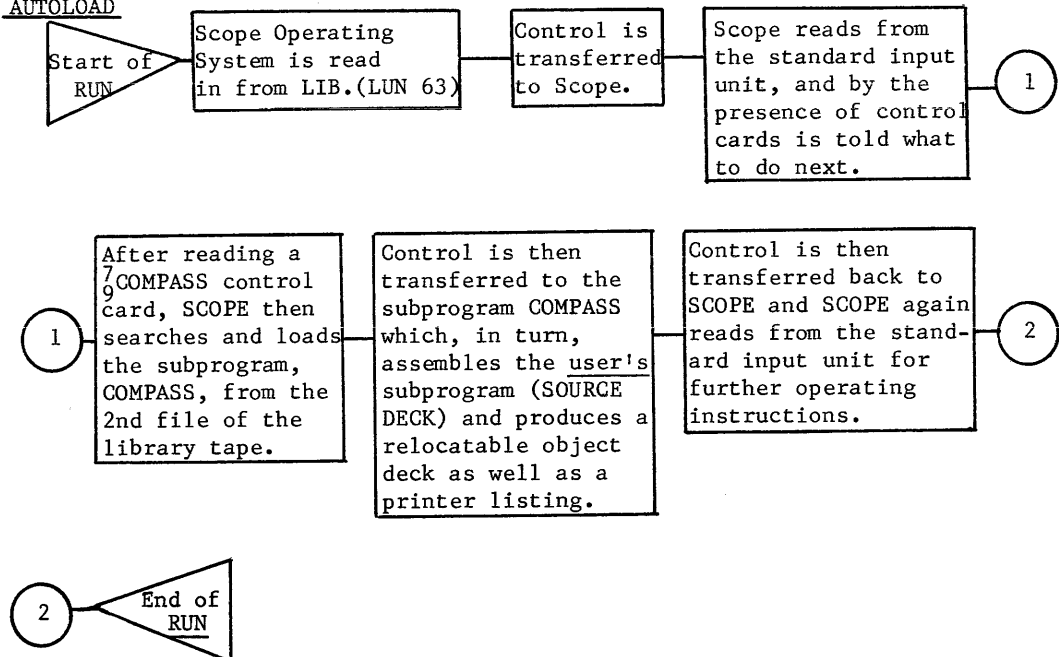
A two-digit decimal number representing a logical reference to a physical I/O unit.

2.1.3 SCOPE Control Card

Refer to Chapter 17

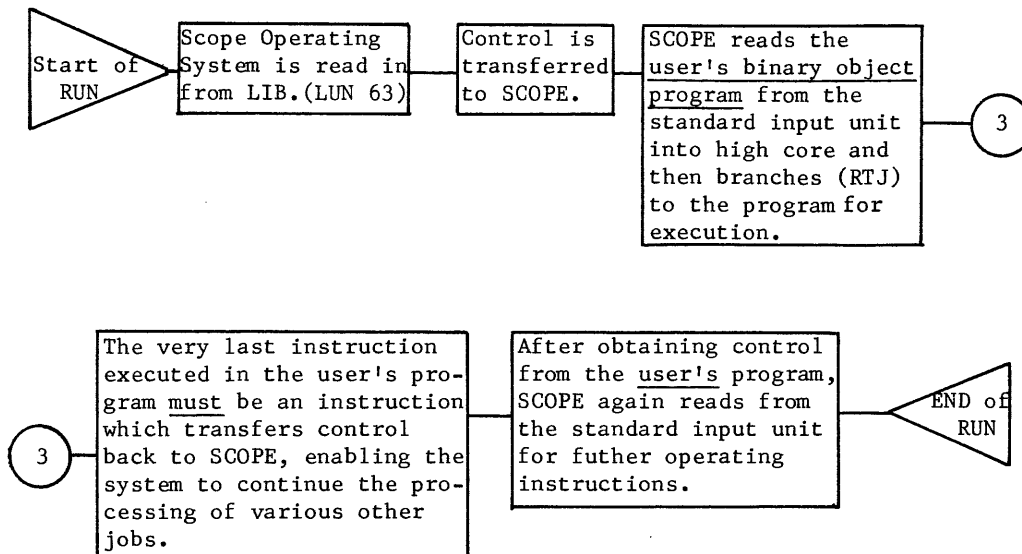
2.1.4 SCOPE/COMPASS Run

AUTOLOAD



SCOPE/COMPASS RUN

2.1.5 SCOPE/Program Execution Run



SCOPE/PROGRAM EXECUTION RUN

2.2 INTRODUCTION TO THE COMPASS ASSEMBLY SYSTEM

- (a) COMPASS is the COMPREhensive ASsembly System for use with Control Data Computers.
- (b) COMPASS operates under the SCOPE monitor system.
- (c) COMPASS enables the programmer to write machine language through the use of mnemonic instructions and symbolic addresses. The COMPASS assembler translates these instructions and addresses into machine language.
- (d) In COMPASS source language, the programmer is also able to specify constants, exercise control over subprogram communication and control the assembly process with a powerful set of PSEUDO (assembly) instructions.
- (e) A COMPASS program consists of a number of linked subprograms, each of which will be assembled independently and linked by the LOADER, prior to execution of the program.
- (f) A COMPASS subprogram consists of lines of coding preceded by an IDENT pseudo instruction, and followed by an END pseudo instruction. The size of the subprogram and the magnitude of the problems solved by it are at the discretion of the programmer.

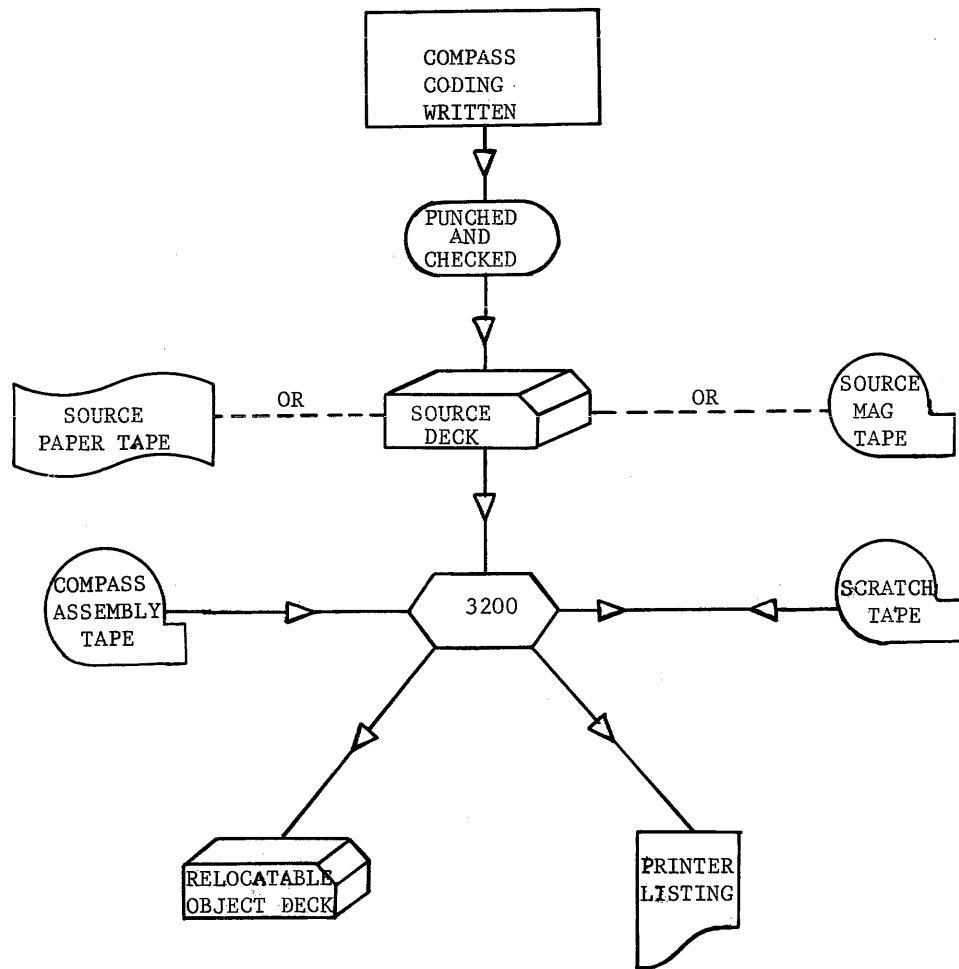
2.2.1 Assembly Process

- (a) Assembly language is much closer to machine code than languages like FORTRAN and COBOL. A line of FORTRAN coding may generate many machine instructions (perhaps in the ratio of 1:10). Most COMPASS instructions generate only one machine instruction.

Example : STA CØUNT
 If CØUNT is located at address 174₈,
 this instruction would be assembled as
 40000174

- (b) The assembly process may be represented diagrammatically as follows:

DIAGRAM OF THE ASSEMBLY PROCESS



- Note: (a) The COMPASS assembly program converts programs written in COMPASS into machine language for execution under the SCOPE monitor system.
- (b) Source programs may be punched on cards or paper tape, or written on magnetic tape.
- (c) Output from the assembler includes an assembly listing and a relocatable binary object deck. The output deck may be punched out on cards, or written on a magnetic tape for immediate execution.

2.2.1 (cont.)

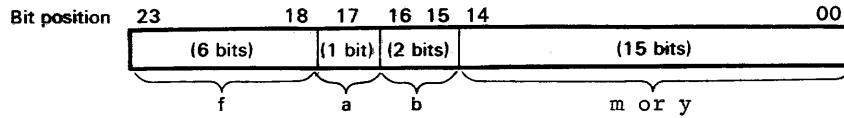
(d) What does the assembler do?

- (i) Allocates storage locations to instructions and to blocks defined in pseudo or area-definition instructions. Note use of "ORGR".
- (ii) Replaces symbolic addresses with allocated storage locations from (i).
- (iii) Replaces symbolic instruction addresses with actual instruction addresses from (i).
- (iv) Indicates index register use and/or indirect addressing in the relevant bits of the necessary instructions.
- (v) Converts mnemonic operation-codes to machine language equivalents.
- (vi) Stores literals in a special literal table, after checking for any prior use of that literal.
- (vii) Replaces literals in the instructions with the addresses of the table locations of the literals.
- (viii) Check for assembly language (source language) errors and generates necessary diagnostic messages.

Most assemblers perform these functions in a number of passes. The source program is first written on to tape, at the same time carrying out some of the above functions. Then the tape is passed against the assembler at least once more to carry out the remainder of the functions.

2.2.2 General Word Addressing Instruction Format

The 3200 (24-bit) word addressing instruction takes the following general form -



Where f = function code of 6 bits.
Range is 00_8 through 77_8 .

This code determines the type of action to be carried out.

e.g. 20_8 = Load the A register.

31_8 = Subtract from the A register.

a = Addressing mode of 1 bit.
If this bit is a zero, direct addressing is carried out.
If this bit is a 1 bit, indirect addressing is carried out.

b = Index designator of 2 bits.
If b = 00, no index is used
If b = 01, index 1 is used
b = 10, index 2 is used
b = 11, index 3 is used.

m = Execution address of 15 bits.
This is the address in memory at which data required for execution will be found. The 24-bit quantity found at that address will be used in a manner determined by the rest of the instruction.

y = Operand of 15 bits.
This specifies a 15 bit quantity which will be used as data in some way. It is the actual data, not the address of the data.

Exercise: Divide up the following octal instructions into their components:

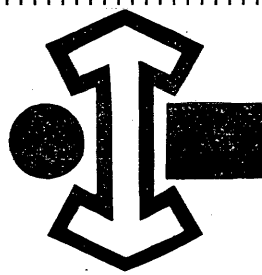
20400100
27100000
56201010
01322222
77654321
46077776
17500000
66766766

Note that the address "m" specified in a word address instruction can be divided further to indicate its actual location in storage.

2.2.3 (cont.)

Following is a sample of the COMPASS CARD.

	LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD	COMMENTS	IDENT
		00000000	00000000	000000000000000000000000	000000000000000000000000
1	11111111	11111111	111111111111111111111111	111111111111111111111111	11111111
2	22222222	22222222	222222222222222222222222		2222222222222222
3	33333333	33333333	333333333333333333333333		3333333333333333
4	44444444	44444444	444444444444444444444444		4444444444444444
5	55555555	55555555	555555555555555555555555		5555555555555555
6	66666666	66666666	666666666666666666666666		6666666666666666
7	77777777	77777777	777777777777777777777777		7777777777777777
8	88888888	88888888	888888888888888888888888		8888888888888888
9	99999999	99999999	999999999999999999999999		9999999999999999



CONTROL DATA CORPORATION
COMPASS

MC-ZT12901

COMPASS CARD (full size)

2.2.3.1 LOCATION Field

Symbolic Address of the instruction written on that line.

e.g. *LøøP* ADA TABLE

-

-

-

UJP *LøøP*

OR Symbolic address of a storage area set up by a pseudo instruction.

e.g. ALEK øCT 4

-

-

-

LDA ALEK

- Restrictions - (a) 1-8 characters (need not be right or left justified).
- (b) First character must be alphabetic, others can be alphanumeric.
- (c) Special characters are not allowed, except period.
- (d) Imbedded blanks are illegal.

Examples:

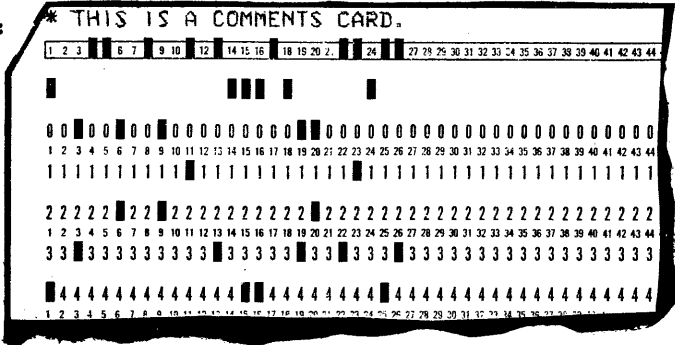
<u>LEGAL</u>		<u>ILLEGAL</u>	
ABCDE	A1A1A1	1A1A1A	Z Z
A1234	XYZ.23	TØM+5	ABCDEFGHIJ
X	PART3	27	FØG*

2.2.3.1 (cont.)

Special Comment Card

An asterisk in Column 1 of the location field means that the rest of the card will be treated as if it is a comment.

Example:



2.2.3.2 OPERATION Field

This field can contain

- (a) Mnemonic operation codes.
- (b) Pseudo instruction Mnemonics.
- (c) Macro instruction names.
- (d) Octal instructions 00_8-77_8 .

The code must begin in col. 10, otherwise it will assemble as 00_8 (halt code). Modifiers are written in this field.

e.g. CHAR BSS,C 4 means set up an area consisting of 4 characters and labelled "CHAR".

Examples:

```
05  
BSS  
ENA,S  
LDA  
LDA,I  
INPC,INT,B,H  
MACNAME
```

2.2.3.3 ADDRESS Field

This field can contain :

- (a) Symbols
- (b) Constants
- (c) Special Characters
- (d) Literals
- (e) Expressions

The address field can begin anywhere after the operation field, provided it is separated from the operation field by at least one blank.

2.2.3.3 (cont.1)

However, it must begin before column 41, and finish before column 73.

(a) Symbols:

A symbol appearing in the address field must be defined by appearance in the location field of an instruction in the subprogram, or be declared as external. If it is not so defined, an error flag "U" - undefined symbol - will be given on the listing. The symbol may be relocatable or non-relocatable. The value assigned to a non-relocatable symbol will not be modified on loading.

Examples:	LDA	CIC.2 (relocatable)
	CIC.2 ENA	0
	LDA	INTADDR (non-relocatable)
	INTADDR EQU	4

(b) Constants:

Integer constants can represent a number of functions depending on the type of instruction -

e.g.	octal address	(LDA)	(LDA 100B)
	constant value	(ENA)	(ENA 100B)
	shift factor	(SHA)	(SHA 6)

Octal integers must be suffixed by the letter B.

NOTE: Numbers in address field are assumed to be decimal unless followed by a B.

(c) Special Characters:

Two special entries may be made in the address field -

- (i) A single asterisk is interpreted as the current value of the address counter when the * is encountered.

Example:	ENA	0
	LDA	*-1

means: Load the A register with the instruction preceding the LDA instruction.
(i.e. with assembled ENA 0)

- (ii) The double asterisk causes the address portion of the instruction to be assembled with a 1-bit in each bit position, and is used where modification of the address will take place during execution.

Example:	START	UJP	**
	Assembled as	01077777	

(d) Literals:

If the address field refers to an operand (a value), the entry may be a literal, expressed as =mv.

where m = the mode
v = the value

Double precision literals are expressed as =2mv.

2.2.3.3 (cont.2)

The mode of the literal may be:

(i) Decimal

DVA	=D23
DVAQ	=2D2753

(ii) Octal

LDA	=Ø777
LDAQ	=2Ø7777777777777777

(Up to 16 octal digits may be specified for double precision)

(iii) Hollerith

LDA	=HABCD	(max: 4 characters)
LDAQ	=2HABCDEFGH	(max: 8 characters)

During assembly, a literal is converted to binary, and assigned a relocatable address which is substituted for the literal in the object code. Literals are all stored at the end of the subprogram. If two literals of the same value and size are specified, they are not duplicated. When COMPASS encounters a literal, the value is compared against all other previously assembled literals. If an identical value exists, the address of the previously assigned literal is substituted in the object code.

(e) Address Expression:

Address expressions may be formed.

Examples:	LDA	TØM-2
	ENA	SYM+46B
	SYM EQU	275B

Subprogram, data and common relocatable symbols may be mixed.

NOTE: External symbols, the double asterisk and literals may not appear in an address expression.

2.2.3.4 COMMENTS Field

Comments may be included with any instruction. A blank column must separate them from the last character in the address field and they may extend to column 72.

Comments have no effect on assembly, but will be included on the assembly listing.

It is highly recommended that liberal use should be made of comments and comments cards.

An example is shown on the following page.

2.2.3.4 (cont.)

	IDENT	ØNE	
*			
* COMPASS	CØURSE	EXERCISE 3	
*	J. SMITH		
*			
	ENTRY	START	*SCOPE ENTRY*
START	UJP	**	
	ENQ	14B	
	ENA	0	INITIALIZATION
	STA	SUM	OF COUNTERS
	STA	CØUNT	
*			
*			
LØØP	LDA	SUM	
	ADA	CØUNT	
	STA	SUM	MAIN
	LDA	CØUNT	LØØP
	AQJ,EQ	START	
	INA	1	
	STA	CØUNT	
	UJP	LØØP	
*			
*			
*			
CØUNT	BSS	1	
SUM	BSS	1	
	END	START	

2.2.4 Coding Simple COMPASS Programs

2.2.4.1 Beginning the Program

The first card in the deck must be an IDENT card. This card has the word IDENT punched beginning at Column 10, and the program name (of 1-8 alphanumeric characters, the first of which must be alphabetic) punched beginning at Column 20.

Examples:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45											
										IDENT										TEST4Z																																				
or										IDENT										G4Z7																																				
or										IDENT										ØPCØDER																																				

2.2.4.2 Ending the Program

The last card in the deck must be an END card. The word END is punched beginning at Column 10, and the entry point (the beginning instruction in the program) is punched beginning at column 20. (However, see later section on END statement in relation to programs containing more than one sub-program).

Examples:

	1	10	20
		END	START
or		END	BEGIN
or		END	LØØPIN

2.2.4.3 SCOPE Entry

So that the program can be run under SCOPE, it is necessary to ensure that control returns to SCOPE when the program is finished. The entry point into the program should take the following form to enable this to be done:

	1	10	20
	START	UJP	**
or	BEGIN	UJP	**
or	LØØPIN	UJP	**

This will be the first executable statement in the program. SCOPE will enter the return address into the location START instead of the two asterisks. The last executable instruction in the program should be a jump back to the beginning so that control can be returned to SCOPE.

Examples:

	1	10	20
	TØM	UJP	START
or	TØM	UJP	BEGIN
or	TØM	UJP	LØØPIN

2.2.4.4 Deck Structure for COMPASS Course Exercises

(i) Assembly only

$\begin{matrix} 7 \\ 9 \end{matrix}$ SEQUENCE,666

$\begin{matrix} 7 \\ 9 \end{matrix}$ JOB,1112115,6404,5

$\begin{matrix} 7 \\ 9 \end{matrix}$ COMPASS,L

{ COMPASS source deck

FINIS (punching begins in column 10)

$\begin{matrix} 77 \\ 88 \end{matrix}$ (end of file card)

(ii) Assembly and execution:

$\begin{matrix} 7 \\ 9 \end{matrix}$ SEQUENCE,666

$\begin{matrix} 7 \\ 9 \end{matrix}$ JOB,1112115,6404,3

$\begin{matrix} 7 \\ 9 \end{matrix}$ EQUIP,56=MT

$\begin{matrix} 7 \\ 9 \end{matrix}$ COMPASS,L,X

{ COMPASS source deck

FINIS

$\begin{matrix} 7 \\ 9 \end{matrix}$ LOAD,56

$\begin{matrix} 7 \\ 9 \end{matrix}$ RUN,4

Data

$\begin{matrix} 77 \\ 88 \end{matrix}$ (end of file card)

(iii) Alternative "assembly only".

Use all cards as for execution, except RUN and DATA.
Any LOADER errors will then be indicated, but the
program will not be executed.

NOTE: 1. The $\begin{matrix} 7 \\ 9 \end{matrix}$ is a multiple punching in column 1 of the card.

2. The end of file card has a multiple 7,8 punching in
columns 1 and 2. It signifies the end of the job.
(EOF cards used on other systems that use

$\begin{matrix} 12 \\ 12 \end{matrix}$

$\begin{matrix} 1 \\ 1 \end{matrix}$

$\begin{matrix} 4 \\ 4 \end{matrix}$

in column 1 thru 4 are also acceptable.)

$\begin{matrix} 7 & 7 & 7 & 7 \end{matrix}$

$\begin{matrix} 8 & 8 \end{matrix}$

STUDENT NOTES

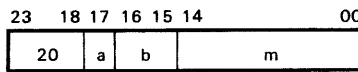
COMPASS INSTRUCTIONS

- 3.1 LOAD INSTRUCTIONS
 - 3.1.1 Load A
 - 3.1.2 Load Q
 - 3.1.3 Load Index
- 3.2 STORE INSTRUCTIONS
 - 3.2.1 Store A
 - 3.2.2 Store Q
 - 3.2.3 Store Index
- 3.3 ARITHMETIC, FIXED POINT, 24-BIT PRECISION
 - 3.3.1 Add to A
 - 3.3.2 Subtract from A
 - 3.3.3 Multiply A
 - 3.3.4 Divide A
 - 3.3.5 Replace Add
- 3.4 REGISTER OPERATIONS WITHOUT STORAGE REFERENCE
 - 3.4.1 Increase A
 - 3.4.2 Increase Q
 - 3.4.3 Increase Index
 - 3.4.4 Enter Register
- 3.5 JUMP INSTRUCTIONS
 - 3.5.1 Unconditional Jump
 - 3.5.2 Compare A with Zero, Jump
 - 3.5.3 Compare A with Q, Jump
 - 3.5.4 Return Jump
 - 3.5.5 Unconditional Halt
 - 3.5.6 Selective Jump
 - 3.5.7 Index Jump (Incremental/Decremental)
- 3.6 SKIP INSTRUCTIONS
 - 3.6.1 Skip if Equal
 - 3.6.2 Skip if Greater Than or Equal
 - 3.6.3 Index Skip Incremental/Decremental
- 3.7 STORE WORD ADDRESS
- 3.8 SHIFT INSTRUCTIONS
 - 3.8.1 Shift Instruction Format
 - 3.8.2 Shift A and Shift Q

3.1 LOAD INSTRUCTIONS

3.1.1 Load A

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	LDA, I	m, b



a = addressing mode designator

b = index designator

m = storage address.

Description : Loads A Register with the 24-bit contents of storage address M, where $M = m + (b^b)$.
NOTE: () indicates contents of.

Examples : (i) If the contents of location 100_8 in memory is 52307777, what will be the contents of A after execution of the following statement?

LDA 100B

Answer : (A) = 52307777

(ii) If location 100_8 is called by the symbolic address $L\phi C$, what will be the contents of A after execution of the following statement?

LDA $L\phi C$

Answer : (A) = 52307777

(iii) A block of memory in octal is as follows:

$L\phi C$	00	00	00	01
	00	00	00	10
	00	00	01	00
	00	00	10	00
	00	01	00	00
	00	10	00	00

What will be the contents of A after execution of the following statement?

LDA $L\phi C + 4$

Answer : (A) = 00 01 00 00

Exercises on Load A:

(i) $L\phi C$ LDA **

LDA $L\phi C$

What will be the contents of A at the end of the above section of a program?

(The function code for LDA = 200).

(ii) If location 101_8 contains

101 010	000 011	111 110	001 100
---------	---------	---------	---------

(in binary)

3.1.1 (cont.)

What is the octal amount loaded in A as a result of the following instruction?

LDA 101B 52037614

(iii) If MASK is location 105₈, and a block of memory is as follows,

104	1234 1234
105	7023 1307
106	1111 2222
107	6073 4261
110	3333 3333
111	4444 3066

What will be in A after execution of the following instruction?

LDA MASK+3

(iv) The instruction to be executed is:

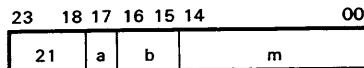
LDA =D43

Which answer gives the correct contents of A after the instruction has been executed?

- a. 43₈
- b. 35₁₀
- c. The contents of location 43
- d. 53₈

3.1.2 Load Q

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	LDQ, I	m, b



a = addressing mode indicator

b = index designator

m = storage address.

Description : Load Q with a 24-bit quantity from storage address M where $M = m + (B^b)$

Examples : (i) LDQ MASK

Load Q with the 24-bit quantity at the symbolic address MASK.

(ii) If MASK contains 00001111

LDQ MASK

(Q) = 0000 1111

3.1.2 (cont.1)

- (iii) If MASK = location 104₈ and a block of memory is as shown, what will be the contents of Q after execution of the following instruction?

LDQ	MASK-4	
76	14 00 00 20	
77	25 25 52 52	
100	11 22 33 44	
101	55 66 77 00	
102	17 53 17 53	
103	20 64 20 64	
104	31 75 31 75	
105	42 06 42 06	
106	53 17 53 17	
107	64 20 64 20	

(Q) = 11 22 33 44

Exercises on Load Q

- (1) What will be the contents of Q after execution of each of the following instructions? (The block of memory to be used is shown below).

00267	00 00 01 01
	20 10 40 40
	11 11 17 17
	20 20 20 20
	16 25 34 07
LPC	60 60 60 60
	21 22 23 24
	10 00 00 10

- (a) LDQ 272B
- (b) LBQ 184
- (c) LDQ LPC-3
- (d) LDQ LPC+2

- (2) If PLACE = location 2617₈, and a block of memory is as shown,

	01 47 26 35
	12 50 37 46
	23 61 40 57
02616	34 72 51 60
	45 03 62 71
	56 14 73 02
	67 25 04 13
	70 36 15 24
	01 47 66 65

3.1.2 (cont.2)

What will be the contents of Q after execution of each of the following instructions?

- (a) LDQ 2613B
- (b) LDQ PLACE-1
- (c) LDQ PLACE+3
- (d) LDQ 1427

3.1.3 Load Index

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	LDI, I	m, b

23	18	17	16	15	14	00
54	a	b	m			

a = addressing mode designator
 b = index designator
 m = storage address

Description : Load Index Register B^b with the lower 15 bits of the contents of storage address m. No address modification using index registers is possible.

"b" indicates which index register is to be loaded.

Examples :

- (i) If (CIØ) = 12345670

What will be the contents of index register 2 as a result of the following instruction?

LDI CIØ, 2

Answer: B₂ = 45670

- (ii) A block of memory is as shown:

LØC	0 0 0 0 0 0 0 1
	7 7 7 7 7 7 7 7
	2 3 4 2 3 4 2 3
	1 0 4 1 0 4 0 4
	4 4 4 4 1 0 0 0
	2 0 0 2 2 1 2 4
	4 1 2 3 4 4 1 4

The index registers are as follows:

B1	0 0 0 0 2
B2	0 0 0 1 4
B3	7 7 7 7 3

What will be the contents of the registers after the following?

LDI LØC+3, 3
 LDI LØC+6, 1
 LDI LØC, 2

3.1.3 (cont.)

Answer: $B_1 = 34414$
 $B_2 = 00001$
 $B_3 = 10404$

Exercises on Load Index instruction

- (1) What will be the contents of the index registers after execution of each of the following instructions, if

(CAB) = 00100001
(STØRE) = 77777771
(TEMP) = 40100010
(HIGH) = 20002020
(LØC3) = 33003303

- (a) LDI CAB,1
(b) LDI HIGH,3
(c) LDI STØRE,2
(d) LDI TEMP,1
(e) LDI LØC3,3
(f) LDI HIGH,2

- (2) A block of memory is as shown:

405	00 00 00 21
	04 04 04 04
	32 57 14 13
	51 21 41 61
	00 00 00 00
	21 04 00 10
	12 74 11 11
	00 07 77 75

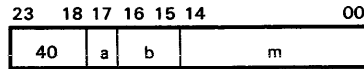
If TAG = Location 410_8 , what will be the contents of the Index registers after execution of the following instructions?

- (a) LDI TAG-2,3
(b) LDI TAG+4,1
(c) LDI 411B,2
(d) LDI TAG,1
(e) LDI TAG+2,3

3.2 STORE INSTRUCTIONS

3.2.1 Store A

LOCATION	OPERATION	MODIFIERS	ADDRESS FIELD
	STA, I		m, b



a = addressing mode designator

b = index designator

m = storage address.

Description : Stores the contents of the A Register in storage location M. ($M = m + (B^b)$). Contents of A are unchanged.

Examples : (i) STA HØLD
The value in A is stored in the location specified by the symbol HØLD.

(ii) (A) = 1212 3434
TEMP = Location 101₈
HØLD = Location 103₈

100	00 00 00 00	
101	11 11 11 11	TEMP
102	22 22 22 22	
103	33 33 33 33	HOLD
104	44 44 44 44	
105	55 55 55 55	
106	66 66 66 66	
107	77 77 77 77	
110	00 00 00 00	

What will be contained in the above block after the following :

```
LDA    100B
STA    TEMP+2
LDA    HØLD-2
STA    106B
LDA    HØLD+5
STA    TEMP
```

Answer:

100	00 00 00 00
101	00 00 00 00
102	22 22 22 22
103	00 00 00 00
104	44 44 44 44
105	55 55 55 55
106	11 11 11 11
107	77 77 77 77
110	00 00 00 00

3.2.1 (cont.)

Exercises on STA Instruction

- (i) If TEMPY is location 302_8 , and the block of memory shown results from the following instruction

STA TEMPY+4

What were the contents of A before execution of the instruction?

301	7723 7723
302	1111 1111
303	2222 2222
304	3044 6133
305	2442 4224
306	1212 3434
307	7777 5555

- (ii) If HOLD = Location 103_8 and TAG = Location 101_8 , what will be the contents of the block of memory below after execution of the following instructions?

LDA HOLD-1
STA TAG+3

100	0000 0000
101	1111 1111
102	2222 2222
103	3333 3333
104	4444 4444
105	5555 5555
106	6666 6666

- (iii) If the above instructions were followed by

LDA 101B
STA 69
LDA TAG+5
STA TAG-1
LDA HOLD
STA HOLD+3

What would be the final contents of the block of memory?

3.2.2 Store Q

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	STQ, I	m, b

23	18	17	16	15	14	00
41	a	b	m			

a = addressing mode designator
b = index designator
m = storage address

3.2.2 (cont.)

Description : Stores the 24-bit quantity in Q at address M,
where $M = m + (B^b)$

Example : LDQ *+1
 STQ RESULT (= 41 00 05 00)
 .
 .
 .
 .
 ORGR 500B
RESULT BSS 1

What would be the contents of RESULT after execution of the above?

Answer: RESULT = 41000500

Exercise :

A block of memory is as shown:

LPC	20 00 01 00
	00 00 04 01
	00 04 44 00
	21 23 25 27
MASK	07 04 07 04
	12 34 56 70
	00 00 00 01

What will be its contents after execution of the following?

LDQ LPC+3
STQ LPC+1
LDQ LPC+2
STQ MASK+1
LDQ MASK-4
STQ LPC+6

3.2.3 Store Index

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	STI, I	m, b

23	18	17	16	15	14	00
47	a	b	m			

a = addressing mode designator

b = index designator

m = storage address

Description: Store the (B^b) in the lower 15 bits of storage address m.
N.B. The upper 9 bits of m remain unchanged. "b" indicates the index register. If b = 0, the lower 15 bits of "m" are set to zero.

3.2.3 (cont.)

Examples: (i) STI TMP2,2 where (TMP2) = 40050012
(B²) = 63636

After execution (TMP2) = 40063636

(ii) If (FIELDA) = 31244444
(CHANGE) = 11111111

What will be the contents of TEMPY after execution of the following?

```
LDA    FIELDA
STA    TEMPY
LDI    CHANGE,3
STI    TEMPY,3
```

Answer: (TEMPY) = 31211111

(iii) If (B¹) = 00502

```
STA    **
LDA    *-1
STA    HØLD
STI    HØLD,1
```

What will be the contents of HØLD?

Answer: (HØLD) = 4000 0502

Exercises on Store Index instruction

(1) If (LØC) = 21043072
(B²) = 00004

What will be contents of LØC after execution of the following?

```
STI    LØC,2
```

(2) If (LØC) = 21043072
(TEMP) = 00140001

What will be the contents of LØC and TEMP after execution of the following?

```
LDA    LØC
LDI    TEMP,2
STA    TEMP
STI    TEMP,2
```

(3) If (TEMP) = 23002000, (BIG) = 77777772, and (HØLD) = 00000001,

What will be contents of TEMP and BIG after execution of the following?

```
LDA    TEMP
STA    HØLD
LDI    BIG,3
STI    HØLD
LDI    TEMP,3
STI    BIG,3
LDA    HØLD
STA    TEMP
```

3.3 ARITHMETIC, FIXED POINT, 24-BIT PRECISION

3.3.1 Add to A

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	ADA, I	m, b

23	18	17	16	15	14	00
30	a	b	m			

a = addressing mode designator

b = index designator

m = storage address

Description: Adds the 24 bit quantity located at address M to the contents of Register A. The result is stored in A.
 $M = m + (B^b)$.

Examples: (i) If $(100_8) = 10001010$

(A) = 00077777

What will be the contents of A after execution of the following instruction?

ADA 100B

Answer: (A) = 10101007

(ii) If $\emptyset PR =$ location 100_8 in the block of memory shown, what will be the contents of A and index register 2 after execution of the following?

100	0 0 0 0 0 0 1 1
101	0 0 1 4 1 3 1 0
102	2 1 0 0 0 0 0 5
103	1 4 0 1 4 0 0 0
104	0 0 7 7 1 7 7 7
105	2 0 2 0 1 0 2 0

$B_2 = 00014$

LDA $\emptyset PR+2$
 STA $\emptyset PR+4$
 LDA $\emptyset PR+3$
 ADA $\emptyset PR+4$
 STA $\emptyset PR+2$
 STI $\emptyset PR+2, 2$
 LDA $\emptyset PR+5$
 ADA $\emptyset PR+2$
 LDI $\emptyset PR+1, 2$

Answer:

(A) = 55201034

(B_2) = 41310

3.3.1 (cont.)

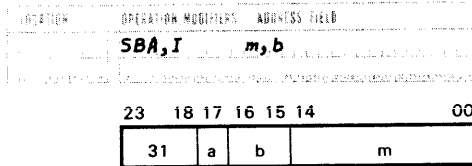
Exercise:

A block of memory is as shown:

100	00 00 00 01
	00 00 00 02
	00 00 00 03
	00 00 00 04
	00 00 00 05
	00 00 00 06
	00 00 00 07
	00 00 00 10

- (a) Write a program segment which will add together the contents of locations 100, 102, 104 and 106 and store the result in 107.
- (b) Write a program segment which will add up locations 101 and 102 and store the result in 103, add 103 and 104 and store the result in 105, and add 105 and 106 and store the result in 107.

3.3.2 Subtract from A



a = addressing mode designator
 b = index designator
 m = storage address

Description: Subtracts the 24-bit quantity located at address M from (A). The difference appears in A. $M = m + (B^b)$.

Examples: (i) If (A) = 10404040
 (100₈) = 10303030
 SBA 100B

Answer: (A) = 00101010

(ii) If (A) = 04444444
 (TEMP) = 02132132
 SBA TEMP

Answer: (A) = 02312312

3.3.2 (cont.)

(iii)

TEMP	1 0 0 0 0 0 0 1
	2 4 0 0 1 0 2 0
	7 7 7 7 7 7 7 7
	1 4 0 2 1 0 2 0
	0 0 0 0 1 0 0 0
	0 0 0 0 2 2 2 2
	0 2 0 2 0 2 0 2

What will be in this section of memory after the following?

```
LDA    TEMP
ADA    TEMP+6
STA    TEMP+2
SBA    TEMP+4
STA    TEMP+5
```

Answer:

1 0 0 0 0 0 0 1	TEMP
2 4 0 0 1 0 2 0	
1 2 0 2 0 2 0 3	
1 4 0 2 1 0 2 0	
0 0 0 0 1 0 0 0	
1 2 0 1 7 2 0 3	
0 2 0 2 0 2 0 2	

Exercises on Subtract from A

(a) LDA =D401
SBA =D30

What will then be the contents of A?

(b) If (CODE) = 00000001

What will be the contents of HOLD after the following?

```
LDA    CODE
ADA    CODE
STA    HOLD
ADA    HOLD
STA    HOLD
ADA    HOLD
ADA    HOLD
STA    HOLD
```

(c) Write a program segment to multiply the contents of X by two and subtract from the answer, the contents of locations Y and Z. Store your answer in ANSWER.

3.3.3 Multiply A

LOCATION	OPERATION/MODIFIERS	ADDRESS FIELD
	MUA, I	m, b

23	18	17	16	15	14	00
50	a	b	m			

a = address mode designator

b = index designator

m = storage address

Description: Multiply the contents of A by the contents of Address M. The 48-bit product appears in QA with the lowest order bits in A.

$$M = m + (B^b)$$

Examples: (i) What will be the contents of QA after execution of the following instructions? A contains 100_8 , and TABLE contains 5.

```

STA    SUM24
LDA    TABLE
MUA    SUM24
    
```

Solution: SUM24 =

00	00	01	00
----	----	----	----

A =

00	00	00	05
----	----	----	----

Answer =

Q	A
00 00 00 00	00 00 05 00

(ii) If TABLE contains 5 and A contains 20000000_8 , what will be the contents of QA after execution of the following instructions?

```

STA    SUM24
LDA    TABLE
MUA    SUM24
    
```

Solution: SUM24 =

20	00	00	00
----	----	----	----

A =

00	00	00	05
----	----	----	----

Answer =

Q	A
00 00 00 01	20 00 00 00

Exercises on Multiply A Instruction

(a) If (A) = 10, (TABLE) = 3, (SUM24) = 5, and (HOLD) = 0 what will be the contents of A after execution of the following instructions?

```

MUA    TABLE
STA    HOLD
LDA    SUM24
MUA    TABLE
ADA    HOLD
    
```


3.3.3 (cont.)

(b) A block of memory is as shown

100	00 00 00 01
	00 00 00 02
	00 00 00 03
	00 00 00 04
	00 00 00 05
	00 00 00 06
	00 00 00 07
	00 00 00 10

Write a program segment to add the contents of 100 and 101, multiply the result by the contents of 102, add the contents of 103, 104 and 105, multiply the result by (106), and then add (107). Store the result in 100.

(c) What will be the contents of A after execution of the following sequence of instructions, if the initial contents of A = 0, (ONE) = 6, (TWO) = 2, (THREE) = 10₁₀.

```

ADA      ONE
MUA      TWO
SBA      THREE
MUA      ONE
MUA      ONE
SBA      TWO
ADA      THREE
    
```

3.3.4 Divide A

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	DVA, I	m, b

23	18	17	16	15	14	00
51	a	b	m			

a = addressing mode designator

b = index designator

m = storage address

Description: Divide the 48 bit quantity in AQ by the quantity in storage at address M ($M = m + (B^b)$). The quotient appears in A, and the remainder with its sign extended appears in Q.

If a divide fault* occurs, the instruction halts and the program advances to P + 1. (The contents of A and Q are usually meaningless in this case.) A divide fault occurs whenever the number of leading sign bits (0 or 1, ie, + or -) in M is greater than or equal to the number of leading sign bits in AQ. A fault can also occur if the number of sign bits in M is one less than that in AQ; however, the actual number in M and in AQ now has to be considered. Since the number in AQ is usually achieved by shifting AQ 24 bits to the right, extending the sign thru A into Q, an overflow rarely occurs.

*Quotient greater than $2^{23}-1$

3.3.4 (cont.)

Example:

If (AQ) =

00000000	37777777
----------	----------

and (D) =

00000001

What will be in A and Q after execution of:

DVA D

Answer: (A) =

37777777

 = Quotient

(Q) =

00000000

 = Remainder

Divide Fault bit = 0, ie, 'No Divide Fault'

Note: Had the initial contents of AQ been just one number larger, a Divide Fault would have occurred.

Exercises:

- (a) If (A) = 0000 0000
 (Q) = 0000 4040
 (TAG) = 0000 2000

What will be in A and Q after execution of the following instruction?

DVA TAG

- (b) If (A) = 00000000
 (Q) = 00000122
 (LOC) = 00000005

What will be in A and Q after execution of the following instruction?

DVA LOC

- (c) If (A) = 7777777
 (Q) = 7777771
 and Index Register 2 contains 00003

TAG	0000	0007
	0014	0140
	0771	7717
	0000	0004
	7777	7773

What would be in A and Q after the instruction:

DVA TAG,2

- (d) If Index Register 2 contains 4, TAG is as above, and (A) and (Q) are as shown:

A	Q
7777	7777
7777	7771

What would be in A and Q after execution of the following instruction?

DVA TAG,2

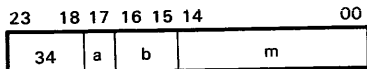
- (e) Draw diagrams of the machine instruction word which would be assembled from the following COMPASS instructions:

(i) DVA 474B,1

(ii) DVA 100B

3.3.5 Replace Add

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	RAD, I	m, b



a = address mode designator

b = index designator

m = storage address.

Description: Replace the quantity at address M with the sum of (M) and the contents of A register. $M = m + (B^b)$.

The A register remains unchanged.

Examples: (i) If $A = 67432136_8$ and

(INC) = 100_8

STA	LØC
LDA	INC
RAD	LØC

LØC now equals 67432236B.

(ii) If $(A) = 200_8$ and (INC) = 500_8

RAD	INC
-----	-----

Answer: (INC) = 700_8

Exercise:

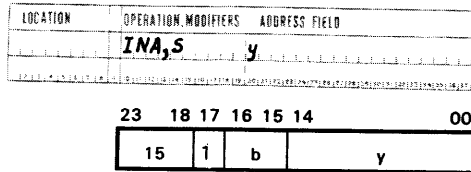
If $(A) = 100_8$ and (INC) = 200_8

RAD	INC
STA	TEMP
LDA	INC
STA	ØNE
LDA	TEMP
RAD	INC
LDA	INC
STA	TWØ
LDA	TEMP
RAD	INC
LDA	INC
STA	THREE

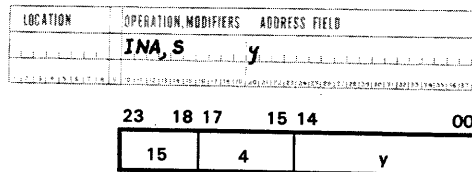
What will be the contents of ØNE, TWØ and THREE?

3.4 REGISTER OPERATIONS WITHOUT STORAGE REFERENCE

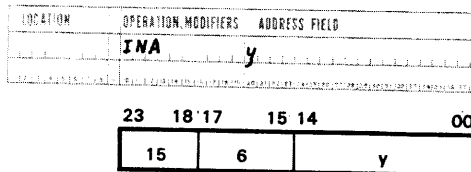
3.4.1 Increase A



b = 0 if sign extension



b = 2 if no sign extension



Description: Adds the amount "Y" to the contents of the A register. If there is no sign extension specified, only the 15 bits of "Y" are added to A. If sign extension is specified, the sign bit of y is extended before addition to A. (Note - if A is increased by a negative number, sign extension should be used.)

Examples: (i) If (A) = 00 00 00 01
 INA 12B
 (A) = 00 00 00 13

(ii) If (A) = 00 00 00 01
 INA, S 4000B
 (y = 777 4 0000)

Answer : (A) = 77740001

Exercises: (ERRØRC) = 00004040B

(a) What will be the contents of ERRØRC after the following?

```
LDA ERRØRC
INA 55565B
STA ERRØRC
```

(b) If the instructions had been

```
LDA ERRØRC
INA, S 55565B
STA ERRØRC
```

What would have been the final contents of A?

3.4.1 (Cont.)

- (c) If $(ERR/RC) = 60_{10}$, write the coding necessary to increase it to 100_{10} .
- (d) If $(A) = 10_8$, what will be the final contents of A after execution of the following?

INA,S 77776B

3.4.2 Increase Q

LOCATION	OPERATION	MODIFIERS	ADDRESS FIELD
	INQ,S		y

23	18	17	16	15	14	00
15	1	b	y			

b = 1 if sign extension

LOCATION	OPERATION	MODIFIERS	ADDRESS FIELD
	INQ,S		y

23	18	17	15	14	00
15	5	y			

b = 3 if no sign extension

LOCATION	OPERATION	MODIFIERS	ADDRESS FIELD
	INQ		y

23	18	17	15	14	00
15	7	y			

Description: Adds the amount "Y" to the contents of the Q register. If there is no sign extension specified, only the 15 bits of "Y" are added to Q. If sign extension is specified, the sign bit of y is extended before addition to Q. (note: if Q is increased by a negative number, sign extension should be used.)

- Examples:
- (i) If $(Q) = 00\ 00\ 00\ 20$
 INQ 60B
Answer : $(Q) = 00\ 00\ 01\ 00$
- (ii) If $(Q) = 00\ 00\ 00\ 20$
 INQ,S 44444B
 (y = 777 44444)
Answer : $(Q) = 777\ 44464$

Exercises:

(i) If $(Q) = 577_8$, what will be the result in Q after the instruction INQ 547B is executed?

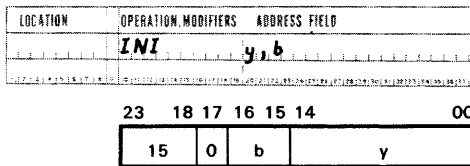
3.4.2 (cont.)

- (ii) If (Q) = 50B, (STAGK) = 500B what will be the result in Q after the following sequence of instructions?

```

      INQ      100B
      STQ      TEMP
      LDA      TEMP
      RAD      STACK
      LDQ      STACK
      INQ,S    27711B
  
```

3.4.3 Increase Index



b = index register designator

Description: Adds the 15-bit quantity "y" to the contents of the index register specified.

Examples: (i) INI 6,2 where (B²) = 12B

What will be the contents of B²?

(B²) = 20B

(ii) LDI 100B,3

INI 10B,3

What will be the contents of B³?

(B³) = (100B) + 10B.

- (iii) If a block of memory is as shown below, what will be in the index registers as a result of the following:

107	20	00	00	04
	40	10	00	21
	60	00	00	01
	77	77	77	77
	00	00	00	20
	04	01	07	01
	22	22	33	33

LDI 110B,1

LDI 111B,2

INI 20B,2

LDI 113B,3

INI 70B,3

Answer: B₁ = 21B
 B₂ = 21B
 B₃ = 110B

3.4.3 (cont.)

Exercises on Increase Index instruction

(i) A block of memory is as shown:

100	00 00 00 01
	00 00 00 10
	00 00 01 00
	00 00 10 00
	00 01 00 00
	00 10 00 00
	01 00 00 01

What will be the contents of the three index registers after execution of the following?

LDA 101B
RAD 100B
LDI 100B,1
INI 100B,1
LDI 104B,2
INI 17777,2
LDI 106B,3
INI 100B,3
STI 106B,3
LDA 106B
RAD 106B
LDI 106B,3

(ii) If index 1 contains 1, write a program using this index to store the numbers 1, 3, 5, 7 and 8 in 5 consecutive locations beginning at location 100.

3.4.4 Enter Register

23	18	17	16	15	14	00
14	d	b	y			

- (a) d = 1
b = 2

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	ENA	y

23	18	17	15	14	00
14	6	y			

- (b) d = 1
b = 0

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	ENA,S	y

23	18	17	15	14	00
14	4	y			

Enter A

- (c) d = 1
b = 3

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	ENQ	y

23	18	17	15	14	00
14	7	y			

Enter Q

- (d) d = 1
b = 1

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	ENQ,S	y

23	18	17	15	14	00
14	5	y			

- (e) d = 0
b = index designator

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	ENI	y, b

23	18	17	16	15	14	00
14	0	b	y			

Enter Index

If b=0, this is a no-operation instruction.

3.4.4 (cont.)

Description: The 15-bit quantity "y" is entered in the specified register. The register is cleared before "y" is entered. If sign extension is specified, the sign bit of "y" is extended before "y" is entered into the register.

Examples:

- (i) ENA 14B
(A) = 00 00 00 14
- (ii) ENQ,S 40001B
(Q) = 777 40001
- (iii) ENI 77B,1
(B¹) = 00077

Exercises on the Enter Register Instructions

- (1) If (Q) = 11025321, and location 70₈ contains 10101₈, what will be the contents of Q after execution of the following?

ENQ 70B

- (2) AA ENA 0
LDA *-1
INA 40
STA AA

What will be the contents of storage location "AA" after execution?

- (3) If (FIELD) = 1010 1010

and FIELD = locaton 456B

What will be the contents of the A Register after execution of the instruction

ENA FIELD

- (4) ENA,S 0
STA 100B
ENI 10B,1
INI 100B,1
STI 100B,1
LDA 100B
INA,S 40000B
STA 100B
LDI 100B,2

What will be the contents of A, B¹ and B² after execution of the above?

3.5 JUMP INSTRUCTIONS

3.5.1 Unconditional Jump

LOCATION	OPERATION	MODIFIERS	ADDRESS FIELD
	UJP, I	m, b	

23	18	17	16	15	14	00
01	a	b	m			

a = addressing mode designator

b = index designator

m = storage address

Description: Unconditionally jump to address M, where $M = m + (B^b)$.

Examples:

```
(i) EOF      LDA      100B
      .
      .
      .
      UJP      EOF
```

Control will jump back to statement EOF

```
(ii) AAA      UJP      *+2      (AAA = address i0)
      Program jumps to address 12.
```

Where the address to which control is to jump is not known at the time of assembly, but will be entered in during execution, the instruction is usually coded as follows, to enable easy recognition of the statement in assembled programs:

```
UJP      **      (assembled as 01077777)
```

This causes 1 bits to be set in the address portion of the word, and this will be replaced during execution by the actual address to which control is to jump.

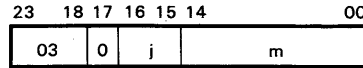
SCOPE entry to a program takes this form. (See section 2.2.4.3)

However, in such cases, the contents of the address portion of the word may be any legal address. Whatever the address is, it will be replaced by the correct address during execution. Thus:

```
UJP      **
UJP      *
UJP      *+7
START    UJP      START
```

will all serve the same purpose, for they will all be modified as required in the program, during execution.

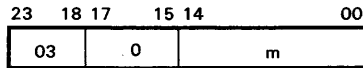
3.5.2 Compare A with Zero, Jump



j = jump designator (0-3)
m = jump address

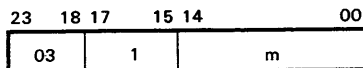
(a) j = 0, jump if A = +0 or -0

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	AZJ, EQ	m



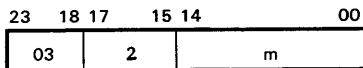
(b) j = 1, jump if A ≠ +0 or -0

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	AZJ, NE	m



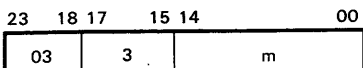
(c) j = 2, jump if A ≥ +0 (-0 < +0)

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	AZJ, GE	m



(d) j = 3, jump if A < +0 (-0 < +0)

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	AZJ, LT	m



Description: (A) are compared with zero to establish test conditions as above. If the condition specified by the modifier is true, the program jumps to address "m". If the test condition is not true, RNI from address P + 1. (RNI = read next instruction).

3.5.2 (cont.1)

Example: If (A) = 1

(i) What will be the RNI address after execution of

AZJ,EQ LØØP

Answer: RNI at address P + 1

(ii) If the instruction had been

AZJ,GE LØØP

Answer: RNI at address LØØP

Exercises:

Given the information below, give the RNI addresses for each problem.

(i) If (A) = 77777777

AZJ,EQ 100B

RNI = _ _ _ _ ?

(ii) If (A) = 00020000

AZJ,EQ STØRE

RNI = _ _ _ _ ?

(iii) If (A) = 00000000

AZJ,NE CARRY+1

RNI = _ _ _ _ ?

(iv) If (A) = 77777776

AZJ,LT NEXT

RNI = _ _ _ _ ?

(v) If (A) = 03675671

AZJ,GE ØNE

RNI = _ _ _ _ ?

(vi) If (A) = 00000000

AZJ,GE LØØP

RNI = _ _ _ _ ?

(vii) ENA,S 40001B

AZJ,GE YES

RNI = _ _ _ _ ?

(viii) ENA 77777B

AZJ,LT MAYBE

RNI = _ _ _ _ ?

(ix) ENA,S 20741B

AZJ,GE 200B

RNI = _ _ _ _ ?

(x) ENA,S 32767

AZJ,LT FINISH

RNI = _ _ _ _ ?

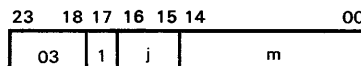
3.5.2 (cont.2)

(xi) Which Halt will be reached after execution of the following program segment?

```

LDA      =0423
SBA      =0424
AZJ,EQ   HLT
UJP      HLT+1
HLT      HLT      1
          HLT      2
    
```

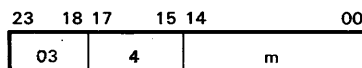
3.5.3 Compare A with Q, Jump



j = 0-3 jump designator (0-3)
m = jump address

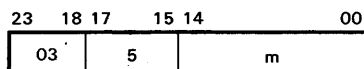
(a) j = 0, jump if A = Q (+0 = -0)

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	AQJ, EQ	m



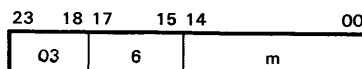
(b) j = 1, jump if A ≠ Q (+0 = -0)

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	AQJ, NE	m



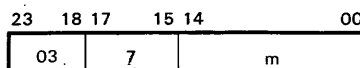
(c) j = 2, jump if A ≥ Q (+0 > -0)

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	AQJ, GE	m



(d) j = 3, jump if A < Q (+0 > -0)

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	AQJ, LT	m



3.5.4 (cont.1)

(ii)	RTJ	SUBRØUT	[100]
	-		
	-		
	-		
	-		
SUBRØUT	UJP	**	[200]
	-		
	-		
	-		
	UJP	SUBRØUT	[210]

If the numbers in brackets indicate the addresses of the respective instructions, the assembled program before execution would appear as:

00100	00700200
	-
	-
	-
00200	01077777
	-
	-
	-
00210	01000200

After execution the contents of memory would appear as:

00100	00700200
	-
	-
	-
00200	01000101
	-
	-
	-
00210	01000200

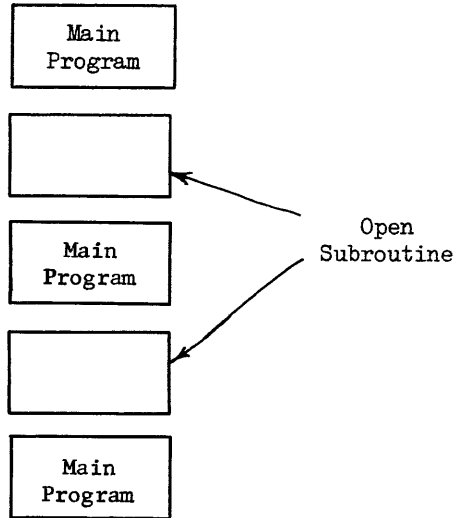
Note: ** is assembled as 77777₈ and is used where the value is to be changed at run time, such as the address portion of the instruction at address 00200. Or, to say it another way, ** = TBC (To Be Clobbered.)

3.5.4 (cont.2)

Use of the RTJ statement

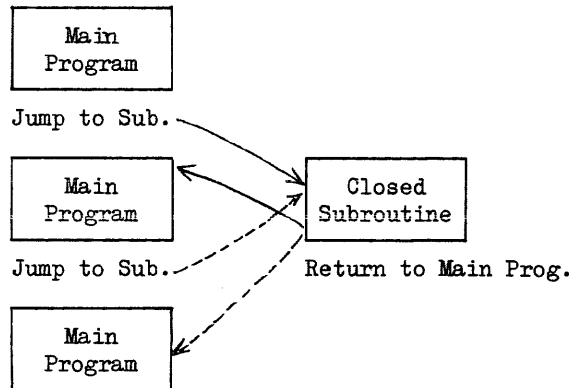
There are two types of subroutines used in COMPASS - open and closed. An open subroutine is a series of instructions which is required more than once during a program, and is inserted where it is required.

e.g.



This method has obvious disadvantages and it is more usual to employ the closed subroutine method.

e.g.



What is the problem here?

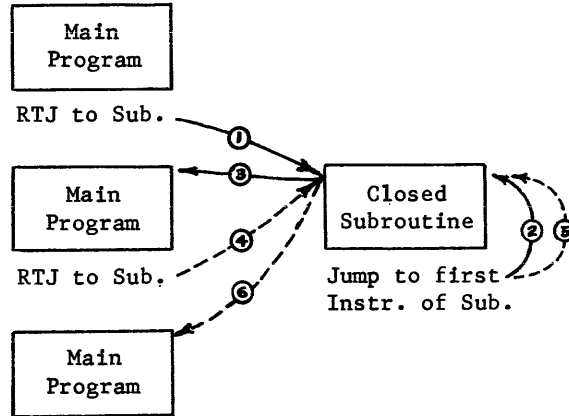
It is that the program jumps to the subroutine from two or more points in the main program, and once the program is in the subroutine, how does it know to which main program instruction it should return?

It doesn't, since it doesn't know where the jump to the subroutine was located in the Main Program.

3.5.4 (cont.3)

This problem is eliminated by using the RTJ (Return Jump) instruction in the Main Program when a jump to the subroutine is desired. The instruction stores the address of the instruction following the RTJ in the lower 15 bits of the first instruction of the subroutine. The first instruction is then skipped and the first instruction executed is actually the second instruction of the subroutine.

e.g.



Following is a section of a Compass Program that illustrates this latter method:

```

ENI      0,1
LDA      100B
SBA      101B
RTJ      SUB
LDA      105B
MUA      155B
RTJ      SUB
LDA      255B
INA      21
.
.
etc.
.
.
SUB     UJP      **
        AZJ.LT  SUB
        INI      1,1
        UJP      SUB

```

When the RTJ instruction is encountered, execution jumps to SUB, and the address portion of SUB is replaced by the address of the RTJ instruction + 1. After execution of the instructions in the subroutine, a jump is made back to SUB, which instruction is now an unconditional jump to the instruction after the RTJ instruction. Execution of the Main Program then continues.

3.5.5 Unconditional Halt

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	HLT	<i>m</i>

23	18 17	15 14	00
00	0	<i>m</i>	

Description: Unconditionally stop at this instruction. Upon restarting, RNI from address "m"

Example:

CTR.1	LDA	CTR.2
	INA	100B
	STA	T M,1
	ISE	10,1
	UJP	CTR.1-3
	HLT	SEVEN
SEVEN	LDQ	MASK
	END	

3.5.6 Selective Jump

23	18 17	15 14	00
00	<i>j</i>	<i>m</i>	

$j = 1 - 6 = \text{SELECTIVE JUMP switch number}$

$j = 1$, jump if "SELECT JUMP 1" switch is on

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	SJ1	<i>m</i>

$j = 2$, jump if "SELECT JUMP 2" switch is on

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	SJ2	<i>m</i>

$j = 3$, jump if "SELECT JUMP 3" switch is on

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	SJ3	<i>m</i>

$j = 4$, jump if "SELECT JUMP 4" switch is on

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	SJ4	<i>m</i>

$j = 5$, jump if "SELECT JUMP 5" switch is on

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	SJ5	<i>m</i>

$j = 6$, jump if "SELECT JUMP 6" switch is on

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	SJ6	<i>m</i>

$j = 0$, see HLT instruction
 $j = 7$, see RTJ instruction

3.5.6 (cont.)

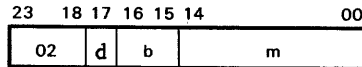
Description: Jump to address m if the jump key specified is set "on". Otherwise RNI P+1.

Example:

	SJ4	BYPASS
CARDS	LDA	CARD,1
	ADA	TEMP,1
	UJP	ENDING
BYPASS	LDA	TAPE,2
	ADA	STORE,2
ENDING	HLT	LOC4
	.	
	.	
	END	

If jump switch 4 is set "on" jump to BYPASS otherwise RNI at address CARDS.

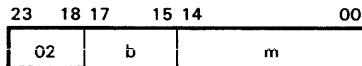
3.5.7 Index Jump (Incremental/Decremental)



b = index register designator

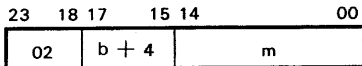
d = 0, Index Jump Increment

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	IJI	m, b



d = 1, Index Jump Decrement

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	IJD	m, b



Description: Jump to "m" if $(B^b) \neq 0$ and increment (or decrement) index by 1. There are 3 possible conditions:

- (i) If $b = 0$, the instruction is a no-op. and RNI from P + 1.
- (ii) If $(B^b) = 0$ RNI from P + 1.
- (iii) If $(B^b) \neq 0$, the jump test condition is satisfied. One is added or subtracted to (B^b) ; jump to address m and RNI.

N.B. The counting is done in a one's complement adder. Negative zero is not generated because the count progresses77775, 77776, 00000, stopping at +0. If a -0 is initially in B^b , the count progresses 77777, 00001, etc.

3.5.7 (cont.)

Examples:

```
ENI          9,2
LØØP        LDA          BUFFER,2
            STA          DATA,2
            IJD          LØØP,2
            HLT
```

How many words will transfer?

Answer: 10_{10}

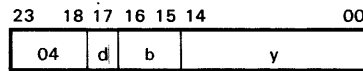
Exercises:

```
ENI          -5,1
LØØP        LDA          TØM
            .
            .
            .
            IJI          LØØP,1
            HLT
```

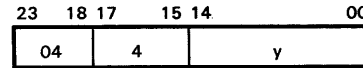
How many times will the loop be executed?

3.6 SKIP INSTRUCTIONS

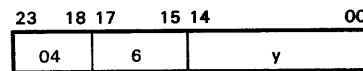
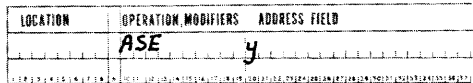
3.6.1 Skip if Equal



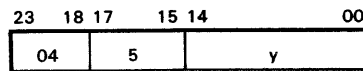
- (i) d = 1
b = 0



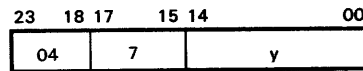
- (ii) d = 1
b = 2



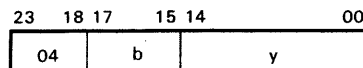
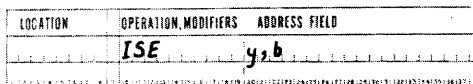
- (iii) d = 1
b = 1



- (iv) d = 1
b = 3



- (v) d = 0
b = index designator



b = index register designator (1-3)

If b=0, y is compared to zero.

3.6.1 (cont.1)

Description: If the instruction is ASE, QSE or ISE; ($A_{\text{lower } 15}$), ($Q_{\text{lower } 15}$) or (B^b) respectively is compared bit for bit to "y". If equal, RNI from P+2, otherwise RNI from P+1.

Note: To be equal to "y", the quantities must be exactly the same, thus if (A) or (Q) = xxx00000 or (B^b) = 00000, "y" must be 00000 to be equal. Or, if (A) or (Q) = xxx77777 or (B^b) = 77777, "y" must be 77777 to be equal.

