SCHEDULE OF TOPICS

DAY 1	DAY 2	DAY 3	DAY 4	DAY 5
.COURSE DEFINITION .COMPUTERS & PROGRAMS .NUMBER SYSTEMS .INTRODUCTION TO 2100 HARDWARE .INSTRUCTION & DATA TYPES .INTRODUCTION TO 2100 SOFTWARE .INSTRUCTIONS (MRG)	.REVIEW .PAGE CONCEPTS & INDIRECT ADDRESSING .PROGRAM LINKAGE .ASSEMBLY LISTINGS .INSTRUCTIONS (MRG, ASG, SRG) .INTRODUCTION TO I/O (.IOC.) .BCS PROCEDURES & OVERVIEW .READING ASSIGNMENT & QUIZ	.REVIEW .ADDITIONAL ASSEMBLY PROCEDURES .ADDITIONAL EDITOR PROCEDURES .INSTRUCTIONS (ASG, SRG, EAG, USER) .FORMATTER .TAPE FORMATS .READING ASSIGNMENT & QUIZ	.REVIEW .INTERRUPT SYSTEM .INSTRUCTIONS (IOG, MRG, SRG) .SIO SYSTEM .ABSOLUTE PROGRAMMING .RESTORING THE BBL .DMA PROGRAMMING .2100 MICROPROGRAMMING .READING ASSIGNMENT & QUIZ	.REVIEW .COMMON .BINARY DATA .DEBUG .EDITOR INFO .I/O EXTENDER .MULTIPLEXER .2100 SPECIFICATIONS .POWER FAIL & MEM. PROT. .DRIVER CONSIDERATIONS .DOCUMENTATION .2100 BIBLIOGRAPHY .CRITIQUES
L U N C H				GRADUATION
.TYPICAL OPERATING PROCEDURES .A PROBLEM SOLVING APPROACH .READING ASSIGNMENT & QUIZ .LAB 1 PREPARATION	LAB 2 PREPARATION	LAB 3 PREPARATION	LAB 4 PREPARATION	OPTIONAL LAB
LAB 1 1.1 SUM OF NUMBERS 1.2 SWITCH REGISTER BIT TEST 1.3 SWITCH REGISTER SUM	LAB 2 2.1 WRITE NAME ON TTY 2.2 SUBROUTINE COMMUNICATION 2.3 FIND ERRORS ON TAPE 2.4 LIMIT CHECK SUBROUTINE	LAB 3 3.1 ECHO NAME 3.2 CONVERT & PUNCH NUMBERS 3.3 READ NUMBERS & CONVERT 3.4 MODIFY LAB 2.2 TO RUN WITH FORMATTER	LAB 4 .MICRO-PROGRAMMING DEMO .BBL RESTORATION 4.1 CONFIGURE ASSEMBLER 4.2 DUPLICATE TAPE (SFS) 4.3 DUPLICATE TAPE (SIO)	

THE COMPUTER

STORED PROGRAM

COMMUNICATION

EXECUTES

MILLIONS OF

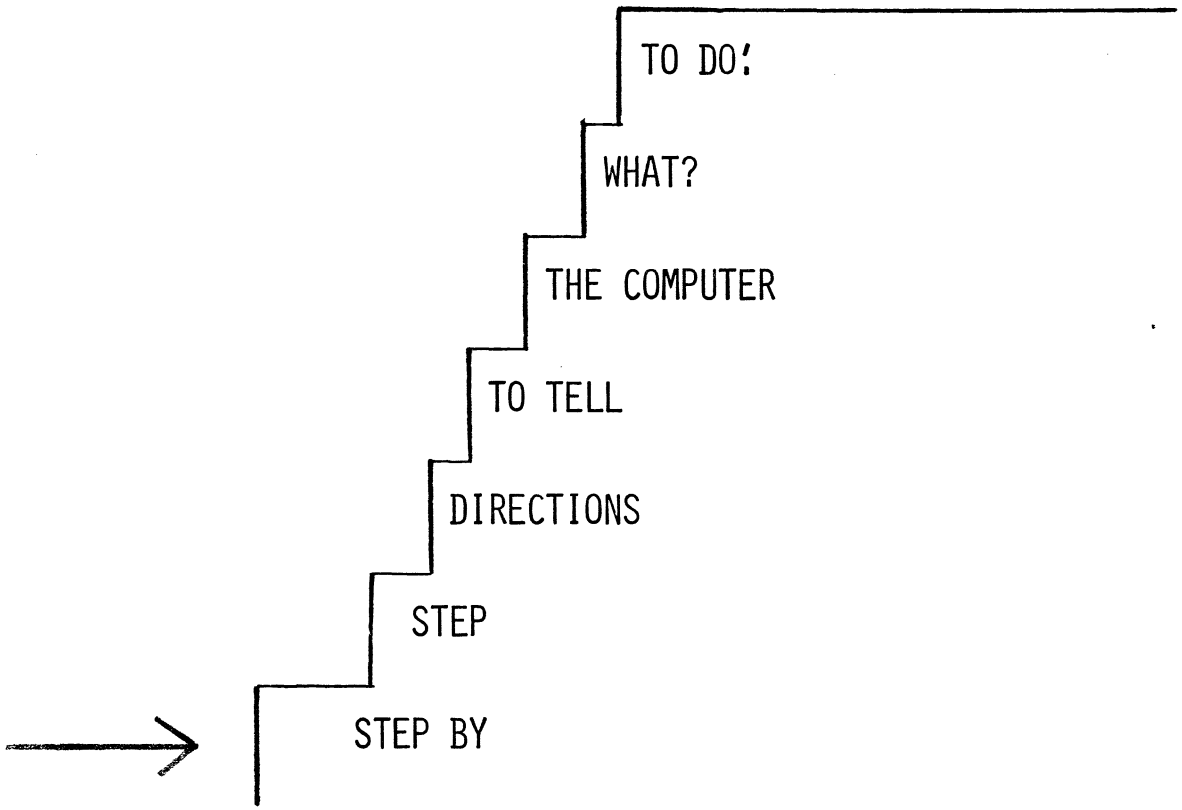
OPERATIONS IN

SECONDS!

LOGIC DECISIONS

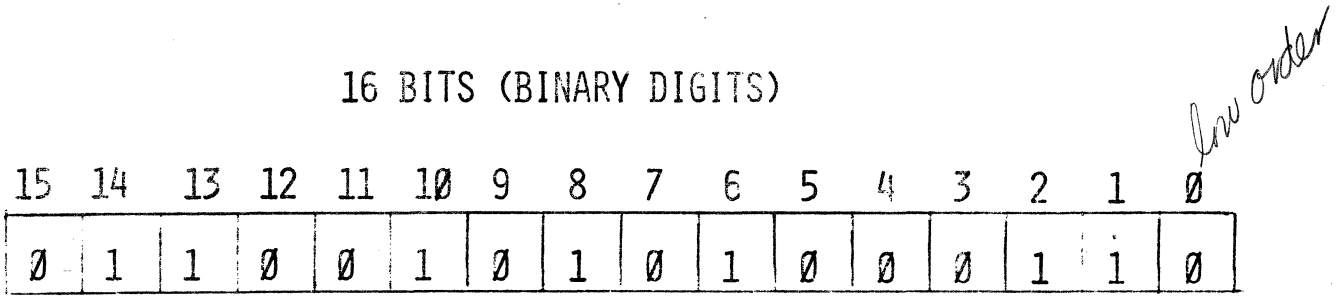
ARITHMETIC OPERATIONS

A PROGRAM



HP COMPUTER WORD

16 BITS (BINARY DIGITS)



A BIT IS EITHER

1 = ON OR

0 = OFF

M A C H I N E I N S T R U C T I O N S

DIRECTIONS THAT THE COMPUTER UNDERSTANDS

```
102000  HLT

060322  LDA T2

103710  STC 10B,C
```

DECIMAL NUMBERS

109_{10} MAY BE REPRESENTED AS:

$(1 \times 10^2) + (0 \times 10^1) + (9 \times 10^0)$

$100 + 0 + 9 = 109_{10}$

OCTAL NUMBERS

DIGITS = Ø,1,2,3,4,5,6,7

155_8 MAY BE REPRESENTED AS:

$(1 \times 8^2) + (5 \times 8^1) + (5 \times 8^Ø)$

$64 + 4Ø + 5 = 1Ø9_{1Ø}$

THUS: $1Ø9_{1Ø} = 155_8$

B I N A R Y N U M B E R S

DIGITS = Ø,1

1101101_2 MAY BE REPRESENTED AS:

$$(1 \times 2^6) + (1 \times 2^5) + (\emptyset \times 2^4) + (1 \times 2^3) + (1 \times 2^2) + (\emptyset \times 2^1) + (1 \times 2^\emptyset)$$

$$64 + 32 + \emptyset + 8 + 4 + \emptyset + 1 = 109_{10}$$

THUS: $109_{10} = 155_8 = 1101101_2$

B I N A R Y / O C T A L C O N V E R S I O N

3 BINARY DIGITS = 1 OCTAL DIGIT

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	1	1	Ø	1	1	Ø	1	BINARY
Ø		Ø			Ø		1			5			5			OCTAL

BINARY	OCTAL
ØØØ	Ø
ØØ1	1
Ø1Ø	2
Ø11	3
1ØØ	4
1Ø1	5
11Ø	6
111	7

DECIMAL TO OCTAL CONVERSION

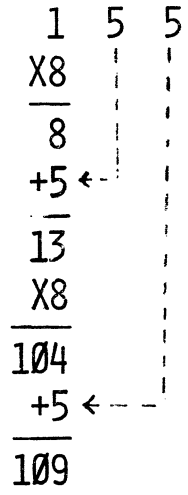
CONVERT 109_{10} TO OCTAL

		REMAINDER	
8	109	5	← LSD
8	13	5	
8	1	1	
		↑	
		READ	

$$109_{10} = 155_8$$

OCTAL TO DECIMAL CONVERSION

CONVERT 155_8 TO DECIMAL



$155_8 = 109_{10}$

NEGATIVE NUMBERS

SIGN BIT
Ø = +
1 = -

SIGN	VALUE	
Ø	1 1	+3
Ø	1 Ø	+2
Ø	Ø 1	+1
Ø	Ø Ø	Ø
1	1 1	-1
1	1 Ø	-2
1	Ø 1	-3
1	Ø Ø	-4

TWO'S COMPLEMENT = COMPLEMENT + 1

CONVERTING A NEGATIVE NUMBER TO MACHINE FORM

$-100_{10} = ?_2$

```
    12
8 ) 100    4

     1
8 ) 12     4

     0
8 ) 1      1
```

$100_{10} = 144_8 = 001\ 100\ 100_2$

BUT, THE ORIGINAL NUMBER IS NEGATIVE, & CONSIDERING THE 2100 16 BIT WORD,

```
0  000  000  001  100  100
1  111  111  110  011  011
                      +1
-------------------------
1  111  111  110  011  100
```

2'S COMPLEMENT

CLASSROOM EXERCISE 1-1A

1. CONVERT THE FOLLOWING DECIMAL NUMBERS TO OCTAL.

   1024_{10} = 2000 8

   32767_{10} = 77778

2. CONVERT THE FOLLOWING OCTAL NUMBERS TO DECIMAL.

   457_8 = 303_{10}

   1024_8 = 532_{10}

3. CONVERT THE FOLLOWING OCTAL NUMBERS TO BINARY.

   17700_8 = $1\,111\,111\,000\,000_2$

   12345_8 = $001\,010\,011\,100\,101_2$

4. WHAT IS THE "EASY" WAY TO CONVERT BINARY NUMBERS TO DECIMAL AND DECIMAL NUMBERS TO BINARY? octal

CLASSROOM EXERCISE 1-1B

1. CONVERT THE FOLLOWING NEGATIVE DECIMAL NUMBERS TO OCTAL.

-100_{10} = 177634_8

-12_{10} = 177764_8

2. CONVERT THE FOLLOWING NEGATIVE OCTAL NUMBERS TO DECIMAL.

177777_8 = -1

100001_8 = -32767

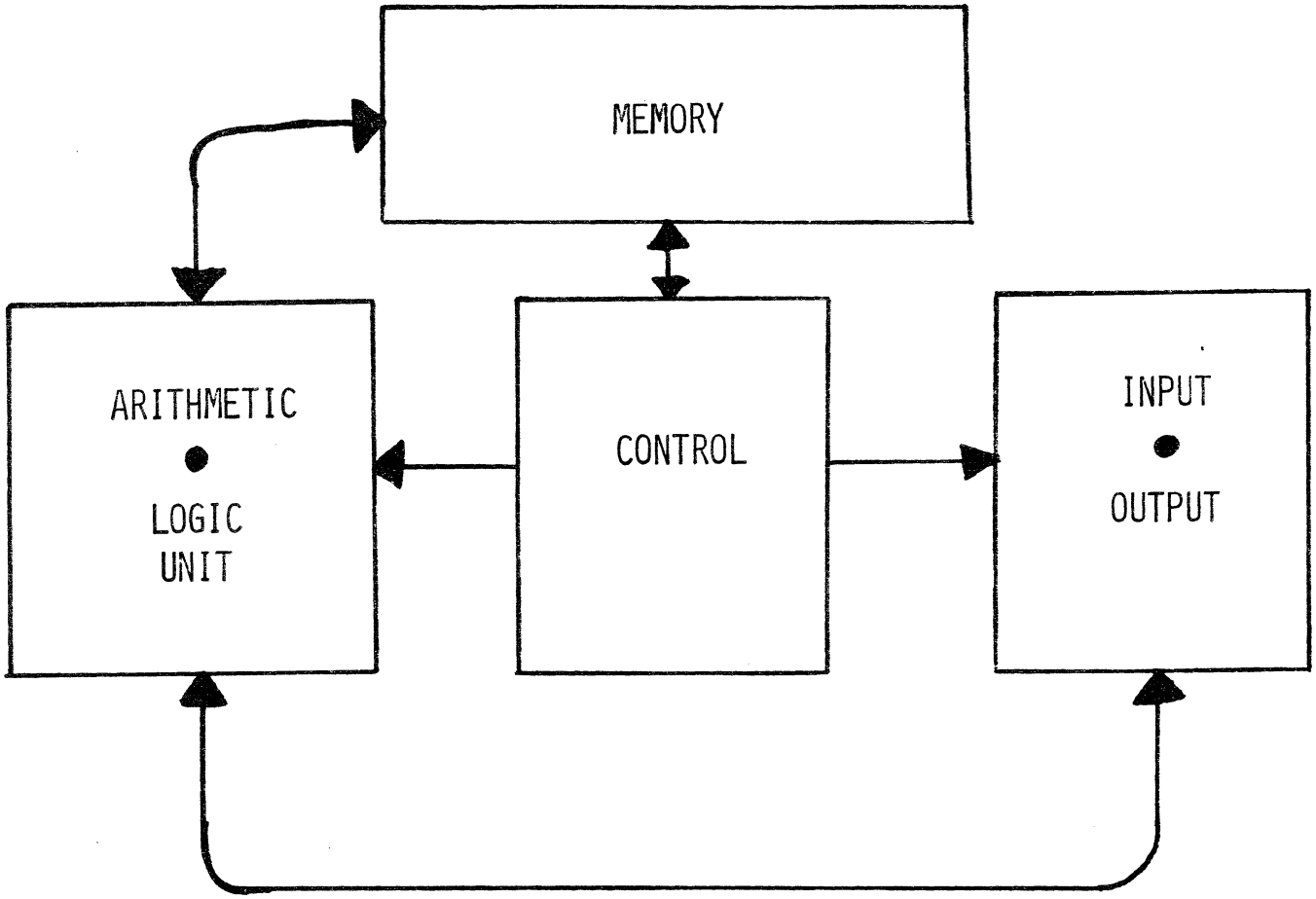
BINARY & OCTAL ADDITION

```
 11  11
011  101  100
001  111  010
101  100  110
```

```
 1
354
172
546
```

NOTE: ADD AS IF DECIMAL,
THEN, IF THE SUM IS ≥ 8,
SUBTRACT 8 & CARRY A 1.

COMPUTER SECTIONS

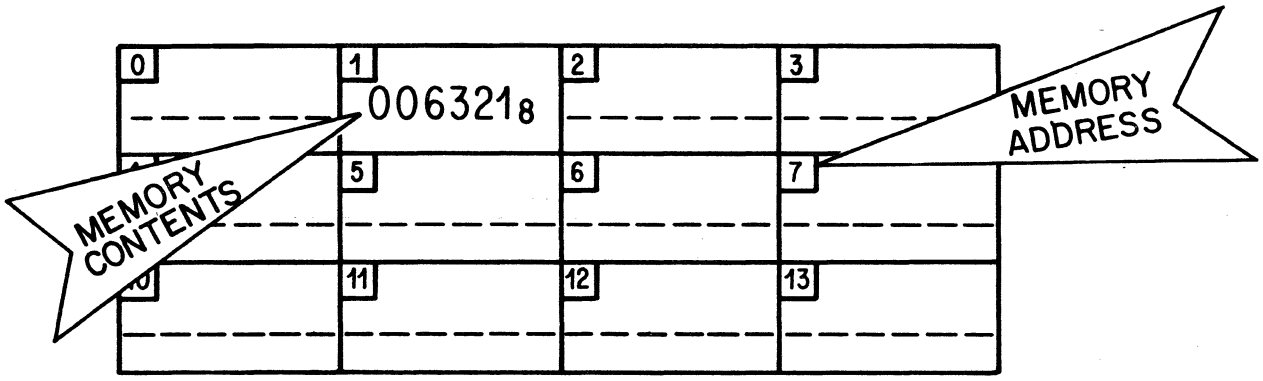


MEMORY

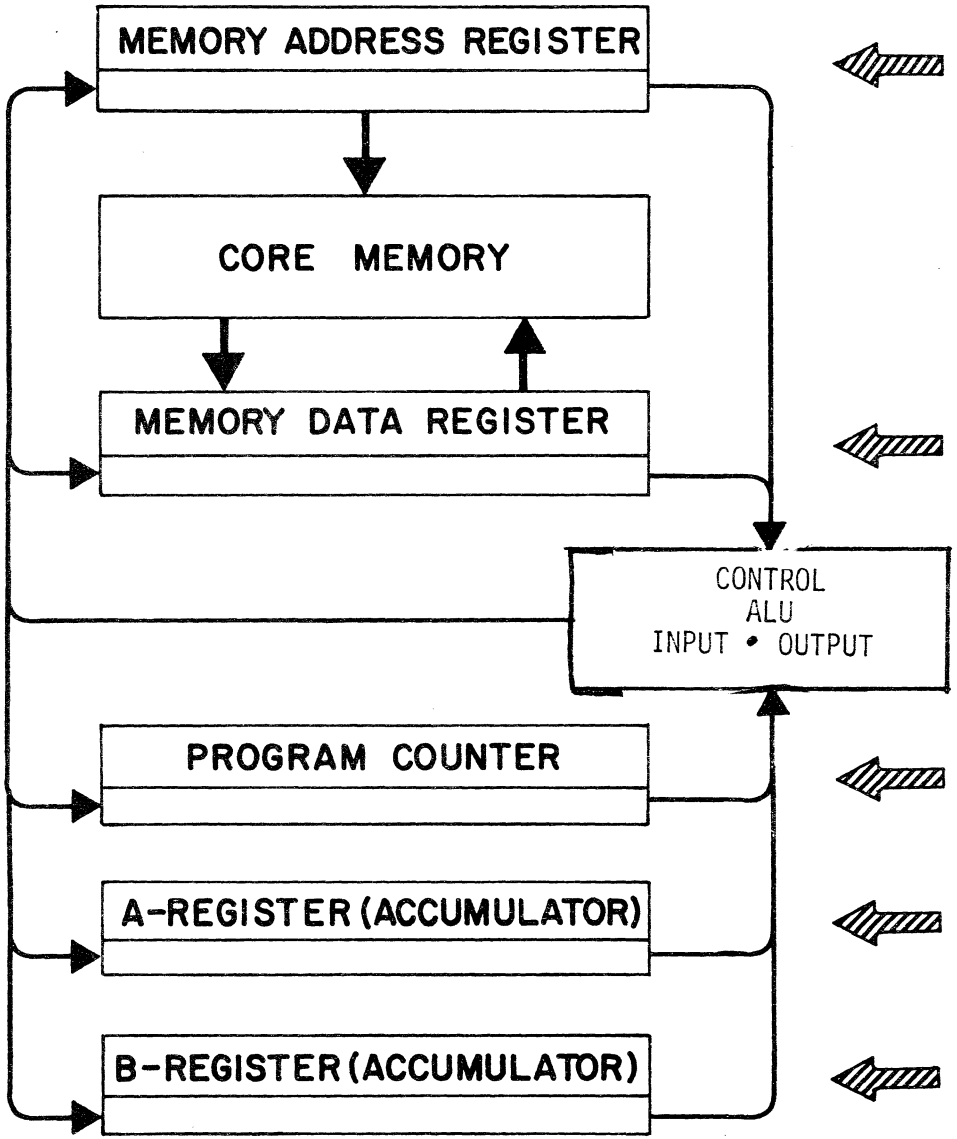
THE MEMORY OF A COMPUTER CONSISTS OF SOME NUMBER OF "MEMORY LOCATIONS"

EACH MEMORY LOCATION IS IDENTIFIED BY A "UNIQUE ADDRESS"

EACH MEMORY LOCATION CONTAINS 16 BITS — THE "COMPUTER WORD" SIZE.



REGISTERS



To access a location its address must be in the M-REGISTER.

All information transferred IN/OUT of memory is routed through the MEMORY DATA REGISTER. (T-REGISTER)

Keeps track of program steps (memory locations containing instructions).

Working register used for arithmetic and logical operations

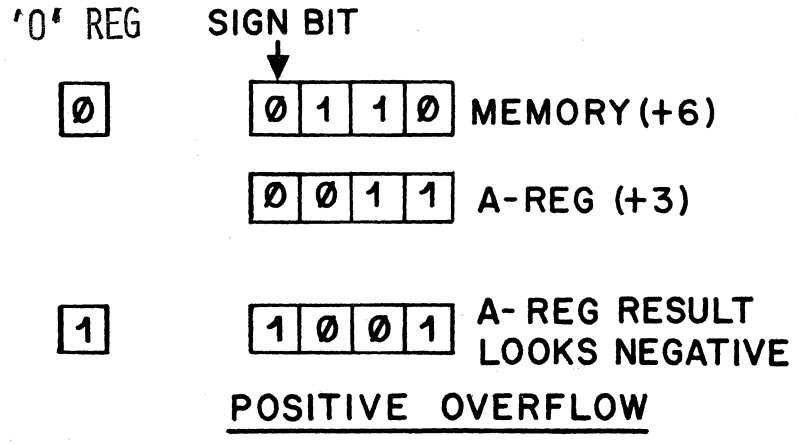
Working register used for arithmetic operations.

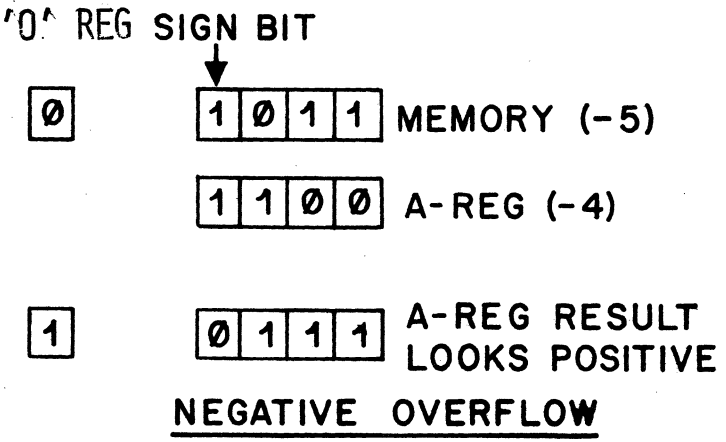
OVERFLOW REGISTER

1 BIT

1. SET WHEN "A" OR "B" REGISTER ARITHMETIC CAPACITY IS EXCEEDED.

2. SET AND CLEARED BY SPECIFIC INSTRUCTIONS.



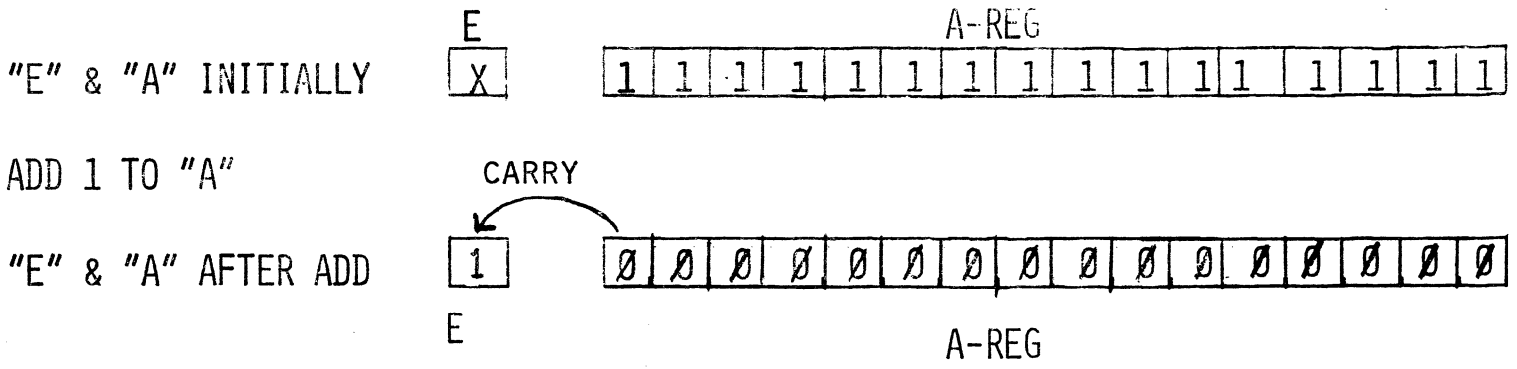


E X T E N D R E G I S T E R

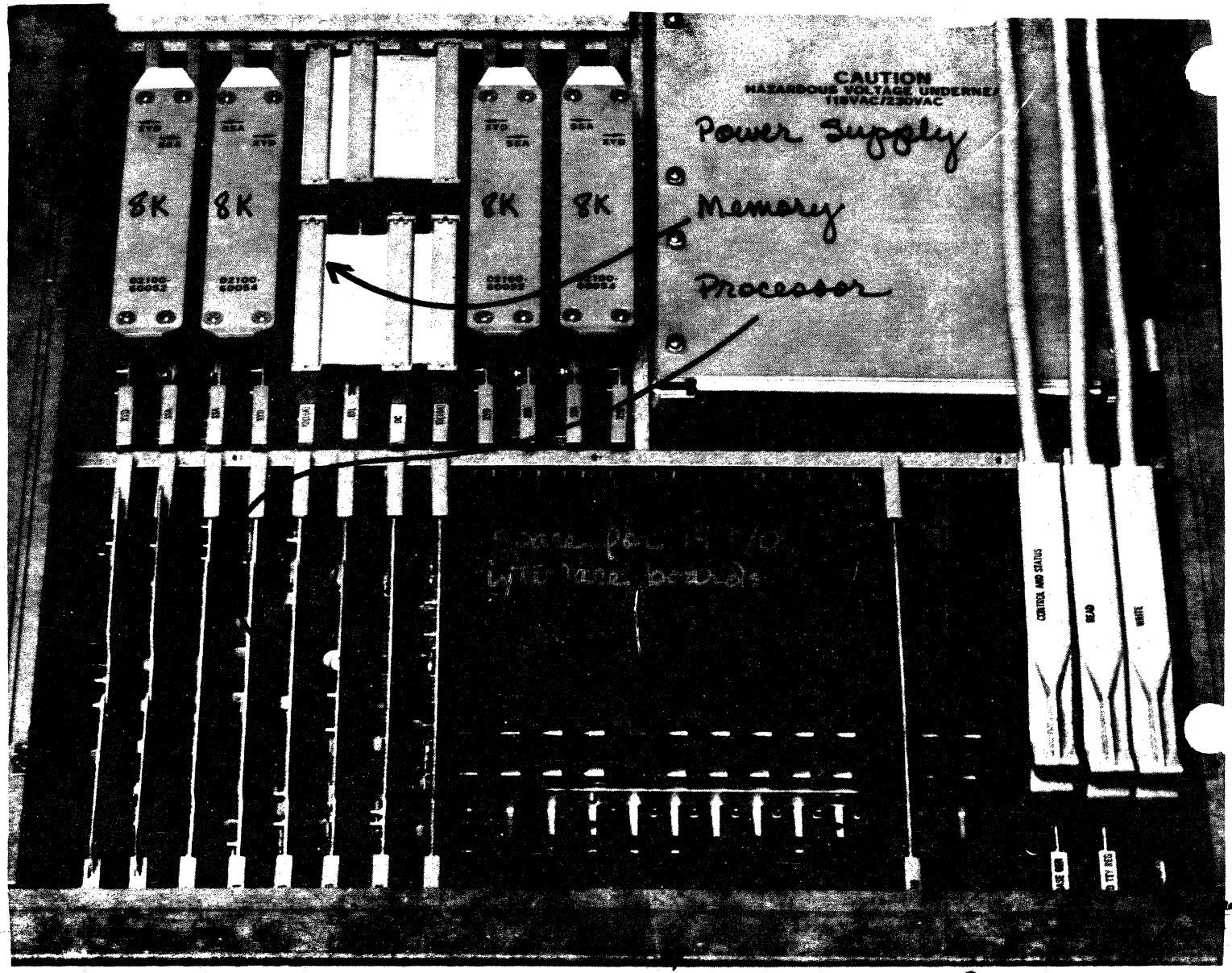
1 BIT

1. SET BY CARRY OUT OF BIT 15 (SIGN BIT) OF "A" OR "B" REGISTERS.

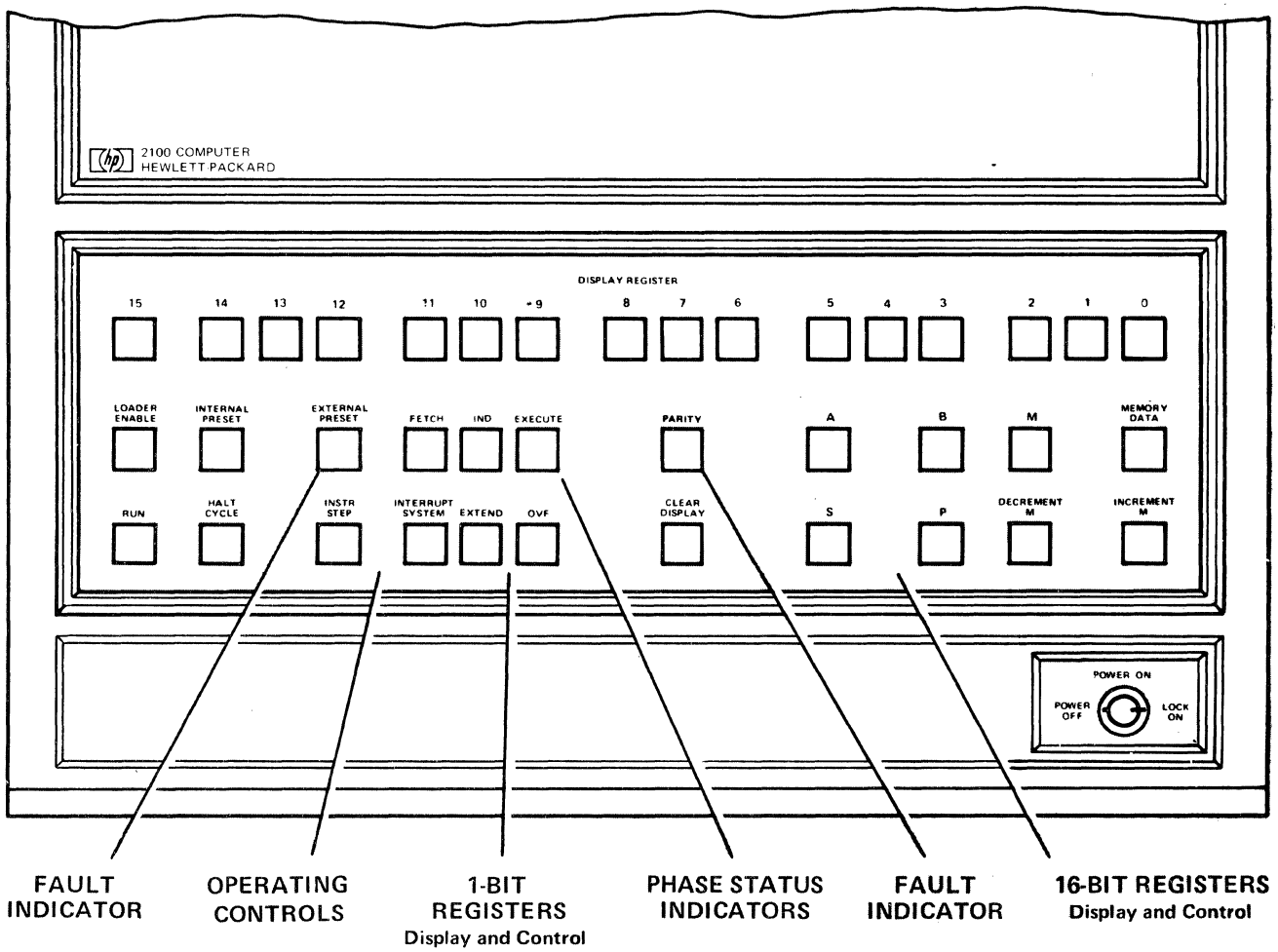
2. SET AND CLEARED BY SPECIFIC INSTRUCTIONS.



AN INSIDE VIEW OF THE CPU



THE 2100 FRONT PANEL



16-BIT REGISTERS

DISPLAY REGISTER. Bit light on = 1, off = 0. Press switch to complement any bit.

MEMORY DATA. Press to display contents of location referenced by M. Lit when selected. Can press again to redisplay unmodified contents. Automatically selected when computer is halted.

INCREMENT M. Press to increment M. If memory data selected, display is updated.

M. Press to display M-register. Can press again to redisplay unmodified contents.

P. Press to display P-register and set Fetch phase. Can press again to redisplay unmodified contents.

B. Press to display B-register. Can press again to redisplay unmodified contents.

A. Press to display A-register. Can press again to redisplay unmodified contents.

S. Press to display S-register. Can press again to redisplay unmodified contents. Automatically selected in run mode.

CLEAR DISPLAY. Press to clear display register.

1-BIT REGISTERS

OVF. Overflow register. Light on = 1, off = 0. Press to complement.

EXTEND. Extend register. Light on = 1, off = 0. Press to complement.

OPERATING CONTROLS

INTERRUPT SYSTEM. Light on indicates interrupt system enabled. Press to complement.

INSTR STEP. Press to execute single instruction.

EXTERNAL PRESET. Press to clear I/O channels.

INTERNAL PRESET. Set Fetch phase, clear parity error indication and overflow, disable interrupt system and memory protect.

HALT/CYCLE. Halt computer or perform one instruction phase.

LOADER ENABLE. Press to enable/disable loader.

RUN. Start execution, disable panel.

POWER OFF/POWER ON/LOCK ON. Key-operated power switch. Panel disabled in LOCK ON position.

PHASE STATUS INDICATORS

FETCH. Indicates Fetch phase is next.

IND. Indicates Indirect phase is next.

EXECUTE. Indicates Execute phase is next.

FAULT INDICATORS

PARITY. Light on indicates that a memory parity error has occurred (if P.E. HALT mode selected).

EXTERNAL PRESET. Light on indicates a power failure occurred (if location 04 contains HLT).

SETTING THE REGISTERS FROM THE FRONT PANEL

A,B,M, MEMORY DATA (T), S, P

1. SELECT THE REGISTER BY PRESSING THE APPROPRIATE BUTTON.

2. THE REGISTER CONTENTS ARE TRANSFERRED TO THE DISPLAY REGISTER.

3. TO MODIFY THE CONTENTS, SET THE DISPLAY REGISTER ACCORDINGLY,
   THEN PRESS ANY OTHER BUTTON (EXCEPT CLEAR DISPLAY, PARITY, FETCH, IND, EXECUTE).

O,E

1. PRESS TO REVERSE STATE.

STORING INTO OR DISPLAYING MEMORY

1. SET THE M REGISTER TO THE DESIRED ADDRESS

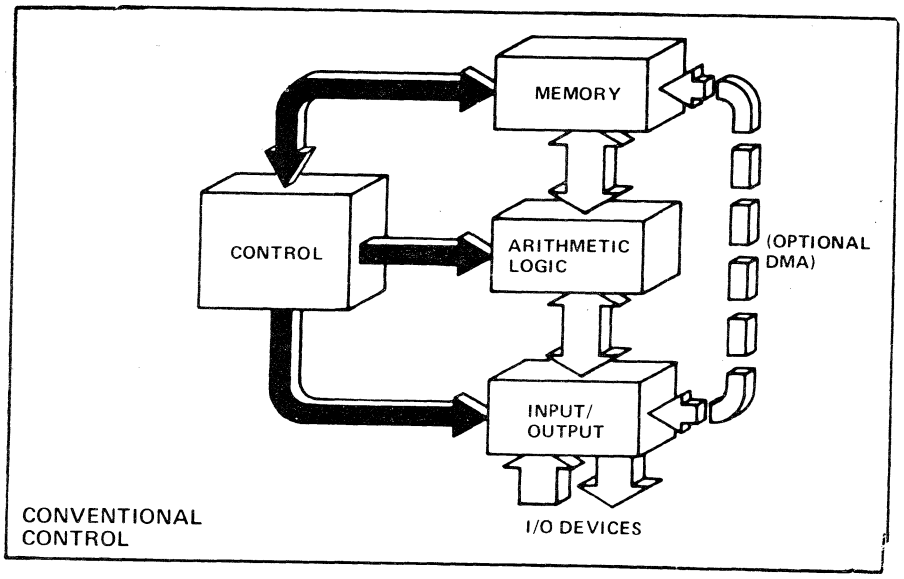
2. PRESS THE MEMORY DATA BUTTON; THE CONTENTS ARE TRANSFERRED INTO THE DISPLAY REGISTER.

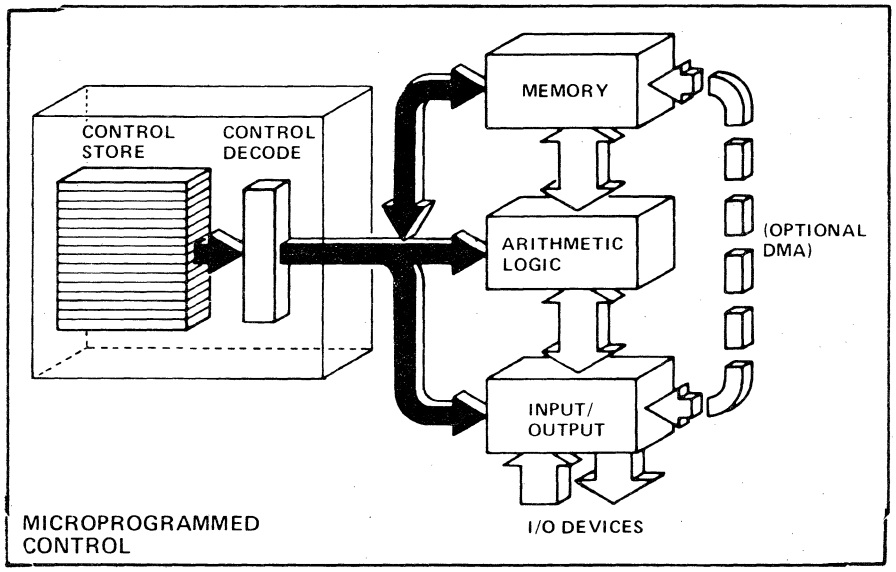
3. SET THE DISPLAY REGISTER TO THE DESIRED CONTENTS.

4. PRESSING INCREMENT OR DECREMENT M TRANSFERS THE DISPLAY REGISTER CONTENTS INTO MEMORY AND STEPS INTO THE NEXT (HIGHER OR LOWER) LOCATION AND DISPLAYS ITS CONTENTS.

5. TO DISPLAY ONLY, SKIP STEP 3.

MICROPROGRAMMED CONTROL





CYCLE TIMES

MEMORY

.980 MICROSECONDS

INSTRUCTION

1.96 MICROSECONDS

ROM

75 NANOSECONDS

MICROINSTRUCTION

196 NANOSECONDS

INSTRUCTION TYPES

1. MEMORY REFERENCE GROUP (MRG)

2. REGISTER REFERENCE
   - SHIFT-ROTATE GROUP (SRG)
   - ALTER-SKIP GROUP (ASG)

3. INPUT/OUTPUT GROUP (IOG)

4. EXTENDED ARITHMETIC GROUP (EAG)

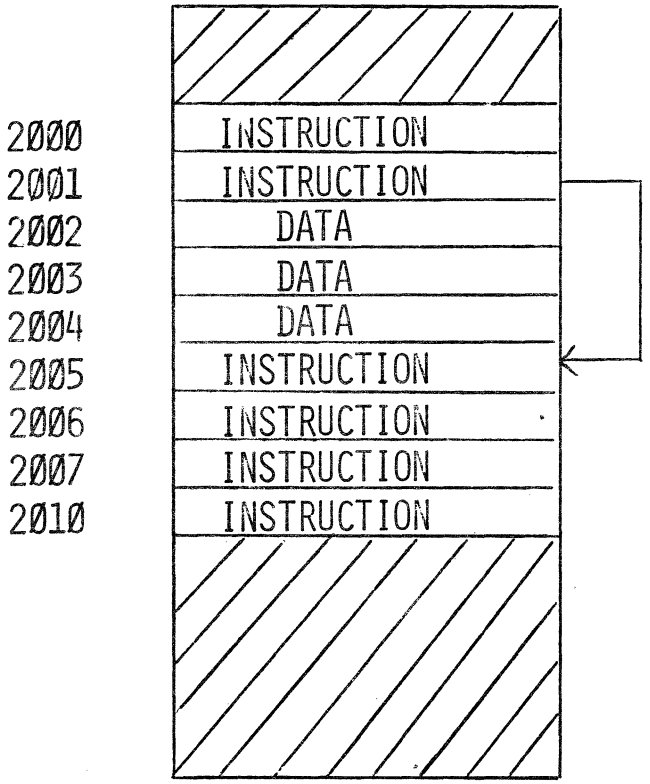
5. MACRO
   - FLOATING POINT
   - FAST FORTRAN PROCESSOR
   - USER WRITTEN

D A T A T Y P E S

1.	INTEGER	3, -4, 32767, -32768
2.	FLOATING POINT	.4, -3 X 10^{5}, 10^{+38}, -10^{+38}
3.	ASCII INFORMATION	NAME,PAYNO,SEX
4.	ADDRESS WORDS	INSTRUCTION → ADDRESS-WORD ← DATA

INSTRUCTION OR DATA?

THE PROGRAMMER MUST ENSURE THAT THE P-REGISTER POINTS TO INSTRUCTIONS ONLY.



S O F T W A R E

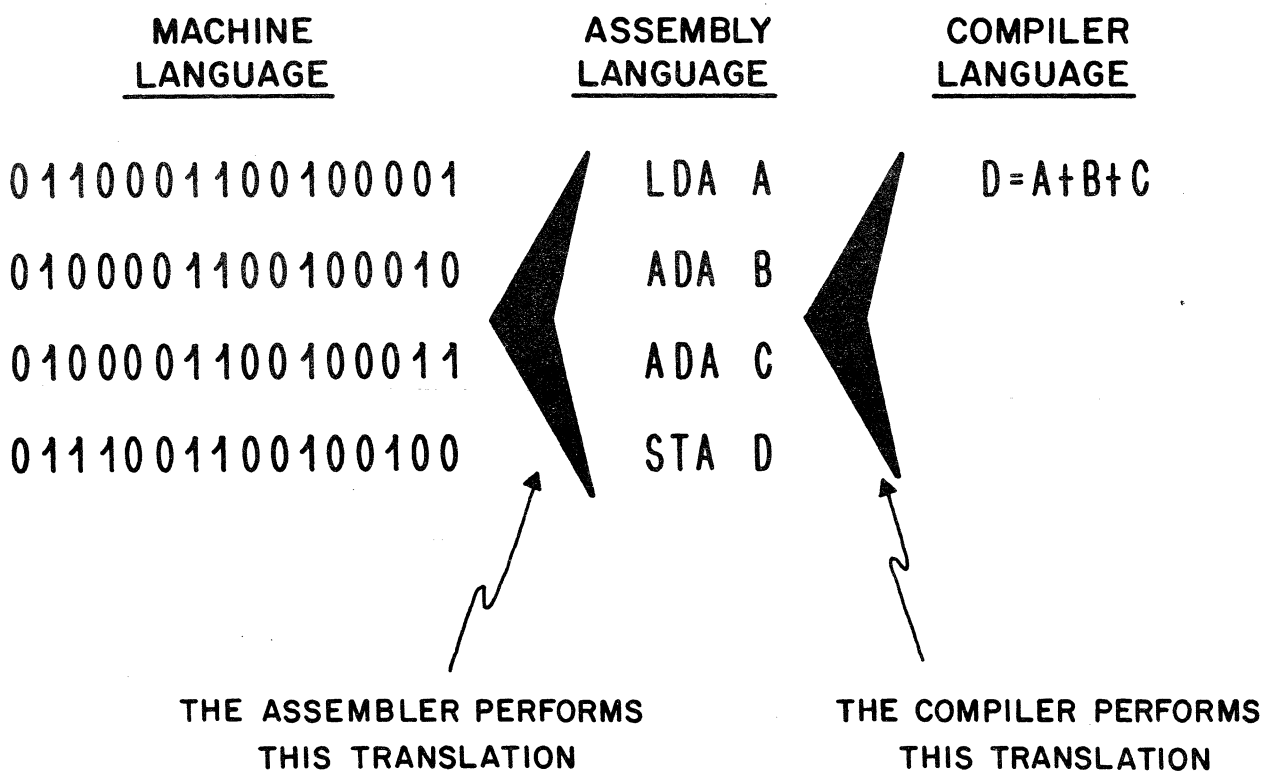
PROGRAMS THAT USE THE HARDWARE TO

SOLVE PROBLEMS!

HP SUPPORTED ⇐

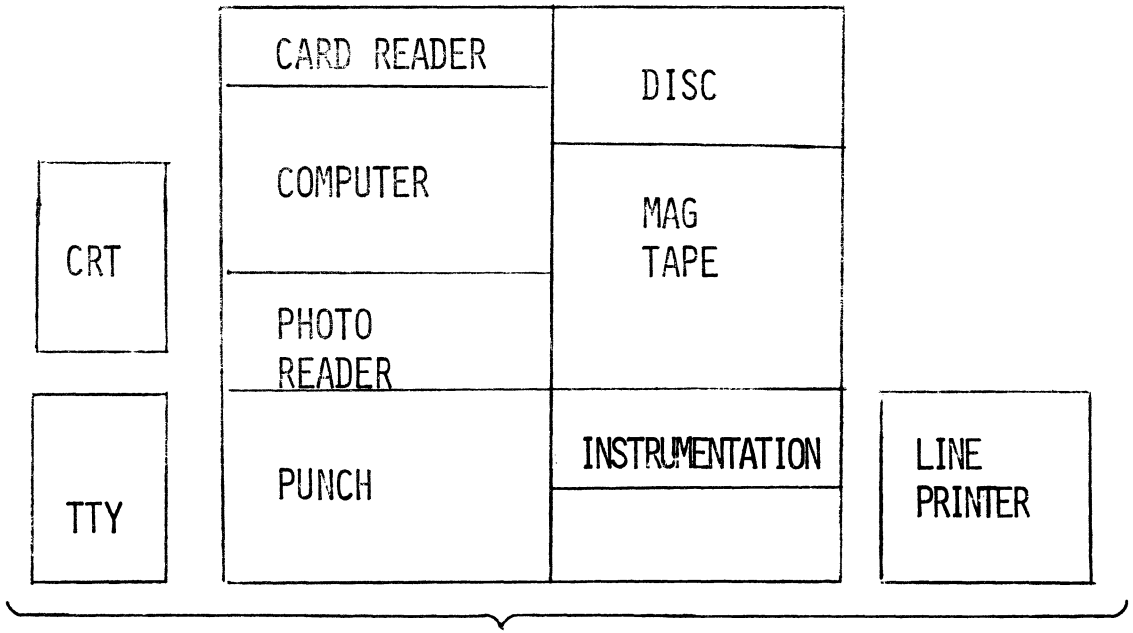
⇒ HP USERS GROUP

TRANSLATORS



OPERATING SYSTEM DEFINITION

A COLLECTION OF HARDWARE AND SOFTWARE DESIGNED TO SIMPLIFY THE TRANSLATION AND EXECUTION OF USER PROGRAMS.



JOB PROCESSOR
RELOCATING LOADER
SCHEDULER
DRIVERS
INPUT/OUTPUT CONTROL
DISC MONITOR
LIBRARIES
COMPILERS
ASSEMBLER
EDITOR
SYSTEM GENERATORS
SIO DRIVERS

SOFTWARE

HP OPERATING SYSTEMS

BCS - BASIC CONTROL SYSTEM

A PAPER TAPE BASED SYSTEM FOR COMPUTERS SYSTEMS WITH NO MASS STORAGE DEVICES.

MTS - MAGNETIC TAPE SYSTEM

THE MTS USED IN CONJUNCTION WITH THE BCS VASTLY INCREASES COMPILATION AND LOADING SPEED.

DOS - DISC OPERATING SYSTEM

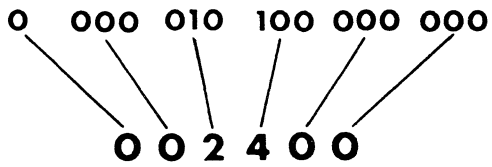
A DISC BASED SYSTEM WHICH PROVIDES BATCH PROCESSING, DISC FILE STORAGE, PLUS ALMOST TOTAL INDEPENDENCE FROM PAPER TAPE HANDLING.

RTE - REAL TIME EXECUTIVE

A REAL TIME SYSTEM ALLOWING MULTIPROGRAMMING (SHARING OF COMPUTER TIME AMONG SEVERAL PROGRAMS).

MNEMONIC CODES

THE MACHINE INSTRUCTION TO CLEAR THE "A" REGISTER IS



THIS INSTRUCTION IS ASSIGNED THE 3 LETTER MNEMONIC CODE CLA.

SIMILARLY EACH COMPUTER INSTRUCTION
IS ASSIGNED A MNEMONIC CODE ——

OTHER EXAMPLES

MNEMONIC	INSTRUCTION	MEANING
CLB	006400	CLEAR "B" REGISTER
CMA	003000	COMPLEMENT "A" REGISTER

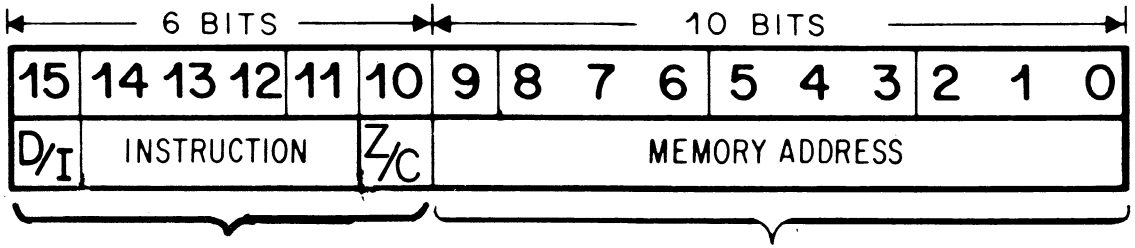
N O P & H L T

NO-OPERATION	HALT
000000	102000
P ← P + 1	P ← P + 1
	COMPUTER HALTS

MEMORY REFERENCE GROUP

INSTRUCTIONS

READING DATA FROM MEMORY
STORING DATA IN MEMORY
ARITHMETIC OPERATIONS
LOGIC OPERATIONS
PROGRAM CONTROL



SELECTS 1 OF 14 INSTRUCTIONS AND DETERMINES ADDRESSING MODE

SPECIFIES THE MEMORY WORD ADDRESS

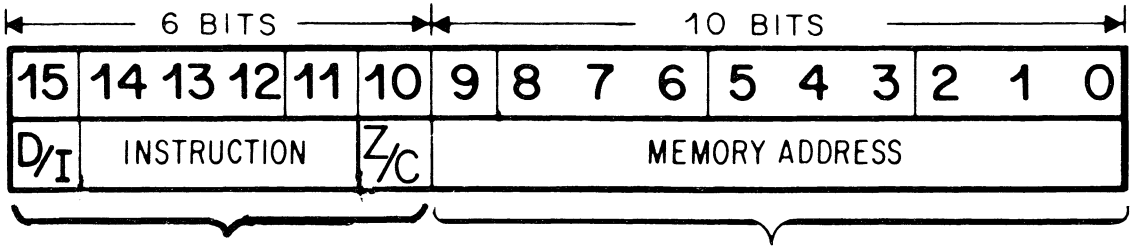
L D A ' A D A ' S T A ' J M P

MEMORY LOCATION	OCTAL CODING	MNEMONIC		
00100	060200	LDA	200B	LOAD A WITH CONTENTS OF 200B
00101	040201	ADA	201B	ADD TO A CONTENTS OF 201B
00102	070202	STA	202B	STORE A INTO LOCATION 202B
00103	102077	HLT	77B	HALT WITH 102077 IN DISPLAY REGISTER
00104	024100	JMP	100B	JUMP TO LOCATION 100B
00200	000001	OCT	1	CONSTANT
00201	000002	OCT	2	CONSTANT
00202	000000	NOP		RESERVE SPACE

MEMORY REFERENCE GROUP

INSTRUCTIONS

READING DATA FROM MEMORY
STORING DATA IN MEMORY
ARITHMETIC OPERATIONS
LOGIC OPERATIONS
PROGRAM CONTROL



SELECTS 1 OF 14 INSTRUCTIONS AND DETERMINES ADDRESSING MODE

SPECIFIES THE MEMORY WORD ADDRESS

N O P & H L T

NO-OPERATION	HALT
000000	102000
P ← P + 1	P ← P + 1
	COMPUTER HALTS

BIT TEST PROBLEM

DO BITS 6-3 OF THE
A REGISTER EQUAL 12_8?

15	14 13 12	11 10 9	8 7 6	5 4 3	2 1 0
0	1 1 0	0 1 1	1 0 ?	? ? ?	0 1 1

AND *m* INSTRUCTION

"LOGICAL AND" THE CONTENTS OF MEMORY LOCATION *m* & THE A-REGISTER.

ISOLATES SPECIFIC BITS

	AND	MASK
MASK	OCT	ØØØ17Ø

If Both Bits=1 then 1
~~If Not~~ Anything else = Ø

A-REGISTER BEFORE "AND *m*" EXECUTED	Ø	11Ø	Ø11	1Ø?	???	Ø11
CONTENTS OF MASK BEFORE "AND *m*" EXECUTED	Ø	ØØØ	ØØØ	ØØ1	111	ØØØ
A-REGISTER AFTER "AND *m*" EXECUTED	Ø	ØØØ	ØØØ	ØØ?	???	ØØØ

CONTENTS OF *m* UNCHANGED

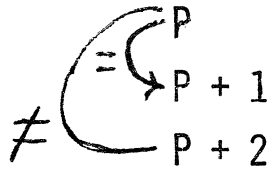
RULE FOR "AND"

1 Ø 1 Ø
1 1 Ø Ø
↓ ↓ ↓ ↓
1 Ø Ø Ø

THE COMPARE INSTRUCTION

COMPARES ALL 16 BITS &
SKIPS IF UNEQUAL

AND MASK
CPA VALUE



MASK OCT 17Ø
VALUE OCT 12Ø

15	14 13 12	11 1Ø 9	8 7 6	5 4 3	2 1 Ø	
Ø	Ø Ø Ø	Ø Ø Ø	Ø Ø ?	? ? ?	Ø Ø Ø	A-REG
Ø	Ø Ø Ø	Ø Ø Ø	Ø Ø 1	Ø 1 Ø	Ø Ø Ø	VALUE

ADDRESSABLE REGISTERS

ADDRESS Ø ↔ A REGISTER

ADDRESS 1 ↔ B REGISTER

```
LDA  1          A ← B
CPB  Ø          COMPARE B WITH A
```

SWITCH REGISTER ACCESS.

Loads the Contents of the Switch register into the A Register.

LIA ~1 I/O address of the Switch Register != Switch register

LIA	1	A-REG ⟵ S-REG	Load into A
OTA	1	A-REG ⟶ S-REG	Output from it

LIA & OTA ARE IOG INSTRUCTIONS

SWITCH REGISTER EXAMPLE

```
100   102501   START   LIA   1        A←S-REG
101   010477           AND   MASK     ISOLATE BITS
102   050500           CPA   VALUE    COMPARE
103   102000           HLT   0        HALT
104   024100           JMP   START    GO TO START

477   000170   MASK    OCT   170
500   000120   VALUE   OCT   120
```

DAY 1 READING ASSIGNMENT

POCKET GUIDE TO THE 2100 COMPUTER

2100A Reference

- Chapter 5, omit 5.2, 5.4, 5.5
- Chapter 3, beginning to 3.2 only
- Chapter 2, para. 2.2 only

BCS

- Chapter 1
- Chapter 2, para. 2.1 only

LAB 1.1 SUM OF NUMBERS

ENTER THE PROGRAM ON 1-40 INTO THE COMPUTER VIA THE FRONT PANEL. EXECUTE THE PROGRAM & VERIFY THAT THE ANSWER IS CORRECT, I.E. CONTENTS OF MEMORY LOCATION 202B = 3,

CONTENTS OF A-REGISTER = 3.

NOTE: EVERYTHING IS TO BE DONE FROM THE FRONT PANEL. DO NOT USE THE ASSEMBLER, BCS, OR EDITOR.

LAB 1.2 SWITCH REGISTER BIT TEST

WRITE A PROGRAM THAT WILL CHECK THE SWITCH REGISTER AND EXECUTE A,

HLT 1 WHEN SWITCH REGISTER BIT Ø IS ON & BIT 1 IS OFF

HLT 2 WHEN SWITCH REGISTER BIT 1 IS ON & BIT Ø IS OFF

HLT 3 WHEN SWITCH REGISTER BITS Ø & 1 ARE BOTH ON.

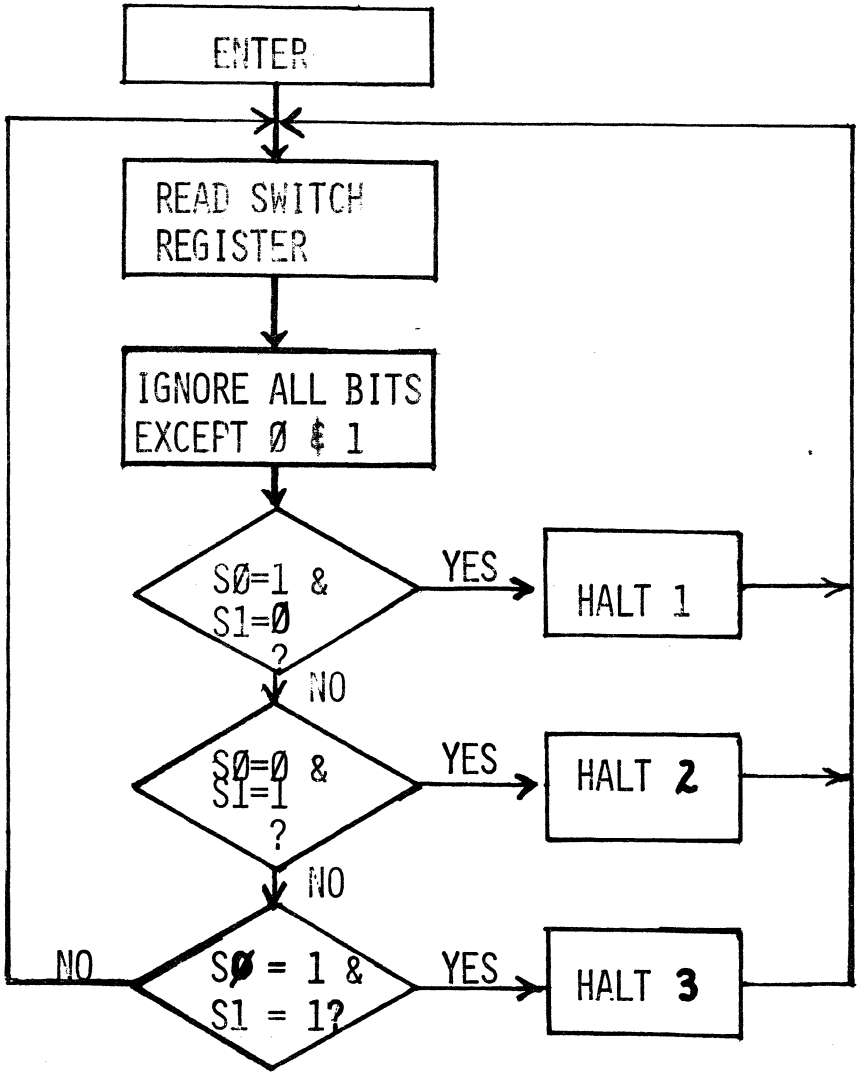
ALL OTHER BITS ARE TO BE IGNORED.

IF BITS Ø & 1 ARE BOTH OFF, THE CPU IS TO REMAIN RUNNING.

AFTER HALTING, PRESSING RUN SHOULD RE-RUN THE PROGRAM.

USE THE ASSEMBLER, BCS, EDITOR AS NEEDED.

LAB 1.2 FLOWCHART



LAB 1.3 SWITCH REGISTER SUM

WRITE A PROGRAM THAT WILL ADD A VALUE IN THE SWITCH REGISTER TO A CONSTANT OF 1ØØØB.

IF THE RESULT < Ø, HLT 1Ø,
= Ø, HLT 2Ø,
> Ø, HLT 3Ø.

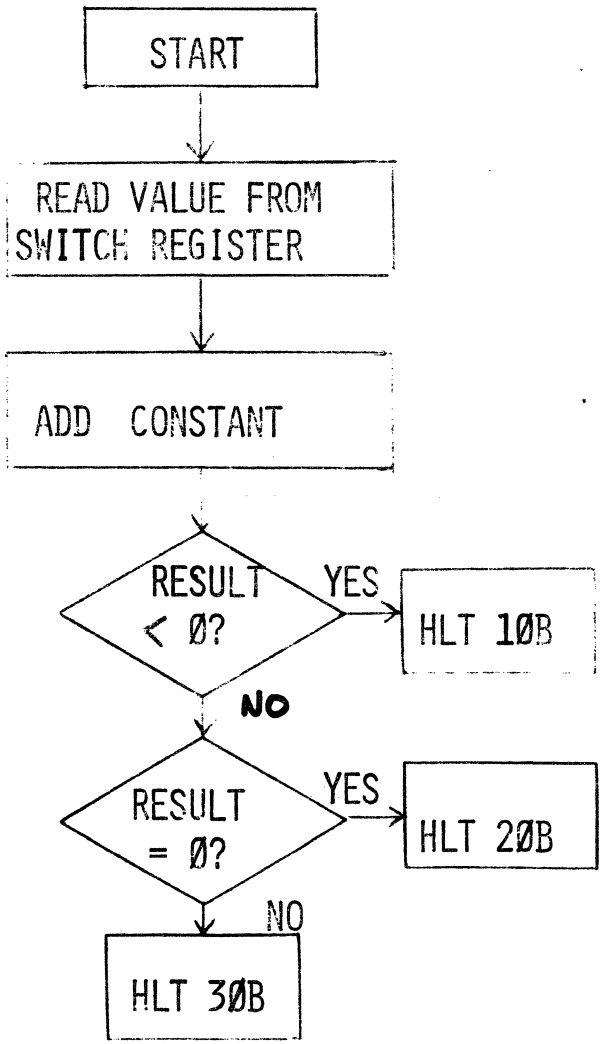
WHEN THE CPU HALTS, THE SUM SHOULD BE IN THE B-REGISTER,

THE ORIGINAL SWITCH REGISTER CONTENTS & THE CONSTANT 1ØØØB ARE TO BE UNALTERED.

THE PROGRAM CODE IS NOT TO BE AUTOMATICALLY RESTARTABLE.

HOW CAN YOU MANUALLY RESTART THE PROGRAM?

LAB 1.3 FLOWCHART



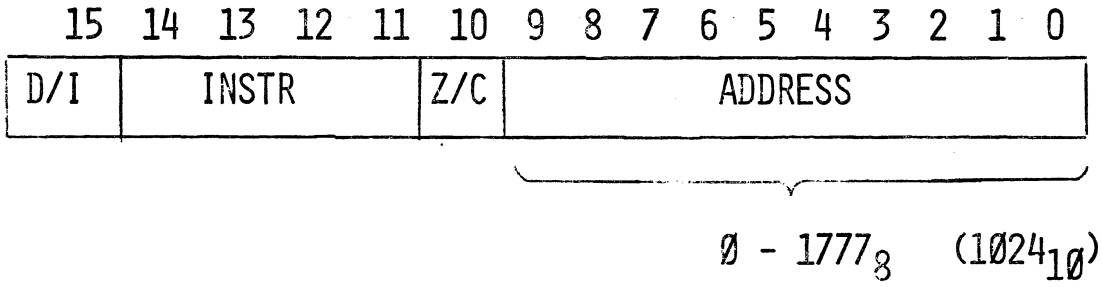
LAB 1.2 SOLUTION

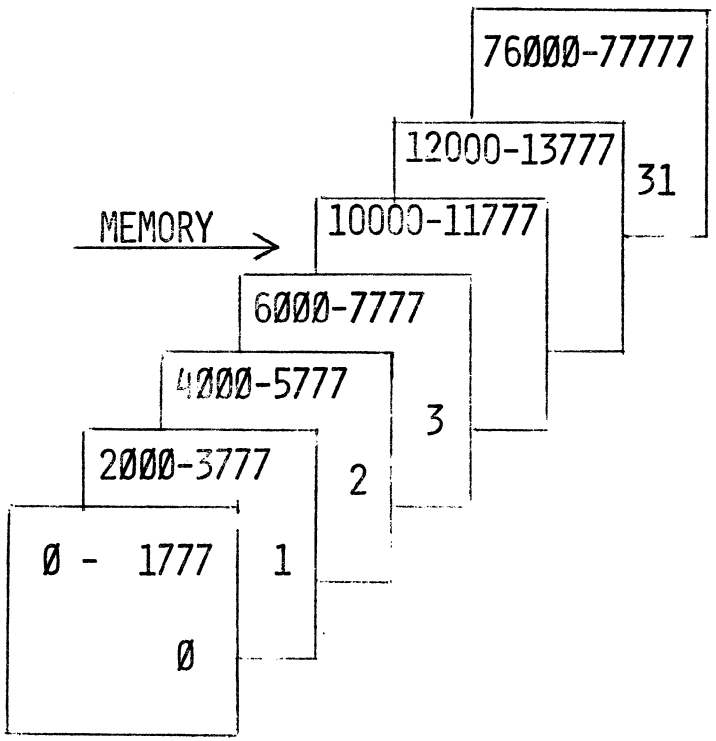
```
0001                    ASMB,R,B,L
0002  00000                     NAM SWHLT    LAB 1.2 SWITCH REGISTER BIT TEST
0003  00000 000003  MASK   OCT 3
0004  00001 000001  FIRST  OCT 1
0005  00002 000002  SECND  OCT 2
0006  00003 000003  BOTH   OCT 3
0007  00004 000000  ENTER  NOP
0008  00005 102501  NEXT   LIA 1          READ SWITCH REGISTER
0009  00006 012000R        AND MASK      IGNORE ALL BITS EXCEPT 0 & 1
0010  00007 052001R        CPA FIRST     S0 = 1 & S1 = 0?
0011  00010 026016R        JMP HLT1        YES, STOP AT HLT1
0012  00011 052002R        CPA SECND     S0 = 0 & S1 = 1?
0013  00012 026020R        JMP HLT2        YES, STOP AT HLT2
0014  00013 052003R        CPA BOTH      S0 = 1 AND S1 = 1?
0015  00014 026022R        JMP HLT3        YES, STOP AT HLT3
0016  00015 026005R        JMP NEXT        NO, TRY AGAIN
0017  00016 102001  HLT1   HLT 1
0018  00017 026005R        JMP NEXT
0019  00020 102002  HLT2   HLT 2
0020  00021 026005R        JMP NEXT
0021  00022 102003  HLT3   HLT 3
0022  00023 026005R        JMP NEXT
0023                        END ENTER
**  NO ERRORS*
```

LAB 1.3 SOLUTION

```
0001                    ASMB,L,R,B
0002   00000                    NAM SWSUM     LAB 1.3 SWITCH REGISTER SUM
0003   00000 000000     START   NOP
0004   00001 106501             LIB 1         READ SWITCH REGISTER
0005   00002 046012R            ADB CONST     ADD CONSTANT
0006   00003 074000             STB 0         SAVE SUM IN B, WORK IN A
0007   00004 012013R            AND NEG       ISOLATE SIGN BIT
0008   00005 052013R            CPA NEG       RESULT < 0?
0009   00006 102010             HLT 10B         YES, HALT 10B
0010   00007 056014R            CPB ZERO      RESULT = 0?
0011   00010 102020             HLT 20B         YES, HALT 20B
0012   00011 102030             HLT 30B         NO,  HALT 30B (RESULT > 0)
0013   00012 001000     CONST   OCT 1000
0014   00013 100000     NEG     OCT 100000
0015   00014 000000     ZERO    OCT 0
0016                            END START
**  NO ERRORS*
```