If sign extension is specified, i.e. the instruction is ASE,S or QSE,S the sign of "y" (bit 14) is extended and the 24 bit quantity is compared with the quantity in the specified register.

Note: In this case, if (A) or (Q) = 77777777 or 00000000 and "y" is 77777 or 00000, the quantities are considered to be equal. For all other values in A or Q the sign extended "y" value must be exactly the same to be considered equal.

Examples:

(i) ISE 300B,2

If (B_2) = 300_8 , RNI P + 2.

(ii) ISE 50,1 (B^1) = 62_8

What is the RNI?

RNI = P + 2

(iii) BEGIN LDQ FIELDA

QSE 64210B

...
...
...

FIELDA OCT 50964210

What is the RNI after QSE?

RNI = P + 2 (=BEGIN + 3)

(iv) Example of use of ASE instruction in a loop.

```

        ENA          0
LOOP STA          CIC.1
        STA          TEMP
        LDA          *-2
        INA          1
        STA          *-4
        LDA          TEMP
        INA          5
        ASE          25
        UJP          LOOP
        HLT
TEMP BSS          1
    
```

This will store numbers 0, 5, 10, 15 and 20 in consecutive locations, beginning at location CIC.1.

3.6.1 (cont.2)

(iv) Use of Index skip-if-equal in a loop.

```

                                ENI            0,1
LOOP LDA            VALUE
                                MUA            =D5
                                DVA            =D8
                                STA            RESULT
                                LDA            LOOP
                                INA            1
                                STA            LOOP
                                LDA            *-4
                                INA            2
                                STA            *-6
                                INI            1,1
                                ISE            100,1
                                UJP            LOOP
                                HLT
VALUE  OCT            1
```

Exercises on SKIP-if-equal instructions

(a) How many times will the following loop be executed?

```

                                ENQ            0
LOOP LDA            ANS
                                MUA            TW
                                STA            ANS
                                INQ            5
                                QSE            100
                                UJP            LOOP
                                HLT
```

(b) Write a program segment to move ten numbers in consecutive locations, beginning at STORE, to consecutive locations beginning at RESULT.

(c) If CHECK contains 20100010, how many times will the following loop be executed?

```

                                LDQ            CHECK
LOOP LDA            TOM
                                INQ            1
                                .
                                .
                                .
                                QSE            12
                                UJP            LOOP
                                HLT
```


3.6.1 (cont.3)

(d) What will be the final contents of HOLD after execution of the following?

ENI	0,1
ENA	0
STA	HOLD
INA	10
INI	1,1
ISE	10,1
UJP	*-3
STA	HOLD

3.6.2 Skip if Greater Than or Equal

23	18	17	16	15	14	00
05	d	b	y			

(i) d = 1
b = 0

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	ASG, S	y

23	18	17	15	14	00
05	4	y			

(ii) d = 1
b = 2

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	ASG	y

23	18	17	15	14	00
05	6	y			

(iii) d = 1
b = 1

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	QSG, S	y

23	18	17	15	14	00
05	5	y			

(iv) d = 1
b = 3

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	QSG	y

23	18	17	15	14	00
05	7	y			

(v) d = 0
b = index designator

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	ISG	y, b

23	18	17	15	14	00
05	b	y			

3.6.2 (cont.)

Description:

If the instruction is ASG, QSG or ISG; ($A_{\text{lower } 15}$), ($Q_{\text{lower } 15}$) or (B^b) respectively is compared to "y". If greater than or equal, RNI from P+2, otherwise RNI from P+1.

Note: If the (A) or (Q) = xxx77777 or (B^b) = 77777, the instruction will skip if "y" is either 77777 or 00000. If the (A) or (Q) = 00000000 or (B^b) = 00000, the instruction will skip if "y" is 00000 but will not skip if it is 77777.

If sign extension is specified, i.e. the instruction is ASG,S or QSG,S the sign of "y" (bit 14) is extended and the 24 bit quantity is compared with the quantity in the specified register. If the sign bits are different the one with a sign bit of zero is larger.

Note: If the (A) or (Q) = 00000000, the instruction will skip if "y" is either 77777 or 00000. If the (A) or (Q) = 77777777, the instruction will skip if "y" is 77777 but will not skip if it is 00000.

Examples:

(i) If (A) = 00002020

ASG 2020B

(A) is equal to y, skip next instruction, RNI at P + 2

(ii) If (A) = 10000001

ASG,S 40002B

y = 77740002 with sign extended

(A) is greater than y, skip next instruction, RNI at P + 2

Exercises in SKIP-IF-GREATER-THAN-OR-EQUAL Instruction

(a) To which STOP location will control jump?

```
ENA,S 40000B
INA 20B
ASG,S 40040B
UJP STOP1
ASG 40020B
UJP STOP2
UJP STOP3
```

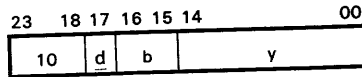
(b) To which location will control jump?

```
ENQ 200B
QSG,S 128
UJP ENDING
UJP PAUSE1
```

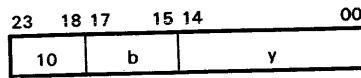
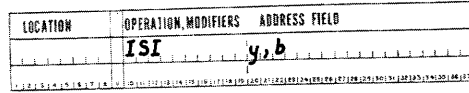
(c) How many times will the loop be executed?

```
ENI 0,1
LDA PERSON,1
MUA RATE1
SHAQ 24
DVA RATE2
STA TAX,1
INI 1,1
ISG 500,1
UJP *-7
HLT
```

3.6.3 Index Skip Incremental/Decremental

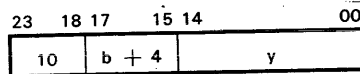
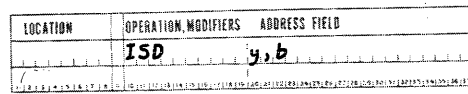


d = 0, Index Skip Incremental



b = index register designator

d = 1, Index Skip Decremental



b = index register designator

Description: If $(B^b) = y$ skip to $P + 2$ and clear the index register.
 If $(B^b) \neq y$ RNI from $P + 1$ and add 1 to (or subtract 1 from) the index register.

Examples: (i) ENI 0, 1
 LDA BUFFER, 1
 STA DATA, 1
 ISI 9, 1
 UJP *-3
 HLT

What is this series of instructions doing?

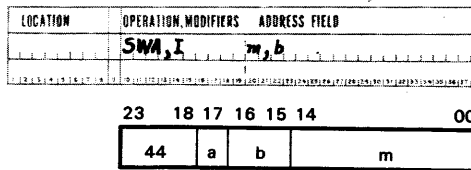
Answer: Moving 10 words from BUFFER to DATA

(ii) How can the same problem be done using the ISD instead of the ISI?

ENI 9, 1
 LDA BUFFER, 1
 STA DATA, 1
 ISD 0, 1
 UJP *-3
 HLT

Exercise: Write a program segment to move 50 numbers from BUFFER to locations beginning at STORE. FIND the total of these numbers and store the answer in RESULT.

3.7 STORE WORD ADDRESS



Description: Stores the lower 15 bits of (A) in the storage location M. The high - order 9 bits of (M) are unchanged.

Examples: (i) SWA INST (A) = 17603216
 :
 :
 INST LDA **

What will be the contents of INST?

(INST) = 20077777 (Before)

(INST) = 20003216 (After)

(ii) INST LDA *+1
 SWA *-1

Where INST = 76B, what will be the contents of words 76B and 77B?

(76B) = 20000076

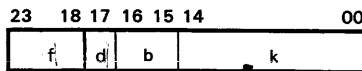
(77B) = 44000076

Exercise: What, if any, is the difference between the modification achieved by the two program segments below?

	<u>Program A</u>		<u>Program B</u>
	LDA F0UR06		LDA INST
	SWA INST		ENA 406
	.		STA INST
	.		.
INST	ENA **		.
	.		.
	.	INST	ENA **
F0UR06	0CT 406		

3.8 SHIFT INSTRUCTIONS

3.8.1 Shift Instruction format



d = an operation designator, which is virtually an extension of the function code. It differentiates between 2 instructions using the same f code.

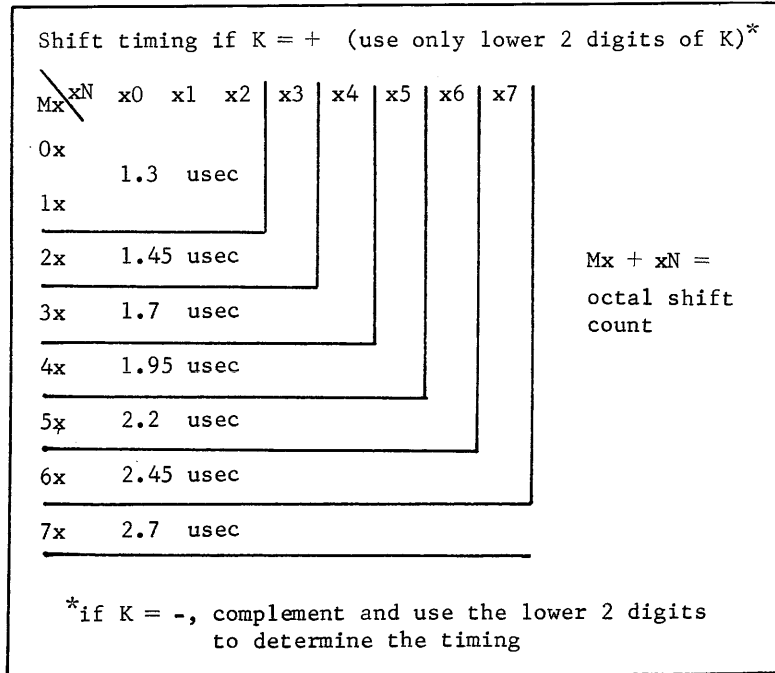
e.g. SHA d = 0
 SHQ d = 1

b = index register designator
 The contents of the index register is treated as a 15 bit signed number and is used as a shift direction and/or number-of-bits modifier.

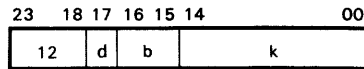
k = Shift count, base
 It may be positive or negative (in complement form) depending on the direction of the shift.

Left shift = positive
 Right shift = negative

K = shift count, actual
 $K = (B^b) + k$ if $b \neq 0$ otherwise $K = k$ if $b = 0$.
 If the sum is 7777₈ or larger, subtract 7777₈ to get the true sum, otherwise the sum is the true sum.
 If the true sum is between 0000 and 3777₈, the righthand two octal digits are the actual shift count for a left shift. If the true sum is between 40000 and 7777₈, the complement of the righthand two octal digits are the actual shift count for a right shift.

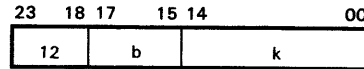


3.8.2 Shift A and Shift Q



d = 0, Shift A

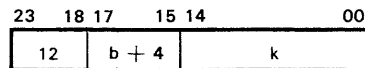
LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	SHA	k, b



b = index register designator

d = 1, Shift Q

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	SHQ	k, b



b = index register designator

Description: The 24-bit contents of the register are shifted according to the magnitude and sign of K (i.e., k + b)

If K is +, instruction is left shift end around
 If K is -, instruction is right shift end off

e.g. Shift left 6 positions: K = 00006

Shift right 6 positions: K = 77771

N.B. (i) During right shift, the sign bit is extended and the low order bits are discarded.

(ii) During left shift, the high order bits are brought end around.

(iii) (B^b) and k, with their signs extended, are added to give K. The computer then senses bits 00-05 and bit 23 to determine the size and direction of shift, respectively.

3.8.2 (cont.)

Examples of Shift A and Shift Q

- (i) SHA 6 (A) = 10203040
After Shift (A) = 20304010
- (ii) SHQ -6 (A) = 12345671
After Shift (A) = ?
(A) = 00123456
- (iii) SHA -6 (A) = 50367123
After Shift (A) = ?
(A) = 77503671

Exercises on Shift A and Shift Q

- (a) If (A) = 20000010

What will be the contents of A after

SHA 4
SHA -4

- (b) If (A) = 21354706

What will be the contents of A after

ENI 2001,1
SHA 8,1

- (c) If (Q) = 10000001

What will be the contents of Q after

SHQ 6

- (d) If (Q) = 12345670

What will be the contents of Q after each shift in the following?

ENI 307,2
SHQ 4,2
SHQ -13

- (e) If (LABEL) = 10421045

What will be the contents of LABEL after

LDA LABEL
SHA -15
STA LABEL
LDQ LABEL
SHQ 1404B
STQ LABEL

STUDENT NOTES

INSTRUCTION MODIFICATION

4.1 SIGN EXTENSION

4.2 ADDRESS MODES

4.3 INDEX MODIFICATION OF WORD ADDRESSING INSTRUCTIONS

4.1 SIGN EXTENSION

Certain instructions offer the option of extending the sign of a 15 or 17-bit operand by putting a modifier "S" in the operation field.

e.g. ENA, S 06370B (A) = 00006370
 ENQ, S 76432B (Q) = 77776432

Examples:

(i) ASE, S 77777B where (A) = 7777777

What is RNI address?

RNI from P + 2.

(ii) ENA, S -6 what will be (A)?

(A) = 77777771

(iii) ENQ, S 405B what will be (Q)?

(Q) = 00000405

(iv) ENA, S 7B what will be (A)?

(A) = 00000007

(v) ENQ, S 43125B what will be (Q)?

(Q) = 77743125

(vi) Where (Q) = 00054631B

QSE, S 54631B will result in the RNI being
(a) P + 1 or (b) P + 2?

Answer : P + 1

(vii) ASG, S 43671B where A = 43671234

what is RNI?

Answer : "y" extended = 77743671

(A) is not greater than or equal to "y",

because "y" is less negative.

(A) = -34106543

y = -00034106

RNI at P + 1

4.2 ADDRESS MODES

There are three address modes as follows:

(a) No address

e.g. Operand ENA 10B
Shift count SHAQ 5

(b) Direct Addressing

a = 0

m - operand address

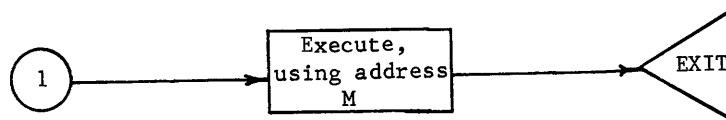
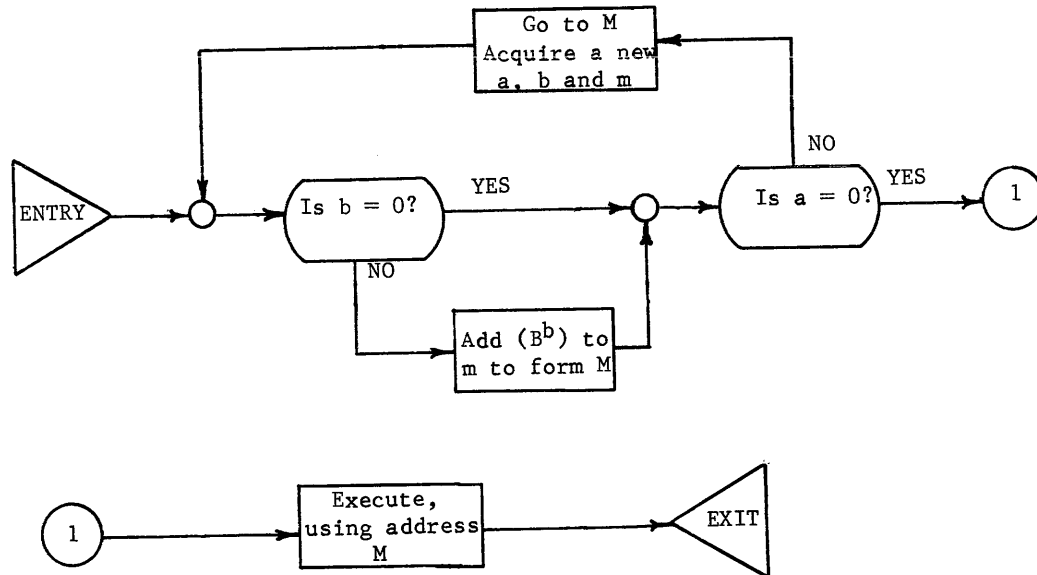
e.g. LDA 45B will load the contents of word 45B into the A register.

(c) Indirect Addressing

a = 1

e.g. LDA,I m the "I" specifies indirect addressing and generates a "1" bit in bit 17.

The best way to explain is with the following flow chart and examples.



Examples:

In the following examples use the contents of these 5 words

100	26000101	
101	37421623	(B ¹) = 2
102	00000103	
103	76543214	(B ²) = 4
104	40500100	

4.2 (cont.)

(i) LDA 100B

What will be the (A) after execution?
(A) = 26000101.

(ii) LDA,I 100B

What will be the (A) after execution?
(A) = 37421673

(iii) LDA,I 100B,1

What will be the (A) after execution?
(A) = 76543214

(iv) LDA,I 100B,2

What will be the (A) after execution?

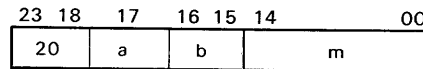
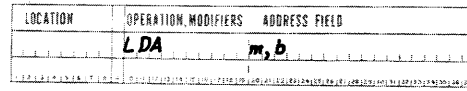
4.3 INDEX MODIFICATION OF WORD ADDRESSING INSTRUCTIONS

The 3200 has 3 Index Registers each of 15-bit capacity. These Index Registers can be used to modify the address fields of many instructions.

e.g. LDA STORE will load the contents of STORE into the A Register. But LDA STORE,1 will load the contents of (STORE + (B¹)) into the A Register, (i.e. if (B¹) = 1, it will load STORE + 1).

How is an indexed instruction assembled?

LDA STORE,2 where STORE = 00142B will assemble as: 20200142
Why?



a = addressing mode designator
b = index register designator

b = 00₂ means no index register

b = 01₂ means index register 1

b = 10₂ means index register 2

b = 11₂ means index register 3

<u>Example</u>	LDA TAG,1 where (B ¹) = 3	100	11111111
	and TAG = 100B	101	22222222
		102	33333333
		103	44444444
		104	55555555

(A) = 44444444

Restrictions on Indexing

The following instructions cannot specify indexing.

LDI	ISE	INA
STI	ASE	INQ
INI	QSE	ENA
ENI		ENQ
		AQJ
		AZJ

Exercises on Index Modification*

1. A block of memory is as shown:

TABLE	00000000
	11111111
	22222222
	33333333
TABLEX	44444444
	55555555
	66666666
	77777777

*Additional material on Index Modification is located in Chapter 8.3

4.3 (cont.)

What will be the contents of memory as a result of the following:

```
ENI    0,1
LDA    TABLE,1
STA    TABLEX,1
INI    1,1
LDA    TABLE,1
STA    TABLEX,1
INI    2,1
LDA    TABLE,1
STA    TABLEX,1
```

2. A block of memory contains the values shown

100	00000001
	00002222
	00000303
	04040404
	50055005
TAG	06606606
	77777777
	00000000
	11111111

What will be contained in the block after execution of the following instructions:

```
ENI    0,1
LDA    100B,1
INI    2,1
STA    100B,1
ENI    3,2
LDA    100B,1
STA    100B,2
ENI    5,3
LDA    TAG-6,3
STA    TAG,1
STA    TAG,2
```


48-BIT OPERATIONS

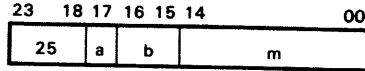
5.1 LOAD AQ

5.2 STORE AQ

5.3 SHIFT AQ

5.1 LOAD AQ

LOCATION	OPERATION/MODIFIERS	ADDRESS FIELD
	LDAQ, I	m, b



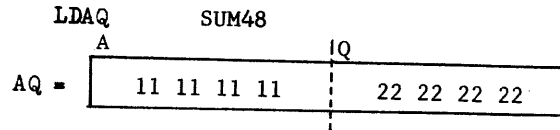
a = addressing mode designator

b = index designator

m = storage address.

Description: Loads Registers A and Q with the two words from address M and M + 1 respectively, where $M = m + (B^b)$.

Example:
 (SUM48) = 11 11 11 11
 (SUM48+1) = 22 22 22 22



Exercises: (i) A block of memory is as follows:

103	1111 1111
104	2000 2000
105	6666 6666
106	2323 2323
107	0111 1110
110	7070 7070
111	3333 4444

This has been set up by a subprogram, part of which was

```

ORGR 100B
BSS 5
FIELD BSS 3
    
```

What would be the contents of AQ after execution of:

```
LDAQ FIELD+3
```

5.1 (cont.)

Exercises: (Continued)

(ii) A block of memory is as follows:

77	11 11 11 11
100	22 22 22 22
101	33 33 33 33
102	44 44 44 44
103	55 55 55 55
104	66 66 66 66
105	77 77 77 77

A portion of a program reads:

```
LDAQ    TABLE
LDQ     TABLE+3
STA     TABLE+2
STQ     TABLE+1

ORGR    100B
TABLE  BSS    10
```

What would be the contents of the block after execution of the above program?

5.2 STORE AQ

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	STAQ, I	m, b

23	18	17	16	15	14	00
45	a	b	m			

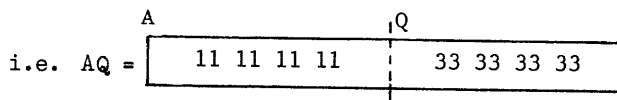
a = addressing mode designator

b = index designator

m = storage address

Description: Store the contents of Registers A and Q in storage locations M and M + 1 respectively.

Example: (i) If (A) = 1111 1111
(Q) = 3333 3333



What will be the contents of locations 100₈ and 101₈ after execution of the following instruction?

STAQ 100B

Answer : (00100) = 11111111
 (00101) = 33333333

(ii) If FIELD = location 204₈
(A) = 22334455
(Q) = 66006600

STAQ FIELD

Answer : (FIELD) becomes 22334455
 (FIELD + 1) becomes 66006600

Exercise :

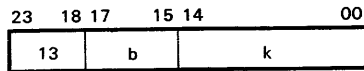
100	2 0 0 0 0 0 0 1
101	3 0 0 0 1 0 0 1
102	0 0 0 0 0 0 0 0
103	1 1 1 1 2 2 0 0
104	0 0 0 0 0 0 0 0
105	1 4 0 1 4 0 1 0
106	0 0 0 0 0 0 0 0

What will be the contents of the above after execution of the following instructions:

LDAQ 102B
STAQ 105B
LDAQ 100B
STAQ 104B

5.3 SHIFT AQ

LOCATION	OPERATION/MODIFIERS	ADDRESS FIELD
	SHAQ	k, b



b = index register designator

Description: The contents of A and Q are shifted as one 48-bit register (AQ).

Everything else is the same as for SHA and SHQ.

Examples: (i) SHAQ -12 where (A) = 12345677
(Q) = 22223333

What will be the contents of A and Q?

(A) = 00001234

(Q) = 56772222

(ii) If (A) = 00010444 and (Q) = 11335577

SHAQ 35

Answer : (A) = 66774000
(Q) = 42220455

STUDENT NOTES

48-BIT, FIXED POINT, ARITHMETIC

6.1 ADD TO AQ

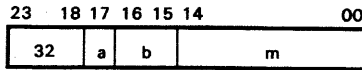
6.2 SUBTRACT FROM AQ

6.3 MULTIPLY AQ

6.4 DIVIDE AQ

6.1 ADD TO AQ

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	ADAQ, I	m, b



a = addressing mode design

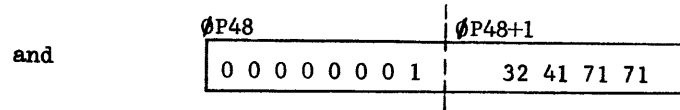
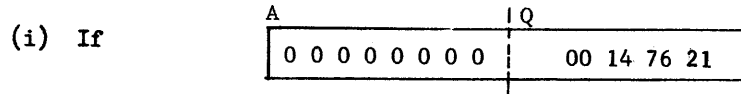
b = index designator

m = storage address

$$M = m + (B^b)$$

Description : Add the 48-bit contents of two consecutive locations M and M + 1 to the contents of AQ. The sum appears in AQ.

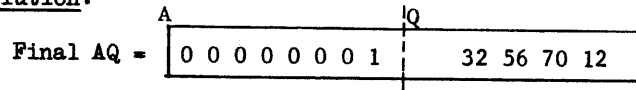
Examples:



What will be the contents of AQ after execution of the following instruction?

ADAQ ØP48

Solution:

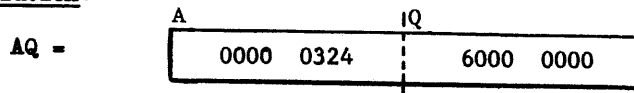


(ii) If (A) = 0000 0123	77	1004 0404
(Q) = 7000 0000	100	0000 0200
(Index register 1) = 1	101	7000 0000
	102	7777 7713

What will be the contents of A and Q after execution of the instruction:

ADAQ 77B,1

Solution:



6.1 (cont.)

Exercises:

- (i) What will be the contents of AQ after execution of the following section of a program?

ENI	0,1
ENA	24314B
STA	100B,1
INA	14741B
INI	1,1
STA	100B,1
ENA,S	34567B
ENQ,S	15677B
ADAQ	100B

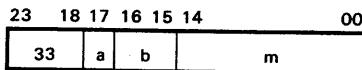
- (ii) If (TAG) = 0202 0303
(TAG+1) = 0404 0505
(A) = 2662 6626
(Q) = 1457 1406

What will be the contents of A and Q after execution of the following instruction?

ADAQ TAG

6.2 SUBTRACT FROM AQ

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	SBAQ, I	m, b



a = addressing mode indicator

b = index designator

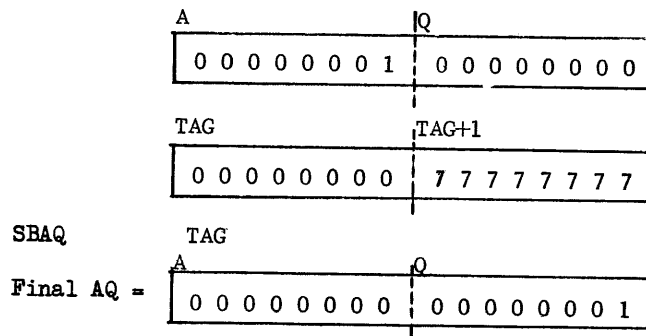
m = storage address

$M = m + (B^b)$

Description: Subtract the contents of 2 consecutive locations M and M + 1 from the contents of AQ. The difference appears in AQ.

Examples:

(i)

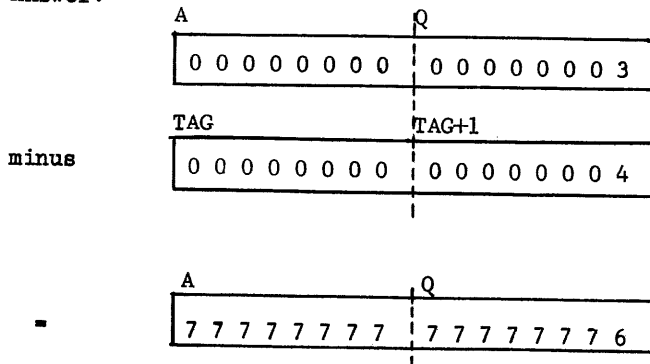


- (ii) If (A) = 0
 (Q) = 0000 0003
 (TAG) = 0
 (TAG+1) = 0000 0004

What will be in A and Q after execution of the following instruction?

SBAQ TAG

Answer:



(using end around borrow in the subtraction)

6.2 (cont.)

Exercise:

(i) (A) = 0005 7777

(Q) = 7777 0000

(TAG) = 0000 0234

(TAG+1) = 5670 0000

What will be in A and Q after execution of the following instruction?

SBAQ TAG

(ii) A block of memory is as shown:

100	21	21	21	21
101	00	00	01	35
102	21	64	16	45
103	14	14	21	21
104	00	00	00	27
105	77	77	77	77
106	56	56	57	57

What will be the contents of AQ after execution of the following instruction?

ENQ,S 0

ENA,S 2222B

SBAQ 101B

SBAQ 104B

6.3 MULTIPLY AQ

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	MUAQ, I	m, b

23	18	17	16	15	14	00
56	a	b	m			

a = addressing mode designator

b = index designator

m = storage address

Description: The instruction uses the 48-bit E register to extend the precision of AQ (by forming AQE). Multiply the contents of AQ by the 48-bit operand in two consecutive locations M and M + 1, where $M = m + (B^b)$. The 96-bit product appears in AQE, with the least significant bits in E.

Example:

A	IQ
0 0 0 0 0 0 0 0	7 6 0 0 0 0 0 0
TAG	TAG+1
0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 2

MUAQ TAG

The final results in AQE =

A	Q	E
0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 1 7 4 0 0 0 0 0 0

Exercise:

A block of memory is as shown

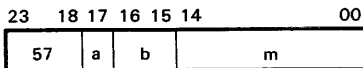
400	00 00 01 11
401	00 00 00 00
402	00 00 00 07
403	25 37 32 10
404	24 24 42 44
405	00 00 17 07

What will be the contents of AQE after execution of the following instructions?

LDA 404B
 SBA 405B
 STA 404B
 LDQ 404B
 LDA 400B
 MUAQ 401B

6.4 DIVIDE AQ

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	DVAQ, I	m, b



a = addressing mode designator

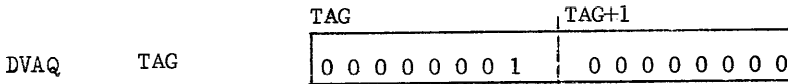
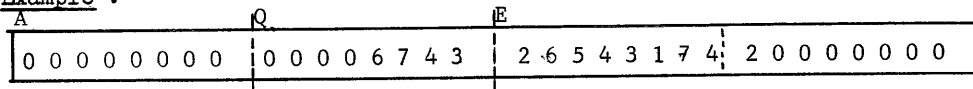
b = index designator

m = storage address

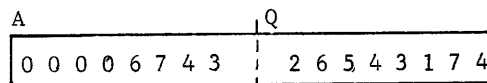
$$M = m + (B^b)$$

Description: Divide (AQE) by the 48 bit contents of two consecutive addresses, M and M + 1. The answer appears in AQ, and the remainder, with sign extended, in E.

Example :

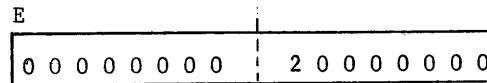


Answer is in AQ =



Remainder is in E =

(Sign extended)



Note : If a divide fault occurs, this operation halts, and the program advances to the next instruction. The final contents of AQ and E are meaningless if this occurs.

STUDENT NOTES

LOGICAL OPERATIONS

7.1 LOGIC TABLES

7.1.1 Logical "AND"

7.1.2 Inclusive "OR"

7.1.3 Exclusive "OR"

7.1.4 Examples of Logical Operations Using Octal Numbers

7.2 LOAD A LOGICAL

Chapter **7**

7.3 LOAD COMPLEMENTS

7.3.1 Load A Complement

7.3.2 Load AQ Complement

7.4 LOGICAL "AND" OPERATIONS

7.4.1 Logical Product A

7.4.2 AND of A and y

7.4.3 AND of Q and y

7.4.4 AND of Index Register B^b and y

7.5 EXCLUSIVE "OR" OPERATIONS

7.5.1 Selectively Complement A

7.5.2 Exclusive OR of A and y

7.5.3 Exclusive OR of Q and y

7.5.4 Exclusive OR of Index Register B^b and y

7.6 SELECTIVELY SET A

7.1 LOGIC TABLES

7.1.1 Logical "AND" (Logical product)

A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

Note: C is only a "1" if both A and B are "1's".

Example : AND of 1001 and 1010 (binary numbers)

1001
1010
1000

7.1.2 Inclusive "OR" (Selective Set)

A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

Note: C is a "1" if either or both A and B are "1's".

Example : Inclusive OR of 1001 and 1010

1001
1010
1011

7.1.3 Exclusive "OR" (Selective Complement)

A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

Note: C is a "1" only if A and B are different.

Example : Exclusive OR of 1001 and 1010

1001
1010
0011

7.1.4 Examples of Logical Operations Using Octal Numbers

(i) What is the logical "AND" of 43B and 62B?

Answer : 42B

(ii) What is the exclusive "OR" of 7021B and 33B?

Answer : 7012B

(iii) What is the inclusive "OR" of 361B and 403B?

Answer : 763B

(iv) What is the exclusive "OR" of 361B and 777B?

Answer : 416B

7.2 LOAD A LOGICAL

OPERATION	MODIFIERS	ADDRESS FIELD
LDL, I		m, b

23	18	17	16	15	14	00
27	a	b	m			

a = addressing mode indicator

b = index designator

m = storage address

$$M = m + (B^b)$$

Description: A is loaded with the logical product (AND) of Q and the contents of location M. In this instruction, Q serves as a mask.

Q	M	A
0	0	0
0	1	0
1	0	0
1	1	1

Note: The bit in A is a "1" only if both of the corresponding bits in Q and M are "1's".

Examples:

(i) LDL 100B

100	32470235
101	12345670
102	24601357

Q = 70770770

(A) = 30470230

(ii) If index register 3 contains 00002

LDL 100B,3

$$M = 00100 + 00002$$

$$= 00102$$

octal

binary

$$(00102B) = 24601357_8$$

$$010\ 100\ 110\ 000\ 001\ 011\ 101\ 111_2$$

$$(Q) = 70770770_8$$

$$111\ 000\ 111\ 111\ 000\ 111\ 111\ 000_2$$

$$(A) = 20600350_8$$

$$010\ 000\ 110\ 000\ 000\ 011\ 101\ 000_2$$

Answer: (A) = 20600350

7.3 LOAD COMPLEMENTS

7.3.1 Load A Complement

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	LCA, I	m, b

23	18	17	16	15	14	00
24	a	b	m			

a = addressing mode indicator

b = index designator

m = storage address

Description: The instruction load A with the complement of the 24-bit word located at M, where $M = m + (B^b)$

Examples:

(i)	LCA	100B	100	36476521
			101	67543210
			102	42454251
	(A) =	41301256	103	72130215

(ii) If index register 2 contains 00003

LCA 100B, 2 $M = 00100 + (B^2) = 00103$

(A) = 05647562

7.3.2 Load AQ Complement

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	LCAQ, I	m, b

23	18	17	16	15	14	00
26	a	b	m			

a = addressing mode indicator

b = index designator

m = storage address

Description: The instruction loads A with the complement of M, and Q with the complement of $M + 1$, where $M = m + (B^b)$

Examples:

(i) LCAQ 100B (using example above)

(A) = 41301256

(Q) = 10234567

(ii) If index register 3 contains 00002

LCAQ 100B, 3 $M = 00100 + (B^3) = 00102$

(A) = 35323526

(Q) = 05647562

7.4 LOGICAL "AND" OPERATIONS

7.4.1 Logical Product A

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	LPA, I	m

23	18	17	16	15	14	00
37	a	b				m

a = addressing mode designator

b = index register designator

m = storage address

Description: Replace (A) with the logical product of (A) and (M)
(logical AND)

Examples: (i) LPA 100B where (100) = 55555555
(A) = 52525252

What will (A) be after execution?

(A) = 50505050

(ii) LDA MASK where (MASK) = 77777777
LPA CIØX and (CIØX) = 43752016

Answer (A) = 43752016

(iii) ENA, S 20301B where (MASK) = 77777777
LPA MASK

Answer (A) = 00020301

(iv) LDA STATUS where (STATUS) = 10203040
LPA STATMASK (STATMASK) = 76543210

Answer (A) = 10003000

7.4.2 AND of A and y

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	ANA, S	y

23	18	17	16	15	14	00
17	1	b	y			

b = 0, if sign extension

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	ANA, S	y

23	18	17	15	14	00
17	4	y			

b = 2, if no sign extension

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	ANA	y

23	18	17	15	14	00
17	6	y			

Description: Enter the 24 bit logical product of (A) and y (with sign extension if specified) into A. If sign extension is not specified, zeros are extended rather than the sign of y.

A	Y	A
0	0	0
0	1	0
1	0	0
1	1	1

Note: The bit in A is a "1" only if both of the corresponding bits in A and y are "1's".

Examples: (i) ANA 23456B

Where (A) = 12345670 001 010 010 100 101 110 111 000
 00023456 000 000 000 010 011 100 101 110
 00001450₈ 000 000 000 000 001 100 101 000₂

(ii) ANA, S 23456B

Where (A) = 07654321 000 111 110 101 100 011 010 001
 00023456 000 000 000 010 011 100 101 110
 00000000₈ 000 000 000 000 000 000 000 000₂

(iii) ANA, S 43456B

Where (A) = 07654321 000 111 110 101 100 011 010 001
 77743456 111 111 111 100 011 100 101 110
 07640000₈ 000 111 110 100 000 000 000 000₂

(iv) ANA, S 70707B

Where (A) = 11111111 001 001 001 001 001 001 001 001
 77770707 111 111 111 111 000 111 000 111
 11110101₈ 001 001 001 001 000 001 000 001₂

7.4.3 AND of Q and y

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	ANQ,S	y

23	18	17	16	15	14	00
17	1	b	y			

b = 1, if sign extension

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	ANQ,S	y

23	18	17	15	14	00
17	5	y			

b = 3, if no sign extension

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	ANQ	y

23	18	17	15	14	00
17	7	y			

Description: Enter the 24 bit logical product of (Q) and y (with sign extension if specified) into Q. If sign extension is not specified, zeros are extended rather than the sign of y.

Examples: As for ANA,S y

7.4.4 AND of Index Register B^b and y

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	ANI	y, b

23	18	17	16	15	14	00
17	0	b	y			

b = index register designator (1-3)

Description: Enter the logical product of (B^b) and y into index register b.

Examples: (i) ENI 77B, 3

ANI 12B, 3

Index 3 contents = 00077

"AND" with 00012

00012

Answer : 00012 in index register 3

7.4.4 (cont.)

Examples: (Continued)

(ii) ENI 12345B,2
ANI 25252B,2

(Index 3) = 12345

"AND" with 25252

00240

Answer : 00240 in index 2

7.5 EXCLUSIVE "OR" OPERATIONS

7.5.1 Selectively Complement A

OPERATION	OPERATION MODIFIERS	ADDRESS FIELD
SCA, I	m, b	

23	18	17	16	15	14	00
36	a	b				m

a = addressing mode designator

b = index register designator

m = storage address

Description: Selectively complements corresponding bits in A for all 1-bits at address M. (Exclusive OR)

Examples: (i) SCA 100B where (100) = 70707070
(A) = 52525252

What will (A) be after execution?

Answer : (A) = 22222222

(ii) ENA, S 77777B where (MASKS) = 10203040
SCA MASKS

Answer : (A) = 67574737

(iii) ENA, S 40001B where (SIGN) = 40000001
SCA SIGN

Answer : (A) = 37740000

(iv) LDA TAG where (TAG) = 77777777
SCA SETBIT (SETBIT) = 52525252

Answer : (A) = 25252525

(v) LDA GTRC where (GTRC) = 40000001
SCA GTRC

Answer : (A) = 0000 0000

7.5.2 Exclusive OR of A and y

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	XØA, S	y

23	18	17	16	15	14	00
16	1	b	y			

b = 0, if sign extension

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	XØA, S	y

23	18	17	15	14	00
16	4	y			

b = 2, if no sign extension

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	XØA	y

23	18	17	15	14	00
16	6	y			

Description: Enter the 24 bit exclusive OR of (A) and y (with sign extension if specified) into A. If sign extension is not specified, zeros are extended rather than the sign of y.

A	y	A
0	0	0
0	1	1
1	0	1
1	1	0

Note: The bit in A is a "1" only if the corresponding bits in A and y are different.

Examples: (i) XØA 44444B if A = 33333333

$$3_8 = 011_2$$

$$4_8 = 100_2$$

$$OR = 7_8 = 111_2$$

$$\begin{array}{r} 33333333 \\ 00044444 \\ \hline 33377777 \end{array} = \text{answer}$$

(ii) XØA, S 44444B if A = 33333333

44444 is a negative number since $4_8 = 100_2$; therefore, find exclusive OR of (A) and 77744444.

$$3_8 = 011_2$$

$$7_8 = 111_2$$

$$OR = 4_8 = 100_2$$

$$\begin{array}{r} 3333 \ 3333 \\ 7774 \ 4444 \\ \hline 4447 \ 7777 \end{array} = \text{answer}$$

7.5.2 (cont.)

(iii) $X\phi A, S$ 40001B

Where $A = 20117070$

40001 is negative and equals 77740001 with sign extended, therefore

20117070
77740001
 57657071

Answer: $(A) = 57657071$

7.5.3 Exclusive OR of Q and y

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	$X\phi Q, S$	y

23	18	17	16	15	14	00
16	1	b	y			

$b = 1$, if sign extension

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	$X\phi Q, S$	y

23	18	17	15	14	00
16	5	y			

$b = 3$, if no sign extension

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	$X\phi Q$	y

23	18	17	15	14	00
16	7	y			

Description: Enter the 24 bit exclusive OR of (Q) and y (with sign extension if specified) into Q. If sign extension is not specified, zeros are extended rather than the sign of y.

Examples: As for $X\phi A, S$

7.5.4 Exclusive OR of Index Register B^b and y

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	$X\phi I$	y, b

23	18	17	16	15	14	00
16	0	b	y			

b = index designator (1-3)

Description: Enter the exclusive OR of (B^b) and Y into B^b
 No sign extension is possible on either (B^b) or y .

Examples: (i) $X\phi I$ 77B,1

where index 1 contains 32B

$$00032 = 000\ 000\ 000\ 011\ 010$$

$$00077 = 000\ 000\ 000\ 111\ 111$$

$$\underline{000\ 000\ 000\ 100\ 101}_2 = 00045_8$$

Answer : Index register 1 will contain 45B

(ii) $X\phi I$ 7777B,2

Where index register 2 contains 74321B

$$74321 = 111\ 100\ 011\ 010\ 001$$

$$77777 = \underline{111\ 111\ 111\ 111\ 111}$$

$$000\ 011\ 100\ 101\ 110_2 = 03456_8$$

Answer : Index register 2 contains 3456B

7.6 SELECTIVELY SET A

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	SSA, I	m, b

23	18	17	16	15	14	00
35	a	b	m			

a = addressing mode designator

b = index register designator

m = storage address

Description: Selectively set 1-bits in A for all 1-bits in M
N.B. This instruction leaves "1" bits which were already present in the register; i.e. it does not clear A before execution.
 This instruction is performing an "inclusive OR" operation.

Examples: (i) SSA 100B where (100) = 70707070
 (A) = 52525252

What will (A) be after execution.

(A) = 72727272

(ii) ENA, S 0 where (ABIT) = 76543210
 SSA ABIT

Answer: (A) = 76543210

(iii) ENA, S 21010B where (MØD) = 11111111
 SSA MØD

Answer: (A) = 11131111

(iv) ENA, S 44444B where (MØD) = 11111111
 SSA MØD

Answer: (A) = 77755555

CHARACTER MODE OF OPERATION

8.1 INTRODUCTION

8.2 CHARACTER ADDRESS INSTRUCTIONS

8.2.1 Load A Character

8.2.2 Load Q Character

8.2.3 Store A Character

8.2.4 Store Q Character

8.2.5 Store Character Address

8.2.6 Enter Character Address into A

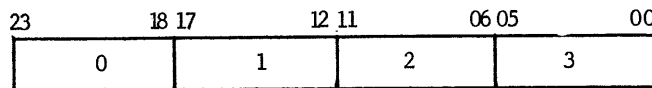
Chapter 8

8.3 INDEX MODIFICATION OF CHARACTER ADDRESSING INSTRUCTIONS

8.1 INTRODUCTION

Data may be stored in two ways in the 3200. It may be stored in full words - that is, the information stored in 24 bits. Or, it may be stored in character form.

Each word in the 3200 may be broken up into four 6-bit characters, which are called characters 0, 1, 2 and 3 in the particular word, as shown:



Information may be stored in characters, instead of in words, for convenience and economy. For instance, if we wish to store fifty digits, it could be more convenient to store them as 50 characters, to take up only $12\frac{1}{2}$ words of storage, rather than put each digit into a separate computer word.

The computer is able to address any character in any word in storage. To find the character address of any particular character, the word address of the word containing the character is multiplied by 4; and the character position is added on.

The process is simpler if the arithmetic is done in binary. To multiply by 4 in binary you suffix two zeros to the righthand end of the number. The character positions are (in binary) 00, 01, 10 and 11. Therefore to find the character address:

- (i) Change the word address to binary
- (ii) Add the character position in binary as 2 extra low order bits
- (iii) Convert back to octal

Example: Character 2 in word 104

- (i) Binary = 001 000 100 Character position
- (ii) Add on character position = 001 000 100 10
- After regrouping bits = 00 100 010 010
- (iii) Convert to octal = 422B = Character Address

Exercise: Find the character addresses of the following characters.

<u>Word Address</u>	<u>Character Position</u>	<u>Character Address</u>
100	3	
405	1	
1021	0	
477	2	
4001	1	
2664	0	
17777	3	

Similarly, if we know a character address, we can determine the word address in which the character is contained, and its position in the word.

8.1 (cont.1)

The steps are:

- (i) Convert to binary
- (ii) Remove the last two binary digits. (These are the character position)
- (iii) Convert the remainder to octal.

Example: Character address 422

- (i) Binary = 100 010 010
- (ii) Divide by 4 = 100 010 0₂ Remainder = 10₂ = 2₈ = Character Position
- (iii) Regrouped = 1 000 100₂ Octal = 104 = word Address

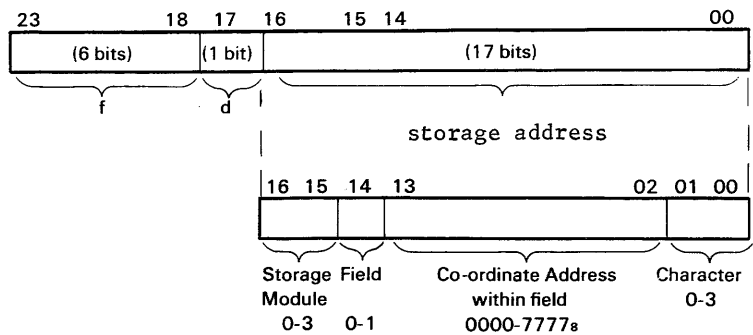
Answer - Character 2 in word 104

Exercise: (i) Find the word address and character position of the following character addresses:

- (a) 201 (e) 2110
- (b) 3 (f) 4231
- (c) 417 (g) 16524
- (d) 555 (h) 66666
- (ii) What is the smallest character address?
- (iii) What is the largest character address in an 8K computer?
- (iv) What is the largest character address in an 32K computer?

Character Address Instruction Format

The instruction format enables the address of a particular character in a word to be specified.



Character-Addressed Instruction Format

- (i) Function Code (f)
 - 6 bit code of action to be carried out.
- (ii) Operation Designator (d) - 1 bit
 - If d = 1, modify address "m" by contents of a set index for the particular instruction.
 - If d = 0, no modifications.
- (iii) Storage Address (17 bits)

The address of the character required is specified. This may be broken up into the module, field and co-ordinate address, as for word addressing, plus two bits to indicate the character position in the word.

8.1 (cont.2)

Exercise:

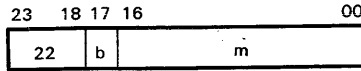
Assume that the following block of memory is in Field 1 of an 8K computer. What will be the character addresses of the characters marked x? The co-ordinate address of the first word in the block is 3124_8 .

			x
	x		
x			
		x	
	x		
		x	
x			
			x

8.2 CHARACTER ADDRESS INSTRUCTIONS

8.2.1 Load A Character

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	LACH	m, 1



m = 17 bit character address

b = index designator - but can be only one bit, so has special use. See section on indexing (section 8.3).

Description: Clears (A) and loads into bits 00-05 of A, the contents of the character specified by M. ($m + (B^1)$)

Examples:

- (i) LACH 403B (A) = 40012431
 (100) = 01020304
 (400) = 05060701
 (403) = 02030405

What will be the contents of the A Register?

(A) = 00000004 since 403B (char.)
 = 100B (word), pos. 3.

- (ii) LACH 5B

What will be the contents of the A Register after execution?

0	01234567
1	12345670
2	23456701
3	34567012
4	45670123
5	56701234

(A) = 00000034

- (iii) LACH HRTABLE+3 HRTABLE
 .
 .
 .
 HRTABLE BSS 4

01	23	45	67
12	34	56	70
11	22	33	44
55	66	77	00

What will be the (A) after execution?

(A) = 00000067

- (iv) In (iii) what would be the (A) if the instruction was LACH HRTABLE+13B
 (A) = 00000044

8.2.2 Load Q Character

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	LQCH	m, 2



Description: Same as LACH except substitute Q Register for A Register, and it uses a different index register (which also is different from SQCH).

8.2.3 Store A Character

LOCATION	OPERATION/MODIFIERS	ADDRESS FIELD
	SACH	m, 2

23	18	17	16	00
42	b	m		

m = 17 bit character address

b = index designator - but can be only one bit, so has special use. See section 8.3 on indexing.

Description: Store the contents of bits 0-5 of the A Register in the specified character address.

N.B. The contents of A and the remaining 3 characters in the storage word, are unchanged.

Examples:

(i) SACH 43B if (A) 20012345

Which character position of which word is affected, and what will be the contents of the character position?

Answer : (1) Word 10
(2) Position 3
(3) 45B

(ii) SACH L0C+3 (A) = 11560133

....

L0C BSS 5

L0C	00	11	22	33
	44	55	66	77
	01	02	03	04
	05	06	07	00
	12	34	56	07

What changes will be made to the 5 words shown?

(L0C) = 00112233

i.e. no change!

8.2.4 Store Q Character

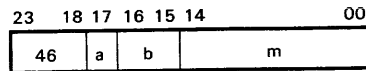
LOCATION	OPERATION/MODIFIERS	ADDRESS FIELD
	SQCH	m, 1

23	18	17	16	00
43	b	m		

Description: Same as SACH except substitute Q register for A register, and it uses a different index register (which also is different then that used by LQCH)

8.2.5 Store Character Address

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	SCHA	m, b



a = addressing mode designator

b = index register designator

m = storage address

Description: Stores the lower 17 bits of (A) in the lower 17 bits of word M. The upper bits of M and the whole of A are unchanged. The instruction is used for character address modification. Compare with SWA instruction.

Examples:

- (i) SCHA 50B (A) = 10747021
(50) = 23064513

What will be the contents of 50 after execution?

(50) = 23347021

(ii)	Location	Instruction	Assembled as
	100	SCHA *+1	46000101
	101	LACH **	22377777

If (A) = 11117643 when 100 is executed, what will be the contents of 101 after execution?

Answer: (101) = 22117643

8.2.6 Enter Character Address into A

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	ECHA,S	y

23	18	17	16	00
11	d	y		

d = 0, if no sign extension

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	ECHA	y

23	18	17	16	00
11	0+	y		

d = 1, if sign extension

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	ECHA,S	y

23	18	17	16	00
11	4+	y		

Description: Enters the 17 bit quantity "y" into the A register. When d = 1, the sign of the quantity (2^{16} bit) is extended in A. When d = 0, the upper 7 bits of the A register are set to zero. "y" is usually a character address. If "y" is a symbolic word address, the actual machine address is multiplied by 4 before it is assembled in the instruction.

Examples: (i) If \emptyset UTBUF is character position 3 in the word at word address 100, what would be the contents of A after:

ECHA \emptyset UTBUF

Answer: (A) = 00000403

(ii) What would be the contents of A after execution of:

ECHA 117640B

Answer: (A) = 00117640

(iii) If TABLE starts at word address 40000, what would be the contents of A after:

ECHA TABLE

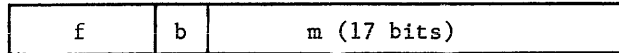
Answer: (A) = 00200000

ECHA,S TABLE

Answer: (A) = 77600000

8.3 INDEX MODIFICATION OF CHARACTER ADDRESSING INSTRUCTIONS

Character address instructions have limited indexing ability.



b is only one bit, therefore it can only indicate whether a specific index is used or not used.

i.e. If the "b" bit is set, the address is modified by either (B^1) or (B^2) depending on the particular instruction.

LACH m,1

LQCH m,2

SACH m,2

SQCH m,1

Also, if the 'm' address is at one or the other end of a table, the table is limited in length to 16383_{10} characters.

i.e. If TABLE equals word address $30000B$ (character address = $100000B$) and the instruction is:

LACH TABLE,1

(i) and (B^1) = $40000B$ 77740000
 00100000

 040001
the character address referenced =

(ii) or (B^1) = $77776B$ 77777776
 00100000

 077777
the character address referenced =

(iii) or (B^1) = $00000B$ 00000000
 00100000

 100000
the character address referenced =

(iv) or (B^1) = $00001B$ 00000001
 00100000

 100001
the character address referenced =

(v) or (B^1) = $37777B$ 00037777
 00100000

 137777
the character address referenced =

Note: If the index register contains 40000 thru 77776 the reference is backward in memory, whereas if the index register contains 00001 thru 37777 the reference is forward in memory. This means that when writing programs in COMPASS, FORTRAN, ALGOL or COBOL one should be careful not to exceed 16383_{10} when defining character arrays.

8.3 (cont.)

Examples: In the following examples assume the contents of the index registers to be:

$$(B^1) = 5$$

$$(B^2) = 2$$

$$(B^3) = -3$$

(i) LDA LABEL+1,3

STA LABEL,2

LDQ LABEL,2

What will be the contents of Q after execution of the program segment above?

LABEL

11111111
22222222
33333333
44444444
55555555
66666666
77777777

$$(Q) = 22222222$$

(ii) LACH LABEL,1

What will be the contents of A?

Answer (A) = 00 00 00 55

Exercises:

1. A block of memory (maximum length = 100 words) contains numbers stored one to a word. The end of the list is indicated by a word containing BCD blanks. Write a routine which will add the numbers in the list. (Use indexing.)
2. Transfer 57 words located 4 memory locations apart, into 57 consecutive memory locations.
3. Transfer 17 words located 5 memory locations apart, into 17 words located 10 memory locations apart.
4. Calculate the sum of the octal numbers 0-14.

INTER-REGISTER TRANSFERS

9.1 TRANSFERS BETWEEN THE A REGISTER AND INDEX REGISTERS

9.1.1 Index Register to A Register

9.1.2 A Register to Index Register

9.2 TRANSFERS BETWEEN THE A REGISTER AND THE REGISTER FILE

9.2.1 Register File to A Register

9.2.2 A Register to Register File

Chapter 9

9.3 TRANSFERS BETWEEN THE Q REGISTER AND THE REGISTER FILE

9.3.1 Register File to Q Register

9.3.2 Q Register to Register File

9.4 TRANSFERS BETWEEN INDEX REGISTERS AND THE REGISTER FILE

9.4.1 Register File to Index Register

9.4.2 Index Register to Register File

9.5 INTER-REGISTER ADDITION

9.5.1 Add Contents of Q to Contents of A

9.5.2 Add Contents of Index Register to Contents of A

9.5.3 Add Contents of A to Contents of Index Register

9.1 TRANSFER BETWEEN THE A REGISTER AND INDEX REGISTERS

9.1.1 Index Register to A Register

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	TIA	b

23	18	17	16	15	14	12	11	00
53	0	b	0					

b = index designator (1-3)
bits 0-11 are not used

Description: Transfer the 15-bit contents of index register B^b to A. (A Register is cleared before the transfer is done.)

Example:

If B² contains 54321 and A contains 7003 2146

TIA 2

(a) clear A = 0000 0000

(b) transfer 15 bits in B² to lower 15 bit positions of A

(A) = 0005 4321

9.1.2 A Register to Index Register

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	TAI	b

23	18	17	16	15	14	12	11	00
53	1	b	0					

b = index designator (1-3)
bits 0-11 are not used

Description: Clear index register B^b, and transfer the lower 15 bits of (A) to it.

Example:

If B³ contains 21B and A contains 7643 1000

TAI 3

(a) Clear Index 3

(b) Load it with lower 15 bits of A

(B³) = 31000

9.2 TRANSFERS BETWEEN THE A REGISTER AND THE REGISTER FILE

(The Register File is a special high speed memory of 100_8 locations numbered from $00_8 - 77_8$.)

9.2.1 Register File to A Register

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	TMA	m

23	18	17	16	15	14	12	11	06	05	00
53	0				2					m

m = Register File Address (00-77)

Bits 6 - 11 and 15 - 16 are not used.

Description: Transfer the contents to Register file m to A
A is cleared prior to the transfer.

Example: If A = 0000 0006, and Register file 20 = 0000 0014, what will be in A after execution of the following instruction?

TMA 20B

Answer : (a) A is cleared

(b) Contents of Reg. File 20 put in A

(A) = 00000014

9.2.2 A Register to Register File

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	TAM	m

23	18	17	16	15	14	12	11	06	05	00
53	1				2					m

m = Register File Address (00-77)

Bits 6 - 11 and 15 - 16 are not used.

Description: Register file m is cleared, and the contents of the A register are transferred to m.

Example: If (A) = 0000 1000

Register file 30 = 0000 0777

TAM 30B

Answer: (a) Register file 30 is cleared

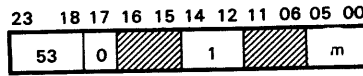
(b) Contents of A placed in Reg. file 30

(Reg File 30) = 00001000

9.3 TRANSFER BETWEEN THE Q REGISTER AND THE REGISTER FILE

9.3.1 Register File to Q Register

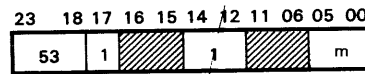
LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	TMQ	<i>m</i>



Description: As for TMA except uses Q register

9.3.2 Q Register to Register File

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	TQM	<i>m</i>



Description: As for TAM except uses Q register

9.4 TRANSFER BETWEEN INDEX REGISTERS AND THE REGISTER FILE

9.4.1 Register File to Index Register

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	TMI	<i>m, b</i>

23 18 17 16 15 14 12 11 06 05 00

53	0	b	3		m
----	---	---	---	--	---

b = index register designator
 m = register file address (00-77g)
 bits 06-11 are not used

Description: Transfer the lower 15 bits of Register m to Index B^b . Index is cleared before transfer.

Example: Index register 3 contains 00010, and Register file 25 contains 0000 2000.

What will be in B^3 after execution of the following statement?

TMI 25B,3

- (a) Index 3 is cleared
- (b) Lower 15 bits of Register file 25 are put in it.

$B^3 = 02000$

9.4.2 Index Register to Register File

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	TIM	<i>m, b</i>

23 18 17 16 15 14 12 11 06 05 00

53	1	b	3		m
----	---	---	---	--	---

b = index register designator
 m = register file address (00-77g)
 bits 06-11 are not used

Description: Clear Register file M_b , and transfer to it the contents of index register B^b .

Example: (i) Index 3 contains 22B

TIM 26B,3

$(M_{26}) = \boxed{0000\ 0022}$

9.5 INTER-REGISTER ADDITION

9.5.1 Add Contents of Q to Contents of A

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	AQA	

23	18	17	15	14	12	11	00
53	0	4	/ / / / / / / /				

Description: Transfer the contents of A plus the contents of Q to A. No information is given in the address field in this instruction.

Example: If (A) = 0000 1000
(Q) = 0000 4104

What will be A after execution of the following instruction?

AQA ---

Answer: (A) and (Q) are added together and put in A
(A) = 00005104

9.5.2 Add Contents of Index Register to Contents of A

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	AIA	b

23	18	17	16	15	14	12	11	00
53	0	b	4	/ / / / / / / /				

b = index register designator (1-3)

Description: Transfer the contents of A plus the contents of B^b to A. The contents of B^b is sign extended to perform the addition.

Example: If (A) = 0000 0100
(B²) = 01404

What will be in A after execution of the following instruction?

AIA 2

Answer: (A) and (B₂) are added and the result stored in A
(A) = 00001504

9.5.3 Add Contents of A to Contents of Index Register

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	IAI	b

23	18	17	16	15	14	12	11	00
53	1	b	4					

b = index designator (1-3)

Description: The contents of A are added to the contents of Index Register (B^b) and the result stored in B^b . The sign of the original B^b is extended prior to addition, and only the lower 15 bits of the answer are placed in B^b .

Example: (B^2) = 00006

(A) = 0000 0007

What will be the contents of index register 2 after execution of the following statement?

IAI 2

Answer: (B^2) = 00015

Exercises: (i) If (B^3) = 40001

(A) = 0000 0007

What will be the contents of B^3 after execution of the following instruction?

IAI 3

(ii) If (B^1) = 40001

(A) = 1000 0007

What will be the contents of B^1 after execution of the following instruction?

IAI 1

STUDENT NOTES

SEARCH AND MOVE OPERATIONS

10.1 BLOCK CONTROL

10.2 SEARCH OPERATIONS

10.2.1 Search for Character Equality

10.2.2 Search for Character Inequality

10.3 MOVE INSTRUCTION

10.4 PAUSE INSTRUCTION

(as used with SEARCH/MOVE Instructions)

10.1 BLOCK CONTROL

Block control is an auxiliary control section within the 3200 processor. In conjunction with the register file (see section 1.2.5) and program control (see section 1.2.4), it directs the following operations:

- (a) External equipment Input/Output
- (b) Search and Move operations
- (c) Real time clock
- (d) Console typewriter Input/Output
- (e) High speed temporary storage in the register file.

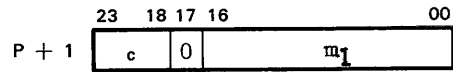
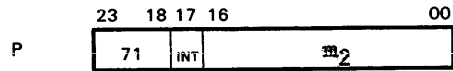
Block control is called in to initiate Search/Move operations. When the operation is initiated, it then hands control back to program control, allowing program execution to continue while the Search/Move operations are being carried out.

Note that only one operation at a time can take place under Block control. An attempt to initiate, say, a Move operation while a Search operation is in progress, will cause the Move operation request to be rejected, and Program control will skip to the reject address following the instruction.

10.2 SEARCH OPERATIONS

10.2.1 Search for Character Equality

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	SRCE, INT	c, m₁, m₂



INT = Interrupt designator

"1" = Interrupt at completion of search

"0" = Do not interrupt at completion of search

C = 00g-77g, BCD character to be looked for in memory

m₁ = First character address to be checked

m₂ = Last character address plus 1 to be checked

Description: This instruction attempts to INITIATE a search through a block of characters in storage, looking for a character equal to character "C" (specified in the instruction.)

SEARCH INITIATE POSSIBLE, i.e., the search/move section of block control is not busy. The instruction transfers the lower 18 bits of the first word of the instruction to register file address 30 and transfers all of the second word of the instruction to register file address 20. The hardware then sets flags in the upper 6 bits of register file address 30, the search is started and the computer RNI's from P+3. (During the search, register file addresses 20 and 30 are not to be disturbed, also the upper six bits of register file address 30 cannot be counted on to be any particular combination of bits).

SEARCH INITIATE NOT POSSIBLE, i.e., the search/move section of block control is busy. The instruction in this case executes as if it were a NOP (no operation instruction). It has no effect on the search/move section of block control, the register file or the interrupts if a search cannot be initiated. P+1 will be bypassed and RNI will be from P+2. This latter location is usually filled by the programmer with

UJP *-2

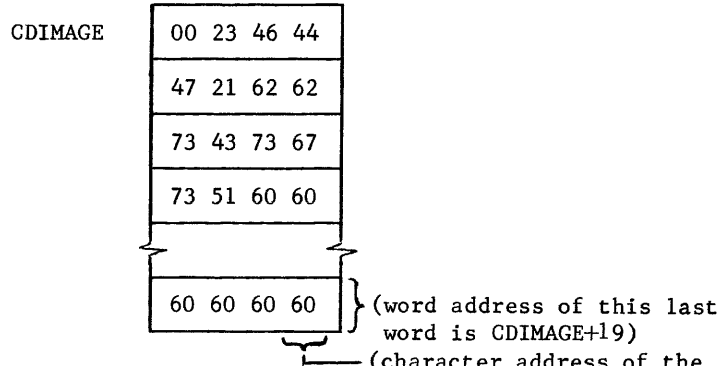
to cause the computer to loop until the previous search/move operation is complete and the search can be started.

BUFFERED SEARCH. The search progresses while other instructions following the search initiate instruction are executed. The search is made, beginning at character address m₁ and continues character by character through memory until a character equal to "C" is found or until the address to be processed is equal to m₂. The programmer can find out that the search is complete at the instant it terminates or when he is ready to check. The first is achieved with the interrupt designator, the second by using a sense instruction (both of which are to be covered later).