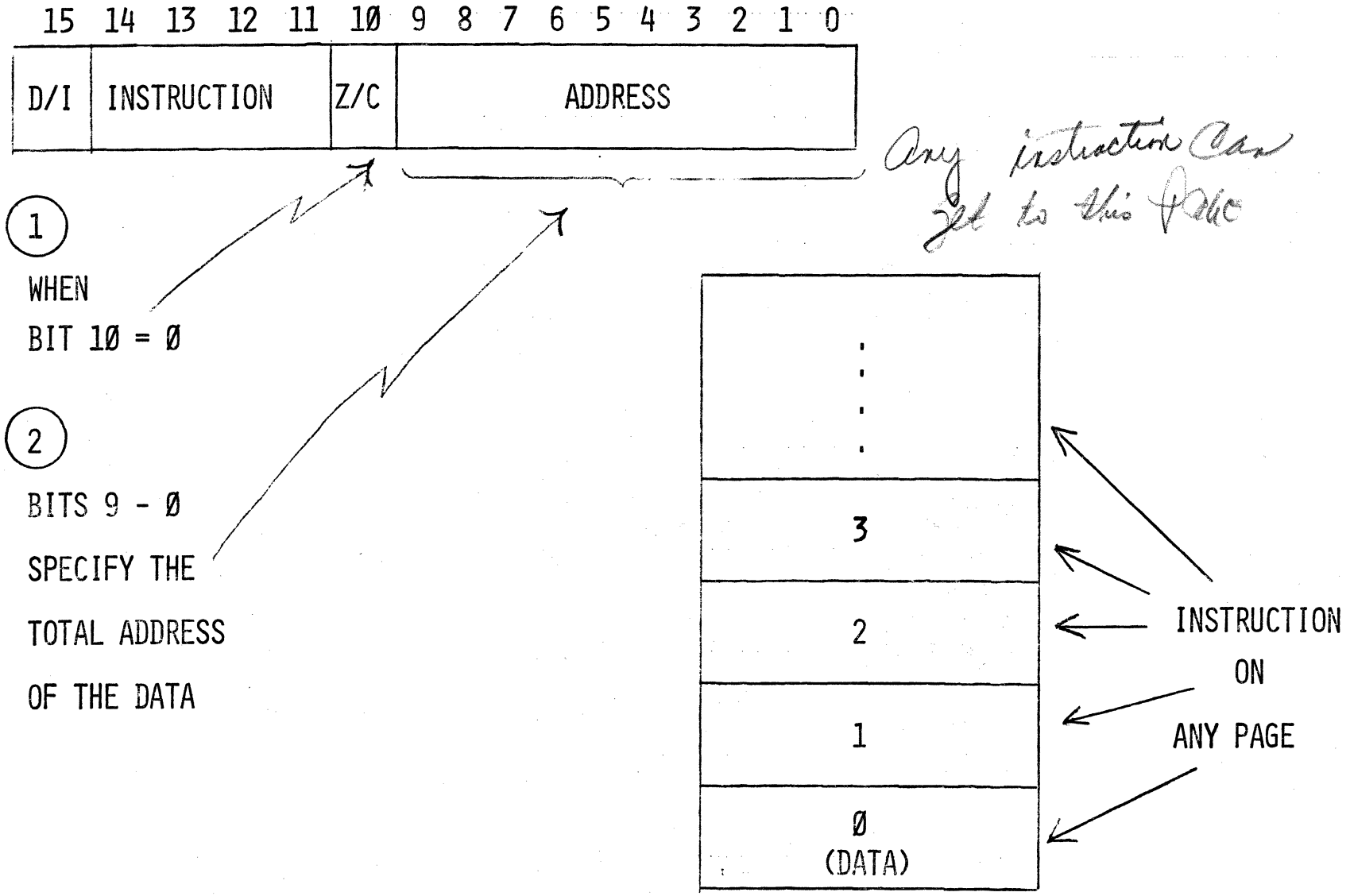
MEMORY ADDRESSING



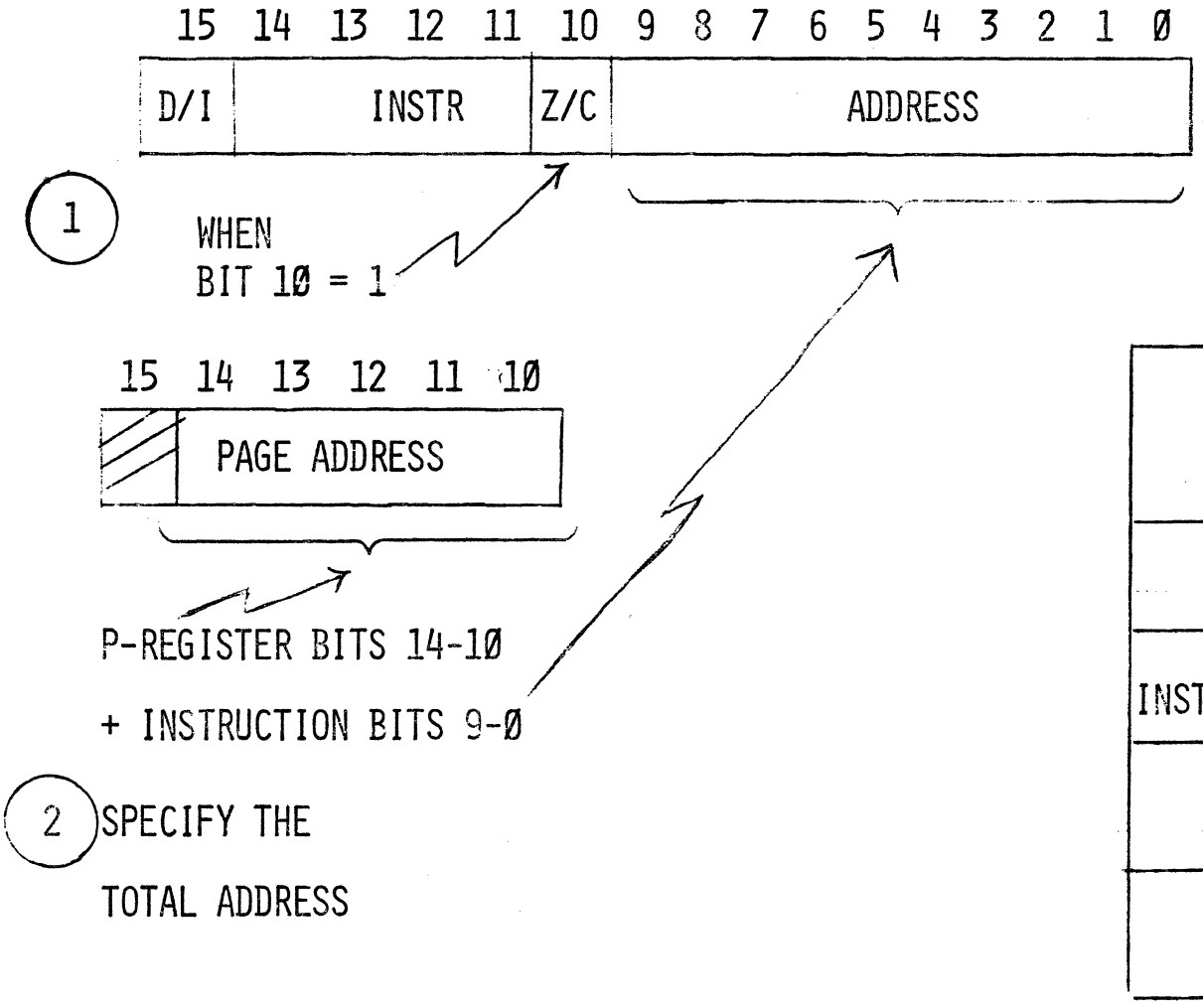


THE "PAGE" CONCEPT

BASE (OR ZERO) PAGE



CURRENT PAGE



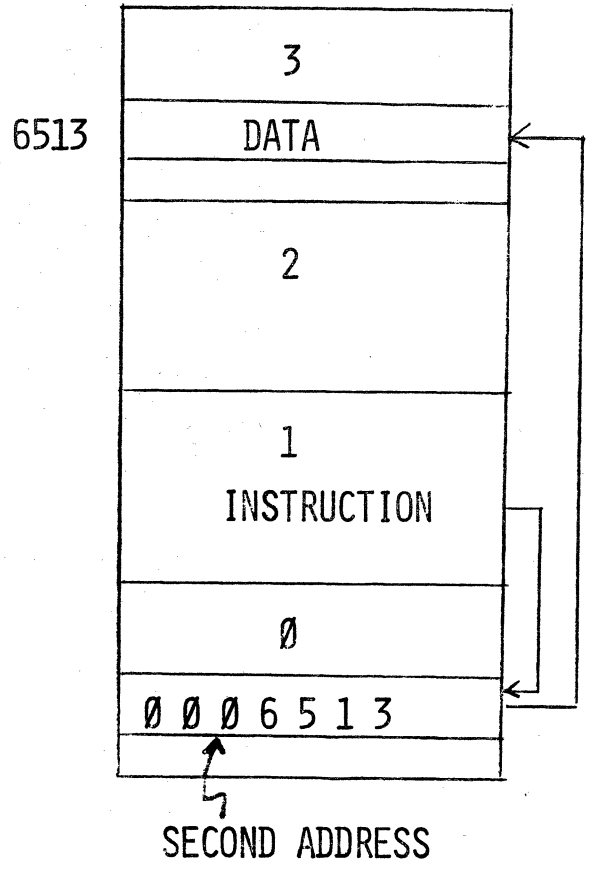
Any instruction that is in the same 1K block as the Current Page

INDIRECT ADDRESSING USING THE BASE PAGE

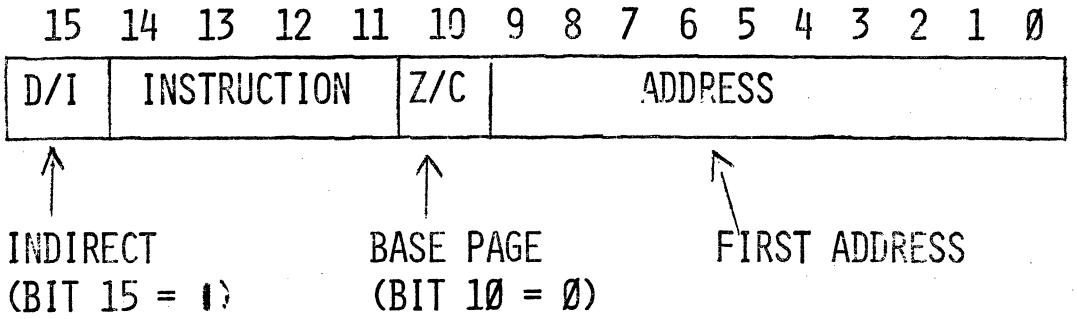
WHAT IF
?

INSTRUCTION
ON PAGE 1

DATA
ON PAGE 3

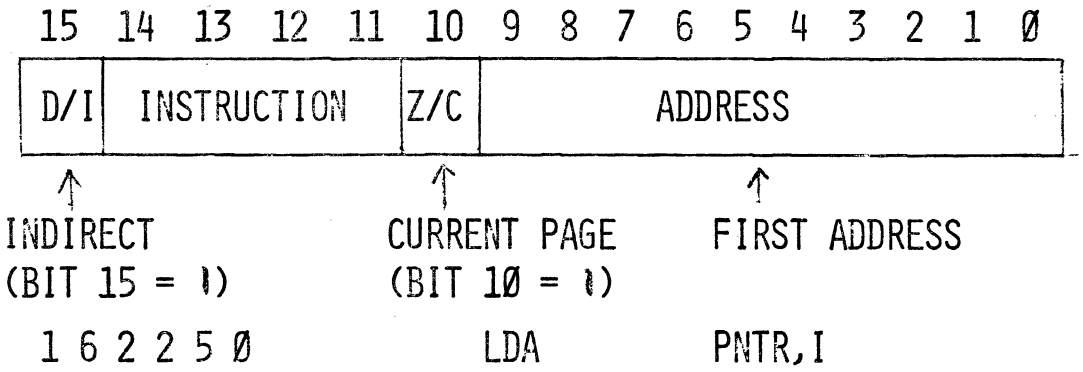


SECOND ADDRESS REQUIRED!

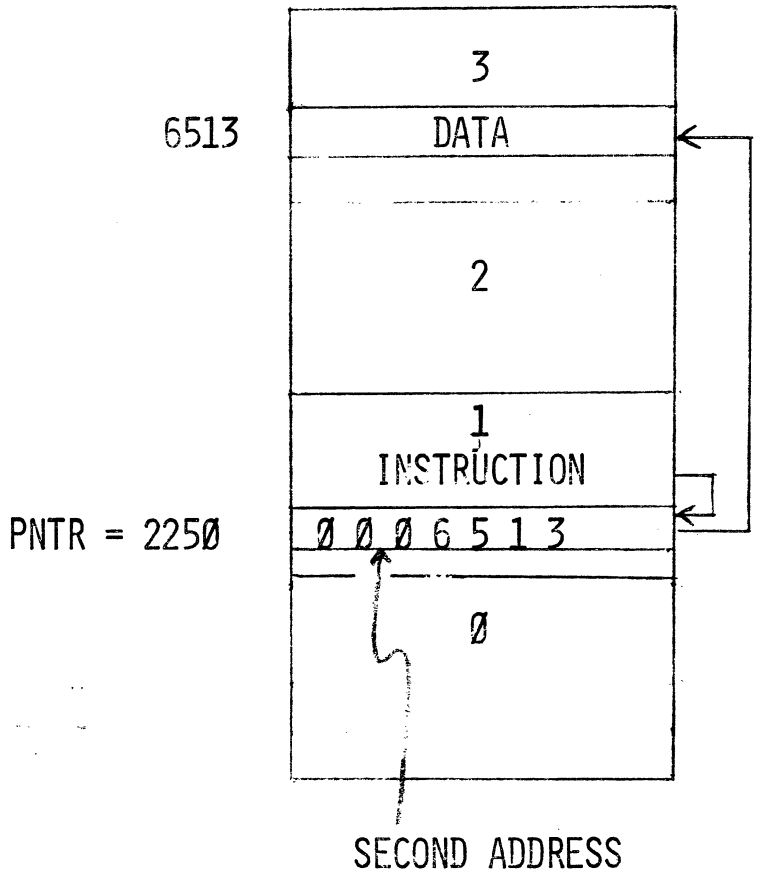


1 6 0 3 0 0 L D A 3 0 0 B , I

INDIRECT ADDRESSING USING THE CURRENT PAGE




```
1 6 2 2 5 Ø        LDA        PNTR,I
```



ADDRESS MODIFICATION PROBLEM

PROBLEM:

WRITE THE INSTRUCTIONS

TO PUT Ø'S IN "ALL"

OF MEMORY

SOLUTION, USING INDIRECT ADDRESSING:

SET A = 5; B = Ø

MEM. ADDR:	MACHINE INSTR:		
00002	174000	STB 0,I	STORE B THRU A
00003	002004	INA	A←A+1
00004	024002	JMP *-2	REPEAT

THE JUMP SUBROUTINE INSTRUCTION (JSB)

THE JUMP SUBROUTINE INSTRUCTION (JSB) PROVIDES A METHOD TO EXECUTE A "SUBROUTINE" AND RETURN TO THE PROPER POINT IN THE "MAIN PROGRAM". TO PERFORM THIS FUNCTION 3 DISTINCT OPERATIONS ARE REQUIRED.

① - PRESERVE THE RETURN ADDRESS. (P+1)

② - TRANSFER CONTROL TO THE SUBROUTINE.

③ - RETURN TO THE "MAIN PROGRAM".

EXAMPLE:

MAIN PROGRAM

LOCATION	LABEL	OP CODE	OPERAND
100		LDA	I
101		JSB	CMP
102		ADA	J
103		HLT	
104	I	OCT	1
105	J	OCT	7

SUBROUTINE

LOCATION	LABEL	OP CODE	OPERAND
			102
200	CMP	~~NOP~~	
201		CMA	
202		INA	
203		JMP	CMP,I

RETURN ADDRESS STORED IN 200B EXECUTION BEGINS WITH 201B.

A JSB EXAMPLE

A SUBROUTINE TO CLEAR THE "A" AND "B" REGISTERS IS SHOWN AS AN EXAMPLE. THE SUBROUTINE IS "ENTERED" FROM 3 DIFFERENT POINTS IN THE "MAIN PROGRAM."

MAIN PROGRAM

LOCATION	LABEL	OP CODE	OPERAND
2001		JSB	CLEAR
2002		INA	
.		.	
.		.	
.		.	
2077		JSB	CLEAR
2100		ADA	J
2101		ADA	K
.		.	
.		.	
.		.	
2500		JSB	CLEAR
2501		HLT	

SUBROUTINE

LOCATION	LABEL	OP CODE	OPERAND
3000	CLEAR	NOP	
3001		CLA	
3002		CLB	
3003		JMP	CLEAR, I

PSEUDO INSTRUCTIONS ENT & EXT

ENTry POINT AND EXTernal PSEUDO INSTRUCTIONS PROVIDE OBJECT PROGRAM LINKAGE CAPABILITY.

FOR EXAMPLE:

THE "MAIN" PROGRAM IS EXECUTED WITH CONTROL PASSING TO THE SUBROUTINE VIA THE EXT AND ENT POINTS.

"MAIN" PROGRAM

```
      ASMB, R, B, L
            NAM TEST
            ENT START
            EXT HERE
START       NOP
              .
              .
              .
            JSB HERE
              .
              .
              .
            END START
```

"SUBROUTINE" PROGRAM

```
      ASMB, R, B, L
            NAM GOOD
            ENT HERE
HERE        NOP
            CLA
              .
              .
              .
            JMP HERE,I
            END
```

A PROGRAM MAY HAVE MULTIPLE ENTry POINTS AND EXTernals.

THE B S S PSEUDO INSTRUCTION

BLOCK STARTING SYMBOL

THIS PSEUDO WILL CAUSE THE ASSEMBLER TO ALLOCATE A BLOCK OF MEMORY LOCATIONS TO A PROGRAM. THE CONTENTS OF THE MEMORY BLOCK CAN NOT BE DETERMINED WHEN THE OBJECT PROGRAM IS LOADED FOR EXECUTION AND MUST BE TAKEN INTO CONSIDERATION BY THE PROGRAMMER.

FOR EXAMPLE:

LOCATION	CONTENTS	LABEL	OP CODE	OPERAND	REMARKS
00500			ORG	500B	
00500	000000	BUFFR	BSS	200	RESERVE 200 (DECIMAL) MEMORY LOCATIONS
01010	002400		CLA		
01011	006400		CLB		
			•		
			•		
			•		

$500_8 + 310_8 = 1010_8$

$310_8 = 200_{10}$

ZERO NOT STORED IN THIS LOCATION

THE DEF PSEUDO INSTRUCTION

THE DEF PSEUDO DEFINES THE MEMORY ADDRESS OF A PROPERLY DEFINED SYMBOL. THE ASSEMBLER GENERATES A 15 BIT MEMORY ADDRESS IN THE OBJECT PROGRAM WHEREVER THE DEF APPEARS.

FOR EXAMPLE:

```
0001                    ASMB,R,B,L
0002  00000                    NAM SRCH
0003  00000 000000  TABLE BSS 100
0004  00144 000000R ADRES DEF TABLE     DEF ADDRESS OF TABLE
0005*
0006  00145 000000  START NOP
0007  00146 066144R       LDB ADRES     GET ADDRESS OF TABLE
0008  00147 160001  NEXT  LDA 1,I       LOAD "A" THRU "B"
0009*                     . . .           (GET TABLE VALUE)
0010  00150 006004        INB           UPDATE TABLE POINTER
0011  00151 026146R       JMP NEXT
0012*                     . . .
0013                      END START
**  NO ERRORS*
```

TECHNIQUES FOR TRANSFER OF PARAMETER DATA TO SUBROUTINES

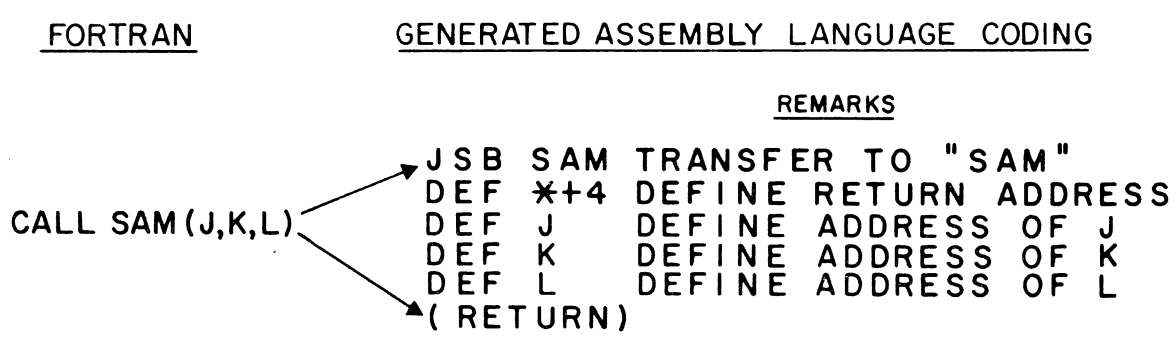
	LABEL	OP CODE	OPERAND	REMARKS
ONE PARAMETER CALLING SEQUENCE		LDA	NUMBR	CONVERT (NUMBR)
		JSB	IABS	TO ITS ABSOLUTE VALUE
		(RETURN)		
TWO PARAMETER CALLING SEQUENCE		LDA	ADDRS	BUFFER ADDRESS IN "A" REGISTER
		LDB	COUNT	COUNT IN "B" REGISTER
		JSB	READ	TRANSFER TO READ ROUTINE
		(RETURN)		
THREE OR MORE PARAMETER CALLING SEQUENCE		JSB	CONVT	TRANSFER TO CONVERT ROUTINE
		OCT	7	CODE WORD
		DEF	BUFFR	BUFFER ADDRESS
		DEF	CNTR	COUNTER
		(RETURN)		

FORTRAN COMPATIBLE ASSEMBLER SUBROUTINES

PART 1 - THE FORTRAN CALL

FORTRAN MAIN PROGRAMS MAY COMMUNICATE WITH ASSEMBLY LANGUAGE SUBROUTINES. THIS FEATURE IS POSSIBLE ONLY IF THE SUBROUTINE IS COMPATIBLE WITH THE STANDARD FORTRAN CALLING SEQUENCE.

FOR EXAMPLE:



THE ACTUAL TRANSFER OF DATA ITEMS IS THE RESPONSIBILITY OF THE SUBROUTINE

FORTRAN COMPATIBLE ASSEMBLER SUBROUTINES

PART 2 - THE ASSEMBLY LANGUAGE SUBROUTINE

A FORTRAN LIBRARY ROUTINE CALLED .ENTR MAY BE USED TO TRANSFER THE ADDRESSES OF THE PARAMETERS FROM THE MAIN PROGRAM TO THE SUBROUTINE.

FOR EXAMPLE:

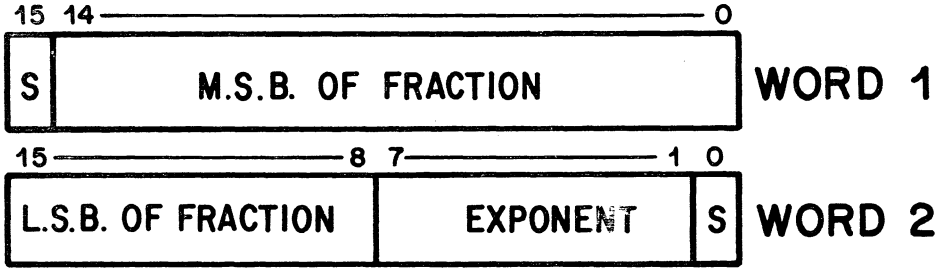
```
LABEL    OPCODE   OPERAND          REMARKS

         NAM      SAM
         ENT      SAM
         EXT      .ENTR
JADDR    BSS      1          LOCATION FOR ADDRESS OF PARAMETER  J
KADDR    BSS      1              "       "       "       "      K
LADDR    BSS      1              "       "       "       "      L
SAM      NOP                 ENTRY POINT
         JSB      .ENTR      TRANSFER TO .ENTR
         DEF      JADDR      DEFINE ADDRESS OF FIRST PARAMETER
         LDA      JADDR,I
         ADA      KADDR,I
         STA      LADDR,I
         JMP      SAM,I      RETURN TO MAIN PROGRAM
```

~~NOTE: MAIN PROGRAM AND SUBROUTINE PARAMETERS MUST AGREE AS TO TYPE:~~

~~REAL OR INTEGER~~

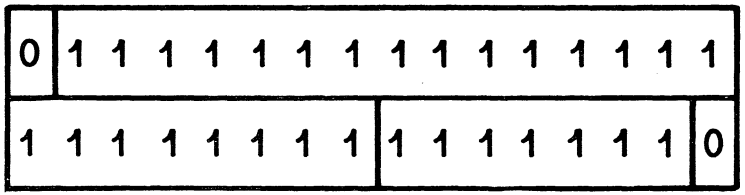
FLOATING POINT NUMBERS

FLOATING POINT NUMBERS USE TWO MEMORY LOCATIONS IN THE FOLLOWING FORM:



FLOATING POINT NUMBERS HAVE A RANGE IN MAGNITUDE OF APPROXIMATELY -10^{38} TO 10^{38}

FOR EXAMPLE: THE GREATEST POSITIVE FLOATING POINT NUMBER WOULD BE:



THE FRACTIONAL VALUE (MANTISSA) OF THE FLOATING POINT NUMBER IS ALWAYS IN THE RANGE $0 \leq n < 1$. IN THE EXAMPLE ABOVE THE VALUE OF THE MANTISSA IS APPROXIMATELY 1. THE EXPONENT VALUE IS 2^7-1 (127_{10}), THE VALUE OF THE NUMBER IS 1×2^{127}; BY RAISING 2 TO THE 127th POWER (USING A LOG TABLE) IT BECOMES APPROXIMATELY 1.7×10^{38}

FLOATING POINT NUMBERS EXAMPLES

```
0001                    ASMB,R,L
0002  00000                     NAM FLTPT
0003  00000 000001  D1    DEC 1          INTEGER 1
0004  00001 040000  F1    DEC 1.         FLOATING POINT 1.0
      00002 000002
0005  00003 040000  F.5   DEC .5         FLOATING POINT 1/2
      00004 000000
0006  00005 054000  F5.5  DEC 5.5        FLOATING POINT 5.5
      00006 000006
0007  00007 177773  DM5   DEC -5         INTEGER -5
0008  00010 130000  FPM5  DEC -5.        FLOATING POINT -5.0
      00011 000006
0009                          END
**  NO ERRORS*
```

EAG & FLOATING POINT INSTRUCTIONS

INSTRUCTION	FUNCTION	OPERATION
MPY	**FIXED POINT MULTIPLICATION**	(A) X (m) → (B± MSB and ALSB)
DIV	**FIXED POINT DIVISION**	(B± MSB and ALSB)/(m) → A, remainder → B
FAD	**FLOATING POINT ADDITION**	(AB) + (m, m+1) → AB
FSB	**FLOATING POINT SUBTRACTION**	(AB) − (m, m+1) → AB
FMP	**FLOATING POINT MULTIPLICATION**	(m, m+1) X (AB) → AB
FDV	**FLOATING POINT DIVISION**	(AB)/ (m, m+1) → AB
DLD	**DOUBLE LOAD**	(m) and (m+1) → A and B
DST	**DOUBLE STORE**	(A) and (B) → m and m+1

LEGEND:

A = Reg. A
B = Reg. B
± = Sign
m = Operand
MSB/LSB = Most/Least Significant Bits

MATH LIBRARY FUNCTIONS

TO PERFORM THE FOLLOWING OPERATIONS:
(FLOATING POINT QUANTITIES)

THE FOLLOWING INSTRUCTIONS CAN BE USED:

```
Y = ABS  (X)  ABSOLUTE VALUE
Y = ATAN (X)  ARCTANGENT
Y = ALOG (X)  NATURAL LOG
Y = COS  (X)  COSINE
Y = EXP  (X)  Y = e^X
Y = SIN  (X)  SINE
Y = SQRT (X)  SQUARE ROOT
Y = TAN  (X)  TANGENT
Y = TANH (X)  HYPERBOLIC TANGENT
```

```
EXT  (NAME)
LDA  X
LDB  X + 1
JSB  (NAME)
STA  Y
STB  Y + 1
```

NOTES

- LIBRARY FUNCTIONS MUST BE DEFINED AS EXTERNAL (EXT) SYMBOLS.
- THEY MAY ONLY BE REFERENCED BY RELOCATABLE PROGRAMS.

EXAMPLE: FIND THE SQUARE ROOT OF QUANTITY "X" AND STORE TO LOCATION "Y"

METHOD 1

```
EXT SQRT
LDA X
LDB X+1
JSB SQRT
STA Y
STB Y+1
```

OR

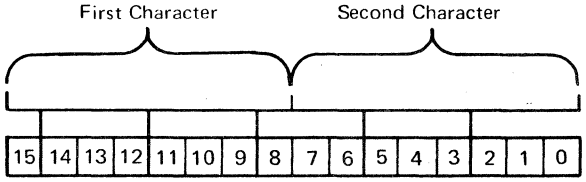
METHOD 2

```
EXT SQRT
DLD X
JSB SQRT
DST Y
```

ASCII CHARACTER CODES

ASCII Character	First Character Octal Equivalent	Second Character Octal Equivalent
A	040400	000101
B	041000	000102
C	041400	000103
D	042000	000104
E	042400	000105
F	043000	000106
G	043400	000107
H	044000	000110
I	044400	000111
J	045000	000112
K	045400	000113
L	046000	000114
M	046400	000115
N	047000	000116
O	047400	000117
P	050000	000120
Q	050400	000121
R	051000	000122
S	051400	000123
T	052000	000124
U	052400	000125
V	053000	000126
W	053400	000127
X	054000	000130
Y	054400	000131
Z	055000	000132
0	030000	000060
1	030400	000061
2	031000	000062
3	031400	000063
4	032000	000064
5	032400	000065
6	033000	000066
7	033400	000067
8	034000	000070
9	034400	000071
space	020000	000040
!	020400	000041
"	021000	000042
#	021400	000043
$	022000	000044
%	022400	000045
&	023000	000046
'	023400	000047
(	024000	000050
)	024400	000051
*	025000	000052
+	025400	000053
,	026000	000054
–	026400	000055
.	027000	000056
/	027400	000057

ASCII Character	First Character Octal Equivalent	Second Character Octal Equivalent
:	035000	000072
;	035400	000073
<	036000	000074
=	036400	000075
>	037000	000076
?	037400	000077
@	040000	000100
[	055400	000133
\	056000	000134
]	056400	000135
↑	057000	000136
←	057400	000137
ACK	036000	000174
①	036400	000175
ESC	037000	000176
DEL	037400	000177
NULL	000000	000000
SUM	000400	000001
EOA	001000	000002
EOM	001400	000003
EOT	002000	000004
WRU	002400	000005
RU	003000	000006
BELL	003400	000007
FE_0	004000	000010
HT/SK	004400	000011
LF	005000	000012
V_{TAB}	005400	000013
FF	006000	000014
CR	006400	000015
SO	007000	000016
SI	007400	000017
DC_0	010000	000020
DC_1	010400	000021
DC_2	011000	000022
DC_3	011400	000023
DC_4	012000	000024
ERR	012400	000025
SYNC	013000	000026
LEM	013400	000027
S_0	014000	000030
S_1	014400	000031
S_2	015000	000032
S_3	015400	000033
S_4	016000	000034
S_5	016400	000035
S_6	017000	000036
S_7	017400	000037



CLEAR · COMPLEMENT · INCREMENT

(CLA/CLB) CLEAR THE INDICATED REGISTER. ALL 16 BITS ARE SET TO Ø.

(CMA/CMB) COMPLEMENT THE CONTENTS OF THE INDICATED REGISTER. ALL Ø'S BECOME 1'S ALL 1'S BECOME Ø'S. THIS IS A 1'S COMPLEMENT.

(INA/INB) INCREMENT THE CONTENTS OF THE INDICATED REGISTER BY 1. OVERFLOW OR EXTEND MAY BE SET BY THIS OPERATION.

'E' REGISTER INSTRUCTIONS

[CLE]

E [1] OR [0] BEFORE

CLEAR THE CONTENTS OF E, E IS RESET TO Ø, THE CONTENTS OF THE OTHER REGISTERS ARE NOT ALTERED BY ANY 'E' REGISTER INSTRUCTIONS.

E [0] [0] AFTER

[CME]

E [0] OR [1] BEFORE

COMPLEMENT E,Ø BECOMES 1,1 BECOMES 0

E [1] [0] AFTER

[CCE]

E [0] OR [1] BEFORE

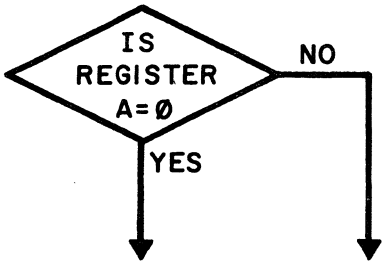
CLEAR AND THEN COMPLEMENT E

E [1] [1] AFTER

DECISION MAKING INSTRUCTIONS

THE ABILITY TO MAKE LIMITED DECISIONS BASED ON PRE-DEFINED CONDITIONS IS VERY IMPORTANT IN COMPUTER PROGRAMS.

FOR EXAMPLE



IN ORDER TO IMPLEMENT THE DECISION SYMBOL ONE OR MORE MACHINE INSTRUCTIONS ARE REQUIRED.

THE INSTRUCTION CODE TO IMPLEMENT THIS DECISION WOULD BE:

SZA (SKIP IF REG."A" IS ZERO)

```
LABEL  OPCODE  OPERAND    REMARKS
         :
         :
       SZA                IS REG A=Ø?
       JMP     N ZERO     NO
       JMP       ZERO     YES
```

NOTE: ALL HP COMPUTER "SKIP-TYPE INSTRUCTIONS WORK IN THIS MANNER.

SKIP INSTRUCTIONS

(SZA/SZB) SKIP THE NEXT SEQUENTIAL INSTRUCTION IF ALL 16 BITS OF THE INDICATED REGISTER ARE ZERO'S (Ø'S). ANY OTHER CONDITION CAUSES THE NEXT SEQUENTIAL INSTRUCTION TO BE EXECUTED.

(SSA/SSB) SKIP THE NEXT SEQUENTIAL INSTRUCTION IF BIT 15 (SIGN BIT) IS EQUAL TO ZERO.

(SLA/SLB) SKIP THE NEXT SEQUENTIAL INSTRUCTION IF BIT Ø IS EQUAL TO ZERO.

(SEZ) SKIP THE NEXT SEQUENTIAL INSTRUCTION IF THE "E" REGISTER IS Ø.

SKIP EXAMPLE

PROBLEM:

READ A 16 BIT VALUE FROM THE CONSOLE SWITCH REGISTER. IF THE VALUE IS POSITIVE TAKE THE 2's COMPLEMENT. IF THE VALUE IS NEGATIVE, CONTINUE THE PROGRAM.

SOLUTION:

Label	Operation	Operand	Remarks
	LIA	1	READ SWITCH REGISTER
	SSA		IS VALUE POSITIVE?
	JMP	CONT	NO
	CMA		YES, TAKE COMPLEMENT
	INA		ADD ONE
CONT	———		CONTINUE PROGRAM

RSS INSTRUCTION

RSS — REVERSE THE SKIP 'SENSE' FOR ALL SKIP INSTRUCTIONS. AN RSS USED WITH A SKIP INSTRUCTION COMPLEMENTS THE SKIP CONDITION.

EXAMPLES:

RSS = UNCONDITIONAL SKIP

SEZ, RSS = SKIP IF E ≠ Ø

SZA, RSS = SKIP IF A ≠ Ø

SLB, RSS = SKIP IF LSB OF B ≠ Ø

SSA, RSS = SKIP IF MSB OF A ≠ Ø

SEZ, SSA, RSS = SKIP MSB OF A ≠ Ø OR $\overline{E \neq Ø}$

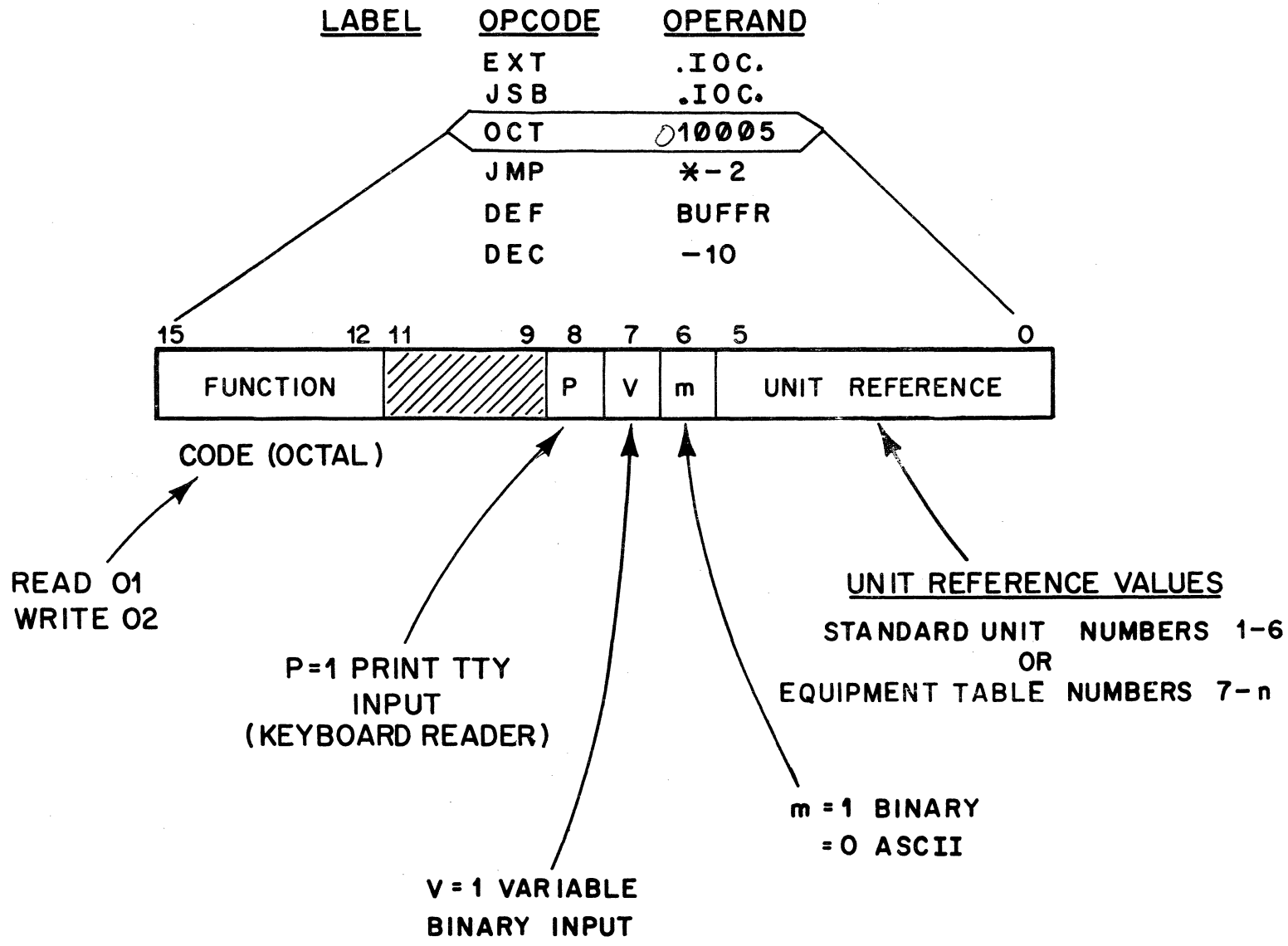
INPUT OUTPUT REQUESTS

```
        EXT     .IOC.
        JSB     .IOC.
        OCT     (<FUNCTION><SUBFUNCTION><UNIT REF.>)
        JMP     (REJECT ADDRESS)
        DEF     (BUFFER ADDRESS)
        DEC     (BUFFER LENGTH OR COUNT)
        PROGRAM CONTINUATION
```

DRIVER OR DEVICE BUSY
ILLEGAL CALL OR DMA
CHANNEL NOT AVAILABLE....
IOC WILL RETURN HERE

IF THE REQUEST IS ACCEPTED
IOC WILL RETURN TO THE
LOCATION FOLLOWING THE
COUNT PARAMETER

IOC CALL (PARAMETER 1)



STANDARD BCS LOGICAL UNIT ASSIGNMENTS

LOGICAL UNIT	FUNCTION	TYPICAL DEVICE
1	KEYBOARD	TTY
2	TELEPRINTER	TTY
3	LIBRARY	PHOTOREADER
4	PUNCH	PUNCH
5	DATA INPUT	PHOTOREADER
6	LIST	LINE PRINTER

COMMAND REJECT

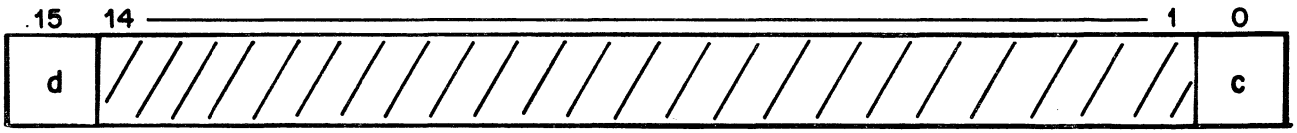
AN I/O REQUEST THAT IS REJECTED WILL CAUSE IOC TO RETURN CONTROL HERE

THE CAUSE OF THE REJECT WILL BE RETURNED IN THE B REGISTER.

```
LABEL   OPCODE   OPERAND
        JSB      .IOC.
        OCT      XXXXX
        JMP      *-2
        DEF      ADDR
        DEC      COUNT
```

A-REGISTER

B-REGISTER



d = 1 DEVICE OR DRIVER BUSY

c = 1 DMA CHANNEL NOT AVAILABLE

d = c = Ø ILLEGAL FUNCTION OR SUBFUNCTION

EXAMPLE CALL TO IOC

```
LABEL          OP CODE        OPERAND        REMARKS

ASMB,R,B,L
               NAM            SHOW
               EXT            .IOC.
               ENT            GO
GO             NOP
                 •
                 •
               JSB            .IOC.          GO TO INPUT/OUTPUT CONTROL
               OCT            10005          READ ASCII, STD INPUT
               JMP *-2
               DEF TABLE                     ADDRESS OF BUFFER
               DEC -20                       NUMBER OF CHARACTERS
                 •
                 •
                 •
TABLE          BSS 10                        RESERVE 10 WORDS
                 •
                 •
                 •
               END
```

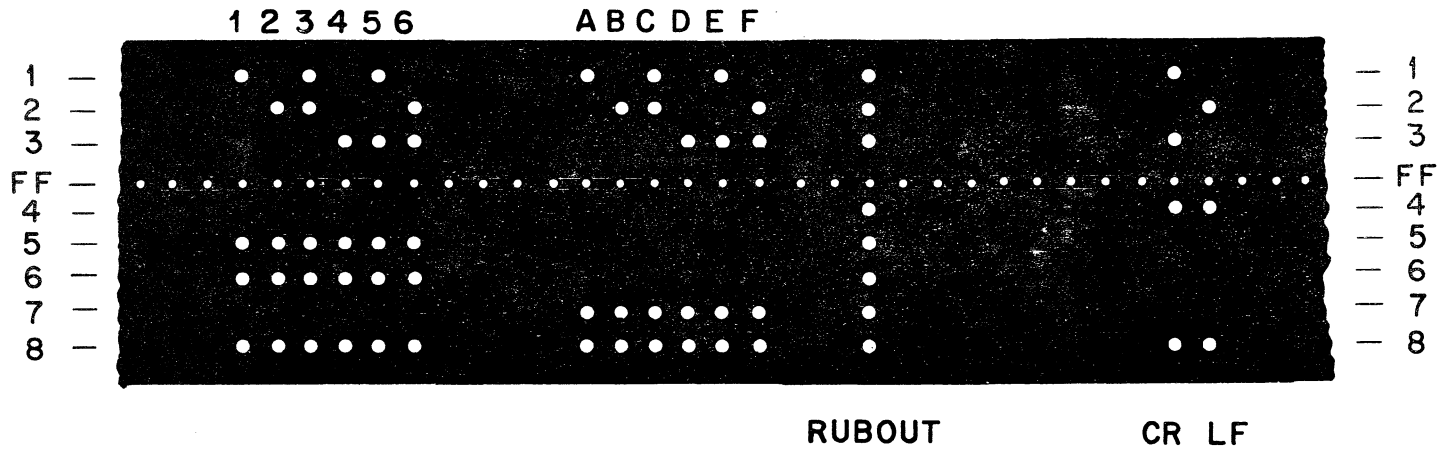
1 - THE ADDRESS OF THE FIRST WORD OF THE BUFFER IS DEFINED USING THE DEF PSEUDO.