10.2.1 (cont.)

Assuming that the programmer knows the search has completed, he can now determine if he found a character equal to character "C". If the lower 17 bits of register file address 20 and 30 are equal, no match was found. However, if they are unequal, the lower 17 bits of register file address 20 will contain the exact character address where the match with character "C" occurred. No ambiguity is possible, since register file address 30 (lower 17 bits) contains the last character address to be searched plus one; if the lower 17 bits of register file address 20 is equal to this address, the hardware has checked all addresses to be checked and is beyond the block of characters.

Example: Given the block of data shown



and the instruction

SRCE 73B,CDIMAGE,CDIMAGE+80

what address will be in the lower 17 bits of register file address 20 at the termination of the search?

Answer: The address equal to CDIMAGE+8.
 If CDIMAGE is equal to word address 1000₈ (character address 4000₈), then (register file 20, Lower 17) = 004010₈

Special Note: When checking for a no-find condition at termination of the search, the checking is easier if the block to be searched is less than 32768₁₀ characters. If this is the case, one of the following two methods can be used:

(m₂ address fixed and lower two bits = 00₂)

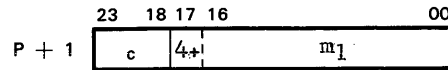
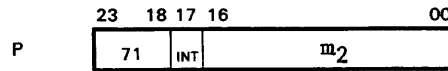
TMA 20B
 SHA -2
 ASE endaddr
 UJP find

(m₂ address variable or lower two bits ≠ 00₂)

TMA 30B
 SWA *+2
 TMA 20B
 ASE **
 UJP find

10.2.2 Search for Character Inequality

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	SACN, INT	C, m₁, m₂
12 11 10 9 8 7 6 5 4 3 2 1 0	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	



INT = Interrupt designator
 "1" = Interrupt at completion of search
 "0" = Do not interrupt at completion of search
 C = 00₈-77₈, BCD character to be checked against
 m₁ = First character address to be checked
 m₂ = Last character address plus 1 to be checked

Description: This instruction attempts to INITIATE a search through a block of characters in storage, looking for the first character not equal to the character "C" (specified in the instruction.)

SEARCH INITIATE POSSIBLE, i.e., the search/move section of block control is not busy. The instruction transfers the lower 18 bits of the first word of the instruction to register file address 30 and transfers all of the second word to the instruction to register file address 20. The hardware then sets flags in the upper 6 bits of register file address 30, the search is started and the computer RNI's at P+3. (during the search, register file addresses 20 and 30 are not to be disturbed, also the upper six bits of register file address 30 cannot be counted on to be any particular combination of bits.)

SEARCH INITIATE NOT POSSIBLE, i.e., the search/move section of block control is busy. The instruction in this case executes as if it were a NOP (no operation instruction). It has no effect on the search/move section of block control, the register file or the interrupts if a search cannot be initiated. P+1 will be skipped and RNI will be from P+2. This latter location is usually filled by the programmer with

UJP *-2

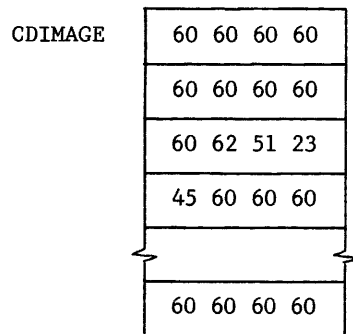
to cause the computer to loop until the previous search/move operation is complete and the search can be started.

BUFFERED SEARCH. The search progresses while other instructions following the search initiate instruction are executed. The search is made, beginning at character address m₁ and continues character by character through memory until the first character not equal to "C" is found or until the address to be processed is equal to m₂. The programmer can find out that the search is complete at the instant it terminates or when he is ready to check. The first is achieved with the interrupt designator, the second by using a sense instruction (both of which are to be covered later.)

10.2.2 (cont.)

Assuming that the programmer knows the search has completed, he can now determine if there were any characters other than the character "C" and where the first of those occurred. If the lower 17 bits of register file address 20 and 30 are equal, no characters others than "C" exist in the block. However, if they are unequal, the lower 17 bits of register file address 20 will contain the exact character address where the first character other than "C" occurred. No ambiguity is possible, since register file address 30 (lower 17 bits) contains the last character address to be searched plus one; if the lower 17 bits of register file address 20 is equal to this address, the hardware has checked all addresses to be checked and is beyond the block of characters.

Example: Given the block of data shown



(word address of this last word is CDIMAGE+19)
(character address of this last character is CDIMAGE+79)

and the instruction

SRCN 60B,CDIMAGE,CDIMAGE+80

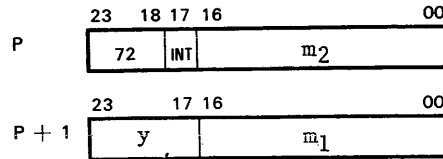
what address will be in the lower 17 bits of register file address 20 at the termination of the search?

Answer: The address equal to CDIMAGE+9.
If CDIMAGE is equal to word address 10000₈
(character address 40000₈), then the contents
of register file address 20_{lower 17 bits} = 040011₈

Special Note: See the Special Note for the SRCE instruction (Section 10.2.1) for techniques in determining type of termination.

10.3 MOVE INSTRUCTION

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	MOVE, INT	y, m ₁ , m ₂



INT = Interrupt designator

"1" = Interrupt at the completion of the Move

"0" = Do not interrupt at the completion of the Move

y = Number of characters to be Moved

If y = 001-177₈, 001-127₁₀ characters are Moved

If y = 000, 128₁₀ characters are Moved

m₁ = First character address of data source

m₂ = First character address of data destination

Description: This instruction attempts to INITIATE the copying of a block of "y" characters in one area in storage into a second area of storage. "y" is specified in the instruction.

MOVE INITIATE POSSIBLE, i.e., the search/move section of block control is not busy. The instruction transfers the lower 18 bits of the first word of the instruction to register file address 31 and transfers all of the second word of the instruction to register file address 21. The hardware then sets flags in the upper 6 bits of register file address 31, the move is started and the computer RNI's at P+3. (During the move, register file addresses 21 and 31 are not to be disturbed, also the upper six bits of register file address 31 cannot be counted on to be any particular combination of bits.)

MOVE INITIATE NOT POSSIBLE, i.e., the search/move section of block control is busy. The instruction in this case executes as if it were a NOP (no operation instruction). It has no effect on the search/move section of block control, the register file or the interrupts if a move cannot be initiated. P+1 will be skipped and RNI will be from P+2. This latter location is usually filled by the programmer with

UJP *-2

to cause the computer to loop until the previous search/move operation is complete and the move can be started.

BUFFERED MOVE. The move is made beginning at character address m₁ (source) and m₂ (destination) and continues through memory until "y" characters have been moved. If both m₁ and m₂ are character addresses that represent character position 0 and the number of characters to be moved is a multiple of four, the data is copied a word at a time. If all three conditions are not met, the copying proceeds a character at a time. The word move only takes one-fourth the time of a character move for the same number of characters, so should be used when possible. As soon as the "y" characters have been moved, the move operation terminates and the search/move section of block control reverts to the not busy status.

10.3 (cont.)

Example:

Given the block of data shown,

MOVE	6,410B,424B	101	21	14	16	32
UJP	*-2	102	60	60	60	60
		103	21	31	41	21
		104	60	60	60	60
		105	17	01	17	01
What is in storage at end of MOVE Operation?		106	25	25	25	25

Answer:

101	21	14	16	32
102	60	60	60	60
103	21	31	41	21
104	60	60	60	60
105	60	60	60	60
106	21	31	25	25

Special Note: If "y" is set by the programmer, he should remember that it is 7 bits. A SACH (or SQCH) into the upper 6 bits of "y" represents a character move of twice the number represented by (A_{lower 6}) or (Q_{lower 6}) (or is twice+1 depending on the least significant bit of the original "y").

Techniques for changing "y":

Word Move, "y" originally set to 0, number of words to be moved in A register (right justified).

```
SHA    1
SACH   Moveinst+4
```

Word Move, "y" current contents unknown, Number of words to be moved in A register (right justified).

```
SHA    2
LDQ    Moveinst+1
SHQ    7
SHAQ   17
STA    Moveinst+1
```

Character Move, Number of characters to be moved in A register(right justified).

```
LDQ    Moveinst+1
SHQ    7
SHAQ   17
STA    Moveinst+1
```

Also if the source and destination blocks overlap, be sure that the source address is the higher address in memory or some of the source data will be destroyed.

10.4 PAUSE INSTRUCTION (as used with SEARCH/MOVE Instructions)

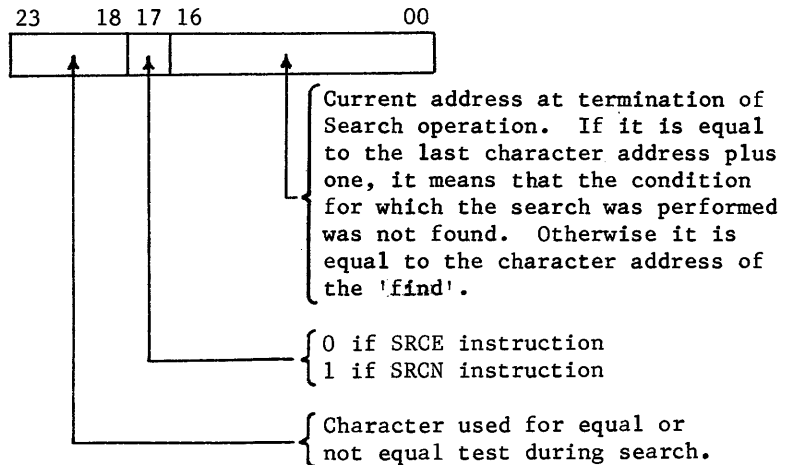
LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	PAUS	4000B



Description: Once the main program has begun a Search/Move operation, it hands over control of that part of the program to the Search/Move section of Block Control. The main program is then resumed at P + 3. If it is necessary to find the result of the Search/Move operation before the main program goes on with further calculations, etc, the PAUS instruction is used.

This instruction senses the status of the Search/Move section of Block Control. If it is busy, the machine main control pauses for 40 msec., or until the Search/Move section becomes not busy. If it stays busy for 40 msec., the computer RNI's at P + 1. If it was originally not busy or becomes not busy during the 40 msec. interval, the computer RNI's at P + 2.

At the completion of a Search/Move operation, Register File Location 20 contains the final current word address for a Search operation and Register File Location 21 contains it for a Move. The contents of that word is generally not useful to the programmer in the case of a Move, but is quite important in the case of a Search. Here is how the word at Register File Location 20 is divided:



Examples: i) The instructions to be performed are:

The block of characters to be searched is:

SRCN	21B,400B,414B	100	21 21 21 21
UJP	*-2		
PAUS	4000B	101	21 21 21 21
UJP	*-1		
TMA	20B	102	21 21 20 21
HLT			

What will be the contents of the A register when the computer halts on the HLT instruction?

Answer: (A) = 21400412 =

23	18	17	16	00
21	1			000414

10.4 (cont.1)

- ii) Given the block of data shown, where the word address of RHT = 17714_8 , what will be the contents of A after execution of:

SRCE	21B,RHT,RHT+33	RHT	00 00 00 25
UJP	*-2		
PAUS	4000B		00 00 00 31
UJP	*-1		
TMA	20B		00 00 00 46
			00 00 00 27
			03 21 60 60
			00 00 00 41
			00 00 00 31
			77 77 77 77

Answer: (A) = 21077501

- iii) If the word address of STORE is 17717_8 , and the character address of CEASE is 77514_8 , given the block of data shown, what will be the contents of A after execution of:

SRCN	60B,STORE,CEASE	STORE	60 60 60 60
UJP	*-2		
PAUS	4000B		60 60 60 60
UJP	*-1		
TMA	20B		60 60 21 60
			60 60 60 60

Answer: (A) = 60477506

$$\begin{aligned}
 \text{STORE} &= \text{word addr } 17717_8 = 001\ 111\ 111\ 001\ 111_2 \\
 &= \text{char position } 0_4 = \qquad\qquad\qquad 00_2 \\
 &= \text{char address } \qquad\qquad 001\ 111\ 111\ 001\ 111\ 00_2 \\
 &\qquad\qquad\qquad \text{regrouped} = 00\ 111\ 111\ 100\ 111\ 100_2 = 77474_8
 \end{aligned}$$

- iv) If the word address of INBUFF is 17000_8 and the word address of OUTBUFF is 17400_8 and the number of characters to be moved is 8, what will be the contents of the A register after the move is complete?

MØVE	8,INBUFF,ØUTBUFF
UJP	*-2
PAUS	4000B
UJP	*-1
TMA	21B

Answer: (A) = 00074010

At the start of the Move, the contents of Register file location 21 was:

04074000

10.4 (cont.2)

- Exercises:
- i) INBUFF currently contains a card image. Somewhere in the card image is a comma. Write the code that will search for the comma and then copy all the information up to the comma into an output area called OUTBUFF. OUTBUFF is large enough to contain all of INBUFF.
 - ii) ZUP currently contains a twelve digit BCD number with leading zeros. Using the Search Instruction, find the most significant digit and replace all leading zeros with 60B codes.
 - iii) Using the Search instruction, check the character in the A register against a table of characters, seven characters long. The table is as follows:

FTNØP

13 20 34 40
54 61 74 00

If the character is equal to any of these characters, transfer control to location WASØP, otherwise continue to the next section of code.

STUDENT NOTES

STORAGE TESTS

11.1 MASKED EQUALITY SEARCH

11.2 MASKED THRESHOLD SEARCH

11.3 STORAGE SHIFT

11.4 COMPARE (WITHIN LIMITS TEST)

11.1 MASKED EQUALITY SEARCH

LOCATION	OPERATION/MODIFIERS	ADDRESS FIELD
	MEQ	m, i

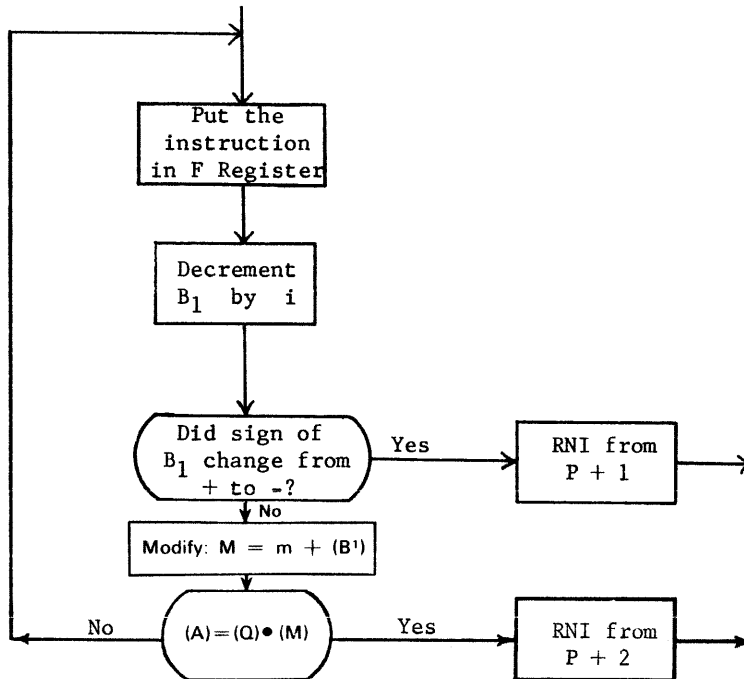
23	18	17	15	14	00
06		i			m

i = 0 thru 7 = interval designator

m = storage address (of first word)

Description: (A) is compared with the logical product of (Q) and (M) where $M = m+(B^1)$

NOTE: This instruction uses B^1 exclusively.



N.B. i = 1 means interval 1
 i = 7 " " 7
 i = 0 " " 8

Examples: (i) Given the following conditions:

$(B^1) = 5$	100	00674320
$(Q) = 77777777_8$	101	21314367
$(A) = 60606060_8$	102	60606060
	103	67676767
	104	47532167

and the following program section:

```

ENI      5,1
ENQ,S   -0
MEQ     100B,1
UJP     NØFIND
INI     100B,1
STI     FIND,1
  
```

do the following:

11.1 (cont.)

Answer:

- (a) Decrement B^1 by 1 and calculate M.

$$\begin{aligned}M &= m + (B^1) \\ &= 100 + 4 \\ &= 104\end{aligned}$$

- (b) Compare (A) with logical product of (Q) and (104)

$$\begin{aligned}(Q) &= 77777777 \\ (104) &= \underline{47532167} \\ &47532167\end{aligned}$$

which is not equal to A.

- (c) Decrement B^1 by 1 and repeat

$$\begin{aligned}(Q) &= 77777777 \\ (103) &= \underline{67676767} \\ &67676767\end{aligned}$$

which is not equal to A.

- (d) Decrement B^1 by 1 and repeat

$$\begin{aligned}(Q) &= 77777777 \\ (102) &= \underline{60606060} \\ &60606060\end{aligned}$$

which is equal to A.

RNI P + 2.

- (e) Contents of FIND = 102.

11.2 MASKED THRESHOLD SEARCH

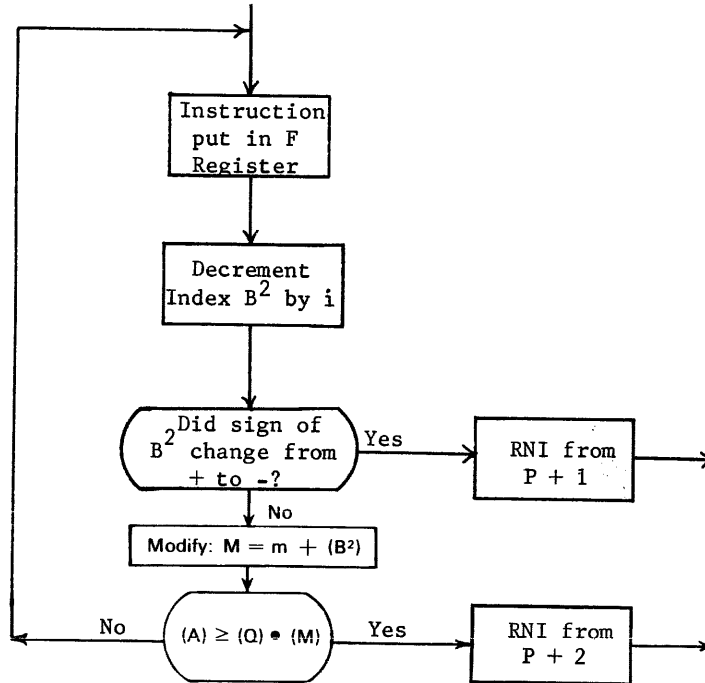
LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	MTH	m, i

23	18	17	15	14	00
07		i			m

i = 0 thru 7 = interval designator

m = storage address (of first word)

Description: This instruction is similar to the MEQ; the only differences being that RNI from P+2 occurs when $(A) \geq (Q) \times (M)$, and that index B^2 is used exclusively.



Example:

Given the following conditions and program section:

$(B^2) = 6$	100	00000067
$(Q) = 77777777_8$	101	00000104
$(A) = 00000040_8$	102	00000040
	103	00000111
	104	00000700
	105	00000077
	106	00000052

MTH	100B,2
UJP	NØFIND
INI	100B,2
STI	FIND,2

- What will be the contents of FIND after this program section is executed?
- What locations will have been checked?

Answers: a) (FIND) = 00000102₈

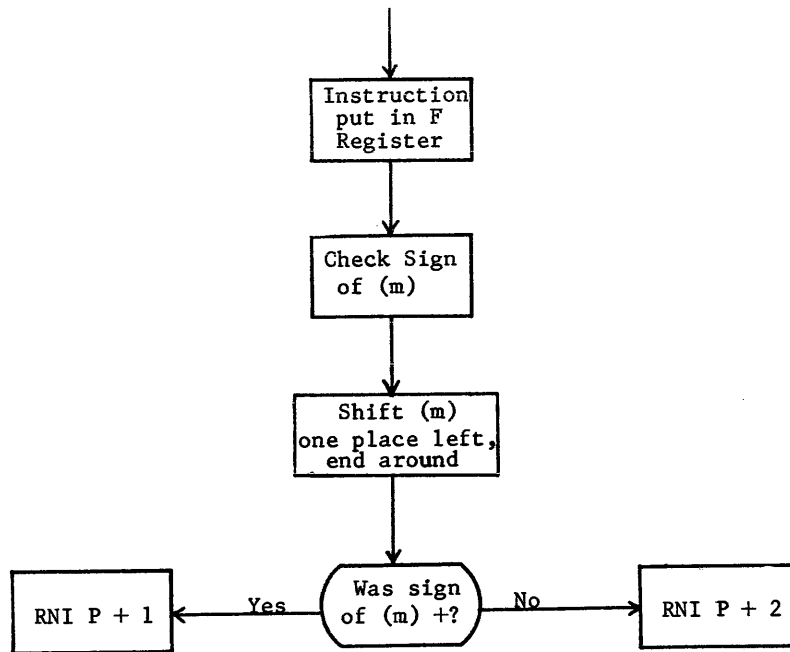
b) Locations checked = 104B
and 102B

11.3 STORAGE SHIFT

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	SSH	<i>m</i>

23	18 17	15 14	00
10	0	<i>m</i>	

Description: Sense bit 23 of (*m*).
 Shift (*m*) one place left, end around, and replace it in storage. If bit 23 of (*m*) was a 1-bit, RNI from P+2, if 0 RNI from P+1.



Examples:

(i) SSH 100B
 UJP STØP
 UJP *-2 (100) = 52525252

STØP HLT

First time through (100) = 101 010 101 010 101 010 101 010
 which is negative, so RNI at P + 2.

Second time through (100) = 010 101 010 101 010 101 010 101
 which is positive, so RNI at P + 1 and jump to Halt.

11.4 COMPARE (WITHIN LIMITS TEST)

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	CPR, I	m, b

23	18	17	16	15	14	00
52	a	b	m			

a = addressing mode designator

b = index designator

m = storage address

Description: The Compare, within limits test, instruction, is a two phase instruction. The first phase compares the contents of the M address against the contents of the A register. If the contents of memory is larger, the computer RNI's at P + 1. If the contents of memory is less than or equal to the contents of the A register, the instruction proceeds to the second phase of the test. This phase compares the contents of the Q register against the contents of the M address. If the contents of Q is larger, the computer RNI's at P + 2. If the contents is less than or equal to control passes to P + 3.

Summarized: If $(M) > (A)$, RNI at P + 1
otherwise, next test

If $(Q) > (M)$, RNI at P + 2
otherwise, RNI at P + 3

Expanded these relations become:

$(Q) > (M)$	$> (A)$	} RNI at P + 1
$(Q) = (M)$	$> (A)$	
$(M) > (Q)$	$> (A)$	
$(M) >$	$(A) = (Q)$	
$(M) >$	$(A) > (Q)$	
$(A) > (Q)$	$> (M)$	} RNI at P + 2
$(A) = (Q)$	$> (M)$	
$(Q) > (A)$	$> (M)$	
$(Q) >$	$(M) = (A)$	
$(A) = (M) = (Q)$		} RNI at P + 3
$(A) = (M) > (Q)$		
$(A) > (M) = (Q)$		
$(A) > (M) > (Q)$		

All possible relationships between A, Q and M are shown.

Normally, the value in the A register is greater than or equal to the value in the Q register. If this is the case, the following indicates the action of the instruction.

$(M) > (A)$ ————— RNI at P + 1
 $(A) \geq (M) \geq (Q)$ ——— RNI at P + 3
 $(Q) > (M)$ — RNI at P + 2

11.4 (cont.1)

While the instruction is almost always used in situations where the contents of A is greater than or equal to the contents of Q, occasionally it is not. Following is a discussion of the instruction in these alternate uses.

Stated in shorthand form, where the contents of the A register is less than the contents of the Q register, the instruction behaves as follows:

(M) > (A) RNI at P + 1
 otherwise, RNI at P + 2

Note: RNI at P + 3 is not possible because if
(M) ≤ (A) then (M) < (Q) as (A) < (Q).

Reversing positions of A and M, the branching conditions become:

(A) < (M) RNI at P + 1
(A) ≥ (M) RNI at P + 2

This then is nothing more than an ASG with a full 24 bit operand. However, one value that A can take on is troublesome and that is 37777777. When this is the contents of A, (A) < (Q) is not possible as 37777777 is the largest possible positive number.

While this form of the instruction looks useful, it rarely is used as it can usually be replaced by a load Q and an AQ test. The coding that way ends up shorter and executes faster.

Another possible use of the instruction exists where the contents of the A register is set to 37777777₈ or the contents of the Q register is set to 40000000₈.

In the case where (Q) is set to 40000000₈, the instruction acts as follows:

(M) > (A) RNI at P + 1
 otherwise, RNI at P + 3

This looks very much like the previous form except for the RNI's. It is essentially an A skip greater than or equal to where the skip is two words rather than one. Since P + 2 is unused, M can be set equal to it with (M) equal to the test value.

e.g. LDQ =04000000
 CPR *+2
 UJP less-than-routine
 ØCT test-value
 greater-than-or-equal-condition-next-instr.

In the case where (A) is set to 37777777₈, the instruction acts as follows:

(Q) > (M) RNI at P + 2
 otherwise, RNI at P + 3

This is essentially a Q test. Also since P + 1 is unused, M can be set equal to it with (M) equal to the test value.

e.g. LDA =03777777
 CPR *+1
 ØCT test-value
 UJP greater-than- routine
 less-than-or-equal-condition-next-instr.

Note in both of the two previous forms that A or Q must be set prior to the execution of the instruction and both A and Q are in use. Here as with the condition earlier, a load A or a load Q coupled with an AQ test usually produces shorter code which executes faster.

Still another possible use of the instruction exists where the contents of the A register is set equal to the contents of M or the contents of the Q register is set equal to the contents of M.

Where $(Q) = (M)$, the instruction executes as indicated:

```
(M) > (A)      RNI at P + 1
                otherwise, RNI at P + 3
```

This is similar to the case where the contents of Q is set to 40000000_8 . As in that case, P + 2 is unused. Therefore, M can be set equal to P + 2 with (M) set equal to (Q) during execution.

```
e.g.   STQ      *+3
        CPR      *+2
        UJP      less-than-routine
        BSS      1
        greater-than-or-equal-to-next-instr.
```

Where $(A) = (M)$, the instruction executes as follows:

```
(Q) > (M)      RNI at P + 2
                otherwise, RNI at P + 3
```

This is similar to the case where the contents of A is set to 37777777_8 . As in that case, P + 1 is unused. Therefore, M can be set equal to P + 1 with (M) set equal to (A) during execution.

```
e.g.   STA      *+2
        CPR      *+1
        BSS      1
        UJP      greater-than-routine
        less-than-or-equal-to-next-instr.
```

Note: In the case where (Q) is set equal to (M), the same thing could have been achieved by:

```
AQJ,GE *+2
```

Likewise, in the case where (A) is set equal to (M), the following instruction serves the same purpose:

```
AQJ,GE *+2
```

Since *+1 will almost certainly be a UJP, these can be replaced by a:

```
AQJ,LT address-in-UJP-instruction
```

Obviously, the forms of the instruction where $(Q) = (M)$ or $(A) = (M)$ are not useful.

11.4 (cont.3)

About the only useful form of the instruction other than the one where it is known that $(A) \geq (Q)$, is where the (Q) is totally unknown and of no concern. When this is the case, the instruction is applied as follows:

(M) > (A) RNI at P + 1
 otherwise, RNI at P + 2 or P + 3

When used in this manner, it is nothing more than an ASG where the test value is a 24 bit operand and (P+2) must be a NOP.

No matter how the CPR instruction is applied, the following is always true:

00000000 \neq 77777777
and, 00000000 > 77777777

Example:

Assuming the following initial conditions, what does this program do?

(A) = 100B
(Q) = 50B

(100) = 00000064
(101) = 00000104
(102) = 70000000
(103) = 00000700

```
BEGIN  CPR    100B
        UJP    *+2
        UJP    *-2
        CPR    101B
        CPR    102B
        UJP    *+2
        CPR    103B
        HLT    0
        UJP    *-5
```

Answer: RNI sequence = BEGIN
 BEGIN+3
 BEGIN+4
 BEGIN+6
 BEGIN+7 (Halt)

Exercise:

Table IQ contains 100 IQ values. Count all the IQ's in the following ranges 0-79, 80-119, 120-up. Use the CPR instruction, then do it without the CPR instruction.

STUDENT NOTES

FLOATING POINT OPERATIONS

12.1 INTRODUCTION

- 12.1.1 Storage of Floating Point Numbers
- 12.1.2 Normalizing the Coefficient
- 12.1.3 Exponent
- 12.1.4 Conversion Procedures
- 12.1.5 Unpacking Floating Point Numbers

12.2 EXECUTION OF FLOATING POINT OPERATIONS

- 12.2.1 Addition
- 12.2.2 Subtraction
- 12.2.3 Rounding of Floating Point Numbers
- 12.2.4 Multiplication
- 12.2.5 Division

12.3 FLOATING POINT INSTRUCTIONS

- 12.3.1 Floating Point ADD
- 12.3.2 Floating Point SUBTRACT
- 12.3.3 Floating Point MULTIPLY
- 12.3.4 Floating Point DIVIDE

12.1 INTRODUCTION

12.1.1 Storage of floating point numbers

Any number can be expressed in the form kB^n

when k = coefficient

B = base

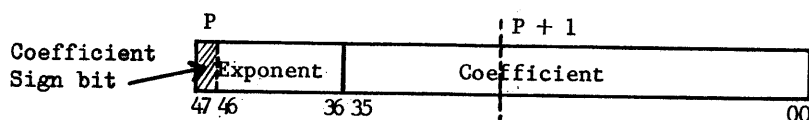
n = exponent

Floating point is always packed into 2 words in the 3200, making a 48-bit representation of the number.

The lower 36 bits = the coefficient

next 11 bits = exponent and sign of exponent

Upper bit = sign of the coefficient



12.1.2 Normalizing the coefficient

The coefficient is a 36-bit fraction, which is adjusted before packing, so that the fraction lies between $\frac{1}{2}$ and 1.

$$\text{i.e. } 1 > \text{FRACTION} \geq \frac{1}{2}$$

The coefficient is always adjusted so that a binary 1 follows the radix point, and nothing is in front of the radix point. The adjustment is made by shifting the radix point and multiplying the answer by the required power of 2.

$$\begin{aligned} \text{e.g. } 4_8 &= 100.0 \text{ in binary} \\ &= .100 \times 2^3 \text{ in binary} \\ &= .4 \times 2^3 \text{ in octal} \\ 101_8 &= 1\ 000\ 001. \text{ in binary} \\ &= .100\ 000\ 100 \times 2^7 \text{ in binary} \\ &= .404 \times 2^7 \text{ in octal} \\ .06_8 &= .000\ 110 \text{ in binary} \\ &= .110 \times 2^{-3} \text{ in binary} \\ &= .6 \times 2^{-3} \text{ in octal} \\ .00001_8 &= .000\ 000\ 000\ 000\ 001 \text{ in binary} \\ &= .100 \times 2^{-16} \text{ in binary} \\ &= .4 \times 2^{-16} \text{ in octal} \end{aligned} \quad \left. \vphantom{\begin{aligned} .00001_8 \\ .100 \times 2^{-16} \\ .4 \times 2^{-16} \end{aligned}} \right\} -16 \text{ exponent} = -16_8 = -14_{10}$$

12.1.3 The Exponent

lies in the range 0000 - 3777₈. However 1777₈ is not possible as it implies an exponent of -0 which will never result when performing a floating point operation. If one of the operands has 1777 for an exponent, it will be used as though it were 2000.

12.1.3 (cont.)

It is always biased before packing to enable comparison with other Floating point numbers.

This biasing ensures that the exponent is always positive.

RULE: If positive, 2000_8 is added to it.

If negative, 1777_8 is added to it.

Thus if the biased value lies in the range $0000_8 - 1776_8$, it is a negative exponent. 1777_8 does not normally occur as it = -0.

But if the biased value lies in the range $2000_8 - 3777_8$, it is a positive exponent.

e.g. $k \times 2^4$ Exponent is positive, therefore 2000 added.
= 2004
and $k \times 2^{407}$ = 2407

e.g. $k \times 2^{-4}$ Exponent is negative, therefore 1777_8 added
= $1777_8 + (-4)$
= 1773_8
and $k \times 2^{-407}$ = 1370_8

12.1.4 Conversion Procedures

- (1) Convert the number to binary
- (2) Normalize the number
- (3) Bias the exponent
- (4) Assemble the number
- (5) If negative, complement the result

Example: 4.0

- (1) Binary = 100.
- (2) Normalize = $.100 \times 2^3$
= $.4_8 \times 2^3$
- (3) Exponent = 3 + 2000
= 2003
- (4) Assemble =

2003	4000	0000	0000
------	------	------	------

Leftmost octal number = 2 = 010
in binary

Sign bit = 0 = POSITIVE NUMBER

e.g. Pack 563_8

- (1) Binary = 101 110 011
- (2) Normalize = $.101 110 011 \times 2^{11} = .563 \times 2^{11}$
- (3) Exponent = 11 + 2000 = 2011

12.1.4 (cont.)

$$(4) \text{ Assemble} = \begin{array}{|c|c|c|c|} \hline 2011 & 5630 & 0000 & 0000 \\ \hline \end{array}$$

e.g. Pack -563_8

Steps (1) to (4) as above

Then, because it is negative, complement the result.

$$= \begin{array}{|c|c|c|c|} \hline 5766 & 2147 & 7777 & 7777 \\ \hline \end{array}$$

Note: Bit 47 (top bit) = 1, because $5 = 101$ in binary.

e.g. Pack -1463_8

$$(1) \text{ Binary} = 001\ 100\ 110\ 011.$$

$$(2) \text{ Normalize} = .110\ 011\ 001\ 100 \times 2^{12} \leftarrow \text{octal. exponent}$$

$$= .6314 \times 2^{12}$$

$$(3) \text{ Exponent} = 12 + 2000 = 2012$$

$$(4) \text{ Assemble} = \begin{array}{|c|c|c|c|} \hline 2012 & 6314 & 0000 & 0000 \\ \hline \end{array}$$

Complement because number is negative

$$= \begin{array}{|c|c|c|c|} \hline 5765 & 1463 & 7777 & 7777 \\ \hline \end{array}$$

e.g. Pack $-.35_8$

$$(1) \text{ Binary} = .011101$$

$$(2) \text{ Normalize} = .111010 \times 2^{-1}$$

$$= .72 \times 2^{-1}$$

$$(3) \text{ Exponent} = 1777 + (-1) = 1776$$

$$(4) \text{ Assemble} = \begin{array}{|c|c|c|c|} \hline 1776 & 7200 & 0000 & 0000 \\ \hline \end{array}$$

Number is negative, so complement the number

$$= \begin{array}{|c|c|c|c|} \hline 6001 & 0577 & 7777 & 7777 \\ \hline \end{array}$$

12.1.5 Unpacking Floating Point Numbers

Rules:

- (1) If upper bit is a one, number is negative. Complement, and note that sign of the final answer must be negative.
- (2) If exponent is less than 1777_8 , exponent is negative. Subtract 1777 from exponent.
- (3) If exponent is greater than 2000_8 , exponent is positive. Subtract 2000 from it.

e.g. UNPACK

$$\begin{array}{|c|c|c|c|} \hline 2000 & 4000 & 0000 & 0000 \\ \hline \end{array}$$

$$(1) \text{ Number is positive (the upper bit} = 0)$$

$$(2) \text{ Exponent is positive } (\geq 2000)$$

$$= 2000 - 2000 = 0$$

$$(3) \text{ Number} = .4 \times 2^0$$

$$= .4$$

12.1.5 (cont.)

e.g. UNPACK

6002	0777	7777	7777
------	------	------	------

(1) Upper bit is 1 - number is therefore negative, so first complement it

=

1775	7000	0000	0000
------	------	------	------

(2) Exponent is < 1777, and is negative
= 1775 - 1777
= -2

Shift coefficient to right 2 places

(3) Coefficient = .7
= .111
Shift = .001 110
= .16
Answer = -0.16_8 or -0.21875_{10}

12.2 EXECUTION OF FLOATING POINT OPERATIONS

12.2.1 Addition

- (1) Equalize exponents by shifting coefficient of the algebraically smaller number to the right.
- (2) Add coefficients, and normalize

e.g. $10_8 + 100_8$

$$10_8 = \begin{array}{|c|c|c|} \hline 2004 & 4000 & 00000000 \\ \hline \end{array}$$

$$100_8 = \begin{array}{|c|c|c|} \hline 2007 & 4000 & 00000000 \\ \hline \end{array}$$

Shift coefficient of 10_8

$$.1_2 \times 2^4 = .0001_2 \times 2^7$$

$$= .04_8 \times 2^7$$

$$= \begin{array}{|c|c|c|} \hline 2007 & 0400 & 00000000 \\ \hline \end{array}$$

Add coefficients and normalize

$$.4000\ 0000\ 0000 \times 2^7$$

$$+.0400\ 0000\ 0000 \times 2^7$$

$$= .4400\ 0000\ 0000 \times 2^7$$

this is normalized already

$$\text{Number} = .100100000000\ 000000000000\ 000000000000_2 \times 2^7$$

$$= 1001000.$$

$$= 001\ 001\ 000_2$$

$$= 1\ 1\ 0_8$$

12.2.2 Subtraction

- (1) Equalize exponents
- (2) Subtract coefficients and normalize

e.g. $43_8 - 6_8$

$$43_8 = .100011 \times 2^6 = 2006430000000000$$

$$6_8 = .110 \times 2^3 = 2003600000000000$$

Equalize exponents, ie, 2003 6000 0000 0000 ($.110_2 \times 2^3$)

becomes

$$2006\ 0600\ 0000\ 0000\ (.000110_2 \times 2^6)$$

Subtract coefficients and normalize

$$.4300\ 0000\ 0000 \times 2^6$$

$$(-).0600\ 0000\ 0000 \times 2^6$$

$$.3500\ 0000\ 0000 \times 2^6$$

$$.7200\ 0000\ 0000 \times 2^5 \text{ after being normalized}$$

$$\text{Number} = .111010000000\ 000000000000\ 000000000000_2 \times 2^5$$

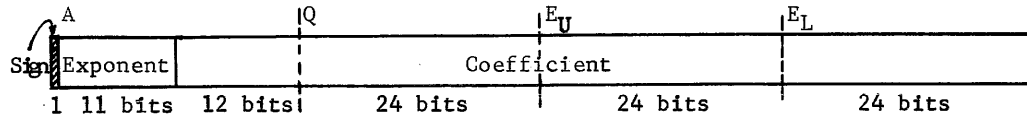
$$= 11101.$$

$$= 011\ 101_2$$

$$= 3\ 5_8$$

12.2.3 Rounding of Floating Point Numbers

In floating point operations, the E register is joined to the AQ register, to form a 96-BIT REGISTER AQE (the uppermost bit is the sign).



To carry out rounding, the sign of AQ is compared with the sign of E. If sign of AQ is not equal to sign of E, number in AQ is rounded.

Example:

(i) If $AQ_{47} = 0$, number is positive.

If $E_{47} = 1$, number in E must begin with either a

4, 5, 6 or 7

i.e. E can begin $100 = 4$

$101 = 5$

$110 = 6$

or $111 = 7$

The number in AQ is therefore rounded up by adding one to AQ.

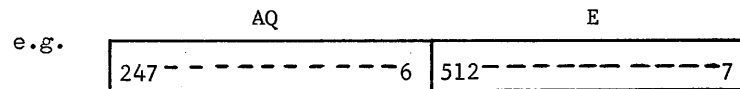
(ii) If $AQ_{47} = 1$, number is negative.

If $E_{47} = 0$, number in E must begin with either a

0, 1, 2 or 3

which is -7, -6, -5, and -4.

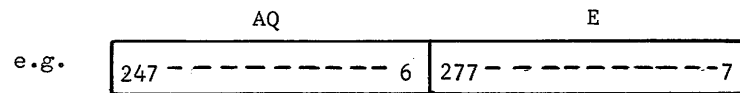
The number in AQ is therefore rounded down by subtracting one from AQ.



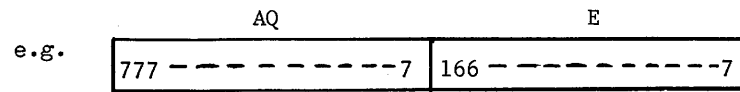
$BIT_{47} = 0$

$BIT_{47} = 1$

Positive number is rounded.



No rounding is necessary.



$BIT_{47} = 1$

$BIT_{47} = 0$

Rounded down because negative number.

Summary: If the sign of AQ is different from the uppermost bit of E, one is added to AQ if AQ is positive or one is subtracted from AQ if AQ is negative.

12.2.4 Multiplication

- (i) Unbias the exponents by subtracting 2000_8 or 1777_8 , as the case may be.
- (ii) Add the unbiased exponents.
- (iii) Multiply the coefficients.

12.2.4 (cont.)

- (iv) Normalize the coefficient resulting and adjust the exponent.
- (v) Assemble the number.

e.g. Multiply 100_8 by 10_8

$$\begin{aligned}
 100_8 &= 001000000 \\
 &= .100000 \times 2^7 \text{ (Binary)} \\
 &= .4 \times 2^7 \text{ (Octal)} \\
 &= \boxed{2007 \mid 4000 \mid 00000000}
 \end{aligned}$$

$$\begin{aligned}
 10_8 &= 001000 \\
 &= .100 \times 2^4 \text{ (Binary)} \\
 &= .4 \times 2^4 \text{ (Octal)} \\
 &= \boxed{2004 \mid 4000 \mid 00000000}
 \end{aligned}$$

- (i) Unbias exponents

2007 becomes 7

2004 becomes 4

- (ii) Add unbiased exponents

$$\begin{array}{r}
 7 \\
 +4 \\
 \hline
 13_8
 \end{array}$$

- (iii) Multiply coefficients

100_8 becomes $.4_8$ when normalized

10_8 becomes $.4_8$ when normalized

product = $.20_8$

- (iv) Normalize resulting coefficient

$$\begin{aligned}
 &.20 \times 2^{13} \\
 &= .010 \times 2^{13} \text{ (Binary)} \\
 &= .100 \times 2^{12} \\
 &= .4 \times 2^{12} \text{ (Octal)}
 \end{aligned}$$

- (v) Assemble answer

$$= \boxed{2012 \mid 4000 \mid 00 \ 00 \ 00 \ 00}$$

- (vi) Checking

$$100_8 \times 10_8 = 1000_8$$

$$= 001 \ 000 \ 000 \ 000_2$$

$$= .100 \times 2^{12} \text{ (Binary - exponent in octal)}$$

$$= .4 \times 2^{12} \text{ (Octal - exponent in octal)}$$

12.2.5 Division

- (i) Unbias the exponents
- (ii) Subtract the unbiased exponents
- (iii) Divide the coefficients
- (iv) Normalize the coefficient resulting and adjust the exponent
- (v) Assemble the numbers

e.g. Divide 100_8 by 10_8

$$\begin{aligned}
 100_8 &= 001\ 000\ 000 \\
 &= .1 \times 2^7 \quad (\text{Binary}) \\
 &= .4 \times 2^7 \quad (\text{Octal}) = \boxed{2007\ 4000\ 00000000}
 \end{aligned}$$

$$\begin{aligned}
 10_8 &= 001\ 000 \\
 &= .1 \times 2^4 \quad (\text{Binary}) \\
 &= .4 \times 2^4 \quad (\text{Octal}) = \boxed{2004\ 4000\ 00000000}
 \end{aligned}$$

- (i) Unbias the exponents

2007 becomes 7

2004 becomes 4

- (ii) Subtract unbiased exponents

$$\begin{array}{r}
 7 \\
 -4 \\
 \hline
 3
 \end{array}$$

- (iii) Divide coefficients

100_8 becomes $.4_8$ when normalized

10_8 becomes $.4_8$ when normalized

quotient = 1.0_8

ie, $.4 \overline{) 1.0}$

- (iv) Normalize resulting coefficient

$$\begin{aligned}
 &1.0 \times 2^3 \\
 &= .1 \times 2^4 \quad (\text{Binary}) \\
 &= .4 \times 2^4 \quad (\text{Octal})
 \end{aligned}$$

- (v) Assemble answer

$$= \boxed{2004\ 4000\ 0000\ 0000}$$

- (vi) Checking

$$\begin{aligned}
 100_8 \div 10_8 &= 10_8 \\
 10_8 &= 001\ 000 \quad (\text{Binary}) \\
 &= .100 \times 2^4 \\
 &= .4 \times 2^4 \quad (\text{Octal})
 \end{aligned}$$

12.3 FLOATING POINT INSTRUCTIONS

12.3.1 Floating Point ADD

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	<i>FAD, I</i>	<i>m, b</i>

23	18	17	16	15	14	00
60	a	b	m			

a = addressing mode designator

b = index designator

m = storage address

Description: Add the contents of two consecutive locations (M and M + 1) to the contents of AQ, where $M = m + (B^b)$. The normalized and rounded sum appears in AQ.

Example:

FAD	FPSUM	AQ =	A	Q
			2 0 0 7	4 0 0 0 0 0 0 0 0 0 0 0
			FPSUM	FPSUM+1
			2 0 0 4	4 0 0 0 0 0 0 0 0 0 0 0

Final result of AQ

A	Q
2 0 0 7	4 4 0 0 0 0 0 0 0 0 0 0

Exercise: If contents of Registers and a part of memory are:

(A) = 20014500

(Q) = 00000000

FPSUM	6060	6060
FPSUM+1	2007	3621
FPSUM+2	00000000	
FPSUM+3	2004	4000
FPSUM+4	0000 0000	

Index register 3 contains 3, and

Index register 1 contains 1,

What would be AQ after execution of

(a) FAD FPSUM,3

(b) FAD FPSUM,1

12.3.2 Floating Point SUBTRACT

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	<i>FSB, I</i>	<i>m, b</i>

23	18	17	16	15	14	00
61	a	b	m			

a = addressing mode designator

b = index designator

m = storage address

12.3.4 Floating Point DIVIDE

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	FDV, J	m, b

23	18	17	16	15	14	00
63	a	b	m			

a = addressing mode designator

b = index designator

m = storage address

Description: Divide the contents of AQ by the contents of 2 consecutive location, M and M + 1, where $M = m + (B^b)$. The result appears in AQ, normalized and rounded.

Example: If (AQ) are as shown below, FP0P contains 2004000, and FP0P+1 contains 0, what would be the contents of AQ after execution of the instruction.

FDV	FP0P				
Solution:	<table border="1"> <tr> <td>A</td> <td>Q</td> </tr> <tr> <td>2 0 0 7</td> <td>4 0 0 0 0 0 0 0 0 0 0 0</td> </tr> </table>	A	Q	2 0 0 7	4 0 0 0 0 0 0 0 0 0 0 0
A	Q				
2 0 0 7	4 0 0 0 0 0 0 0 0 0 0 0				
	<table border="1"> <tr> <td>FP0P</td> <td>FP0P+1</td> </tr> <tr> <td>2 0 0 4</td> <td>4 0 0 0 0 0 0 0 0 0 0 0</td> </tr> </table>	FP0P	FP0P+1	2 0 0 4	4 0 0 0 0 0 0 0 0 0 0 0
FP0P	FP0P+1				
2 0 0 4	4 0 0 0 0 0 0 0 0 0 0 0				
Final result of AQ	<table border="1"> <tr> <td>A</td> <td>Q</td> </tr> <tr> <td>2 0 0 4</td> <td>4 0 0 0 0 0 0 0 0 0 0 0</td> </tr> </table>	A	Q	2 0 0 4	4 0 0 0 0 0 0 0 0 0 0 0
A	Q				
2 0 0 4	4 0 0 0 0 0 0 0 0 0 0 0				

48-BIT REGISTER OPERATIONS

- 13.1 48-BIT E REGISTER
 - 13.1.1 Introduction
 - 13.1.2 Trapped Instructions for the E Register

- 13.2 TRANSFERS BETWEEN A AND E_U
 - 13.2.1 Transfer E_U to A
 - 13.2.2 Transfer A to E_U

- 13.3 TRANSFERS BETWEEN Q AND E_L
 - 13.3.1 Transfer E_L to Q
 - 13.3.2 Transfer Q to E_L

- 13.4 TRANSFERS BETWEEN AQ AND E
 - 13.4.1 Transfer E to AQ
 - 13.4.2 Transfer AQ to E

- 13.5 SCALE AQ

- 13.6 USE OF THE SCALE AQ INSTRUCTION

13.1 48-BIT E REGISTER

13.1.1 Introduction

The E register is a 48-bit, octal register* used as a supplement to AQ in floating point and 48-bit precision operations. It extends the size of AQ to 96 bits by forming AQE.

All 3200 computers do not contain the hardware for the E register, which therefore cannot be displayed on the console of these machines. (Where the full hardware is available, the contents of the E register can be displayed on the console in the displays usually containing A and Q.)

* The E register is the lower 48 bits of the 52 bit + sign E_D register. The upper 4 bits of E_D and the sign of E_D are not affected by Floating point, 48-bit multiply or divide, or inter-register operations.

13.1.2 Trapped Instructions for the E Register

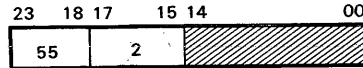
Floating point and 48-bit precision operations are handled on the 3204 basic processor by a special software package. The instructions are detected by the hardware, and "trapped". They are then processed by the special software programs OPTBOXS and FDPBOXS. OPTBOXS examines each trapped instruction to see if it is a Floating point/48-bit instruction or a BCD instruction (see Section 14.3). If it is a BCD instruction, the program BCDBOXS is used to process it. If it is a Floating point or a 48-bit precision instruction, the program FDPBOXS is used.

FDPBOXS simulates the hardware for the E register so that the instructions can be executed without the hardware being present.

13.2 TRANSFERS BETWEEN A AND E_U

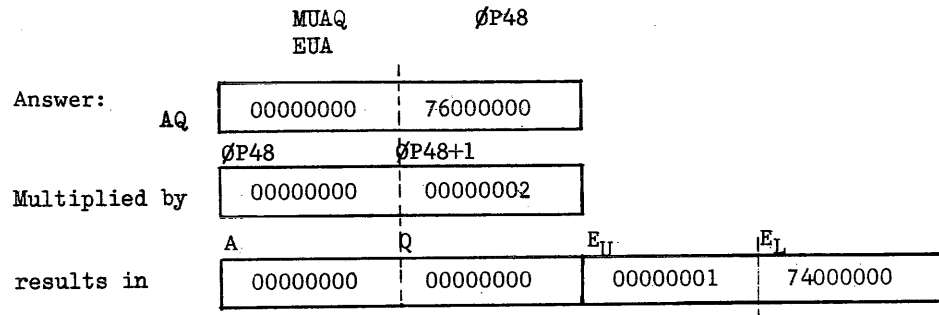
13.2.1 Transfer E_U to A

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	EUA	



Description: This instruction transfers the contents of E_{upper} (bits 47-24) to the A register. The E register is not disturbed by the transfer.

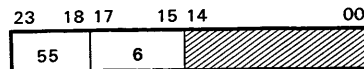
Example: Suppose $\emptyset P48$ contains 0, and $\emptyset P48+1$ contains 2. If A contains 0, and Q contains 76000000, what will be in A as a result of:



Transferring E_U to A, (A) = 00000001

13.2.2 Transfer A to E_U

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	AEU	



Description: This instruction transfers the contents of the A register to E_{upper} (bits 47-24). E register bit positions 51 thru 48 and 23 thru 00 are not disturbed by the transfer. Also the A register remains unchanged as does the sign of E.

Example: What will be in E_U after execution of the following instructions:

ENA 20321B
AEU

Answer: E_U = 00020321

13.3 TRANSFERS BETWEEN Q AND E_L

13.3.1 Transfer E_L to Q

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	ELQ	

23	18 17	15 14	00
55	1		

Description: The lower 24 bits of E, ie, E_{lower}, are transferred to the Q register. The E register remains unchanged at the end of the transfer.

Example: In the EUA example, if the instructions had been:

MUAQ ØP48
EUA
ELQ

What would be in Q after execution?

Answer: (Q) = 74000000

13.3.2 Transfer Q to E_L

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	QEL	

23	18 17	15 14	00
55	5		

Description: The contents of the Q register is transferred to E_{lower} (bits 23-00). Bit positions 51 thru 24 and the sign of E are not disturbed by the transfer. Likewise the Q register remains unchanged.

Example: In the following example:

ENQ 20321B
QEL

What will be in E_L after execution of the instructions?

Answer: (E_L) = 00020321

13.4 TRANSFERS BETWEEN AQ AND E

13.4.1 Transfer E to AQ

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	EAQ	

23	18	17	15	14	00
55		3			

Description: Transfer the 48 bit contents of E (bits 47-00) to AQ. E_U (bits 47-24) is transferred to A and E_L (bits 23-00) is transferred to Q. The contents and sign of E is not disturbed by the transfer.

Example: In the EUA example, if the instructions had been:

MUAQ $\emptyset P48$
EAQ

What would be in AQ after execution?

A Q

Answer: AQ =

00000001	74000000
----------	----------

13.4.2 Transfer AQ to E

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	AQE	

23	18	17	15	14	00
55		7			

Description: Transfer the 48 bit contents of AQ to E (bits 47-00). A is transferred to E_U (bits 47-24) and Q is transferred to E_L (bits 23-00). E_O (bits 51-48) and the sign of E are unaffected by the transfer. The contents of A and Q are not disturbed and remain after the transfer.

Example: Divide the 48 bit operand in AQ by $\emptyset P48$. Write the appropriate coding. ($\emptyset P48$) = 2.

Answer: AQE Transfer Number to E for Div.
 SHAQ -47 Set AQ to sign of E
 DVAQ $\emptyset P48$ (AQ) = Quotient, (E) = remainder

A Q

At Start
AQ =

77777777	77777765
----------	----------

A Q E_U E_L

After AQE
AQE =

77777777	77777765	77777777	77777765
----------	----------	----------	----------

A Q

After SHAQ
AQ =

77777777	77777777
----------	----------

A Q E_U E_L

After DVAQ
AQE =

77777777	77777772	00000000	00000000
----------	----------	----------	----------

13.5 SCALE AQ

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	SCAQ	k, b

23	18	17	16	15	14	00
13	1	b	k			

b = index designator
 k = shift designator

Description: AQ is shifted left, end around, until the upper two bits (46 and 47 are unequal.)

During the operation, the computer makes a shift count. A quantity $K = k$ minus the shift count.

If $b = 0$, this residue is discarded

If $b = 1-3$, the residue is placed in index register B^b .

Example:

	A	Q
AQ initial contents	0 3 0 0 0 0 0 0	0 0 0 0 0 0 0 0

SCAQ 24,2

Top bits of A = 000 011 000 etc.

To get top 2 bits unequal, shift left 3 places
 = 011 000 etc.

AQ becomes	3 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
------------	-----------------	-----------------

$$k = 24_{10}$$

$$= 30_8$$

$$K = k - \text{shift count}$$

$$= 30_8 - 3$$

$$= 25_8, \text{ which is placed in Index Reg. 2.}$$

Exercise:

If A contains 100B, and Q contains zero, what would be in A, Q and Index register 1 after execution of the following instruction?

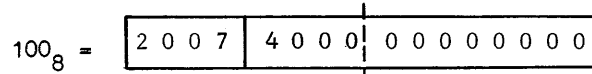
SCAQ

2027B,1

13.6 USE OF THE SCALE AQ INSTRUCTION

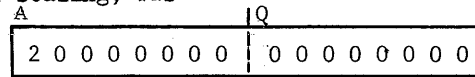
It is used to pack floating point numbers.

Example:



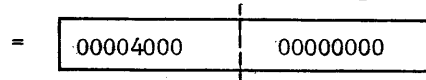
In previous example (using 100_8)

AQ, after scaling, was

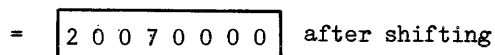
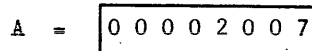


and Index Register 1 = 2007

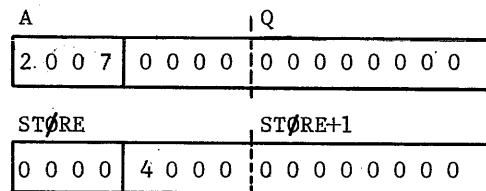
(a) Then shifting AQ to right 11 places



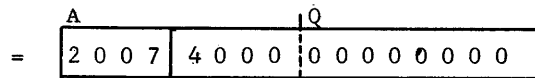
(b) by storing AQ elsewhere,
Reading Index Register 1 into A,
and shifting it left 12 places,
we have the exponent at the top of the AQ.



(c) By then adding back to AQ the stored value previously
in AQ, (the normalized exponent) we have the packed
floating point number



Final AQ



(d) This will only work with positive numbers.

(e) For negative numbers, complement the number first.
Then pack the number, as above and complement the
packed number.

13.6 (cont.)

Alternate method for packing floating point numbers:

If the computer has floating-point hardware and if the number to be packed is a single precision integer (24-bit operand), then the following method is used instead of the one just illustrated:

	SHAQ	-24	FORM 48-BIT SIGNED OPERAND
	SCA	K2044	MERGE WITH PROPER EXPONENT
	FAD	K2044	NORMALIZE THE F.P. NUMBER
	.		
	.		
	.		
	.		
	UJP		
*			
K2044	ØCT	20440000,0	F.P. CØNSTANT = 2044000000000000

Here the FAD instruction is used to do the normalizing instead of the SCAQ. Also the normalizing is done after forming the floating point number rather than before. If the operand to be packed is larger than 24-bits or if the computer doesn't have floating point hardware, the number can be packed faster using the SCAQ instruction.

BCD DIGIT OPERATIONS

14.1 INTRODUCTION

- 14.1.1 BCD Digits
- 14.1.2 Field
- 14.1.3 Sign Bits
- 14.1.4 E_D Register (In Machine)
- 14.1.5 E_D Register (On Console)
- 14.1.6 BCD Fault

14.2 BCD INSTRUCTIONS

- 14.2.1 Shift E_D Register
- 14.2.2 E_D Equal to ZERO Jump
- 14.2.3 E_D Less Than ZERO Jump
- 14.2.4 E_D Overflow Jump
- 14.2.5 Setting Field Length in D Register
- 14.2.6 Load E_D
- 14.2.7 Store E_D
- 14.2.8 Add to E_D
- 14.2.9 Subtract From E_D

14.3 BCD TRAPPED INSTRUCTIONS

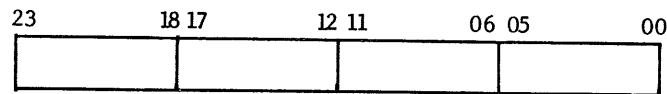
14.1 INTRODUCTION

14.1.1 BCD Digits

The BCD instructions handle 4-bit BCD DIGITS. These digits are the decimal digits 0-9 represented as follows:

0000 = 0
0001 = 1
0010 = 2
0011 = 3
.
.
.
.
.
1001 = 9

Each 24-bit word of storage is divided into 4 BCD Digit characters of 6 bits, as shown:



The lower 4 bits in each character are the BCD Digit. The upper 2 bits in the least significant character is used to represent the sign of the field.

14.1.2 Field

A field is a group of BCD Digits, of a maximum length of 12 digits. The length of the field is stored in the D register, which consists of a 4 bit register within the hardware. It is not displayed on the Console.

14.1.3 Sign Bits

This represents the sign of the field as a whole - not of the individual BCD digit. It is stored in the least significant digit in the field. The signs bits of the other digits in the field must be 00, or a fault is generated.

If the sign bit stored is 10,
the field is negative.

For all other combinations, it is positive

i.e. $\left(\begin{array}{l} 00 \text{ xxxx} \\ 01 \text{ xxxx} \\ 11 \text{ xxxx} \end{array} \right)$ positive field

Example:

00 0001	00 0100	00 0011	00 1000	= 1438
00 0110	00 0010	00 1001	00 0000	= 6290

If the Field is 7 digits long, the sign is stored with the least significant digit (i.e. the rightmost of the field, = 9). It is 00. so the number in memory is positive = 1,438,629.

14.1.4 The E_D Register (in machine) - where hardware is available.

This is a decimal register, consisting of 12 BCD decimal digits and 1 overflow BCD decimal digit.

14.1.5 The E_D Register (on console) - where hardware is available.

Displayed in the AQ register as decimal numbers. Displays the full E_D register plus 3 additional characters, as shown.

Sign of
digit being
currently
accessed
+ or -

+	1	+	1	12 DEC DIGITS
---	---	---	---	---------------

Digits Sign of Overflow
being E_D digit
currently + or -
accessed

Max number in E_D = 999,999,999,999

If one more is added, the digit in the overflow position will become a one.

14.1.6 BCD Fault

Programmer can arrange for an interrupt to occur if a BCD fault is discovered. He can also arrange to keep sensing for a fault without an interrupt.

3 conditions will produce a fault.

(a) If the upper 2 bits of any digit (except those of the least significant digit in the field) are not 00.

(b) If an illegal digit is present
i.e. any four bit combination greater than 9.

e.g. 1011 = 11 is illegal

(c) If the contents of the D register are greater than 12 (14₈)

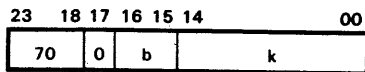
BCD fault is sensed by the SENSE Internal Status Instruction:

INS 4000B
If BCD fault, RNI P + 1
If no fault, RNI P + 2

14.2 BCD INSTRUCTIONS

14.2.1 Shift E_D Register

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	SFE	k, b



k = shift count
 b = index designator

Description: The E_D register is shifted in one character steps (i.e. 4 bits at a time)

$K = k + (B^b)$ with sign extension

(The instruction senses bits 0-3 and 23 only of the sum of k and (B^b)).

If bit 23 = 0, Shift is left, end off, zero fill.

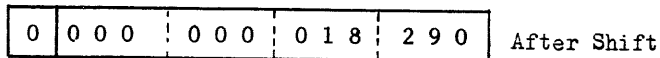
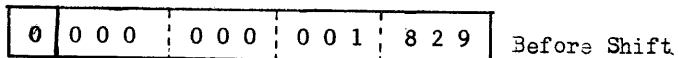
If bit 23 = 1, Shift is right, end off, zero fill.)

N.B. : BOTH SHIFTS ARE END OFF.

Example:

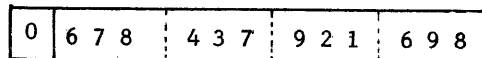
SFE 1

Shifts E_D 1 character to left (i.e. shifts one digit to the left.)



Exercises:

(i) If the E_D register contains



What will it contain after the following instructions

ENI 1,1

SFE 3,1

(ii) If the above result is followed by

SFE -8,1

What will be the final contents of E_D?

14.2.2 E_D equal to ZERO Jump

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	EZJ, EQ	m

23	18 17	15 14	00
70	4	m	

m = storage address

Description: The contents of the 52-bit E_D register are compared with zero

If (E_D) = 0, RNI address m

If (E_D) \neq 0, RNI P + 1

Example:

EZJ, EQ	ENDL OP P
SFE	1,2
UJP	*+5
ENDL OP P	LDA 144B

etc.

14.2.3 E_D less than ZERO Jump

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	EZJ, LT	m

23	18 17	15 14	00
70	5	m	

m = storage address

Description: The contents of the 52-bit E_D register are compared with zero

If (E_D) < 0 RNI address m

If (E_D) \geq 0 RNI P + 1

14.2.4 E_D Overflow Jump

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	EOJ	m

23	18 17	15 14	00
70	6	m	

m = storage address

Description: If the upper 4 bits of the E_D register contain anything but zero, control jumps to address m.

If Upper 4 bits contain zero, RNI P + 1

14.2.4 (cont.)

Examples:

0	4 2 1	3 5 7	6 1 9	8 7 8	- no overflow, jump RNI P + 1
---	-------	-------	-------	-------	-------------------------------

4	0 0 0	0 0 0	0 0 0	2 4 1	- O/flow, jump RNI Address m.
---	-------	-------	-------	-------	-------------------------------

14.2.5 Setting Field Length in D Register

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	SET	y

23	18 17	15 14	04 03 00
70	7		y

y = field length indicator

Description: The instruction takes the lower 4 bits of y and puts them in the D register. Maximum length of field is 14B for all operations except STE, when maximum length of the field is 15B. The D register remains at the value set until it is set again.