2 - THE NUMBER OF WORDS/CHARACTERS TO BE TRANSFERRED IS DEFINED BY:

WORDS = A POSITIVE VALUE

CHARACTERS = A NEGATIVE VALUE

(TELETYPE) 8 LEVEL TAPE



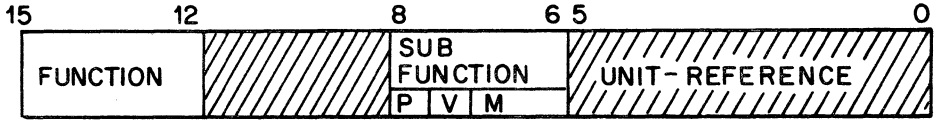
1) **LEVELS 1 - 7 CONSTITUTE THE ASCII CODE**

2) **THE DRIVERS WITHIN BCS REMOVE THE 8th LEVEL ON INPUT AND PUNCH THE 8th LEVEL ON OUTPUT WHEN READING/WRITING**

3) **CARRIAGE RETURN, LINE FEED IS THE ASCII END OF RECORD MARK**

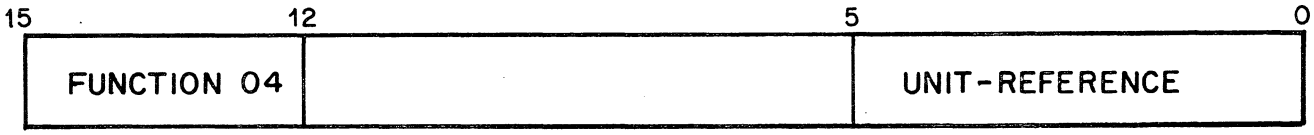
4) **A 'RUB OUT' CHARACTER FOLLOWED BY C.R., L.F. WILL CAUSE THE DRIVER TO IGNORE THE PRECEDING RECORD**

ALLOWABLE COMBINATIONS OF READ/WRITE FUNCTIONS



OPERATION				
READ ASCII RECORD	0	1	0	0
READ ASCII RECORD AND PRINT	0	1	0	4
READ BINARY RECORD	0	1	0	1
READ VARIABLE LENGTH BINARY RECORD	0	1	0	3
WRITE ASCII RECORD	0	2	0	0
WRITE BINARY RECORD	0	2	0	1

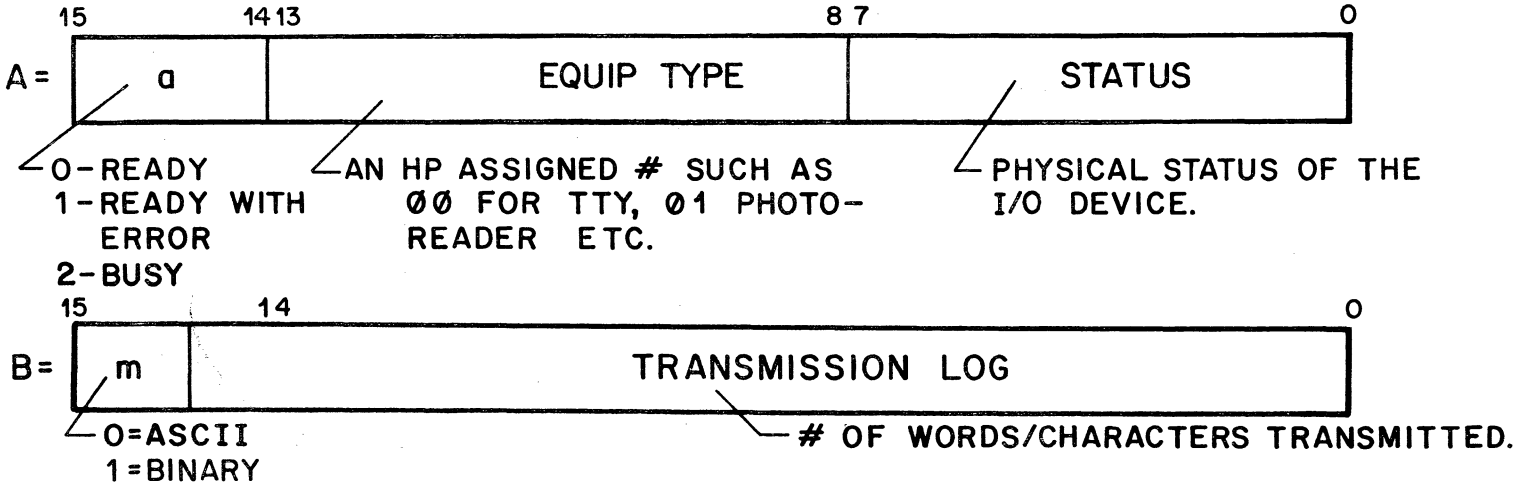
STATUS REQUEST (DEVICE)



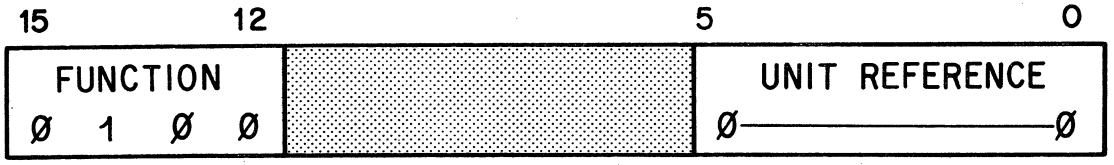
(CALLING SEQUENCE)

```
JSB     .IOC.
OCT     <FUNCTION> <UNIT-REF>
```

IOC WILL RETURN TO THE MAIN PROGRAM WITH WORD 2 OF THE EQT IN THE A REGISTER, AND WORD 3 OF THE EQT IN THE 'B' REGISTER.



STATUS REQUEST (SYSTEM)



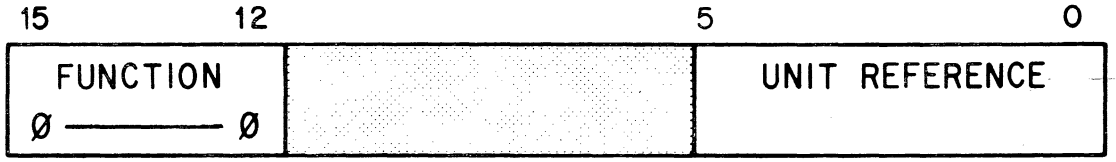
CALLING SEQUENCE

```
JSB .IOC.
OCT 40000
(RETURN)
```

IOC WILL RETURN TO THE MAIN PROGRAM WITH REGISTER "A":

POSITIVE - No system devices are busy
NEGATIVE - A system device is busy

CLEAR REQUESTS



(CALLING SEQUENCE)

```
JSB     .IOC.
OCT     (FUNCTION) (UNIT-REF)
```

THE ONLY OTHER PARAMETER REQUIRED IS THE UNIT-REFERENCE NUMBER.

THE CLEAR REQUEST TERMINATES A PREVIOUSLY ISSUED INPUT OR OUTPUT OPERATION BEFORE ALL DATA IS TRANSMITTED.

A SYSTEM CLEAR REQUEST (OCT Ø) CAUSES IOC TO CLEAR ALL DEVICES AND DRIVERS DEFINED BY THE EQUIPMENT TABLE. THIS MAKES ALL DEVICES AVAILABLE FOR INITIATING AN OPERATION, AND IS USUALLY ISSUED AT THE BEGINNING OF A PROGRAM.

MESSAGE OUTPUT EXAMPLE

```
LABEL     OP CODE     OPERAND          REMARKS
                •
          ENT         START
          EXT         .IOC.
START     NOP
          JSB         .IOC.            SYSTEM CLEAR
          OCT         Ø
          - - - - - - - - - - - - - - -
          JSB         .IOC.            WRITE ON UNIT #2
          OCT         2ØØØ2              WRITE ASCII
          JMP         *-2                REJECT, RETRY
          DEF         MESSG            THIS MESSAGE
          DEC         -37              CHARACTER LENGTH
          - - - - - - - - - - - - - - -
          JSB         .IOC.            STATUS CHECK UNIT #2
          OCT         4ØØØ2
          SSA                          WAIT UNTIL
          JMP         *-3              FINISHED
          - - - - - - - - - - - - - - -
          JSB         .IOC.            READ FROM UNIT #1
          OCT         1Ø4Ø1              READ ASCII & PRINT
          JMP         *-2                REJECT, RETRY
          DEF         INBUF            STORE DATA IN BUFFER
          DEC         -13              CHARACTER LENGTH
                •
                •
                •
                •
MESSG     ASC         19, PLEASE ENTER DATE & TIME  3/27/65/Ø624
INBUF     BSS         7
                •
                •
          END         START
```

PROGRAMS TO BE RUN UNDER EXPANDED BCS PROCEDURE

```
0001                        ASMB,R,B,L
0002  00000                       NAM MAIN
0003                              ENT BEGIN
0004                              EXT SUB
0005  00000 000000  BEGIN NOP
0006  00001 102501              LIA 1
0007  00002 016001X             JSB SUB
0008  00003 102077              HLT 77B
0009  00004 026001R             JMP BEGIN+1
0010                              END BEGIN
**  NO ERRORS*
```

```
0001                        ASMB,R,B,L
0002  00000                       NAM SBRTN
0003                              ENT SUB
0004                              EXT FLOAT
0005  00000 000000  SUB    NOP
0006  00001 016001X             JSB FLOAT
0007  00002 126000R             JMP SUB,I
0008                              END
**  NO ERRORS*
```

SAMPLE EXPANDED BCS PROCEDURE CONSOLE LISTING

EXECUTION

```
 MAIN

 02000 02004

*LOAD

  SBRTN

 02005 02007

*LOAD

  FLOAT

 02010 02014

  .PACK

 02015 02111

 00100 00107

*LST
.IOC.   16026
.SQT.   16003
.MEM.   15775
.BUFR   16177
HALT    15770
BEGIN   02000
SUB     02005
FLOAT   02010
.PACK   02015

  *LINKS
  01775 01777

*RUN
```

ABSOLUTE TAPE PUNCHED

```
 MAIN

 02000 02004

*LOAD

  SBRTN

 02005 02007

*LOAD

  FLOAT

 02010 02014

  .PACK

 02015 02111

 00100 00107

*LST
.IOC.   16026
.SQT.   16003
.MEM.   15775
.BUFR   16177
HALT    15770
BEGIN   02000
SUB     02005
FLOAT   02010
.PACK   02015

  *LINKS
  01775 01777

*END
```

DAY 2 READING ASSIGNMENT

POCKET GUIDE TO THE 2100 COMPUTER

ASSEMBLER

CHAPTER 2, PAGES 2-11 THRU 2-14 ONLY
CHAPTER 4, PAGES 4-17, 4-22 THRU 4-25 ONLY
CHAPTER 5, PAGES 5-1, 5-2 ONLY

BCS

CHAPTER 2, PARA. 2.2 THRU 2.5 ONLY

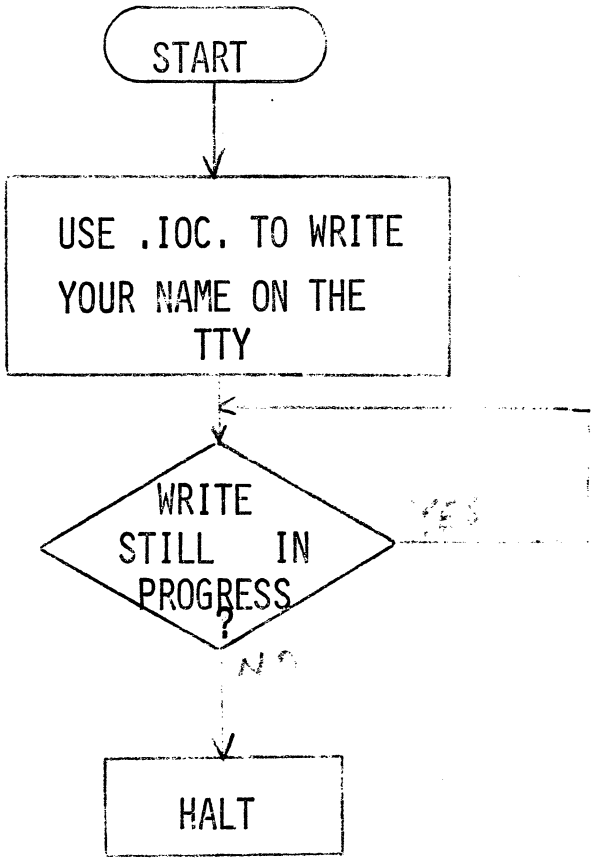
FORTRAN

CHAPTER 7, PARA. 7.3, PAGE 7-4 AND FIRST PARAGRAPH ON 7-5 ONLY

LAB 2.1 WRITE NAME ON TTY

WRITE A PROGRAM THAT USES A CALL TO .IOC. TO WRITE YOUR NAME ON THE TTY.

LAB 2.1 FLOWCHART



LAB 2.1 CONSOLE RUN SHEET

```
  L21JH

 02000 02020

*LOAD

*LST

*RUN
DAVE PACKARD
```

LAB 2.2 SUBROUTINE COMMUNICATIONS

PUNCH SOURCE TAPES FOR THE MAIN PROGRAM & SUBROUTINE ON SLIDE 2-37

AFTER SUCCESSFULLY EXECUTING THE PROGRAMS, USE BCS TO PUNCH AN ABSOLUTE TAPE. RERUN THE PROGRAMS USING THE ABSOLUTE TAPE.

REFER TO SLIDE 2-40 FOR THE CONSOLE RUN SHEETS.

LAB 2.3 FIND ERRORS ON TAPE

A paper tape is to be searched for error messages. Control characters within each record are examined to determine error records.

The specific problem requirements are outlined in the flowchart, however, the following information is also provided.

a. Read a data message consisting of twenty characters.

b. If characters 15 & 16 are equal to ** print ERROR MESSAGE on the teleprinter.

c. If characters 1 & 2 are equal to @@ the end of the message tape has been reached — halt the computer.

d. If neither b or c are met, read the next record.

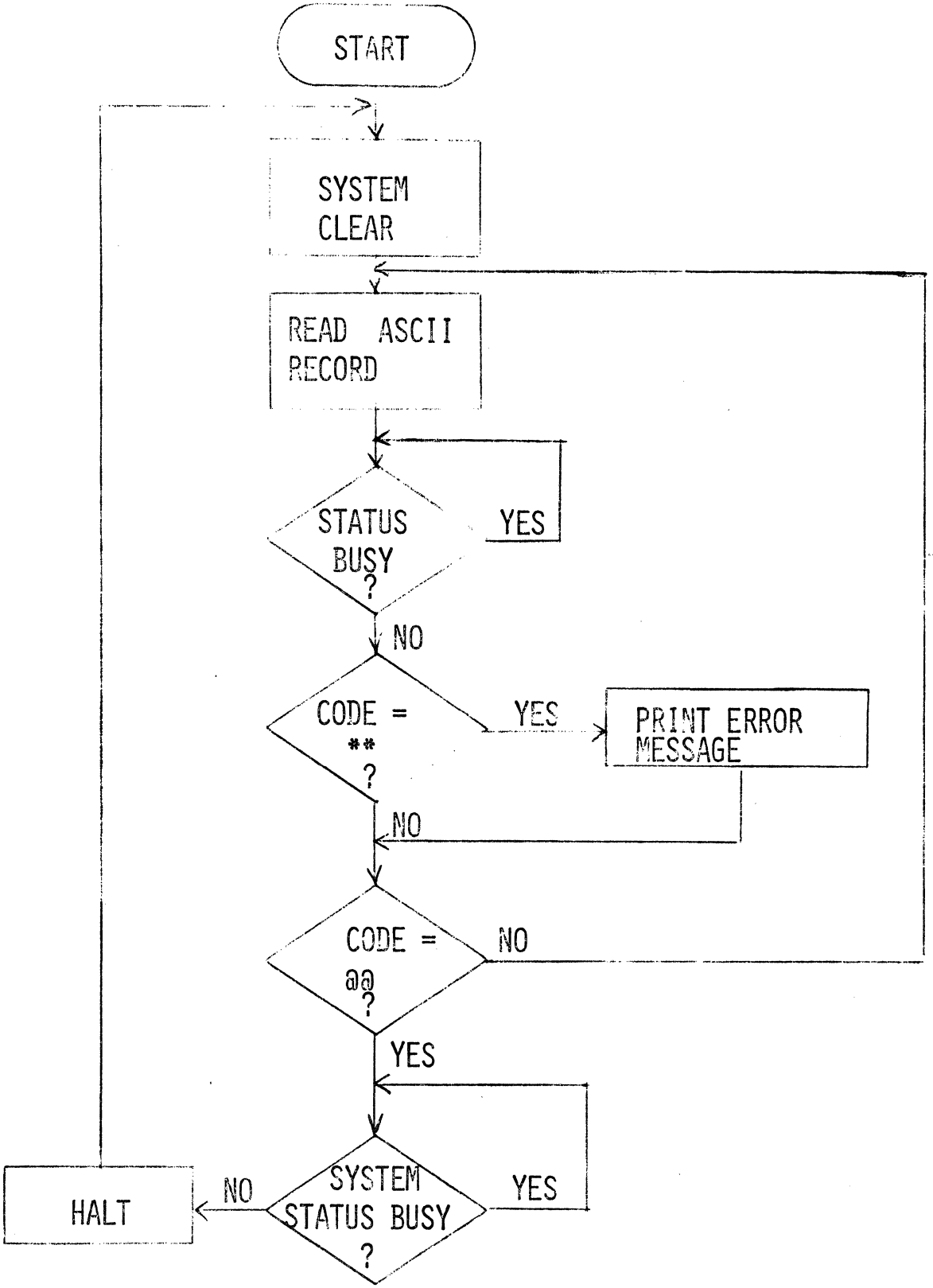
e. Use Unit #5 for input and Unit #2 for output.

f. A system status check should be made to insure that all output operations are complete before the computer is brought to a halt.

RESULTS

ERROR MESSAGE should be printed 10 times.

LAB 2.3 FLOWCHART



LAB 2.3 CONSOLE RUN SHEET

```
L23JH

02000 02062

*LOAD

*LST

*RUN
ERROR MESSAGE
ERROR MESSAGE
ERROR MESSAGE
ERROR MESSAGE
ERROR MESSAGE
ERROR MESSAGE
ERROR MESSAGE
ERROR MESSAGE
ERROR MESSAGE
ERROR MESSAGE
```

LAB 2.4 LIMIT CHECK SUBROUTINE

WRITE A FORTRAN CALLABLE SUBROUTINE WHICH WILL COMPARE A TEST VALUE AGAINST UPPER & LOWER LIMITS AND SET THE A-REGISTER TO

+1 IF THE TEST VALUE IS $\geq$ UPPER LIMIT,

Ø IF THE TEST VALUE IS WITHIN LIMITS,

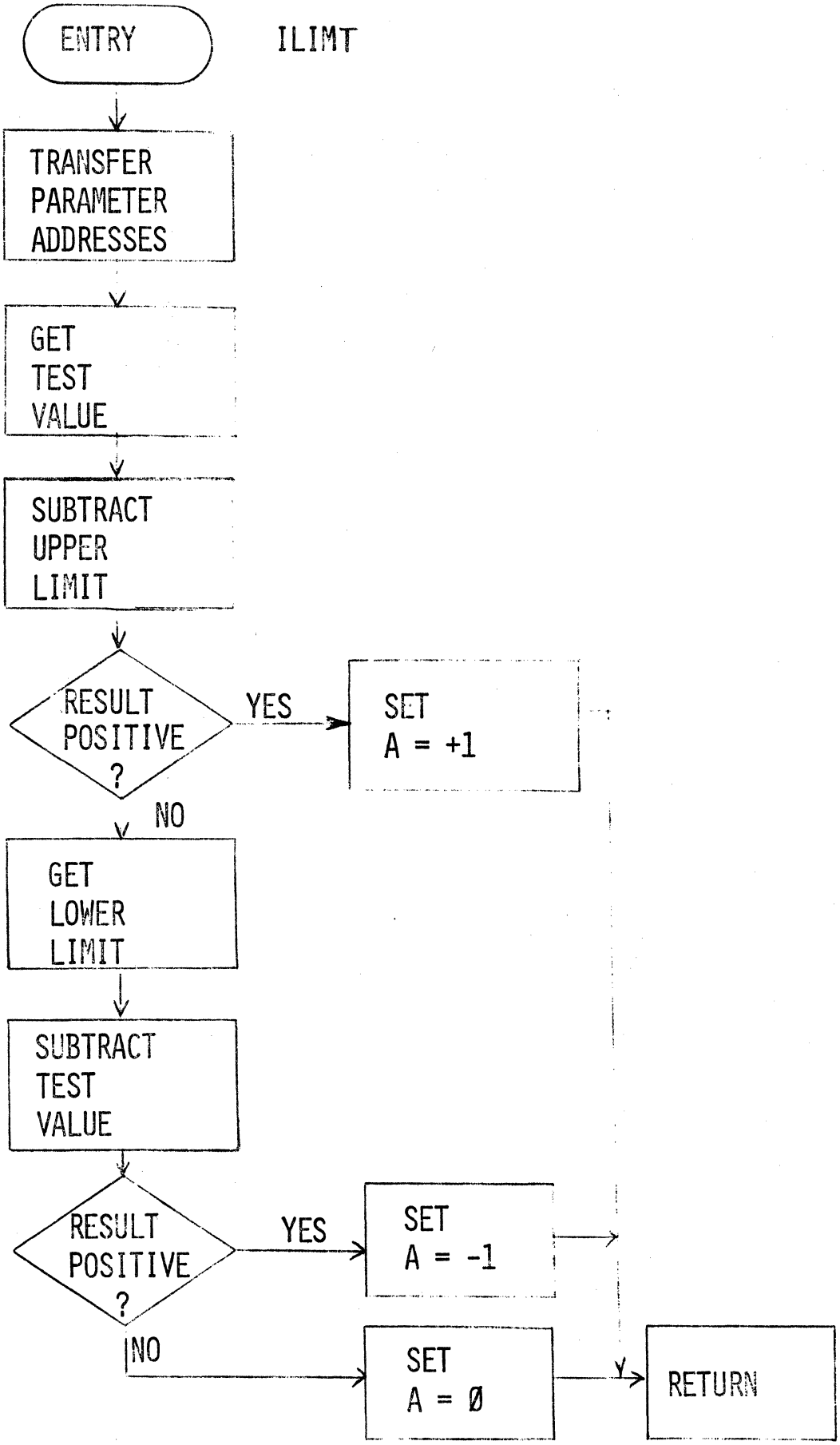
-1 IF THE TEST VALUE IS $\leq$ LOWER LIMITS.

THE ENTRY POINT FOR THE SUBROUTINE MUST BE ILIMT.

THE UPPER LIMIT, LOWER LIMIT, & TEST VALUE ARE FLOATING POINT NUMBERS.

THE FORTRAN MAIN PROGRAM RELOCATABLE TAPE WILL BE SUPPLIED BY THE INSTRUCTOR.

LAB 2.4 FLOWCHART



LAB 2.4 FORTRAN MAIN PROGRAM

```
FTN,B,L
C
C ASSEMBLER LAB 2.4 FTN MAIN
C
      PROGRAM LMTST
5     WRITE (2,100)
100   FORMAT (" ENTER VALUES FOR UPPER, LOWER, AND TEST"/)
      READ (1,*) X,Y,Z
      IF (X-Y) 6,7,7
6     WRITE (2,150)
150   FORMAT (" LIMIT ERROR")
      GO TO 5
7     IF (ILIMT(X,Y,Z)) 10,20,30
10    WRITE (2,200)
200   FORMAT (/" <= LOWER LIMIT"/)
      GO TO 5
20    WRITE (2,300)
300   FORMAT (/" WITHIN LIMITS"/)
      GO TO 5
30    WRITE (2,400)
400   FORMAT (/" => UPPER LIMIT"/)
      GO TO 5
      END
      END$
```

LAB 24 CONSOLE RUN SHEET (1 OF 2)


```
  LMTST

 02000 02250

*LOAD

  L24JH

 02251 02300

*LOAD

*LST
.IOC.   16026
.SQT.   16003
.MEM.   15775
.BUFR   16177
HALT    15770
LMTST   02000
CLRIO   05345
.DIO.   04275
.DTA.   04407
.IOR.   04125
.DST    05234
.DLD    05231
.FSB    04727
ILIMT   02254
.STOP   05310
.ENTR   05163
.BIO.   04354
.IAR.   04231
.IOI.   04155
.RAR.   04205
OLDIO   02507
.FLUN   05325
.PACK   05066
ENDIO   05336
FLOAT   05061
IFIX    05256
.FAD    04724
.DIV    05226
.MPY    05223

 *LINKS
 01714 01777

*RUN
```

LAB 2.4 CONSOLE RUN SHEET (2 OF 2)

```
 ENTER VALUES FOR UPPER, LOWER, AND TEST

10,15,5
 LIMIT ERROR
 ENTER VALUES FOR UPPER, LOWER, AND TEST

15,10,5

 <= LOWER LIMIT

 ENTER VALUES FOR UPPER, LOWER, AND TEST

15,10,10

 <= LOWER LIMIT

 ENTER VALUES FOR UPPER, LOWER, AND TEST

15,10,12.5

 WITHIN LIMITS

 ENTER VALUES FOR UPPER, LOWER, AND TEST

15,10,15

 => UPPER LIMIT

 ENTER VALUES FOR UPPER, LOWER, AND TEST

15,10,20

 => UPPER LIMIT

 ENTER VALUES FOR UPPER, LOWER, AND TEST

-10,-15,-5

 => UPPER LIMIT

 ENTER VALUES FOR UPPER, LOWER, AND TEST
```

LAB 2.1 SOLUTION

```
0001                    ASMB,R,B,L
0002*
0003* ASSEMBLER LAB 2.1 WRITE NAME ON TTY
0004*
0005  00000                     NAM L21JH
0006                            EXT .IOC.
0007  00000 000000  START NOP
0008*
0009* USE .IOC. TO WRITE YOUR NAME ON THE TTY
0010  00001 016001X             JSB .IOC.     CALL .IOC.
0011  00002 020002              OCT 20002     OPERATION = WRITE ON TTY
0012  00003 026001R             JMP *-2       TRY AGAIN IF TTY BUSY
0013  00004 000013R             DEF BUFFR     POINTER TO BUFFER
0014  00005 177764              DEC -12       # OF CHARACTERS
0015*
0016* WRITE STILL IN PROGRESS ?
0017  00006 016001X             JSB .IOC.     CALL .IOC.
0018  00007 040002              OCT 40002     OPERATION = STATUS TTY
0019  00010 002020              SSA           TTY BUSY ?
0020  00011 026006R             JMP *-3         YES, STATUS AGAIN
0021*
0022* HALT
0023  00012 102077              HLT 77B         NO, HALT
0024*
0025  00013 042101  BUFFR ASC 6,DAVE PACKARD
      00014 053105
      00015 020120
      00016 040503
      00017 045501
      00020 051104
0026                            END START
**  NO ERRORS*
```

LAB 2.3 SOLUTION

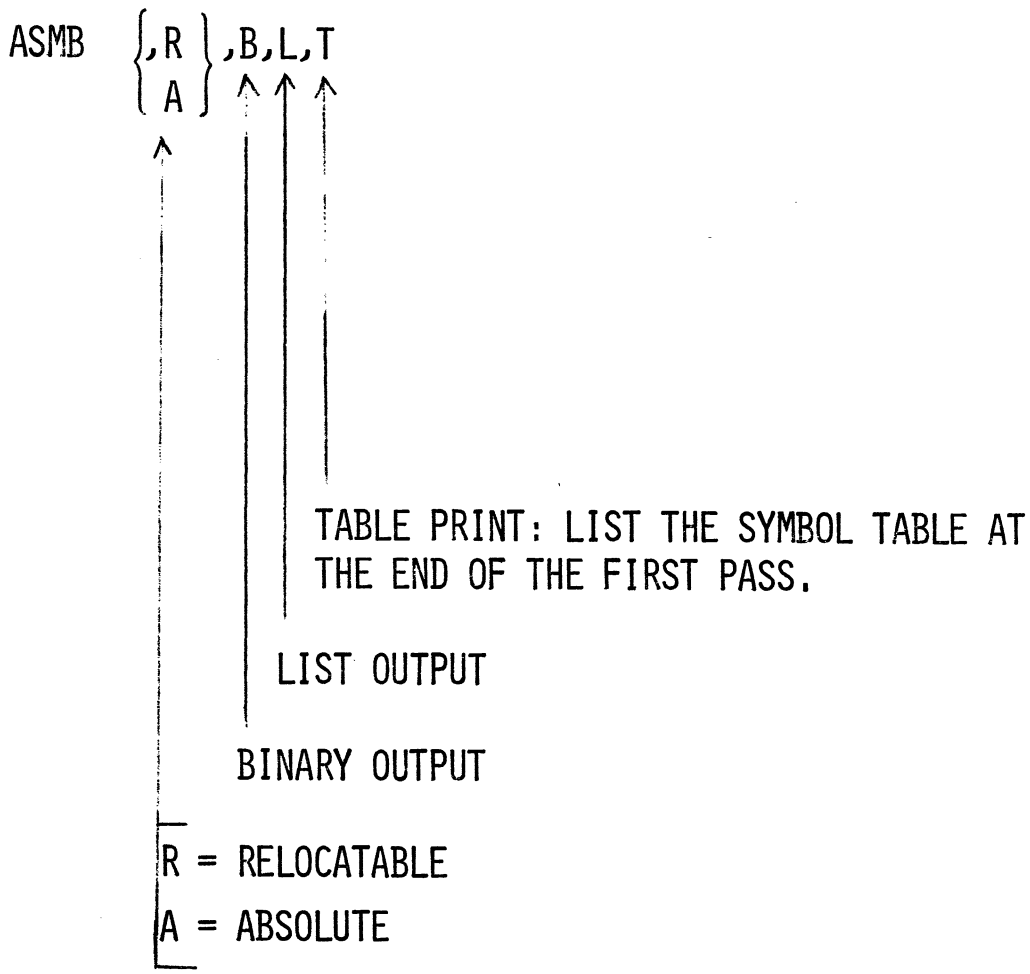
PAGE 0002 #01

```
0001                    ASMB,R,B,L
0002*
0003* ASSEMBLER LAB 2.3 FIND ERRORS ON TAPE
0004*
0005  00000                    NAM L23JH
0006                           EXT .IOC.
0007  00000 000000  START NOP
0008  00001 016001X AGAIN JSB .IOC.     SYSTEM CLEAR
0009  00002 000000         OCT 0
0010*
0011  00003 016001X READ  JSB .IOC.     READ ASCII RECORD
0012  00004 010005         OCT 10005
0013  00005 026003R        JMP READ
0014  00006 000040R        DEF RDBUF
0015  00007 177754         DEC -20
0016*
0017  00010 016001X STAT  JSB .IOC.     STATUS BUSY ? YES, WAIT
0018  00011 040005         OCT 40005
0019  00012 002020         SSA
0020  00013 026010R        JMP STAT
0021*
0022  00014 062047R        LDA C1516     CODE = ** ?
0023  00015 052061R        CPA =A**
0024  00016 016031R        JSB WEMSG       YES, PRINT ERROR MESSAGE
0025*
0026  00017 062040R        LDA RDBUF     CODE = 00 ?
0027  00020 052062R        CPA =A00
0028  00021 026023R        JMP SHTDN       YES, START SHUTDOWN
0029  00022 026003R        JMP READ        NO, READ NEXT RECORD
0030*
0031  00023 016001X SHTDN JSB .IOC.     SYSTEM STATUS BUSY ?
0032  00024 040000         OCT 40000
0033  00025 002020         SSA               NO, HALT
0034  00026 026023R        JMP SHTDN         YES, WAIT
0035*
0036  00027 102077         HLT 77B
0037*
0038  00030 026001R        JMP AGAIN     RESTART
0039*
0040  00031 000000  WEMSG NOP           PRINT ERROR MESSAGE SUBROUTINE
0041  00032 016001X        JSB .IOC.
0042  00033 020002         OCT 20002
0043  00034 026032R        JMP *-2
0044  00035 000052R        DEF MSG
0045  00036 177763         DEC -13
0046  00037 126031R        JMP WEMSG,I
0047*
0048  00040 000000  RDBUF BSS 7
0049  00047 000000  C1516 BSS 3
0050                       SUP
0051  00052 042522  MSG   ASC 7,ERROR MESSAGE
0052                       END START
**  NO ERRORS*
```