It is not cleared in Master Clear operations.

Example: SET 14B

Sets D register to 14B for field length

14.2.6 Load E_D

ONLY INDEX REGISTER 1
CAN BE USED

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	LDE	m, I

23	18 17 16	00
64	b	m

b = index designator

If b = 0, m is the unmodified address

If b = 1, m is modified by (B¹) sign extended

m = storage address (character address)

Description: The instruction loads the E_D register with a field of up to 12 numeric BCD characters. The field length is specified by the D register. Characters are put in the lower end of E_D, with zero fill to the left.

Example: SET 7

LDE 400B

Find least significant digit

100
101

3	6	8	9
4	5	6	8

14.2.6 (cont.)

$$= M + (D - 1)$$

$$= M + (6) = 406$$

Loads this character into E_D first, into leftmost bits of E_D . The R shifts one character, adds in next digits, shifts, etc. When loaded D characters, zero fills rest of E_D from left.

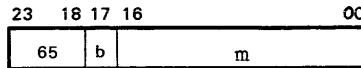
Answer : E =

0	000003689456
---	--------------

14.2.7 Store E_D

LOCATION	OPERATION/MODIFIERS	ADDRESS FIELD
	STE	m,2

ONLY INDEX REGISTER 2
CAN BE USED



b = index designator
 If b = 0, m is the unmodified address
 If b = 1, m is modified by (B^2) only
 m = storage address (character address)

Description: The instructions stores a field of up to 13 decimal digits (3.C.D. numeric characters), beginning at address M (the address of the most significant digit in the field). The length of the field is determined by the D register. The least significant digit is stored first, and the register is then Shifted right one digit. The next digit is stored, and the register shifted right again, and so on. NOTE THAT STORING DESTROYS THE CONTENTS OF E_D .

Example:

SET 14B

STE 400B

E =

0	746871264789
---	--------------

Field = 12 characters

Address of least significant character = $(D - 1) + M$
 = $(13) + 400$
 = 413

Character address = $= 100\ 001\ 011$
 = Word 102
 character 3

12 digits in E stored as follows:

100	7	4	6	8
101	7	1	2	6
102	4	7	8	9

14.2.7 (cont.)

NOTE:

DUMP OF MEMORY:

If an area of memory containing BCD numeric characters is dumped out, the 6 bits for each character (2 sign bits, and 4 bits for the digit) will be dumped in octal numbers:

e.g. 00 1000 = 8 in BCD digit

But would be 001 000 = 10 in octal during dump.

Example: A block of memory is as follows (Characters are BCD Digits)

100	4	7	8	9
101	6	4	3	1
102	8	7	4	9
103	6	2	1	3

A program working on this is as follows:

```

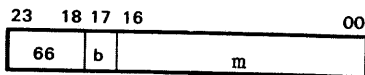
SET 12B
LDE 402B (a)
SET 6
STE 405B (b)
SET 14B
LDE 403B (c)
    
```

What would be contained in E_D at the end of the operations?

14.2.8 Add to E_D

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	ADE	m, 3

MODIFIED BY INDEX
REGISTER 3 ONLY



b = index designator

If b = 0, m is the unmodified address, i.e. $M = m$

If b = 1, m is modified by (B^3) only, $M = m + (B^3)$

m = storage address (character address)

Description: This instruction adds 'D' numeric BCD digits to the E_D register. The 'D' digits are lined up with the lower 'D' digits of the E_D register before they are added. The E_D register has a maximum capacity of 13 digits, i.e., 12 digits plus overflow digit. The maximum number of digits in the number to be added is 12, i.e., D may not be greater than 12. M is the most significant digit of the number to be added. $M + D - 1$ is the least significant digit.

14.2.8 (cont.)

Example: If $(E_D) =$

0	0 0 0	0 0 0	8 7 6	5 4 3
---	-------	-------	-------	-------

And a block of memory is

100	7	2	4	9
101	1	5	8	2
102	6	1	3	6
103	2	0	8	4

What will be the contents of E_D after the following instructions are executed?

ENI 1,3
 SET 10B
 ADE 400B,3

Answer: (a) $M = m + (B^3)$
 $= 400 + 1 = 401$

Add to E_D 8 character the least significant of which will be

$M + (D - 1)$
 $= 401 + 7$
 $= 410$

$E_D =$

0	000	000	876	543
---	-----	-----	-----	-----

$+$

0	000	024	915	826
---	-----	-----	-----	-----

Final $E_D =$

0	000	025	792	369
---	-----	-----	-----	-----

14.2.9 Subtract from E_D

OPERATION	OPERATION MODIFIERS	ADDRESS FIELD
SBE	$m, 3$	

23	18	17	16	00
67	b	m		

$b =$ index designator
 If $b = 0$, m is the unmodified address, i.e., $M = m$
 If $b = 1$, m is modified by (B^3) only, $M = m + (B^3)$
 $m =$ storage address (character address)

Description: As for ADE, except that the field of up to 12 BCD digits is subtracted from the E_D register.

14.3 BCD TRAPPED INSTRUCTIONS

All BCD instructions may be used in any 3200 Computer, regardless of the model of the processor. Where the processor lacks the BCD hardware package necessary for direct processing of BCD instructions, the implementation of these instructions is carried out by a special software package. The instructions are then known as "trapped" instructions.

(It should be noted that where the software package is used, the contents of the Ed register cannot be displayed on the console, because the Ed register hardware does not exist.)

The B.C.D. instructions are detected by a translator as they appear in the Function register, and trapped. They are processed like interrupts, and the following action takes place

- (a) $P + 1$ is stored in the lower 15 bits of address 00010
- (b) The upper 6 bits of the Function register are stored in the lower 6 bits of 00011 - the upper 18 bits of 00011 remain unchanged.
- (c) Program control is transferred to 00011, and a RNI cycle is executed.

COMPASS PSEUDO INSTRUCTIONS

- 15.1 CONCEPTS OF PSEUDO INSTRUCTIONS
- 15.2 PROGRAM DEFINITION
 - 15.2.1 IDENT Instruction
 - 15.2.2 END Instruction
 - 15.2.3 FINIS Instruction
- 15.3 ASSEMBLY AREAS
 - 15.3.1 Introduction
 - 15.3.2 DATA Area
 - 15.3.3 Return Assembly Control to Subprogram PRG Area
 - 15.3.4 COMMON Area
 - 15.3.5 ORGR Instruction
- 15.4 STORAGE RESERVATIONS
 - 15.4.1 Word Block
 - 15.4.2 Character Block
- 15.5 ENTRY AND EXTERNAL INSTRUCTIONS
 - 15.5.1 ENTRY Pseudo Instruction
 - 15.5.2 EXTERNAL Pseudo Instruction
 - 15.5.3 SCOPE Loading of Subprogram
- 15.6 SYMBOL DEFINITION BY EQUIVALENCING
 - 15.6.1 Introduction
 - 15.6.2 Word Equating
 - 15.6.3 Character Equating
- 15.7 COMPASS OUTPUT LISTING CONTROL
 - 15.7.1 REMarks
 - 15.7.2 NO LIST Instruction
 - 15.7.3 Resume LISTing Instruction
 - 15.7.4 SPACE Instruction
 - 15.7.5 New Page EJECT Instruction
 - 15.7.6 TITLE Instruction
 - 15.7.7 Comments

15.1 CONCEPTS OF PSEUDO INSTRUCTIONS

A better name would be "Assembly-Control" instructions. They are simply instructions from the programmer to the assembler.

They will be used during assembly only. Program execution can make no use of them.

e.g. BSS 4 is an instruction to the assembler to set aside 4 words of storage somewhere in the storage area. The 4 words are set up during assembly and then the function and usefulness of the BSS instruction is finished.

15.2 PROGRAM DEFINITION

15.2.1 IDENT Instruction

LOCATION	OPERATION/MODIFIERS	ADDRESS FIELD
	IDENT	m

Description: The Location field is blank. However if a symbol is written in it, COMPASS will ignore it. COMPASS picks up 8 or less alphanumeric characters from the address field, the first of which must be alphabetic. The Address field terminates at the first blank or the eighth alpha-numeric character, whichever is the first encountered. A period may appear in m.

m = PROGRAM NAME

This will appear on the top of each page of the assembly listing.

The IDENT card must be the first card in the program or the job will be terminated.

Examples: (a) IDENT TEST FOR ILLEGAL SYMBOLS
.
.
END

Program name will be TEST

(b) IDENT CONTINUOUS TESTING OF TAPES
.
.
.
END

Program name will be CONTINUOUS

15.2.2 END Instruction

LOCATION	OPERATION/MODIFIERS	ADDRESS FIELD
	END	m

Description: The location field should be blank. If a symbol is present, it is ignored by COMPASS. The END instruction terminates the sub program. The final instruction in a COMPASS Sub-program must be an END instruction.

Symbolic Transfer Address

The "m" signifies a symbol in the address field of some subroutine which has declared it to be an entry point. This address is called the Symbolic Transfer Address and need not be in the subroutine terminated by this END card.

Examples: (i) A program of one subprogram.

The symbolic transfer address must appear, and the symbol must be defined within the subprogram as an entry point.

e.g.

	IDENT	TEST
	ENTRY	FIRST
FIRST	UJP	**
	.	
	.	
	END	FIRST

15.2.2 (cont.1)

(ii) A program of more than one Compass subprogram

The symbolic transfer address must appear in one of the END statements, and be defined as an entry point as before.

```
Example:      IDENT    TEST
              ENTRY    FIRST
FIRST        UJP      **
              .
              .
              .
              .
              END

              IDENT    WRITER
              .
              .
              .
              .
              .
              END      FIRST
```

Note that this program could have appeared as follows:

```
              IDENT    TEST
              ENTRY    FIRST
FIRST        UJP      **
              .
              .
              .
              .
              END      FIRST

              IDENT    WRITER
              .
              .
              .
              .
              .
              END
```

(iii) A program of COMPASS and FORTRAN subprograms.

A symbolic transfer address should not appear in any END statement in the program, if the FORTRAN program is the main program.

15.2.2 (cont.2)

Example: PROGRAM TEST
 CALL ONE
 CALL TWO
 END

 IDENT ONE
 .
 .
 .
 END

 IDENT TWO
 .
 .
 .
 END

If the FORTRAN is a subroutine of a COMPASS subprogram a transfer address should appear in the END statement in the COMPASS subprogram.

TRA Card diagnostic

Where an error occurs in using symbolic transfer addresses, the error is flagged by the Loader when the program is loaded, and execution is not attempted.

The flag "TR" is shown (after the Load card is listed) on the standard output unit.

15.2.3 FINIS Instruction

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	FINIS	

Description: Symbols in the location and address fields are ignored. The instruction tells the assembler that it has reached the end of the assembly, and that all sub-programs have been assembled. If the FINIS card is put in out of order, the assembly will be terminated when it is reached. Control is returned to SCOPE when the FINIS card is read.

Example: IDENT TYPØUT
 .
 .
 .
 END

 IDENT SØUT
 .
 .
 .
 END

 IDENT TYPIN
 .
 .
 END
 FINIS

15.2.3 (cont.)

Where FORTRAN and COMPASS subprograms are used in the one subprogram, the FINIS card is used to indicate the end of each group of subprograms.

Example:

```
7FORTRAN,L,X
9
PROGRAM ONE
.
.
.
END

SUBROUTINE TWO
.
.
.
END
FINIS
```

```
7COMPASS,L,X
9
IDENT THREE
.
.
.
END
FINIS
```


15.3 ASSEMBLY AREAS

15.3.1 Introduction

There are 3 areas in any sub program assembly

(a) Subprogram Area.

In this area all normal parts of the subprogram are assembled.

(b) COMMON Area

Parts of the program declared to be in common are assembled in this area.

(c) DATA Area.

Parts declared to contain Data are assembled in this area.

Three counters are used at assembly time to put parts of program in sequential places in the areas above. These counters are incremented to give the current address of the instruction being assembled in the particular area in which the assembly is taking place.

15.3.2 DATA Area

LOCATION	OPERATION/MODIFIERS	ADDRESS FIELD
	DATA	

Description: Information may be put into the Data area at assembly time.

There may be no reference to an external symbol, nor can any symbol in the DATA area be an entry point for the subprogram in which it occurs.

The instruction specifies that all information following is to be stored or identified as part of the DATA area, until PRG or COMMON or END occurs. Any instruction or pseudo instruction may follow DATA. The DATA area is shared by all subprograms at execution time.

The Location and Address fields should be blank.

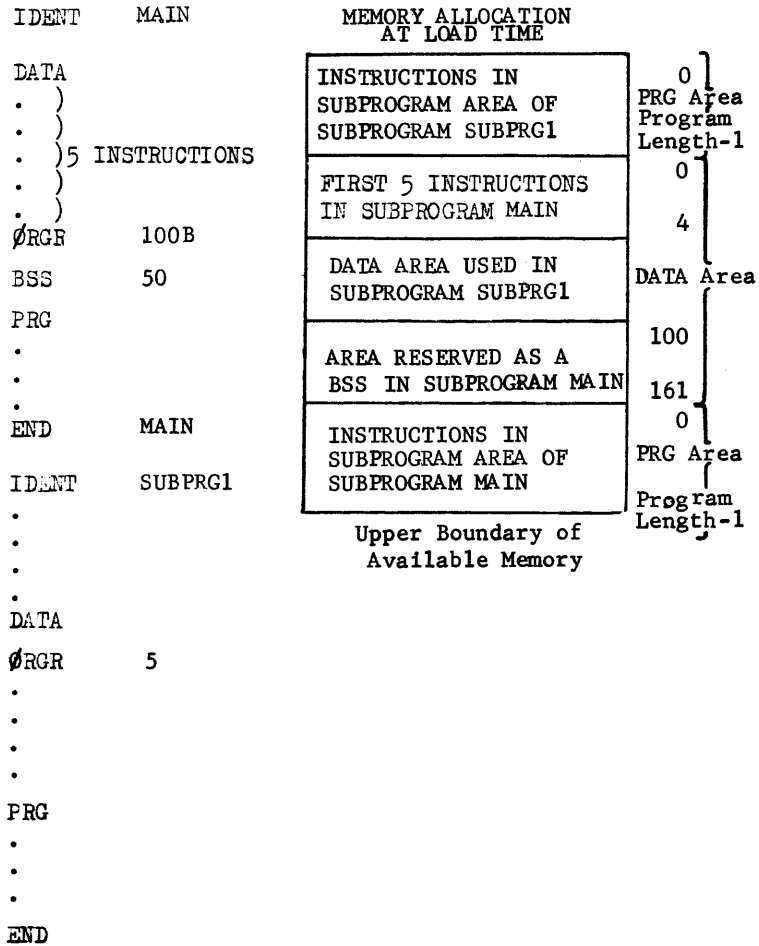
Example:

```
IDENT    BINBCD
.
.
.
DATA
BSS      2
DATA     ØCT    273
.
.
.
END
```

NOTE: The total DATA area must be defined in the first subprogram loaded.

15.3.2 (cont.)

Example:



15.3.3 Return Assembly Control to Subprogram PRG Area

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	PRG	

Description: All instructions that follow are to be assembled in the subprogram area.

The PRG instruction may be used to signal the end of the DATA or the COMMON areas.

Example:

IDENT	BLOCKER
DATA	
SCALEF	ØCT 273
	ØCT -0
	COMMON
INBUFF	BSS 100
PRG	
.	
.	
.	
END	

15.3.4 COMMON Area

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	COMMON	

Description: The instruction labels and reserves space in the common area.

No information can be put into the area at assembly time. If this is attempted, an error listing is given. COMPASS assumes that a PRG card had been included before the instruction and resumes assembly in the subprogram area.

The COMMON area is shared by all subprograms at execution time.

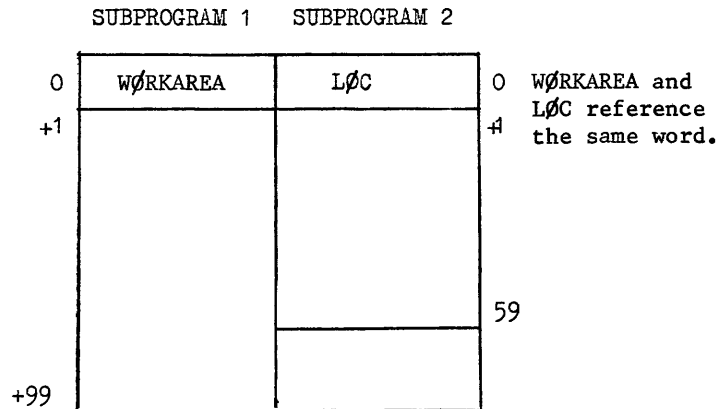
Note: COMMON is the same for all subprograms. If it is desired to have separate areas, ORGR instructions must be used to separate them.

```

Example:
IDENT      TEST
.
.
.
COMMON
WORKAREA  BSS      100
.
.
.
END
IDENT      SUBEDIT
.
.
.
COMMON
LOC       BSS      60
.
.
.
END
  
```

In this program, the common area will be overlapped by the two subprograms.

COMMON



This can be used to reference the same words in common in two subprograms.

15.3.4 (cont.1)

Example: Setting up common area in 2 subprograms

```

IDENT          MAX
COMMON
TEMP          BSS          10
CTABLE        BSS,C       6
PRG
.
.
.
END
IDENT          SUBPRG
.
.
.
COMMON
QTOTAL        BSS          12
FLAGS         BSS,C       4
PRG
.
.
.
END

```

The area will be set up as follows:

TEMP					QTOTAL
TEMP+1					QTOTAL+1
TEMP+2					QTOTAL+2
TEMP+3					QTOTAL+3
TEMP+4					QTOTAL+4
TEMP+5					QTOTAL+5
TEMP+6					QTOTAL+6
TEMP+7					QTOTAL+7
TEMP+8					QTOTAL+8
TEMP+9					QTOTAL+9
	CTABLE	CTABLE+1	CTABLE+2	CTABLE+3	QTOTAL+10
	CTABLE+4	CTABLE+5	not used by MAX		QTOTAL+11
	FLAGS	FLAGS+1	FLAGS+2	FLAGS+3	FLAGS

15.3.4 (cont.2)

Note: The only instructions which can be used in the COMMON area are as follows,

BSS	ØRGR
BSS,C	IFT
EQU	IFN
EXT	IFF
ENTRY	IFZ

COMMON is terminated by PRG
DATA
or END

15.3.5 ORGR Instruction

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	ØRGR	m

Description: This instruction controls the relocatable address for storage of instructions, constants, or the reservation of space in any of the three storage areas. The location field is ignored by COMPASS, but printed on the listing. Any symbol used in the address field must have been previously defined in the storage area being referenced. If COMPASS is assembling into one area, and an ORGR occurs with a different area relocatable symbol in the address field, an error results. COMPASS ignores the ORGR, but puts an error flag on the listing.

Example:

	IDENT	ØRGRT
	DATA	
	BSS	2
DTAG	ØCT	227
	.	
	.	
	PRG	
	.	
	.	
A	ØRGR	DTAG+1
	.	
	.	
	END	

Note: the error flag "A" set to show the error in the address field. DTAG is in the DATA area, and cannot be used in an ORGR statement in the SUBPROGRAM area.

15.3.5 (cont.)

```

ii)          IDENT      HARRY
              ØRGR      100B
              LDA       CIT
              .
              .
              .
              END
  
```

The first instruction (LDA CIT) will be assembled in Location 100, and the rest will be stored following it.

```

iii)          IDENT      ØGRTEST
              ENTRY      START      STORAGE ADDRESS
              ØRGR      50
START         UJP       **          00062
              LDA       CIPØ
              STA       CIPØBLØCK
              .
              .
              .
              UJP       02010B
CONTABLE     ØCT       0,-1        00076 and 77
              ØRGR      *+50
INPFLAG     ØCT       0           00162
              ØRGR      INPFLAG+20
ØNE         ØCT       1           00206
              END       START
  
```

15.4 STORAGE RESERVATIONS

This is made in the area currently being used. The address field will determine the number of words or character positions to be reserved.

15.4.1 Word Block

LOCATION	OPERATION/MODIFIERS	ADDRESS FIELD
	BSS	m

m = a constant, a symbol, or an address expression.

- Description:** (i) The instruction reserves and labels a block of word storage. A symbol in the location field is the 15-bit, relocatable word address of the first word in the block of storage.
- (ii) The address field specifies the number of locations to be reserved. It may be

(a) a constant

```
Example:  STATUS    BSS    2
          DISKBUF   BSS    745
```

(b) a symbol

```
Example:  VARINP    EQU    5
          INPAREA   BSS    VARINP
```

Note that the symbol must not be a relocatable address, or an error results:

```
          VARINP    ØCT    5
A        INPAREA   BSS    VARINP
```

(The "A" indicates the address field error.)

(c) An address expression which results in a non relocatable value.

```
Example:  VARINP    EQU    5
          INPAREA   BSS    VARINP+7
```

If the symbol is a relocatable address, an error results.

```
          VARINP    ØCT    5
A        INPAREA   BSS    VARINP-2
```

(The "A" indicates the address field error).

(iii) The double asterisk is illegal

```
i.e.     PRINTBUF  BSS    **
```

Description: (Continued)

- (iv) If an address field is zero or in error, the symbol is defined, but no storage is reserved.

Example:

TAG1	BSS	1
TAG2	BSS	**
TAG3	BSS	1
TAG4	BSS	2

The second instruction is illegal, and no location will be reserved for TAG2, but TAG2 is defined. It will reference the same word as TAG3.

i.e.

TAG1	
TAG2 and TAG3	
TAG4	
TAG4+1	

- (v) All symbols used in the subprogram must have storage allocated to them:

	IDENT	MATRIX
	LDA	ØUNT
	INA	
	STA	ØUNT
	.	
	.	
	.	
	.	
	.	
ØUNT	BSS	1
	.	
	.	
	.	
	.	
	.	
	END	

- (vi) NOTE: Where no symbol is used, storage is reserved but not labelled. It may be referenced from other labelled locations.

Example:

TAG1	ØCT	4
	BSS	3
TAG2	ØCT	27
	BSS	3
TAG3	ØCT	14

15.4.1 (cont.2)

Description: (Continued)

The block of storage set up will be:

TAG1	00	00	00	04

TAG2	00	00	00	27
TAG3	00	00	00	14

The location marked *** may be referenced as:

- TAG1+2
- or TAG2-2
- or TAG3-6

15.4.2 Character Block

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	BSS,C	m

Description: Reserves and labels a block of character storage in the area currently in use.

- (i) A symbol in the location field is the 17 bit, relocatable character address of the first character in the block.
- (ii) The address field specifies the number of characters to be reserved. It may contain
 - (a) a constant

Example: CTAG1 BSS,C 6
 CTAG2 BSS,C 2

Storage will be reserved as follows:

CTAG1	CTAG1+1	CTAG1+2	CTAG1+3
CTAG1+4	CTAG1+5	CTAG2	CTAG2+1

(ii) Continued

(b) a symbol

Example: SIZE EQU 5
 PARLIST BSS,C SIZE

Note that the symbol must not be a relocatable address or an error results.

 SIZE OCT 5
 A PARLIST BSS,C SIZE

(The "A" indicates the address field error).

(c) an address expression which results in a non-relocatable value.

Example: SIZE EQU 15
 PARLIST BSS,C SIZE-3

(iii) If a BSS,C instruction is followed by a BSS 0 instruction, it forces any following character reservation to a new word, even if the last character word is not filled.

Example: INPFLAGS BSS,C 6
 BSS 0
 ØUTFLAGS BSS,C 2

Storage is reserved as follows:

INPFLAGS	INPFLAGS+1	INPFLAGS+2	INPFLAGS+3
INPFLAGS+4	INPFLAGS+5	Not Used	
ØUTFLAGS	ØUTFLAGS+1	Not Used	

15.5 ENTRY AND EXTERNAL INSTRUCTIONS

15.5.1 ENTRY Pseudo Instruction

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	ENTRY	m ₁ , m ₂ , m ₃ , --- -- m _n

Description The location field should be blank, but if a symbol does appear, it will be ignored by COMPASS. The address field contains one or more location names separated by commas. No blanks may occur. The field terminates at the first blank, or at Column 73. If there are more entry points to be defined than will fit on one card, a second card can be used. Each of the address field location names contains a symbol defined as a subprogram relocatable word address by appearance in a location field elsewhere in the subprogram.

Example:

```

IDENT          ONE
ENTRY          START, INT0
START UJP      **
              ENA      0
              .
              .
INT0  UJP      **
              .
              .
              END      START
    
```

Note that more than one entry card can be used.

```

IDENT          TWO
ENTRY          START
ENTRY          INT0
START UJP      **
              .
              .
INT0  UJP      **
              .
              .
              END      START
    
```

15.5.2 EXTERNAL Pseudo Instruction

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	EXT	m ₁ , m ₂ , m ₃ , --- -- m _n

Description: The location field should be blank. Any symbol appearing there will be ignored by COMPASS. The address field contains one or more location names up to Column 73. These must be separated by commas. No blanks may occur. A symbol in the address field may not be defined in the subprogram in which the EXT instruction appears. The EXT instruction can only reference location names in the subprogram area of another subprogram. It cannot reference DATA or COMMON areas.

15.5.2 (cont.1)

Example:

```
IDENT    MAIN
ENTRY    NUMBER
.
.
.
NUMBER   ØCT    37
.
.
.
END
IDENT    SUBPRØG
EXT      NUMBER
.
.
.
LDA      NUMBER
.
.
.
END
```

All subprograms within a main program are assembled independently, and all symbols in a subprogram are local to that program **only**, unless declared as external symbols in another subprogram, and as entry points in the former program.

Example:

```
IDENT    DRIVER01
.
.
.
LDA      TAG
.
.
.
TAG      ØCT    14
END
IDENT    DRIVER02
.
.
.
LDA      TAG
.
.
.
TAG      ØCT    27B
END
```

The two Symbols (TAG) are not linked in any way. Each will be referenced only by the subprogram in which it appears.

15.5.2 (cont.2)

BUT: If they refer to the same symbol, the Program could appear as follows:

```
IDENT    EDIT
EXT      CARDBUFF
.
.
LDA      CARDBUFF
.
.
END

IDENT    READ
ENTRY    CARDBUFF
.
.
LDA      CARDBUFF
.
.
CARDBUFF  OCT      24
.
.
END
```

Address arithmetic is not permissible with external symbols

e.g. LDA CARDBUFF+2

But, address modification is permissible

e.g. LDA CARDBUFF,3

15.5.3 SCOPE Loading of Subprograms

The Assembler establishes links between the subprogram as directed by the EXT and ENTRY instructions. These linkages are then set up by the Loader (a part of the SCOPE monitor) when the assembled program is loaded prior to execution.

If an external is referenced in a subprogram, but there is no ENTRY for it in any other subprogram, SCOPE will look through the Library Tape to see if it can find a Library Routine to enter.

```
IDENT    QUAD
EXT      SQRTF
.
.
RTJ      SQRTF
.
.
END

IDENT    DISC
.
.
END
```

If it can find no Library routine of the same name, it will give an error symbol, and terminate the run. (This is done at Load Time, not in Assembly). The error symbol appears on the listing after the LOAD card print out e.g. LOAD, 56

```
DISC    UD    SQRTF
```

15.5.3 (cont.)

The COMPASS assembly listing of a subprogram containing external symbol references will have the usual format, except that the address field will be prefaced by an X.

Example:

00010 00 1 X00003

The digits following the X are the relocatable word address of a previous instruction in the subprogram area which references the external symbol. The first (or only) reference to the external symbol will contain X77777 in the address field. COMPASS thus produces a "threaded list" of instructions referencing the external symbol.

Example:

	<u>Program</u>	<u>Listing</u>
IDENT	TEST	
EXT	CIØ	
.		
.		
.		
ENA	10B	00000 14 6 00010
.		
.		
.		
RTJ	CIØ	00010 00 7 X77777
.		
.		
.		
RTJ	CIØ	00030 00 7 X00010
.		
RTJ	CIØ	00050 00 7 X00030
.		
END		

The address of the last instruction referencing the external is placed by COMPASS into the XNL Loader Card to begin the backward threaded list. When the loader loads the program, it enters the actual address of the external symbol into the address portion of each instruction referencing the external. It does this by saving the address portion of the last instruction referencing the external symbol, it then replaces the address with the actual address of the external symbol. The loader then, repeats the process using the saved address as the new last instruction address referencing the external symbol. The process continues until the address of the next instruction in the list is 77777, which indicates all instructions referencing this external symbol have been modified.

Thus, if CIØ is loaded at address 00106, the program will be loaded into storage as:

00000	14600010	(ENA 10B)
.	.	.
00010	00700106	(RTJ CIØ)
.	.	.
00030	00700106	(RTJ CIØ)
.	.	.
00050	00700106	(RTJ CIØ)

15.6 SYMBOL DEFINITION BY EQUIVALENCING

15.6.1 Introduction

A symbol may be defined by equivalencing it to another symbol, a constant, or an expression. The symbol may be defined as an absolute value, a relocatable word or relocatable character address. The symbol in the location field is equivalenced to the value of the address field. A symbol which is declared an entry point must not be equated to a symbol which is declared external. When symbols are equivalenced they are identical and interchangeable.

All symbols in the address field of an equivalence must have been previously defined by the appearance in the location field of a preceding instruction, or in an EXT pseudo instruction.

15.6.2 Word Equating

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	EQU	<i>m</i>

Description: The symbol is equivalenced to another symbol, a 15-bit word address, or a 15-bit value. The symbol in the location field will be non-relocatable or relocatable, as determined by the address field.

If the location field is blank, an error occurs.
The address field may contain

- (i) An integer, modulo $2^{15}-1$ (15 bits or less)
- (ii) A symbol, previously defined
- (iii) An address expression containing symbols previously defined.

If a symbol in the address is defined as relocatable in a given area, the symbol in the location field will also be relocatable in that area.

Examples:

SYMBOL	EQU	57641B
DATE	EQU	27B
TEMP	BSS	1
SYM	EQU	TEMP
TEMP2	EQU	SYM+6

Assembled as follows:

SYMBOL	57641	
DATE	00027	
00165	TEMP	Storage reserved
SYM	00165	
TEMP2	00173	

TEMP, SYM and TEMP2 are relocatable.

15.7.5 New Page EJECT Instruction

LOCATION	OPERATION/MODIFIERS	ADDRESS FIELD
	EJECT	

Description: The page being printed is fed through the printer, and the line following the EJECT instruction will be the first line of listing on the new page. The address field must be blank or an error will occur, although comments may be inserted from column 41.

15.7.6 TITLE Instruction

LOCATION	OPERATION/MODIFIERS	ADDRESS FIELD
	TITLE	<i>title to be used</i>

Description: Normally the name of the subprogram will appear at the top of each page of listing of the subprogram. If another title is required instead, it can be inserted using this instruction. The heading obtained from the IDENT or previous TITLE instruction is replaced, and the first page following the TITLE instruction will have the new heading. If the new heading is to be inserted immediately, the instruction EJECT should immediately follow TITLE. If the new title is to be used on the first page of the listing, the TITLE instruction must immediately follow IDENT. The title must be contained in columns 20-72 of the address field.

Examples:

(i) IDENT TEST
TITLE TEST FØR ILLEGAL CHARACTERS
.
.
.

This will cause the full title to be printed on the first and subsequent pages of the listing of the subprogram.

(ii) IDENT TEST
TITLE TEST FØR ILLEGAL CHARACTERS
.
.
.
TITLE PRINT ØUT CHARACTERS FØUND
EJECT
.
.
.
END

The first title will be printed on all pages until the second TITLE instruction is found. A new page will be begun by the EJECT instruction, and it will bear the new title.

15.7.7 Comments

When COMPASS detects a card with an asterisk in column 1, it prints the content of the card as a comment. No other action is performed.
Note: The asterisk itself is not printed. (See also Section 2.2.3.1)

SCOPE ORGANIZATION OF INPUT/OUTPUT

16.1 INTRODUCTION

16.1.1 Programmer Units

16.1.2 Scratch Units

16.1.3 Systems Units

16.2 CENTRAL INPUT/OUTPUT ROUTINE

16.2.1 Introduction

16.2.2 Calling Sequences

16.2.3 Input/Output Operations

16.2.4 Tape Control Operations

16.2.5 Unit Status Requests

16.2.6 Format Selection

16.2.7 Page Control of the Line Printer

16.1 INTRODUCTION

Under SCOPE, Input/Output devices are specified by Logical Unit Numbers (LUN's) which are organized according to function. The programmer or operator assigns the logical unit to a particular type or unit of hardware, through SCOPE control.

Logical Units may be specified as

- (i) Programmer units,
- (ii) Scratch units,
- or (iii) System units.

16.1.1 Programmer Units

They are for general purpose use by the programmer, and they are unrestricted as to use in any run in a job.

Once defined, the definition of the programmer unit is fixed for the whole job. They are released by SCOPE at the end of the job, unless saved by the programmer by the use of a SCOPE unload card.

Programmer units are numbered 1-49.

16.1.2 Scratch Units

Scratch units must be defined for each run, and are released at the end of the run. They are assigned and used by Library programs, and may be accessed by the programmer for temporary use.

Scratch units are numbered 50-55.

They cannot be saved by the programmer.

16.1.3 Systems Units

Systems units are assigned to specific physical equipment within SCOPE, but these assignments may be altered by the operator. They are used for certain common functions, and may be protected by SCOPE from input-output requests which might destroy their contents.

Systems units are numbered 56-63, as follows:

- 56 Load and Go (for storage of object decks from assembly, prior to Loading and Execution)
- 57 Accounting
- 58 Comments from operator (only read requests allowed)
- 59 Comments to operator (only write requests allowed)
- 60 Standard Input (protected against writing, etc.)
- 61 Standard Output - holds listable output
- 62 Standard punch - output from COMPASS, etc.
- 63 Library.

16.2 CENTRAL INPUT/OUTPUT ROUTINE

16.2.1 Introduction

Input/Output requests in COMPASS programs are written as calling sequences for monitor routines controlled by a central Input/Output routine called CIO.

CIO performs the following functions:

- (i) Selects an available channel
- (ii) Rejects requests if
 - (a) the unit is not available (e.g. due to an operator error).
 - (b) no access channel is available
 - (c) an illegal instruction (function code) is given.
- (iii) Provides the current status for all requests.
- (iv) Initiates all I/O operations, and then returns control to the main program so that processing may continue while the I/O operation is carried out.
- (v) Responds to external interrupts, and transfers control to a routine specified by the programmer.

16.2.2 Calling Sequences

Input/Output operations are specified by entering an octal function code and other parameters into a calling sequence. The function codes are

Function Code	Request
01 02	Read Write
03 04	Read backwards Rewind
05 06	Unload Backspace
07 10	Space forward past 1 EOF Space backwards past 1 EOF
11 12	Write EOF Erase
13 14	Status Format

There are 4 operations performed with CIO

- (i) I/O operations
- (ii) Tape control operations
- (iii) Unit Status operations
- (iv) Format selection operations.

16.2.3 Input/Output Operations

The function codes used by Input/Output operations are:

- (a) 01 READ n words, starting at FWA (First word address)
- (b) 02 WRITE n words, starting from FWA
- (c) 03 READ BACKWARDS, n words, and store backwards, starting at FWA+n-1.

Calling Sequence:

Input/Output operations are requested by the following sequence of instructions.

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD	COMMENTS
(L)	RTJ	CIØ	
(L+1)	Function Code	Logical Unit Number, Interrupt Indicator	
(L+2)	UJP	Reject Address	
(L+3)	Mode	First Word Address	
(L+4)		Number of Words	
(L+5)		Interrupt Address	
(L+6)	FIRST INSTRUCTION EXECUTED UPON RETURN FROM CIØ		
	Note: CIØ must be declared external somewhere in the subprogram		
	also if Interrupt Indicator = 0,		
	no Interrupt Address is specified and the		
	first instruction to be executed upon return		
	from CIØ should be located at L+5 instead		
	of L+6.		

- Notes:
- (a) The logical unit number (LUN) is defined by an EQUIP card, and may be 1-63, depending on function code.
 - (b) The function code is an octal number, 1-14 as defined previously.
 - (c) The interrupt indicator selects an interrupt on normal or abnormal end of operation when set.
 - 0 = no interrupt
 - 1 = interrupt on ABNORMAL end of operation only (end of tape, EOF mark, load point, parity error, lost data for mag. tape.)
 - 2) interrupt on an end of operation, whether normal
 - 3) or abnormal.
 - (d) JUMP is any legitimate jump to the reject address.
 - (e) Interrupt address is the address of a closed subroutine to which control goes when the specified interrupt occurs.
 - (f) Reject address is a symbolic address to which control goes in the event of CIØ rejecting the calling routine. In the event of a reject because a channel or a unit is not available the A register will contain zero. If the reject is due to an illegal function code, the A register will contain a non-zero quantity. For both types of rejects, the Q register will contain the status of the unit.

16.2.3 (cont.1)

- (g) The mode designates the method of recording, and is given as an octal number.

If no mode is designated, binary mode is assumed and density is under the control of the operator.

Code	Density	Parity
00	Do not select a new mode	
40	none	even
41	none	odd
50	low	even
51	low	odd
60	medium	even
61	medium	odd
70	high	even
71	high	odd

- (h) First Word Address (FWA) is the symbolic address of the first word in the input or output area.
- (i) Number of words (n) is the decimal number of words to be transmitted.
- (j) If no interrupt is specified, the normal return is written in location L+5.
- (k) On interrupt, before control is passed to the interrupt address, SCOPE saves A, Q and the 3 index registers. On completion of the interrupt subroutine, the programmer must return control to SCOPE, which will then restore the A, Q and Index registers to their original values. Note that no values obtained in the subroutine can be returned to the main program in these registers.

Example:

Write at 556 bpi in BCD on a magnetic tape that has previously been defined as LUN 20, a 27 word block of data commencing at the symbolic address ORIGIN.

After successfully initiating the write operation, jump to the symbolic address PROCESS and continue the execution of the program.

In the event of an abnormal end of operation, go to the symbolic address ABANDON; and if the write request is rejected, jump to PAUSE.

16.2.3 (cont.2)

Answer:

OPERATION MODIFIERS													ADDRESS FIELD																	
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
RTJ													CIØ																	
02													20, 1																	
UJP													PAUSE																	
60													ØRIGIN																	
													27																	
													ABANDØN																	
UJP													PRØCESS																	

Use of interrupt facility (See also Section 21.3)

If an interrupt address is specified, and the interrupt indicator is non zero, control transfers to the interrupt address at the end of the operation, or upon an abnormal condition interrupt.

Before giving control to the interrupt address, SCOPE saves the contents of the A, Q and three index registers. It then enters the current condition and status of the unit in the A and Q registers respectively.

C = Condition of unit
 0 = dynamic
 1 = static

LF = Last function code (other than 13) given for the unit

TCA = Terminating character address of data transmission contained in the Buffer Control Register.

LS = Logical Status of the Unit

00 = (a) if a status request, unit is static, channel is available
 (b) for reject return, hardware reject
 (c) for normal return, unit is dynamic

01 = Channel is not available for any requests

10 = previous operation is incomplete

11 = previous operation is complete, but an interrupt request is being processed.

LC = Last channel to which the unit was connected (or is still connected).

R = Retention code (from standard 3200 tape label)
 0 = tape may be used for output
 1 = contents of tape should not be destroyed.

STATUS See following chart.

16.2.3 (cont.3)

UNIT STATUS TABLE

STATUS BIT	MT	CR	CP	PR	PT	TY
00	Ready	Ready	Ready	Ready	Ready	Ready
01	Busy	Busy	Busy		Busy	Busy
02	Write enable					
03	File mark	EØF				
04	Load point					
05	EØT	Hopper empty			Tape supply low	
06	<u>DENSITY</u> 00 = low					
07	01 = med. 10 = high					
08	Lost data	Fail to read	Fail to feed			
09	End of operation					
10	Parity error	Reader error	Compare error			Parity error
11	Binary mode	Binary card	Binary mode		Binary mode	
12		Stacker full or jammed				
13						
14						
15						
16						

NOTES: (a) Density is signified by combinations of 2 bits - bits 6 and 7 - as shown.

(b) Bits 13-16 are not used.

16.2.3 (cont.5)

Example,

Read 500 words from tape LUN 3 into a buffer commencing at BUFF. When the Operation has been initiated, continue program execution. When the operation is complete, set location FLAG to a non-zero value. If the read request is rejected because of an illegal code, jump to ABANDON. If it is rejected due to channel or unit not being available, jump to PAUSE.

<u>Answer:</u>	RTJ	CIO
	01	3,2
	01	REJ
	51	BUFF
		500
		INTER
	LDA	
	.	
	.	
	.	
	INTER UJP	**
	ENA	1111B
	STA	FLAG
	UJP	INTER
	REJ AZJ, EQ	PAUSE
	UJP	ABANDON

16.2.4 Tape Control Operations

The codes used in tape control calling sequence are

04	REWIND
05	UNLOAD
06	BACKSPACE
07	SPACE FORWARD PAST ONE EOF MARK
10	SPACE BACKWARDS PAST ONE EOF MARK
11	WRITE EOF
12	ERASE

The calling sequence:

Location L	RTJ	CIO
L+1	function code	LUN, INTERRUPT INDICATOR
L+2	JUMP	REJECT ADDRESS
L+3		INTERRUPT ADDRESS
L+4	NORMAL RETURN	

Note: (a) If no interrupt is requested, the normal return is written in Location L+3.

(b) The notes for I/O control apply here also.

16.2.4 (cont.)

Examples:

(i) To rewind logical unit 56

RTJ	CIO
04	56
UJP	*-2
normal return	

(ii) To write an end-of-file on LUN 20

RTJ	CIO
11	20
UJP	*-2
normal return	

(iii) To space forward past an end-of-file mark on LUN 17

RTJ	CIO
07	17
UJP	*-2
normal return	

Notes: (i) The direction of tape motion following a BACKSPACE request depends upon whether the last operation was a READ or a READ BACKWARDS operation -

(a) If the last operation was a READ operation, the tape will move backwards.

(b) If it was a READ BACKWARDS operation, the tape will move forward one record.

Other motion requests indicate the true direction of the tape and are not affected by READ BACKWARDS.

(ii) Tape control operations require the channel only during initiation of the function. They do not cause the channel to be busy while the function is carried out. However, if an interrupt at the end of operation is requested, CIO considers the channel to be busy until the interrupt occurs.

16.2.5 Unit Status Requests

For codes returned as status replies, see table on (cont.3) of section 16.2.3.

The calling sequence:

Location L	RTJ	CIO
L+1	13	LUN, Dynamic flag
L+2	NORMAL	RETURN

NOTES: (a) The function code in L+1 is always 13.

(b) SCØPE provides the status in the Q register, and the current condition in the A register (A is negative if the unit is static, positive if it is dynamic).

(c) Dynamic flag:

If this is non-zero, the unit is interrogated for status unconditionally. (Status is given if unit is

16.2.5 (cont.)

busy or not). If the flag is zero:

- (i) If it is not busy, the status of the last completed operation is given.
- (ii) If the unit is busy, the current status is returned.

16.2.6 Format Selection

The codes used in this sequence are -

- 1 = BCD
- 2 = BINARY
- 3 = LOW
- 4 = MEDIUM
- 5 = HIGH

The calling sequence:

RTJ	CIO
14	LUN, FORMAT CODE (as above)
JUMP	REJECT ADDRESS
NORMAL	RETURN

16.2.7 Page Control of the Line Printer

The first character of the output buffer is used to position the paper prior to and after printing. This is accomplished by the following sequence of events.

- (a) The first character is removed and replaced with a blank.
- (b) The corresponding function code is found and selected.
- (c) The output buffer is printed.

16.2.7 (cont.1)

The character codes used in Page Control are:

<u>User Character</u>	<u>Action Before Print</u>	<u>Action After Print</u>
1	Skip to channel 8	Space 1 line
2	Skip to channel 7	Space 1 line
3	Skip to channel 6	Space 1 line
4	Skip to channel 5	Space 1 line
5	Skip to channel 4	Space 1 line
6	Skip to channel 3	Space 1 line
7	Skip to channel 2	Space 1 line
8	Skip to channel 1	Space 1 line
A	No space	Skip to channel 8
B	No space	Skip to channel 7
C	No space	Skip to channel 6
D	No space	Skip to channel 5
E	No space	Skip to channel 4
F	No space	Skip to channel 3
G	No space	Skip to channel 2
H	No space	Skip to channel 1
blank	No space	Space 1
0	Space 1	Space 1
-	Space 2	Space 1
*	No space	No space
other	No space	Skip to channel 1

Examples:

(a) To advance paper to the top of the new page

RTJ	CIØ
02	61
RTJ	REJX
00	PAGE
00	1

PAGE	BCD	1,1
------	-----	-----

16.2.7 (cont.2)

(b) To advance one line

RTJ	CIØ
02	61
RTJ	REJX
00	DS
00	1

DS BCD 1,

Note: If unit 61 is assigned to tape by the operator and the tape is listed later or is listed off-line, trouble may develop because of the short print record (4 chars). Therefore it is recommended that all records be at least 24 characters so that they won't be considered 'noise' records by tape input routines.

STUDENT NOTES

SCOPE CONTROL CARDS

- 17.1 INTRODUCTION
- 17.2 SEQUENCE CARD
- 17.3 JOB CARD
- 17.4 ENDScope STATEMENT
- 17.5 ENDREEL STATEMENT
- 17.6 CTO STATEMENT
- 17.7 REWIND STATEMENT
- 17.8 UNLOAD STATEMENT
- 17.9 EQUIP STATEMENT
 - 17.9.1 Hardware Definition
 - 17.9.2 Equating Logical Units
 - 17.9.3 Physical Unit Assignment
- 17.10 TRANSFER STATEMENT
- 17.11 LOAD STATEMENT
- 17.12 COMPASS LIBRARY CALLING STATEMENT
- 17.13 RUN STATEMENT
- 17.14 DIAGRAMMATIC DECK

17.1 INTRODUCTION

SCOPE control cards have a 7,9 punch in column one.
There must be no other punchings in column 1.

Columns 2 through 80 contain Hollerith information or blanks. The first information on each card must be the statement name, followed by a comma.

e.g. 7
 9RUN,

 7
 9JOB,

17.2 SEQUENCE CARD

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
7SEQUENCE, j		

The sequence statement assigns a number (j) to the job which it precedes. j must lie between 1 and 999. This card is normally supplied by the operating staff.

An EOF must precede each SEQUENCE statement except the first on input.

When a SEQUENCE statement is detected, the statement is typed out on the console typewriter, and listed on the standard output unit. SCØPE writes an EOF on the standard output tape (when tape is being used), and on the punch tape (if assigned), and releases all programmer units held from the last job.

SCØPE also closes off the last job's accounting, and opens a new accounting record for the new job.

If the sequence card is not followed by a JØB card, the job is terminated, and SCØPE searches forward until it finds an EOF. The following statement must be a SEQUENCE, ENDREEL or ENDSØPE. If it is a SEQUENCE followed by a JØB card, SCØPE proceeds normally.

After SEQUENCE is read, SCØPE pauses for comments from operator to be typed in on the console typewriter. These enable the operator to specify the services for program execution (e.g. special EQUIP statements). This is the only opportunity the operator has to enter such statements.

Note: If the column following "j" contains a comma, comments may follow.

Example:

```
7SEQUENCE,027, COMMENT
```

17.3 JOB CARD

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
JOB, c, i, t	NS, NP, ND	

The symbols used in the statement are interpreted as follows:

- c charge number, 0-8 characters
- i programmer identification, 0-4 characters
- t time limit in minutes for the entire job, including all operator setting up, idle time, etc.

NS indicates a single non stacked job. If NS is specified, NP is implied also.

All system units are rewound and unloaded, making all I/O units available to the programmer.

NP Suppresses system I/O protection for stacked job.

ND Suppresses the normal post execution dump in octal of the non system part of memory should abnormal termination of the job occur.

The c, i, and t fields are mandatory. If a field is blank, the comma showing this must appear.

e.g. $\begin{matrix} 7 \\ 9 \end{matrix}$ JOB,c,,t

The job card is written on the standard output unit, and also on the CTØ (comments to operator) if the job is part of a stack. It must be immediately preceded by a SEQUENCE card or the run will be terminated.

17.4 ENDSCOPE STATEMENT

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
3	ENDSCOPE	

This indicates that a SCOPE run is to be terminated. On the standard input unit, it should follow the EOF terminating the last in the stack. This card is normally supplied by the operations staff.

The card is listed on the CTØ unit and on ØUT, and the standard input unit is unloaded, if magnetic tape.

The library tape is rewound, and the accounting file is closed off.

A double EOF and one BCD word, consisting of ER^^ are written on the standard output tape (when tape is used), which is then unloaded. If the punch tape is assigned, it is treated in the same way.

When all action is completed, the computer stops.

17.5 ENDREEL STATEMENT

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
ENDREEL		

The statement terminates a reel of magnetic tape containing a job stack. (Normally it will be placed in a card job stack by the operations staff during card to tape operations preparing a tape for use as a standard input tape.)

SCOPE requires an EOF both immediately before and after ENDREEL.

When the statement is detected, SCOPE prints a message on the CTØ unit requesting the operator to mount the next reel of input, and halts the computer until the operator takes the action.

17.6 CTØ STATEMENT

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
$\frac{7}{9}$ CTØ,statement		

The programmer may provide instructions of messages to the operator. The message is punched in Hollerith.

Examples: $\frac{7}{9}$ CTØ,PLEASE UNLOAD TAPE TWO.

$\frac{7}{9}$ CTØ,SNAP DUMPS WILL OCCUR.

The message is printed on the CTØ, and also listed on ØUT. CTØ cards may be placed in the deck where SCØPE control cards may appear, except as follows:

- (i) Before or after SEQUENCE, ENDSCØPE or ENDREEL.
- (ii) After CØMPASS or FØRTRAN.
- (iii) Between RUN and last data card.

17.7 REWIND STATEMENT

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
7	REWIND, U, U ₂ , . . . , U _n	

The magnetic tapes specified are rewound to load point. U is the logical unit number, and may be 1 through 57 or 63. The statement is copied onto the standard output unit, and on the CTØ unit.

If U is not 1-57,63 or not a magnetic tape unit, the request is ignored for that unit, but the rest of the units on the card are processed.

Example: ⁷REWIND,21,42,01,55
 ₉

17.8 UNLOAD STATEMENT

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
<u>UNLOAD, U₁, U₂, . . . , U_n</u>		

The Logical units, U (1-57), may be unloaded by the programmer. The statement acts similarly to the REWIND, except that the unit is unloaded after rewinding.

Example: ⁷UNLOAD, 21, 42, 01, 55

17.9 EQUIP. STATEMENTS

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
7	EQUIP, X=d ₁ , X=d ₂ , etc	

Where X = the logical unit number

d = a declaration about the unit.

17.9.1 Hardware definition

7
9 EQUIP, X₁=hh₁, X₂=hh₂, etc.

hh is a hardware type mnemonic

X is a L.U.N.

<u>Mnemonic</u>	<u>Type</u>
MT	Magnetic Tape
CR	Card Reader
PR	Printer
CP	Card Punch
TY	Console typewriter
PT	Paper tape station
DP	Disk Pack

SCOPE assigns the LUN to an available equipment of the specified type.

If no equipment is available, a diagnostic is given, and the job is terminated.

Examples: 7
9 EQUIP, 51=MT

7
9 EQUIP, 51=MT, 43=PR, 26=PT

17.9.2 Equating Logical Units

Logical units are equated by this statement

7
9 EQUIP, X₁=X₂

A system unit (57-63) may not be specified on the left hand side of the statement. If it is, the job is terminated.

Examples: 7
9 EQUIP, 43=60 is permissible

but 7
9 EQUIP, 60=43 is illegal

7
9 EQUIP, 22=MT }
7
9 EQUIP, 23=22 } Here both LUNS 22 and 23 will
reference the same magnetic tape.

17.9.3 Physical Unit Assignment

${}^7_9\text{EQUIP},X=hhC_c E_e U_{uu}$

c = channel number (0-7), prefixed by C

e = equipment number (controller), prefixed by E

uu = unit number (device), prefixed by U

Example:

${}^7_9\text{EQUIP},15=\text{MTCOE2UO3}$

It is possible to omit some parameters in the statement.

The following table sets out permissible combinations.

hh	C _c	E _e	U _{uu}
x			
x	x		
x	x	x	
x	x	x	x
	x	x	x
	x	x	

Use of non-existent c, e or uu, will cause a diagnostic and termination of the job.

Example:

Assign PUN (LUN 15) and LGO (Load and Go, 56) to the physical unit on channel 0, Equipment 1, and unit 7.

Answer:

${}^7_9\text{EQUIP},15=\text{MTCOE1UO7}$

${}^7_9\text{EQUIP},56=15$

or ${}^7_9\text{EQUIP},15=\text{MTCOE1UO7},56=15$

17.10 TRANSFER STATEMENT

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
XFER,U		

U = Magnetic Tape Unit,
defined as 1-56,
or undefined.

Description: SCOPE transfers all the information following the XFER statement from the Standard input unit (INP) to the magnetic tape LUN U, until another SCOPE statement is encountered.

The records must be binary records and they are written on LUN U in odd parity. When the next SCOPE statement is found, an EOF is written on the tape, and SCOPE then backspaces over the EOF.

Uses:

- (i) Programmer binary data cards* may be transferred from INP to a magnetic tape unit. A card with a ⁷/₉ punch in column 1 is sufficient to terminate the XFER operation.
- (ii) Binary object subprograms may similarly be stored on another magnetic tape for future use. The unit must be rewound before the LOAD operation if it is a programmer or scratch unit.

Example:

⁷/₉XFER,03

.

.

Binary Data *

.

.

.

⁷

₉

The data is written on unit 3; when the ⁷/₉ card is found, SCOPE writes EOF and backspaces over it.

*Data cards must be binary data cards where

column 1 has a ⁷/₉ punch plus at least one punch in the + - 0 1 2 or 3 position. The data used is usually a subprogram binary deck.

17.11 LOAD STATEMENT

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
7	L0AD, U ₁ , U ₂ , U ₃	

(U₁, U₂ and U₃ are Mag. tape units, previously defined by EQUIP statements as LUN 1-56. If omitted, SC0PE will try to load from the INP unit 60)

Description: Not more than 3 units can be specified, and the loading is done in the order indicated. Unit U₁ will be loaded until an E0F is found. Then U₂ is loaded until E0F, and finally U₃ is loaded. If there are 3 parts on one unit, each terminated by an E0F, the unit number can be repeated:

7
9 L0AD,23,23,23

When the units designated have been loaded, the Standard input unit is examined to see if binary object subprograms follow the L0AD card.

NOTE: If the LG0 unit is being used, the L0AD card must be of the form:

7
9 L0AD,56

The L0AD statement calls the loader to load binary subprograms into memory from programmer units, scratch units, LG0 or INP. Only one L0AD statement may appear in a run.

If 56 (LG0) is specified, it will first be rewound by SC0PE. Other units must be rewound by rewind statements before loading is attempted.

Example:

7
9 L0AD,56,3,25

LG0 is rewound by SC0PE and loaded until E0F is found. Units 3 and 25 are then loaded to E0F marks.

NOTE: If only binary object programs are to be loaded (on INP), no L0AD card is necessary.

Example:

7
9 J0B,11121156,404,2

Binary object deck

7
9 RUN,1

77
88

17.12 COMPASS LIBRARY CALLING STATEMENT

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
<u>6</u> COMPASS	<u>I, P, L, X, R</u>	

Description: The COMPASS library program is called in and loaded into memory, so that source programs in COMPASS may be assembled and executed.

The parameter letters are free field, and may thus appear in any order. Each parameter must start with the character shown.

I = INPUT (the source subprogram input unit)

Specified as I = u, when u = LUN, when a source subprogram is to be loaded from a unit other than INP.

If the I parameter is absent, input is assumed to be from unit 60 - INP.

Example:

7
9EQUIP,MT=23

.
.
.

7
9COMPASS,I=23,etc.

P = PUNCH-UNIT

Specified as P = u, where the punch unit is to be assigned to an output device.

If P only occurs, punching is on the binary punch unit (PUN) - unit 62.

If the parameter is absent, no binary output is produced.

X = EXECUTE (X = u)

Assigns the Load-Go unit (LG ϕ) to logical unit U, which must have been previously defined as a MT unit.

If absent, no LG ϕ tape will be produced, and the program will be assembled and listed only.

If only X appears, output will be put on the standard LG ϕ unit (56).

L = LIST OPTION (L = u)

Output is listed on unit u, which must have been previously defined.

If L only appears, listing will be done on the OUT unit (61).

If the parameter is absent, no listing of the program is given.

17.12 (cont.)

R = REFERENCE

When R appears, a symbol reference list is produced on the assembly listing. This is an alphabetic list of all symbols used in the program, with the address, or the value, of the symbol shown. The symbol list appears immediately following the listing of the assembled program.

Example:

A	77777		P00013
ABNORMAL		EXTERNAL	P00001
B	00020		P00013
JOHN	P00002		P00011
NAME	P00011		P00013
X	00057		P00012
			P00013

SYMBOLS NOT REFERENCED

ABLE	P00013	LOC	P00007	START	P00000
------	--------	-----	--------	-------	--------

Note that X is referenced at two different locations in the program (12 and 13).

Character addresses are shown as follows:

CAN	P00166	0
BILL	P00156	1

17-Bit non-relocatable symbols are shown

TOM	00076	2
-----	-------	---

17.13 RUN STATEMENT

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
7 9	RUN,t,NM	

t = execution time in minutes

May be in the range 0 through 999.

The time is not used by SCOPE, but is entered in the installation accounting file.

If not specified, maximum time is assumed.

NM = NO MEMORY MAP

If NM appears, it suppresses the memory map that would otherwise be written on ØUT, before program is executed. (All absolute memory allocations are given in the map).

Examples:

(i) $\begin{matrix} 7 \\ 9 \end{matrix}$ RUN,2

(the execution time is assumed to be two minutes.
The memory map will appear on ØUT.)

(ii) $\begin{matrix} 7 \\ 9 \end{matrix}$ RUN,727,NM

(the execution time is assumed to be 727 minutes,
and no memory map will appear on OUT).

(iii) $\begin{matrix} 7 \\ 9 \end{matrix}$ RUN

(maximum execution time is assumed. Map will be
printed).

(iv) $\begin{matrix} 7 \\ 9 \end{matrix}$ RUN,,NM

(maximum execution time assumed, and no map will
be listed on ØUT).

17.14 DIAGRAMMATIC DECK

7
9 SEQUENCE,001

7
9 JOB,3200,STORMA,10

7
9 EQUIP,56=MT

7
9 FORTRAN,(Parameters)

PROGRAM MAIN

.
. .
. .
. .

CALL START

.
. .
. .
. .

END

FINIS

} FORTRAN Main Program

7
9 COMPASS,(Parameters)

IDENT TWØ

ENTRY START

START UJP **

.
. .
. .
. .

END

FINIS

} COMPASS Subprogram

7
9 LOAD,56

7
9 RUN,5

data deck

77 (End-of-File Card)
88

7
9 ENDSCOPE

77
88

STUDENT NOTES

SCOPE DEBUGGING AIDS

18.1 OCTAL CORRECTION CARDS

- 18.1.1 Location Symbols
- 18.1.2 Octal Corrections
- 18.1.3 Relocation Factors
- 18.1.4 Error Indicators

18.2 SNAP DUMPS

- 18.2.1 Errors
- 18.2.2 Location
- 18.2.3 First and Last Word Addresses
- 18.2.4 Mode
- 18.2.5 Identification
- 18.2.6 Note
- 18.2.7 Example
- 18.2.8 Rules for Using SNAP

18.3 OTHER DEBUGGING AIDS

- 18.3.1 Memory Map
- 18.3.2 Abnormal Termination Dump

18.4 COMPASS ERROR CODES

18.1 OCTAL CORRECTION CARDS

$\overline{7}$ $\overline{9}$ CC,location,octal correction,-----,octal correction.

Description: Octal corrections may be made to binary object subprograms after loading, using this instruction.

The parameters are free field.

If a period is used to terminate the card, comments may follow it.

The statement may be used to -

- (i) define corrections
- (ii) enter corrections
- (iii) enter additions to a subprogram by establishing a program extension area after the subprogram area.

18.1.1 Location Symbols

(i) (Program name) k

Corrections on this card are loaded beginning with relative address "k" in the named subprogram. The subprogram name must appear in parentheses.

Examples:

$\overline{7}$ $\overline{9}$ CC,(TEST)15,01000000

$\overline{7}$ $\overline{9}$ CC,(BUFFIN)107,20

Example of use:

		IDENT	TW $\overline{0}$
		ENTRY	START
0	START	UJP	**
		.	
		.	
10		ENA	20B
		.	
		.	
		END	

To change ENA 20B to ENA 30B

$\overline{7}$ $\overline{9}$ CC,(TW $\overline{0}$)10,14600030

(ii) Data Area Corrections

Dk

Corrections are loaded beginning with location "k" in the Data area.

Example: $\overline{7}$ $\overline{9}$ CC,D70,14000000

18.1.1 (cont.)

(iii) Program Extension Area

Xk

(a) First occurrence:

Defines a program extension area of length k words.
Corrections on this card are ignored.

(b) Subsequent occurrences:

Corrections are loaded beginning at location k
of the program extension area.

Example: $\begin{matrix} 7 \\ 9 \end{matrix} \emptyset \text{CC}, \text{X30}$

$\begin{matrix} 7 \\ 9 \end{matrix} \emptyset \text{CC}, \text{X3}, 14000010, 14000250$

(iv) Continuation Cards

+k

Increment k locations from the last location plus 1, corrected
by the previous $\emptyset \text{CC}$ card. k must be octal.

$\begin{matrix} 7 \\ 9 \end{matrix} \emptyset \text{CC}, (\text{TEST})136, 00000100$

$\begin{matrix} 7 \\ 9 \end{matrix} \emptyset \text{CC}, +5, 14000600$

Location 136 in Subprogram TEST, and location 144 in TEST
are changed.

18.1.2 Octal Corrections

- (i) Corrections may be of up to 8 digits, and are in the form of machine instructions.
- (ii) Each correction is separated from the preceding correction by a comma.
- (iii) Leading zeros may be omitted. Each value is stored right justified with zero fill, in successive computer words.
- (iv) Locations may be omitted for correction by using commas to indicate the omissions -

e.g. $\begin{matrix} 7 \\ 9 \end{matrix} \emptyset \text{CC}, (\text{TEST})10, 140, , 162, , 10014100$

Location 10 is altered to 140
11 is unchanged
12 is altered to 162
13 is unchanged
14 is altered to 10014100

18.1.3 Relocation Factors

- (i) none given - the quantity specified is absolute.
- (ii) (Subprogram name) - the address field of the correction is

18.1.3 (cont.)

- relative to the subprogram's first location.
- (iii) Data area - D
Relocate the word address portion of the Octal correction relative to the DATA area.
 - (iv) Common Area - C
Relocate the word address portion of the Octal correction relative to the ~~COMMON~~ area.
 - (v) Program Extension Area - X
Relocate relative to Prog. Ext. Area.
 - (vi) Last subprogram area referred to - *
Relocate relative to the last subprogram named in this or a preceding \emptyset CC or SNAP statement.

Example: $\begin{matrix} 7 \\ 9 \end{matrix} \emptyset CC, (TW\emptyset) 30, 20000040 (TW\emptyset)$

Correction made to location 30 in subprogram $TW\emptyset$, with the address part of the correction relocated by the factor by which $TW\emptyset$ itself is relocated.

(i.e., if subprogram $TW\emptyset$ itself is relocated by 1000, the actual correction loaded would be 2001040).

Note: The above example could also have been written:

$\begin{matrix} 7 \\ 9 \end{matrix} \emptyset CC, (TW\emptyset) 30, 20000040*$

18.1.4 Error Indicators

Errors in \emptyset CC cards prevent execution of the program.

If the extension area is incorrectly defined, the following message is printed on \emptyset UT:

***Xnnn

where nnn is the 3-digit octal length of the $SC\emptyset$ PE defined extension area.

The format for all other errors is:

*mn COL nn

where mn = error mnemonic

nn = column number on the card.

18.1.4 (cont.)

The following table sets out the error mnemonics used.

Mnemonic	Meaning
PN	Program name
BS	Common or data storage is undefined and referenced
AD	Address or location field begins with an illegal character
8F	Octal field contains a non-octal character
XA	Program extension area error
WR	Wrap around of location field address - exceeds core size
AN	Antecedent reference to a program or loading address
RL	Relocation factor error

18.2 SNAP DUMPS

SCOPE provides selective memory dumps during execution, using this statement.

⁷₉SNAP,(Parameters)

The statement must appear after the program is loaded and before the RUN card is encountered.

The dump is carried out by the library routine SNAPSHOT, which must be defined in the program as an external. (If it is not declared as external, the routine is not loaded into storage at load time. A diagnostic of this is given on the OUT listing as follows:

```
***      NØSD
        RUN  ABØRTED
```

The job is terminated.)

Snap statement parameters define

- (a) Where the dump is to be taken
i.e. when execution reaches a predetermined point, the dump is taken.
- (b) The area to be dumped.
- (c) The format of the dump.

The SNAP statement is of the following form:

⁷₉SNAP,location,beginning address,ending address,mode,identification,comments.

Each parameter is separated by commas.

Program names are always enclosed in parentheses.

18.2.1 Errors

If there is an error in any subfield, the following diagnostic is ren.

```
*mn,      COL      nn
           mn =     error mnemonic
           nn =     the card column number, in which the
                   error occurs.
```

The error mnemonics are set out in the table on the following page.

18.2.1 (cont.)

Mnemonic	Meaning
PN	Program name
BS	Common or data storage is undefined
AD	Address or location field begins with illegal character
8F	Octal field contains a non-octal character
XA	Program extension area is undefined or too small
WR	Location field address wrap around - exceeds core size
OV	Overflow of memory will recur if this SNAP is loaded
IM	Illegal mode
RG	Range to be snapped has FWA greater than LWA

When an error occurs, the SNAP statement is ignored, and execution continues as if no SNAP statement had been given.

18.2.2 Location

- (i) (Subprogram name) k
Replace location k in the subprogram with a RTJ to SNAP calling sequence.
- (ii) Program extension area - Xk
Replace location k in the P.EXT area by a RTJ to SNAP calling sequence.
- (iii) Data area - Dk
Replace statement k in the DATA area by a RTJ to the SNAP calling sequence.

18.2.3 First and Last Word Addresses

The ending address must always be greater than the beginning address, or the SNAP statement is ignored.

The address can be

- (i) Dk - dump begins or ends with word k in the DATA area.
- (ii) Ck - dump begins or ends with word k in the ~~COMMON~~ area.

18.2.3 (cont.)

- (iii) Xk - dump begins or ends with word k in the Program Extension Area,
- (iv) (Subprogram name) k - dump begins or ends with the location k in the specified subprogram.
- (v) *k - dump begins with the location k in the last named subprogram in a preceding \emptyset CC or SNAP statement.

18.2.4 Mode

The dump may be in one of 3 formats, and may include the Register file or not, as specified.

O = octal

C = 6-Bit Characters

F = Floating point

If the register file is to be included, R is used. Thus \emptyset R will give the dump printed out in \emptyset CTAL, plus the register file contents.

18.2.5 Identification

0 to 4 BCD characters will be printed out on the SNAP output to identify the dump. (Used if several dumps are to be made).

18.2.6 Note

If the location specified is in a loop, the contents of the area will be dumped out each time the location is encountered in the loop.

18.2.7 Example

Dump after the execution of the 5th instruction of subprogram SUB1 (assuming the first location of the subprogram is the entry point and is entered by a RTJ). Dump from location 441 in SUB1 to location 465 in SUB1 in OCTAL, with the Register File also, and identify the dump as \emptyset NE.

Answer: $\frac{7}{9}$ SNAP,(SUB1)6,*441,*465, \emptyset R, \emptyset NE

*441 could also be written (SUB1)441

18.2.8 Rules for Using SNAP

- (1) DON'T specify SNAP for an instruction using more than one word,

e.g. SRCE
SRCN
MOVE

- (2) DON'T SNAP jumps or tests, e.g., AQJ,EQ. - The jumps will not be executed correctly.
- (3) DON'T SNAP indirectly addressed instructions.
- (4) DON'T SNAP instructions which will be modified by program execution.
- (5) AVOID using SNAP instruction in a loop.
- (6) DON'T modify the location at which the SNAP occurs by an \emptyset CC statement.
- (7) DON'T specify SNAP for the following instructions:

MEQ
MTH
SSH
CPR
CON
SEL
EXS
INS
INTS
PAUS

- (8) DON'T specify SNAP for any SKIP Instruction
e.g. ISI, ASE, QSG, etc.
- (9) DON'T specify SNAP for INPUT/ \emptyset UTPUT instructions.

18.3 OTHER DEBUGGING AIDS

18.3.1 Memory Map

The programmer may secure a map of memory allocated to a loaded program at the time the run card is encountered. This map may be suppressed by a parameter of the RUN control statement (NM).

The map contains the following information:

- (1) Absolute address of the first location in each subprogram loaded.
- (2) All entry point symbols and their absolute addresses.
- (3) The absolute addresses of the first and last locations in the common area.
- (4) The absolute address of the first location in the data area.
- (5) The absolute address of the first location in the program extension area.

18.3.2 Abnormal Termination Dump

If a job is terminated abnormally, a post execution dump of all the non system part of memory is written on the standard output unit, unless the programmer has specified in the job card that it be suppressed (ND parameter).

The dump consists of the console conditions, the register file, and all the non-system part of memory.

Should the contents of words making up a line of print be exactly the same as both the last word on the preceding line, and the first word on the following line, the line is not printed, and the word "GAP" appears instead.

e.g. If all the locations between 10010 and 10017 are the same as both 10007 and 10020, the line will not be printed, and GAP will indicate the omission.

18.4 COMPASS ERROR CODES

- A Format error in address field.

- C Attempt to assemble information into ~~COMMON~~.
 (Instructions are processed as if a PRG was encountered.)

- D Doubly defined symbol.
 (The first time the symbol is used it is legal, and no flag is issued. Subsequent errors are flagged, and the instructions using the symbol are assembled as if no symbol occurred.)

- F Full symbol table.
 (All F flagged symbols are undefined, and reference to them in address fields of other instructions will produce U errors.)

- L Location field error.

- M Modifier error.

- O Operation code error. (The field is assembled as zeros.)

- U Undefined symbol.

- T Truncation error.
 (A symbol defined as a 17-bit character address is used in a subfield of only 15 bits. The 2 least significant bits are lost in truncation, and the flag indicates this loss.)

STUDENT NOTES

COMPASS ASSEMBLY OF CONSTANTS

- 19.1 OCTAL CONSTANT PSEUDO INSTRUCTIONS
- 19.2 DECIMAL CONSTANTS, FIXED POINT
- 19.3 DOUBLE PRECISION AND/OR FLOATING POINT CONSTANTS
- 19.4 BCD CONSTANTS
- 19.5 BCD CHARACTER CONSTANTS
- 19.6 VARIABLE FIELD CONSTANTS
 - 19.6.1 Introduction
 - 19.6.2 Octal Mode
 - 19.6.3 Hollerith Mode
 - 19.6.4 Word Address Arithmetic Mode
 - 19.6.5 Character Address Mode
 - 19.6.6 Example of VFD Instruction

19. COMPASS ASSEMBLY OF CONSTANTS

Constants may be

- (i) stated as octal, decimal or character in the source language.
- (ii) single, double or variable precision.
- (iii) fixed or floating point format.
- (iv) placed into bit positions of variable length fields.

19.1 OCTAL CONSTANT PSEUDO INSTRUCTIONS



Description: The instruction expresses constants as signed or unsigned octal constants. The octal integer may consist of 8 or less digits.

As many constants as can be contained on a card may be expressed in the address field, separated by commas. No blanks may appear between constants. The field terminates at the first blank or at Column 73.

The octal constants are assembled, right justified, in consecutive locations. The symbol in the location field is the 15 bit word address of the first constant in the field.

Example: CVTABLE $\emptyset CT$ -17,32,12345670,5742,-361

CVTABLE	7 7 7 7 7 7 6 0
	0 0 0 0 0 0 3 2
	1 2 3 4 5 6 7 0
	0 0 0 0 5 7 4 2
	7 7 7 7 7 4 1 6

Binary Scale factor:

An optional binary scale factor may be stated by suffixing the constants by B, and expressing the scale factor as a signed or unsigned decimal integer of not more than 2 digits. The magnitude of the constant after scaling must be less than 2^{24} . The scaling factor is used to save space in coding

e.g. CVTABLE $\emptyset CT$ 200000 could be written as
 CVTABLE $\emptyset CT$ 2B15

Examples: (i) CVTABLE $\emptyset CT$ 72B2

72 = 111 010 in binary
 = 111 010 00 scaled by 2 (shift binary point 2 places right)
 = 11 101 000 regrouped
 = 3 5 0

(CVTABLE) =

0000 0350

19.1 (cont.)

(ii) L ϕ C ϕ CT 36B3,4B12,270B-2

$$36 = 011\ 110_2$$

$$= 011\ 110\ 000\ (\text{scaled})$$

$$= 3\ 6\ 0_8$$

$$4 = 100_2$$

$$= 100\ 000\ 000\ 000\ 000_2\ (\text{scaled})$$

$$= 40000_8$$

$$270 = 010\ 111\ 000_2$$

$$= 010\ 111\ 0_2\ (\text{scaled} - 2)$$

$$= 56_8$$

L ϕ C	0000 0360
	0000 4000
	0000 0056

(iii) NUMBER ϕ CT 2416,311B16,3417B-8,-372

NUMBER	0000 2416
	6220 0000
	0000 0007
	7777 7405

Note: In negative scaling, digits are discarded from the right. If the number after scaling is greater than $2^{24}-1$, the field is set to zero and the A flag is set.

Example: ϕ CT 6666B20
Assembled as A = 00000000

19.2 DECIMAL CONSTANTS, FIXED POINT

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	DEC	d_1, d_2, \dots, d_n

Description: The instruction expresses constants as single precision fixed point binary constants. The constant may consist of a sign, and not more than 7 digits, with a magnitude of less than 2^{23} . The symbol in the location field is the address of the first constant in the field. The address field may contain as many constants separated by commas as the card may contain. The field terminates at first blank or Column 73.

Example: LØC DEC 1,82,-38

LØC	0 0 0 0 0 0 0 1	$(1_8 = 1_{10})$
	0 0 0 0 0 1 2 2	$(122_8 = 82_{10})$
	7 7 7 7 7 3 1	$(-46_8 = -38_{10})$

Scaling Factors:

Both decimal and/or binary scaling factors may be stated by suffixing the constant with D and/or B, and expressing the scale factor as signed or unsigned decimal integers. The magnitude of the constant after scaling must be less than 2^{23} .

Steps in the conversion:

- (i) Decimal integer is converted to binary.

The result must be less than 2^{23} .

Example: $36_{10} = 44_8$
 $= 100\ 100$

- (ii) Binary integer is multiplied or divided by 10^d , where d is the scaling factor.

The result must be less than 2^{23} .

Example: $36_{10} D2 = 36 \times 100$
 $= 44_8 \times 144_8$
 $= 100100 \times 001100100$
 $= 111000 \times 010\ 000$
 $= 7020_8$

- (iii) Shift the result the number of bits specified by the binary scaling factor.

Negative factor = RIGHT shift

Positive factor = LEFT shift

Example:

$36_{10} D2 B2$
 $36_{10} D2 = 111000010000$
 $= 11100001000000 = 34100_8$

19.3 DOUBLE PRECISION AND/OR FLOATING POINT CONSTANTS

LOCATION	OPERATION/MODIFIERS	ADDRESS FIELD
	DECD	d_1, d_2, \dots, d_n

Description: Decimal values may be stored as double precision fixed point constants, or as floating point constants. Either format requires 48 bits for storage (2 consecutive words). Up to 14 decimal digits may be specified, and the value of the expression must be less than 2^{47} .

Decimal and binary scaling factors may be used, as in the DEC instruction.

The signed 48-bit result is stored in two consecutive computer words.

Example: SYM.TAG DECD 32,64D2B4

SYM.TAG	00 00 00 00
	00 00 00 40
	00 00 00 00
	00 31 00 00

$$\begin{aligned}
 64_{10} &= 100_8 \\
 D^2 &= 10^2 = 100 = 144_8 \\
 100_8 \times 144_8 &= 1\ 44\ 00_8 \\
 &= 001100100000000 \\
 &= 11001000\ 000\ 000\ 000 \\
 &= 3\ 1\ 0\ 0\ 0\ 0
 \end{aligned}$$

Floating Point Constants:

1. Floating point constants contain a decimal point.

Examples: 300.246
 2.040117321
 .111
 1765122.1

2. They are stored in two consecutive 24 bit words as a 12 bit characteristic (exponent) and a 36-bit mantissa (coefficient).

WORD 1	BIASED EXP	COEFF
WORD 2	COEFF	

3. A floating point constant may contain not more than 14 decimal digits and a decimal point.
4. Binary scaling is not permitted, but decimal scaling is.
5. The result after scaling must not exceed the capacity of the hardware ($10^{\pm 308}$)

Example: SYM.TAG DECD 17643.463214

$$\begin{aligned}
 &17643.463214_{10} \\
 &= 42353.3551246_8 \\
 &= .423533551246_8 \times 2^{15}
 \end{aligned}$$

$$=$$

2017	4235
3355	1246

19.4 BCD CONSTANTS

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	BCD	n, C_1, C_2, \dots, C_n

Description: Characters are assembled for store into consecutive computer words as 6-bit BCD character codes, in addressable character positions. The code used is internal BCD.

The decimal integer n = the number of words to be used.

The maximum number of characters to be stored would be $4 \times n$.

The instruction reserves " n " words of storage, and any character positions not filled from the instruction are filled with blanks.

Characters specified in excess of $(4 \times n)$ are treated as comments.

The symbol in the location field is the 15-bit address of the first word.

Example: (i) ERRØRMSG BCD 3,I/Ø ERRØR

ERRØRMSG	I	/	Ø	60
	E	R	R	Ø
	R	60	60	60

3 words reserved, therefore the maximum number of characters is 12. Here only 9 appear before the end of message, so all are included.

(ii) ERRØRMSG BCD 2,I/Ø ERRØR

ERRØRMSG	I	/	Ø	60
	E	R	R	Ø

2 words reserved, therefore the maximum number of characters is 8. Here 9 appear, therefore the last one is treated as a comment and discarded.

19.5 BCD CHARACTER CONSTANTS

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	BCD,C	n, C ₁ , C ₂ , C _n

Description: Characters are assembled into consecutive character positions. n signifies the number of character positions to be reserved, and must be an integer less than 2^{15} . Characters in excess of n are treated as comments; positions reserved but not used are filled with blanks.

The symbol in the location field is the 17-bit address of the first character.

Storage

If the line of coding before BSS,C assigns character storage, the character string begins in the next available character position. If not, it begins with the first character position in the first available word.

If the number of characters specified does not exactly fill the last word, the rest of the word is zero filled. However, if the next instruction in the program which consumes space is a BCD,C instruction, it will fill up the remaining character positions in the last word.

Examples: (i) MSG BCD,C 9,TEST PRØG

MSG	T	E	S	T
	60	P	R	Ø
	G	O	O	O

(ii) MSG BCD,C 6,TEST PRØG

MSG	T	E	S	T
	60	P	O	O

(iii) MSG BCD,C 10,TEST PRØG
BCD,C 4,END

MSG	T	E	S	T
	60	P	R	Ø
	G	60	E	N
	D	60	O	O

(Note: 60 = blank read after PRØG above.)

(iv) Example of actual listing of assembled BCD and BCD,C constants:

<u>Assembled Constants</u>		<u>Instruction</u>
31 61 46 60	BCD	2,I/Ø ERRØR
25 51 51 46		
31 61 46 60	BCD	3,I/Ø ERRØR
25 51 51 46		
51 60 60 60		
63 25 62 63	BCD,C	9,TEST PRØG
60 47 51 46		
27		
63 25 62	BCD,C	6,TEST PRØG
63 60 47		
63	BCD,C	10,TEST PRØG
25 62 63 60		
47 51 46 27		
60		
25 45 24	BCD,C	4,END
60 00 00 00		

19.6 VARIABLE FIELD CONSTANTS

19.6.1 Introduction

The general form of the instruction is:

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	VFD	$Mn/v, \dots, Mn/v$

where M = mode indicator

n = positive decimal integer denoting the number of bit positions in the variable field specified by this subfield. (NOTE: the range of values of n varies with the mode M).

v = the content of the field. This varies according to the mode and is restricted by the declared length.

As many address subfields as may be contained on a single card are allowed. Each subfield is terminated by a comma, except the last, which is terminated by a blank.

Use:

The instruction is used to enter information in one of the following modes into a field of a designated bit-length.

- (i) octal numbers
 - (ii) character codes (BCD)
 - (iii) relocatable addresses (either word or character addresses)
- or (iv) constants.

It enables information to be packed into computer words during assembly time.

Packing:

- (i) Values are entered right adjusted in the field, with sign extension.
- (ii) Character strings are entered left adjusted, with blank fill.

e.g. A B X is a character string.
If entered in a 24-bit character
VFD, it will be entered as:

21 22 69 60

Relocation:

If relocatable addresses are entered into a field, the addresses will be relocated when the assembled subprogram is loaded. (The listing will show only the assembled fields.)

19.6.2 Octal Mode

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	VFD	$\emptyset n/v$

where \emptyset indicates the octal mode.

n indicates the number of bit positions in the field.
It may be in the range 1-24 inclusive.

19.6.3 (cont.)

If the field is longer than the number of characters specified, the work is blank filled. Characters are stored LEFT justified in the field.

Example: LØC VFD H18/AB =

21	22	60	00
----	----	----	----

19.6.4 Word Address Arithmetic Mode

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
VFD	A _n /v	

The address field v may be:

- (i) A constant
- (ii) A symbol
- (iii) An expression formed by the rules of address field arithmetic.

If the expression yields a relocatable word address, the programmer must arrange so that it will be right justified on bit 0 in a word, and be contained in 15 bits. If it is not relocatable, the full 24 bits can be used.

Examples: (i) ALØC VFD A24/A+3

If A is a relocatable word address, previously defined, then answer must be 15 bits stored Right justified in word on bit 0.

If A is an absolute value, 24 bits can be stored.

- (ii) Using relocatable word address

VFD Ø9/010,A15/BUFR

where BUFR is a symbol, address 00002 in the program.

Assembled as 01000002

- (iii) Using non relocatable symbol

A EQU 77777B

B EQU 57B

C EQU 20B

VFD A21/A-B+C

Assembled as 777740 in the designated 21 bits.

19.6.5 Character Address Mode

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
VFD	C _n /v	

A minimum of 17 bits is required for character addressing.

n must therefore not be less than 17.

19.6.5 (cont.)

If the address is relocatable, the field must be right justified on bit 0 in the word.

Examples: (i) VFD C24/237B
Assembled as 00000237

VFD C24/237
Assembled as 00000355

(ii) Suppose NAME is assembled at location 00025 in the subprogram, and is a word address.

VFD C24/NAME
Assembled as 00000124

(Note: 124 is the character address of word address 25)

(iii) CADR EQU,C 237B
ØPAD VFD Ø7/04,C17/CADR

CADR: is assembled as a 17-bit character address
= 000237
= Word 47, character position 3
= 000473 on the listing.

ØPAD: is assembled as a 24-bit word
= 02000237

04 is entered right justified in a 7-bit field

= 0000100
= 000 010 0

CADR is the 17-bit character address
000237

= 00 000 000 010 011 111

(iv) Suppose TMPC is assembled as character 0 in word 20 in the subprogram

JAC VFD Ø7/4,C17/TMPC

Assembled as 02000100

(100 is character address of TMPC)

19.6.6 Example of VFD instruction

If NAME is located at address 00011 in the assembled program, what will be assembled as a result of:

A EQU 77777B

X EQU 57B

B EQU 20B

VFD Ø12/-737,A21/A-X+B,H24/HA3,A15/NAME+2,H12/BQ

Answer:

70	40	77	77
74	03	02	10
36	00	00	13
22	50	00	00

INPUT/OUTPUT WITHOUT CIO

- 20.1 INPUT/OUTPUT CHARACTERISTICS
 - 20.1.1 Introduction
 - 20.1.2 Interface Signals
 - 20.1.3 System Configuration
 - 20.1.4 Logical Sequence of Events for Initiating
INPUT/OUTPUT Operations
- 20.2 CONNECT
- 20.3 SELECT
- 20.4 WORD ADDRESSED INPUT TO STORAGE
- 20.5 WORD ADDRESSED OUTPUT FROM STORAGE
- 20.6 CHARACTER ADDRESSED INPUT TO STORAGE
- 20.7 CHARACTER ADDRESSED OUTPUT FROM STORAGE
- 20.8 INPUT/OUTPUT TO AND FROM THE A REGISTER
 - 20.8.1 Input Character to A
 - 20.8.2 Input Word to A
 - 20.8.3 Output Character from A
 - 20.8.4 Output Word from A
- 20.9 SENSING INSTRUCTIONS
 - 20.9.1 Sense External Status
 - 20.9.2 Copy External Status and Interrupt Mask Register
 - 20.9.3 Sense Interrupt
 - 20.9.4 Sense Internal Status
 - 20.9.5 Copy Internal Status and Interrupt Mask Register
 - 20.9.6 Comments on Internal Status and the Interrupt Mask Register
- 20.10 CONSOLE TYPEWRITER INPUT/OUTPUT
 - 20.10.1 General Description
 - 20.10.2 Operation
 - 20.10.2.1 Set Tabs, Margins and Spacing
 - 20.10.2.2 Clear
 - 20.10.2.3 Status Checking
 - 20.10.2.4 Type In and Type Load
 - 20.10.2.5 Type Out and Type Dump
 - 20.10.3 Typewriter Console Switches and Indicators
 - 20.10.4 Character Codes
 - 20.10.5 Input/Output Instructions
 - 20.10.5.1 Set Console Typewriter Input
 - 20.10.5.2 Set Console Typewriter Output
 - 20.10.5.3 Pause Instruction
(as used with console typewriter instructions)

20.1 INPUT/OUTPUT CHARACTERISTICS

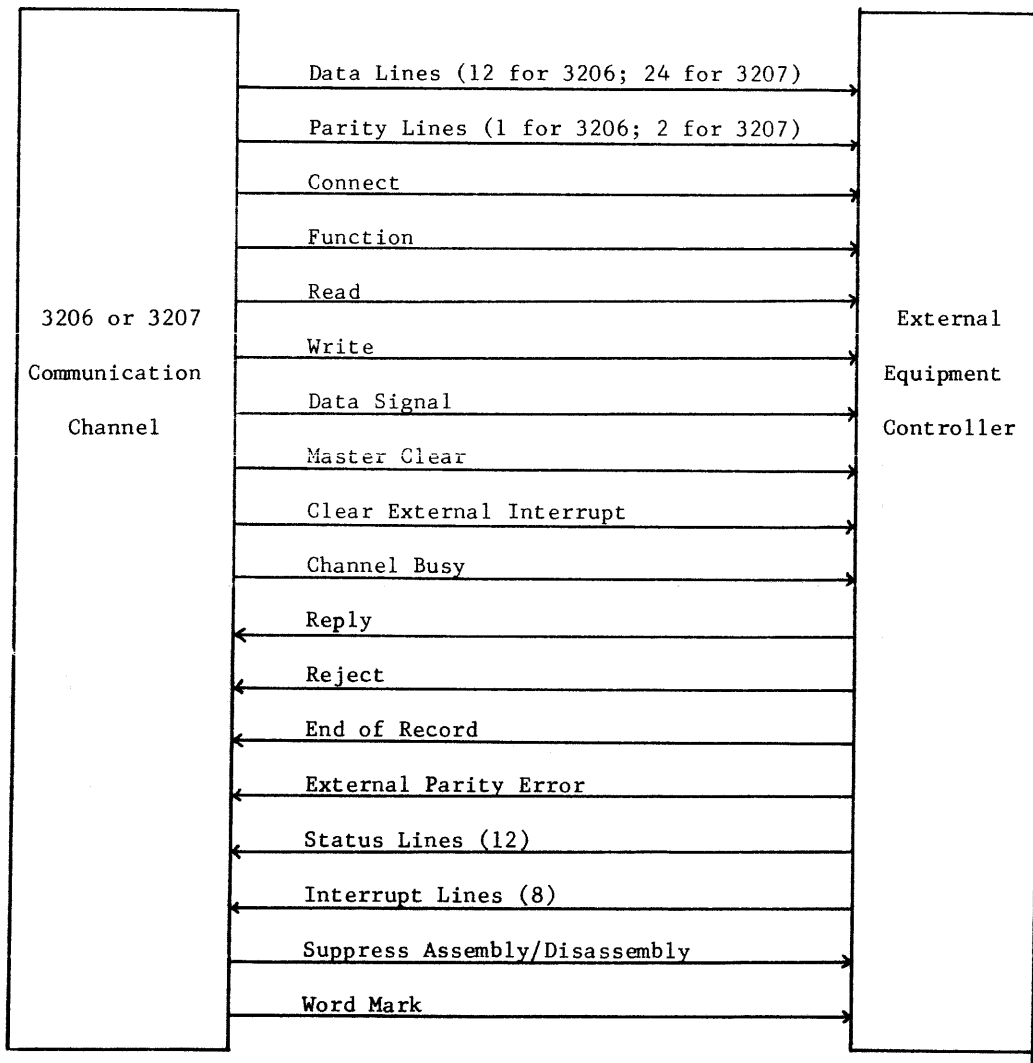
20.1.1 Introduction

The Input/Output section of the computer is responsible for transferring data to and from the computer and to and from an external device. Data is transferred between a 3200 Computer and its associated external equipment via a 3206 or 3207 Communication Channel. For programming purposes, the eight possible 3206 channels in a system are designated by numbers 0 through 7. A 3207 replaces the 3206 type I/O channels 2 and 3 in expanded systems. It is programmed as channel 2.

20.1.2 Interface signals

Up to eight external equipment controllers may be attached in parallel to each 3206 Communication Channel. The following chart shows the principal signals which flow between a 3206 and its external equipment. The 12 status lines are active only between the channel and the controller to which it has been connected by the CON (77.0) instruction*.

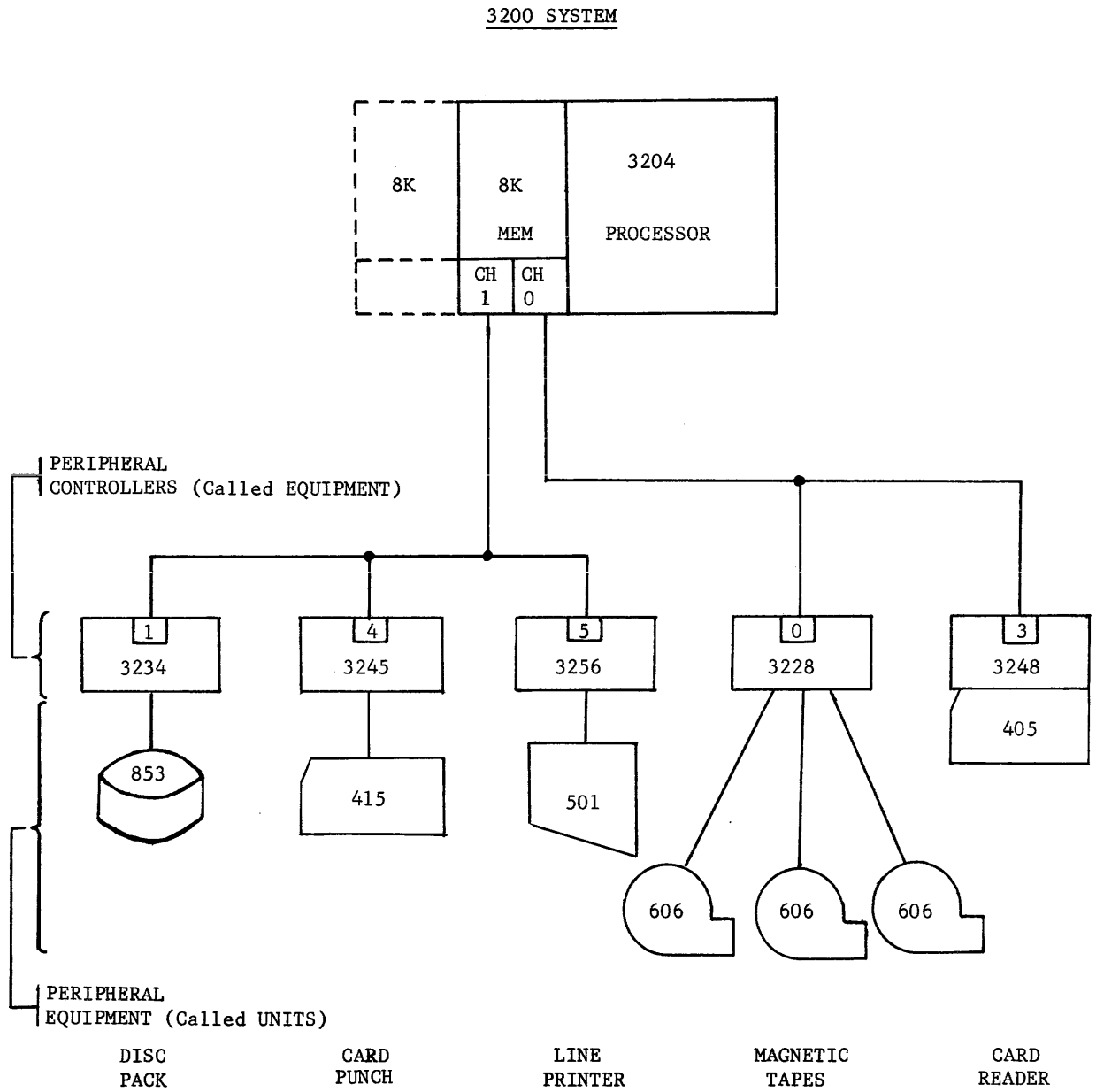
The eight interrupt lines, designated 0-7, connect to all eight controllers attached to a channel. These lines match the Equipment Selector switch setting on each controller. For a complete description of the I/O interface signals as well as an I/O timing chart, refer to the 3000 Series I/O Specification, publication number 60048800.



* The connect instruction selects one of eight controllers which may be attached to the channel.

20.1.3 System configuration

A typical configuration is:



20.1.4 Logical Sequence of Events for Initiating INPUT/OUTPUT Operations

Execute a connect instruction which will indicate the channel, controller and unit which is to receive or transmit the data.

Sense the status of the controller to determine if the unit is available and capable of performing the function and/or data transmission ("PRE-STATUS").

Send the appropriate select functions. Select functions are used to format the INPUT/OUTPUT device.

Execute the data transmission instructions.

At end of operation, sense the status of the controller to determine if the function and/or data transmission was successful ("POST-STATUS" check).

20.2 CONNECT

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	CØN	x, ch

23	18 17	15 14	12 11	00
77	0	ch	x	

ch = I/O channel designator, 0-7
 x = 12 bit connect code. Bits 09-11 select one of eight controllers which may be attached to channel ch. Bits 00-08 select the peripheral units connected to the controller.

Description: This instruction sends a 12-bit connect code along with a connect enable to an external equipment controller on I/O channel 'ch'. If a Reply is received from the controller within 100 usec, the next instruction is read from address P + 2. If a Reject is received or there is no response within 100 usec, a reject instruction is read from address P + 1. If the I/O channel is busy, a reject instruction is read immediately from address P + 1.

Once the connect is successful (RNI at P + 2) it will remain in effect until another connect is attempted on the same channel or a clear function is performed.

Examples:

CØN 1001B,2

This instruction connects controller number 1 and unit 1 on channel 2.

The reject instruction coded at P + 1 is usually a jump back to the connect to cause the computer to wait until the connect is successful.

e.g. CØN 1001B,2
 UJP *-1

20.2 (cont.)

In the following example, which controllers and units will be connected on channel 1 and 2 when the computer stops. Assume all connects can be made.

```

CØN      2004B,1
UJP      *-1
CØN      3004B,2
UJP      *-1
CØN      5B,1
UJP      *-1
UCS
  
```

Answers:

```

Channel 1 = Equip 0, Unit 5
Channel 2 = Equip 3, Unit 4
  
```

20.3 SELECT

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	SEL	x, ch

23	18 17	15 14	12 11	00
77	1	ch	x	

ch = I/O channel designator, 0-7
 x = 12-bit function code. Each piece of external equipment has a unique set of function codes to specify operations within that device. Refer to the 3000 Series Computer Systems Peripheral Equipment Codes publication No. 60113400 for a complete list of function codes.

Description: This instruction sends a 12-bit function code along with a function enable to the unit connected to I/O channel 'ch'. If a Reply is received from the unit within 100 usec, the next instruction is read from P + 2. If a Reject is received or there is no response within 100 usec, a reject instruction is read from address P + 1. If the I/O channel is busy, a reject instruction is read immediately from address P + 1.

The following conditions or combination of conditions will result in a Reject:

- 1) No Unit or Equipment Connected: The referenced device is not connected to the system and cannot recognize a Function instruction. If no response is received within 100 usec, the Reject signal is generated automatically by the I/O channel.
- 2) Undefined Code: When the Function code x is not defined for the specific device, a Reject may be generated by the device. However, in some cases an undefined code will cause the device to generate a Reply although no operation is performed. (Refer to the reference manual for the specific device.)
- 3) Equipment or Unit Busy or Not Ready: The device cannot perform the operation specified by the function code x without damaging the equipment or losing data. For example, a Write End of File code is rejected by a tape unit if the tape unit is rewinding.
- 4) Channel Busy: The selected data channel is currently performing a Read or Write operation.

20.3 (cont.)

The function codes for magnetic tape and card reader are:

MAGNETIC TAPE CONTROLLERS FUNCTION CODES	3248 CARD READER CONTROLLER FUNCTION CODES
0000 Release	0001 Negate Hollerith to Internal BCD Conversion
0001 Binary	0002 Release Negate Hollerith to Internal BCD Conversion
0002 Coded	0004 Set Gate Card
0003 556 BPI	0005 Clear
0004 200 BPI	0020 Interrupt on Ready and Busy
0005 Clear	0021 Release Interrupt on Ready and Busy
0006 800 BPI	0022 Interrupt on End of Operation
0010 Rewind	0023 Release Interrupt on End of Operation
0011 Rewind Unload	0024 Interrupt on Abnormal End of Operation
0012 Backspace	0025 Release Interrupt on Abnormal End of Operation
0013 Search Forward to File Mark	
0014 Search Backward to File Mark	
0015 Write File Mark	
0016 Skip Bad Spot	
0020 Interrupt on Ready and Busy	
0021 Release Interrupt on Ready and Busy	
0022 Interrupt on End of Operation	
0023 Release Interrupt on End of Operation	
0024 Interrupt on Abnormal End of Operation	
0025 Release Interrupt on Abnormal End of Operation	
0040 Clear Reverse Read	
0041 Set Reverse Read	

Example:

```

CØN      2B,1      Connects Controller 0 Unit 2 on channel 1
UJP      REJX
SEL      10B,1     Rewinds Tape Unit 2 to Load Point
UJP      REJX
SEL      3B,1     Sets the Density to 556 bpi
UJP      REJX
SEL      1B,1     Sets the Mode to binary
UJP      REJX
.
.
.
UCS
  
```

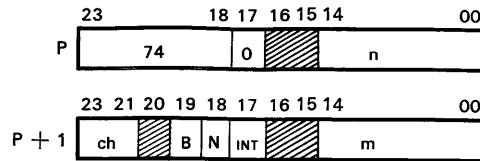
Exercise: What does the following group of select function codes for Tape Unit 6 accomplish?

```

CØN      6B,0
UJP      REJX
SEL      6B,0
UJP      REJX
SEL      13B,0
UJP      REJX
.
.
.
UCS
  
```

20.4 WORD ADDRESSED INPUT TO STORAGE

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	INPW, INT, B, N	ch, m, n



- B = "1" for backward storage
- ch = I/O channel designator, 0-7
- INT = "1" for interrupt upon completion
- N = "0" for 12- to 24-bit assembly
= "1" for no assembly
- m = first word address of I/O data block;
becomes current address as I/O operation progresses
- n = last word address of input data block,
plus one (minus one, for backward storage)

Description: This instruction transfers a word-addressed data block from an external equipment to storage. Transferring 12-bit bytes or 24-bit words depends upon the type of I/O channel used. The 3206 utilizes 12-bit bytes and the 3207 uses 24-bit words.

During forward storage and 12- to 24-bit assembly, the first byte of a block of data is stored in the upper half of the memory location specified by the storage address. Conversely, during backward storage, the first byte is stored in the lower half of the memory location.

This instruction is an initiate type instruction, which means all the foregoing actions will occur only if the instruction executes normally. And RNI will be at P + 3. However, if the input/output control for the specified channel in Block Control is busy, the instruction will act as a double NOP and RNI from P + 2. The instruction at that location is termed the 'Reject Instruction'.

Examples:

(i)	C/N	1001B,1
	RTJ	REJX
	SEL	2B,1
	RTJ	REJX
	SEL	3B,1
	RTJ	REJX
	INPW	1, INBUFF, INBUFF+50
	RTJ	REJX

This example initiates an Input from tape unit 1, controller 1 on channel 1. The tape is written in BCD mode at 556 bpi.

20.4 (cont.2)

Answers:

(1)

INBUFF 21222324
25262730
31414121
22232425
26273031
41422122
23242526
27303141
42212223
24252627
30314142
21222324
25262730
31414221
22232425
26273031
41422122
23242526
27303141
42212223

(2)

INBUFF 22334221
31412730
25262324
21224142
30312627
24252223
42213141
27302526
23242122
41423031
26272425
22234221
31412730
25262324
21224142
30312627
24252223
42213141
27302526
23242122

(3)

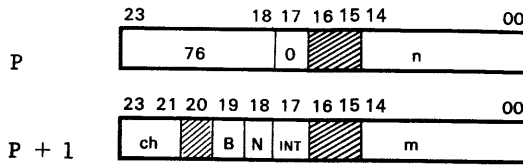
INBUFF 77772122
77772324
77772526
77772730
77773141
77774221
77772223
77772425
77772627
77773031
77774142
77772122
77772324
77772526
77772730
77773141
77774221
77772223
77772425
77772627
77773031
77774142
77772122
77772324
77772526
77772730
77773141
77774221
77772223
77772425
77772627
77773031
77774342
77772122
77772324
77772526
77772730
77773141
77774221
77772223

(4)

INBUFF 77772223
77774221
77773741
77772730
77772526
77772324
77772122
77774142
77773031
77772627
77772425
77772223
77774221
77773141
77772730
77772526
77772324
77772122
77774142
77773031
77772627
77772425
77772223
77774221
77773141
77772730
77772526
77772324
77772122
77774142
77773031
77772627
77772425
77772223
77774221
77773141
77772730
77772526
77772324
77772122

20.5 WORD ADDRESSED OUTPUT FROM STORAGE

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	\emptyset UTW, INT, B, N	ch, m, n



- B = "1" for backward storage
 ch = I/O channel designator
 INT = "1" for interrupt upon completion
 N = "0" for 24- to 12 bit disassembly
 = "1" for straight 24-bit (no disassembly)
 data transfer
 m = first word address of I/O data block;
 ecomes current address as I/O
 operation progresses
 n = last word address of output data
 block, plus one (minus one, for
 backward output)

Description: This instruction transfers a word-addressed block of data consisting of 12-bit bytes or 24-bit words, from storage to an external equipment.

With no disassembly, 12 or 24-bit transfer capability depends upon whether a 3206 or 3207 I/O channel is used. If an attempt is made to send a 24-bit word over a 3206 I/O channel, the upper byte will be lost.

This instruction is an initiate type instruction, which means all the foregoing actions will occur only if the instruction executes normally. And RNI will be at P + 3. However, if the input/output control for the specified channel in Block Control is busy, the instruction will act as a double NOP and RNI from P + 2. The instruction at that location is termed the 'Reject Instruction'.

Examples: A block of memory is as shown:

	Numerical Value	Alphabetical Representation
\emptyset UTBUFF	21222324	A B C D
	25262730	E F G H
	31414221	I J K A
	22232425	B C D E
	26273031	F G H I
	41422122	J K A B
	23242526	C D E F
	27303141	G H I J
	42212223	K A B C
	24252627	D E F G
	30314142	H I J K
	21222324	A B C D
	25262730	E F G H
	31414221	I J K A
	22232425	B C D E
	26273031	F G H I
	41422122	J K A B
	23242526	C D E F
	27303141	G H I J
	42212223	K A B C

20.5 (cont.)

What will be printed on the line printer as a result of the following coding examples?

- (1) CØN 5000B,1
 UJP REJX
 ØUTW 1,ØUTBUFF,ØUTBUFF+20
 UJP REJX

- (2) CØN 5000B,1
 UJP REJX
 ØUTW,B 1,ØUTBUFF+19,ØUTBUFF-1
 UJP REJX

- (3) CØN 5000B,1
 UJP REJX
 ØUTW,N 1,ØUTBUFF,ØUTBUFF+20
 UJP REJX

- (4) CØN 5000B,1
 UJP REJX
 ØUTW,B,N 1,ØUTBUFF+19,ØUTBUFF-1
 UJP REJX

Answers:

- (1)
ABCDEF GHIJK ABCDEF GHIJK ABCDEF GHIJK ABCDEF GHIJK ABCDEF GHIJK ABCDEF GHIJK ABC

- (2)
BCKAIJGHEFC DABJKHIFGDEBCKAIJGHEFC DABJKHIFGDEBCKAIJGHEFC DABJKHIFGDEBCKAIJGHEFC DAB

- (3)
CDGHK ADEHIABEFIJACFGJKCDGHK ADEHIABEFIJBC

- (4)
BCIJEFABHIDEKAGHCDJKFGBCIJEFABHIDEKAGHCD

20.6 (cont.)

What will be the contents of memory as a result of the following coding examples?

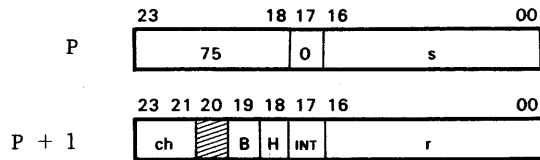
- | | | |
|-----|----------|----------------------|
| (1) | CØN | 3000B,0 |
| | UJP | REJX |
| | INPC | 0,INBUFF,INBUFF+80 |
| | UJP | REJX |
| (2) | CØN | 3000B,0 |
| | UJP | REJX |
| | INPC,B | 0,INBUFF+79,INBUFF-1 |
| | UJP | REJX |
| (3) | CØN | 3000B,0 |
| | UJP | REJX |
| | INPC,H | 0,INBUFF,INBUFF+80 |
| | UJP | REJX |
| (4) | CØN | 3000B,0 |
| | UJP | REJX |
| | INPC,B,H | 0,INBUFF+79,INBUFF-1 |
| | UJP | REJX |

Answers:

- | | |
|------------|------------|
| (1) | (2) |
| 21222324 | 23222142 |
| 25262730 | 41313027 |
| 31414221 | 26252423 |
| 22232425 | 22214241 |
| 26273031 | 31302726 |
| 41422122 | 25242322 |
| 23242526 | 21424131 |
| 27303141 | 30272625 |
| 42212223 | 24232221 |
| 24252627 | 42413130 |
| 30314142 | 27262524 |
| 21222324 | 23222142 |
| 25262730 | 41313027 |
| 31414221 | 26252423 |
| 22232425 | 22214241 |
| 26273031 | 31302726 |
| 41422122 | 25242322 |
| 23242526 | 21424131 |
| 27303141 | 30272625 |
| 42212223 | 24232221 |
| (3) | (4) |
| Same as #1 | Same as #2 |

20.7 CHARACTER ADDRESSED OUTPUT FROM STORAGE

LOCATION: Φ UTC, INT, B, H ch, r, s



- B = "1" for backward storage
- ch = I/O channel designator, 0-3
- H = "0" for 24- to 6-bit disassembly
= "1" for 24- to 12-bit disassembly
- INT = "1" for interrupt upon completion
- r = first character address of I/O data block; becomes current address as I/O operation progresses
- s = last character address of output data block, plus one (minus one, for backward output)

Description: This instruction transfers a character-addressed block of data, consisting of 6-bit characters or 12-bit bytes, from storage to an external equipment.

This instruction is an initiate type instruction, which means all the foregoing actions will occur only if the instruction executes normally. And RNI will be at P + 3. However, if the input/output control for the specified channel in Block Control is busy, the instruction will act as a double NOP and RNI from P + 2. The instruction at that location is termed the 'Reject Instruction'.

Examples: A block of memory is as shown:

	Numerical Value	Alphabetical Representation
Φ OUTBUFF	21222324	A B C D
	25262730	E F G H
	31414221	I J K A
	22232425	B C D E
	26273031	F G H I
	41422122	J K A B
	23242526	C D E F
	27303141	G H I J
	42212223	K A B C
	24252627	D E F G
	30314142	H I J K
	21222324	A B C D
	25262730	E F G H
	31414221	I J K A
	22232425	B C D E
	26273031	F G H I
	41422122	J K A B
	23242526	C D E F
	27303141	G H I J
	42212223	K A B C

20.7 (cont.)

What will be printed on the line printer as a result of the following coding examples?

- (1) CØN 5000B,1
 UJP REJX
 ØUTC 1,ØUTBUFF,ØUTBUFF+80
 UJP REJX
- (2) CØN 5000B,1
 UJP REJX
 ØUTC,B 1,ØUTBUFF+79,ØUTBUFF-1
 UJP REJX
- (3) CØN 5000B,1
 UJP REJX
 ØUTC,H 1,ØUTBUFF,ØUTBUFF+80
 UJP REJX
- (4) CØN 5000B,1
 UJP REJX
 ØUTC,B,H 1,ØUTBUFF+79,ØUTBUFF-1
 UJP REJX

Answers:

(1)

ABCDEFGHIJKABCDEFGHIJKABCDEFGHIJKABCDEFGHIJKABCDEFGHIJKABCDEFGHIJKABCDEFGHIJKABC

(2)

CBAKJIHGFEDCBAKJIHGFEDCBAKJIHGFEDCBAKJIHGFEDCBAKJIHGFEDCBAKJIHGFEDCBAKJIHGFEDCBA

(3)

Sames as Number 1

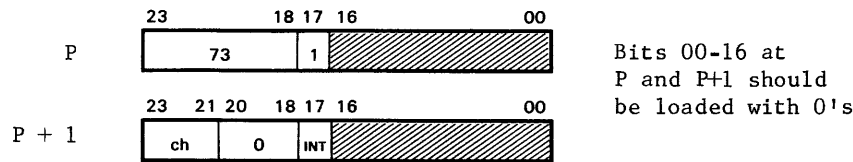
(4)

Same as Number 2

20.8 INPUT/OUTPUT TO AND FROM THE A REGISTER

20.8.1 Input Character to A

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	INAC, INT	ch



ch = I/O channel designator, 0-7
 INT = "1" for interrupt upon completion
 "0" for do not interrupt upon completion

Description: This instruction transfers a 6-bit character from an external equipment into the lower 6-bits of the A register. A is cleared prior to loading and the upper 18-bits remain cleared.

Main control is stalled until the character is received from the external equipment. At that time it resumes reading instructions and RNI's from P + 3. If the input/output control for the specified channel in Block Control is busy, the instruction will act as a double NOP and RNI from P + 2. The instruction at that location is termed the 'Reject Instruction'.

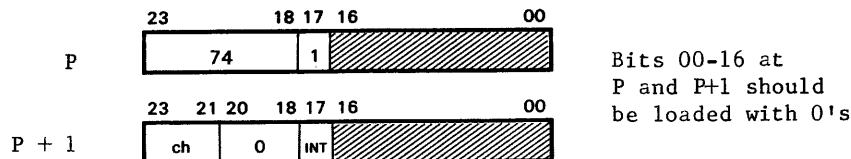
Example:

CØN	3000B,1
UJP	*-1
INAC	1
UJP	*-2

This example inputs 6-bits from controller 3, unit 0 on channel 1 into the lower 6-bits of the A register. A is cleared prior to loading and the upper 18-bits remain cleared.

20.8.2 Input Word to A

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	INAW, INT	ch



ch = I/O channel designator, 0-7
 INT = "1" for interrupt upon completion
 "0" for do not interrupt upon completion

Description: This instruction transfers a 12-bit byte into the lower 12-bits of A or a 24-bit word into all of A from an external equipment. Transferring 12 or 24 bits depends upon whether a 3206 or 3207 I/O channel is used. (A) is cleared prior to loading and, in the case of a 12-bit input, the upper 12 bits remain cleared.

20.8.2 (cont.)

Main control is stalled until the byte (word) is received from the external equipment. At that time it resumes reading instructions and RNI's from P + 3. If the input/output control for the specified channel in Block Control is busy, the instruction will act as a double NOP and RNI from P + 2. The instruction at that location is termed the 'Reject Instruction'.

Note: Bits 18, 19 and 20 are all zeros when a 3206 data channel is used. If the operation with A involves the use of a 3207, these bits take on the following significance:

- Bit 20 = always a "0"
- Bit 19 = If bit 18 = "1", the state of bit 19 is of no consequence.
If bit 18 = "0", a "1" in bit 19 signifies backward operation. A "0" in bit 19 signifies a forward operation.

Examples: If the connect code for the card reader is 3000B and a data card has ABCD in the first four columns, what will be the contents of the A register as a result of the following coding examples? Assume a 3206 data channel.

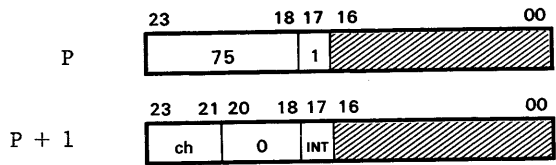
```

CØN      3000B,0
UJP      REJX
INAW     0
UJP      REJX
    
```

Answer: (A) = 00002122 Also, this is all the information that is available from that card. Another input to A would cause the next card to be read.

20.8.3 Output Character from A

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	<i>ØTAC, INT ch</i>	



Bits 00-16 at P and P+1 should be loaded with 0's

ch = I/O channel designator, 0-7
 INT = "1" for interrupt upon completion
 "0" for do not interrupt upon completion

Description: This instruction transfers a character from the lower 6-bits of A to an external equipment. The original contents of A are retained.

After outputting the character, main control proceeds immediately and RNI's at P + 3. Should the input/output control for the specified channel in Block Control be busy, the instruction will act as a double NOP and RNI from P + 2. The instruction at that location is termed the 'Reject Instruction'.

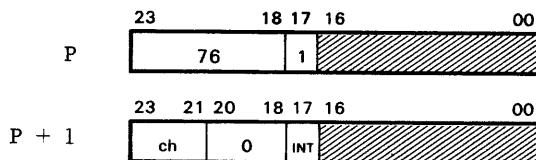
20.8.3 (cont.)

Example: If the connect code for the external device is 3002B, it is connected to channel 4, and it requires a data output of 21B to perform a certain task, what coding is necessary to perform the task.

Ans: CØN 3002B,4
 UJP *-1
 ENA 21B
 ØTAC 4
 UJP *-2

The data sent to the external device is 0021B accompanied by a suppress assembly/disassembly control signal. This latter informs the device to ignore the upper 6 bits and recognize the 21B only.

20.8.4 Output Word from A



Bits 00-16 at
 P and P+1 should
 be loaded with 0's

ch = I/O channel designator, 0-7
 INT = "1" for interrupt upon completion
 "0" for do not interrupt upon completion

Description: This instruction transfers a 12-bit byte from the lower 12 bits of A (or all 24-bits of A) to an external equipment depending upon the type of I/O channel (3206 or 3207) that is used. The contents of A is not disturbed and is retained.

After outputting the information, main control proceeds immediately and RNI's at P + 3. Should the input/output control for the specified channel in Block Control be busy, the instruction will act as a double NOP and RNI from P + 2. The instruction at that location is termed the 'Reject Instruction'.

Note: Bits 18, 19 and 20 are all zeros when a 3206 data channel is used. If the operation with A involves the use of a 3207, these bits take on the following significance:

Bit 20 = always a "0"
 Bit 19 = If bit 18 = "1", the state of bit 19 is of no consequence.
 If bit 18 = "0", a "1" in bit 19 signifies backward operation. A "0" in bit 19 signifies a forward operation.

Example: Assuming a remote typewriter connected to a data channel (not to be confused with the console typewriter) has a connect code of 5000B and is connected to channel 2, what code is necessary to cause that typewriter to perform a carriage return tab sequence of operations? The code for a carriage return is 77B and for the tab is 75B.

20.8.4 (cont.)

Answer: CØN 5000B,2
 UJP *-1
 ENA 7775B
 ØTAW 2
 UJP *-2

The data transmitted to the typewriter is 7775B.

20.9 SENSING INSTRUCTIONS

20.9.1 Sense External Status

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD
	EXS	x, ch

23	18	17	15	14	12	11	00
77		2		ch			x

ch = I/O channel designator, 0-7
 x = external status sensing mask code

Description: When a peripheral equipment controller is connected to an I/O channel by the CØN (77.0) instruction, the EXS instruction can sense conditions within that controller. Twelve status lines run between each controller and its I/O channel. Each line may monitor one condition within the controller, and each controller has a unique set of line definitions. To sense a specific condition, a "1" is placed in the bit position of the status sensing mask that corresponds to the line number. When this instruction is recognized in a program, RNI at address P + 1 if an external status line is active when its corresponding mask bits are "1". RNI at address P + 2 if no selected line is active.

Status Codes: Following are the status response codes for magnetic tape and card reader:

MAGNETIC TAPE STATUS CODES		3248 CARD READER STATUS CODES
XXX1 Ready		XXX1 Ready
XXX2 Channel and/or Read/Write Control and/or Unit Busy		XXX2 Busy
XXX4 Write Enabled		XXX4 Binary Card
XX1X File Mark Read		XX1X File Card Read
XX2X At Loadpoint		XX2X Fail to Feed, Stacker Full or Jam
XX4X End-of-Tape Read		XX4X Input Tray Empty
X1XX Density 2 ⁰ bit**		X1XX Input Tray Empty and End of File Switch On
X2XX 2 ¹ bit**		X2XX Ready and not Busy
** 00 = 200 bpi		Interrupt Present
01 = 556 bpi		X4XX End of Operation
10 = 800 bpi		Interrupt Present
11 = undefined		1XXX Abnormal End of Operation Interrupt Present
X4XX Lost Data		2XXX Read Compare, Preread Error or Illegal Suppress Assembly
1XXX End of Operation		
2XXX Vertical or Longitudinal Parity Error		
4XXX Reserved by another control (multiple channel controllers only)		

20.9.1 (cont.)

Comments: Refer to the 3000 Series Computer Systems Peripheral Equipment Codes manual, publication no. 60113400 for a complete list of status response codes.

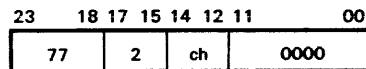
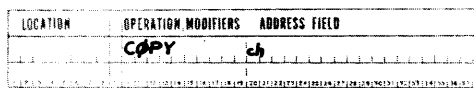
Example: The example below accomplishes the following:

- a) Connects the card reader on channel 0.
- b) Checks the status of the card reader.
If busy it waits until it becomes not busy.
- c) Initiates reading of a card at the card reader and transfer of the data read to memory.
- d) Waits until the INPUT is complete.

```

CØN      3000B,0      CONNECT CARD READER
UJP      *-1
EXS      2B,0         SENSE BUSY
UJP      *-1         LOOP IF BUSY'
INPW     0,INBUFF,INBUFF+20  INITIATE INPUT
UJP      *-2
EXS      2B,0         SENSE BUSY
UJP      *-1         LOOP UNTIL DONE
.
.
.
UCS
    
```

20.9.2 Copy External Status and Interrupt Mask Register



ch = I/O channel designator, 0-7

Description: This instruction performs the following functions:

- (1) The external status code from I/O channel ch is loaded into the lower 12 bits of A. See EXS instruction.
- (2) The contents of the Interrupt Mask register are loaded into the upper 12 bits of A.
- (3) RNI from address P + 1.

Examples: If the INTERRUPT MASK REGISTER is 0032₈, and the external status on channel 1 = 1004₈, what will be in A after execution of the following instruction?

COPY 1

Answer:

- (a) A is cleared.
- (b) Contents of INTERRUPT MASK REGISTER put in upper 12 bits of A.
- (c) External status on channel 1 put in lower 12 bits of A.

(A) = 00321004

20.9.3 Sense Interrupt

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	INTS	x, ch

23	18	17	15	14	12	11	00
77	4	ch	x				

ch = I/O channel designator, 0-7
x = Interrupt sensing mask code

Description: Sense for the interrupt conditions listed in the following table. RNI from P + 1 if an interrupt line is active corresponding to any "1" bit in the mask. If none of the selected lines are active, RNI from P + 2. Internal interrupts are cleared as soon as they are sensed. External interrupts are cleared by connecting the unit and executing one of the SEL functions controlling peripheral interrupts.

INTERRUPT SENSING MASK BIT ASSIGNMENTS

Mask Bit Position	Mask Code x	Interrupt Condition Represented	
00	0001	I/O Equip # 0 on designated channel	External Interrupts
01	0002	1	
02	0004	2	
03	0010	3	
04	0020	4	
05	0040	5	
06	0100	6	
07	0200	7	Internal Interrupts
08	0400	Real-time Clock	
09	1000	Exponent Overflow/Underflow (flt pt) and BCD Fault	
10	2000	Arithmetic Overflow and Divide Fault (integer)	
11	4000	Search/Move Completion	

Example: i) If there is an interrupt line active on channel 2, which instruction will be executed after executing the INTS instruction in the following section of code?

```
INTS    4B,2
UJP     INTADD
UJP     NØINT
```

Answer: UJP INTADD if equip. nr. 4 generated the interrupt, else UJP NØINT

ii) The contents of the A register = 37777777 and the instruction executed is:

```
INA     1
```

An arithmetic overflow is generated. What code is necessary to sense this fact?

Answer: INTS 2000B
UJP overflow-routine (P+1)
normal non-overflow next instruction (P+2)

Note: The "ch" designator is required only when sensing external interrupts originating on a channel.

20.9.4 Sense Internal Status



23	18 17 15 14	12 11	00
77	3	ch	x

ch = I/O channel designator, 0-7
 x = Internal Status Sensing Mask Code

Description: Sense for the internal conditions listed in the following table. RNI from P + 1 if an internal condition is present corresponding to any "1" bit in the mask. If none of the selected conditions are present, RNI from P + 2. Internal conditions are cleared as soon as they are sensed, except when they are related to channel "ch".

INTERNAL STATUS SENSING MASK BIT ASSIGNMENTS

Mask Bit Position	Mask Code x	Internal Condition Present
00	0001	Parity Error
01	0002	Read Active
02	0004	Write Active
03	0010	External (controller) Reject
04	0020	Internal (no response) Reject
05	0040	Illegal Write
06	0100	CØN or CØN and SEL executed and channel currently not busy
07	0200	Block Control Interrupt waiting
		Write: Buffer complete
		Read: Buffer complete or End of Record at peripheral
08	0400	Exponent Overflow/Underflow fault (flt pt option)
09	1000	Arithmetic Overflow fault (main arithmetic section)
10	2000	Divide fault (main arithmetic or flt pt opt)
11	4000	BCD fault (BCD option)

Example: The following coding example sums a table of 50 numbers. It detects and corrects for arithmetic overflow using the INS instruction.

	ENI	49,1	SET LOOP INDEX TO MAX NR - 1
	ENA	0	CLEAR THE ACCUMULATOR
LOOP	ADA	TABLE,1	ADD NEXT ENTRY OF TABLE TO THE ACCUMULATOR
*			
	INS	1000B,0	WAS THERE ARITHMETIC OVERFLOW
*	UJP	ØV	YES, JUMP TO CORRECT AT LOCATION ØV
*			
*	IJD	LOOP,1	AFTER LAST TIME THROUGH LOOP - NO DECREASE B1 BY 1 OR JUMP TO LOOP
FINIS	SHAQ	-24	CONVERT 24 BIT SUM TO 48 BITS
*	ADAQ	SUM48	ADD CONVERTED SUM TO 48 BIT ACCUMULATOR
ØV	UJP	EXIT	
	SHAQ	-24	CORRECT FOR ARITHMETIC OVERFLOW BY COMPLEMENTING THE SIGN
	XØA,S	-0	
*	ADAQ	SUM48	ADD CONVERTED SUM TO 48 BIT ACCUMULATOR
	STAQ	SUM48	
	ENA	0	CLEAR 24 BIT ACCUMULATOR
	IJD	LOOP,1	DECREASE B1 AND JUMP TO LOOP
	UJP	FINIS	
SUM48	DECD	0	
TABLE	BSS	50	

20.9.4 (cont.)

Exercise: Assuming that data channel 5 is busy at the moment, determine if the operation in progress is a Read or a Write. Write the coding required to do this check.

20.9.5 Copy Internal Status and Interrupt Mask Register

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	CINS	ch

23	18 17 15 14	12 11	00
77	3	ch	0000

ch = I/O channel designator, 0-7

Description: The CINS instruction performs the following functions:

- 1) The internal status information (see table in Section 20.9.4) is loaded into the lower 12 bit positions of the A register.
- 2) The contents of the Interrupt Mask register are loaded into the upper 12 bit positions of the A register.
- 3) RNI from P + 1

Example: If the Interrupt Mask register contains 0121_8 and the internal status is 0111_8 , what will be in the A register after execution of the following instruction?

CINS 1

Answer: a) The A register is cleared

b) The contents of the Interrupt Mask Register is copied in the upper 12 bits of A.

c) The current internal status is copied in the lower 12 bits of A.

(A) = 01210111_8

Exercises: i) If the Interrupt Mask register contains 7400_8 and a parity error has occurred on the channel for data channel 6 (called a "transmission parity error"), what will be in A after the execution of:

CINS 6

- ii) If the Interrupt Mask register contains 0403_8 and the reject instruction following a SEL was executed, what coding would be necessary to determine the cause of the reject. (Remember there are three sources of SEL rejects 1) channel busy, 2) function illegal, and 3) function not recognized)

20.9.6 Comments on Internal Status and the Interrupt Mask Register

At first glance, it appears that the INTS and INS instructions perform essentially the same function with relation to internal conditions. However, there is one important difference and that is the INTS instruction needs both the condition and the interrupt mask bit set to get a match. The INS instruction senses the internal condition without the necessity of having the interrupt mask register bit set.

20.10 CONSOLE TYPEWRITER INPUT/OUTPUT

20.10.1 General Description

The 3192 Console Typewriter is an on-line input-output (I/O) device; i.e., it requires no connection to a communication channel and no function codes are issued. The typewriter receives output data directly from storage via the lower 6 bits of the Data Bus. Inputs to storage are handled in the same manner.

The console typewriter consists of an electric typewriter and a typewriter control panel mounted on a desk console.

Used in conjunction with block control and the Register File, the typewriter may be used to enter a block of internal binary-coded characters into storage and to print out data from storage. The two storage addresses that define the limits of the block must be stored in the register file prior to an input or output operation. Register 23* contains the initial character address of the block, and register 33 contains the last character address, plus one. Because the initial character address is incremented for each storage reference, it always shows the address of the character currently being stored or dumped. Output operations occur at the rate of 15 characters per second. Input operations are limited by the operator's typing speed.

*The upper seven bits of registers 23 and 33 should be "0".

20.10.2 Operation

The general order of events when using the console typewriter for an input or output operation is:

- 1) Set tabs, margins and spacing. Turn on typewriter.
- 2) Clear
- 3) Check status
- 4) Type out or type in

20.10.2.1 Set tabs, margins, and spacing

All tabs, margins, and paper spacing must be set manually prior to the input or output operation. A tab may be set for each space on the typewriter between margins.

20.10.2.2 Clear

There are three types of clears which may be used to clear all conditions (except ENCODE FUNCTION) existing in the typewriter control. These are:

- 1) Internal Clear or a Master Clear

This signal clears all external equipment, the communications channels, the typewriter control, and sets the typewriter to lower case.

20.10.2.2 (cont.)

- 2) Clear Channel, Search/Move Control, or Type Control instruction (77.51).

This instruction selectively clears a channel, the S/M control, or, by placing a "1" in bit 08 of the instruction, the typewriter control, and sets the typewriter to lower case.

- 3) Clear Switch on typewriter.

This switch clears the typewriter control and sets the typewriter to lower case.

20.10.2.3 Status Checking

The programmer may wish to check the status of the typewriter before proceeding. This is done with the Pause instruction. Status response is returned to the computer via two status lines.