LAB 2.4 SOLUTION

```
0001                    ASMB,R,B,L
0002*
0003* ASSEMBLER LAB 2.4 LIMIT CHECK SUBROUTINE
0004*
0005  00000                     NAM L24JH
0006                            ENT ILIMT
0007                            EXT .ENTR
0008*
0009* TRANSFER PARAMETER ADDRESSES
0010  00000 000000  UPPER NOP
0011  00001 000000  LOWER NOP
0012  00002 000000  TEST  NOP
0013  00003 000000  ILIMT NOP
0014  00004 016001X       JSB .ENTR
0015  00005 000000R       DEF UPPER
0016*
0017* GET TEST VALUE, SUBTRACT UPPER LIMIT
0018  00006 104200        DLD TEST,I
      00007 100002R
0019  00010 016002X       FSB UPPER,I
      00011 100000R
0020*
0021* RESULT POSITIVE ? YES, SET A = +1, RETURN
0022  00012 002020        SSA
0023  00013 026016R       JMP LWTST
0024  00014 002404        CLA,INA
0025  00015 026027R       JMP RETRN
0026*
0027* GET LOWER LIMIT, SUBTRACT TEST VALUE
0028  00016 104200  LWTST DLD LOWER,I
      00017 100001R
0029  00020 016002X       FSB TEST,I
      00021 100002R
0030*
0031* RESULT POSITIVE ?
0032*   YES, SET A = -1
0033*   NO, SET A = 0
0034  00022 002020        SSA
0035  00023 026026R       JMP OK
0036  00024 003400        CCA
0037  00025 026027R       JMP RETRN
0038  00026 002400  OK    CLA
0039*
0040  00027 126003R RETRN JMP ILIMT,I
0041                        END
**  NO ERRORS*
```

ASMB OPTIONS



NOTE: IF PASS 1 OF THE ASSEMBLER IS STARTED AT 120B, THE ASMB STATEMENT MAY BE ENTERED THRU THE TTY (BCS ONLY)

ASMB OPTIONS EXAMPLE

```
 PAGE 0001


0001                    ASMB,R,B,L,T
X       R 000004
Y       R 000014
XX      R 000005
START   R 000000
TEST    R 000001
YYYY    R 000010
**  NO ERRORS*
```

```
 PAGE 0002 #01


0001                    ASMB,R,B,L,T
0002  00000                     NAM OPTON
0003  00000 000000  START NOP
0004  00001 002400  TEST  CLA
0005  00002 072014R       STA Y
0006  00003 102022        HLT 22B
0007*
0008  00004 000000  X     BSS 1
0009  00005 000000  XX    BSS 3
0010  00010 000000  YYYY  BSS 4
0011  00014 000000  Y     BSS 1
0012*
0013                      END START
**  NO ERRORS*
```

OBTAINING A LIST

```
                    HP SYMBOLIC EDITOR

EDIT FILE DEVICE?
/T

*
/L
/E

SYMBOLIC FILE SOURCE DEVICE?
/P

0001  ASMB,B,R,L
0002        NAM LIST
0003        ENT START
0004  START NOP
0005  NEXT  LIA 1
0006        CMA,INA
0007        HLT 77B
0008        JMP NEXT
0009        END START


**END-OF-TAPE
*
/E

*END
```

DUPLICATING A SOURCE TAPE

HP SYMBOLIC EDITOR

```
EDIT FILE DEVICE?
/T

*
/E

SYMBOLIC FILE SOURCE DEVICE?
/P

SYMBOLIC FILE DESTINATION DEVICE?
/P


**END-OF-TAPE
*
/E

*END
```

INSTRUCTION CODES IN BINARY

15	14 13 12	11 10 9	8 7 6	5 4 3	2 1 0
D/I	AND 001	0 Z/C	← Memory Address →		
D/I	XOR 010	0 Z/C			
D/I	IOR 011	0 Z/C			
D/I	JSB 001	1 Z/C			
D/I	JMP 010	1 Z/C			
D/I	ISZ 011	1 Z/C			
D/I	AD* 100	A/B Z/C			
D/I	CP* 101	A/B Z/C			
D/I	LD* 110	A/B Z/C			
D/I	ST* 111	A/B Z/C			

15	14 13 12	11 10 9	8 7 6	5 4 3	2 1 0
0	SRG 000	A/B 0 D/E	*LS 000	†CLE D/E ‡SL*	*LS 000
		A/B 0 D/E	*RS 001	D/E	*RS 001
		A/B 0 D/E	R*L 010	D/E	R*L 010
		A/B 0 D/E	R*R 011	D/E	R*R 011
		A/B 0 D/E	*LR 100	D/E	*LR 100
		A/B 0 D/E	ER* 101	D/E	ER* 101
		A/B 0 D/E	EL* 110	D/E	EL* 110
		A/B 0 D/E	*LF 111	D/E	*LF 111
		NOP 000	000	000	000

15	14 13 12	11 10 9	8 7 6	5 4 3	2 1 0
0	ASG 000	A/B 1	CL* 01 CLE 01	SEZ SS* SL*	IN* SZ* RSS
		A/B	CM* 10 CME 10		
		A/B	CC* 11 CCE 11		

15	14 13 12	11 10 9	8 7 6	5 4 3	2 1 0
1	IOG 000	1 H/C	HLT 000	← Select Code →	
		1 0	STF 001		
		1 1	CLF 001		
		1 0	SFC 010		
		1 0	SFS 011		
		A/B 1 H/C	MI* 100		
		A/B 1 H/C	LI* 101		
		A/B 1 H/C	OT* 110		
		0 1 H/C	STC 111		
		1 1 H/C	CLC 111		
		1 0	STO 001	000	001
		1 1	CLO 001	000	001
		1 H/C	SOC 010	000	001
		1 H/C	SOS 011	000	001

15	14 13 12	11 10 9	8 7 6	5 4 3	2 1 0
1	EAG 000	MPY** 000	010	000	000
		DIV** 000	100	000	000
		DLD** 100	010	000	000
		DST** 100	100	000	000
		ASR 001	000	0 1	← number of bits →
		ASL 000	000	0 1	
		LSR 001	000	1 0	
		LSL 000	000	1 0	
		RRR 001	001	0 0	
		RRL 000	001	0 0	

Notes: * = A or B, according to bit 11.

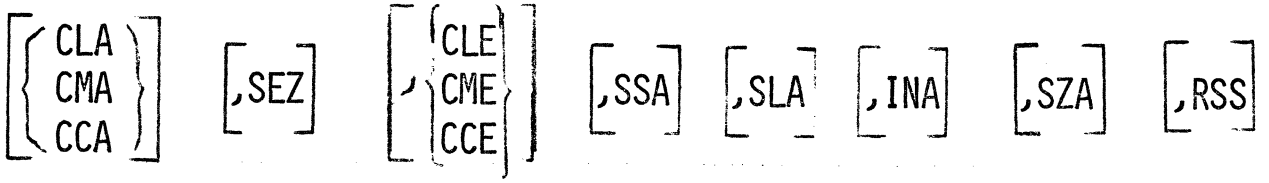
D/I, A/B, Z/C, D/E, H/C coded: 0/1.

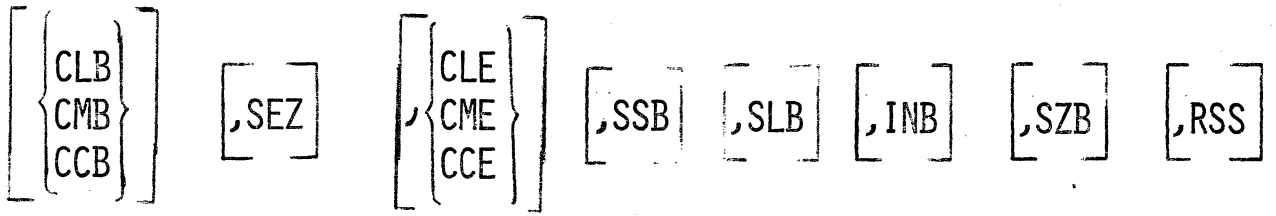
**Second word is Memory Address.

†CLE: Only this bit is required.

‡SL*: Only this bit and bit 11 (A/B as applicable) are required.

OP CODE





1. INSTRUCTIONS ARE COMBINED FROM LEFT TO RIGHT IN THE ORDER SHOWN
2. IF TWO OR MORE SKIP CONDITIONS ARE INCLUDED, A SKIP OCCURS IF EITHER OR BOTH CONDITIONS ARE PRESENT: EXCEPTION, SSA/B, SLA/B, RSS; BOTH CONDITIONS MUST BE MET.

COMBINING GUIDE
ALTER - SKIP INSTRUCTIONS

RSS SKIP EXAMPLE

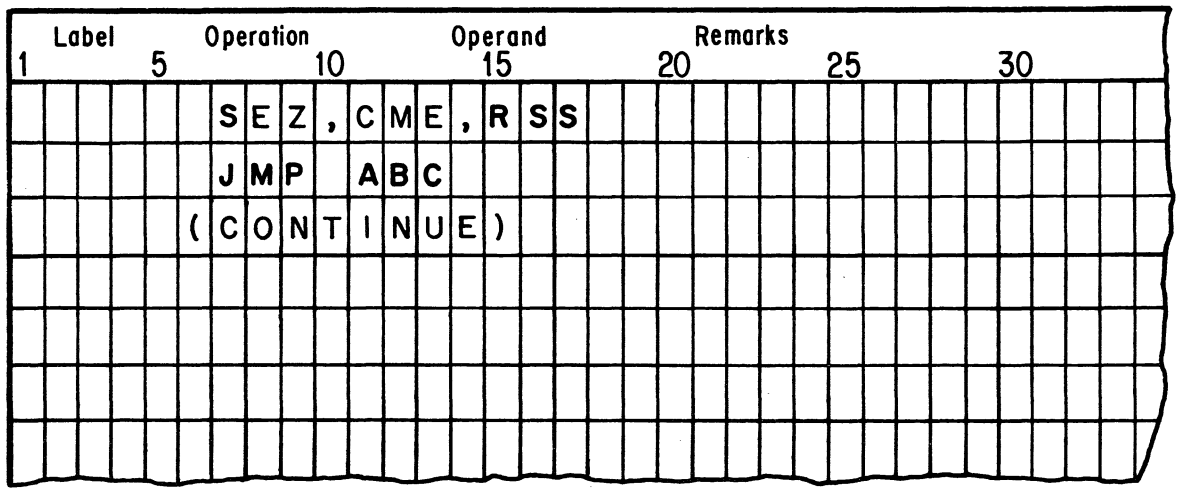
PROBLEM:

TEST THE "E" REGISTER.
IF E = Ø, EXECUTE THE NEXT INSTRUCTION
AND SET E = 1

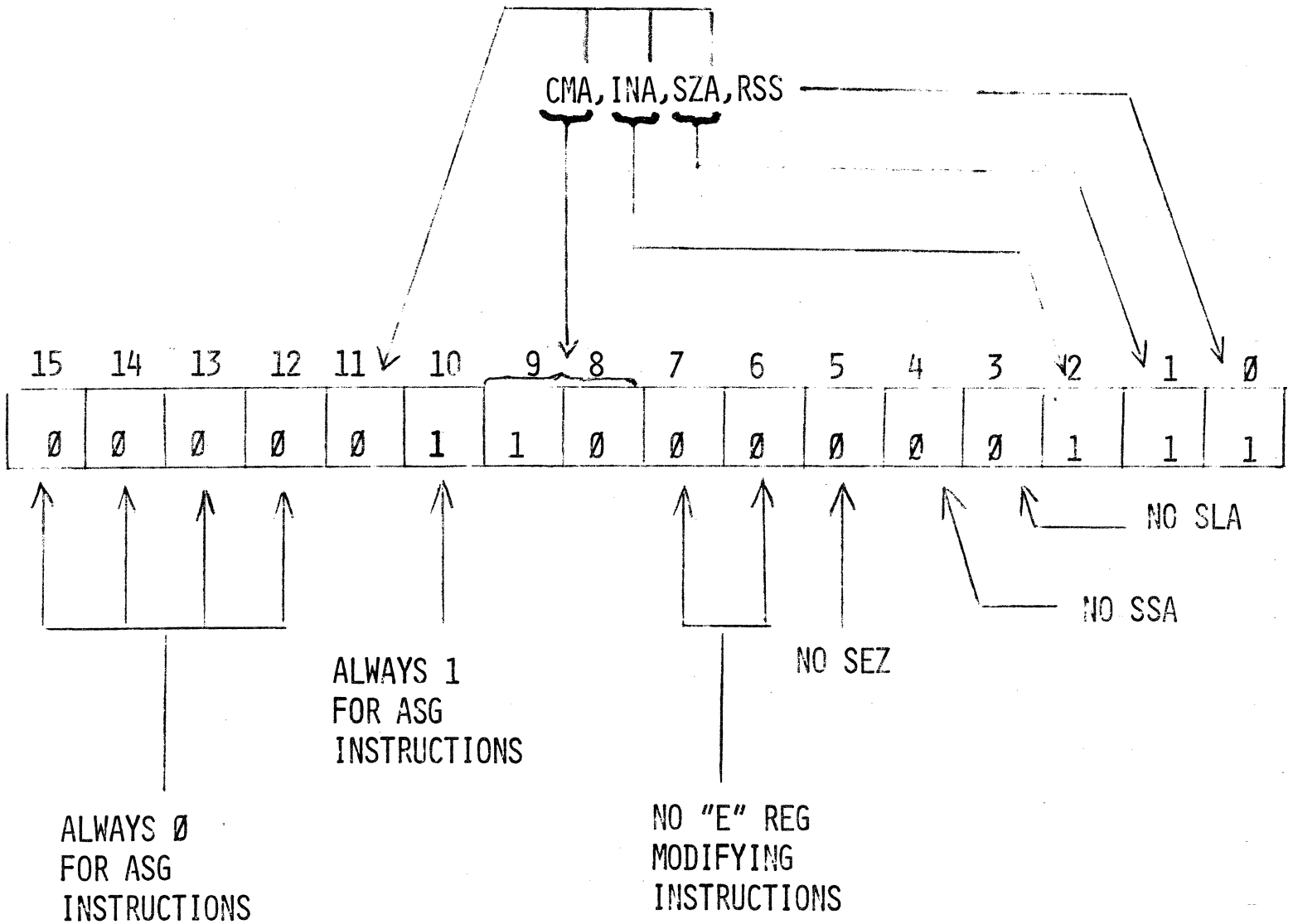
— OR —

IF E = 1, SKIP THE NEXT INSTRUCTION
AND SET E = Ø

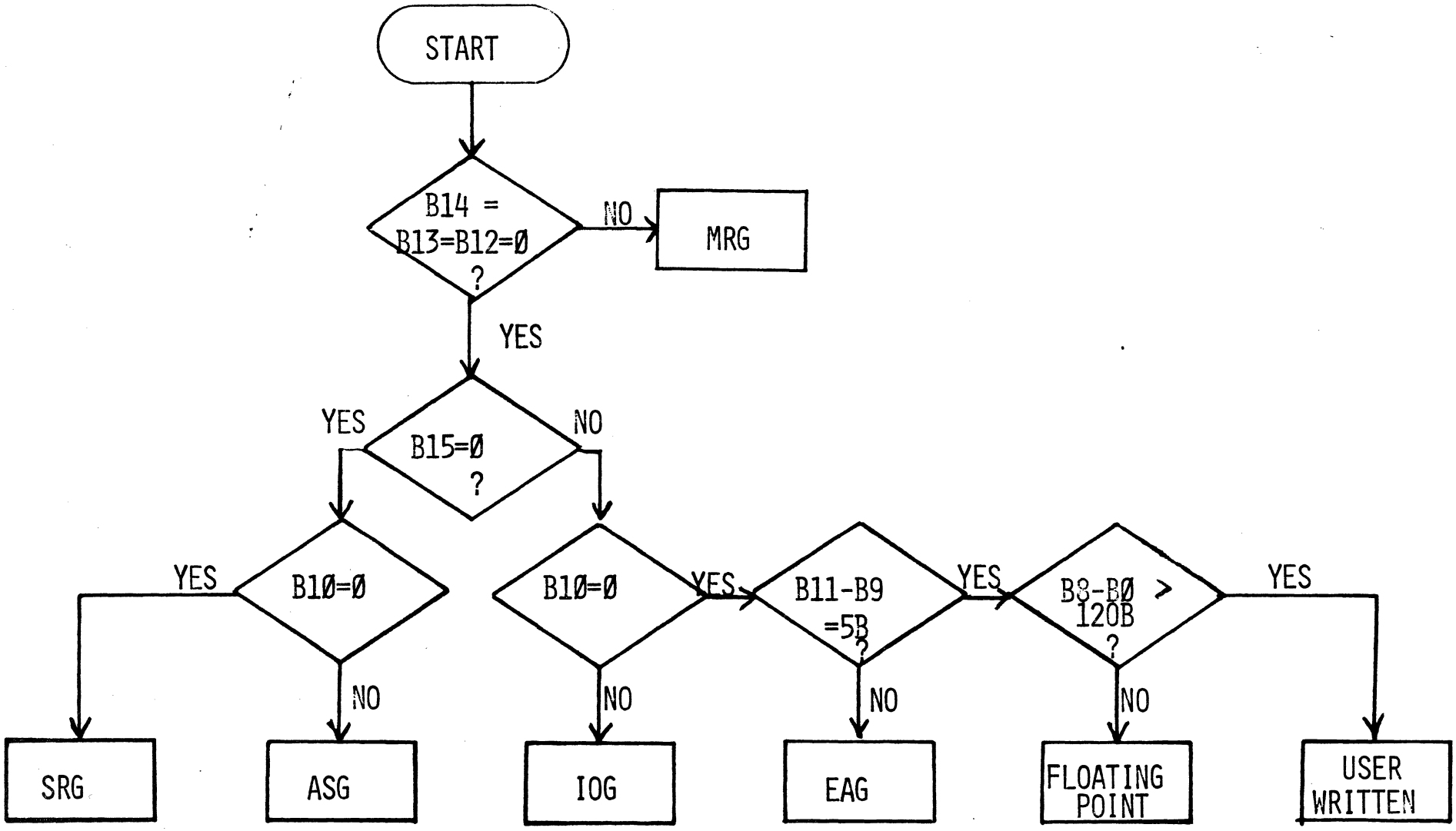
SOLUTION:



MNEMONICS TO MACHINE CODE EXAMPLE

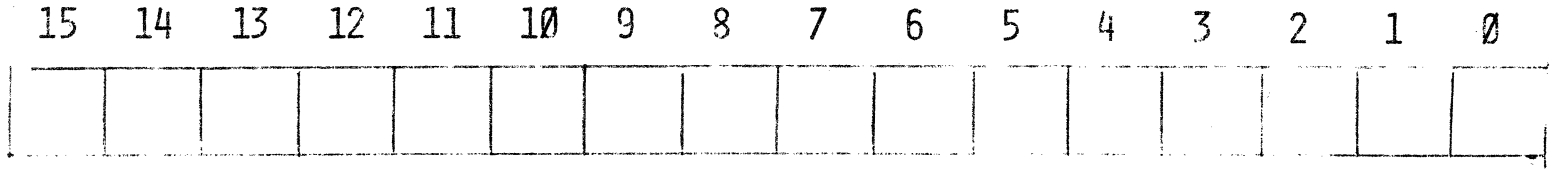


DETERMINING THE INSTRUCTION GROUP

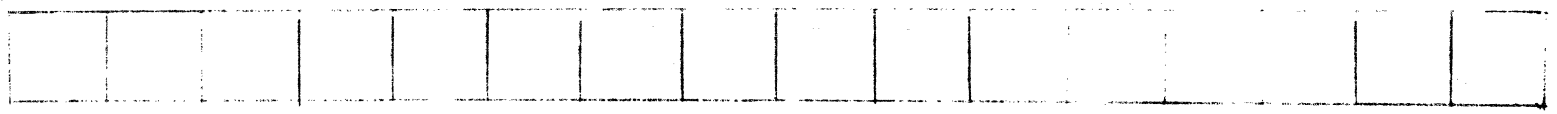


MNEMONICS TO MACHINE CODE CLASSROOM EXERCISE

WHAT MACHINE CODE WILL THE ASSEMBLER GENERATE FOR THE FOLLOWING INSTRUCTIONS?



CMB,SEZ,CCE,SLB



ISZ 50,I

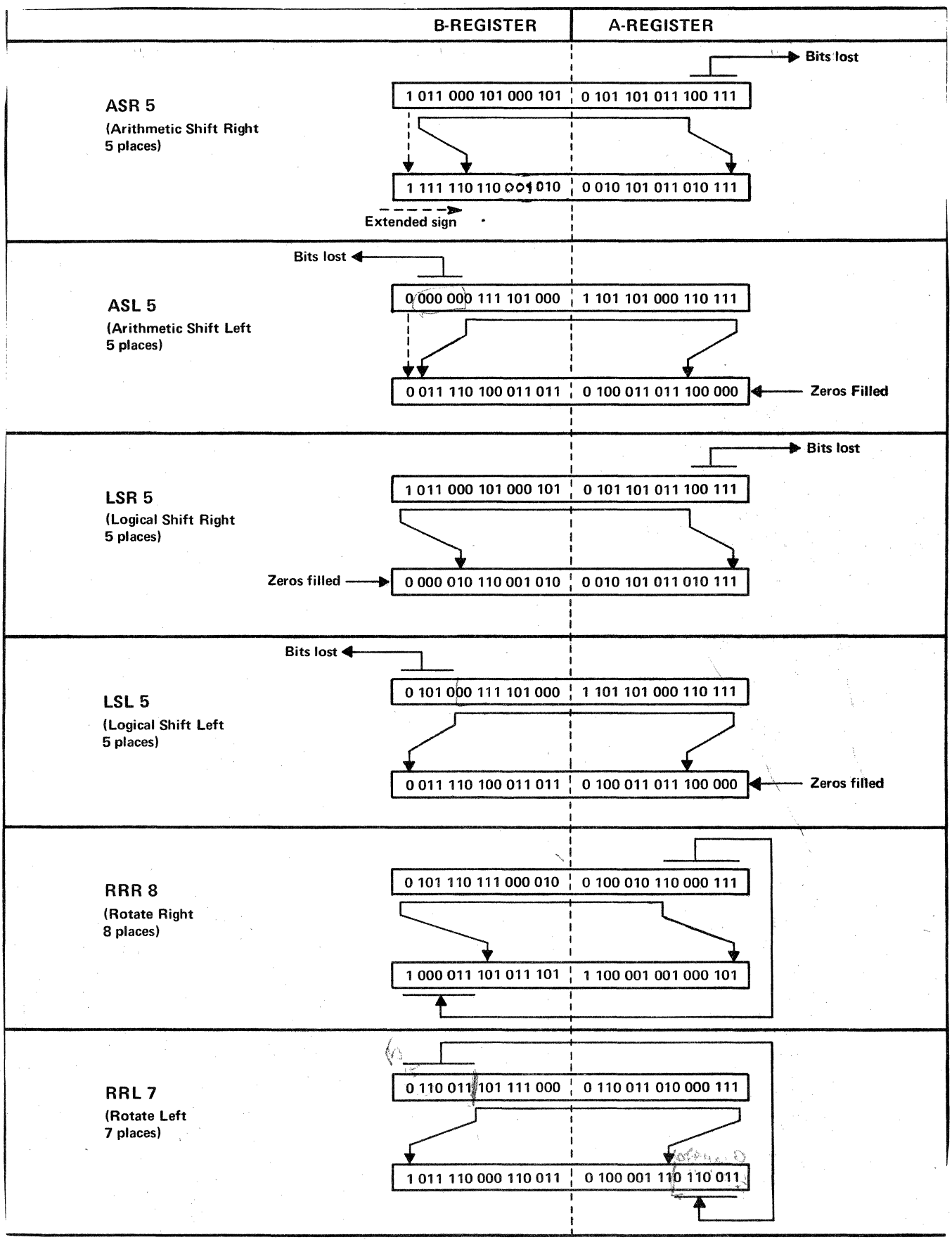
ALF,ALF

MACHINE CODE TO MNEMONICS CLASSROOM EXERCISE

WHAT ARE THE MNEMONICS THAT CORRESPOND TO THE FOLLOWING MACHINE CODES?

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	1	0	0	0	1	0	0	1	1	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	1	0	0	0	0	0	1	0	1	0	0	0
0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0

ADDITIONAL EAG INSTRUCTIONS



USER INSTRUCTIONS

USER UNIQUE INSTRUCTIONS MAY BE DEFINED USING THE 2100 MICROPROGRAMMING FACILITY.

THE OP-CODES USED MUST BE

105XXX,

WHERE XXX RANGES FROM 000 TO 377 OCTAL.

LITERALS

E.G. LDA =D1 PLACES + 1 IN THE A-REGISTER.

VALID LITERAL FORMS:

=D A DECIMAL INTEGER, IN THE RANGE -32768 TO +32767.

=F A FLOATING POINT NUMBER, ANY POSITIVE OR NEGATIVE REAL NUMBER IN THE RANGE 10^{-38} TO 10^{+38}.

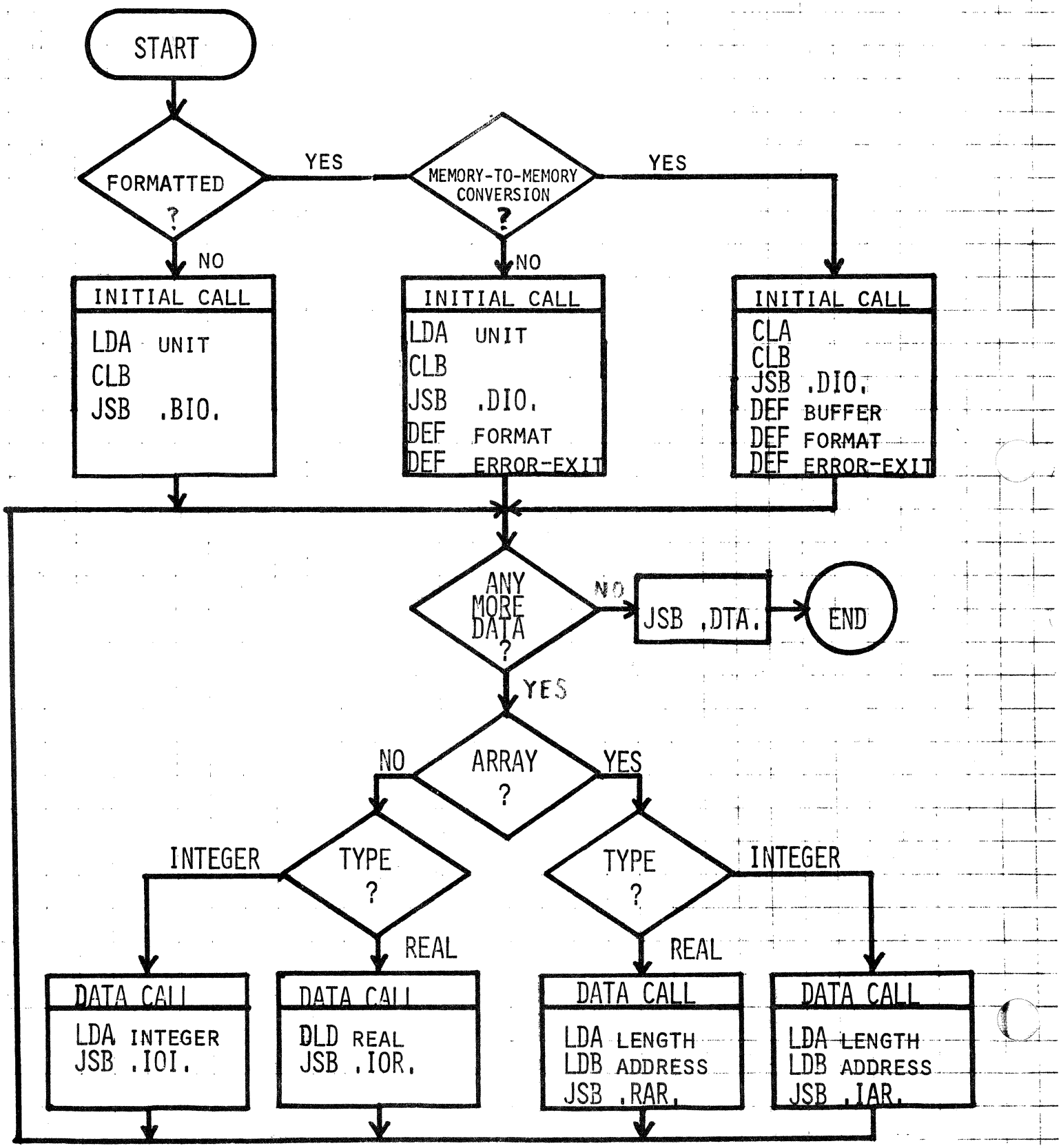
=B AN OCTAL INTEGER, 1 TO 6 DIGITS YXXXXX WHERE Y MAY BE Ø OR 1, AND X MAY BE Ø TO 7.

=A TWO ASCII CHARACTERS

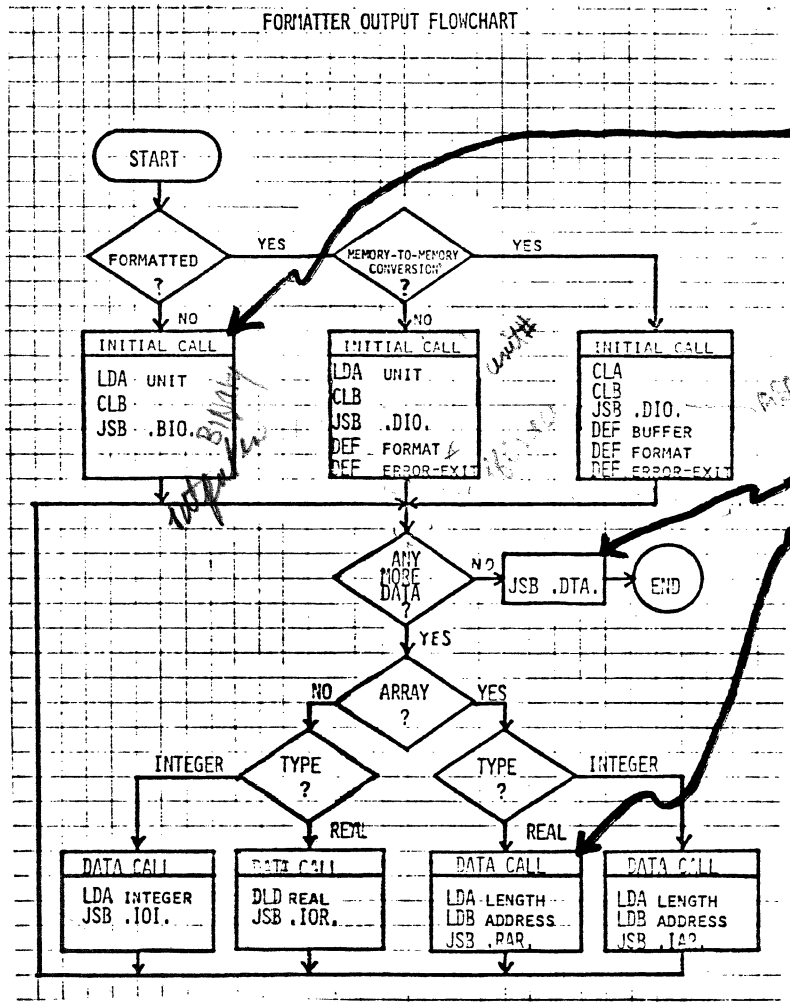
USER INSTRUCTION & LITERAL EXAMPLE

```
0001                    ASMB,R,B,L
0002   00000                    NAM UL        USER INSTRUCTION & LITERALS
0003   00000 000000  BEGIN NOP
0004*
0005   00001 062021R            LDA =B10005   OCTAL LITERAL
0006   00002 072006R            STA CTLWD
0007*
0008   00003 062022R            LDA =D-10     DECIMAL LITERAL
0009   00004 072011R            STA LNGTH
0010*
0011                            EXT .IOC.
0012   00005 016001X            JSB .IOC.
0013   00006 000000  CTLWD BSS 1
0014   00007 026005R            JMP *-2
0015   00010 000014R            DEF BFR
0016   00011 000000  LNGTH BSS 1
0017*
0018   00012 105200             OCT 105200    USER DEFINED INSTRUCTION
0019*
0020   00013 102000             HLT
0021   00014 000000  BFR   BSS 5
       00021 010005
       00022 177766
0022                            END BEGIN
**  NO ERRORS*
```

FORMATTER OUTPUT FLOWCHART

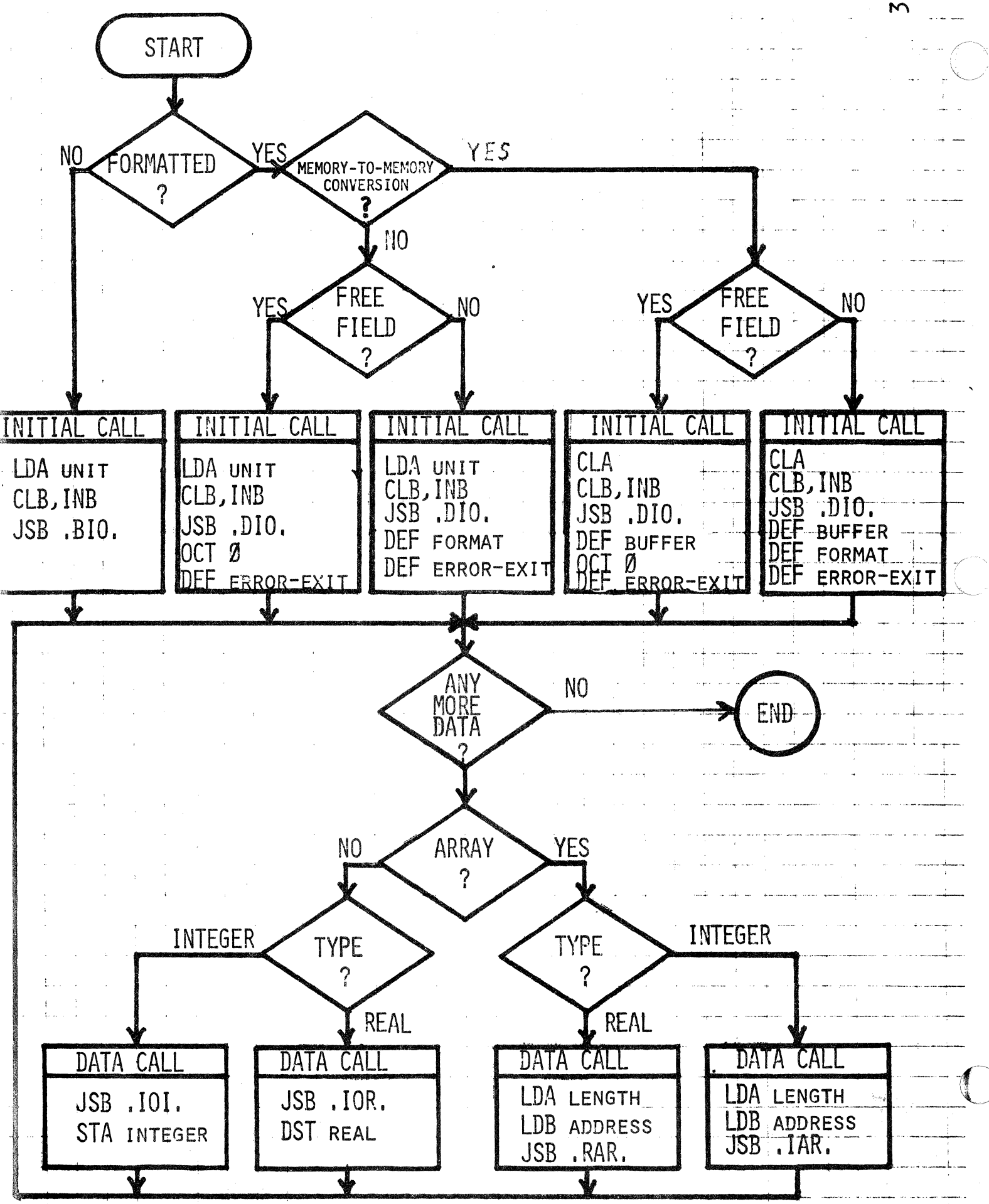


FORMATTER OUTPUT EXAMPLE 1.

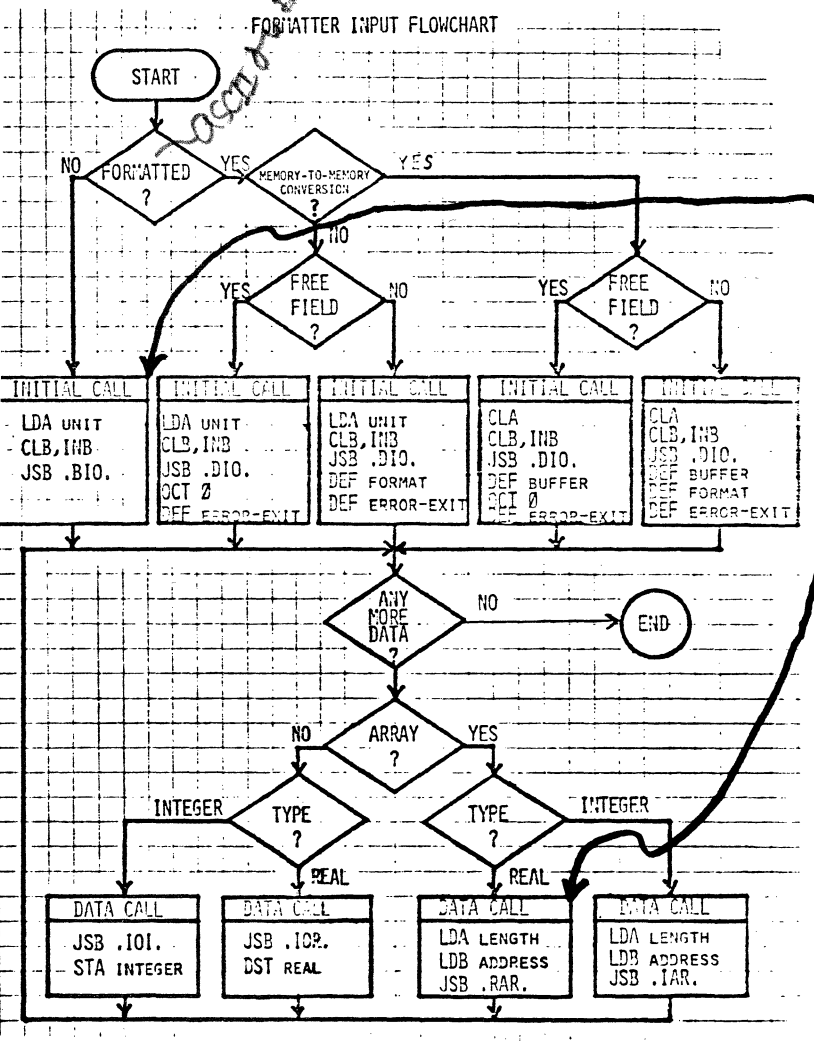



```
0001                    ASMB,B,L,R
0002*
0003* AN ENTIRE ARRAY OF "REAL NUMBER" DATA IS TO BE OUTPUT IN BINARY
0004* FORM USING UNIT #4. THE DATA BLOCK HAS 100 ELEMENTS AND MAY BE
0005* REFERENCED BY THE LABEL "BUFFR".
0006*
0007  00000                     NAM FMT01    FORMATTER OUTPUT EXAMPLE 1.
0008                            EXT .BIO.,.RAR.,.DTA.
0009                            EXT .IOC.
0010  00000 000000  FMT01 NOP
0011*
0012*
0013  00001 062016R             LDA UNIT
0014  00002 006400              CLB            INDICATES OUTPUT
0015  00003 016001X             JSB .BIO.      BINARY I-O OPERATION
0016*
0017  00004 062017R             LDA NMBR       NUMBER OF ELEMENTS IN ARRAY
0018  00005 066020R             LDB BUFAD      ADDRESS OF BUFFER ADDRESS
0019  00006 016002X             JSB .RAR.      BEGIN REAL ARRAY OUTPUT
0020*
0021  00007 016003X             JSB .DTA.      OUTPUT LAST RECORD
0022*
0023*
0024  00010 016004X             JSB .IOC.     .CHECK STATUS OF PUNCH
0025  00011 040004              OCT 40004
0026  00012 002020              SSA            PUNCH BUSY?
0027  00013 026010R             JMP *-3         YES, CHECK AGAIN
0028*
0029  00014 102077              HLT 77B         NO, HALT
0030  00015 026001R             JMP FMT01+1
0031  00016 000004  UNIT  OCT 4               PUNCH LOGICAL UNIT
0032  00017 000144  NMBR  DEC 100
0033  00020 000021R BUFAD DEF BUFFR          BUFAD CONTAINS ADDRESS OF BUFFR
0034                      SUP
0035                BUFFR REP 10
0036  00021 040000              DEC 1.,2.,3.,4.,5.,6.,7.,8.,9.,10.
0036  00045 040000              DEC 1.,2.,3.,4.,5.,6.,7.,8.,9.,10.
0036  00071 040000              DEC 1.,2.,3.,4.,5.,6.,7.,8.,9.,10.
0036  00115 040000              DEC 1.,2.,3.,4.,5.,6.,7.,8.,9.,10.
0036  00141 040000              DEC 1.,2.,3.,4.,5.,6.,7.,8.,9.,10.
0036  00165 040000              DEC 1.,2.,3.,4.,5.,6.,7.,8.,9.,10.
0036  00211 040000              DEC 1.,2.,3.,4.,5.,6.,7.,8.,9.,10.
0036  00235 040000              DEC 1.,2.,3.,4.,5.,6.,7.,8.,9.,10.
0036  00261 040000              DEC 1.,2.,3.,4.,5.,6.,7.,8.,9.,10.
0036  00305 040000              DEC 1.,2.,3.,4.,5.,6.,7.,8.,9.,10.
0037                            UNS
0038*
0039* THE OPERAND IN THE END STATEMENT (I.E. FMT01) TELLS THE
0040* RELOCATING LOADER WHERE TO ENTER THE PROGRAM
0041                            END FMT01
**  NO ERRORS*
```

FORMATTER INPUT FLOWCHART



FORMATTER INPUT EXAMPLE 1.




```
0001                    ASMB,B,L,R
0002*
0003* READ THE TAPE PUNCHED IN OUTPUT EXAMPLE 1.
0004* USE UNIT #5 FOR INPUT.
0005*
0006  00000                     NAM FMTI1    FORMATTER INPUT EXAMPLE 1.
0007                            EXT .BIO.,.RAR.
0008  00000 000000  FMTI1 NOP
0009*
0010*
0011  00001 062011R AGAIN LDA UNIT
0012  00002 006404         CLB,INB         INDICATES INPUT
0013  00003 016001X         JSB .BIO.      BINARY I-O OPERATION
0014*
0015  00004 062012R         LDA NMBR       NUMBER OF ELEMENTS IN ARRAY
0016  00005 066013R         LDB BUFAD      ADDRESS OF BUFFER ADDRESS
0017  00006 016002X         JSB .RAR.      BEGIN REAL ARRAY INPUT
0018*                                        .
0019*
0020  00007 102077          HLT 77B
0021  00010 026001R         JMP AGAIN      RESTART
0022*
0023  00011 000005  UNIT    OCT 5          PHOTOREADER LOGICAL UNIT
0024  00012 000144  NMBR    DEC 100
0025  00013 000014R BUFAD   DEF BUFFR      BUFAD CONTAINS ADDRESS OF BUFFR
0026                        SUP
0027                  BUFFR REP 10
0028  00014 000000          DEC 0.,0.,0.,0.,0.,0.,0.,0.,0.,0.
0028  00040 000000          DEC 0.,0.,0.,0.,0.,0.,0.,0.,0.,0.
0028  00064 000000          DEC 0.,0.,0.,0.,0.,0.,0.,0.,0.,0.
0028  00110 000000          DEC 0.,0.,0.,0.,0.,0.,0.,0.,0.,0.
0028  00134 000000          DEC 0.,0.,0.,0.,0.,0.,0.,0.,0.,0.
0028  00160 000000          DEC 0.,0.,0.,0.,0.,0.,0.,0.,0.,0.
0028  00204 000000          DEC 0.,0.,0.,0.,0.,0.,0.,0.,0.,0.
0028  00230 000000          DEC 0.,0.,0.,0.,0.,0.,0.,0.,0.,0.
0028  00254 000000          DEC 0.,0.,0.,0.,0.,0.,0.,0.,0.,0.
0028  00300 000000          DEC 0.,0.,0.,0.,0.,0.,0.,0.,0.,0.
0029                        UNS
0030                        END FMTI1
**  NO ERRORS*
```

FORMAT SPECIFICATIONS

CONVERSION

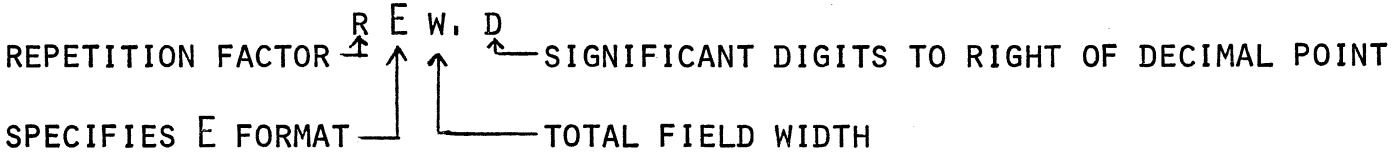
rEw.D	REAL NUMBER WITH EXPONENT
rFw.D	REAL NUMBER WITHOUT EXPONENT
rIw	INTEGER, DECIMAL
r@w	INTEGER, OCTAL
rAw	ASCII CHARACTER

EDITING

rX	SPACES
nH CHARACTER-STRING	
r "CHARACTER-STRING"	
r/	BEGIN NEW RECORD

E SPECIFICATION

DEFINES A FIELD FOR A REAL NUMBER WITH EXPONENT



NOTE: PROPER OUTPUT WILL ALWAYS RESULT IF W≥D + 6.

OUTPUT EXAMPLES

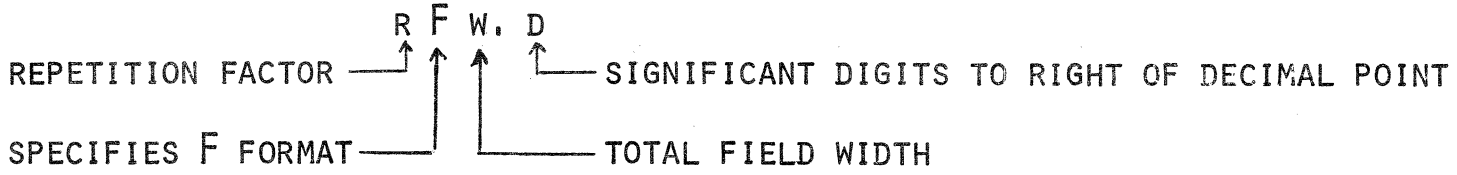
INTERNAL VALUE	FORMAT	EXTERNAL RESULT	REMARKS
+10.4365	E10.3	^^.104E+02	
-34.56	E12.4	^^-.3456E+02	
-34.56	E12.3	^^^-.346E+02	
-64.4365	E 6.1	$$E+02	W<D + 6

INPUT EXAMPLES

EXTERNAL VALUE	FORMAT	INTERNAL RESULT	REMARKS
+1.2345E2	E9.3	123.45	DECIMAL POINT OVERRIDES FORMAT
1234	E4.2	12.34	FORMAT INSERTS DECIMAL POINT

F SPECIFICATION

DEFINES A FIELD FOR A REAL NUMBER WITHOUT AN EXPONENT



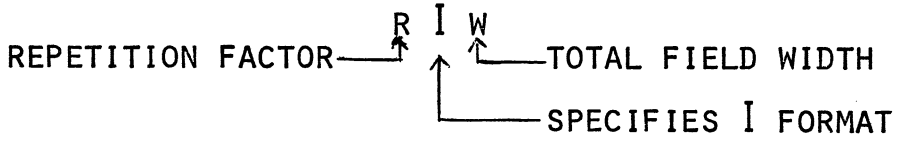
OUTPUT EXAMPLES

INTERNAL VALUE	FORMAT	EXTERNAL VALUE
123.456	F8.3	^123.456
123.456	F8.1	^^^123.5
123.456	F3.2	123
123.456	F2.1	$$

INPUT (IDENTICAL TO E SPECIFICATION)

I SPECIFICATION

DEFINES A FIELD FOR DECIMAL INTEGERS



OUTPUT EXAMPLES

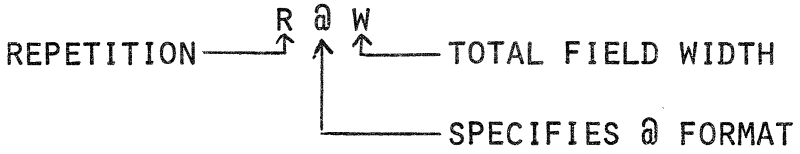
INTERNAL VALUE	FORMAT	EXTERNAL VALUE
175	I4	^175
1.75	I4	^^^2

INPUT EXAMPLES

EXTERNAL VALUE	FORMAT	INTERNAL VALUE
^^132	I5	132

@ SPECIFICATION

DEFINES A FIELD FOR OCTAL INTEGERS



OUTPUT EXAMPLES

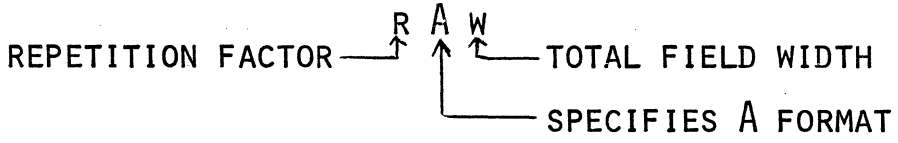
INTERNAL VALUE	FORMAT	EXTERNAL VALUE
132_{10}	@ 5	^^204_8
$32,767_{10}$	@ 6	^77777_8

INPUT EXAMPLES

EXTERNAL VALUE	FORMAT	INTERNAL VALUE
^377_8	@ 4	255_{10}
123456_8	2 @ 3	123_8 AND 456_8

A SPECIFICATION

DEFINES A FIELD FOR ASCII CHARACTERS



OUTPUT EXAMPLES

INTERNAL VALUE	FORMAT	EXTERNAL VALUE
AB	A2	AB
AB	A4	^^AB
AB	A1	B

INPUT EXAMPLES

EXTERNAL VALUE	FORMAT	INTERNAL VALUE
AB	A2	AB
ABC	A3	BC
ABC	A1	ØA

↑ BINARY ZERO, NOT ASCII

X SPECIFICATION

DEFINES SPACES (OUTPUT) & CHARACTER SKIPS (INPUT)

REPETITION FACTOR —→ R X ←— SPECIFIES X FORMAT

OUTPUT EXAMPLE

DATA VALUES	FORMAT	OUTPUT FIELD
32,1.32	I3,4X,F4.2	^32^^^^1.32

INPUT EXAMPLE

INPUT FIELD	FORMAT	INTERNAL VALUES
WEIGHT^^10	8X,I2	10

" " & H SPECIFICATIONS

PROVIDE FOR TRANSFER, WITHOUT CONVERSION, OF ASCII CHARACTERS

OUTPUT EXAMPLE

INTERNAL VALUE IN FORMAT STATEMENT	EXTERNAL VALUE
"ABCDE" OR 5HABCDE	ABCDE

INPUT EXAMPLE

EXTERNAL VALUE	FORMAT
H∧INPUT∧ALLOWS∧VARIABLE∧HEADERS	31H∧∧∧... ∧

INTERNAL RESULT

31HH∧INPUT∧ALLOWS∧VARIABLE∧HEADERS

/ SPECIFICATION

TERMINATES THE CURRENT RECORD

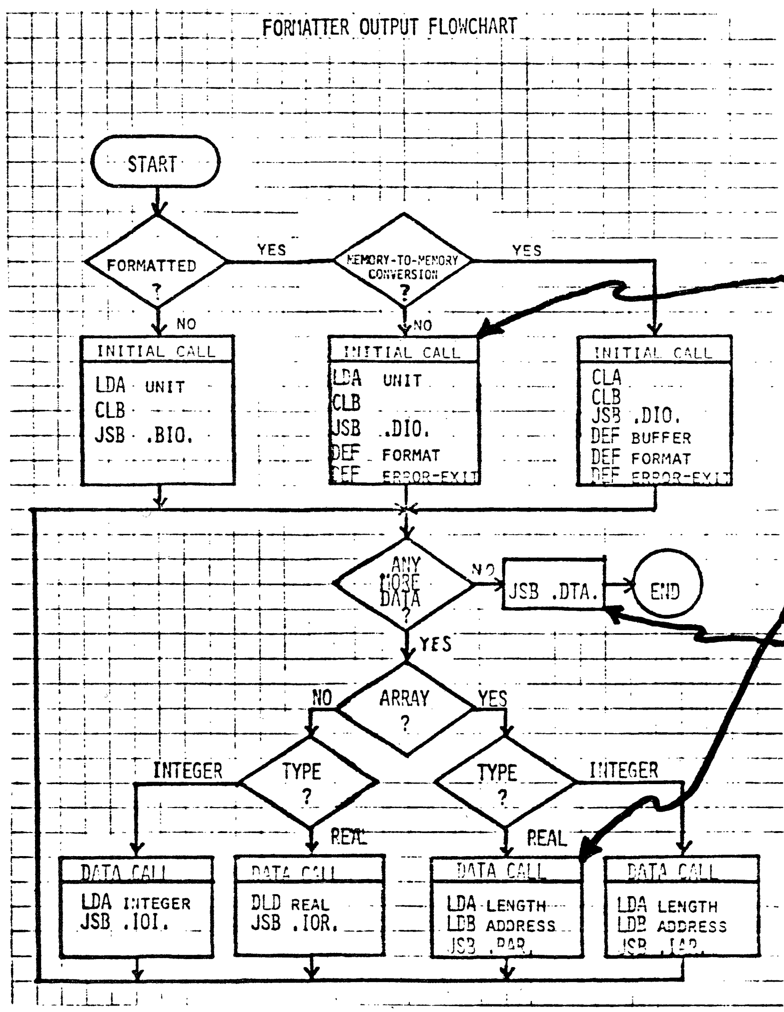
OUTPUT EXAMPLE

FORMAT	EXTERNAL RESULT
"/TERMINATES" / "RECORDS"	/ TERMINATES RECORDS

INPUT EXAMPLE

EXTERNAL DATA VALUES	FORMAT	INTERNAL RESULT
123456 (CR) (LF) 123456 (CR) (LF)	I3,/,I4	123 AND 1234

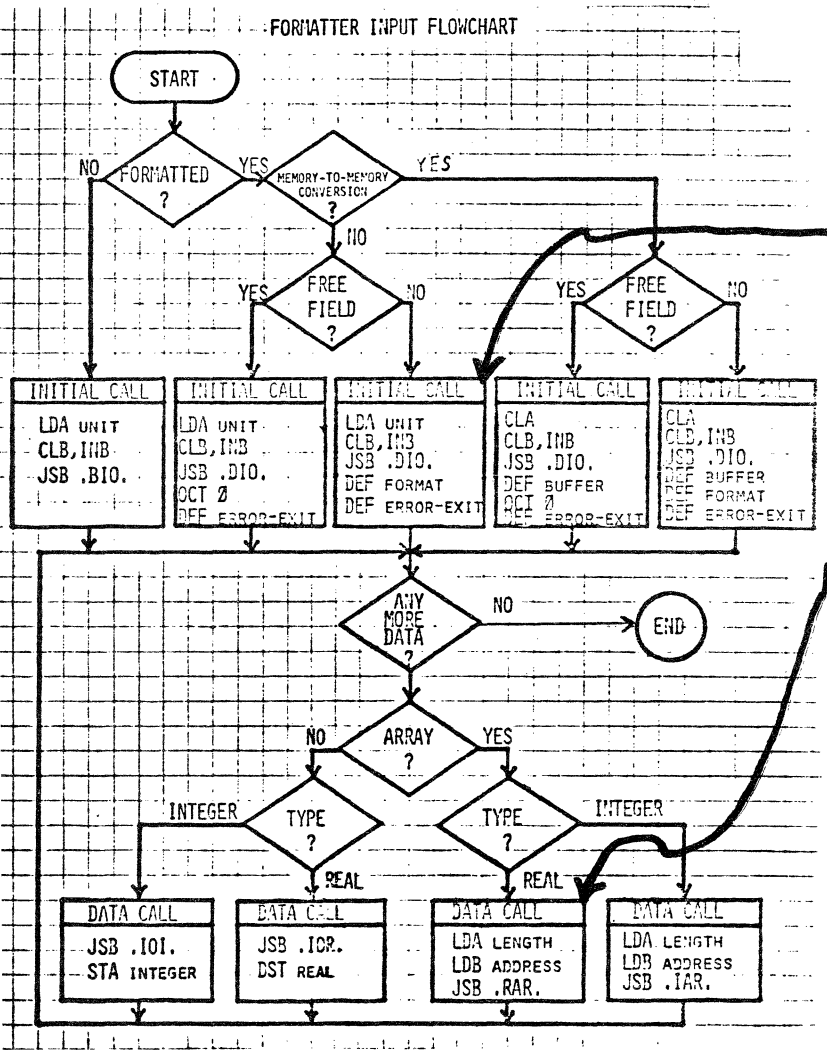
FORMATTER OUTPUT EXAMPLE 2.




```
                           ASMB,B,L,R
*
* AN ENTIRE ARRAY OF REAL NUMBERS IS TO BE OUTPUT ON UNIT #4 AS
* "FORMATTED" DATA. THE ARRAY NAME IS SAM AND IT HAS 10 ELEMENTS.
* THE FORMAT TO BE USED SHOULD PROVIDE THE FOLLOWING OUTPUT FORM.
*              SXXX.XX
*
 00000                     NAM FMT02    FORMATTER OUTPUT EXAMPLE 2.
                           EXT .DIO.,.RAR.,.DTA.
                           EXT ENDIO
 00000 000000       FMT02  NOP
*
*
 00001 062016R             LDA UNIT
 00002 006400              CLB          INDICATES OUTPUT
 00003 016001X             JSB .DIO.    ASCII I-O OPERATION
 00004 000017R             DEF FMT      ADDRESS OF FORMAT SPECIFICATION
 00005 000012R             DEF EXIT     ADDRESS OF ERROR EXIT
*
 00006 062022R             LDA NMBR     NUMBER OF ELEMENTS IN ARRAY
 00007 066023R             LDB BUFAD    ADDRESS OF BUFFER ADDRESS
 00010 016002X             JSB .RAR.    BEGIN REAL ARRAY OUTPUT
*
 00011 016003X             JSB .DTA.    OUTPUT LAST RECORD
*
*
 00012 016004X EXIT        JSB ENDIO    WAIT FOR SYSTEM I-O TO END
 00013 000014R             DEF *+1      RETURN ADDRESS
*
 00014 102077              HLT 77B
 00015 026001R             JMP FMT02+1
*
 00016 000004       UNIT   OCT 4        PUNCH LOGICAL UNIT
 00017 024106       FMT    ASC 3,(F7.2) FORMAT SPECIFICATION
 00020 033456
 00021 031051
 00022 000012       NMBR   DEC 10
 00023 000024R      BUFAD  DEF SAM      BUFAD CONTAINS ADDRESS OF BUFFER
                           SUP
 00024 040000       SAM    DEC 1.,2.,3.,4.,5.,6.,7.,8.,9.,10.
                           UNS
                           END FMT02
** NO ERRORS*
```

IF Halt is here it

FORMATTER INPUT EXAMPLE 2.



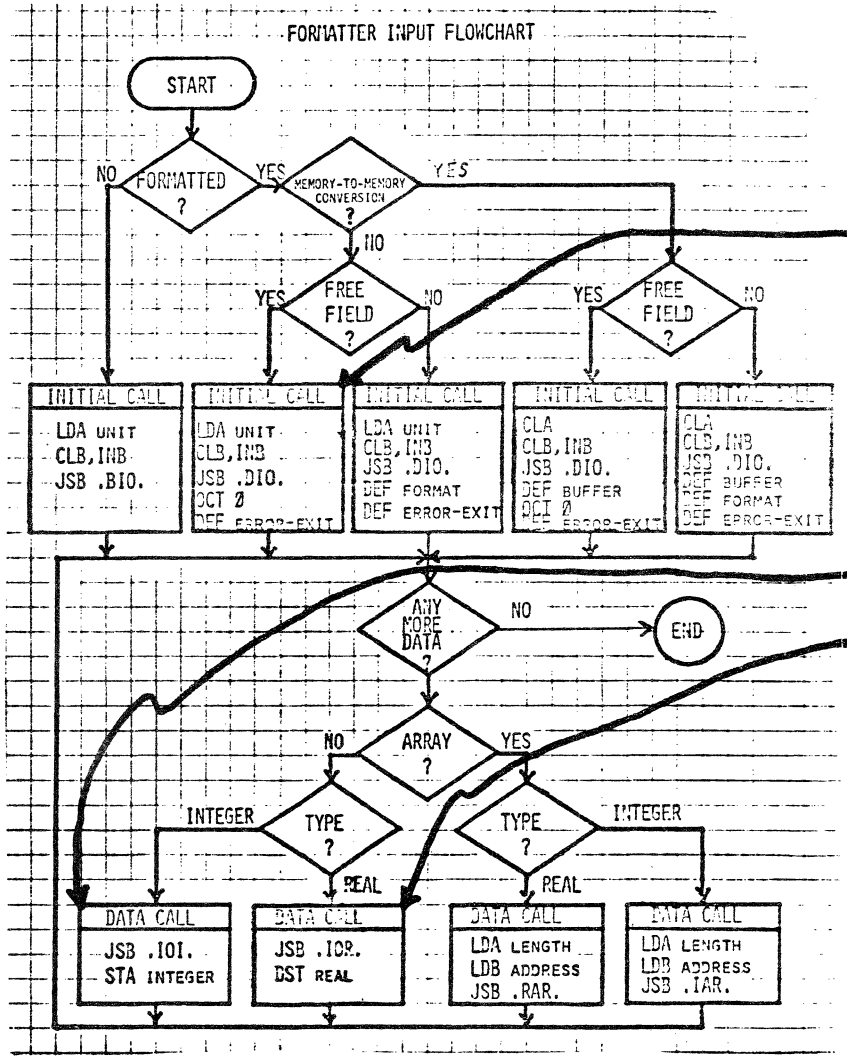

```
0001                    ASMB,B,L,R
0002*
0003* READ THE TAPE PUNCHED IN OUTPUT EXAMPLE 2.
0004* USE UNIT #5 FOR INPUT.
0005*
0006  00000                     NAM FMTI2    FORMATTER INPUT EXAMPLE 2.
0007                            EXT .DIO.,.RAR.
0008  00000 000000  FMTI2 NOP
0009*
0010*
0011  00001 062013R             LDA UNIT
0012  00002 006404              CLB,INB      INDICATES INPUT
0013  00003 016001X             JSB .DIO.    ASCII I-O OPERATION
0014  00004 000014R             DEF FMT      ADDRESS OF FORMAT SPECIFICATION
0015  00005 000011R             DEF EXIT     ADDRESS OF ERROR EXIT
0016*
0017  00006 062017R             LDA NMBR     NUMBER OF ELEMENTS IN ARRAY
0018  00007 066020R             LDB BUFAD    ADDRESS OF BUFFER ADDRESS
0019  00010 016002X             JSB .RAR.    BEGIN REAL ARRAY INPUT
0020*
0021*
0022  00011 102077  EXIT  HLT 77B
0023  00012 026001R       JMP FMTI2+1
0024*
0025  00013 000005  UNIT  OCT 5          PHOTOREADER LOGICAL UNIT
0026  00014 024106  FMT   ASC 3,(F7.2) FORMAT SPECIFICATION
      00015 033456
      00016 031051
0027  00017 000012  NMBR  DEC 10
0028  00020 000021R BUFAD DEF SAM        BUFAD CONTAINS ADDRESS OF BUFFER
0029  00021 000000  SAM   DEC 0.,0.,0.,0.,0.,0.,0.,0.,0.,0.
      00022 000000
      00023 000000
      00024 000000
      00025 000000
      00026 000000
      00027 000000
      00030 000000
      00031 000000
      00032 000000
      00033 000000
      00034 000000
      00035 000000
      00036 000000
      00037 000000
      00040 000000
      00041 000000
      00042 000000
      00043 000000
      00044 000000
0030                            END FMTI2
** NO ERRORS*
```

FREE-FIELD INPUT

WHEN FREE FIELD INPUT IS USED, A FORMAT IS NOT NECESSARY. SPECIAL SYMBOLS WITHIN THE DATA DIRECT THE CONVERSION PROCESS.

SPACE OR ,	DATA ITEM DELIMITERS
/	RECORD TERMINATOR
+ -	SIGN OF ITEM
. E + -	FLOATING POINT NUMBER
@	OCTAL INTEGER
"..."	COMMENTS

FREE-FIELD INPUT EXAMPLE




```
0001                        ASMB,R,B,L
0002*
0003* FORMATTER FREE-FIELD FACILITY
0004*
0005  00000                 NAM FRFLD    FREE-FIELD FACILITY
0006                        EXT .DIO.,.IOI.,.IOR.
0007  00000 000000  FRFLD NOP
0008*
0009*
0010  00001 062022R NEXT  LDA UNIT
0011  00002 006404        CLB,INB      INDICATES INPUT
0012  00003 016001X        JSB .DIO.    INITIALIZES I/O OPERATION
0013  00004 000000        OCT 0        INDICATES FREE-FIELD
0014  00005 000020R        DEF EXIT     ADDRESS OF ERROR EXIT
0015*
0016  00006 016002X        JSB .IOI.    INPUT DECIMAL INTEGER
0017  00007 072023R        STA INTGR
0018*
0019  00010 016002X        JSB .IOI.    INPUT OCTAL INTEGER WITH B
0020  00011 072024R        STA OCTAL
0021*
0022  00012 016003X        JSB .IOR.    INPUT FLTNG PNT #, NO EXP.
0023  00013 104400        DST FLPTF
      00014 000025R
0024*
0025  00015 016003X        JSB .IOR.    INPUT FLTNG PNT #, WITH EXP.
0026  00016 104400        DST FLPTE
      00017 000027R
0027*
0028*
0029  00020 102077  EXIT  HLT 77B
0030  00021 026001R       JMP NEXT
0031*
0032  00022 000001  UNIT  OCT 1        TTY LOGICAL UNIT
0033  00023 000000  INTGR BSS 1
0034  00024 000000  OCTAL BSS 1
0035  00025 000000  FLPTF BSS 2
0036  00027 000000  FLPTE BSS 2
0037                       END FRFLD
** NO ERRORS*
```

```
 PAGE 0002 #01  EXAMINATION OF ASSEMBLY LISTINGS


0001                    ASMB,R,B,L
0002*
0003* AN "*" IN COLUMN 1 INDICATES A COMMENT
0004*
0006  00000                NAM LSTNG


0008                       EXT .DIO.,.IOI.,.DTA.
0009                       ENT ENTRY
0010  00000 000000  ENTRY NOP
0011*
0012  00001 062056R        LDA UNIT+5    A = LOGICAL UNIT
0013  00002 006400        CLB           0 IN B INDICATES OUTPUT
0014  00003 016001X        JSB .DIO.     INITIALIZE FOR OUTPUT
0015  00004 002035R        DEF FMT       POINT TO DATA FORMAT
0016  00005 000011R        DEF EXIT      POINT TO ERROR EXIT
0017*
0018  00006 062040R        LDA X         GET NUMBER TO BE WRITTEN
0019  00007 016002X        JSB .IOI.     OUTPUT NUMBER
0020*
0021  00010 016003X        JSB .DTA.     INDICATE LAST RECORD
0022*

PG0000
UN  0023  EXIT  JSB ENDIO     WAIT FOR DEVICE TO WRITE
0023  00011 016000  EXIT  JSB ENDIO     WAIT FOR DEVICE TO WRITE
0024  00012 000013R        DEF *+1
0025*
0026  00013 102077        HLT 77B
0027  00014 026001R        JMP ENTRY+1
0028*
0029  00015 000000  DMY   BSS 2020B
0030*
0031  02035 024111  FMT   ASC 3,(I10)
      02036 030460
      02037 024440
0032*
0033  02040 000001  X     DEC 1
0034                 Y     REP 2
0035  02041 000001        OCT 1,2
      02042 000002
0035  02043 000001        OCT 1,2
      02044 000002
0036*
0037                       SUP           COMPRESSES ASC, DIV, FAD, FSB, OC
0038                       REP 2          DLD, FDV, MPY, DEC, DST, FMP
0039  02045 000001        OCT 1,2
0039  02047 000001        OCT 1,2
0040  02051 000001  UNIT  DEC 1,2,3,4,5,6
0041                       END

PG0002
**0001 ERRORS*
```

RESULTS OF TRYING TO EXECUTE A "BAD" RELOCATABLE TAPE

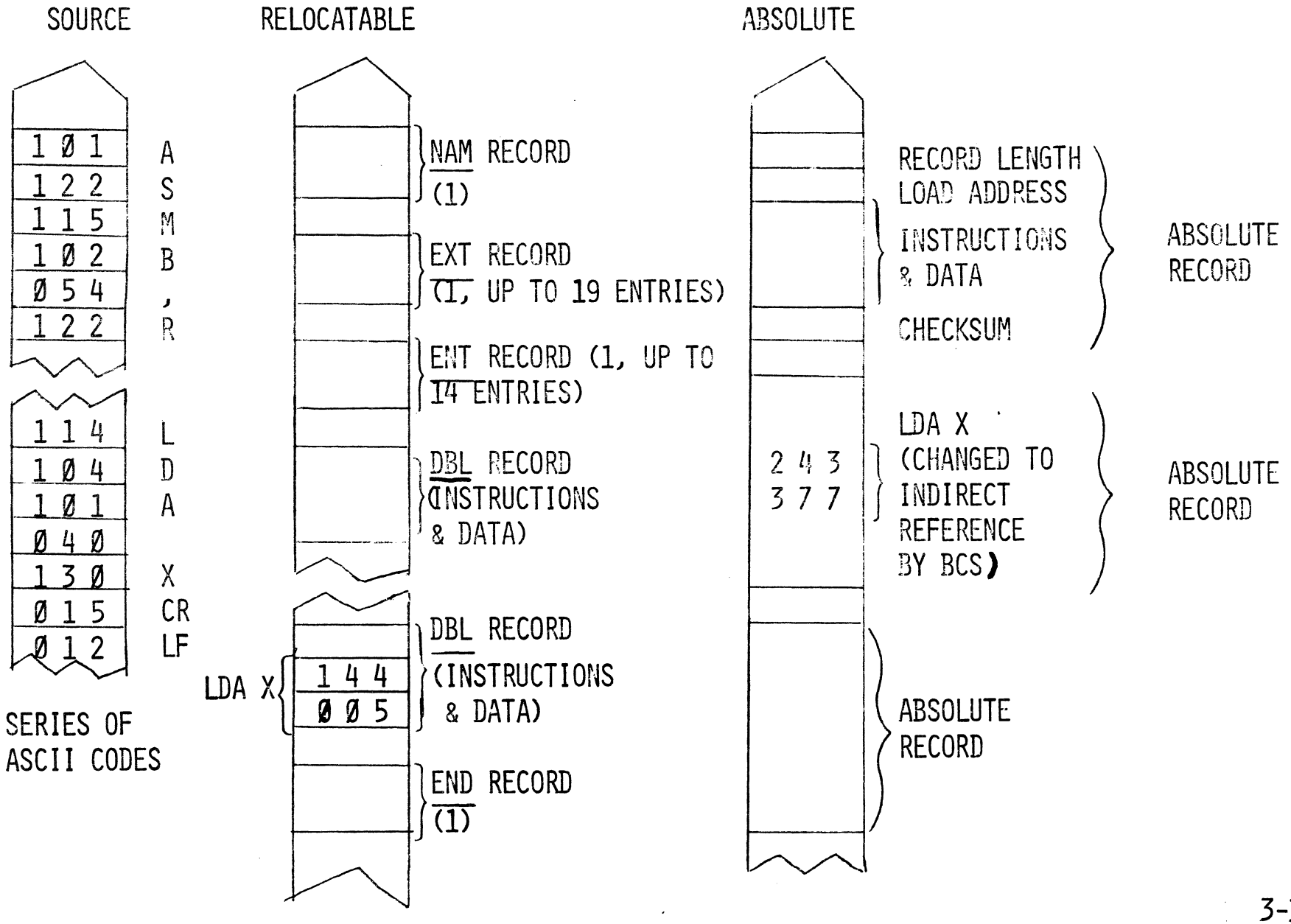
CONSOLE SHEET

THE PROGRAM SHOULD PRINT "1" ONCE & HALT. INSTEAD, IT PRINTS "1" ON SUCCEEDING LINES AD INFINITUM.