The typewriter control transmits two status signals that are checked by the Busy Comparison Mask using the Pause instruction. These status signals are:

Bit 09	Type Not Finish
Bit 10	Type Not Repeat

An additional status bit appears on sense line 08. This code is Type Busy, and is transmitted by block control in the computation section when a typewriter operation has been selected. If the programmer is certain of the status of the typewriter, this operation may be omitted.

20.10.2.4 Type In and Type Load

The Set Type In instruction or pressing the TYPE LOAD switch on the console or typewriter permits the operator to enter data directly into storage from the typewriter. When the TYPE LOAD indicator on the console or typewriter glows, the operator may begin typing. The Encode function switch must be depressed to enable backspace, tab, carriage return, and case shifts to be transmitted to the computer during a typewriter input operation.

Input is in character mode only. As each character is typed, the information is transmitted via the Data Bus to the storage address specified by block control. This address is incremented as characters are transmitted. When the current address equals the terminating address, the TYPE LOAD indicator goes off and the operation is terminated. Data is lost if the operator continues typing after the TYPE LOAD indicator goes off.

20.10.2.5 Type Out and Type Dump

The typewriter begins to type out when the computation section senses a Set Type Out instruction or the operator presses the TYPE DUMP switch on the console or typewriter. Single 6-bit characters are sent from storage to the typewriter via the lower 6 bits of the Data Bus. When the current address equals the terminating address, the TYPE DUMP indicator goes off and the operation is terminated.

During a Type Out operation, the keyboard is locked to prevent loss of data in the event a key is accidentally pressed.

20.10.3 Typewriter Console Switches and Indicators

The following table shows the function of each switch and indicator for the console typewriter.

Name	Switch (S) Indicator (I)	Description
HIGH TEMP	I	This indicator glows when the ambient temperature within the typewriter cabinet exceeds 110°F.
BUSY	I	This indicator shows that the TYPE LOAD or TYPE DUMP switch has been pressed and the operation is in progress.
POWER ON	I	This indicator shows that power is applied to the typewriter.
TYPE DUMP	S & I	This switch is in parallel with the TYPE DUMP switch on the main console and causes the computer to send data to the typewriter for print-out. It is a momentary contact switch that is illuminated until the last character in the block has been printed or the CLEAR button is pressed.
TYPE LOAD	S & I	This switch is in parallel with the TYPE LOAD switch on the main console and allows the computer to receive a block of input data from the typewriter. The TYPE LOAD indicator remains on until either the FINISH, REPEAT or CLEAR button is pressed, or until the last character of the block has been stored. If the program immediately reactivates the typewriter, it may appear that the light does not go off.
REPEAT	S & I	This switch is pressed during a Type Load operation to indicate that a typing error occurred. This switch deactivates busy sense line 10 (see PAUS instruction). If the computer does not respond, this light remains on.
FINISH	S & I	This switch is pressed during a Type Load operation to indicate that there is no more data in the current block. This action is necessary if the block that the operator has entered is smaller than the block defined by registers 23 and 33. This switch also deactivates busy sense line 09. If the computer does not respond, this light remains on.
INTERRUPT	S & I	This switch is in parallel with the MANUAL INTERRUPT switch on the console and is used to manually interrupt the computer program.
ENCODE FUNCTION	S & I	This switch enables the typewriter to send to storage the special function codes for backspace, tab, carriage return, upper-case shift, and lower-case shift.
CLEAR	S & I	This switch clears the typewriter controls and sets the typewriter to lower case but does not cancel Encode Function.

20.10.4 Character Codes

The following table lists the internal BCD codes, typewriter print-out and upper- or lower-case shift that applies to the console typewriter. All character transmission between the computation section and the typewriter is in the form of internal BCD. The typewriter logic makes the necessary conversion to the machine code.

Note: Shifting to upper case (57) or lower case (32) is not necessary except on keyboard letters where both upper and lower case is available. The standard type set for the 3192 has two sets of upper case letters and no lower case letters. This eliminates the need for specifying a case shift.

Those characters that are strictly an upper or lower case character are not affected by the case currently selected. Case selection only affects those characters that may be printed in either upper or lower case.

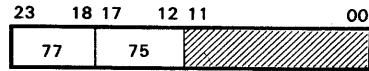
CONSOLE TYPEWRITER CODES

Print-out	Case	Internal BCD Code	Print-out	Case	Internal BCD Code
0	L	00	-	L	40
1	L	01	J	U or L	41
2	L	02	K	U or L	42
3	L	03	L	U or L	43
4	L	04	M	U or L	44
5	L	05	N	U or L	45
6	L	06	O	U or L	46
7	L	07	P	U or L	47
8	L	10	Q	U or L	50
9	L	11	R	U or L	51
+ =	U	12	° (degree)	U	52
"	L	13	\$	U	53
:	U	14	*	U	54
;	L	15	#	U	55
?	U	16	%	U	56
+	U	17	(Shift to UC)		57
A	U or L	20	(Space)		60
B	U or L	21	/	L	61
C	U or L	22	S	U or L	62
D	U or L	23	T	U or L	63
E	U or L	24	U	U or L	64
F	U or L	25	V	U or L	65
G	U or L	26	W	U or L	66
H	U or L	27	X	U or L	67
I	U or L	30	Y	U or L	70
(Shift to LC)		31	Z	U or L	71
.	U or L	32	&	U	72
)	U	33	,	U or L	73
!	L	34	(U	74
@	U	35	(Tab)		75
!	L	36	(Backspace)		76
		37	(Carriage Return)		77

20.10.5 Input/Output Instructions

20.10.5.1 Set Console Typewriter Input

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	CTI	



Bits 00 - 11 should be loaded with zeros

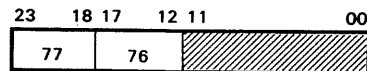
Description: This instruction, like the TYPE LOAD switch, permits a block of data to be entered into storage as soon as the Type Load indicator lights. If a block of data smaller than the one defined by registers 23 and 33 is to be typed, the FINISH switch should be depressed when the typing is completed. If more data is entered than the defined block can hold, the excess data is lost. If a typing error occurs, the REPEAT button should be depressed, the typewriter input operation is terminated and the appropriate status bits (09 and 10) may be sensed with the PAUS instruction.

Example: The below example illustrates initiating a typewriter INPUT. The INPUT data is to be stored in address INBUFF through INBUFF + 12.

ECHA	INBUFF	
TAM	23B	SET R.F. L0CA 23 T0 FCA
INA	13	
TAM	33B	SET R.F. L0CA 33 T0 LCA+1
CTI		INITIATE TYPEWRITER INPUT
.		
.		
.		
UCS		
INBUFF	BSS,C	13

20.10.5.2 Set Console Typewriter Output

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	CTO	



Bits 00 - 11 should be loaded with zeros

Description: This instruction, like the TYPE DUMP switch, causes the typewriter to print out the block of data defined by the character addresses in registers 23 and 33.

Note: The CTI and CTO instructions are mutually exclusive. Any attempt to execute one while the other is being executed will be ignored by the computer. Typewriter busy should be checked before these instructions are used and before registers 23 and 33 are altered.

20.10.5.2 (cont.)

Example: The following example illustrates initiating a typewriter OUTPUT. The OUTPUT will be from address ØUTBUFF through ØUTBUFF+15.

ECHA	ØUTBUFF	
TAM	23B	SET R.F. LØCA 23 TØ FCA
INA	16	
TAM	33B	SET R.F. LØCA 33 TØ LCA+1
CTØ		INITIATE TYPEWRITER ØUTPUT
.		
.		
.		
UCS		
ØUTBUFF	BSS,C	16

20.10.5.3 Pause Instruction (as used with Console Typewriter Instructions)

LOCATION	OPERATION MODIFIERS	ADDRESS FIELD
	PAUS	x



Bits 00 - 11 should be loaded with zeros

Description: This instruction allows the program to halt for a maximum of 40 ms if a condition (excluding typewriter-see note) defined by the pause sensing mask exists. If a "1" appears on a line that corresponds to a mask bit that is set, the count in P will not advance. If the advancement of P is delayed for more than 40 ms, a reject instruction is read from address P + 1. If none of the lines being sensed is active, or if they become inactive during the pause, the program immediately skips to address P + 2. If an interrupt occurs and is enabled during a PAUS, the pause condition is terminated, the interrupt sequence is initiated and the address of the PAUS instruction is stored as the interrupted address.

Note: If either bit 08, 09 or 10 (or any combination of these bits) is set and the sensed condition exists, a pause will not occur and the instruction at P + 1 is read up immediately. If these bits are set but the condition(s) does not exist, the program immediately skips to P + 2. For all other bits, the normal PAUS routine is followed.

PAUSE Sensing Mask Table

Mask Bit	Mask Code	Condition	Notes
00	0001	I/O channel 0 Busy	Channel Read or Write operation in progress, or the external Master Clear logic within the channel is set.
01	0002	1	
02	0004	2	
03	0010	3	
04	0020	4	
05	0040	5	
06	0100	6	
07	0200	7	
08	0400	Typewriter BUSY	Typewriter input or output in progress
09	1000	Typwrtr NOT FINISH	FINISH switch not depressed
10	2000	Typwrtr NOT REPEAT	REPEAT switch not depressed
11	4000	Search/Move control BUSY	Search or Move operation in progress

20.10.5.3 (cont.1)

Examples: i) This example initiates and waits until a typewriter OUTPUT is complete.

```

                ECHA      ØUTBUFF
                TAM        23B          SET R.F. LØCA 23 TØ FCA
                INA        16B
                TAM        33B          SET R.F. LØCA 33 TØ LCA+1
                CTØ
                PAUS       400B         INITIATE TYPEWRITER ØUTPUT
                UJP        *-1         WAIT UNTIL ØUTPUT IS CØMPLETE
                .
                .
                .
                UCS
ØUTBUFF BCD,C   16, THIS IS THE END

```

ii) Study the following example and determine what it will do.

```

*
                PAUS       400B
                UJP        *-1
                ECHA       INBUF
                TAM        23B
                INA        80
                TAM        33B
                CTI
*
                PAUS       400B
                UJP        *-1
                ECHA       CR
                TAM        23B
                INA        1
                TAM        33B
                CTØ
*
                PAUS       400B
                UJP        *-1
                ECHA       INBUFR
                TAM        23B
                INA        10
                TAM        33B
                CTI
*
                .
                .
                .
                UJP        ELSE
*
INBUF  BSS,C   80
CR     ØCT    -0
INBUFR BSS,C   10
        .
        .
        .
        etc.

```

It looks like the program will do an input, then a carriage return and then call for more input. If the first input is terminated by a REPEAT or FINISH, and 1) few or no characters were inputted or 2) if the operator is slow in getting his finger off the switch, then the second input will be terminated before he has a chance to type any further characters.

Verify this for yourself by studying the actions listed on the next page. The key is that the CTI can be initiated during the 50msec interval following the end of a type-out as the PAUS 400B senses buffer busy not typewriter busy.

20.10.5.3 (cont.2)

Note: Some programs use the REPEAT and FINISH switches as self clearing sense switches. This use is possible but is not recommended except under controlled conditions. The reason for this will be evident when the actions of the REPEAT and FINISH switches and PAUS instruction are studied below.

REPEAT or FINISH switch depressed

Typewriter NOT busy and not in the 50msec interval following a type-out operation

Action: Set REPEAT or FINISH status bit, turn on light

Typewriter executing a type-out or in the 50msec interval following a type-out operation

Action: None

Typewriter waiting for type-in

Action: Set REPEAT or FINISH status bit, turn on light and terminate type-in.

End of 50msec interval at end of type-out operation and REPEAT or FINISH switch being held down

Typewriter not busy.

Action: Set REPEAT or FINISH status bit, turn on light

Typewriter busy with new output

Action: None

Typewriter busy with type-in

Action: Set REPEAT or FINISH status bit, turn on light and terminate type-in.

At any time when the REPEAT or FINISH light is on and status bit set and the REPEAT or FINISH switch is being held down.

REPEAT or FINISH being sensed by PAUS instruction

Action: Clear REPEAT or FINISH status bit and turn off light. (Switch must be released and depressed again to reset) RNI at P+2, where otherwise it would be P+1.

Exercise: Write a subroutine that will output a carriage return and then tab the console typewriter. Then initiate a type-in of 1 character and wait until that character is received or a NOT busy, repeat or finish condition arises. Clear repeat and/or finish if present.

STUDENT NOTES

INTERRUPTS

21.1 INTERRUPTS USING CIC

21.2 INTERRUPTS WITHOUT USING CIC

21.3 I/O USING CIC

21.4 INTERRUPT MASK REGISTER BIT ASSIGNMENTS

21.5 CIT ASSIGNMENTS

21.1 INTERRUPTS USING CIC

Description: There are twelve groups of conditions for which, when one or more of these conditions occur, the programmer may wish to jump to a special sequence of coding, and upon completion of that task, resume processing in his original task. Processing in this manner is called "interrupt processing".

The twelve groups of conditions mentioned above are listed in the table "Interrupt Mask Register Bit Assignments" in Section 21.4 at the end of this discussion. This register (IMR) is set by the programmer.

In addition to the IMR, there are twelve flip-flops, each representing one of the conditions specified by the IMR. (Schematically, we may consider this as a 12-bit Interrupt Register.) When one of the 12 interruptible conditions occurs, the hardware sets the corresponding flip-flop in this Interrupt Register. If the corresponding bit has been set by the programmer in the IMR (and the interrupt system is enabled), interrupt processing will take place. The programmer must have previously placed the address of his interrupt routine in the CIT table.

Example: Add the integers in BUF thru BUF+99. If an arithmetic overflow occurs, jump to a routine to process the sum using double-precision (48-bit) integer arithmetic.

	EXT	CIT	
SUM	ØCT	0,0	
START	UJP	**	
	ENA	INTADD	*STORE ADDRESS
	ENI	7,1	ØF INTERRUPT ROUTINE AT CIT+7
	SWA	CIT,1	(SEE TABLE 21.5)
	SSIM	2000B	*SET IMR FOR ØVERFLOW CONDITION
	EINT		*ENABLE INTERRUPT SYSTEM
	.		
	.		
	.		
	.		
	ENA	0	
	ENI	99,1	
LØØP	ADA	BUF,1	*ADDITION LØØP
	IJD	LØØP,1	
*			*100 WORD SUM IS COMPLETE, EXCEPT
	SHAQ	-24	*ADD 24-BIT SUM IN *A*
	ADAQ	SUM	Ø 48-BIT SUM IN *SUM*
	STAQ	SUM	*ALL DONE
	.		
	.		
	.		
	.		
	UJP,I	START	
*			
INTADD	UJP	**	
	ENA	INTADD	1
	SWA	CIT,1	2
	ENI	1,1	3
	LDA	CIT,1	4
	SHAQ	-24	5
	XØA,S	-0	6
	ADAQ	SUM	7
	STAQ	SUM	8
	ENA	0	9
	STA	CIT,1	10
	UJP,I	INTADD	11

21.1 (cont.1)

Discussion of START+1 thru START+5 in the Example

Initialization to recognize an interrupt

- (1) the programmer must store the address of his interrupt routine in the lower 15 bits of the proper CIT table entry (See Section 21.5) and
- (2) the programmer must set the proper bit or bits in the IMR (See Section 21.4) and
- (3) the programmer must enable the interrupt system.

Discussion of INTADD in the Example

Processing the interrupt

- (1)* and (2)* CIC replaced the lower 15 bits of CIT+7 with the address of ABNORMAL. If we expect to get another overflow interrupt, we must again place the address of our routine in CIT+7. (Location CIT+7 applies only to overflow interrupts, see Section 21.5) Upon entry to an interrupt routine, $CIT+(B^1)$ equals the CIT table location where the FWA of an Interrupt Routine must be placed to process the next interrupt of the type currently being processed.
- (3)* and (4)* CIC stored the original contents (the "overflowed" sum) of the A register at CIT+1 and used the A register for its own purposes. We are merely re-loading the original A register contents at the time of overflow so that we may process the overflow.
- (4)*, (5)*, (6)*, (7)* & (8)* are processing the overflow condition into a 48-bit true sum.
- (9)* and (10)* Since CIC will load the A register from CIT+1 before returning to our original sequence, we must place into CIT+1 whatever quantity we desire to be in the A register upon re-entry into our original sequence.
- (11)* Returns us to CIC for housekeeping (explained later) before giving control to our original sequence.

Discussion of the Function of CIC

When an interruptible condition occurs, the Interrupt Register is compared to the IMR. If (a) any corresponding bits are set, and (b) the interrupt system has been enabled, then

- (1) the interrupt system is disabled,
- (2) the address of the instruction which would normally be executed next is stored in the lower 15 bits of absolute location 4,
- (3) an identifying code is stored in the lower 12 bits of location 5 and,
- (4) control is given to location 5.

Here are the contents of locations 4, 5 and 6 under SCOPE.

00004	UJP	**
00005	NØP	0
00006	UJP	CIC

*refers to card number in INTADD example

21.1 (cont.2)

Since the hardware gives control to location 5 (which is a no operation instruction), we see that a jump is made to CIC (Central Interrupt Control). CIC saves the contents of A, Q, B¹, B² and B³ at CIT+1 thru CIT+5. Since an identifying code (showing what condition caused the interrupt) is set in the lower 12 bits of location 00005, CIC knows which entry in CIT contains the address of the programmers interrupt routine.

CIC does a return jump to this location, and the programmer's interrupt routine (INTADD in this example) is executed. The interrupt routine does a UJP or UJP,I thru it's entry point to return to CIC when finished.

CIC restores the five registers, clears the flip-flop which was set by the interruptible condition (arithmetic overflow in the example), enables the interrupt system, and does an indirect jump through location 00004. The indirect jump gives control back to the instruction which was originally interrupted.

21.2 INTERRUPTS WITHOUT USING CIC*

If the programmer does not wish to use CIC to help in the processing of his interrupt, then at location 00006 he must store a jump to the address to which he wishes control to be given. In his interrupt routine, he must save and restore his registers (if his routine uses them) if he wishes them to have their original contents upon return to the main coding. He must also clear the flip-flop in the Interrupt Register (with an INCL instruction) which signalled the interruptible condition. His last two instructions should be:

```
EINT
UJP,I    4
```

If the last instruction in the interrupt routine does not use an indirect reference to location 00004, the control may never return to the main program. This is because after an EINT he is assured of being able to execute only one more instruction before the interrupt system can recognize another interrupt. Hence, if he has programmed for interrupt on more than one condition and another is present, the computer will interrupt again and the address portion of the jump instruction at location 00004 will be changed before it can be executed. Since it will be changed to 00004, the computer will stall on a jump to location 00004 after completing the processing of the current interrupt.

*Since CIC is essential for SCOPE I/O, it is assumed that no I/O will be done or that SCOPE isn't being used.

21.3 I/O USING CIC (See also Section 16.2.3)

In programming for I/O interrupts using CIO, all the housekeeping is done by CIO. (The programmer does not EINT, SWA CIT+k, or set the IMR with an SSIM). CIO actually uses CIC in the same manner that we indicated, but the programmer does not have to attend to the details.

Example: Suppose that we wish to output 30 words from locations BUFF thru BUFF+29 onto logical unit 15. We wish to continue executing instructions while the output is going on. However, as soon as the output is finished, we wish to be "interrupted" and process a special sequence of code, then return to our original sequence.

21.3 (cont.)

Following is the code that will accomplish the task:

```

EXT      CI0
START   UJP  **      ENTRY
      .
      .
      .
      ENA    0          SET BUFFER FLAG T0 BUSY
      STA   BFRBUSY
      RTJ   CI0       G0 T0 CI0 T0 -
      O2    15,2      WRITE WITH INTERRUPTS ON LU 15
      UJP   REJECT    WRITE REJECTED IF WE GET HERE
      40    BUFF      START OUTPUTTING FROM L0CA *BUFF*
      0     30        OUTPUT 30 WORDS
      0     INTADD    WHEN INTRPT OCCURS, G0 T0 INTADD
      .
      .
      .
      SSH   BFRBUSY   NEED *BUFF* NOW, IS WRITE DONE
      UJP   *-1       IF N0, L00P UNTIL FINISHED
      .
      .
      .
      UJP,I START     PR0GRAM ALL DONE, EXIT T0 SC0PE
*
INTADD  UJP  **      ENTRY P0INT 0F INTERRUPT ROUTINE
      .
      .
      .
      .
      ENA,S  -0       SET BUFFER FLAG T0 NOT BUSY
      STA   BFRBUSY
      UJP,I  INTADD   EXIT T0 MAIN C0DE VIA CIC
*
BFRBUSY 0CT   -0     +0 = BUSY, -0 = NOT BUSY

```

} Interrupt Routine
Special Sequence of Code

21.4 INTERRUPT MASK REGISTER BIT ASSIGNMENTS

Mask Bit Positions	Mask Codes (x)	Interrupt Conditions Represented
00	0001	I/O Channel 0 (includes interrupts generated within the channel and external equipment interrupts)
01	0002	1
02	0004	2
03	0010	3
04	0020	4
05	0040	5
06	0100	6
07	0200	7
08	0400	Real-time clock
09	1000	Exponent overflow/underflow & BCD faults
10	2000	Arithmetic overflow & divide faults
11	4000	Search/Move completion

21.5 CIT ASSIGNMENTS

Symbolic Location	Content	Explanation
CIT+0	Interrupt flag	If +0, no interrupt occurred. If -0, interrupt occurred.
+1	(A)	} Contents of these registers when last interrupt occurred. Registers are restored from here on exit from CIC.
+2	(Q)	
+3	(B ¹)	
+4	(B ²)	
+5	(B ³)	
+6	Real Time Clock	} Initially contains UJP ABNØRMAL. Address of user interrupt routine stored in CIT by user main program when the corresponding interrupt is selected.
+7	Arithmetic Overflow	
+8	Divide Fault	
+9	Exponent Overflow	
+10	BCD Fault	
+11	Search/Move	
+12	Manual Interrupt	
+13	Associated Processor Interrupt	
+14	Channel 0	} Initially contains UJP ABNØRMAL. Address is set by CIO when interrupt is selected. Table is extended for each pair of channels added to the hardware configuration*.
+15	Channel 1	
+16	Channel 2	
+17	Channel 3	
+18	Channel 4	
+19	Channel 5	
+20	Channel 6	
+21	Channel 7	

* The Central Interrupt Table (CIT) length varies from 15 to 21 entries and is dependent on the number of channels present in the system. The length of the table is specified at the time CIC is assembled under COMPASS prior to doing a PRELIB (Prepare library tape) of SCOPE.

SAMPLE COMPASS PROGRAMS

SEQUENCE,001	A1T
EXAMPLES	A1B - A9T
EXAMPLE 20 (AVERAGE)	A9B - A10B
EXAMPLE 19A(GR2PR)	A11T - A12T
EXAMPLE 21 (CODE)	A12B - A13B
EXAMPLE 22 (MEQ.TEST)	A14T - A15T
EXAMPLE 23 (TY.OUT)	A15B - A16B
EXAMPLE 24 (DEMO)	A17T - A21B
EXAMPLE 25 (MTDRIVER)	A22T - A24B
MAC	A25T - A26B
EXAMPLE 26 (COPY)	A27T - A29T
CIC VERSION SI 0.0	A29B - A31B
MTMTCIO	A32T - A33B
TYPEIN	A34T - A35T
SORT	A35B - A36B
TYPEOUT	A37T - A38T
FLOATF	A38B - A39B
CRDTP	A40T - A41B
MTMT	A42T - A43B
CVTBCDB	A44T - A45T
CVTBBCD	A45B - A46B
SEQUENCE,002	A47T
IDC	A47B - A48T
- Load/Snap/Run -	A48B
- Memory Map -	A49T
- Snap Number 1 & 2 -	A50T - A50B
SEQUENCE,003 and OCC's	A51T
- Memory Map -	A51B
- Memory Dump -	A52B

SEQUENCE,001
JOB,***
COMPASS,L,R

COMPASS-32 (2.1)

EXAMPLES

11/21/66 PAGE 1

UNDEFINED SYMBOLS
SAM
SYM
SYMBOL

MULTIPLY-DEFINED SYMBOLS
GEORGE

EXTERNAL SYMBOLS
BCDBOXS
FDPBOXS

LENGTH OF SUBPROGRAM	00372
LENGTH OF COMMON	00000
LENGTH OF DATA	00000

----- DEFINITION OF OPERATION CODE MODIFIERS -----

```
EQ    EQUAL
NE    NOT EQUAL
GE    GREATER THAN OR EQUAL TO
LT    LESS THAN
I     INDIRECT ADDRESSING
S     SIGN EXTENSION
INT   INTERRUPT
B     BACKWARD
H     HALF ASSEMBLY OR DISASSEMBLY
N     NO ASSEMBLY OR DISASSEMBLY
```

----- DEFINITION OF TERMS AND SYMBOLS -----

```
A     THE #A# REGISTER
Q     THE #Q# REGISTER
AQ    48 BIT REG ; (A AND Q COMBINED RESPECTIVELY)
QA    48 BIT REG ; (Q AND A COMBINED RESPECTIVELY)
E     THE 48 BIT #E# REGISTER
EL    LOWER 24 BITS OF #E#
EU    UPPER 24 BITS OF #E#
ED    THE 52 BIT (13 CHAR) #E DECIMAL# REGISTER. SAME AS
      #E# BUT EXTENDED FROM 48 TO 52 BITS.
AQE   96 BIT REG ; (A+Q AND E COMBINED RESPECTIVELY)
B     ANY ONE OF THE THREE INDEX REGISTERS
R1    INDEX REGISTER ONE ONLY
R2    INDEX REGISTER TWO ONLY
R3    INDEX REGISTER THREE ONLY
D     4 BIT #D# REGISTER
IMR   INTERRUPT MASK REGISTER

M     15 BIT WORD ADDRESS
R     17 BIT CHARACTER ADDRESS
FSCA  FIRST SOURCE CHARACTER ADDRESS
FDCA  FIRST DESTINATION CHARACTER ADDRESS
V     6 BIT REGISTER FILE ADDRESS
Y     15 BIT OPERAND
C     6 BIT SEARCH CHARACTER
FL    4 BIT OPERAND (BCD FIELD LENGTH)
P     CURRENT INSTRUCTION ADDRESS AS INDICATED BY THE P REG

( )   CONTENTS OF
(M)   24 BIT CONTENTS OF MEMORY LOCATION M
(M,M+1) 48 BIT CONTENTS OF MEM LOCS M AND M+1 RESPECTIVELY
(M)   6 BIT CONTENT OF MEMORY CHARACTER LOCATION M
(V)   24 BIT CONTENTS OF REG FILE LOCATION V
```

```
RNI   HEAD NEXT INSTRUCTION
      WHERE DATA IS BEING STORED INTO PART OF A MEMORY
      LOCATION, THE REMAINDER OF THAT LOCATION REMAINS
      UNCHANGED
```

ALL POSSIBLE OPERATION MODIFIERS ARE SHOWN WITH EACH MNEMONIC CODE, SEPARATED BY (.). THERE IS NO ORDER, WHERE MULTIPLE MODIFIERS ARE POSSIBLE. THE FOLLOWING MODIFIERS ARE OPTIONAL AND CAN BE DROPPED BY ELEMENATING BOTH THE MODIFIER AND THE PRECEDING COMMA.

I , S , INT , B , H , N ,

ON ALL CONDITIONAL JUMP AND SKIP INSTRUCTIONS, IF THE CONDITION IS NOT MET THEN RNI P+1

----- 3200 INSTRUCTION SET -----

LOAD CLASS

00000	20725252	20 1	25252 3	LDA+I	M,B	(M) TO A
00001	21725252	21 1	25252 3	LDQ+I	M,B	(M) TO Q
00002	22425253	22 1	05252 3	LACH	R,B1	(R) TO A 05-00 , A 23-06 CLEARED
00003	23425253	23 1	05252 3	LQCH	R,B2	(R) TO Q 05-00 , Q 23-06 CLEARED
00004	24725252	24 1	25252 3	LCA+I	M,B	(M) COMPLEMENT TO A
00005	25725252	25 1	25252 3	LDAQ+I	M,B	(M+M+1) TO AQ
00006	26725252	26 1	25252 3	LCAQ+I	M,B	(M+M+1) COMPLEMENT TO AQ
00007	27725252	27 1	25252 3	LDL+I	M,B	(M) L(M) TO A , M AND Q UNCHANGE
00010	54725252	54 1	25252 3	LDI+I	M,B	(M) [4-00] TO B

STORE CLASS

00011	40725252	40 1	25252 3	STA+I	M,B	(A) TO M
00012	41725252	41 1	25252 3	STQ+I	M,B	(Q) TO M
00013	42425253	42 1	05252 3	SACH	R,B2	(A 05-00) TO R
00014	43425253	43 1	05252 3	SQCH	R,B1	(Q 05-00) TO R
00015	44725252	44 1	25252 3	SWA+I	M,B	(A 14-00) TO M 14-00
00016	45725252	45 1	25252 3	STAQ+I	M,B	(AQ) TO M+M+1
00017	46725252	46 1	25252 3	SCHA+I	M,B	(A 16-00) TO M 16-00
00020	47725252	47 1	25252 3	STI+I	M,B	(B) TO M 14-00

24 BIT ARITH CLASS

00021	30725252	30 1	25252 3	ADA+I	M,B	(A)+(M) TO A
00022	31725252	31 1	25252 3	SBA+I	M,B	(A)-(M) TO A
00023	50725252	50 1	25252 3	MUA+I	M,B	(A)*(M) TO QA
00024	51725252	51 1	25252 3	DVA+I	M,B	(AQ)/(M) TO A , REMAINDER TO Q
00025	34725252	34 1	25252 3	RAD+I	M,B	(A)+(M) TO M , A IS UNCHANGED

48 BIT ARITH CLASS

00026	32725252	32 1	25252 3	ADAQ+I	M,B	(AQ)+(M+M+1) TO AQ
00027	33725252	33 1	25252 3	SBAQ+I	M,B	(AQ)-(M+M+1) TO AQ
00030	56725252	56 1	25252 3	MUAQ+I	M,B	(AQ)*(M+M+1) TO AQE
00031	57725252	57 1	25252 3	DVAQ+I	M,B	(AQE)/(M+M+1) TO AQ, REMUR TO E

FLOATING POINT ARITH CLASS

00032	60725252	60 1	25252 3	FAD+I	M,B	FP(AQ) + FP(M,M+1) TO AQ
00033	61725252	61 1	25252 3	FSR+I	M,B	FP(AQ) - FP(M,M+1) TO AQ
00034	62725252	62 1	25252 3	FMU+I	M,B	FP(AQ) * FP(M,M+1) TO AQ
00035	63725252	63 1	25252 3	FDV+I	M,B	FP(AQ) / FP(M,M+1) TO AQ

BCD CLASS

00036	64425253	64 1	05252 3	LOE	FCA,R1	(FCA THRU FCA+(D)-1) TO ED RIGHT JUSTIFIED
00037	65425253	65 1	05252 3	STF	FCA,R2	(ED) TO FCA THRU FCA+(D)-1 , RIGHT JUSTIFIED
00040	66425253	66 1	05252 3	ADE	FCA,R3	(ED)+(FCA THRU FCA+(D)-1) TO FD
00041	67425253	67 1	05252 3	SBE	FCA,R3	(ED)-(FCA THRU FCA+(D)-1) TO FD
00042	70300014	70 0	00014 3	SFE	K,B	SHIFT ED , K DIGITS LEFT OR RIGHT (END OFF)
00043	70700012	70 1	00012 3	SET	FL	FL TO D
00044	70625252	70 1	25252 2	EOJ	M	RNI M IF ED OVERFLOW CHARACTER IS NONZERO
00045	70425252	70 1	25252 0	EZJ+EQ	M	RNI M IF ED CONTAINS ZERO
00046	70525252	70 1	25252 1	EZJ+LT	M	RNI M IF ED IS NEGATIVE

24 BIT INTER-REGISTER TRANSFER CLASS

00047	53300000	53 0	00000 3	TIA	R	(B) TO A 14-00 , A 23-15 CLEARED
00050	53700000	53 1	00000 3	TAI	R	(A 14-00) TO B
00051	53010077	53 0	10077 0	TMQ	V	(V) TO Q
00052	53410077	53 1	10077 0	TQM	V	(Q) TO V
00053	53020077	53 0	20077 0	TMA	V	(V) TO A
00054	53420077	53 1	20077 0	TAM	V	(A) TO V
00055	53330077	53 0	30077 3	TMI	V,B	(V 14-00) TO B
00056	53730077	53 1	30077 3	TMV	V,R	(B) TO V 14-00 , V 23-15 CLEARED
00057	53040000	53 0	40000 0	AQA		(A)+(Q) TO A
00060	53340000	53 0	40000 3	AIA	R	(A)+(B) TO A
00061	53740000	53 1	40000 3	IAI	R	(A)+(R) TO B

48 BIT INTER-REGISTER TRANSFER

00062	55100000	55 0	00000 1	ELQ		(EL) TO Q
00063	55500000	55 1	00000 1	QEL		(Q) TO EL
00064	55200000	55 0	00000 2	EUA		(EU) TO A
00065	55600000	55 1	00000 2	AEU		(A) TO EU
00066	55300000	55 0	00000 3	EAQ		(E) TO AQ
00067	55700000	55 1	00000 3	AQF		(AQ) TO E

STOP AND JUMP CLASS

00070	14000000	14 0	00000 0	NOP		DO NOTHING
00071	00025252	00 0	25252 0	HLT	M	STOP, RNI M
00072	77770000	77 1	70000 3	UCS		STOP, RNI P+1
00073	01725252	01 1	25252 3	UJP,I	M,B	RNI M
00074	00725252	00 1	25252 3	RTJ	M	(P) TO M 14-00 , RNI M+1
00075	00125252	00 0	25252 1	SJ1	M	RNI M IF KEY 1 IS SET
00076	00225252	00 0	25252 2	SJ2	M	RNI M IF KEY 2 IS SET
00077	00325252	00 0	25252 3	SJ3	M	RNI M IF KEY 3 IS SET
00100	00425252	00 1	25252 0	SJ4	M	RNI M IF KEY 4 IS SET
00101	00525252	00 1	25252 1	SJ5	M	RNI M IF KEY 5 IS SET
00102	00625252	00 1	25252 2	SJ6	M	RNI M IF KEY 6 IS SET
00103	03025252	03 0	25252 0	AZJ,EQ	M	RNI M IF (A) EQ + OR = ZERO
00104	03125252	03 0	25252 1	AZJ,NE	M	RNI M IF (A) NE + OR = ZERO
00105	03225252	03 0	25252 2	AZJ,GE	M	RNI M IF (A) POSITIVE
00106	03325252	03 0	25252 3	AZJ,LT	M	RNI M IF (A) NEGATIVE
00107	03425252	03 1	25252 0	AQJ,EQ	M	RNI M IF (A) EQ (Q)
00110	03525252	03 1	25252 1	AQJ,NE	M	RNI M IF (A) NE (Q)
00111	03625252	03 1	25252 2	AQJ,GE	M	RNI M IF (A) GE (Q)
00112	03725252	03 1	25252 3	AQJ,LT	M	RNI M IF (A) LT (Q)
00113	02325252	02 0	25252 3	IJI	M,B	IF (B) NE 0 (R)+1 TO B AND RNI M. OW RNI P+1
00114	02725252	02 1	25252 3	IJD	M,B	IF (B) NE 0 (R)=1 TO B AND RNI M. OW RNI P+1

SKIP CLASS

WITHOUT SIGN EXTENSION , ONLY THE LOWER 15 BITS OF A OR Q ARE COMPARED WITH Y.
WITH SIGN EXTENSION , BIT 14 OF Y IS EXTENDED THRU BIT 23 MAKING Y 24 BITS FOR COMPARISON WITH ALL OF A OR Q.

00115	04412345	04 1	12345 0	ASE,S	Y	RNI P+2 IF (A) EQ Y , OW RNI P+1
00116	04512345	04 1	12345 1	QSE,S	Y	RNI P+2 IF (Q) EQ Y , OW RNI P+1
00117	04312345	04 0	12345 3	ISE	Y,B	RNI P+2 IF (B) FQ Y , OW RNI P+1
00120	04012345	04 0	12345 0	ISE	Y	RNI P+2 IF Y EQ 0 , OW RNI P+1
00121	05412345	05 1	12345 0	ASG,S	Y	RNI P+2 IF (A) GE Y , OW RNI P+1
00122	05512345	05 1	12345 1	QSG,S	Y	RNI P+2 IF (Q) GE Y , OW RNI P+1
00123	05312345	05 0	12345 3	ISG	Y,B	RNI P+2 IF (B) GE Y , OW RNI P+1
00124	10712345	10 1	12345 3	IS0	Y,B	IF (B) EQ Y CLEAR B AND RNI P+2. OTHERWISE
00125	10312345	10 0	12345 3	IS1	Y,B	IF (B) EQ Y CLEAR B AND RNI P+2. OTHERWISE (B)+1 TO B AND RNI P+1 (B)-1 TO B AND RNI P+1

SHIFT CLASS

POSITIVE SHIFT COUNT (K) INDICATES LEFT SHIFT. ALL LEFT SHIFTS ARE END AROUND.
NEGATIVE SHIFT COUNT (K) INDICATES RIGHT SHIFT. ALL RIGHT SHIFTS ARE END OFF WITH SIGN EXTENDED.

00126	12300014	12 0	00014 3	SMA	K,B	SHIFT (A) LEFT OR RIGHT K BITS
00127	12700014	12 1	00014 3	SHQ	K,B	SHIFT (Q) LEFT OR RIGHT K BITS
00130	13300014	13 0	00014 3	SHAQ	K,B	SHIFT (AQ) LEFT OR RIGHT K BITS
00131	13700014	13 1	00014 3	SCAQ	K,B	SCALE (AQ)

ENTER CLASS

WITHOUT SIGN EXTENSION , THE UPPER BITS OF A OR Q ARE CLEARED.
WITH SIGN EXTENSION , BIT 14 OF Y (BIT 16 OF R) IS EXTENDED MAKING Y OR R 24 BITS FOR ENTRY INTO ALL OF A OR Q.

00132	14412345	14 1	12345 0	ENA,S	Y	Y TO A 14-00
00133	14512345	14 1	12345 1	ENQ,S	Y	Y TO Q 14-00
00134	14312345	14 0	12345 3	ENT	Y,B	Y TO B

INCREASE CLASS

WITH OR WITHOUT SIGN EXTENSION , Y WILL BE TREATED AS A 24 BIT VALUE WHEN ADDED TO A OR Q.
WITHOUT SIGN EXTENSION , BITS 23-15 OF Y WILL BE ZERO.
WITH SIGN EXTENSION , BIT 14 OF Y IS EXTENDED THRU BIT 23 MAKING Y A 24 BIT VALUE.

IF Y IS A NEGATIVE VALUE , THE REGISTER WILL BE DECREASED BY THAT AMOUNT. (MUST USE SIGN EXT IN THIS CASE FOR A AND Q)

00135	15412345	15 1	12345 0	INA,S	Y	INCREASE (A) BY Y
00136	15512345	15 1	12345 1	INQ,S	Y	INCREASE (Q) BY Y
00137	15312345	15 0	12345 3	INT	Y,B	INCREASE (B) BY Y

LOGICAL INSTRUCTIONS WITHOUT STORAGE REFERENCE

WITH OR WITHOUT SIGN EXTENSION, Y IS TREATED AS A 24 BIT VALUE. WITHOUT SIGN EXTENSION, BITS 23-15 OF Y ARE ZEROS. WITH SIGN EXTENSION, BIT 14 OF Y IS EXTENDED THRU BIT 23 MAKING Y A 24 BIT VALUE FOR THE OPERATION.

00140 16412345 16 1 12345 0 XDA,S Y SC (A) BY Y
00141 16512345 16 1 12345 1 X0Q,S Y SC (Q) BY Y
00142 16312345 16 0 12345 3 X0I Y,B SC (B) BY Y
00143 17412345 17 1 12345 0 ANA,S Y LM (A) BY Y
00144 17512345 17 1 12345 1 ANQ,S Y LM (Q) BY Y
00145 17312345 17 0 12345 3 ANI Y,B LM (B) BY Y

LOGICAL INSTRUCTIONS WITH STORAGE REFERENCE

00146 35725252 35 1 25252 3 SSA,I 4,B SS (A) BY (M)
00147 36725252 36 1 25252 3 SCA,I 4,B SC (A) BY (M)
00150 37725252 37 1 25252 3 LPA,I 4,B LM (A) BY (M)

BUFFERED SEARCH AND MOVE CLASS

00151 71425261 71 1 05254 1 SRCE,INT C,FCA,LCA+1 SEARCH (FCA THRU LCA) FOR EQ C
00152 23025253 23 0 05252 3 UJP *-2 REJECT INSTRUCTION
00153 01000151 01 0 P00151 0 SRCN,INT C,FCA,LCA+1 SEARCH (FCA THRU LCA) FOR NE C
00154 71425261 71 1 05254 1 UJP *-2 REJECT INSTRUCTION
00155 23425253 23 1 05252 3 MOVE,INT FL,FSCA,FDCA MOVE (FSCA THRU FSCA+FL-1) TO
00156 01000154 01 0 P00154 0 FUCA THRU FDCA+FL-1
00157 72425653 72 1 05352 3 UJP *-2 REJECT INSTRUCTION
00160 05025253 05 0 05252 3

STORAGE TEST CLASS

00162 06225252 06 0 25252 2 MEQ 4,INTERVAL MASKED EQUALITY SEARCH
00163 07225252 07 0 25252 2 MTH 4,INTERVAL MASKED THRESHOLD SEARCH
00164 10025252 10 0 25252 0 SSH M STORAGE TEST AND SHIFT
00165 52725252 52 1 25252 3 CPR,I 4,B COMPARE (M) WITH (A) AND (Q)

INPUT / OUTPUT

00166 77067007 77 0 67007 0 CON CC,CH CONNECT TO I/O EQUIPMENT
00167 01000166 01 0 P00166 0 UJP *-1 REJECT INSTRUCTION
00170 77160022 77 0 60022 1 SEL FC,CH SELECT FUNCTION ON I/O EQUIPMENT
00171 01000170 01 0 P00170 0 UJP *-1 REJECT INSTRUCTION
00172 73025261 73 0 05254 1 INPC,INT,B,M CH,FCA,LCA+1 CHAR ADRS INPUT TO STORAGE
00173 63425253 63 1 05252 3 UJP *-2 REJECT INSTRUCTION
00174 01000172 01 0 P00172 0
00175 73400000 73 1 00000 0 INAC,INT CH CHAR ADRS INPUT TO A
00176 60400000 60 1 00000 0 UJP *-2 REJECT INSTRUCTION
00177 01000175 01 0 P00175 0
00200 74025322 74 0 25322 0 INPW,INT,B,N CH,FWA,LWA+1 WORD ADRS INPUT TO STORAGE
00201 63425252 63 1 25252 0 UJP *-2 REJECT INSTRUCTION
00202 01000200 01 0 P00200 0
00203 74400000 74 1 00000 0 INAW,INT CH WORD ADRS INPUT TO A
00204 60400000 60 1 00000 0 UJP *-2 REJECT INSTRUCTION
00205 01000203 01 0 P00203 0
00206 75025261 75 0 05254 1 OUTC,INT,B,M CH,FCA,LCA+1 CHAR ADRS OUTPUT FROM STORAGE
00207 63425253 63 1 05252 3 UJP *-2 REJECT INSTRUCTION
00210 01000206 01 0 P00206 0
00211 75400000 75 1 00000 0 OTAC,INT CH CHAR ADRS OUTPUT FROM A
00212 60400000 60 1 00000 0 UJP *-2 REJECT INSTRUCTION
00213 01000211 01 0 P00211 0
00214 76025322 76 0 25322 0 OUTW,INT,B,N CH,FWA,LWA+1 WORD ADRS OUTPUT FROM STORAGE
00215 63425252 63 1 25252 0 UJP *-2 REJECT INSTRUCTION
00216 01000214 01 0 P00214 0
00217 76400000 76 1 00000 0 OTAW,INT CH WORD ADRS OUTPUT FROM A
00220 60400000 60 1 00000 0 UJP *-2 REJECT INSTRUCTION
00221 01000217 01 0 P00217 0
00222 77750000 77 1 50000 3 CTI CONSOLE TYPEWRITER INPUT
00223 77760000 77 1 60000 3 CTO CONSOLE TYPEWRITER OUTPUT

SENSING AND COPY CLASS

00224	77260000	77 0	60000 2	COPY	CH	COPY EXTERNAL STATUS AND IMR TO A
00225	77360000	77 0	60000 3	CINS	CH	COPY INTERNAL STATUS AND IMR TO A
00226	77267777	77 0	67777 2	EAS	X,CH	SENSE EXTERNAL STATUS
00227	77367777	77 0	67777 3	INS	X,CH	SENSE INTERNAL STATUS
00230	77467777	77 1	67777 0	INTS	X,CH	SENSE INTERRUPT
00231	77607777	77 1	07777 2	PAUS	X	PAUSE

CONTROL CLASS

00232	77507777	77 1	07777 1	INCL	X	INTERRUPT CLEAR
00233	77517777	77 1	17777 1	IOCL	X	INPUT/OUTPUT CLFAR
00234	77710000	77 1	10000 3	SFPF		SET FLOATING POINT FAULT
00235	77720000	77 1	20000 3	SBCD		SET BCD FAULT

INTERRUPT CLASS

00236	77527777	77 1	27777 1	SSIM	X	SELECTIVELY SET IMR
00237	77537777	77 1	37777 1	SCIM	X	SELECTIVELY CLEAR IMR
00240	77730000	77 1	30000 3	DINT		DISABLE INTERRUPT SYSTEM
00241	77740000	77 1	40000 3	EINT		ENABLE INTERRUPT SYSTEM
00242	11425253	11 1	05252 3	ECHA,S	R	R TO A 16-00
00243	77570000	77 1	70000 1	IAPR		INTERRUPT ASSOCIATED PROCESSOR

----- TABLE OF MASK BIT ASSIGNMENTS -----

BIT CODE	SSIM	SCIM	INCL	INTS	IOCL	PAUS	TNS
00 0001	CH 0	CH 0	LINE 0	CH 0	CH 0	BUSY	PARITY ERR ON CH X
01 0002	CH 1	CH 1	LINE 1	CH 1	CH 1	BUSY	CH X BUSY READING
02 0004	CH 2	CH 2	LINE 2	CH 2	CH 2	BUSY	CH X BUSY WRITING
03 0010	CH 3	CH 3	LINE 3	CH 3	CH 3	BUSY	EXT REJECT ON CH X
04 0020	CH 4	CH 4	LINE 4	CH 4	CH 4	BUSY	NO RESP REJ ON CH X
05 0040	CH 5	CH 5	LINE 5	CH 5	CH 5	BUSY	* ILLEGAL WRITE
06 0100	CH 6	CH 6	LINE 6	CH 6	CH 6	BUSY	CH X PRESET CON/SEL
07 0200	CH 7	CH 7	LINE 7	CH 7	CH 7	BUSY	INTERNAL INT ON CH X
08 0400	REAL TIME	CLOCK	INT	TY	TY	BUSY	*EXP OVFLW FAULT
09 1000	*EXP OVFLW/BCD	FAULT	----	NOT	FINISH		*ARITH OVFLW FAULT
10 2000	*ARITH OVFLW/DIV	FAULT	----	NOT	REPEAT		*DIVIDE FAULT
11 4000	SEARCH/MOVE	COMPLETE	S/M	S/M	BUSY		*RCD FAULT

* INTERNAL FAULTS CLEARED BY SENSING

----- COMPASS CODING TECHNIQUES -----

ASSEMBLY OF CONSTANTS

00244	00000001	OCT	1,2,-2,50,-77
00245	00000002		
00246	77777775		
00247	00000050		
00250	77777700		
00251	00000001	DEC	1,2,50,-50,99,987654
00252	00000002		
00253	00000062		
00254	77777715		
00255	00000143		
00256	03611006		
00257	00000000	DEC0	1,-1,98,-1,-98,,9,8
00260	00000001		
00261	77777777		
00262	77777776		
00263	00000000		
00264	00000142		
00265	57763777		
00266	77777777		
00267	20076100		
00270	00000000		
00271	20044714		
00272	63146315		
00273	44256262	BCD	2,MESSAGE
00274	21272560		
00275	442562	BCD,C	3,MES
002753	62	BCD,C	5,SAGE
00276	21272560		
00277	77304643	HOLMSG	VFD 06/77,H18/HOL,09/0,A15/HOLMSG,C24/HOLMSG
00300	00002277		
00301	00001374		

STORAGE RESERVATION

00302		TOM	BSS	3
00305		TOMC	BSS,C	3
003053		TOMC1	BSS,C	1
00306	20000302 20 0 P00302 0	LDA	TOM	
00307	22001424 22 0 P00305 0	LACH	TOMC	
00310	22001427 22 0 P00305 3	LACH	TOMC1	

ADDRESSING MODES

ABSOLUTE

00311	20000012	20	0	00012	0		LDA	10
00312	20000010	20	0	00010	0	SYMR1	LDA	10B
00313	20077740	20	0	77740	0	SYMR2	LDA	-37B
00314	20000001	20	0	00001	0		LDA	SYMR2-SYMR1
00315	20000025	20	0	00025	0		LDA	ADRS
				00025		ADRS	EQU	25B

RELOCATABLE

00316	20000317	20	0	P00317	0		LDA	TEMP
00317	20000321	20	0	P00321	0	TEMP	LDA	TEMP+2
00320	20000317	20	0	P00317	0		LDA	TEMP1
				00317		TEMP1	EQU	TEMP

LITERALS

00321	20000361	20	0	P00361	0		LDA	=D9
00322	25000362	25	0	P00362	0		LDAW	=209
00323	25000364	25	0	P00364	0		LDAW	=209.
00324	20000360	20	0	P00360	0		LDA	=07777777
00325	25000366	25	0	P00366	0		LDAW	=2073737373737373
00326	20000357	20	0	P00357	0		LDA	=HABCD
00327	25000370	25	0	P00370	0		LDAW	=2HARCEFGH

SPECIAL CHARACTERS

00330	20000330	20	0	P00330	0		LDA	*
00331	20000332	20	0	P00332	0		LDA	*+1
00332	20000330	20	0	P00330	0		LDA	*-2
00333	20077777	20	0	77777	0		LDA	**

INTERCHANGE OF WORD AND CHARACTER ADDRESSES

	00334			WORDADR	EQU	*		
	00334			CHARADR1	EQU+C	*		
	003343			CHARADR2	EQU+C	*+3		
00334	20000334	20	0	P00334	0		LDA	CHARADR1
T 00335	20000334	20	0	P00334	0		LDA	CHARADR2
00336	22001560	22	0	P00334	0		LACH	WORDADR

TRUNCATION ERROR

SYMBOL EQUATIONS

25252	ADDRESS	EQU	25252B
25252	M	EQU	ADDRESS
052523	R	EQU+C	M+1
00003	B	EQU	3
00001	B1	EQU	1
00002	B2	EQU	2
00003	B3	EQU	3
052523	FCA	EQU+C	R
05254	LCA	EQU+C	R+5
00014	K	EQU	12
00077	V	EQU	77B
12345	Y	EQU	123453
053523	FUCA	EQU+C	FCA+256
052523	FSCA	EQU+C	FCA
00012	FL	EQU	12B
00006	CH	EQU	6
25252	FwA	EQU	4
25321	LwA	EQU	M+39
07777	X	EQU	7777B
07007	CC	EQU	7007B
00022	FC	EQU	22B
00002	INTERVAL	EQU	2
00023	C	EQU	23B

COMPASS ASSEMBLY ERRORS

```

A 00337 20000000 20 0 00000 0      LUA      PRB      ADDRESS FORMAT ERROR
                                COMMON
C 00340 20000005 20 0 00005 0      LUA      5      ATTEMPT TO ASSEMBLE INFO IN
00341      00000007      OCT      7      THE COMMON AREA
                                PRG
DA 00342 20000343 20 0 P00343 0     LUA      GEORGE     MULTIPLY DEFINED SYMBOL
00343      00000001      GEORGE    OCT      1
D 00344      00000012      GEORGE    DEC      10
L 00345 20000005 20 0 00005 0 123  LUA      5      LOCATION FIELD ERROR
M 00346 00000005 00 0 00005 0      LUA+1    5      OPERATION MODIFIER ERROR
O 00347 00000000 00 0 00000 0      LID      5      OPERATION CODE ERROR
U 00350 20000000 20 0 00000 0      LUA      SAM      UNDEFINED SYMBOL
T 00351 20000352 20 0 P00352 0     LUA      CHAR     TRUNCATION ERROR
      003523      CHAR     EWIJ+C  **3
*****
      LITERAL IN A CHARACTER ADDRESS INSTRUCTION
      LACH      =012
*****
      BLANK CHARACTERS IN SYMBOLS
L 00353 20000353 20 0 P00353 0     SYM BOL  LUA      *
U 00354 01000000 01 0 00000 0      UJP      SYM BOL
U 00355 01000000 01 0 00000 0      UJP      SYMBOL
*****

```

END

LITERALS

```

00356      00000012      00357      21222324      00360      77777777
00361      00000011
00362      000000000000000011      00364      20044400000000000
00366      7373737373737373      00370      2122232425242730

```

NUMBER OF LINES WITH DIAGNOSTICS 13

```

ADDRESS      25252      P25252
ADMS         00025      P00315
B            00003      P00000
                                P00001
                                P00006      P00004      P00005
                                P00007      P00010      P00011
                                P00012      P00015      P00016      P00017
                                P00020      P00021      P00022      P00023
                                P00024      P00025      P00026      P00027
                                P00030      P00031      P00032      P00033
                                P00034      P00035      P00042      P00047
                                P00050      P00055      P00056      P00060
                                P00061      P00073      P00113      P00114
                                P00117      P00123      P00124      P00125
                                P00126      P00127      P00130      P00131
                                P00134      P00137      P00142      P00145
                                P00146      P00147      P00150      P00165
                                P00002      P00014      P00036
                                P00003      P00013      P00037
                                P00040      P00041
                                P00151      P00154
                                P00166
                                P00166      P00170      P00172      P00175
                                P00200      P00203      P00206      P00211
                                P00214      P00217      P00224      P00225
                                P00226      P00227      P00230
                                P00351
                                P00334
                                P00335
                                P00170
                                P00036      P00037      P00040      P00041
                                P00151      P00154      P00172      P00206
                                P05352      P05252
                                P00157
                                P00043      P00157
                                P00157
                                P00200      P00214
                                P00342
                                P00277      P00277
                                P00162      P00163
                                P00042      P00176      P00177      P00130
                                P00131
                                P00151      P00154      P00172      P00206
                                P00200      P00214
                                P00000      P00001      P00004      P00005
                                P00006      P00007      P00010      P00011
                                P00012      P00015      P00016      P00017
                                P00020      P00021      P00022      P00023
                                P00024      P00025      P00026      P00027
                                P00030      P00031      P00032      P00033
                                P00034      P00035      P00044      P00045
                                P00046      P00071      P00073      P00074
                                P00075      P00076      P00077      P00100
                                P00101      P00102      P00103      P00104
                                P00105      P00106      P00107      P00114

```

MULTIPLE-DEFINED

```

CHAR      P00352 3
CHARADR1  P00334 0
CHARADR2  P00334 3
FC        00022
FCA       05252 3

```

```

FUCA      05352 3
FL        00012
FSCA      05252 3
FMA       25252
GEORGE    P00343
HOLMSG    P00277
INTERVAL  00002
K         00014

```

```

LCA       05254 0
LWA       25321
M         25252

```

COMPASS-32 (2.1)

EXAMPLES

11/21/66 PAGE 2

R 05252 3

SAM 00000 UNDEFINED
 SYM 00000 UNDEFINED
 SYMB1 P00312
 SYMB2 P00313
 SYMBOL 00000 UNDEFINED
 TEMP P00317
 TEMP1 P00317
 TOM P00302
 TOMC P00305
 TOMC1 P00305 3
 V 00077

WORDADRS P00334
 X 07777
 Y 12345

LITERAL P00366 73737373737373
 LITERAL P00360 7777777
 LITERAL P00362 0000000000000011
 LITERAL P00361 00000011
 LITERAL P00356 00000012
 LITERAL P00364 2004440000000000
 LITERAL P00370 2122232425262730
 LITERAL P00357 21222324

P00111
 P00146
 P00163
 P25252
 P00002
 P00242
 P00350
 P00354
 P00314
 P00314
 P00355
 P00316
 P00320
 P00306
 P00307
 P00310
 P00051
 P00055
 P00336
 P00226
 P00232
 P00115
 P00121
 P00125
 P00135
 P00141
 P00145
 P00325
 P00324
 P00322
 P00321
 P00352
 P00323
 P00327
 P00326

P00112
 P00147
 P00164
 P25321
 P00003
 P05252
 P00317
 P00052
 P00056
 P00227
 P00233
 P00116
 P00122
 P00132
 P00136
 P00142

P00113
 P00150
 P00165
 P00013
 P05254
 P00317
 P00053
 P00230
 P00236
 P00117
 P00123
 P00133
 P00137
 P00143

P00114
 P00162
 P05252
 P00014
 P00054
 P00231
 P00237
 P00120
 P00124
 P00134
 P00140
 P00144

SYMBOLS NOT REFERENCED

BCDROXS EXT FDPBOXS EXT

COMPASS-32 (2.1)

EXAMPLE 20 (AVERAGE)

11/21/66 PAGE 1

ENTRY-POINT SYMBOLS
START 00000

LENGTH OF SUBPROGRAM 00016
 LENGTH OF COMMON 00000
 LENGTH OF DATA 00000

THIS PROGRAM WILL COMPUTE THE AVERAGE OF THREE POSITIVE NUMBERS

00000	01077777	01 0	77777	0	START	ENTRY	START	
00001	20000011	20 0	P00011	0		UJP	**	EXIT TO MONITOR PROGRAM
00002	30000012	30 0	P00012	0		LDA	NUMB1	LOAD FIRST NUMBER INTO A
00003	30000013	30 0	P00013	0		ADA	NUMB2	ADD SECOND NUMBER
00004	13077777	13 0	77777	0		ADA	NUMB3	ADD THIRD NUMBER
00005	51000014	51 0	P00014	0		SHAW	-24	CONVERT TO A 48 BIT VALUE
00006	40000015	40 0	P00015	0		DVA	THREE	DIVIDE BY THREE
00007	77770000	77 1	70000	3		STA	AVG	STORE RESULT
00010	01000000	01 0	P00000	0		UCS		STOP
00011	00000100				NUMB1	UJP	START	GO TO EXIT ON RESTART
00012	00000011				NUMB2	OCT	100	CONSTANT (100 OCTAL)
00013	00000020				NUMB3	DEC	9	CONSTANT (9 DECIMAL, 11 OCTAL)
00014	00000003				THREE	OCT	20	CONSTANT (20 OCTAL)
00015					AVG	DEC	3	CONSTANT (3)
						BSS	1	RESERVED LOCATION FOR RESULT
						END	START	

NUMBER OF LINES WITH DIAGNOSTICS 0

AVG	P00015	P00006
NUMB1	P00011	P00001
NUMB2	P00012	P00002
NUMB3	P00013	P00003
START	P00000	P00010
THREE	P00014	P00005

ENTRY-POINT SYMBOLS
START 00000

LENGTH OF SUBPROGRAM 00045
LENGTH OF COMMON 00000
LENGTH OF DATA 00000

ENTRY START
THIS PROGRAM WILL READ ONE HOLLERITH CARD AND COPY IT ON THE LINE PRINTER

00000	01077777	01 0	77777 0	START	UJP	**	
00001	77012000	77 0	12000 0	REPEAT	CON	2000B,1	ESTABLISH CONNECTION TO CR
00002	01000001	01 0	P00001 0		UJP	*-1	REJECT INSTRUCTION
00003	74000045	74 0	P00045 0		INPW	1,BUF,BUF+20	INITIATE INPUT OF ONE CARD
00004	10000021	10 0	P00021 0				
00005	01000003	01 0	P00003 0		UJP	*-2	REJECT INSTRUCTION
00006	77310002	77 0	10002 3		INS	2,1	SENSE TO SEE IF CHANNEL 1 IS BUSY
00007	01000006	01 0	P00006 0		UJP	*-1	BUSY - THEN KEEP CHECKING
00010	77013000	77 0	13000 0		CON	3000B,1	NOT BUSY - THEN CONNECT TO PR
00011	01000010	01 0	P00010 0		UJP	*-1	REJECT INSTRUCTION
00012	76000045	76 0	P00045 0		OUTW	1,BUF,BUF+20	INITIATE OUTPUT OPERATION
00013	10000021	10 0	P00021 0				
00014	01000012	01 0	P00012 0		UJP	*-2	REJECT INSTRUCTION
00015	77310004	77 0	10004 3		INS	4,1	SENSE TO SEE IF CHANNEL 1 IS BUSY
00016	01000015	01 0	P00015 0		UJP	*-1	BUSY - THEN KEEP CHECKING
00017	77770000	77 1	70000 3		UCS		NOT BUSY - THEN STOP
00020	01000001	01 0	P00001 0		UJP	REPEAT	REPEAT ON RESTART
00021				BUF	BSS	20	
					END	START	

NUMBER OF LINES WITH DIAGNOSTICS 0

COMPASS-32 (2.1)

EXAMPLE 19A (CR2PR)

11/21/66 PAGE 1

BUF P00021
REPEAT P00001

P00003
P00020

P00003

P00012

P00012

SYMBOLS NOT REFERENCED

*START P00000

COMPASS-32 (2.1)

EXAMPLE 21 (CODE)

11/21/66 PAGE 1

ENTRY-POINT SYMBOLS
IN. OUT 00000

LENGTH OF SUBPROGRAM 00014
LENGTH OF COMMON 00000
LENGTH OF DATA 00000

COMPASS-32 (2.1)

EXAMPLE 21 (CODE)

11/21/66 PAGE 2

ENTRY IN,OUT
THIS PROGRAM WILL HANG IN A LOOP UNTIL THE SELECTIVE JUMP KEYS ARE
SET TO A CODE OF 35 OCTAL. PROGRAM REPEATS ON RESTART.

00000	01077777	01 0	77777 0	IN,OUT	UJP	**		
00001	00100003	00 0	P00003 1	LOOP	SJ1	**2	TEST BIT 0	
00002	01000001	01 0	P00001 0		UJP	LOOP		
00003	00200001	00 0	P00001 2		SJ2	LOOP	TEST BIT 1	
00004	00300006	00 0	P00006 3		SJ3	**2	TEST BIT 2	
00005	01000001	01 0	P00001 0		UJP	LOOP		
00006	00400010	00 1	P00010 0		SJ4	**2	TEST BIT 3	
00007	01000001	01 0	P00001 0		UJP	LOOP		
00010	00500012	00 1	P00012 1		SJ5	**2	TEST BIT 4	
00011	01000001	01 0	P00001 0		UJP	LOOP		
00012	00600001	00 1	P00001 2		SJ6	LOOP	TEST BIT 5	
00013	00000001	00 0	P00001 0		HLT	IN,OUT+1	STOP	
					END	IN,OUT		

NUMBER OF LINES WITH DIAGNOSTICS 0

COMPASS-32 (2.1)

EXAMPLE 21 (CODE)

11/21/66 PAGE 1

IN,OUT	P00000	P00013			
LOOP	P00001	P00002	P00003	P00005	P00007
		P00011	P00012		

ENTRY-POINT SYMBOLS
START 00000

LENGTH OF SUBPROGRAM 00014
LENGTH OF COMMON 00000
LENGTH OF DATA 00000

ENTRY START
THIS PROGRAM WILL COUNT THE NUMBER OF EVEN LOCATIONS IN MEMORY (8K)
WITH 35 IN CHARACTER POSITION 2.