W H Y ?

```
 LSTNG

 02000 04060

*LOAD

*L08

*LST

*RUN
          1
          1
          1
          1
          1
          1
          1
          1
          1
          1
          1
          1
          1
          1
          1
     . . . .
```

TAPE FORMATS



DAY 3 READING ASSIGNMENT

POCKET GUIDE TO THE 2100 COMPUTER

ASSEMBLER

CHAPTER 3

APPENDIX E, PAGES E-1, E-2, E-3 (TOP HALF), ONLY

2100A REFERENCE

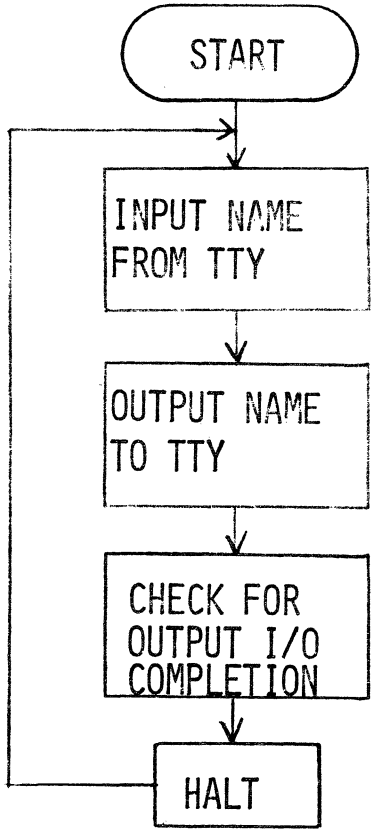
CHAPTER 2, PARA. 2.5 ONLY

CHAPTER 4

LAB 3.1 ECHO NAME (TTY TO TTY)

WRITE A PROGRAM USING THE FORMATTER THAT WILL INPUT YOUR NAME FROM THE TTY (LOGICAL UNIT 1) & WILL THEN OUTPUT YOUR NAME BACK TO THE TTY.

LAB 3.1 FLOWCHART



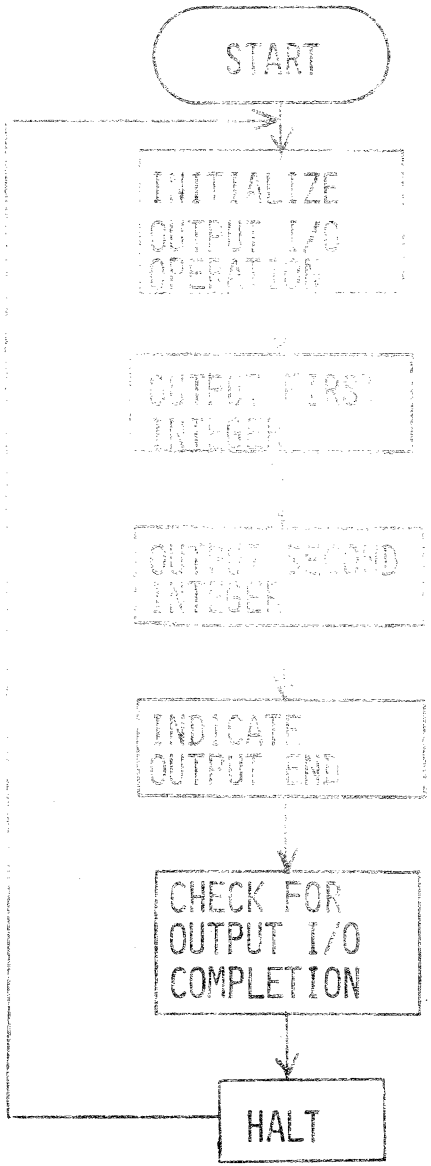
LAB 3.1 CONSOLE RUN SHEET

3-39

LAB 3.2 CONVERT & PUNCH NUMBERS

WRITE A PROGRAM THAT WILL OUTPUT 2 INTEGER VALUES ON THE PUNCH (LOGICAL UNIT 4) AS "FORMATTED" DATA. THE FORMAT SHOULD PROVIDE 3 DIGITS OF OUTPUT FOR THE FIRST INTEGER AND 4 DIGITS OF OUTPUT FOR THE SECOND INTEGER.

LAB 3.2 FLOWCHART

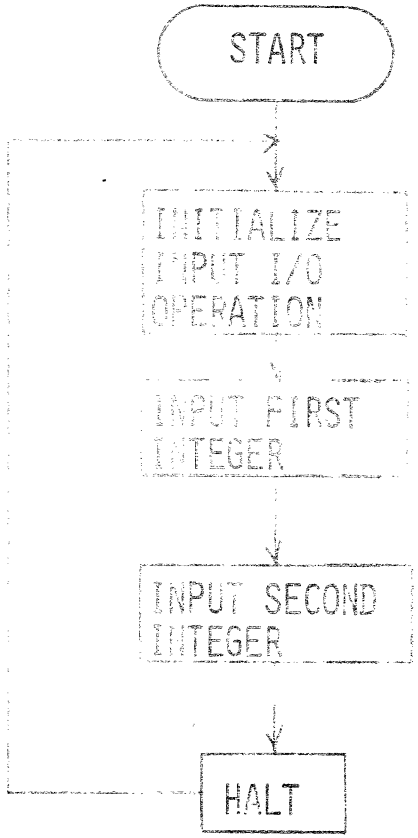


LAB 3.3 READ NUMBERS & CONVERT

WRITE A PROGRAM THAT USES THE FORMATTER TO READ THE TAPE PUNCHED BY LAB 3.2.

THE PHOTOREADER IS LOGICAL UNIT 5.

LAB 3.3 FLOWCHART



LAB 3.2 & 3.3 CONSOLE RUN SHEETS

LAB 3.2

```
PNCHN

02000 02025

*LOAD

*LST

*RUN
```

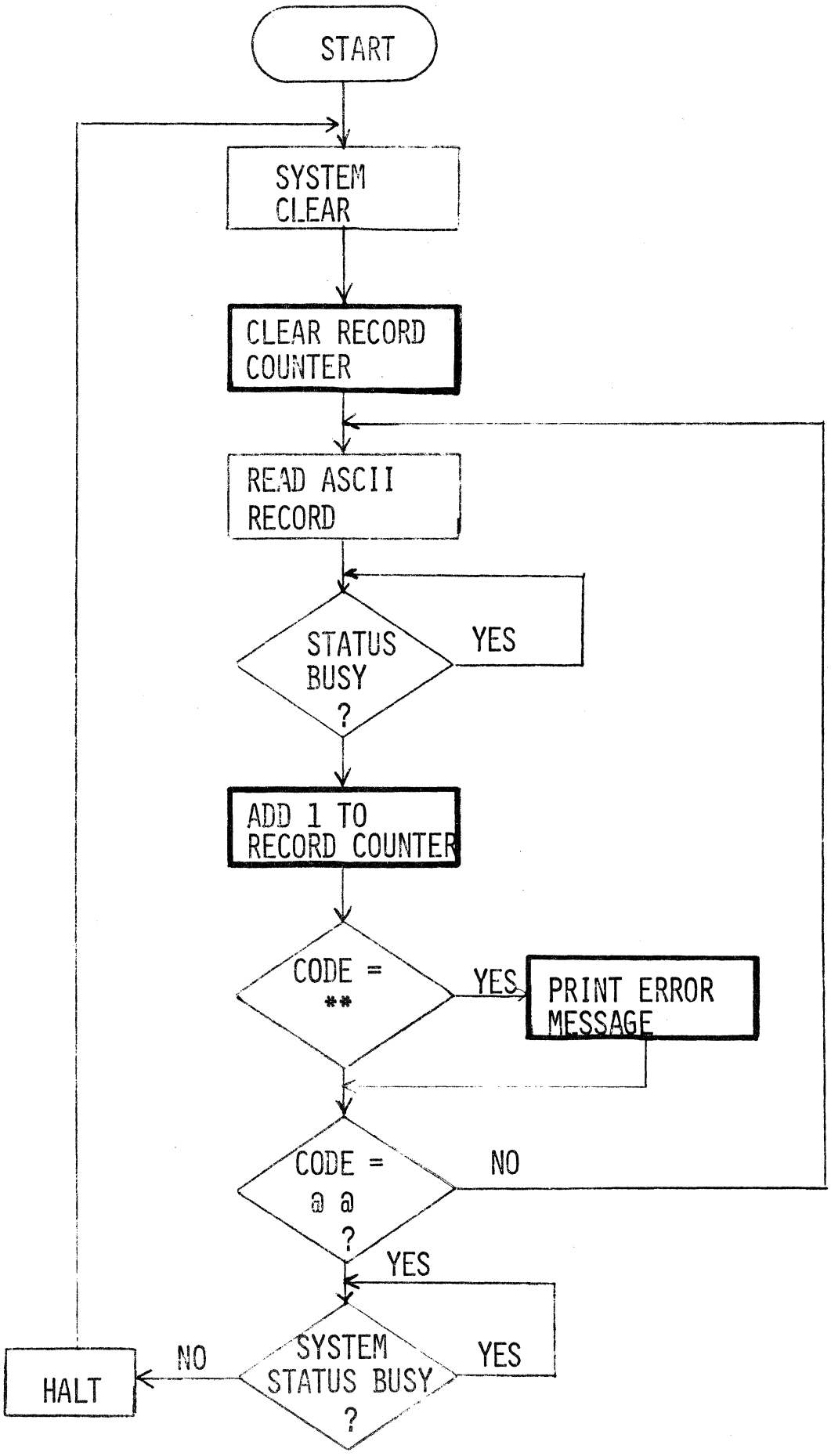
LAB 3.3

```
READN

02000 02022

*LOAD

*LST

*RUN
```

LAB 3.4 MODIFY LAB 2.3 TO RUN WITH FORMATTER

MODIFY LAB 2.3 (FIND ERRORS ON TAPE) SO THAT THE ERROR RECORD NUMBER WILL BE PRINTED IN ADDITION TO THE WORDS "ERROR MESSAGE".

LAB 3.4 FLOWCHART



LAB 3.4 CONSOLE RUN SHEET

```
  L34JH

  02000 05:04

*LOAD

*LIST

*LIST
  ERROR MESSAGE, RECORD     4
  ERROR MESSAGE, RECORD    17
  ERROR MESSAGE, RECORD    33
  ERROR MESSAGE, RECORD    34
  ERROR MESSAGE, RECORD    50
  ERROR MESSAGE, RECORD    76
  ERROR MESSAGE, RECORD    96
  ERROR MESSAGE, RECORD   122
  ERROR MESSAGE, RECORD   138
  ERROR MESSAGE, RECORD   163
```

LAB 3.1 SOLUTION

```
0001                    ASMB,R,B,L
0002*
0003* LAB 3.1 ECHO NAME (TTY TO TTY)
0004*
0005  00000                     NAM ECHO
0006                            EXT .DIO.,.IAR.,.DTA.,.IOC.
0007  00000 000000  START       NOP
0008*
0009* INPUT NAME FROM TTY
0010  00001 062032R AGAIN       LDA INLU .
0011  00002 006404              CLB,INB         INDICATES INPUT
0012  00003 016001X              JSB .DIO.       INITIALIZE I/O OPERATION
0013  00004 000034R              DEF FMT         ADDRESS OF FORMAT SPECIFICATION
0014  00005 000030R              DEF EXIT        ADDRESS OF ERROR EXIT
0015*
0016  00006 062037R              LDA NMBR        NUMBER OF ELEMENTS IN ARRAY
0017  00007 066040R              LDB BUFAD       ADDRESS OF BUFFER ADDRESS
0018  00010 016002X              JSB .IAR.       BEGIN INTEGER ARRAY INPUT
0019*
0020* OUTPUT NAME TO TTY
0021  00011 062033R              LDA OUTLU
0022  00012 006400               CLB             INDICATES OUTPUT
0023  00013 016001X              JSB .DIO.       INITIALIZE I/O OPERATION
0024  00014 000034R              DEF FMT         ADDRESS OF FORMAT SPECIFICATION
0025  00015 000030R              DEF EXIT        ADDRESS OF ERROR EXIT
0026*
0027  00016 062037R              LDA NMBR        NUMBER OF ELEMENTS IN ARRAY
0028  00017 066040R              LDB BUFAD       ADDRESS OF BUFFER ADDRESS
0029  00020 016002X              JSB .IAR.       BEGIN INTEGER ARRAY OUTPUT
0030*
0031  00021 016003X              JSB .DTA.       OUTPUT LAST RECORD
0032*
0033* CHECK FOR OUTPUT I/O COMPLETION
0034  00022 016004X              JSB .IOC.       STATUS TTY
0035  00023 040002               OCT 40002
0036  00024 002020               SSA             TTY BUSY ?
0037  00025 026022R              JMP *-3           YES, STATUS TTY AGAIN
0038*
0039* HALT
0040  00026 102077               HLT 77B           NO, HALT
0041  00027 026001R              JMP AGAIN
0042*
0043  00030 102000  EXIT          HLT 0B
0044  00031 026001R              JMP AGAIN
0045*
0046  00032 000001  INLU          OCT 1           INPUT  DEVICE LOGICAL UNIT
0047  00033 000002  OUTLU         OCT 2           OUTPUT DEVICE LOGICAL UNIT
0048                            SUP
0049  00034 024261  FMT           ASC 3,(15A2)
0050  00037 000017  NMBR          DEC 15
0051  00040 000041R BUFAD         DEF BFR
0052  00041 000000  BFR           BSS 15
0053                            END START
**  NO ERRORS*
```

LAB 3.2 SOLUTION

```
0001                    ASMB,B,L,R
0002*
0003* LAB 3.2 CONVERT & PUNCH NUMBERS
0004*
0005* 2 INTEGER VALUES ARE TO BE OUTPUT ON LOGICAL UNIT 4 AS
0006* "FORMATTED" DATA. THE LABELS FOR THE VALUES ARE "J" AND "K".
0007* THE FORMAT SHOULD PROVIDE 3 DIGITS OF OUTPUT FOR "J" AND 4
0008* DIGITS FOR "K".
0009*
0010  00000                 NAM PNCHN
0011                        EXT .DIO.,.IOI.,.DTA.
0012                        EXT ENDIO
0013  00000 000000  PNCHN NOP
0014*
0015* INITIALIZE OUTPUT I/O OPERATION
0016  00001 062017R AGAIN LDA UNIT
0017  00002 006400        CLB             INDICATES OUTPUT
0018  00003 016001X       JSB .DIO.       ASCII I-O OPERATION
0019  00004 000020R       DEF FMT         ADDRESS OF FORMAT SPECIFICATION
0020  00005 000015R       DEF EXIT        ADDRESS OF ERROR EXIT
0021*
0022* OUTPUT FIRST INTEGER
0023  00006 062024R       LDA J           PASS J TO .IOI.
0024  00007 016002X       JSB .IOI.       BEGIN INTEGER OUTPUT
0025*
0026* OUTPUT SECOND INTEGER
0027  00010 062025R       LDA K           PASS K TO .IOI.
0028  00011 016002X       JSB .IOI.       CONTINUE INTEGER OUTPUT
0029*
0030* INDICATE OUTPUT END
0031  00012 016003X       JSB .DTA.       OUTPUT LAST RECORD
0032*
0033* CHECK FOR I/O COMPLETION
0034  00013 016004X       JSB ENDIO       WAIT FOR SYSTEM I-O TO END
0035  00014 000015R       DEF *+1         RETURN ADDRESS
0036*
0037* HALT
0038  00015 102077  EXIT  HLT 77B
0039  00016 026001R       JMP AGAIN       RESTART
0040*
0041  00017 000004  UNIT  OCT 4           PUNCH LOGICAL UNIT
0042  00020 024111  FMT   ASC 4,(I3,I4) FORMAT SPECIFICATION
      00021 031454
      00022 044464
      00023 024440
0043  00024 000036  J     DEC 30
0044  00025 000037  K     DEC 31
0045                      END PNCHN
**  NO ERRORS*
```

LAB 3.3 SOLUTION

```
0001                    ASMB,B,L,R
0002*
0003* LAB 3.3 READ NUMBERS & CONVERT
0004*
0005* READ THE TAPE PUNCHED IN LAB 3.2
0006*
0007  00000                    NAM READN
0008                           EXT .DIO.,.IOI.
0009  00000 000000  READN NOP
0010*
0011* INITIALIZE INPUT I/O OPERATION
0012  00001 062014R AGAIN LDA UNIT
0013  00002 006404         CLB,INB        INDICATES INPUT
0014  00003 016001X         JSB .DIO.      ASCII I-O OPERATION
0015  00004 000015R         DEF FMT        ADDRESS OF FORMAT SPECIFICATION
0016  00005 000012R         DEF EXIT       ADDRESS OF ERROR EXIT
0017*
0018* INPUT FIRST INTEGER
0019  00006 016002X         JSB .IOI.      GET J FROM .IOI.
0020  00007 072021R         STA J          STORE J
0021*
0022* INPUT SECOND INTEGER
0023  00010 016002X         JSB .IOI.      GET K FROM .IOI.
0024  00011 072022R         STA K          STORE K
0025*
0026* HALT
0027  00012 102077  EXIT   HLT 77B
0028  00013 026001R         JMP AGAIN      RESTART
0029*
0030  00014 000005  UNIT   OCT 5           PHOTOREADER LOGICAL UNIT
0031  00015 024111  FMT    ASC 4,(I3,I4) FORMAT SPECIFICATION
      00016 031454
      00017 044464
      00020 024440
0032  00021 000000  J      NOP
0033  00022 000000  K      NOP
0034                        END READN
**  NO ERRORS*
```

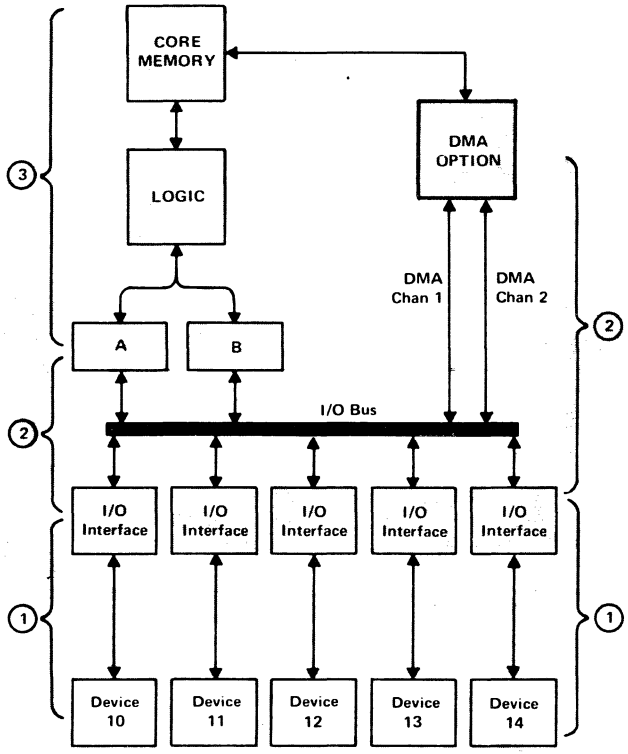
LAB 3.4 SOLUTION (1 OF 2)

```
0001                    ASMB,D,E,L
0002*
0003* LAB 3.4 MODIFY LAB 2.3 (FIND ERRORS ON TAPE) TO RUN WITH
0004*   FORMATTER
0005*
0006  00000                    NAM LOOP
0007                           ENT .IOC.
0008  00000 000000  START  NOP
0009  00001 016001X AGAIN  JSB .IOC.     SYSTEM CLEAR
0010  00002 000003         OCT 3
0011*
0012  00003 002400         CLA           CLEAR RECORD COUNTER
0013  00004 072063R         STA CNTR
0014*
0015  00005 016001X READ   JSB .IOC.     READ ASCII RECORD
0016  00006 010005         OCT 10005
0017  00007 026005R         JMP READ
0018  00010 000050R         DEF RDBUF
0019  00011 177754          DEC -20
0020*
0021  00012 016001X STAT   JSB .IOC.     STATUS BUSY ? YES, WAIT
0022  00013 040005         OCT 40005
0023  00014 002020         SSA
0024  00015 026012R         JMP STAT
0025*
0026  00016 062063R         LDA CNTR      ADD 1 TO RECORD COUNTER
0027  00017 002004         INA
0028  00020 072063R         STA CNTR
0029*
0030  00021 062057R         LDA C1516     CODE = ** ?
0031  00022 052133C         CPA =A**
0032  00023 016036R         JSB WRMSG       YES, PRINT ERROR MESSAGE
0033*
0034  00024 062050R         LDA RDBUF     CODE = 00 ?
0035  00025 052104R         CPA =A00
0036  00026 026030R         JMP SHTDN       YES, START SHUTDOWN
0037  00027 026005R         JMP READ        NO, READ NEXT RECORD
0038*
0039  00030 016001X SHTDN  JSB .IOC.     SYSTEM STATUS BUSY ?
0040  00031 040000         OCT 40000
0041  00032 002020         SSA             NO, HALT
0042  00033 026030R         JMP SHTDN       YES, WAIT
0043*
0044  00034 102077         HLT 77B
0045*
0046  00035 026001R         JMP AGAIN     RESTART
```

LAB 3.4 SOLUTION (2 OF 2)

```
0047  00036 000000  WEMSG NOP         . PRINT ERROR MESSAGE SUBROUTINE
0048                      EXT .DIO.,.IOI.,.DTA.                          ←
0049  00037 062062R       LDA UNIT                                       ←
0050  00040 006400        CLB           INDICATES OUTPUT                 ←
0051  00041 016002X       JSB .DIO.     ASCII I/O OPERATION              ←
0052  00042 000064R       DEF FMT       ADDRESS OF FORMAT SPECIFICATION  ←
0053  00043 000047R       DEF EXIT      ADDRESS OF ERROR EXIT            ←
0054*
0055  00044 062063R       LDA CNTR      GET RECORD NUMBER                ←
0056  00045 016003X       JSB .IOI.     OUTPUT ERROR MESSAGE             ←
0057*
0058  00046 016004X       JSB .DTA.     OUTPUT LAST RECORD               ←
0059*
0060  00047 126036R EXIT  JMP WEMSG,I                                    ←
0061*
0062  00050 000000  RDBUF  BSS 7
0063  00057 000000  C1516  BSS 3
0064  00062 000002  UNIT   OCT 2         TTY LOGICAL UNIT                 ←
0065                      SUP
0066  00063 000000  CNTR   BSS 1         RECORD COUNTER                   ←
0067  00064 024042  FMT    ASC 15,(" ERROR MESSAGE, RECORD ",I4)          ←
0068                      END START
**  NO ERRORS*
```

INPUT/OUTPUT SYSTEM



INPUT/OUTPUT METHODS

SINCE MOST INPUT/OUTPUT DEVICES ARE MUCH SLOWER THAN THE COMPUTER, THE STRUCTURE OF THE COMPUTER PROVIDES FOR TWO DISTINCT METHODS OF INPUT/OUTPUT DATA TRANSFERS.

NON-INTERRUPT METHOD

THE USER COMMANDS THE I/O DEVICE TO CYCLE AND THEN WAITS FOR THE DEVICE CYCLE TO COMPLETE BEFORE PROCEEDING WITH FURTHER EXECUTION OF THE PROGRAM.

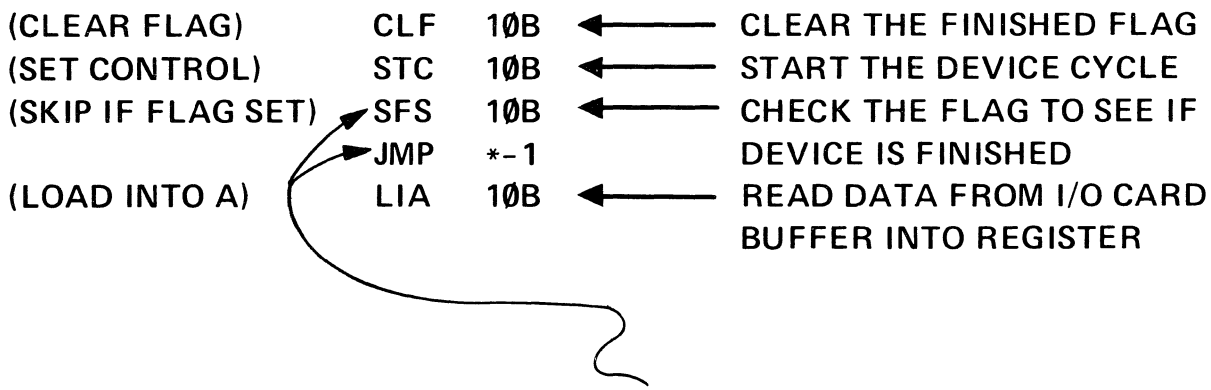
ADVANTAGE: EASY TO USE
DISADVANTAGE: INEFFICIENT USE OF COMPUTER TIME

INTERRUPT METHOD

THE USER COMMANDS THE I/O DEVICE TO CYCLE AND THEN IMMEDIATELY PROCEEDS WITH EXECUTION OF THE PROGRAM. THE COMPLETION OF THE DEVICE CYCLE INTERRUPTS PROGRAM EXECUTION AND TRANSFERS CONTROL TO A SUBROUTINE WHICH HANDLES THE DATA TRANSFER.

ADVANTAGE: EFFICIENT USE OF COMPUTER TIME.
DISADVANTAGE: REQUIRES MORE PROGRAMMING EFFORT.

READING ONE CHARACTER FROM THE PHOTO READER

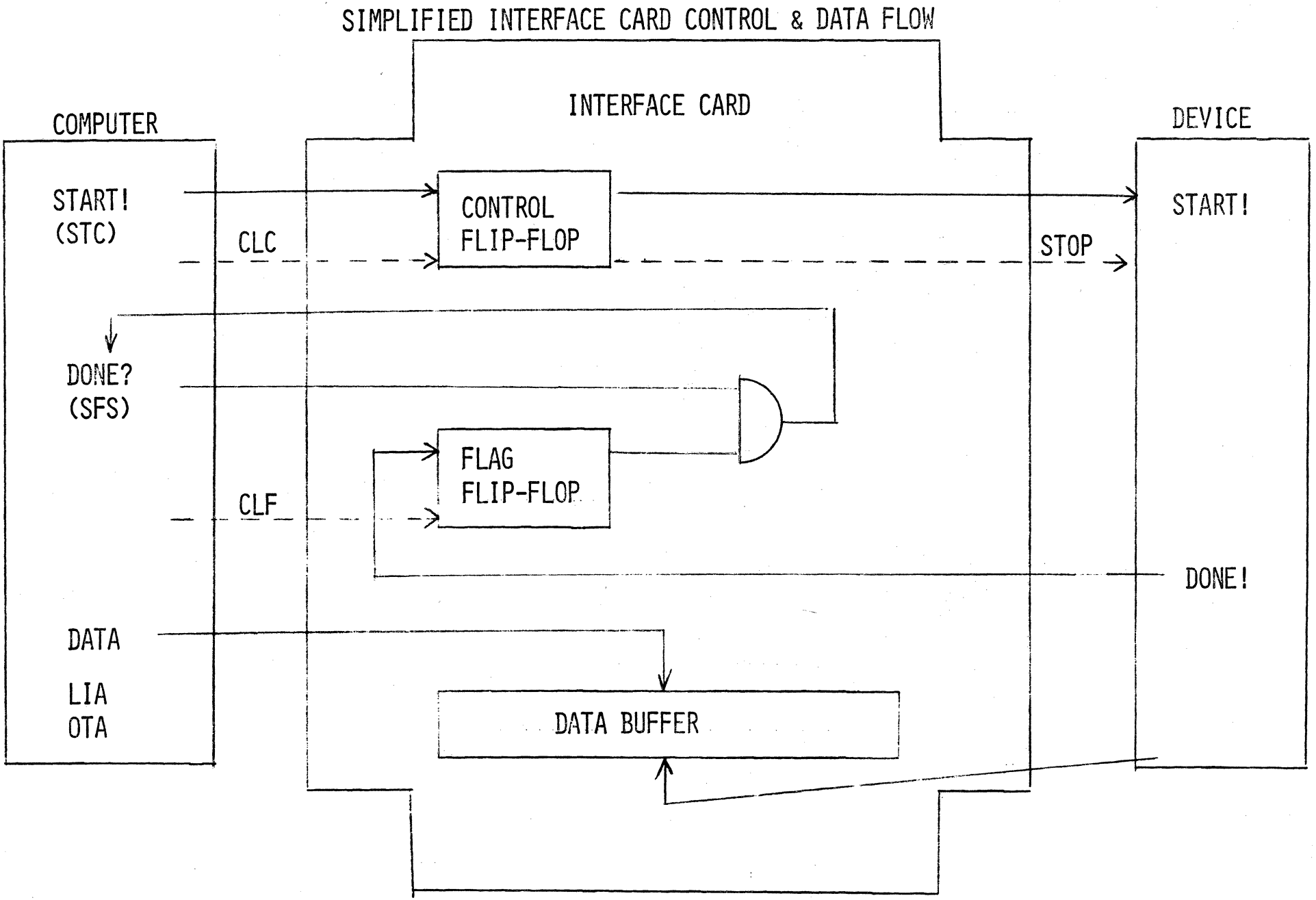


THIS IS CALLED THE "WAIT FOR FLAG" LOOP.

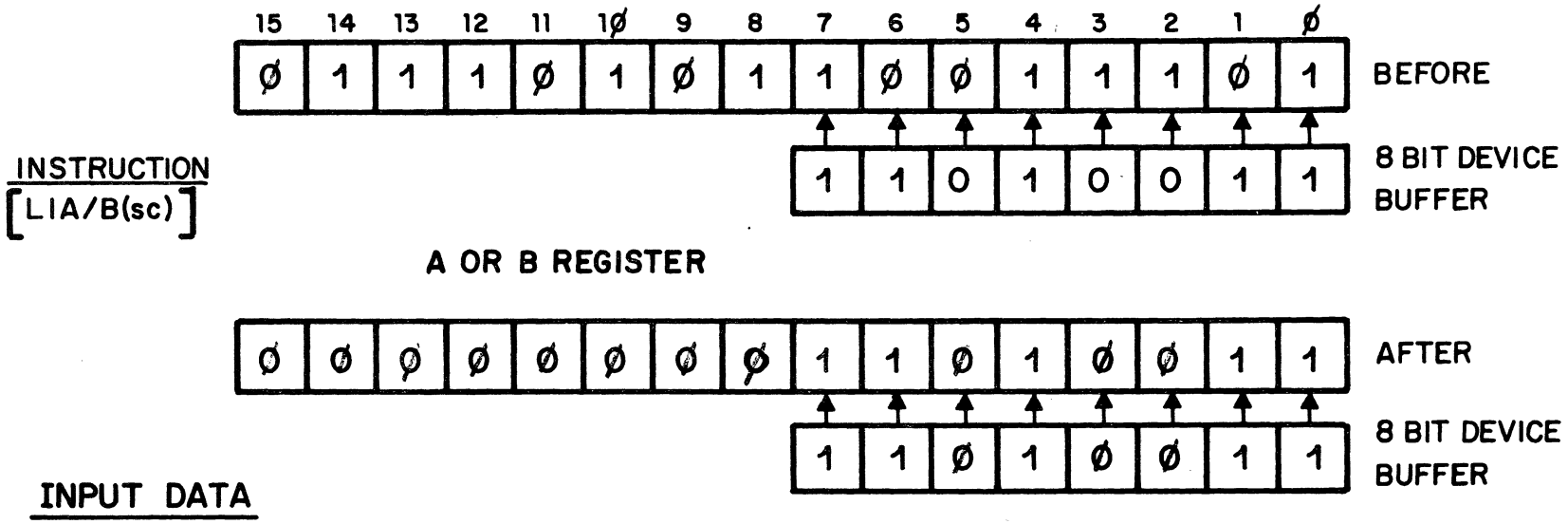
THE CLF 10B IS REQUIRED SINCE THE FLAG FLIP-FLOP IS NORMALLY SET WHEN THE DEVICE IS NOT IN USE. (TURNING COMPUTER POWER ON OR PRESSING EXTERNAL PRESET SETS THE FLAG FLIP-FLOP)

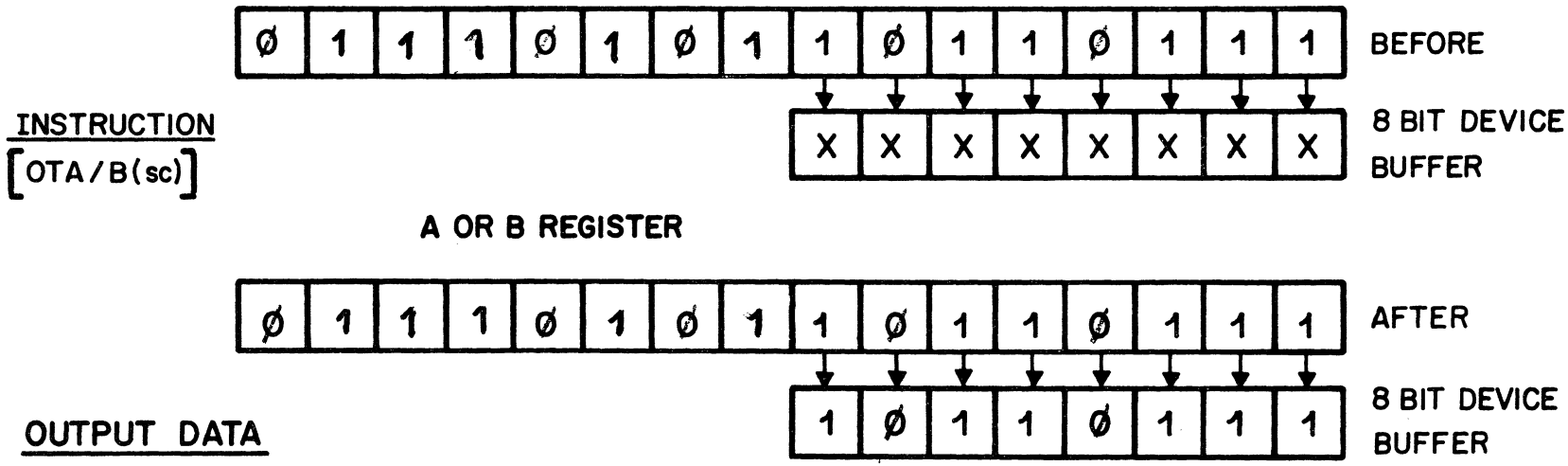
THE CLF 10B INSTRUCTION MAY BE COMBINED WITH THE STC 10B

STC 10B,C SET CONTROL & CLEAR FLAG


SIMPLIFIED INTERFACE CARD CONTROL & DATA FLOW
COMPUTER
INTERFACE CARD
DEVICE
START!
(STC)
CLC
CONTROL
FLIP-FLOP
START!
STOP
DONE?
(SFS)
FLAG
FLIP-FLOP
CLF
DONE!
DATA
LIA
OTA
DATA BUFFER

DATA TRANSFERS (8 BIT DEVICE)





A DATA TRANSFER EXAMPLE

THE PROGRAM BELOW WILL READ TWO (8 LEVEL) PAPER TAPE CHARACTERS AND PACK THEM INTO ONE 16 BIT WORD.
THE PROGRAM ASSUMES A PHOTO READER IS AVAILABLE AND ASSIGNED A SELECT CODE OF 17_8

LABEL	OPCODE	OPERAND	REMARKS
	STC	17B,C	START READER
WAIT 1	SFS	17B	IS FLAG SET?
	JMP	WAIT 1	NO, STAY IN LOOP
	LIA	17B	YES, LOAD 1ST 8 BITS
	ALF, ALF		ROTATE TO HI "A"
	STC	17B,C	START READER
WAIT 2	SFS	17B	IS FLAG SET?
	JMP	WAIT 2	NO, STAY IN LOOP
	MIA	17B	YES, MERGE 2ND 8 BITS
	HLT		HALT

ALF = A left 4

THE EQU PSEUDO INSTRUCTION

THE EQU PSEUDO INSTRUCTION EQUATES THE OPERAND SYMBOL TO THE LABEL FIELD.

FOR EXAMPLE:

LOCATION	CONTENTS	LABEL	OPCODE	OPERAND	REMARKS
02000			ORG	2000B	
00017		READR	EQU	17B	READR "EQUALS" 17B
02000	103717		STC	READR,C	
02001	102317		SFS	READR	
02002	026001		JMP	*-1	
02003	102517		LIA	READR	
02004	102077		HLT	77B	

NOTE:

THE VALUE OF THE LABEL "READR" DEPENDS ENTIRELY ON THE OPERAND OF THE EQU PSEUDO INSTRUCTION. ALL OPERAND REFERENCES TO THE "READR" SYMBOL ARE ASSIGNED THE VALUE 17_8.

SUMMARY OF IOG INSTRUCTIONS

15	14 13 12	11	10	9	8 7 6	5 4 3	2 1 0
1	IOG 000		1	H/C	HLT 000	← Select Code →	
			1	0	STF 001		
			1	1	CLF 001		
			1	0	SFC 010		
			1	0	SFS 011		
		A/B	1	H/C	MI* 100		
		A/B	1	H/C	LI* 101		
		A/B	1	H/C	OT* 110		
		0	1	H/C	STC 111		
		1	1	H/C	CLC 111		
			1	0	STO 001	000	001
			1	* 1	CLO 001	000	001
			1	* H/C	SOC 010	000	001
			1	* H/C	SOS 011	000	001

H/C REFERS TO THE FLAG.