00000	01077777	01	0	77777	0	START	UJP	**	
00001	14300000	14	0	00000	3		ENI	0,3	CLEAR COUNTER (R3)
00002	14117776	14	0	17776	1		ENI	17776R,1	SET R1 TO 17776B
00003	14603500	14	1	03500	2		ENA	3500B	SET SEARCH PATTERN IN A
00004	14707700	14	1	07700	3		ENQ	7700B	SET SEARCH MASK IN Q
00005	06200000	06	0	00000	2	REENTER	MEQ	0,2	*** SEARCH ***
00006	01000011	01	0	P00011	0		UJP	STOP	** NOT FOUND ** THEN STOP
00007	15300001	15	0	00001	3		INI	1,3	** FOUND ** THEN INCREASE COUNT
00010	01000005	01	0	P00005	0		UJP	REENTER	AND CONTINUE SEARCH
00011	77770000	77	1	70000	3	STOP	UCS		DISPLAY RESULTS IN B3
00012	00100000	00	0	P00000	1		SJ1	START	EXIT IF JUMP KEY 1 IS SET
00013	01000001	01	0	P00001	0		UJP	START+1	OTHERWISE REPEAT
							END	START	

NUMBER OF LINES WITH DIAGNOSTICS 0

COMPASS-32 (2.1)

EXAMPLE 22 (MEW,TEST)

11/21/66 PAGE 1

REENTER	P00005	P00010	
START	P00000	P00012	P00013
STOP	P00011	P00006	

COMPASS-32 (2.1)

EXAMPLE 23 (TY,OUT)

11/21/66 PAGE 1

ENTRY-POINT SYMBOLS
TY,OUT 00000

LENGTH OF SUBPROGRAM	00010
LENGTH OF COMMON	00000
LENGTH OF DATA	00000

COMPASS-32 (2.1)

EXAMPLE 23 (TY.0UT)

11/21/66 PAGE 2

ENTRY TY.0UT
THIS IS A CLOSED SUBROUTINE FOR TYPING MESSAGE ON THE CONSOLE TY.

CALLING SEQUENCE ECHA FCA (FIRST CHAR ADRS OF MSGE)
ENW NUMR CHARS (LENGTH OF MESSAGE)
RTJ TY.0UT
NEXT INSTRUCTION

NOTE TY.0UT MUST BE DECLARED EXTERNAL (EXT TY.0UT)

00000	01077777	01 0	77777	0	TY.0UT	UJP	**	
00001	77600400	77 1	00400	2		PAUS	400B	IS TY CURRENTLY BUSY
00002	01000001	01 0	P00001	0		UJP	*-1	YES - THEN KEEP CHECKING
00003	53420023	53 1	20023	0		IAM	23B	NO - THEN PLACE FCA IN REG 23B
00004	53040000	53 0	40000	0		AWA		FORM LCA+1
00005	53420033	53 1	20033	0		IAM	33B	PLACE LCA+1 IN REG 33B
00006	77760000	77 1	60000	3		CTO		INITIATE OUTPUT
00007	01000000	01 0	P00000	0		UJP	TY.0UT	RETURN TO USFR
						END		

NUMBER OF LINES WITH DIAGNOSTICS 0

COMPASS-32 (2.1)

EXAMPLE 23 (TY.0UT)

11/21/66 PAGE 1

TY.0UT P00000

P00007

ENTRY-POINT SYMBOLS
START 00000

LENGTH OF SUBPROGRAM 00614
LENGTH OF COMMON 00000
LENGTH OF DATA 00000

00000 01077777 01 0 77777 0 START ENTRY UJP START **

THE FOLLOWING INSTRUCTIONS WILL INITIATE AN OUTPUT TO THE CONSOLE TY

00001	11000034	11 0 P00007 0	ECHA	MSG	FIRST CHAR ADRS TO A
00002	53420023	53 1 20023 0	TAM	23B	THEN TO REG FILF LOC 23B
00003	11003060	11 0 P00614 0	ECHA	MSGEND	LAST CHAR ADRS TO A
00004	53420033	53 1 20033 0	TAM	33B	THEN TO REG FILF LOC 33B
00005	77760000	77 1 60000 3	CTO		INITIATE TYPF OHT
00006	00000000	00 0 P00000 0	HLT	START	EXIT BACK TO SCOPE ON RESTART

FOLLOWING IS THE MESSAGE

00007	31602144	MSG	BCD+C	40,I AM A CONTROL DATA 3200 COMPUTER. I AM
00010	60216023			
00011	46456351			
00012	46436024			
00013	21632160			
00014	03020000			
00015	60234644			
00016	47646325			
00017	51336031			
00020	60214460			
00021	77575757	OCT	77575757	
00022	21602231	BCD+C	44,A BINARY, FIXED WORD LENGTH, STORED PROGRAM,	
00023	45215170			
00024	73602631			
00025	67252460			
00026	66465124			
00027	60432545			
00030	27633073			
00031	60626346			
00032	51252460			
00033	47514627			
00034	51214473			
00035	77575757	OCT	77575757	
00036	24312731	BCD+C	16,DIGITAL COMPUTER	
00037	63214360			
00040	23464447			
00041	64632551			
00042	77775757	OCT	77775757	
00043	63302562	BCD+C	28,THESE ARE MY SPECIFICATIONS	
00044	25602151			
00045	25604470			
00046	60624725			
00047	23312631			
00050	23216331			
00051	46456260			
00052	77775757	OCT	77775757	
00053	01330205	BCD+C	36,1.25 MICRO SECOND MEMORY CYCLE TIME	
00054	60443123			
00055	51466062			

COMPASS-32 (2.1)

EXAMPLE 24 (DEMO)

11/21/66 PAGE 3

00056	25234645		
00057	24604425		
00060	44465170		
00061	60237023		
00062	43256063		
00063	31442560		
00064	77575757	OCT	77575757
00065	10427360	BCD+C	40,8K, 16K, OR 32K WORDS OF CORE STORAGE
00066	01064273		
00067	60465160		
00070	03024260		
00071	60664651		
00072	24626046		
00073	26602346		
00074	51256062		
00075	63465121		
00076	27256060		
00077	77575757	OCT	77575757
00100	02046022	BCD+C	20,24 BIT WORD LENGTH
00101	31636066		
00102	46512460		
00103	43254527		
00104	63306060		
00105	77575757	OCT	77575757
00106	03603145	BCD+C	20,3 INDEX REGISTERS
00107	24256760		
00110	51252731		
00111	62632551		
00112	62606060		
00113	77575757	OCT	77575757
00114	44644363	BCD+C	32,MULTI-LEVEL INDIRECT ADDRESSING
00115	31404325		
00116	65254360		
00117	31452431		
00120	51252363		
00121	60212424		
00122	51256262		
00123	31452760		
00124	77575757	OCT	77575757
00125	23302151	BCD+C	32,CHARACTER HANDLING INSTRUCTIONS
00126	21236325		
00127	51603021		
00130	45244331		
00131	45276031		
00132	45626351		
00133	64236331		
00134	46456260		
00135	77575757	OCT	77575757
00136	22642626	BCD+C	48,BUFFERED CHARACTER SEARCH AND MOVE OPERATIONS
00137	25512524		
00140	60233021		
00141	51212363		
00142	25516062		
00143	25215123		

COMPASS-32 (2.1)

EXAMPLE 24 (DEMO)

11/21/66 PAGE 4

00144	30602145		
00145	24604446		
00146	65256046		
00147	47255121		
00150	63314645		
00151	62606060		
00152	77575757	OCT	77575757
00153	23464562	BCD+C	20,CONSOLE TYPEWRITER
00154	46432560		
00155	63704725		
00156	66513163		
00157	25516060		
00160	77575757	OCT	77575757
00161	51252143	BCD+C	16,REAL TIME CLOCK
00162	60633144		
00163	25602343		
00164	46234260		
00165	77757575	OCT	77757575
00166	62632145	BCD+C	32,STANDARD ARITHMETIC CAPABILITIES
00167	24215124		
00170	60215131		
00171	63304425		
00172	63312360		
00173	23214721		
00174	22314331		
00175	63312562		
00176	77760600	OCT	77760600
00177	02046022	BCD+C	28,24 BIT ADD, SUB, MULT + DIV
00200	31636060		
00201	21242473		
00202	60626422		
00203	73604464		
00204	43636020		
00205	60243165		
00206	77576060	OCT	77576060
00207	04106022	BCD+C	20,48 BIT ADD + SUB
00210	31636060		
00211	21242460		
00212	20606264		
00213	22606060		
00214	77775757	OCT	77775757
00215	46476331	BCD+C	32,OPTIONAL ARITHMETIC CAPABILITIES
00216	46452143		
00217	60215131		
00220	63304425		
00221	63312360		
00222	23214721		
00223	22314331		
00224	63312562		
00225	77760600	OCT	77760600
00226	26476124	BCD+C	52,FP/DP PACKAGE FLOATING POINT ADD, SUB, MULT +
00227	47604721		
00230	23422127		
00231	25606060		

COMPASS-32 (2.1)

EXAMPLE 24 (DEMO)

11/21/66 PAGE 5

00232	60602643			
00233	46216331			
00234	45276047			
00235	46314563			
00236	60212424			
00237	73606264			
00240	22736044			
00241	64436360			
00242	20606060			
00243	77576060	OCT	77576060	
00244	60606060	BCD+C	48,	DIVIDE AND FIXED POINT 48 BIT
00245	60606060			
00246	60606060			
00247	60606060			
00250	60602431			
00251	65312425			
00252	60214524			
00253	60263167			
00254	25246047			
00255	46314563			
00256	60041060			
00257	22316360			
00260	77576060	OCT	77576060	
00261	60606060	BCD+C	28,	MULT + DIV
00262	60606060			
00263	60606060			
00264	60604464			
00265	43636020			
00266	60243165			
00267	77576060	OCT	77576060	
00270	22232460	BCD+C	52,BCD PACKAGE	BINARY CODED DECIMAL ADD + SUB
00272	47212342			
00273	21272560			
00274	60606060			
00275	60602231			
00276	45215170			
00277	60234624			
00300	25246024			
00301	25233144			
00302	21436060			
00303	21242460			
00304	20606264			
00305	22606060			
00306	7775757	OCT	7775757	
00307	31454764	BCD+C	28,INPUT/OUTPUT CAPARILITIES	
00310	63614664			
00311	63476463			
00312	60232147			
00313	21223143			
00314	31633125			
00315	62606060			
00316	77776060	OCT	77776060	
00317	02401060	BCD+C	20,2-8 DATA CHANNELS	

COMPASS-32 (2.1)

EXAMPLE 24 (DEMO)

11/21/66 PAGE 6

00320	24216321			
00321	60233021			
00322	45452543			
00323	62606060			
00324	77576060	OCT	77576060	
00325	01401060	BCD+C	44,1-8 EQUIPMENT CONTROLLERS PER DATA CHANNEL	
00326	25506431			
00327	47442545			
00330	63602346			
00331	45635146			
00332	43432551			
00333	62604725			
00334	51602421			
00335	63216023			
00336	30214545			
00337	25436060			
00340	77776060	OCT	77776060	
00341	60606060	BCD+C	40, AVAILABLE EQUIPMENT	SPEEDS
00342	21652131			
00343	43212243			
00344	25602550			
00345	64314744			
00346	25456360			
00347	60606060			
00350	60606060			
00351	60606247			
00352	25252462			
00353	77576060	OCT	77576060	
00354	60606060	BCD+C	48, MAGNETIC TAPES	7.5KC - 120KC
00355	60604421			
00356	27452563			
00357	31236063			
00360	21472562			
00361	60606060			
00362	60606060			
00363	60606060			
00364	60073305			
00365	42236040			
00366	60010200			
00367	42236060			
00370	77576060	OCT	77576060	
00371	60606060	BCD+C	48, CARD READER	1200 CARDS/MIN
00372	60602321			
00373	51246051			
00374	25212425			
00375	51566060			
00376	60606060			
00377	60606060			
00400	60606060			
00401	60010200			
00402	00602321			
00403	51246261			
00404	44314560			
00405	77576060	OCT	77576060	

COMPASS-32 (2.1)

EXAMPLE 24 (DEMO)

11/21/66 PAGE 7
150 - 250 CARDS/MIN

00406 60606060
 00407 60602321
 00410 51246047
 00411 64452330
 00412 25626060
 00413 60606060
 00414 60606001
 00415 05006040
 00416 60020500
 00417 60232151
 00420 24626144
 00421 31456060
 00422 77576060
 00423 60606060
 00424 60604721
 00425 47255160
 00426 63214725
 00427 60626321
 00430 63314645
 00431 62606003
 00432 05006040
 00433 60010000
 00434 00602347
 00435 44606051
 00436 25212060
 00437 77576060
 00440 60606060
 00441 60606060
 00442 60606060
 00443 60606060
 00444 60606060
 00445 60606060
 00446 60606060
 00447 60606060
 00450 60600101
 00451 00602347
 00452 44606047
 00453 64452330
 00454 77576060
 00455 60606060
 00456 60604331
 00457 45256047
 00460 51314563
 00461 25516260
 00462 60606060
 00463 60606001
 00464 05006040
 00465 60010000
 00466 00604331
 00467 45256261
 00470 44314560
 00471 77576060
 00472 60606060
 00473 60602431

OCT 77576060
BCD+C 48, PAPER TAPE STATIONS 350 - 1000 CPS READ

OCT 77576060
BCD+C 48, 110 CPS PUNCH

OCT 77576060
BCD+C 48, LINE PRINTERS 150 - 1000 LINES/MIN

OCT 77576060
BCD+C 52, DISC PACKS 97 MILLI SEC AVG ACCESS

COMPASS-32 (2.1)

EXAMPLE 24 (DEMO)

11/21/66 PAGE 8

00474 62236047
 00475 21234262
 00476 60606060
 00477 60606060
 00500 60606011
 00501 07604431
 00502 43433160
 00503 62252360
 00504 21652760
 00505 21232325
 00506 62626060
 00507 77576060
 00510 60606060
 00511 60606060
 00512 60606060
 00513 60606060
 00514 60606060
 00515 60606060
 00516 60606007
 00517 01426023
 00520 30215161
 00521 62252360
 00522 21652760
 00523 67262551
 00524 77576060
 00525 60606060
 00526 60602451
 00527 64446062
 00530 63465121
 00531 27256060
 00532 60606060
 00533 60606001
 00534 07604431
 00535 43433160
 00536 62252360
 00537 21652760
 00540 21232325
 00541 62626060
 00542 77576060
 00543 60606060
 00544 60606060
 00545 60606060
 00546 60606060
 00547 60606060
 00550 02050042
 00551 60406002
 00552 44602330
 00553 21516162
 00554 25236024
 00555 21632160
 00556 67262551
 00557 77576060
 00560 60606060
 00561 60604743

OCT 77576060
BCD+C 48, 71K CHAR/SEC AVG XFER

OCT 77576060
HCD+C 52, DRUM STORAGE 17 MILLI SEC AVG ACCESS

OCT 77576060
BCD+C 48, 250K - 2M CHAR/SEC DATA XFER

OCT 77576060
HCD+C 16, PLOTTERS

COMPASS-32 (2.1)

EXAMPLE 24 (DEMO)

11/21/66 PAGE 9

00562 46636325
 00563 51626060
 00564 77576060
 00565 60606060
 00566 60604721
 00567 27256051
 00570 25212425
 00571 51606060
 00572 77576060
 00573 60606060
 00574 60602351
 00575 63602431
 00576 62474321
 00577 70606445
 00600 31636260
 00601 77576060
 00602 60606060
 00603 60602145
 00604 24604421
 00605 45706046
 00606 63302551
 00607 60624725
 00610 23312143
 00611 31712524
 00612 60242565
 00613 31232562

OCT 77576060
 BCD+C 20, PAGE READER

OCT 77576060
 BCD+C 24, CRT DISPLAY UNITS

OCT 77576060
 BCD+C 40, AND MANY OTHER SPECIALIZED DEVICES

MSGEND EQU+C *
 END START

NUMBER OF LINES WITH DIAGNOSTICS 0

COMPASS-32 (2.1)

EXAMPLE 24 (DEMO)

11/21/66 PAGE 1

MSG P00007
 MSGEND P00614 0
 START P00000

P00001
 P00003
 P00006

ENTRY-POINT SYMBOLS
MTDRIVER 00000

LENGTH OF SUBPROGRAM 00101
LENGTH OF COMMON 00000
LENGTH OF DATA 00000

ENTRY MTDRIVER
THIS SUBPROGRAM IS AN I/O DRIVER DESIGNED TO HANDLE ALL MAG TAPE OPERATIONS. USE OF THIS DRIVER WILL GREATLY REDUCE THE AMOUNT OF EFFORT REQUIRED ON THE PROGRAMMER'S PART WHEN WORKING WITH MAG TAPE. HE NEED NOT EVEN BE CONCERNED WITH THE EQUIPMENT CONFIGURATION OF THE PARTICULAR MACHINE IN USE, SINCE THIS IS HANDLED BY MTDRIVER.

----- CALLING SEQUENCE -----

1. SET PARAMETERS INTO A + Q
 - (A) A = TAPE UNIT NUMBER (0-7)
 - (B) Q 23-18 = FUNCTION CODE
 - (C) Q 14-00 = BUFFER FWA (NEEDED FOR I/O REQUEST ONLY)
2. RTJ TO MTDRIVER (MTDRIVER MUST BE DECLARED EXTERNAL)

----- OPERATIONS AVAILABLE -----

OPERATION	FUNCTION CODE (OCTAL)
1. READ 30 WDS IN BCD (120 CHARS)	01
2. READ 40 WDS IN BINARY	02
3. WRITE 30 WDS IN BCD (120 CHARS)	03
4. WRITE 40 WDS IN BINARY	04
5. REWIND	10
6. UNLOAD	11
7. BACKSPACE	12
8. SEARCH FILE FORWARD	13
9. SEARCH FILE BACKWARD	14
10. WRITE FILE MARK	15
11. SKIP BAD SPOT	16

----- ON RETURN -----

1. B REGS ARE UNCHANGED.
2. IF REQUEST WAS I/O (FC 1-4), THE OPERATION WILL BE COMPLETE.
3. A 11-00 = STATUS FROM THE TAPE AT END OF OPERATION IF REQUEST WAS I/O (FC 1-4).
4. A IS MEANINGLESS IF NOT I/O REQUEST.
5. Q IS MEANINGLESS.

----- ERROR PROCEDURES -----

1. IF TAPE UNIT IS FOUND #NOT READY# A DIAGNOSTIC WILL BE TYPED (READY OR X). PROGRAM WILL CONTINUE WHEN UNIT IS READY.
2. ALL REJECTS WILL LOOP.

***** INITIALIZING PROCEDURE *****

00001	42000157	42	0	P00033	3	SACH	CONNECT*3	ESTABLISH CONNECT CODE	
00002	42000371	42	0	P00076	1	SACH	RDYMSG*9	SET TAPE NO. IN READY MESSAGE	
00003	14600000	14	1	00000	2	ENA	0	PLACE FUNCTION CODE	
00004	13000006	13	0	00006	0	SHAW	6	IN A 5=0.	
00005	42000400	42	0	P00100	0	SACH	FC	SAVE FUNCTION CODE	
00006	05600017	05	1	00017	2	ASG	17B	IS THIS AN	
00007	05600010	05	1	00010	2	ASG	10B	I/O REQUEST	
00010	01000013	01	0	P00013	0	UJP	INPOUT	YES - THEN JUMP	
00011	42000247	42	0	P00051	3	SACH	SELECT*3	NO - THEN SET SELECT CODE AND	
00012	01000033	01	0	P00033	0	UJP	CONNECT	GO TO CONNECT PROCEDURE	
00013	05600005	05	1	00005	2	INPOUT	ASG	5	TEST TO SEE IF FUNCTION CODE IS
00014	05600001	05	1	00001	2	ASG	1	LEGAL.	
00015	00000015	00	0	P00015	0	HLT	*	BLIND HALT ON ILLEGAL CODE	
00016	12000005	12	0	00005	0	SHA	5	POSITION FC AND FWA IN A	
00017	13000022	13	0	00022	0	SHAW	18	A 14=00 = FWA OF USER'S BUFFER	
								A 23 = 1 IF BCD REQUEST *0 IF BINARY	
00020	14700002	14	1	00002	3	ENQ	2	BCD SELECT CODE TO Q *	
00021	03300023	03	0	P00023	3	AZJ*LT	**2	IS REQUEST BCD	
00022	14700001	14	1	00001	3	ENQ	1	NO - THEN BINARY CODE TO Q	
00023	43000247	43	0	P00051	3	SUCH	SELECT*3	PLACE SELECTED CODE IN SEL INSTRUCTION	
00024	44000066	44	0	P00066	0	SWA	INPUT*1	SET FWA IN INPUT/OUTPUT	
00025	44000062	44	0	P00062	0	SWA	OUTPUT*1	INSTRUCTIONS	
00026	15600036	15	1	00036	2	INA	30	FORM BCD LWA*1	
00027	03300031	03	0	P00031	3	AZJ*LT	**2	IS REQUESTBCD	
00030	15600012	15	1	00012	2	INA	10	NO - THEN FORM BINARY LWA*1	
00031	44000065	44	0	P00065	0	SWA	INPUT	SET LWA*1 IN INPUT/OUTPUT	
00032	44000061	44	0	P00061	0	SWA	OUTPUT	INSTRUCTIONS	

***** CONNECT PROCEDURE *****

00033	77001000	77	0	01000	0	CONNECT	CUN	CONCODE,CH	CONNECT TO SPECIFIED MAG TAPE
00034	01000033	01	0	P00033	0	UJP	**1		

***** TEST FOR READY *****

00035	77200001	77	0	00001	2	EXS	0001,CH	IS THE CONNECTED UNIT READY
00036	01000051	01	0	P00051	0	UJP	SELECT	YES - THEN GO TO SELECT PROCEDURE
00037	77600400	77	1	00400	2	PAUS	400B	NO - THEN TYPE DIAGNOSTIC
00040	01000037	01	0	P00037	0	UJP	**1	WAIT FOR TY NOT BUSY
00041	11000360	11	0	P00074	0	ECHA	RDYMSG	-
00042	53420023	53	1	20023	0	TAM	23B	-
00043	15600013	15	1	00013	2	INA	11	- INITIATE TY OUTPUT
00044	53420033	53	1	20033	0	TAM	33B	-
00045	77760000	77	1	60000	3	CTU		
00046	77200001	77	0	00001	2	EXS	0001,CH	-
00047	01000051	01	0	P00051	0	UJP	**2	- LOOP UNTIL READY
00050	01000046	01	0	P00046	0	UJP	**2	-

***** SELECT PROCEDURE *****

00051	77100000	77	0	00000	1	SELECT	SEL	0000,CH	INITIATE NON-I/O OPERATION OR
00052	01000051	01	0	P00051	0	UJP	**1		SELECT MODE FOR I/O OPERATION
00053	22000400	22	0	P00100	0	LACH	FC	FUNCTION CODE TO A	
00054	05600006	05	1	00006	2	ASG	6	IS THIS AN I/O REQUEST	
00055	01000055	01	0	P00055	0	UJP	*	NO - THEN GO TO READ/WRITE PROCEDURE	
00056	01000072	01	0	P00072	0	UJP	EXIT	YES - THEN GO TO EXIT	

***** READ/WRITE PROCEDURE *****

00057	05600003	05	1	00003	2	ASG	3	IS THIS A READ REQUEST	
00060	01000065	01	0	P00065	0	UJP	INPUT	YES - GO TO INPUT INSTRUCTION	
00061	76077777	76	0	77777	0	OUTPUT	OUTW	CH,*,**	NO - THEN INITIATE OUTPUT
00062	00077777	00	0	77777	0				
00063	01000061	01	0	P00061	0	UJP	**2	LOOP ON REJECT	
00064	01000070	01	0	P00070	0	UJP	WAIT	GO TO WAIT BEFORE EXIT	
00065	74077777	74	0	77777	0	INPUT	INPW	CH,*,**	INITIATE INPUT
00066	00077777	00	0	77777	0				
00067	01000065	01	0	P00065	0	UJP	**2	LOOP ON REJECT	


```

***** WAIT FOR END OF OPERATION *****
00070 77200002 77 0 00002 2 WAIT   EXS      0002,CH      IS UNIT STILL BUSY
00071 01000070 01 0 P00070 0      UJP      *-1          YES - THEN LOOP
                                     -
***** EXIT PROCEDURE *****
00072 77200000 77 0 00000 2 EXIT   COPY     CH          RETURN TO USER
00073 01000000 01 0 P00000 0      UJP      MTDRIVER
***** MISCELLANEOUS *****
00074          51252124  RUYMSG  BCU,C    10,READY MT X
00075          70604463
00076          6067
00077          77000000  FC        VFD      06/77
00100          00000    FC        BSS,C    1
***** SYMBOL EQUATIONS *****
00000    CH      EQU      0          THIS DRIVER MAY BE ALTERED TO FIT ANY
01000    CONCODE EQU     1000B      EQUIPMENT CONFIGURATION BY ONLY THESE
                                     NUMBER OF LINES WITH DIAGNOSTICS      0

```

CH	00000	P00033	P00035	P00046	P00051
CONCODE	01000	P00061	P00065	P00070	P00072
CONNECT	P00033	P00033			
EXIT	P00072	P00001	P00012		
FC	P00100	P00056			
INPUT	P00013	P00005	P00053		
INPUT	P00065	P00010			
MTDRIVER	P00000	P00024	P00031	P00060	
OUTPUT	P00061	P00073			
RDYMSG	P00074	P00025	P00032		
SELECT	P00051	P00002	P00041		
WAIT	P00070	P00011	P00023	P00036	
		P00064			

EXTERNAL SYMBOLS
CIO

LENGTH OF SUBPROGRAM 00076
LENGTH OF COMMON 00000
LENGTH OF DATA 00000

LIRM READS,REWIND,STATUS,FORMAT

MACRO EXAMPLES

```

READS  MACRO (L,R,FWA,N,X,IA,M,C)
      EXT CIO
      RTJ CIO
      01 L,X
      IFT /*,R,1
      UJP *-2
      IFF /*,R,1
      UJP R
      M FWA
      C N
      IFF /0,X,1
      00 IA
      ENDM
REWIND MACRO (L,R,X,IA)
      EXT CIO
      RTJ CIO
      04 L,X
      IFT /*,R,1
      UJP *-2
      IFF /*,R,1
      UJP R
      IFF /0,X,1
      00 IA
      ENDM
STATUS MACRO (L,X)
      EXT CIO
      RTJ CIO
      13 L,X
      ENDM
FORMAT MACRO (L,R,X)
      EXT CIO
      RTJ CIO
      14 L,X
      IFT /*,R,1
      UJP *-2
      IFF /*,R,1
      UJP R
      ENDM
READS (14,*,INBUFF,40,,,40)
      EXT CIO
      RTJ CIO
      01 14,0
      IFT *,*,1
      UJP *-2
      IFF *,*,1
      40 INBUFF
      0 40

```

```

00000 00777777 00 1 X77777 3
00001 01000016 01 0 00016 0
00002 01000000 01 0 P00000 0
00003 40000026 40 0 P00026 0
00004 00000050 00 0 00050 0

```

HEADS
HEADS
HEADS
HEADS
HEADS
HEADS
HEADS
HEADS

COMPASS-32 (2.1) MAC

11/21/66 PAGE 3

IFF 0*0*1

READS

00005 00700000 00 1 X00000 3
 00006 01200016 01 0 00016 2
 00007 01000024 01 0 P00024 0
 00010 40000026 40 0 P00026 0
 00011 40000050 40 0 00050 0
 00012 00000022 00 0 P00022 0

READS (14,REJX,INBUFF,40*2,INTADD,40*40)
 EXT CIO
 RTJ CIO
 01 14*2
 IFT *REJX*1
 IFF *REJX*1
 UJP REJX
 40 INBUFF
 40 40
 IFF 0*2*1
 00 INTADD

READS
 READS
 READS
 READS
 READS
 READS
 READS
 READS
 READS
 READS

00013 00700005 00 1 X00005 3
 00014 04000016 04 0 00016 0
 00015 01000013 01 0 P00013 0

REWIND (14,*)
 EXT CIO
 RTJ CIO
 04 14*0
 IFT ***1
 UJP **2
 IFF ***1
 IFF 0*0*1

REWIND
 REWIND
 REWIND
 REWIND
 REWIND
 REWIND
 REWIND

00016 00700013 00 1 X00013 3
 00017 14200016 14 0 00016 2
 00020 01000016 01 0 00016 0

FORMAT (14,14,2)
 EXT CIO
 RTJ CIO
 14 14*2
 IFT *14*1
 IFF *14*1
 UJP 14

FORMAT
 FORMAT
 FORMAT
 FORMAT
 FORMAT

00021 77770000 77 1 70000 3
 00022 01077777 01 0 77777 0
 00023 01400022 01 1 P00022 0
 00024 01077777 01 0 77777 0
 00025 01400024 01 1 P00024 0
 00026

INTADD UCS
 REJX UJP **
 INBUFF UJP,I INTADD
 UJP **
 UJP,I REJX
 BSS 40
 END

NUMBER OF LINES WITH DIAGNOSTICS 0

COMPASS-32 (2.1)

MAC

11/21/66 PAGE 1

CIO
 INBUFF P00026
 INTADD P00022
 REJX P00024

EXTERNAL

P00000
 P00003
 P00012
 P00007
 P00005
 P00010
 P00023
 P00025

P00013 P00016

EXTERNAL SYMBOLS
PRDRIVER
CRDRIVER
MTDRIVER

ENTRY-POINT SYMBOLS
COPY 00000

LENGTH OF SUBPROGRAM 00112
LENGTH OF COMMON 00000
LENGTH OF DATA 00000

ENTRY COPY
EXT MTDRIVER,CRDRIVER,PRDRIVER
THIS SUBPROGRAM IS DESIGNED TO TEST THE DRIVERS THAT WERE WRITTEN
PREVIOUSLY (EXAMPLE 25 AND WORKSHEET 8). IT COPIES A DECK OF HOLLERITH
CARDS TO MAG TAPE, REWINDS THE TAPE AND COPIES THE TAPE ON THE PRINTER.
PROGRAM REPEATS ON RESTART UNLESS JUMP KEY 1 IS SET IN WHICH CASE
IT WILL EXIT BACK TO SCOPE.

00000 01077777 01 0 77777 0 COPY UJP ** LINKAGE BACK TO SCOPE

----- REWIND TAPE 1 -----

00001 14600010 14 1 00010 2 ENA 108 PLACE FUNCTION CODE
00002 13077771 13 0 77771 0 SHAQ -6 IN Q 23-1R (REWIND)
00003 14600001 14 1 00001 2 ENA 1 PLACE TAPE NUMBER IN A 5-1
00004 00777777 00 1 X77777 3 RTJ MTDRIVER

----- READ NEXT CARD -----

00005 14600054 14 1 P00054 2 NEXTCARD ENA BUF
00006 13077755 13 0 77755 0 SHAQ -18
00007 14600001 14 1 00001 2 ENA 1
00010 13077771 13 0 77771 0 SHAQ -6
00011 00777777 00 1 X77777 3 RTJ CRDRIVER

----- TEST FOR END OF FILE CARD -----

00012 17600010 17 1 00010 2 ANA 108
00013 03100023 03 0 P00023 1 AZJ,NE FILEMARK

----- OUTPUT CARD IMAGE TO MAG TAPE 1 -----

00014 14600054 14 1 P00054 2 ENA BUF
00015 13077755 13 0 77755 0 SHAQ -18
00016 14600003 14 1 00003 2 ENA 3
00017 13077771 13 0 77771 0 SHAQ -6
00020 14600001 14 1 00001 2 ENA 1
00021 00700004 00 1 X00004 3 RTJ MTDRIVER
00022 01000005 01 0 P00005 0 UJP NEXTCARD

----- CLOSE OUT MAG TAPE 1 WITH A FM -----

00023	14600015	14	1	00015	2	FILEMARK	ENA	15B
00024	13077771	13	0	77771	0		SHAQ	-6
00025	14600001	14	1	00001	2		ENA	1
00026	00700021	00	1	X00021	3		RTJ	MTDRIVER

----- REWIND MAG TAPE 1 -----

00027	14600010	14	1	00010	2		ENA	10B
00030	13077771	13	0	77771	0		SHAQ	-6
00031	14600001	14	1	00001	2		ENA	1
00032	00700026	00	1	X00026	3		RTJ	MTDRIVER

----- READ NEXT RECORD FROM MAG TAPE 1 -----

00033	14600054	14	1	P00054	2	NEXTREC	ENA	BUF
00034	13077755	13	0	77755	0		SHAQ	-1B
00035	14600001	14	1	00001	2		ENA	1
00036	13077771	13	0	77771	0		SHAQ	-6
00037	14600001	14	1	00001	2		ENA	1
00040	00700032	00	1	X00032	3		RTJ	MTDRIVER

----- TEST FOR END OF FILE -----

00041	17600010	17	1	00010	2		ANA	10B
00042	03100051	03	0	P00051	1		AZJ,NE	EXIT

----- OUTPUT RECORD TO PRINTER -----

00043	14600054	14	1	P00054	2		ENA	BUF
00044	13077755	13	0	77755	0		SHAQ	-1B
00045	14600003	14	1	00003	2		ENA	3
00046	13077771	13	0	77771	0		SHAQ	-6
00047	00777777	00	1	X77777	3		RTJ	PRDRIVER
00050	01000033	01	0	P00033	0		UJP	NEXTREC

----- EXIT PROCEDURE -----

00051	77770000	77	1	70000	3	EXIT	UCS	
00052	00100000	00	0	P00000	1		SJ1	COPY
00053	01000001	01	0	P00001	0		UJP	COPY+1

----- BUFFER AREA -----

00054						BUF	BSS	30
							END	COPY

NUMBER OF LINES WITH DIAGNOSTICS 0

COMPASS-32 (2.1) EXAMPLE 26 (COPY) 11/21/66 PAGE 1

BUF	P00054		P00005	P00014	P00033	P00043
COPY	P00000	EXTERNAL	P00052	P00053		
CRURIVER			P00011			
EXIT	P00051		P00042			
FILEMARK	P00023	EXTERNAL	P00013	P00021	P00026	P00032
MTURIVER			P00004			
			P00040			
NEXTCARD	P00J05		P00022			
NEXTREC	P00033	EXTERNAL	P00050			
PRURIVER			P00047			

COMPASS-32 (2.1) CIC VERSION SI 0.0 12/07/64 11/21/66 PAGE 1

EXTERNAL SYMBOLS
CUI
ABNORMAL

ENTRY-POINT SYMBOLS
CIC 00000
CIT 00044

LENGTH OF SUBPROGRAM 00072
LENGTH OF COMMON 00000
LENGTH OF DATA 00000

WHENEVER AN INTERRUPT OCCURS, THE HARDWARE
 1. DISABLES THE INTERRUPT SYSTEM 00006
 2. REPLACES (00004, BITS 14-00) WITH THE LOCATION OF THE 00007
 INTERRUPTED INSTRUCTION 00009
 3. REPLACES (00005, BITS 11-00) WITH AN IDENTIFYING CODE 00010
 4. TRANSFERS CPU CONTROL TO LOCATION 00005. 00011

SCOPE-32 LOADS THE FOLLOWING LINKAGE TO CIC 00013
 (00004) UJP 0 00014
 (00005) NOP 0 00015
 (00006) UJP CIC 00016

CIC PERFORMS THE FOLLOWING TASKS 00018
 1. LOGICALLY IGNORES AN INTERRUPT ON A DINT INSTRUCTION 00019
 2. SAVES THE (A), (Q), (B1), (B2) AND (B3) 00020
 3. SETS LOCKOUT FLAG TO IDENTIFY INTERRUPT PROCESSING 00021
 4. TRANSFORMS INTERRUPT CODE INTO RELATIVE CIT ENTRY 00022
 5. SETS ENTRY LOCATION (FROM CIT) TO LINK TO USER 00023
 6. SETS INCL INSTRUCTION FOR EXECUTION ON USER RETURN 00024
 7. RESTORES ABNORMAL ADDRESS TO CIT ENTRY 00025
 8. ENTERS USER INTERRUPT PROCEDURE 00026
 9. CLEARS NON-I/O INTERRUPTS 00027
 10. RESTORES LOCK-OUT FLAG 00028
 11. RESTORES SAVED REGISTERS 00029
 12. ENABLES INTERRUPT SYSTEM 00030
 13. EXITS TO INTERRUPTED PROGRAM 00031
 ***** 00032

ENTRY 00034
 EXT ABNORMAL,COI 00035

00000 45000045 45 0 P00045 0 CIC STAQ CIT+1 SAVE (A) AND (Q) 00037
 LOGICALLY IGNORE INTERRUPT ON DINT 00038
 00001 24400004 24 1 00004 0 LCA+1 4 ((A))= INTERRUPTED INSTRUCTION 00039
 00002 37000043 37 0 P00043 0 LPA DINT MASK DINT INST (ZERO BITS 11-0) 00040
 00003 03100006 03 0 P00006 1 AZJ+NE CIC2.1 JUMP IF ((A)) NOT DINT 00041
 00004 20000045 20 0 P00045 0 LDA CIT+1 ELSE RESTORE (A) 00042
 00005 01400004 01 1 00004 0 UJP+1 4 AND EXIT, INTERRUPT DISABLED 00043
 SAVE REGISTERS IN CIT 00044
 00006 47100047 47 0 P00047 1 CIC2.1 STI CIT+3,1 SAVF (B1) 00045
 00007 47200050 47 0 P00050 2 STI CIT+4,2 (B2) 00046
 00010 47300051 47 0 P00051 3 STI CIT+5,3 (B3) 00047
 SET LOCK-OUT FLAG 00048
 00011 14477777 14 1 77777 0 ENA+S -0 00049
 00012 40000044 40 0 P00044 0 STA CIT 00050
 TRANSFORM INTERRUPT CODE TO RELATIVE CIT ENTRY 00051
 00013 54100005 54 0 00005 1 LUI 5,1 LOAD AND 00052
 00014 17107777 17 0 07777 1 ANI 77778,1 MASK INTERRUPT CODE 00053
 00015 05100110 05 0 00110 1 ISG 1108,1 SKIP IF NON-I/O INTERRUPT 00054
 00016 01000040 01 0 P00040 0 UJP CIC4.1 ELSE GO PROCESS THEM 00055
 00017 15177675 15 0 77675 1 INI -1028,1 COMPUTE INDEX RELATIVE TO CIT, 00056
 SET USER ENTRY (FROM CIT) INTO RTJ 00057
 00020 20100044 20 0 P00044 1 CIC5.0 LDA CIT,1 USER ENTRY LOCATION GOES 00058
 00021 44000026 44 0 P00026 0 SWA CIC8.0 TO RTJ INSTRUCTION 00059
 SET INCL MASK FOR CLEARING INTERRUPT 00060
 00022 12000006 12 0 00006 0 SHA 6 CHARACTER 0 CARRIES MASK 00061
 00023 42000136 42 0 P00027 2 SACH CIC9.0+2 00062
 RESTORE ABNORMAL ADDRESS TO CIT ENTRY 00063
 00024 14677777 14 1 X77777 2 ENA ABNORMAL 00064
 00025 44100044 44 0 P00044 1 SWA CIT,1 00065
 ENTER USER INTERRUPT PROCEDURE 00066
 00026 00700026 00 1 P00026 3 CIC8.0 RTJ * 00067
 CLEAR INTERRUPT IF NON-I/O 00068
 00027 77500000 77 1 00000 1 CIC9.0 INCL 0 *MASK WILL BE ZERO ON I/O 00069
 REM INTERRUPTS, CLEARED WITHIN CIO 00070
 RESTORE LOCK-OUT FLAG 00071
 00030 14400000 14 1 00000 0 ENA+S 0 00072
 00031 40000044 40 0 P00044 0 STA CIT 00073
 RESTORE SAVED REGISTERS 00074
 00032 54100047 54 0 P00047 1 LUI CIT+3,1 00075
 00033 54200050 54 0 P00050 2 LDI CIT+4,2 00076
 00034 54300051 54 0 P00051 3 LDI CIT+5,3 00077
 00035 25000045 25 0 P00045 0 LDAQ CIT+1 00078
 ENABLE INTERRUPT AND EXIT TO INTERRUPTED PROGRAM 00079
 00036 77740000 77 1 40000 3 EINT 00080
 00037 01400004 01 1 00004 0 UJP+1 4 00081
 TRANSFORM I/O INTERRUPT CODE 00082
 00040 17100007 17 0 00007 1 CIC4.1 ANI 7,1 MASK CHANNEL BITS 00083
 00041 15100016 15 0 00016 1 INI CIT*0-CIT,1 INCREASE (B1) TO RELATIVE 00084
 00042 01000020 01 0 P00020 0 UJP CICS.0 CHANNEL ENTRY AND CONTINUE 00085
 CONSTANTS AND CENTRAL INTERRUPT TABLE 00086
 00043 77730000 77 1 30000 3 DINT DINT 00087

COMPASS-32 (2.1) CIC VERSION SI 0.0 12/07/64 11/21/66 PAGE 4

C I T

00044 77777777 CIT OCT 77777777 INTERRUPT LOCK-OUT FLAG.
 REM VALUE IS =0 WHEN INTERRUPT PROCESSING IN PROGRESS,
 REM INTERRUPT DISABLED, AND IS +0 OTHERWISE

00045 00000000
 00046 00000000
 00047 00000000
 00050 00000000
 00051 00000000

00052 04000024 04 0 X00024 0
 00053 20000052 20 0 X00052 0
 00054 20000053 20 0 X00053 0
 00055 10000054 10 0 X00054 0
 00056 10000055 10 0 X00055 0
 00057 40000056 40 0 X00056 0
 00060 00000057 00 0 X00057 0
 00061 00000060 00 0 X00060 0

00062 00077777 00 0 X77777 0 CITCH0
 00063 00000062 00 0 X00062 0
 00064 00000063 00 0 X00063 0
 00065 00000064 00 0 X00064 0
 00066 00000065 00 0 X00065 0
 00067 00000066 00 0 X00066 0
 00070 00000067 00 0 X00067 0
 00071 00000070 00 0 X00070 0

CONTENTS OF REGISTERS ON ENTRY TO CIC SAVED IN CIT AND RESTORED FROM THESE LOCATIONS

OCT 0 (A)
 OCT 0 (Q)
 OCT 0 (R1)
 OCT 0 (R2)
 OCT 0 (R3)

NON-I/O USER INTERRUPT LINKAGE CELLS. INCL MASK IS IN CHARACTER 0 AND GOES INTO BIS 11-06 OF INCL INSTRUCTION. ADDRESS IS INITIALIZED TO ABNORMAL AND RESTORED PRIOR TO ENTRY TO A USER ROUTINE.

04 ABNORMAL CODE=0110 REAL TIME CLOCK
 20 ABNORMAL 0111 ARITHMETIC OVERFLOW
 20 ABNORMAL 0112 DIVIDE FAULT
 10 ABNORMAL 0113 EXPONENT OVERFLOW
 10 ABNORMAL 0114 BCD FAULT
 10 ABNORMAL 0115 SEARCH/MOVE
 40 ABNORMAL 0116 MANUAL FROM CONSOLE
 00 ABNORMAL 0117 ADJACENT PROCESSOR

I/O INTERRUPT LINKAGE CELLS. INCL MASK ACTS AS NOP - A CLEAR MUST BE GIVEN IN THE I/O INTERRUPT PROCEDURE. ADDRESS INITIALIZATION IS PERFORMED BY CIO.

00 CDI CHANNEL 0
 00 CDI CHANNEL 1
 00 CDI CHANNEL 2
 00 CDI CHANNEL 3
 00 CDI CHANNEL 4
 00 CDI CHANNEL 5
 00 CDI CHANNEL 6
 00 CDI CHANNEL 7

END

NUMBER OF LINES WITH DIAGNOSTICS 0

COMPASS-32 (2.1) CIC VERSION SI 0.0 12/07/64 11/21/66 PAGE 1

ABNORMAL EXTERNAL P00024 00064 P00052 00108 P00053 00109 P00054 00110
 P00055 00111 P00056 00112 P00057 00113 P00060 00114

CDI EXTERNAL P00061 00115
 P00062 00119 P00063 00120 P00064 00121 P00065 00122
 P00066 00123 P00067 00124 P00070 00125 P00071 00126

CIC2.1 P00006
 CIC4.1 P00040
 CIC5.0 P00020
 CIC8.0 P00026
 CIC9.0 P00027
 CIT P00044

CITCH0 P00062
 DINT P00043

P00003 00041
 P00016 00055
 P00042 00085
 P00021 00059
 P00023 00062
 P00000 00037
 P00010 00047
 P00031 00073
 P00035 00078
 P00041 00084
 P00002 00040

P00004 00042 P00006 00045 P00007 00046
 P00012 00050 P00020 0005R P00025 00065
 P00032 00075 P00033 00074 P00034 00077
 P00041 00084

SYMBOLS NOT REFERENCED
 *CIC P00000

EXTERNAL SYMBOLS
CIO

ENTRY-POINT SYMBOLS
MTMTCIO 00000

LENGTH OF SUBPROGRAM 00274
LENGTH OF COMMON 00000
LENGTH OF DATA 00000

THIS SUBPROGRAM COPIES A VARIABLE SIZE RECORD INPUT TAPE (LOGICAL UNIT 14) TO ANOTHER TAPE (LOGICAL UNIT 15) THE DENSITY OF THE OUTPUT TAPE WILL BE THE DENSITY SET BY THE OPERATOR. THIS SUBPROGRAM DOES NOT CHECK FOR TAPE ERRORS. PARITY IS CHECKED ONLY TO INSURE THAT THE MODE OF THE INPUT TAPE IS THE SAME AS THE MODE SELECTED.

	ENTHY	MTMTCIO	
00000	01077777 01 0 77777 0	MTMTCIO	
00001	00777777 00 1 X77777 3	EXT	CIO
00002	04000016 04 0 00016 0	UJP	**
00003	01000001 01 0 P00001 0	RTJ	CIO
00004	00700001 00 1 X00001 3	04	14
00005	04000017 04 0 00017 0	UJP	*-2
00006	01000004 01 0 P00004 0	RTJ	CIO
00007	00700004 00 1 X00004 3	04	15
00010	14200016 14 0 00016 2	UJP	*-2
00011	01000007 01 0 P00007 0	RTJ	CIO
00012	00700007 00 1 X00007 3	14	14+2
00013	01000016 01 0 00016 0	UJP	*-2
00014	01000012 01 0 P00012 0	RTJ	CIO
00015	00000374 00 0 P00077 0	01	14
00016	40000764 40 0 00764 0	UJP	*-2
00017	00700012 00 1 X00012 3	RTJ	CIO
00020	13100016 13 0 00016 1	13	14+1
00021	03200017 03 0 P00017 2	AZJ+GE	*-2
00022	17702010 17 1 02010 3	ANQ	2010R
00023	04700000 04 1 00000 3	QSF	0
00024	01000046 01 0 P00046 0	UJP	TEST
00025	37000076 37 0 P00076 0	LPA	MASK
00026	31000015 31 0 P00015 0	SBA	FCA
00027	44000041 44 0 P00041 0	SWA	NOC
00030	22000041 22 0 P00010 1	LACH	MODE+1
00031	42000155 42 0 P00033 1	SACH	OUTMODE+1
00032	00700017 00 1 X00017 3	RTJ	CIO
00033	14000017 14 0 00017 0	14	15+0
00034	01000032 01 0 P00032 0	UJP	*-2
00035	00700032 00 1 X00032 3	RTJ	CIO
00036	02000017 02 0 00017 0	02	15
00037	01000035 01 0 P00035 0	UJP	*-2
00040	00000374 00 0 P00077 0	00,C	INBUFF
00041	40000000 40 0 00000 0	40	0
00042	00700035 00 1 X00035 3	RTJ	CIO
00043	13100017 13 0 00017 1	13	15+1
00044	03200042 03 0 P00042 2	AZJ+GE	*-2
00045	01000007 01 0 P00007 0	UJP	MODE-1
00046	17700010 17 1 00010 3	TEST	ANQ 10B
00047	04700000 04 1 00000 3	QSF	0
00050	01000061 01 0 P00061 0	UJP	EOF

REWIND INPUT TAPE

REWIND OUTPUT TAPE

ASSUME AND SET INPUT TAPE TO BINARY MODE (MODE MAY BE CHANGED LATER)

INITIATE READING INPUT RECORD FROM LOGICAL UNIT 14 (MAX RECORD SIZE IS 500 CHARACTERS)

WAIT UNTIL THE INPUT IS COMPLETE

MASK OUT EVERY SATAUS BIT EXCEPT EOF AND PARITY

WAS THERE AN EOF OR PARITY ERROR YES JUMP TO TEST.

AFTER STATUS REQUEST (A) = LCA+1 CALCULATE NUMBER OF INPUT CHARACTERS AND STORE IN OUTPUT

FETCH MODE OF INPUT RECORD AND STORE TO SET OUTPUT MODE THE SAME AS INPUT.

INITIATE WRITING OUTPUT BUFFER

WAIT UNTIL OUTPUT IS COMPLETE

LOOP TO READ NEW INPUT RECORD

WAS AN EOF READ YES JUMP TO EOF

COMPASS-32 (2.1) MTMTCIO
00051 77770000 77 1 70000 3

UCS

11/21/66 PAGE 3
IF PARITY ERROR STOP SO THE
OPERATOR MAY DECIDE IF IT IS A
LEGAL PARITY ERROR OR WRONG MODE

00052 00700042 00 1 X00042 3
00053 06000016 06 0 00016 0
00054 01000052 01 0 P00052 0
00055 22000041 22 0 P00010 1
00056 16600030 16 1 00030 2
00057 42000041 42 0 P00010 1
00060 01000007 01 0 P00007 0

RTJ CIO
06 14
UJP *-2
LACH MODE+1
XOA 308
SACH MODE+1
UJP MODE-1

BACKSPACE INPUT TAPE A RECORD
FETCH ASSUMED MODE AND
CHANGE IT.
SET NEW MODE IN SEL CALL AND
TRY AGAIN.

00061 00700052 00 1 X00052 3 EOF
00062 11000017 11 0 00017 0
00063 01000061 01 0 P00061 0
00064 77700000 77 1 00000 3

RTJ CIO
11 15
UJP *-2
SLS

WRITE EOF ON OUTPUT TAPE

00065 00100067 00 0 P00067 1
00066 01000007 01 0 P00007 0

SJ1 EXIT
UJP MODE-1

STOP IF SELECTIVE STOP SWITCH
SET ON CONSOLE.
EXIT AND UNLOAD TAPES IF JUMP
KEY 1 IS SET ON CONSOLE.
IF NOT JUMP TO READ INPUT RECORD

00067 00700061 00 1 X00061 3 EXIT
00070 05000016 05 0 00016 0
00071 01000067 01 0 P00067 0
00072 00700067 00 1 X00067 3
00073 05000017 05 0 00017 0
00074 01000072 01 0 P00072 0
00075 01400000 01 1 P00000 0
00076 00377777 MASK
00077 INBUFF

RTJ CIO
05 14
UJP *-2
RTJ CIO
05 15
UJP *-2
UJP+I MTMTCIO
OCT 377777
BSS+C 500
END MTMTCIO

UNLOAD INPUT TAPE
UNLOAD OUTPUT TAPE
EXIT TO SCOPE

NUMBER OF LINES WITH DIAGNOSTICS 0

COMPASS-32 (2.1) MTMTCIO

11/21/66 PAGE 1

CIO
EOF P00061
EXIT P00067
FCA P00015
INBUFF P00077
MASK P00076
MODE P00010
MTMTCIO P00000
NOC P00041
OUTMODE P00033
TEST P00046

EXTERNAL

P00001
P00017
P00052
P00050
P00065
P00026
P00015
P00025
P00030
P00060
P00075
P00027
P00031
P00024

P00004
P00032
P00061
P00040
P00045
P00066

P00007
P00035
P00067
P00055
P00057

P00012
P00042
P00072

ENTRY-POINT SYMBOLS
TYPEIN 00000

LENGTH OF SUBPROGRAM 00027
LENGTH OF COMMON 00000
LENGTH OF DATA 00000

THIS SUBROUTINE ALLOWS FOR TYPING IN A VARIABLE QUANTITY OF
FOUR DECIMAL DIGIT NUMBERS. THE INPUT IS TERMINATED WHEN A
PERIOD IS TYPED IN. THE CALLING SEQUENCE IS.

CALL (P) = RTJ TYPEIN
(P+1) = RETURN
(A) = FIRST CHARACTER ADDRESS OF INPUT BUFFER

OUTPUT (B1) = NUMBER OF NUMBERS

0000	01077777	01 0	77777 0	TYPEIN	ENTHY	TYPEIN	
00001	14100000	14 0	00000 1		UJP	**	
00002	46000020	46 0	P00020 0		ENI	0,1	CLEAR B1
00003	46000010	46 0	P00010 0		SCHA	CKPERIOD	
00004	15400004	15 1	00004 0		SCHA	SET23	SAVE INPUT PAR.
00005	77600400	77 1	00400 2		INA+S	4	
00006	01000005	01 0	P00005 0		PAJS	4008	IS THE TYPEWRITER BUSY
00007	53420033	53 1	20033 0		UJP	*-1	YES LOOP
					TAM	338	SET LAST CHARACTER ADDRESS+1 OF INPUT BUFFER INTO REGISTER FILE LOCATION 338.
00010	11377777	11 0	77777 3	SET23	ECHA	**	
00011	53420023	53 1	20023 0		TAM	238	SET FIRST CHARACTER ADDRESS INTO REGISTER FILE LOCATION 238.
00012	77750000	77 1	50000 3		CTI		SET TYPE IN
00013	77602000	77 1	02000 2	LOOP	PAUS	2000R	HAS THE REPEAT SWITCH ON TYPE-
00014	01000016	01 0	P00016 0		UJP	FINISH	WRITER BEEN PUSHED-YES JUMP TO
00015	01000010	01 0	P00010 0		UJP	SET23	ADDRESS SET23 AND START OVER-
00016	77601000	77 1	01000 2	FINISH	PAUS	1000R	NO JUMP TO FINISH.
00017	01000013	01 0	P00013 0		UJP	LOOP	HAS THE FINISH SWITCH BEEN
00020	22377777	22 0	77777 3	CKPERIOD	LACH	**	PUSHED-NO JUMP TO LOOP.
00021	04600033	04 1	00033 2		ASE	338	IS THE FIRST CHARACTER A PERIOD-
00022	01000024	01 0	P00024 0		UJP	BYPASS	YES EXIT-NO JUMP TO BYPASS
00023	01400000	01 1	P00000 0		UJP+I	TYPEIN	
00024	15100001	15 0	00001 1	BYPASS	INI	1+1	INCREASE B1 BY 1
00025	53020033	53 0	20033 0		TMA	338	SET NEW INPUT BUFFER AND LOOP
00026	01000002	01 0	F00002 0		UJP	TYPEIN+2	TO TYPEIN+2
					END		

NUMER OF LINES WITH DIAGNOSTICS 0

COMPASS-32 (2.1)

TYPEIN

11/21/66 PAGE 1

BYPASS P00024
CKPERIOD P00020
FINISH P00016
LOOP P00013
SET23 P00010
TYPEIN P00000

P00022
P00002
P00014
P00017
P00003 F00015
P00023 P00026

COMPASS-32 (2.1)

SORT

11/21/66 PAGE 1

ENTRY-POINT SYMBOLS
SORT 00000

LENGTH OF SUBPROGRAM 00030
LENGTH OF COMMON 00000
LENGTH OF DATA 00000

COMPASS-32 (2.1) SORT

11/21/66 PAGE 2

THIS SUBPROGRAM SORTS A BLOCK OF NUMBERS INTO ASCENDING ORDER. UPON ENTRY THE ADDRESS OF THE FIRST NUMBER IS LOCATED IN THE LOWER 15 BITS OF THE A REGISTER AND THE NUMBER OF NUMBERS IS IN B2.

CALL (P) =RTJ SORT
(P+1)=RETURN
(A) =ADDRESS OF THE FIRST NUMBER
(B1) =NUMBER OF NUMBERS

00000 01077777 01 0 77777 0 SORT ENTRY SORT
00001 4+000010 4+ 0 P00010 0 UJP **
SWA INIT-1 SAVE FIRST WORD ADDRESS OF DATA BLOCK.
00002 53640000 53 1 40000 2 IAI 2 ADD FIRST WORD ADDRESS TO NUMBER OF ENTRIES TO CALCULATE LAST WORD ADDRESS+1 OF INPUT BLOCK.
00003 15277776 15 0 77776 2 INI -1,2
00004 47200022 47 0 P00022 2 STI INLOOP,2 SET LIMIT OF INLOOP PASSES
00005 15277776 15 0 77776 2 INI -1,2
00006 47200024 47 0 P00024 2 STI OUTLOOP,2 SET LIMIT OF OUTLOOP PASSES
00007 47100026 47 0 P00026 1 STI BSAVE,1 SAVE (B1)
00010 14177777 14 0 77777 1 ENI **+1 ENTER B1 WITH FIRST WORD ADDRESS OF INPUT BLOCK
00011 53100000 53 0 00000 1 INIT TIA 1
00012 15600001 15 1 00001 2 INA 1 SET B2 EQUAL TO (B1)+1
00013 53600000 53 1 00000 2 TAI 2
00014 21100000 21 0 00000 1 LDQ 0,1 LOAD Q WITH FIRST NUMBER FOR THIS PASS.
00015 20200000 20 0 00000 2 LOOP LDA 0,2 LOAD A WITH NEXT NUMBER
00016 03600022 03 1 P00022 2 AUJ,GE INLOOP IS A GE Q YES JUMP TO INLOOP
00017 41200000 41 0 00000 2 STQ 0,2 IF (A) WAS LESS THAN (Q) SWAP THE REGISTER AND ALSO SWAP THEM IN CORE.
00020 40100000 40 0 00000 1 STA 0,1
00021 13000030 13 0 00030 0 SHAQ 24
00022 10277777 10 0 77777 2 INLOOP ISI **+2
00023 01000015 01 0 P00015 0 UJP LOOP
00024 10177777 10 0 77777 1 OUTLOOP ISI **+1
00025 01000011 01 0 P00011 0 UJP INIT
00026 14177777 14 0 77777 1 BSAVE ENI **+1
00027 01400000 01 1 P00000 0 UJP,I SORT
END

NUMBER OF LINES WITH DIAGNOSTICS 0

COMPASS-32 (2.1) SORT

11/21/66 PAGE 1

BSAVE P00026
INIT P00011
INLOOP P00022
LOUP P00015
OUTLOOP P00024
SORT P00000
P00007
P00001
P00004
P00023
P00006
P00027
P00025
P00016

COMPASS-32 (2.1) TYPEOUT

11/21/66 PAGE 1

ENTRY-POINT SYMBOLS
TYPEOUT 00000

LENGTH OF SUBPROGRAM 00010
LENGTH OF COMMON 00000
LENGTH OF DATA 00000

COMPASS-32 (2.1) TYPEOUT

11/21/66 PAGE 2

THIS SUBROUTINE ALLOWS FOR TYPING OUT A VARIABLE NUMBER OF CHARACTERS.THE CALLING SEQUENCE IS.

CALL (A) =FIRST CHARACTER ADDRESS OF OUTPUT BUFFER
(B1) =NUMBER OF CHARACTERS IN OUTPUT BUFFER
(P) =RTJ TYPEOUT
(P+1)=RETURN

00000 01077777 01 0 77777 0 TYPEOUT
00001 77600400 77 1 00400 2
00002 01000001 01 0 P00001 0
00003 53420023 53 1 20023 0

00004 53140000 53 0 40000 1
00005 53420033 53 1 20033 0
00006 77760000 77 1 60000 3
00007 01400000 01 1 P00000 0

ENTRY TYPEOUT
UJP **
PAUS 400B
UJP *-1
TAM 23B

AIA 1

TAM 33B
CTO
UJP,I TYPEOUT
END

IS THE TYPEWRITER BUSY
YES LOOP AND WAIT UNTIL NOT BUSY
TRANSFER FIRST CHARACTER ADDRESS
INTO REGISTER FILE LOCATION 23B.
CALCULATE LAST CHARACTER ADDRESS
+1 BY ADDING (A) AND (B1).
TRANSFER LCA+1 INTO RF 33B
SET TYPE OUT

NUMBER OF LINES WITH DIAGNOSTICS 0

COMPASS-32 (2.1)

TYPEOUT

11/21/66 PAGE 1

TYPEOUT P0000

P00007

COMPASS-32 (2.1)

FLOATF

11/21/66 PAGE 1

ENTRY-POINT SYMBOLS

FLOATF 00000
FLOAT 00000

LENGTH OF SUBPROGRAM 00025
LENGTH OF COMMON 00000
LENGTH OF DATA 00000

COMPASS-32 (2.1)

FLOATF

11/21/66 PAGE 2

THIS SUBPROGRAM CONVERTS AN INTEGER NUMBER INTO FLOATING POINT FORMAT, UPON ENTRY THE ADDRESS OF THE INTEGER NUMBER IS LOCATED IN THE LOWER 15 BITS OF P+1.

CALL (P) = RTJ FLOATF
(P+1)= ADDRESS OF THE INTEGER NUMBER
(P+2)= RETURN

OUTPUT (AQ)= FLOATING POINT EQUIVALENT OF THE INTEGER NUMBER

00000	01077777	01 0	77777 0	FLOATF	ENTRY	FLOAT,FLOATF	
			00000	FLOAT	UJP	**	
00001	47100021	47 0	P00021 1		EQV	FLOATF	EQUATE FLOAT TO FLOATF
00002	54100000	54 0	P00000 1		STI	FOUT,1	SAVE B1
00003	20500000	20 1	00000 1		LDI	FLOATF,1	LOAD B1 WITH ADDRESS OF INTEGER
00004	15100001	15 0	00001 1		LOA#I	0+1	LOAD A WITH INTEGER NUMBER
00005	47000022	47 0	P00022 0		INI	1+1	INCREASE AND SET RETURN ADDRESS
00006	13077747	13 0	77747 0		STI	EXIT	TO P+2 OF CALLING ROUTINE
00007	03400021	03 1	P00021 0		SHAQ	+24	
00010	40000023	40 0	P00023 0		AQJ#EQ	FOUT	DID THE INPUT NUMBER=0 YES EXIT
00011	13502057	13 1	02057 1		STA	MASKO	SAVE SIGN OF INPUT NUMBER
					SCAQ	2057B,1	NORMALIZE THE INTEGER BY USING
							THE SCALE INSTRUCTION
00012	13077764	13 0	77764 0		SHAQ	-11	MERGE THE EXPONENT AND BIAS
							(CONTENTS OF B1) WITH THE
00013	41000024	41 0	P00024 0		STQ	TEMP	MANTISSA
00014	13077763	13 0	77763 0		SHAQ	-12	
00015	53100000	53 0	00000 1		IIA	1	
00016	36000023	36 0	P00023,0		SCA	MASKO	COMPLEMENT EXPONENT AND BIAS IF
							THE INPUT NUMBER WAS NEGATIVE.
00017	13000014	13 0	00014 0		SHAQ	12	
00020	21000024	21 0	P00024 0		LDQ	TEMP	
00021	14177777	14 0	77777 1	FOUT	ENI	**+1	
00022	01077777	01 0	77777 0	EXIT	UJP	**	
00023	00000000			MASKO	OCT	0	
00024				TEMP	BSS	1	
					END		

NUMBER OF LINES WITH DIAGNOSTICS 0

COMPASS-32 (2.1)

FLOATF

11/21/66 PAGE 1

EXIT	P00022	P00005	
FLOATF	P00000	P00000	P00002
FOUT	P00021	P00001	P00007
MASKO	P00023	P00010	P00016
TEMP	P00024	P00013	P00020

SYMBOLS NOT REFERENCED

*FLOAT P00000

ENTRY-POINT SYMBOLS
CRDTP 00000

LENGTH OF SUBPROGRAM 00121
LENGTH OF COMMON 00000
LENGTH OF DATA 00000

THIS SUBPROGRAM COPIES A FILE OF CARDS FROM A CARD READER (UNIT 0, CONTROLLER 3, ON CHANNEL 0) TO A TAPE UNIT (UNIT 1, CONTROLLER 0 ON CHANNEL 1). THIS SUBPROGRAM DOES NOT CHECK FOR CARD OR TAPE ERRORS. THE MODE OF EACH TAPE RECORD IS DETERMINED BY THE MODE OF THE CARD RECORD. THE DENSITY OF THE OUTPUT TAPE WILL BE THE DENSITY SET BY THE OPERATOR.

				ENTRY	CRDTP	
00000	01077777	01 0	77777 0	CRDTP UJP	**	
00001	77003000	77 0	03000 0	CUN	3000H,0	CONNECT CARD READER ON CHANNEL 0
00002	01000001	01 0	P0001 0	UJP	*-1	
00003	77010001	77 0	10001 0	CUN	1B,1	CONNECT OUTPUT TAPE ON CHANNEL
00004	01000003	01 0	P0003 0	UJP	*-1	
00005	77110010	77 0	10010 1	SEL	10B,1	REWIND OUTPUT TAPE
00006	01000005	01 0	P0005 0	UJP	*-1	
00007	77200000	77 0	00000 2	INPUT COPY	0	IS THE CARD READER READY AND
00010	17600003	17 1	00003 2	ANA	3B	NOT BUSY-YES INITIATE INPUT-
00011	04600001	04 1	00001 2	ASE	1B	NO LOOP UNTIL IT BECOMES READY
00012	01000007	01 0	P0007 0	UJP	INPUT	AND NOT BUSY.
00013	74000121	74 0	P00121 0	INP#	0,BUFF,BUFF**0	INITIATE READING A BINARY OR
00014	00000051	00 0	P00051 0			
00015	01000013	01 0	P00013 0	UJP	*-2	RCU CARD,
00016	77200002	77 0	00002 2	EKS	2H,0	WAIT UNTIL THE CARD READER
00017	01000016	01 0	P00016 0	UJP	*-1	BECOMES NOT BUSY.
00020	77200004	77 0	00004 2	EKS	4H,0	WAS THE CARD JUST READ A BINARY
00021	01000004	01 0	P0004 0	JJP	BINARY	CARD YES JUMP TO BINARY.
00022	77200010	77 0	00010 2	EKS	10H,0	WAS A FILE CARD READ (ROW 7 AND
00023	01000004	01 0	P0004 0	UJP	EOF	R PUNCHED) YES JUMP TO EOF
00024	14600002	14 1	00002 2	ENA	2H	
00025	42000133	42 0	P00026 3	SACH	SELMODE*3	SET RCU MODE FOR OUTPUT TAPE.
00026	77110000	77 0	10000 1	SELMODE SEL	0+1	
00027	01000026	01 0	P00026 0	UJP	*-1	
00030	53020000	53 0	20000 0	IMA	0	FETCH LAST CHARACTER ADDRESS+1
00031	12077775	12 0	77775 0	SMA	-2	AND CHANGE IT TO LAST WORD
						ADDRESS+1
00032	44000033	44 0	P00033 0	SWA	OUTPUT	STORE ADDR IN OUTPUT INSTRUCTION
00033	76077777	76 0	77777 0	OUTPUT OUTW	1,BUFF,**	INITIATE WRITING OUTPUT BUFFER
00034	10000051	10 0	P00051 0			
00035	01000033	01 0	P00033 0	UJP	*-2	
00036	77210002	77 0	10002 2	EKS	2H,1	WAIT UNTIL THE OUTPUT TAPE
00037	01000036	01 0	P00036 0	UJP	*-1	BECOMES NOT BUSY
00040	01000007	01 0	P00007 0	UJP	INPUT	

00041	14600001	14 1	00001 2	BINARY ENA	1	SET BINARY MODE FOR OUTPUT TAPE.
00042	42000133	42 0	P00026 3	SACH	SELMODE*3	
00043	01000026	01 0	P00026 0	JJP	SELMODE	

00044	77110015	77 0	10015 1	EOF SEL	15H,1	WRITE EOF ON OUTPUT TAPE
00045	01000004	01 0	P00004 0	UJP	*-1	
00046	77110011	77 0	10011 1	SEL	11B,1	INLOAD OUTPUT TAPE (UNIT 1)

COMPASS-32 (2.1) CRDTP 11/21/66 PAGE 3
00047 01000046 01 0 P00046 0 UJP *-1
00050 01400000 01 1 P00000 0 UJP+I CRDTP EXIT TO SCOPE
00051 BUFF BSS 40
END CRDTP
NUMBER OF LINES WITH DIAGNOSTICS 0

COMPASS-32 (2.1) CRDTP 11/21/66 PAGE 1

BINARY	P00041	P00021		
BUFF	P00051	P00013	P00013	P00033
CRDTP	P00000	P00050		
EOF	P00044	P00023		
INPUT	P00007	P00012	P00040	
OUTPUT	P00033	P00032		
SELMODE	P00026	P00025	P00042	P00043

ENTRY-POINT SYMBOLS
MTMT 00000

LENGTH OF SUBPROGRAM 00256
LENGTH OF COMMON 00000
LENGTH OF DATA 00000

THIS SUBPROGRAM COPIES A VARIABLE SIZE RECORD INPUT TAPE (UNIT 1, CONTROLLER 0, ON CHANNEL 0) TO ANOTHER TAPE (UNIT 2, CONTROLLER 0, ON CHANNEL 1). THE DENSITY OF THE OUTPUT TAPE WILL BE THE DENSITY SET BY THE OPERATOR. THIS SUBPROGRAM DOES NOT CHECK FOR TAPE ERRORS, PARITY IS CHECKED ONLY TO INSURE THAT THE MODE OF THE INPUT TAPE IS THE SAME AS THE MODE SELECTED.

	ENTRY	MTMT	
00000 01077777 01 0 77777 0	MTMT UJP	**	
00001 77000001 77 0 00001 0	CON	18,0	CONNECT INPUT TAPE (UNIT 1)
00002 01000001 01 0 P00001 0	UJP	**1	ON CHANNEL 0
00003 77010002 77 0 10002 0	CON	28,1	CONNECT OUTPUT TAPE (UNIT 2)
00004 01000003 01 0 P00003 0	UJP	**1	ON CHANNEL 1
00005 77100010 77 0 00010 1	SEL	108,0	REWIND INPUT TAPE
00006 01000005 01 0 P00005 0	UJP	**1	
00007 77110010 77 0 10010 1	SEL	108,1	REWIND OUTPUT TAPE
00010 01000007 01 0 P00007 0	UJP	**1	
00011 77100001 77 0 00001 1	MODE SEL	18,0	ASSUME AND SET INPUT TAPE TO
00012 01000011 01 0 P00011 0	UJP	**1	BINARY MODE (MODE MAY BE CHANGED LATER).
00013 73001270 73 0 P00256 0	INPC	0, INRUFF, INBUFF+500	READ INPUT RECORD FROM TAPE UNIT
00014 00000304 00 0 P00061 0	UJP	**2	1 (MAX RECORD SIZE IS 500
00015 01000013 01 0 P00013 0	UJP	**1	CHARACTERS)
00016 77200002 77 0 00002 2	EKS	28,0	WAIT UNTIL INPUT IS COMPLETE
00017 01000016 01 0 P00016 0	UJP	**1	
00020 77200010 77 0 00010 2	EKS	108,0	WAS AN EOF READ
00021 01000007 01 0 P00007 0	UJP	EOF	YES JUMP TO EOF
00022 77202000 77 0 02000 2	EKS	20008,0	WAS THERE A PARITY ERROR
00023 01000000 01 0 P00000 0	UJP	SETMODE	YES JUMP TO CHANGE MODES
00024 53020000 53 0 20000 0	TMA	0	FETCH LAST CHARACTER ADDRESS OF
00025 46000032 46 0 P00032 0	SCHA	OUTPUT	INPUT RECORD+1
00026 22000007 22 0 P00011 3	LACH	MODE*3	STORE CHARACTER ADDRESS IN OUT-
00027 42000143 42 0 P00030 3	SACH	OUTMODE*3	PUT INSTRUCTION.
00030 77110000 77 0 10000 1	SEL	0,1	FETCH MODE OF INPUT RECORD AND
00031 01000030 01 0 P00030 0	UJP	**1	STORE TO SET OUTPUT MODE THE
00032 75377777 75 0 77777 3	OUTPUT OUTC	1, INBUFF,**	SAME AS INPUT
00033 10000304 10 0 P00061 0	UJP	**2	INITIATE WRITING OUTPUT RECORD
00034 01000032 01 0 P00032 0	UJP	**1	
00035 77210002 77 0 10002 2	EKS	28,1	WAIT UNTIL OUTPUT IS COMPLETE
00036 01000035 01 0 P00035 0	UJP	**1	
00037 01000011 01 0 P00011 0	UJP	MODE	LOOP TO READ NEW INPUT RECORD
00040 77770000 77 1 70000 3	SETMODE UCS		*****
00041 77100012 77 0 00012 1	SEL	128,0	IF PARITY ERROR STOP SO THE

COMPASS-32 (2.1) MTMT

11/21/66 PAGE 3

```

00042 01000041 01 0 P00041 0 UJP *-1
00043 22000047 22 0 P00011 3 LACH MODE+3
00044 16600003 16 1 00003 2 XOL 3
00045 42000047 42 0 P00011 3 SACH MODE+3
00046 01000011 01 0 P00011 0 UJP MODE
*****
00047 77110015 77 0 10015 1 EOF SEL 15B,1
00050 01000047 01 0 P00047 0 UJP *-1
00051 77700000 77 1 00000 3 SLS
00052 00100054 00 0 P00054 1 SJ1 EXIT
00053 01000011 01 0 P00011 0 JJP MODE
*****
00054 77100011 77 0 00011 1 EXIT SEL 11B,0
00055 01000054 01 0 P00054 0 UJP *-1
00056 77110011 77 0 10011 1 SEL 11B,1
00057 01000056 01 0 P00056 0 UJP *-1
00060 01400000 01 1 P00000 0 UJP,I MTMT
00061 INBUFF BSS,C 500
END MTMT

```

```

FETCH ASSUMED MODE AND
CHANGE IT
SET NEW MODE IN SEL INSTRUCTION
AND TRY AGAIN.
WRITE EOF ON OUTPUT TAPE
STOP IF SELECTIVE STOP SWITCH
SET ON CONSOLE.
EXIT AND UNLOAD TAPES IF JUMP
KEY 1 IS SET ON CONSOLE.
IF NOT JUMP TO READ INPUT RECORD
UNLOAD INPUT TAPE
UNLOAD OUTPUT TAPE
EXIT TO SCOPE

```

NUMBER OF LINES WITH DIAGNOSTICS 0

COMPASS-32 (2.1) MTMT

11/21/66 PAGE 1

```

EOF P00047 P00021
EXIT P00054 P00052
INBUFF P00061 P00013 P00013 P00032
MODE P00011 P00026 P00037 P00043 P00045
P00046 P00053
MTMT P00060
OUTMODE P00030 P00027
OUTPUT P00032 P00025
SETMODE P00040 P00023

```

ENTRY-POINT SYMBOLS
CVTBCDB 00000

LENGTH OF SUBPROGRAM 00017
LENGTH OF COMMON 00000
LENGTH OF DATA 00000

THIS SUBPROGRAM CONVERTS A POSITIVE FOUR CHARACTER BCD NUMBER
TO A 24 BIT BINARY NUMBER.

CALL (P) =RTJ CVTBCDB
(P+U)=RETURN
(A) =THE FOUR CHARACTER BCD NUMBER

OUTPUT (A)=24 BIT BINARY NUMBER

00000	01077777	01 0	77777 0	CVTBCDB	ENTRY	CVTBCDB	
00001	40000015	40 0	P00015 0		UJP	**	
00002	47200013	47 0	P00013 2		STA	TEMP	SAVE INPUT PAR.
00003	14200000	14 0	00000 2		STI	B2SAVE,2	SAVE (B2)
00004	22000064	22 0	P00015 0		ENI	0,2	
					LACH	TEMP	LOAD A WITH MOST SIGNIFICANT CHARACTER.
00005	50000016	50 0	P00016 0	LOOP	MVA	=012	MULTIPLY (A) BY AN OCTAL 12
00006	15200001	15 0	00001 2		INI	1,2	
00007	23400064	23 1	P00015 0		LQCH	TEMP,2	LOAD Q WITH NEXT SIGNIFICANT CHARACTER.
00010	53040000	53 0	40000 0		AQA		
00011	04200003	04 0	00003 2		ISE	3,2	IS B2 EQUAL TO 3 YES EXIT
00012	01000005	01 0	P00005 0		UJP	LOOP	NO JUMP TO LOOP
00013	14277777	14 0	77777 2	B2SAVE	ENI	**2	RESTORE B2
00014	01400000	01 1	P00000 0		UJP+I	CVTBCDB	
00015				TEMP	BSS	1	
					END		

LITERALS
00016 00000012

NUMBER OF LINES WITH DIAGNOSTICS 0

COMPASS-32 (2.1)

CVT8CDB

11/21/66 PAGE 1

B2SAVE	P00013		P00002		
CVT8CDB	P00000		P00014		
LOOP	P00005		P00012		
TEMP	P00015		P00001	P00004	P00007
LITERAL	P00016	00000012	P00005		

COMPASS-32 (2.1)

CVT8BCD

11/21/66 PAGE 1

ENTRY-POINT SYMBOLS
CVT8BCD 00000

LENGTH OF SUBPROGRAM 00033
LENGTH OF COMMON 00000
LENGTH OF DATA 00000

COMPASS-32 (2.1)

CVTBBCD

11/21/66 PAGE 2

THIS SUBPROGRAM CONVERTS A 24 BIT BINARY NUMBER TO A EIGHT CHARACTER BCD NUMBER WITH BLANKS INSERTED TO THE LEFT OF THE MOST SIGNIFICANT DIGIT.

CALL (P) =RTJ CVTBBCD
(P+1)=RETURN
(A) = 24 BIT BINARY NUMBER

OUTPUT (AQ) = EIGHT CHARACTER BCD NUMBER

00000 01077777 01 0 77777 0 CVTBBCD ENTRY CVTBBCD
00001 47100025 47 0 P00025 1 UJP **
00002 21000027 21 0 P00027 0 STI B1SAVE+1 SAVE (B1)
00003 41000030 41 0 P00030 0 LDQ BLANKS
00004 41000031 41 0 P00031 0 STQ BCDANS SET OUTPUT AREA TO BLANKS
00005 43000113 43 0 P00022 3 SQCM SETSIGN+3 SET ASSUMED SIGN TO POSITIVE
00006 03200012 03 0 P00012 2 AZJ+GE HYPASS IS INPUT NUMBER POSITIVE YES JUMP
00007 14100040 14 0 00040 1 ENI 40B+1 NO SET SIGN
00010 47100022 47 0 P00022 1 STI SETSIGN+1
00011 16477777 16 1 77777 0 XOA+S =0 COMPLEMENT THE 24 BIT NUMBER
00012 14100007 14 0 00007 1 BYPASS ENI 7+1 SET MAX TIMES THROUGH LOOP
00013 13077747 13 0 77747 0 LOOP SMAW =012 SHIFT FOR 48 BIT DIVIDEND
00014 51000032 51 0 P00032 0 DVA =012 DIVIDE BY OCTAL 12
00015 43400140 43 1 P00030 0 SQCM BCDANS+1 STORE NEXT SIGNIFICANT DIGIT
00016 02500020 02 1 P00020 1 IJD **2+1
00017 01000022 01 0 P00022 0 UJP SETSIGN
00020 04400000 04 1 00000 0 ASE+S 0 IS THE QUOTIENT ZERO-YES SKIP
00021 01000013 01 0 P00013 0 UJP LOOP
00022 14700060 14 1 00060 3 SETSIGN ENQ 60B PLACE SIGN JUST TO THE LEFT OF
00023 43400140 43 1 P00030 0 SQCM BCDANS+1 MOST SIGNIFICANT DIGIT.
00024 25000030 25 0 P00030 0 LUAW BCDANS
00025 14177777 14 0 77777 1 B1SAVE ENI **+1 RESTORE (B1)
00026 01400000 01 1 P00000 0 UJP+I CVTBBCD EXIT
00027 60606060 BLANKS BCD 1,
00030 BCDANS BSS 2
END

LITERALS
00032 00000012

NUMBER OF LINES WITH DIAGNOSTICS 0

COMPASS-32 (2.1)

CVTBBCD

11/21/66 PAGE 1

B1SAVE P00025 P00001
BCDANS P00030 P00003 P00004 P00015 P00023
P00024
P00002
P00006
P00026
P00021
P00005 P00010 P00017
P00014
BLANKS P00027
BYPASS P00012
CVTBBCD P00000
LOOP P00013
SETSIGN P00022
LITERAL P00032 00000012 P00014

SEQUENCE,002
JOB,****
COMPASS,L,X

COMPASS-32 (2,1)

IDC

11/21/66 PAGE 1

EXTERNAL SYMBOLS
CID

ENTRY-POINT SYMBOLS
IDC 00000

LENGTH OF SUBPROGRAM 00032
LENGTH OF COMMON 00050
LENGTH OF DATA 00000

COMPASS-32 (2.1)

IDC

11/21/66 PAGE 2

00000	01077777	01	0	77777	0	IDC	ENTRY	IDC
00001	00777777	00	1	X77777	3	READ	EXT	CIO
00002	01000074	01	0	00074	0		UJP	**
00003	01000001	01	0	P00001	0		RTJ	CIO
00004	00000000	00	0	C00000	0		01	60
00005	00000050	00	0	00050	0		UJP	**2
00006	00700001	00	1	X00001	3			INBUFF
00007	13000074	13	0	00074	0		RTJ	40
00010	03200006	03	0	P00006	2		13	CIO
00011	17700010	17	1	00010	3		AZJ,GE	60
00012	04700000	04	1	00000	3		ANQ	10B
00013	01000031	01	0	P00031	0		QSE	0
00014	22000000	22	0	C00000	0		UJP	EOF
00015	04600041	04	1	00041	2		LACH	INBUFF
00016	01000001	01	0	P00001	0		ASE	41B
00017	00700006	00	1	X00006	3		UJP	READ
00020	02000073	02	0	00073	0		RTJ	CIO
00021	01000017	01	0	P00017	0		02	59
00022	00000002	00	0	C00002	0		UJP	**2
00023	00000002	00	0	00002	0			INBUFF*2
00024	00700017	00	1	X00017	3		RTJ	2
00025	13000073	13	0	00073	0		13	CIO
00026	03200024	03	0	P00024	2		AZJ,GE	59
00027	77700000	77	1	00000	3		SLS	**2
00030	01000001	01	0	P00001	0		UJP	READ
00031	01400000	01	1	P00000	0	EOF	UJP,I	IDC
00000							COMMON	
						INBUFF	BSS	40
							END	IDC

NUMBER OF LINES WITH DIAGNOSTICS 0

LOAD=56
 SNAP,(IDC)15,C0,C47,0,0P
 R,N,10

SUBP
15547 IODRAIN 15611 IOPACK 16245 SNAPSHOT 17705 IDC

ENTR
15557 IODRAIN 16134 READINP 16012 READLUN 15611 WRITECTO 16200 WRITEACC 16172 WRITEPUN
15621 WRITELUN 16163 WRITEOUT 16307 PROGDUMP 16266 FORTDUMP 16245 SNAPSHOT 17705 IDC
00745 SEL 02130 UST 02170 CST 00060 CIT 02241 RMT 02201 AET
02402 HDCKSUM 02305 HDCKF1 02015 BCDBOXS 02015 FDPBOXS 00107 CIO 02453 START2
02737 LOADER 02261 ACCOUNTS 02303 MEMORY 02015 ABNORMAL

COMM
03206 03255

DATA
NONE

EXTA
NONE

DP LOC 17722 A 00000060 Q 00000000 B1 00000 B2 17705 B3 00000 IMR 0003
OCTAL MEMORY
03206 00000 60606060 60605125 21436044 64734564 73432144 22242179 31475131 44257331
03216 00010 45657360 31626321 51734547 65734473 44222151 73443021 63734447 73442221
03226 00020 51477344 30216347 60606060 60606060 00702015 00702015 00702015 00702015
GAP

* END *

DP LOC 17722 A 00000060 Q 00000000 B1 00000 B2 17705 B3 00000 IMR 0003
OCTAL MEMORY
03206 00000 60606060 60017345 64212273 60606060 60606060 60606060 60606060 60606060
GAP
03226 00020 60606060 60606060 60606060 60606060 00702015 00702015 00702015 00702015
GAP

* END *

SEQUENCE#003
 JOM,JC#482,15
 OCC#X16
 OCC (IDC)27,01000000X
 OCC#X,20000024 (IDC),44000003X,44000010X,00777777
 OCC#+,02000073,01000003X,00000015X,00000001
 OCC#+,00777777,13000073,03200010X,77700000
 OCC#+,01000030 (IDC),77606060
 RUN#10

SUBP
 17635 IDC

ENTR	17635 IDC	00745 SEL	02130 UST	02170 CST	00060 CIT	02241 RHT
	02201 AET	02402 ROCKSUM	02305 HDCKFJ	02015 RCDROXS	02015 FDPDQXS	00107 CIO
	02453 START2	02737 LOADER	02261 ACCOUNTS	02303 MEMORY	02015 ABNORMAL	

COMM
 NONE

DATA
 NONE

EXTA
 17617

LOC 00043 A 00002015 Q 40004003 R1 00014 R2 17635 R3 00000 TC 0116 IA 17645 TC 00 TA 02015 IMR 0003

REGISTER FILE

00	00077460	20413534	00770077	00770077	00770077	00770077	00770077	00770077	00770077
10	04077574	26013534	00770077	00770077	00770077	00770077	00770077	00770077	00770077
20	60445400	00045254	26571102	00007725	47030634	00002015	40004003	00000014	00000014
30	71045400	72045404	23575770	00007725	00770077	00117645	00000000	00300007	00000000
40	00006243	00002403	00077776	00003320	00000247	00000000	00017302	00000000	00000000
50	01007331	00003274	00000000	00000000	00000067	00000005	00000000	00002137	00000000
60	00000006	00000070	00000000	60606060	00000006	00003271	00004171	00000013	00000013
70	00000002	04077776	00000003	00000050	00000144	00000013	60606060	00000000	00000000

MEMORY

GAP									
03200	60606060	60606060	60606060	60606060	60606060	60606060	00702015	00702015	
GAP									
17610	00702015	00702015	00702015	00702015	00702015	00702015	00702015	0017661	
17620	44017622	44017627	00777777	02000073	01017622	00017634	00000001	00777777	
17630	13000073	03217627	77700000	01017665	77606060	01002037	00700107	01000074	
17640	01017636	00017667	00000050	00700107	13000074	03217643	17700010	04700000	
17650	01017666	22077334	04600041	01017636	00700107	02000073	01017654	00017671	
17660	00000002	00700107	13000073	03217661	01017617	01017636	01417635	00054010	
17670	20102001	11002001	44002020	11022100	11021004	00000000	00000000	00000000	

GAP									
17730	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00702015	

FN0

SEQ ERR
SEQ ERR

SEQUENCE:004

ADDITIONAL EXERCISES (Assignment Sheets)

3200 Computer Characteristics	B1 - B2
Load Instructions	B3
Store Instructions	B4
Arithmetic, Fixed Point	B5 - B6
Register Operations Without Storage Reference	B7 - B8
Stop and Jump Instructions	B9 - B11
Inter-Register Transfer	B12 - B13
Search and Move Operations	B14 - B17
Storage Tests	B18 - B19
BCD Operations	B20 - B22

3200 COMPUTER CHARACTERISTICS

1. For each of the following word addresses and character positions, provide the equivalent character address and the storage module in which it is contained.

<u>Word Address</u>	<u>Character Position</u>	<u>Character Address</u>	<u>Storage Module</u>
a) 00027	#0		
b) 06313	#1		
c) 15346	#2		
d) 20476	#3		

2. For each of the following character addresses provide the equivalent word address, character position and storage module.

<u>Character Address</u>	<u>Word Address</u>	<u>Character Position</u>	<u>Storage Module</u>
a) 300630			
b) 152063			
c) 325076			
d) 150762			

3. If Character Address 340556 were specified, what character address would actually be referenced in an 8K system?
4. If Character Address 160560 were specified, what character address would actually be referenced in a 16K system?
5. The 3200 system can consist of a maximum of _____ 12-bit data channels.

6. A 24-bit data channel may be used in place of _____ 12-bit data channel(s). It will be programmed as channel number _____.
7. A 3200 data channel may control a maximum of _____ peripheral equipment controllers.
8. Give a brief description of the four (4) types of Processors.
9. What memory locations are always permanently protected and why?
10. What address or block of addresses are protected by the Storage Protect Switches below?

111 111 110 NNN NNN = _____
11. Describe what happens when an instruction attempts to write into a Protected address?
12. Give the function of each location within the register file.

LOAD INSTRUCTIONS

GIVEN: (A) = 00000000

(B¹) = 2

(Q) = 00007777

(B²) = 2(B³) = 0

TEMP = Memory Location 13314

TEMP	27	31	33	17
+1	40	51	33	15
+2	20	41	33	14
+3	23	14	26	15
+4	00	71	33	16

What are the contents of the indicated registers after the execution of each instruction? Assume initial condition above for each problem.

- | | | | |
|-----|----------|----------|---------------------|
| 1. | LDA | TEMP | (A) = |
| 2. | LDQ | TEMP+1 | (Q) = |
| 3. | LACH | TEMP,1 | (A) = |
| 4. | LDA,I | TEMP,2 | (A) = |
| 5. | LDL | TEMP,2 | (A) = |
| 6. | LGAQ | TEMP+1 | (A) =
(Q) = |
| 7. | 20113315 | | (A) = |
| 8. | 22055462 | | (A) = |
| 9. | 54113316 | | (B ¹) = |
| 10. | LDI,I | TEMP+4,1 | (B ¹) = |
| 11. | LQCH | TEMP+7 | (Q) = |
| 12. | LACH | TEMP+2,1 | (A) = |

STORE INSTRUCTIONS

GIVEN: (A) = 24130120

 $(B^1) = -1$

(Q) = 13000003

 $(B^2) = 10$ $(B^3) = 1$

TEMP = Memory Location 33444

TEMP1 = Memory Location 33446

TEMP	01	33	34	43
	77	53	34	44
TEMP1	66	73	34	45
	65	03	34	46
	44	55	66	77

What are the contents of the indicated registers or memory locations after the execution of each instruction? (Assume original settings at beginning of each instruction set.)

1. STA TEMP+4 (A) =
2. STQ,I TEMP (TEMP) =
3. SACH TEMP,2 (TEMP+2) =
4. SWA TEMP+3 (TEMP+3) =
5. STAQ,I TEMP Where are A & Q stored =
6. SCHA TEMP+1 (TEMP+1) =
7. STI TEMP+4 (TEMP+4) =
8. SQCH TEMP1,1 (Modified location) =
9. STI TEMP1,3 (TEMP1) =
10. SWA,I TEMP1+1 (TEMP1) =

ARITHMETIC, FIXED POINT

The following problem set is to familiarize the student with the arithmetic instructions.

1. MC, set P to "GO" and start.

GO	ENA,S	604B
	ENQ	77777B
	MUA	INTEGER
	HLT	GO
INTEGER	HLT	4

$(A)_f =$

$(Q)_f =$

2. MC, set P to "GO" and start.

GO	ENA,S	0
	ENQ,S	600B
	DVA	DIVISOR
	HLT	GO
DIVISOR	OCT	-7

$(A)_f =$

$(Q)_f =$

3. MC, set P to "GO" and start.

GO	LDA	NUMB
	ADA	NUMB+1
	ANA	77777B
	RAD	NUMB+2
	HLT	
NUMB	OCT	-10,-2000,1

$(A)_f =$

4. MC, set P to "GO" and start.

GO	ENI	7,1
	ENA,S	6
	INA,S	1
	SBA	MINUS
	HLT	
MINUS	OCT	10

$(A)_f =$

5. What are the contents of A register when program halts?

	ENI	-6,1
	LDA	TEMP
	SHA	5,1
	HLT	
TEMP	OCT	40123400
	(A) =	

REGISTER OPERATIONS WITHOUT STORAGE REFERENCE

For each of the following short routines, indicate the final contents of designated register. Assume that a M.C. was performed prior to starting each program.

1.	<u>M.L.</u>	<u>CONTENTS</u>
	00000	ENI 40000B,1
	00001	INI 100B,1
	00002	XOI 7677B,1
	00003	ANI 7777B,1
	00004	HLT

$(B^1)_f =$

2.	00000	ENA,S 40000B
	00001	XOA,S 40000B
	00002	INA,S 50000B
	00003	ANA,S 40000B
	00004	HLT

$(A)_f =$

3.	00000	ENQ 60000B
	00001	XOQ 70000B
	00002	INQ 77777B
	00003	ANQ 77777B
	00004	HLT

$(Q)_f =$

4.	00000	ECHA 2003B
	00001	ENQ 0
	00002	SHA -2
	00003	SHAQ 24
	00004	SHQ 1
	00005	SHAQ 35430B
	00006	HLT

$(A)_f =$

$(Q)_f =$

GIVEN: (A) = 40372156

 $(B^1) = 77777$

(Q) = 35642761

 $(B^2) = 41745$ $(B^3) = 72156$

Regarding each of the following instructions as separate problems, indicate which will RNI at $P + 1$ by placing a check (✓) in the space provided.

1. _____ISE 0,1
2. _____ISE 41745B,3
3. _____ISE 43745B,2
4. _____ASE,S 27614B
5. _____ASE 72156B
6. _____QSE 47321B
7. _____QSE,S 34763B
8. _____ISE 72156B,3
9. _____ISE 0,0
10. _____ASE,S 72156B
11. _____ISG 77777B,1
12. _____ISG 27317B,2
13. _____ISG 77345B,3
14. _____ASG,S 73452B
15. _____QSG,S 04210B
16. _____ASG 72156B
17. _____QSG 35016B
18. _____ISG 72156B,3
19. _____ISG 17774B,0
20. _____ISG 37432B,2

1. MC, set P to 06000 and start.

<u>M.L.</u>	<u>CONTENTS</u>	
06000	LDA	6004B
06001	INA,S	-1
06002	AZJ,NE	6005B
06003	HLT	
06004	OO	0
06005	INA,S	2
06006	UJP	6001B

$(A)_f =$

2. MC, set P to 05120 and start.

<u>M.L.</u>	<u>CONTENTS</u>	
05120	ENI	77B,1
05121	INA,S	10000B
05122	IJD	5121B,1
05123	INQ,S	10000B
05124	ISI	77B,1
05125	UJP	5123B
05126	ANA	0
05127	HLT	

$(A)_f =$

$(Q)_f =$

$(B^1)_f =$

3. MC, set P to 07000 and start.

<u>M.L.</u>	<u>CONTENTS</u>	
07000	IJI	7007B,1
07001	UJP	7006B,3
07002	UJP	7004B
07003	IJI	7002B,2
07004	UJP	7005B,1
07005	UJP	7002B
07006	RTJ	7002B
07007	HLT	

$(P)_f =$

$(07002) =$

4. What does this program do?

	ENI	4,1
AGAIN	LDA	TEMP,1
	RTJ	ROUTINE
	UJP	*+4
	*RAD	TALLY
	IJD	AGAIN,1
	HLT	**
	RAD	COUNT
	UJP	*-3
ROUTINE	UJP	**,2
	ENI	0,2
	ASG	100B
	INI	1,2
	ENA	1
	UJP,I	ROUTINE
TEMP	OCT	100
	OCT	21
	OCT	77
	OCT	101
	OCT	10
TALLY	OCT	0
COUNT	OCT	0

*The RAD instruction must have been discussed before the above problem can be done.

5. MC, set P to "C.0" and start. The selective stop switch and jump switches 1 and 6 are set.

<u>M.L.</u>	<u>CONTENTS</u>	
C.0	UJP	C.6
C.1	HLT	C.0
C.2	SJ1	C.1
	SLS	
C.4	SJ3	C.10
	SJ6	C.2
C.6	SJ2	C.10
	UJP	C.4
C.10	SLS	

$(P)_f =$

If the machine were restarted from where it stopped, what memory location would it go to?

INTER-REGISTER TRANSFER

1. What are the contents of the A register and Q register when program is executed?

<u>M.L.</u>			
00000		LDQ	TEMP
00001		ENA	GET
00002		SWA	*+1
00003		LDA	**
00004		AQA	
00005		STA	*+2
00006		ENQ	-6
00007		HLT	**
00010		HLT	**
00011	GET	ENI	0,0
00012	TEMP	OCT	02677771

(A) =

(Q) =

2. MC, set P to GO and start.

GO	ENA	10
	TAI	3
	ENQ	10
	INI	-1,3
	HLT	

(A)_f =

(Q)_f =

(B³)_f =

3. Write the octal coding that will be generated by the following COMPASS coding.

	<u>COMPASS</u>		<u>OCTAL</u>
1)	AQA	=	
2)	INA	3	=
3)	TIA	2	=
4)	TQM	22B	=
5)	AIA	1	=
6)	TIM	20B,2	=
7)	TAM	77B	=
8)	TMI	60B,3	=
9)	TMA	27B	=
10)	TMQ	55B	=
11)	TAI	1	=

4. MC, set P to BEGIN and start.

BEGIN	ENI	-100B,1
	ENI	-200B,2
	ENI	300B,3
	ENQ,S	-0
	TQM	24B
	TIA	1
	SHAQ	-24
	TIA	3
	AQA	
	TAM	25B
AIA	AIA	2
	TAM	26B
	ENA,S	-1
	IAI	1
	TIM	27B,2
	TMQ	24B
	ENA	0
	AQA	
	HLT	

$(A)_f =$

$(Q)_f =$

$(B^1)_f =$

$(V24) =$

$(V25) =$

$(V26) =$

$(V27) =$

5. MC, set P to BEGIN and start.

BEGIN	LDAQ	NUMB
	MUAQ	INTEGER
	DVAQ	DIVISOR
	STAQ	ANSWER
	ELQ	
	EUA	
	STAQ	ANSWER+2
	HLT	
NUMB	OCT	-0,-100
INTEGER	OCT	0,4
DIVISOR	OCT	0,2
ANSWER	OCT	0,0,0,0

$(ANSWER) =$

$(ANSWER+1) =$

$(ANSWER+2) =$

$(ANSWER+3) =$

SEARCH AND MOVE OPERATIONS

1.

	IDENT	MAIN
	ENTRY	START
	EXT	DELBLANK
START	.	
	.	
	.	
	ENA	DATACARD
	RTJ	DELBLANK
	.	
	.	
	.	
DATACARD	BSS	20
	END	START
	IDENT	DELETE
	ENTRY	DELBLANK
BELBLANK	UJP	**
	STI	SAVEB2,2
	ENI	0,2
	SHA	2
	SCHA	FCA
	SCHA	SRC+1
	INA,S	80
	SCHA	SRC
SRC	SRCN	60B,**,**
	UJP	*-2
	PAUS	4000B
	UJP	*-1
	TMA	30B
	LPA	MASK
	SHAQ	24
	TMA	20B
	LPA	MASK
	AQJ,EQ	EXIT
	SCHA	*+3
INCRE	INA,S	1
	SCHA	SCR+1
	LACH	**
FCA	SACH	**,2
	INI	1,2
	UJP	SRC
EXIT	LDA	*-3
	LPA	MASK
	AIA	2
SAVEB2	ENI	**,2
	UJP,I	DELBLANK
MASK	OCT	00377777
	END	
	FINIS	

Assume program MAIN had previously read in one card image into the area labeled DATACARD before it transfers control to a commonly used subprogram called DELETE.

- A. Why is Index Register No. 2 saved upon entrance to the DELETE subroutine?

- B. Why does the program MAIN execute an ENA instruction before it does a return jump to the DELETE subroutine?

- C. What is the significance of the double asterisk (**) in the coding of the SRCN instruction in the DELETE subroutine?

- D. What is the function of the unconditional jump instruction which follows the SRCN instruction in the DELETE subroutine?

- E. What two conditions will cause the PAUS instruction within the DELETE ROUTINE to do a SKIP EXIT (P + 2)?

- F. Why at location INCRE is the character address in A updated by 1?
- G. What does the A register contain when control is transferred back to program MAIN?

2.

	ENTRY	PROG1
PROG1	ENI	0,1
	ENI	0,2
MOVE	LACH	FILE1,1
	SACH	FILE2,2
	INI	1,2
	ISI	7,1
	UJP	MOVE
	HLT	
	END	PROG1
	ENTRY	PROG2
PROG2	MOVE	10B,FILE1,FILE2
	UJP	+2
	PAUS	4000B
	UJP	*-1
	HLT	
	END	PROG2

Given the above routines, which one will accomplish the move faster?

Assume both routines do a character by character move.

Does the MOVE instruction buy the programmer much time if he has to wait until the MOVE is completed?

3. Write a short program which will search a character block (character addresses CARD to CARD+80) for the first COMMA(,). Loop until the search is complete and determine if a COMMA(,) was found; if one were found, load the ADDRESS into the A register; if not, set A = 0.

STORAGE TESTS

1.

	ENTRY	SEARCH
SEARCH	ENI	0,2
	ENI	7,1
	ENA	200
	ENQ	100
	CPR	LIST,1
	UJP	*+3
	UJP	*+2
	RTJ	FIND
	IJD	*-4,1
	HLT	
FIND	UJP	**
	ENA	LIST
	AIA	1
	STA	ADDRESS,2
	INI	1,2
	UJP,I	FIND
LIST	DEC	1,150,20,300,5,200,100,77
ADDRESS	OCT	0,0,0,0,0,0,0,0
	END	

What does the above program do?

When the program comes to a HALT, what do the following locations contain?

ADDRESS =

ADDRESS+1 =

ADDRESS+2 =

ADDRESS+3 =

2. The number in the A register after executing a SSH instruction is?

- Zero
- The original number from memory, unshifted.
- The original number from memory, shifted.
- The same number that was in A.

3.

	ENTRY	SCAN
SCAN	ENI	0,3
	ENI	0,2
	ENI	3,1
	LDQ	MASK
CONTINUE	LDA	DATA
	MEQ	LIST,1
	UJP	CHECK
	ENA	LIST
	AIA	1
	SHA	2
CHARADDR	AIA	2
	STA	LIST1,3
	INI	1,3
	UJP	CONTINUE
CHECK	SSH	CONTROL
	HLT	
	SHQ	18
	SHA	18
	STAQ	MASK
	INI	1,2
	UJP	SCAN+2
CONTROL	OCT	73567356
MASK	OCT	77000000
DATA	OCT	45000000
LIST	BSS	3
LIST1	OCT	0,0,0,0,0,0,0,0,0,0,0
	END	

What does the above program do?

4. Write a program which will search a table called INFO, looking for the value 62_8 in bit positions 11 - 06. Search every other location in INFO, which is a total of 16_{10} locations long, starting with the last location within the INFO table. The program must keep track of all addresses, where a find had occurred.

BCD OPERATIONS

1. The decimal quantity +32,768,987 is contained in a field beginning at character address 010000. Indicate the octal contents of the appropriate memory locations.

MEMORY LOCATIONS	CONTENTS
------------------	----------

2. After executing a SET 12 and a LDE AA+2 instruction, indicate the sign and contents of the E register. AA = 65136

	CONTENTS
AA	11071002
(E) _f =	02050711
	06010210
	11117102

3. After executing a SET 10 and a ADE BB+2 instruction, indicate the final sign and contents of the E register. BB = 13417

	CONTENTS
(E) _i = -0,000,987,654,321	BB 04060701
(E) _f =	10010010
	11100000

4. After executing a SET 4 and a SBE CC+2 instruction, indicate the final sign and contents of the E register. CC = 40000

	CONTENTS
(E) _i = +0,000,000,000,989	CC 01141011
(E) _f =	02650411

5. After executing a SET 13 and a ADE DD+1,3 instruction, indicate the final sign and contents of the E register. (B3) = 77765 DD = 04002

	CONTENTS
(E) _i = +1,000,000,000,000.	DD 01100611
(E) _f =	11070302
	10100301
	00111151

6. After executing a SET 9 and a STE 10B instruction, indicate the final sign and contents of the E register as well as the contents of the storage locations affected.

(E)_i = -9,568,765,867,898. M.L. CONTENTS
 (E)_f =

7. Fill in the six (6) locations starting with "CHARS" with their octal contents, after the following has been executed. Place X's where contents not known.

(E_D) = +0000045468799

	EZJ,EQ	OUT
	EZJ,LT	OUT
	SET	10
	SFE	2
	STE	ANSWER
	NOB	
OUT	HLT	
CHARS	BCD,C	12,TOTAL EQUAL
ANSWER	BCD,C	10,

CHARS				

(E)_f =
 (D) =

8. Fill in the octal contents of the four (4) locations starting at "DATA", after the following program has been executed. Place X's where the contents are not known.

$$E_D = -0405060708090$$

```

                SET      1
                ENI      0,2
LOOP           EZJ,EQ    DONE
                SFE      -1
                STE      DATA,2
                INI      1,2
                UJP      LOOP
DONE           HLT
DATA          BCD,C      12,
    
```

DATA

$$(E)_f =$$

ANSWER SHEETS

3200 Computer Characteristics - Answers	C1 - C3
Load Instructions - Answers	C4
Store Instructions - Answers	C5
Arithmetic, Fixed Point - Answers	C6
Register Operations Without Storage Reference - Answers	C7
Stop and Jump Instructions - Answers	C8
Inter-Register Transfer - Answers	C9
Search and Move Operations - Answers	C10 - C11
Storage Tests - Answers	C12
BCD Operations - Answers	C13

3200 COMPUTER CHARACTERISTICS - ANSWERS

- | | | | |
|----|--------------------------|-----------------------|--|
| 1. | <u>Character Address</u> | <u>Storage Module</u> | |
| | a) 000134 ₈ | 0 | |
| | b) 031455 ₈ | 0 | |
| | c) 065632 ₈ | 0 | |
| | d) 102373 ₈ | 1 | |
-
- | | | | |
|----|-----------------------|---------------------------|-----------------------|
| 2. | <u>Word Address</u> | <u>Character Position</u> | <u>Storage Module</u> |
| | a) 60146 ₈ | 0 | 3 |
| | b) 32414 ₈ | 3 | 1 |
| | c) 65217 ₈ | 2 | 3 |
| | d) 32174 ₈ | 2 | 1 |
-
- | | | | |
|----|--------------------------|---------------------|---------------------------|
| 3. | <u>Character Address</u> | <u>Word Address</u> | <u>Character Position</u> |
| | 040556 ₈ | 10133 ₈ | 2 |
-
- | | | | |
|----|--------------------------|---------------------|---------------------------|
| 4. | <u>Character Address</u> | <u>Word Address</u> | <u>Character Position</u> |
| | 160560 ₈ | 34134 ₈ | 0 |
-
5. 4
6. a) 2
b) #2
7. 8
8. a) 3204 (Basic Processor) contains the logic to perform 24-bit fixed point arithmetic, 48-bit fixed point addition and subtraction, Boolean, character and word handling and decision making operations.
b) 3205 (Scientific Processor) contains the capability of the 3204, plus floating point and 48-bit precision fixed point multiplication and division.

12. (cont.)

Register Numbers	Reserved for:
22	Real-time clock, current time
23	Current character address (typewriter control)
24 - 27	Temporary storage
30	Instruction word containing the last character address +1 (search control)
31	Instruction word containing the destination address (move control)
32	Real-time clock, interrupt mask
33	Last character address + 1 (typewriter control)
34 - 77	Temporary storage

LOAD INSTRUCTIONS - ANSWERS

1. 27313317
2. 40513315
3. 00000033
4. 23142615
5. 00003314
6. 37264462
57364463
7. 23142615
8. 00000033
9. 13314
10. 42615
11. 00000015
12. 00000040

STORE INSTRUCTIONS - ANSWERS

1. 24130120
2. 01333443
3. 20733445
4. 65030120
5. (TEMP) = A
(TEMP+1) = Q
6. 77530120
7. 44500000
8. (TEMP+1) = 77533403
9. 66700001
10. 66730120

ARITHMETIC, FIXED POINT - ANSWERS

1. $(A)_f = 00003020$
 $(Q)_f = 00000000$ (can't be a negative zero)

2. $(A)_f = 77777711$ (-66_8)
 $(Q)_f = 00000006$ Same Sign as Dividend

3. $(A)_f = 00075767$

4. $(A)_f = 77777776$

5. $(A)_f = 60051600$

REGISTER OPERATIONS WITHOUT STORAGE REFERENCE - ANSWERS

1. $(B^1)_f = 07777$
2. $(A)_f = 77740000$
3. $(Q)_f = 00007777$
4. $(A)_f = 00001000$
 $(Q)_f = 00000000$

1. ✓
2. ✓
3. ✓
4. ✓
- 5.
6. ✓
7. ✓
- 8.
- 9.
10. ✓
- 11.
- 12.
13. ✓
14. ✓
- 15.
- 16.
- 17.
- 18.
19. ✓
- 20.

STOP AND JUMP INSTRUCTIONS - ANSWERS

- | | | | |
|-----|---|----|--|
| 1. | ✓ | 1. | $(A)_f = 00000000$ |
| 2. | ✓ | 2. | $(A)_f = 00000000$ |
| 3. | | | $(Q)_f = 01000000$ |
| 4. | | | $(B^1)_f = 00000$ |
| 5. | | 3. | $(P)_f = 07007$ |
| 6. | ✓ | | $(07002) = \text{UJP } 07007B$ |
| 7. | ✓ | 4. | Looks at all values of TEMP TEMP+4,
for values greater or equal to 100
octal. A count of all values equal
to or greater than 100 octal are
kept in COUNT. A count of all values
less than a 100 octal are kept in
TALLY. |
| 8. | | | |
| 9. | ✓ | 5. | $(P)_f = \begin{matrix} C.1 \\ C.0 \end{matrix}$ |
| 10. | ✓ | | |
| 11. | | | |
| 12. | | | |
| 13. | | | |
| 14. | ✓ | | |
| 15. | ✓ | | |
| 16. | | | |
| 17. | ✓ | | |
| 18. | | | |
| 19. | ✓ | | |
| 20. | ✓ | | |

INTER-REGISTER TRANSFERS - ANSWERS

1. (A) = 16600000

(Q) = 00077771

2. (A) = 00000012

(Q) = 00000012

$(B^3)_f = 00011$

3. 1) 53040000

2) 53740000

3) 53200000

4) 53410022

5) 53140000

6) 53630020

7) 53420077

8) 53330060

9) 53020027

10) 53010055

11) 53500000

4. $(A)_f = 00000000$

$(Q)_f = 77777777$

$(B^1)_f = 77676$

$(V24) = 77777777$

$(V25) = 00100177$

$(V26) = 00177776$

$(V27) = 00077577$

5. 77777777

77777577

00000000

00000000

SEARCH AND MOVE OPERATIONS - ANSWERS

1. A) All subroutines which are used by other programs, should save any register that it uses in performing its task. Index Register #2 may contain valuable data to program MAIN, at the time it transfers control to the DELETE subroutine; so if (B²) was not saved upon entrance and restored before exiting from the DELETE subroutine, it would contain invalid data upon return to program MAIN.
- B) The DELETE subroutine must have the first word address of the card image area in the A Register upon entrance to its routine. So program MAIN must obtain the first word address of the card image area into A, before it transfers control to the DELETE subroutine. It does this by an ENA DATACARD.
- C) Normally a field represented by a double asterisk(**) will be modified during the execution of a program and the double asterisk (**) provides a convenient way to see if the modification took place. (**) presets the field to all 1₂'s.
- D) Control is transferred to this instruction if a SRCE, SRCN, or MOVE instruction is already in progress and the search instruction preceding this jump instruction will not be initiated until any previous search or move is completed.
- E)
 1. When a comparison occurs between the search character (non-blank) and a non-blank character in storage.
 2. When register file 20 is equal to register file 30 (no non-blank character found).
- F) The search is to continue until all blank characters are deleted. Upon a find, the search is terminated and register file location #20 contains the address of the find. So to continue the search this address must be updated by 1 or the program will be hung in a loop.

- G) Upon exiting from the DELETE subroutine, A will contain the last character address + 1 of the DATACARD area with all blank characters deleted.

2.	A)	PROG1	PROG2
		1.8	90.2
		1.8	<u>1.8</u>
		3.5	92.0 usec.
		3.5 Repeated 8 times	
		1.8	
		2.6	
		1.8	
		<u>1.8</u>	
		109.2 usec.	

- B) On the above move he saves only 17.2 usec. (approximately 2 usec. per character). On a maximum move of 128 characters, the MOVE instruction would be approximately 256 usec.

	IDENT	PROGRAM
	ENTRY	START
START	SRCE	73B,CARD,CARD+80
	UJP	*-2
	PAUS	4000B
	UJP	*-1
	TMA	30B
	LPA	MASK
	SHAQ	24
	TMA	20B
	LPA	MASK
	AQJ,EQ	EXIT
	HLT	
EXIT	ENA,S	+0
	UJP	*-2
MASK	ØCT	377777
	END	

STORAGE TESTS - ANSWERS

1. A) The program searches a list of operands from LIST thru LIST+7 (10 locations) looking for values that are within or equal to the upper limit in A (200_{10}) and the lower limit in Q (100_{10}). On a find, the addresses of the operands are stored in a table called ADDRESS.

B) <u>M.L.</u>	<u>Contents of</u>
ADDRESS	LIST+6
ADDRESS+1	LIST+5
ADDRESS+2	LIST+1
ADDRESS+3	0

2. Answer is d; contents of A is not altered.
3. The program searches a list of 6 bit characters, from LIST thru LIST+2 (3 locations) for a 45 code in any character position (0 thru 3) of each location. Upon a find, the character address is generated and stored into a table called LIST1.

4.	IDENT	PROGRAM	
	ENTRY	START	
START	ENI	17,1	$B^1 = \text{SIZE OF TABLE (16)} + \text{INTERVAL}-1$
	ENI	0,2	$B^2 = \text{ADDRESS COUNTER}$
	ENQ	7700B	$Q = 00007700 \text{ (MASK)}$
SEARCH	ENA	6200B	
	MEQ	INF \emptyset ,2	$\text{INF}\emptyset = \text{START OF TABLE}, 2 = \text{INTERVAL}$
	UJP	THRU	SEARCH COMPLETE
	ENA	INF \emptyset	$(A) = 000\text{INF}\emptyset$
	AIA	1	$(A) = 000\text{INF}\emptyset + B^1$
	STA	LIST,2	SAVE FIND ADDRESS
	INI	1,2	PREPARE FOR NEXT FIND
	UJP	SEARCH	
THRU	HLT		
INF \emptyset	BSS	16	
LIST	\emptyset CT	0,0,0,0,0,0,0	

Index register 1 must start with 17_{10} , because it will be decremented by the interval (2) before the search begins. This will start the search at $\text{INF}\emptyset+15$, which is the last location of the table.

BCD OPERATIONS - ANSWERS

1. 02000 03020706
 02001 10111007

2. $(E)_f = +0822579612899$

3. $(E)_f = +0006193435479$

4. $(E)_f = +0000000009914$

5. $(E)_f = +0026711689001$

6. 00010 = 07060510
 00011 = 06071011
 00012 = 50XXXXXX
 $(E)_f = -0000000009568$

7. CHARS T O T A
 L E Q
 U A L
 04 05 04
 06 10 07 11
 11 00 00 XX

$(E)_f = +0000000000000$

$(D) = 10_{10}$ or 12_B

8. DATA 51 50 47 46
 45 44 XX XX
 XX XX XX XX
 XX XX XX XX

$(E)_f = +0000000000000$

automatically changes to a +0

SUMMARY OF INSTRUCTION EXECUTION TIMES, μsec .

2.5	ADA	2.5	LDI
3.8	ADAQ	2.5	LDL
11.5*	ADE	2.5	LDQ
1.3*	AEU	2.5	LPA
1.3	AIA	2.5	LQCH
1.3	ANA		
1.3	ANI	4.2 + 4.2n	MEQ
1.3	ANQ	3.3	MOVE
1.3	AQA	4.2 + 4.2n	MTH
1.3*	AQE	7.8-11.0	MUA
1.9	AQJ	16.0-21.0*	MUAQ
1.9	ASE		
1.9	ASG	3.3	OTAC
1.9	AZJ	3.3	OTAW
		3.3	OUTC
1.3-1.7	CINS	3.3	OUTW
***	CON		
1.3-1.7	COPY	2.0 us-40 ms	PAUS
2.5-3.4	CPR		
1.3	CTI	1.3*	QEL
1.3	CTO	1.9	QSE
		1.9	QSG
1.3	DINT		
11.25	DVA	3.8	RAD
22.5*	DVAQ	2.5	RTJ
1.3*	EAQ	2.5	SACH
1.3	ECHA	2.5	SBA
1.3	EINT	3.8	SBAQ
1.3*	ELQ	1.3	SBCD
1.3	ENA	11.5*	SBE
1.3	ENI	2.5	SCA
1.3	ENQ	1.9-3.9	SCAQ
1.3*	EOJ	2.5	SCHA
1.3*	EUA	1.3	SCIM
1.3-1.7	EXS	***	SEL
1.3*	EZJ	1.3*	SET
		1.3-4.3*	SFE
10.0-12.0*	FAD	1.3	SFPF
20.0*	FDV	1.3-2.7	SHA
14.0-18.0*	FMU	1.3-2.7	SHAQ
10.0-12.0*	FSB	1.3-2.7	SHQ
		1.3	SJ1-6
—	HLT	1.3	SLS
		2.5	SQCH
1.3	IAI	3.3	SRCE
**	IAPR	3.3	SRCN
1.9	IJD	2.5	SSA
1.9	IJI	3.8	SSH
1.3	INA	1.3	SSIM
***	INAC	2.5	STA
***	INAW	3.8	STAQ
1.3	INCL	8.0*	STE
1.3	INI	2.5	STI
3.3	INPC	2.5	STQ
3.3	INPW	2.5	SWA
1.3	INQ	1.3	TAI
1.3-1.7	INS	1.8	TAM
1.3-1.7	INTS	1.3	TIA
1.3	IOCL	1.8	TIM
1.9	ISD	1.8	TMA
1.9	ISE	1.8	TMI
1.9	ISG	1.8	TMQ
1.9	ISI	1.8	TQM
2.5	LACH	—	UCS
2.5	LCA	1.3	UJP
3.8	LCAQ		
2.5	LDA	1.3	XOA
3.8	LDAQ	1.3	XOI
8.0*	LDE	1.3	XOQ

n = number of words searched.

* = Trapped instruction in computers without the appropriate optional hardware package.

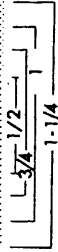
** = Dependent upon interrupt response.

*** = Dependent upon a variable signal response time from an external source of equipment.

INSTRUCTION INDEX

BY OCTAL OPERATION CODE			BY MNEMONIC OPERATION CODE			BY MNEMONIC OPERATION CODE			
OCTAL OPERATION CODE	MNEMONIC OPERATION CODE	SECTION NUMBER	OCTAL OPERATION CODE	MNEMONIC OPERATION CODE	SECTION NUMBER	MNEMONIC OPERATION CODE	SECTION NUMBER	MNEMONIC OPERATION CODE	SECTION NUMBER
00.0	HLT	3.5.5	40	STA,I	3.2.1	ADA,I	3.3.1	LDA,I	3.1.1
00.1	SJ1	3.5.6	41	STQ,I	3.2.2	ADAQ,I	6.1	LDAQ,I	5.1
00.2	SJ2	3.5.6	42	SACH	8.2.3	ADE	14.2.8	LDE	14.2.6
00.3	SJ3	3.5.6	43	SQCH	8.2.4	AEU	13.2.2	LDI;I	3.1.3
00.4	SJ4	3.5.6	44	SWA,I	3.7	AIA	9.5.2	LDL,I	7.2
00.5	SJ5	3.5.6	45	STAQ,I	5.2	ANA	7.4.2	LDQ,I	3.1.2
00.6	SJ6	3.5.6	46	SCHA,I	8.2.5	ANA,S	7.4.2	LPA,I	7.4.1
00.7	RTJ	3.5.4	47	STI,I	3.2.3	ANI	7.4.4	LQCH	8.2.2
01	UJP,I	3.5.1	50	MUA,I	3.3.3	ANQ	7.4.3	MEQ	11.1
02.0	No Operation		51	DVA,I	3.3.4	ANQ,S	7.4.3	MOVE,INT	10.3
02.1-3	IJI	3.5.7	52	CPR,I	11.4	AQA	9.5.1	MTH	11.2
02.4	No Operation		53.01	TMQ	9.3.1	AQE	13.4.2	MUA,I	3.3.3
02.5-7	IJD	3.5.7	53.02	TMA	9.2.1	AQJ,EQ	3.5.3	MUAQ,I	6.3
03.0	AZJ,EQ	3.5.2	53.04	AQA	9.5.1	AQJ,GE	3.5.3	NOP	
03.1	AZJ,NE	3.5.2	53.(0+b)0	TIA	9.1.1	AQJ,LT	3.5.3	OTAC,INT	20.8.3
03.2	AZJ,GE	3.5.2	53.(0+b)3	TMI	9.4.1	AQJ,NE	3.5.3	OTAW,INT	20.8.4
03.3	AZJ,LT	3.5.2	53.(0+b)4	ATA	9.5.2	ASE	3.6.1	OUTC,INT,	20.7
03.4	AQJ,EQ	3.5.3	53.41	TQM	9.3.2	ASE,S	3.6.1	B,H	
03.5	AQJ,NE	3.5.3	53.42	TAM	9.2.2	ASG	3.6.2	OUTW,INT,	20.5
03.6	AQJ,GE	3.5.3	53.(4+b)0	TAI	9.1.2	ASG,S	3.6.2	B,H	
03.7	AQJ,LT	3.5.3	53.(4+b)3	TIM	9.4.2	AZJ,EQ	3.5.2	PAUS	10.4 &
04.0	ISE	3.6.1	53.(4+b)4	IAI	9.5.3	AZJ,GE	3.5.2		20.10.5.3
04.1-3	ISE	3.6.1	54	LDI,I	3.1.3	AZJ,LT	3.5.2	QEL	13.3.2
04.4	ASE,S	3.6.1	55.0	No Operation		AZJ,NE	3.5.2	QSE	3.6.1
04.5	QSE,S	3.6.1	55.1	ELQ	13.3.1	CINS	20.9.5	QSE,S	3.6.1
04.6	ASE	3.6.1	55.2	EUA	13.2.1	CON	20.2	QSG	3.6.2
04.7	QSE	3.6.1	55.3	EAQ	13.4.1	COPY	20.9.2	QSG,S	3.6.2
05.0	ISG	3.6.2	55.4	No Operation		CPR,I	11.4	RAD,I	3.3.5
05.1-3	ISG	3.6.2	55.5	QEL	13.3.2	CTI	20.10.5.1	RTJ	3.5.4
05.4	ASG,S	3.6.2	55.6	AEU	13.2.2	CTO	20.10.5.2	SACH	8.2.3
05.5	QSG,S	3.6.2	55.7	AQE	13.4.2	DINT		SBA,I	3.3.2
05.6	ASG	3.6.2	56	MUAQ,I	6.3	DVA,I	3.3.4	SBAQ,I	6.2
05.7	QSG	3.6.2	57	DVAQ,I	6.4	DVAQ,I	6.4	SBCD	
06.0-7	MEQ	11.1	60	FAD,I	12.3.1	EAQ	13.4.1	SBE	14.2.9
07.0-7	MTH	11.2	61	FSB,I	12.3.2	ECHA	8.2.6	SCA,I	7.5.1
10.0	SSH	11.3	62	FMU,I	12.3.3	ECHA,S	8.2.6	SCAQ	13.5
10.1-3	ISI	3.6.3	63	FDV,I	12.3.4	EINT	21.1-2	SCHA,I	8.2.5
10.4	ISD	3.6.3	64	LDE	14.2.6	ELQ	13.3.1	SCIM	
10.5-7	ISD	3.6.3	65	STE	14.2.7	ENA	3.4.4	SEL	20.3
11.0	ECHA	8.2.6	66	ADE	14.2.8	ENA,S	3.4.4	SET	14.2.5
11.4	ECHA,S	8.2.6	67	SBE	14.2.9	ENI	3.4.4	SFE	14.2.1
12.0-3	SHA	3.8.2	70.0-3	SFE	14.2.1	ENQ	3.4.4	SFFP	
12.4-7	SHQ	3.8.2	70.4	EZJ,EQ	14.2.2	ENQ,S	3.4.4	SHA	3.8.2
13.0-3	SHAQ	5.3	70.5	EZJ,LT	14.2.3	EOJ	14.2.4	SHAQ	5.3
13.4-7	SCAQ	13.5	70.6	EOJ	14.2.4	EUA	13.2.1	SHQ	3.8.2
14.0	NOP		70.7	SET	14.2.5	EXS	20.9.1	SJI	3.5.6
14.1-3	ENI	3.4.4	71	SRCE,INT	10.2.1	EZJ,EQ	14.2.2	SJ2	3.5.6
14.4	ENA,S	3.4.4	71	SRCN,INT	10.2.2	EZJ,LT	14.2.3	SJ3	3.5.6
14.5	ENQ,S	3.4.4	72	MOVE,INT	10.3	FAD,I	12.3.1	SJ4	3.5.6
14.6	ENA	3.4.4	73	INPC,INT,	20.6	FDV,I	12.3.4	SJ5	3.5.6
14.7	ENQ	3.4.4		B,H		FMU,I	12.3.3	SJ6	3.5.6
15.0	No Operation		73	INAC,INT	20.8.1	FSB,I	12.3.2	SLS	
15.1-3	INI	3.4.3	74	INPW,INT,	20.4	HLT	3.5.5	SQCH	8.2.4
15.4	INA,S	3.4.1		B,N		IAI	9.5.3	SRCE,INT	10.2.1
15.5	INQ,S	3.4.2	74	INAW,INT	20.8.2	IAPR		SRCN,INT	10.2.2
15.6	INA	3.4.1	75	OUTC,INT,	20.7	IJD	3.5.7	SSA,I	7.6
15.7	INQ	3.4.2		B,H		IJI	3.5.7	SSH	11.3
16.0	No Operation		75	OTAC,INT	20.8.3	INA	3.4.1	SSIM	21.1
16.1-3	XOI	7.5.4	76	OUTW,INT,	20.5	INA,S	3.4.1	STA,I	3.2.1
16.4	XOA,S	7.5.2		B,N		INAC,INT	20.8.1	STAQ,I	5.2
16.5	XOQ,S	7.5.3	76	OTAW,INT	20.8.4	INAW,INT	20.8.2	STE	14.2.7
16.6	XOA	7.5.2	77.0	CON	20.2	INCL	21.2	STI,I	3.2.3
16.7	XOQ	7.5.3	77.1	SEL	20.3	INI	3.4.3	STQ,I	3.2.2
17.0	No Operation		77.2	EXS	20.9.1	INPC,INT,	20.6	SWA,I	3.7
17.1-3	ANI	7.4.4	77.2	COPY	20.9.2	B,H		TAI	9.1.2
17.4	ANA,S	7.4.2	77.3	INS	20.9.4	INPW,INT,	20.4	TAM	9.2.2
17.5	ANQ,S	7.4.3	77.3	CINS	20.9.5	B,N		TIA	9.1.1
17.6	ANA	7.4.2	77.4	INTS	20.9.3	INQ	3.4.2	TIM	9.4.2
17.7	ANQ	7.4.3	77.50	INCL	21.2	INQ,S	3.4.2	TMA	9.2.1
20	LDA,I	3.1.1	77.51	IOCL		INS	20.9.4	TMI	9.4.1
21	LDQ,I	3.1.2	77.52	SSIM	21.1	INTS	20.9.3	TMQ	9.3.1
22	LACH	8.2.1	77.53	SCIM		IOCL		TQM	9.3.2
23	LQCH	8.2.2	77.54-56	No Operation		ISD	3.6.3	UCS	
24	LCA,I	7.3.1	77.57	IAPR		ISE	3.6.1	UJP,I	3.5.1
25	LDAQ,I	5.1	77.6	PAUS	10.4 &	ISG	3.6.2	XOA	7.5.2
26	LCAQ,I	7.3.2			20.10.5.3	ISI	3.6.3	XOA,S	7.5.2
27	LDL,I	7.2	77.70	SLS		LACH	8.2.1	XOI	7.5.4
30	ADA,I	3.3.1	77.71	SFFP		LGA,I	7.3.1	XOQ	7.5.3
31	SBA,I	3.3.2	77.72	SBCD		LCAQ,I	7.3.2	XOQ,S	7.5.3
32	ADAQ,I	6.1	77.73	DINT					
33	SBAQ,I	6.2	77.74	EINT	21.1-2				
34	RAD,I	3.3.5	77.75	CTI	20.10.5.1				
35	SSA,I	7.6	77.76	CTO	20.10.5.2				
36	SCA,I	7.5.1	77.77	UCS					
37	LPA,I	7.4.1							

CONTROL DATA



▶ ▶ CUT OUT FOR USE AS LOOSE-LEAF BINDER TITLE TAB

CONTROL DATA
CORPORATION

CORPORATE HEADQUARTERS, 8100 34th AVE. SO., MINNEAPOLIS, MINN, 55440
SALES OFFICES AND SERVICE CENTERS IN MAJOR CITIES THROUGHOUT THE WORLD