BIT 9 = 1, CLEAR THE DEVICE FLAG

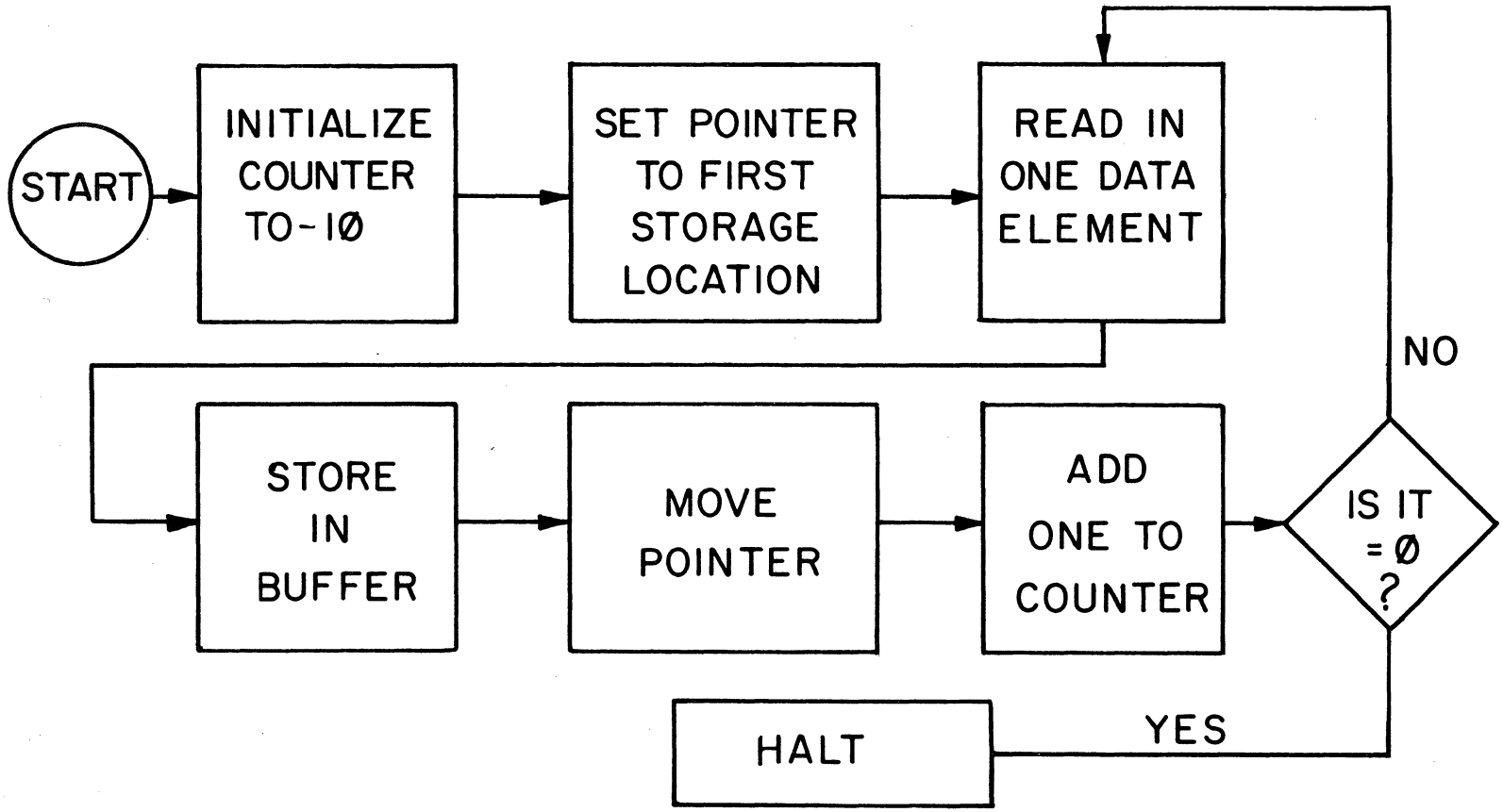
BIT 9 = Ø, HOLD THE DEVICE FLAG AS IS

* H = HOLD OVERFLOW AS IS

* C = CLEAR OVERFLOW AT END OF EXECUTION.

PROGRAM EXAMPLE-FLOWCHART

READ 10 DATA ELEMENTS AND STORE THEM IN CONSECUTIVE MEMORY LOCATIONS.



PROGRAM EXAMPLE-CODE

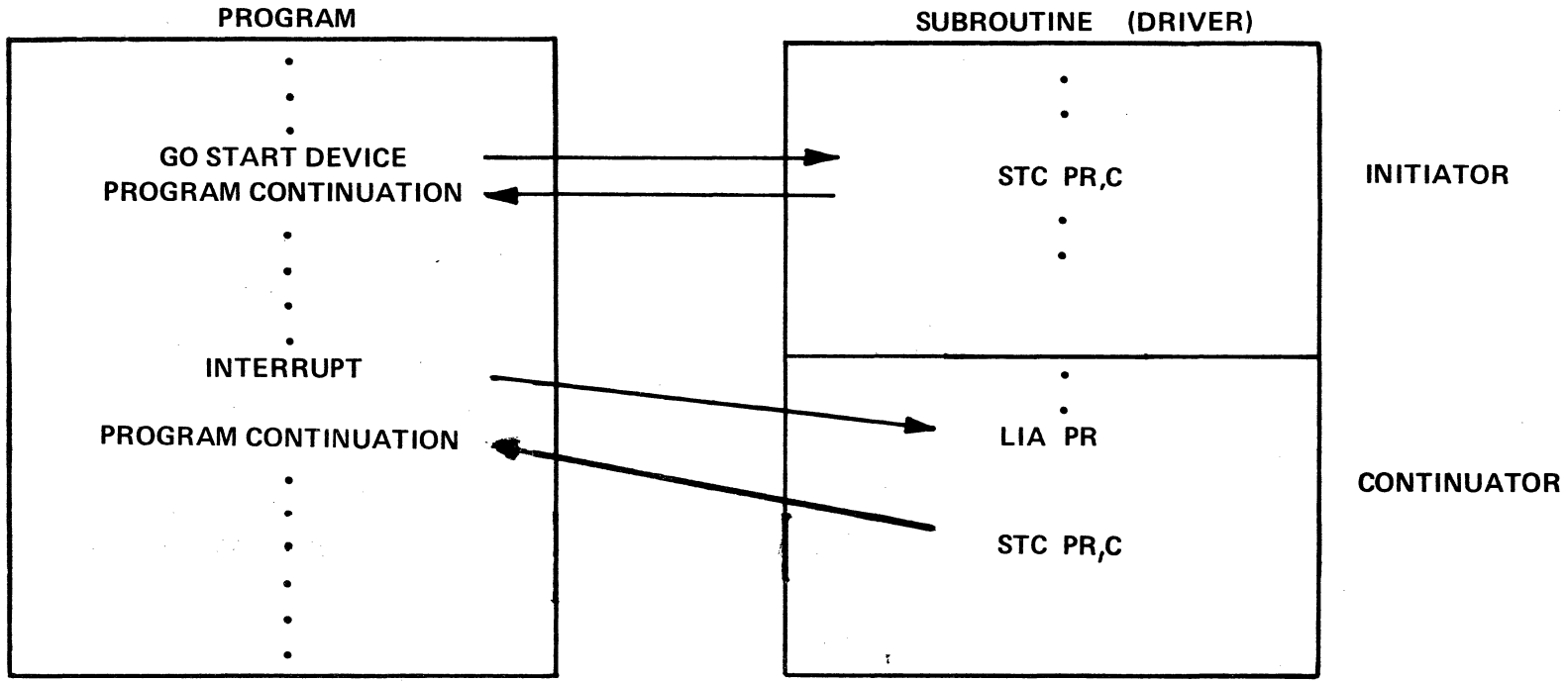
```
                    ASMB,A,B,L
00100                           ORG 100B
00100  060133  START    LDA D10      GET COUNT CONSTANT
00101  070130           STA CNTR     PUT IN WORKING COUNTER
00102  060132           LDA BUFAD    GET STARTING ADDRESS
00103  070131           STA PNTR     PUT IN ADDRESS POINTER
00104  103721  GO       STC INPUT,C  START I/O DEVICE
00105  102321           SFS INPUT    WAIT FOR
00106  024105           JMP *-1      FLAG
00107  102521           LIA INPUT    READ IN DATA
00110  170131           STA PNTR,I   STORE IN BUFFER
00111  034131           ISZ PNTR     MOVE ADDRESS POINTER
00112  034130           ISZ CNTR     INCREMENT COUNTER
00113  024104           JMP GO       NOT EQUAL TO ZERO
00114  102077           HLT 77B      STOP
00115  024100           JMP START    RESTART
00021          INPUT     EQU 21B      I/O DEVICE IN 21B
00116  000000  BUFFR     BSS 10       RESERVE FOR DATA
00130  000000  CNTR      BSS 1        RESERVE FOR COUNTER
00131  000000  PNTR      BSS 1        RESERVE FOR POINTER
00132  000116  BUFAD     DEF BUFFR    FIRST ADDRESS OF BUFFER
00133  177766  D10       DEC -10      DECIMAL -10
                         END
```

THE INTERRUPT SYSTEM

WE HAVE DISCUSSED INPUT/OUTPUT USING THE WAIT FOR FLAG TECHNIQUE, BUT FIND THAT THE COMPUTER COULD SPEND A MAJOR PORTION OF ITS TIME IN A WAIT LOOP.

IN MANY CASES THIS FACT IS OF NO CONSEQUENCE, BUT THERE IS A WAY TO CIRCUMVENT THIS LOSS OF COMPUTER TIME BY GIVING THE PERHIPERAL DEVICE A WAY OF REQUESTING SERVICE ONLY WHEN IT NEEDS IT

INTERRUPT METHOD



USING AN I/O MONITOR

BCS → INPUT/OUT CONTROL

JSB .IOC.

DOS → CENTRAL INTERRUPT CONTROL

JSB EXEC

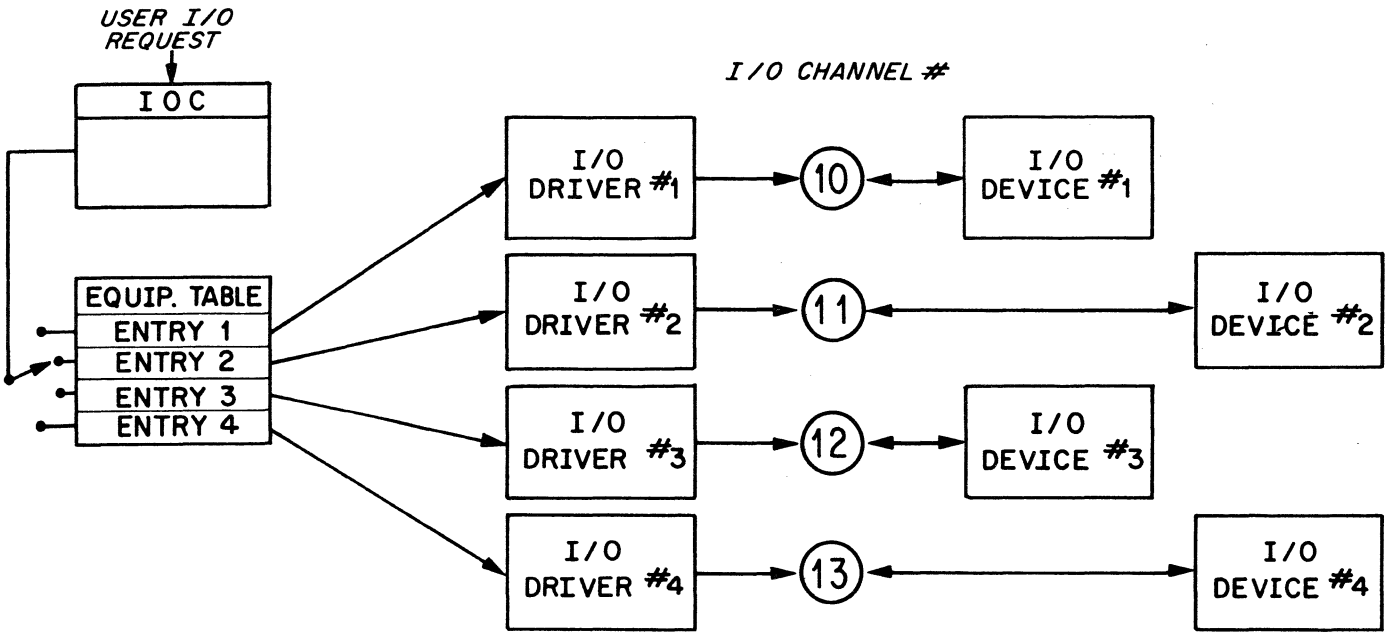
LOGICAL UNIT NUMBERS

PROVIDE INDEPENDENCE FROM I/O CHANNEL NUMBERS (SELECT CODES).

A NUMBER ($1 - 77_8$) IS ASSIGNED TO EACH DEVICE IN THE SYSTEM DURING GENERATION OF THE SYSTEM.

THE OPERATING SYSTEM REQUIRES LOGICAL UNITS 1-6 TO HAVE SPECIFIC FUNCTIONS.

SIMPLIFIED IOC BLOCK DIAGRAM



① The user requests an I/O operation using a logical unit reference number.

② IOC finds the logical unit entry in the equipment table.

③ The equipment table entry contains the address of the driver and the physical channel number of the device.

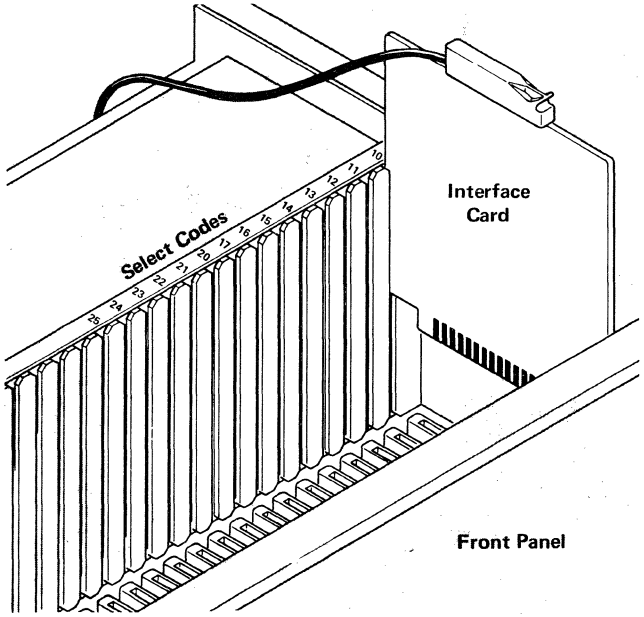
SYSTEM SELECT CODES & INTERRUPT ADDRESSES

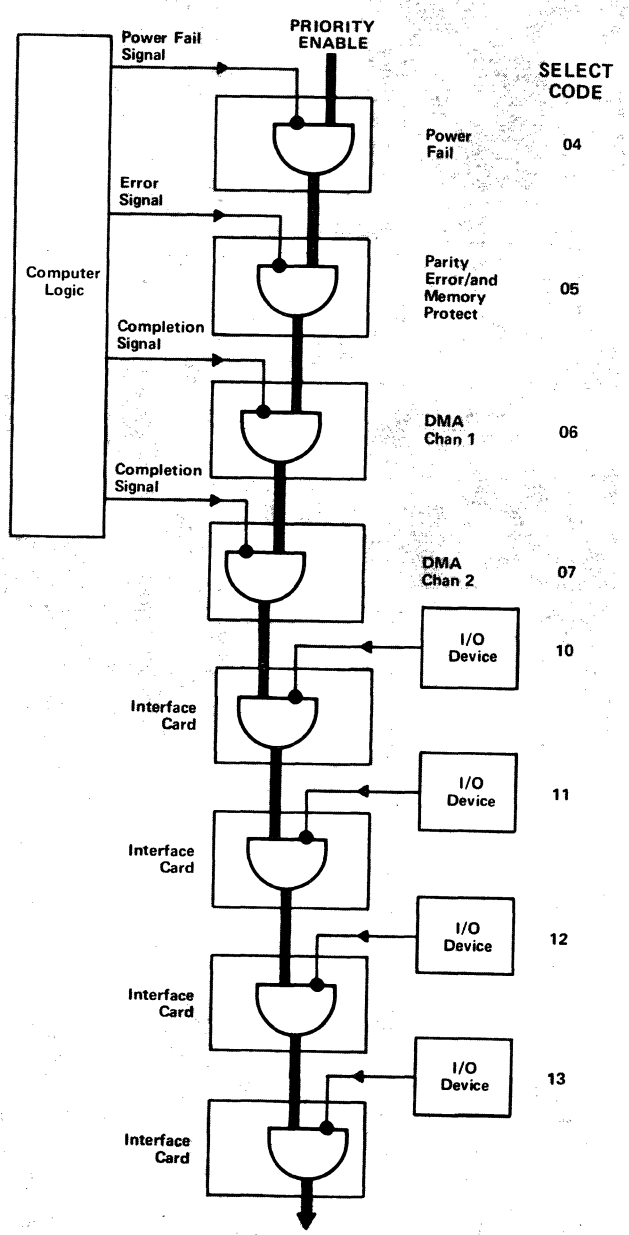
SELECT CODE	INTERRUPT LOCATION (TRAP CELL)	FUNCTIONAL ASSIGNMENTS
Ø	NONE	ENABLE/DISABLE I/O AND INT SYST.
1	NONE	SWITCH REGISTER
2	NONE	DMA CM1
3	NONE	DMA CM2
4	4	POWER FAIL (HIGHEST PRIORITY)
5	5	MEMORY PROTECT
6	6	DMA CM1
7	7	DMA CM2
1Ø	1Ø	I/O DEVICE
11	11	I/O DEVICE
12	12	I/O DEVICE
•	•	•
•	•	•
•	•	
77	77	I/O DEVICE (LOWEST PRIORITY)

THE INTERRUPT SYSTEM HAS A PRIORITY CHAIN BEGINNING WITH SELECT CODE 4B.

WHEN A DEVICE INTERRUPTS, LOWER PRIORITY DEVICES ARE CUT OFF FROM THIS CHAIN UNTIL THE INTERRUPT IS PROCESSED.

PRIORITY LINKAGE





INTERRUPT & I/O CONTROL SUMMARY (1 OF 2)

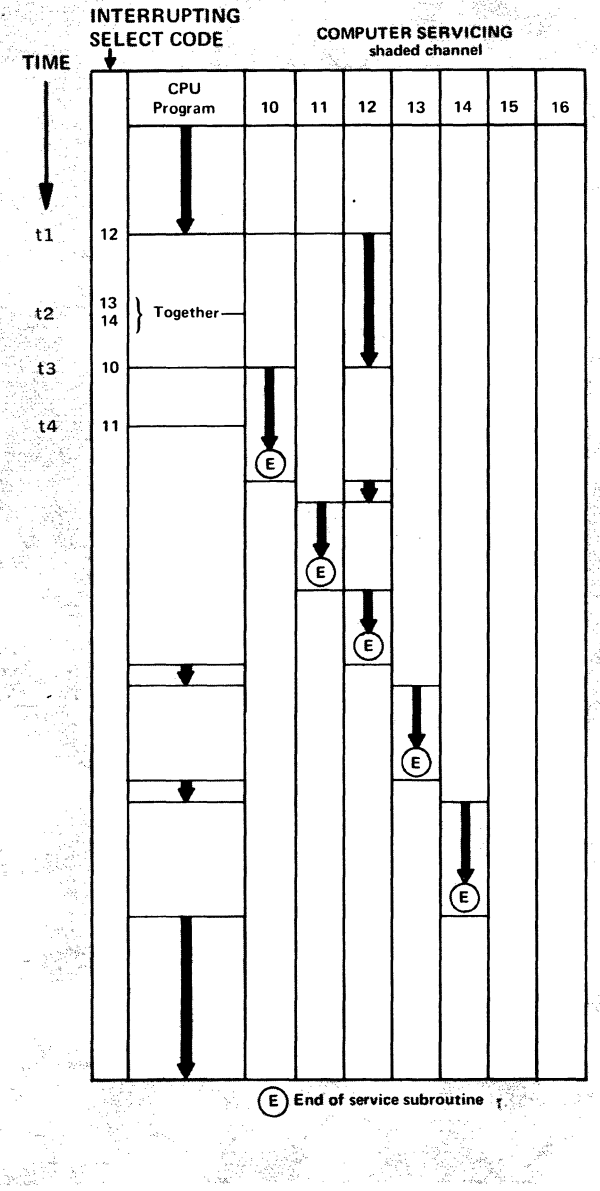
switch registers

INST	S.C. 00	S.C. 01	S.C. 02	S.C. 03
STC	NOP	NOP	Prepares DMA channel 1 to receive and store the block length in 2's complement form.	Prepares DMA channel 2 to receive and store the block length in 2's complement form.
CLC	Clears all Control FF's from S.C. 06 and up; effectively turns off all I/O devices.	NOP	Prepares DMA channel 1 to receive and store the direction of data flow and the starting memory address.	Prepares DMA channel 2 to receive and store the direction of data flow and the starting memory address.
STF	Turns on interrupt system.	STO sets overflow bit.	NOP	NOP
CLF	Turns off interrupt system except power fail (S.C. 04) and parity error (S.C. 05).	CLO clears overflow bit.	NOP	NOP
SFS	Skip if interrupt system is on.	SOS	NOP	NOP
SFC	Skip if interrupt system is off.	SOC	NOP	NOP
LIA/B	NOP	Loads S-register contents into A/B register.	Loads present contents of DMA channel 1 word count register into A/B register.	Loads present contents of DMA channel 2 word count register into A/B register.
MIA/B	NOP	Merges S-register contents into A/B register.	Merges present contents of DMA channel 1 word count register into A/B register.	Merges present contents of DMA channel 2 word count register into A/B register.
OTA/B	NOP	Outputs A/B register contents into display register.	1. Outputs to DMA channel 1 the block length in 2's complement form (previously prepared by an STC 02 instruction). 2. Outputs to DMA channel 1 the direction of data flow and the starting memory address (previously prepared by a CLC 02 instruction).	1. Outputs to DMA channel 2 the block length in 2's complement form (previously prepared by an STC 03 instruction). 2. Outputs to DMA channel 2 the direction of data flow and the starting memory address (previously prepared by a CLC 03 instruction).

INTERRUPT AND I/O CONTROL SUMMARY (2 OF 2)

	S.C. 04 (Power fail)	S.C. 05 (Parity error)	S.C. 06	S.C. 07	S.C. 10-77
STC	Re-initializes power-fail logic and restores interrupt capability to lower priority functions.	Turns on memory protect.	Sets Control FF on DMA channel 1 (activates DMA).	Sets Control FF on DMA channel 2 (activates DMA).	Sets PCA Control FF and turns on device on channel specified by S.C.
CLC	Re-initialize power-fail logic.	NOP	DMA. Control FF on DCPC channel 1 (reestablishes priority with STF; does not turn off DMA.	Clears Control FF on DMA channel 2 (reestablishes priority with STF; does not turn off DMA).	Clears PCA Control FF and turns off device (aborts data transfer if programmed).
STF	Flag FF sets automatically when power comes up. (No program control possible.)	Turns on parity error interrupt.	1. Hardware Controlled: Sets Flag FF when word count is reached and turns off DMA channel 1. 2. Program Controlled: Aborts data transfer.	1. Hardware Controlled: Sets Flag FF when word count is reached and turns off DMA channel 2. 2. Program Controlled: Aborts data transfer.	PCA Flag FF sets when data transfer is complete.
CLF	Flag FF clears automatically when power fail occurs. (No program control possible.)	Turns off parity error interrupt.	Clears Flag FF on DMA channel 1.	Clears Flag FF on DMA channel 2.	Clears PCA Flag FF.
SFS	NOP	Skip if parity error interrupt is on.	Tests if DMA channel 1 data transfer is complete.	Tests if DMA channel 2 data transfer is complete.	Tests if I/O channel data transfer is complete.
SFC	Skip if power fail has occurred.	Skip if parity error interrupt is off.	Tests if DMA channel 1 data transfer is still in progress.	Tests if DMA channel 2 data transfer is still in progress.	Tests if I/O channel data transfer is still in progress.
LIA/B	Loads contents of central interrupt register (S.C. of last interrupting device) into least-significant bits of A/B register.	Loads contents of violation register into A/B register: Bit 15 = 1 = PE Bit 15 = 0 = MPV	NOP	NOP	Loads contents of PCA buffer into A/B register.
MIA/B	Merges contents of central interrupt register into least-significant bits of A/B register.	Merges contents of violation register into A/B register.	NOP	NOP	Merges contents of PCA buffer into A/B register.
OTA/B	NOP	Outputs first addressable memory location to fence register.	Outputs to DMA channel 1 the S.C. of I/O channel. Specify STC after each word; CLC after block.	Outputs to DMA channel 2 the S.C. of I/O channel. Specify STC after each word; CLC after block.	Outputs data from A/B register into PCA buffer.

INTERRUPT SEQUENCES



I O R *m*

merge

"INCLUSIVE OR" THE CONTENTS OF MEMORY LOCATION *m* & THE A-REGISTER.

A-REGISTER	BEFORE "IOR *m*" EXECUTED	1 111 111 1Ø1 01Ø
CONTENTS OF *m*	BEFORE "IOR *m*" EXECUTED	Ø 111 Ø1Ø ØØ1 1ØØ
A-REGISTER	AFTER "IOR *m*" EXECUTED	1 111 111 1Ø1 11Ø

CONTENTS OF *m* UNCHANGED

RULE FOR "INCLUSIVE OR"

```
1 Ø 1 Ø
1 1 Ø Ø
↓ ↓ ↓ ↓
1 1 1 Ø
```

X O R *m*

"EXCLUSIVE OR" THE CONTENTS OF MEMORY LOCATION *m* & THE A-REGISTER. add

A-REGISTER	BEFORE "XOR *m*" EXECUTED	1	111	111	101	010
CONTENTS OF *m*	BEFORE "XOR *m*" EXECUTED	0	111	010	001	100
A-REGISTER	AFTER "XOR *m*" EXECUTED	1	000	101	100	110

CONTENTS OF *m* UNCHANGED

test for ≠
if not equal = 1
if equal = 0 unequal = 1

RULE FOR "EXCLUSIVE OR"

1 0 1 0
1 1 0 0
↓ ↓ ↓ ↓
0 1 1 0

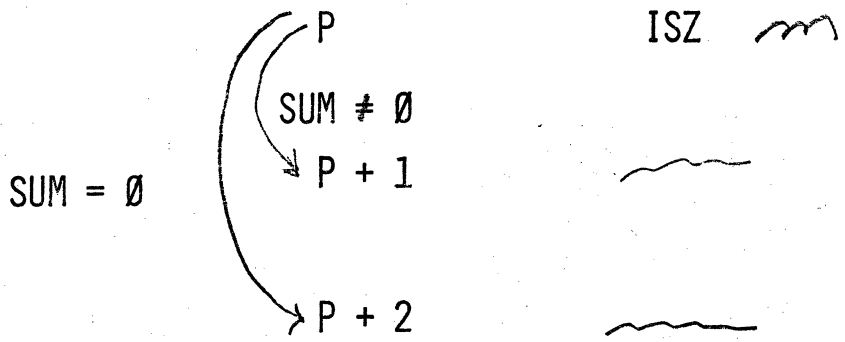
I S Z m

ADD 1 TO CONTENTS OF MEMORY LOCATION m.

IF THE RESULT OF THIS ADDITION IS ZERO,

THEN SKIP THE NEXT LOCATION

ELSE PROCEED TO THE NEXT LOCATION



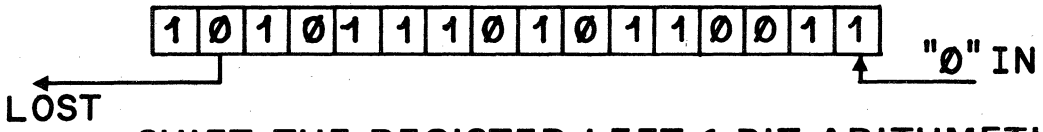
IOR.XOR.ISZ (CLASSROOM EXERCISE)

MEMORY LOCATION			
2000	START	LDB	REPS
		CLA	
	AGAIN	IOR	IMASK
		XOR	XMASK
		ISZ	1
		JMP	AGAIN
		HLT	
	REPS	OCT	-2
	IMASK	OCT	5307
	XMASK	OCT	7037

THE PROGRAM STARTS AT START. WHAT WILL BE IN T, P, A, & B WHEN EXECUTION STOPS T = 10200 , P = 2007 , A = 300 , B = 0

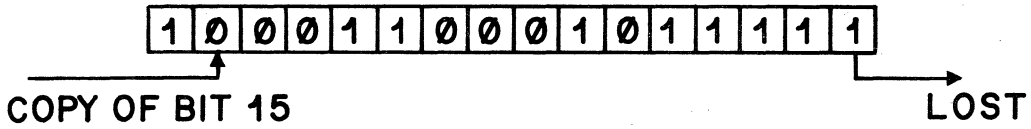
SHIFT & ROTATE INSTRUCTIONS

ALS / BLS



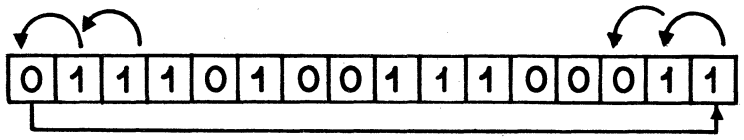
SHIFT THE REGISTER LEFT 1 BIT ARITHMETICALLY. BITS 0 THRU 14 SHIFT LEFT. SIGN, BIT 15, IS NOT AFFECTED.

ARS / BRS



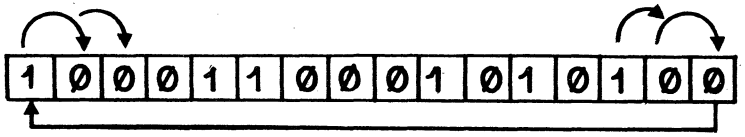
SHIFT THE REGISTER RIGHT 1 BIT ARITHMETICALLY. BITS 14 THRU 0 SHIFT RIGHT. SIGN, BIT 15, IS NOT AFFECTED. COPY OF SIGN BIT SHIFTED INTO BIT 14.

RAL / RBL



ROTATE THE REGISTER LEFT 1 BIT.

RAR / RBR



ROTATE THE REGISTER RIGHT 1 BIT.

ROTATE EXAMPLE

PROBLEM:

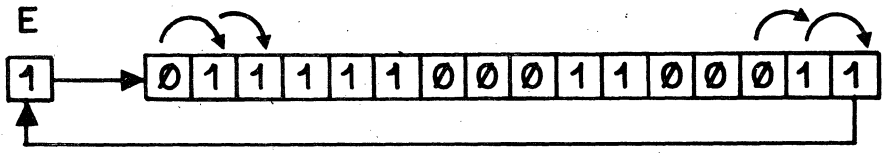
TEST BIT 15 OF THE "A" REGISTER.
IF BIT 15 = 1 , JMP TO LOCATION BUSY
IF BIT 15 = 0 , TEST BIT 14
IF BIT 14 = 1 , JMP TO LOCATION ERROR
IF BIT 14 = 0 , (PROGRAM CONTINUATION)

SOLUTION:

Label	Operation	Operand	Remarks
	SSA		BIT 15 = 1
	JMP	BUSY	YES
	RAL		NO
	SSA		BIT 14 = 1
	JMP	ERROR	YES
	PROGRAM	CONTINUATION	

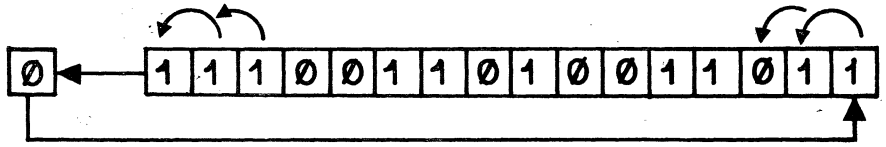
ROTATE INSTRUCTIONS

ERA / ERB



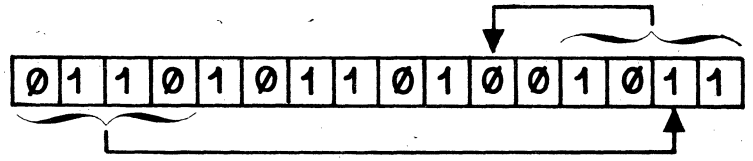
ROTATE THE REGISTER RIGHT, 1 BIT, WITH THE EXTEND (E) REGISTER. BIT Ø IS ROTATED INTO E. E IS ROTATED INTO BIT 15.

ELA/ELB

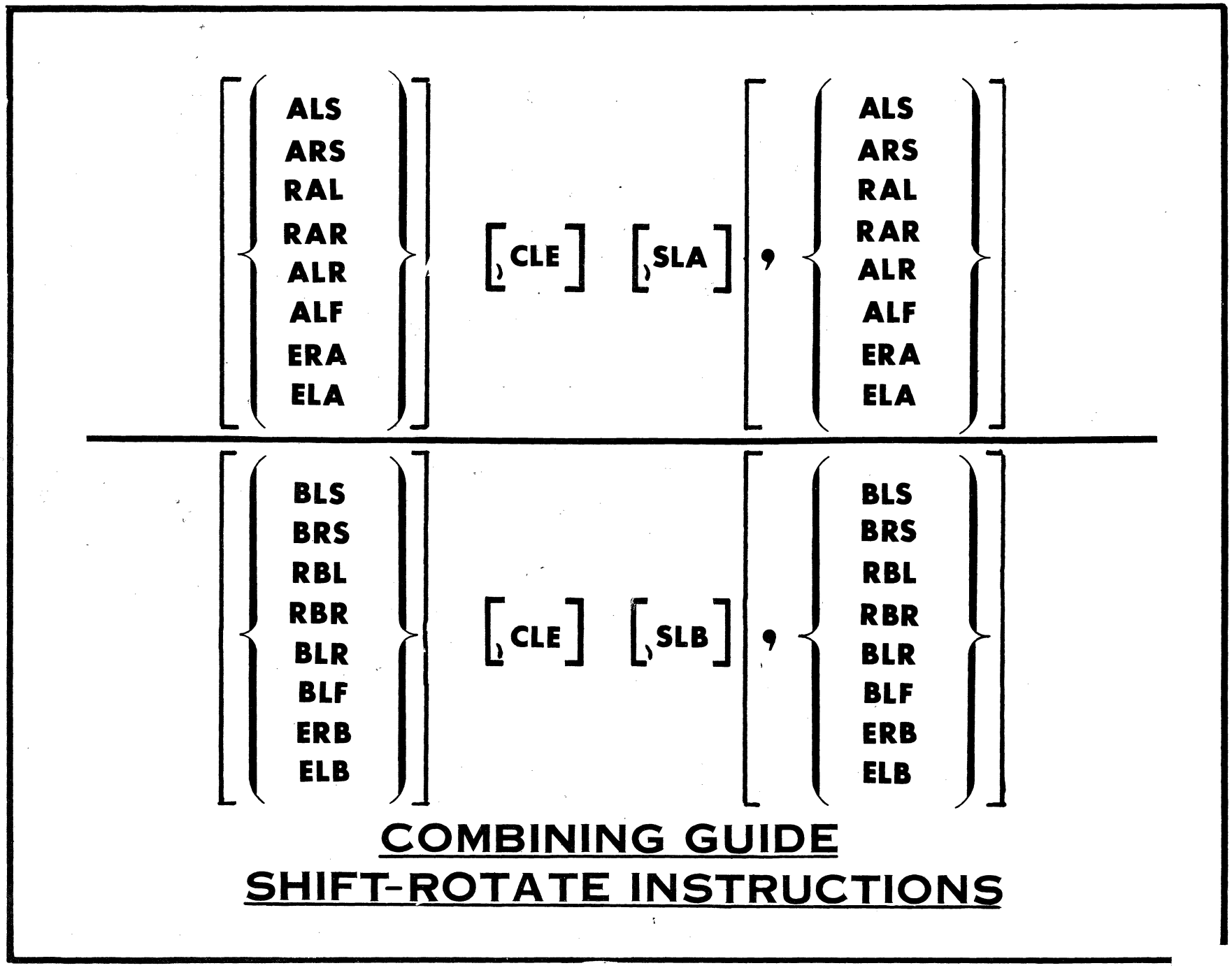


ROTATE THE REGISTER LEFT, 1 BIT, WITH THE EXTEND (E) REGISTER. BIT 15 IS ROTATED INTO E. E IS ROTATED INTO BIT Ø.

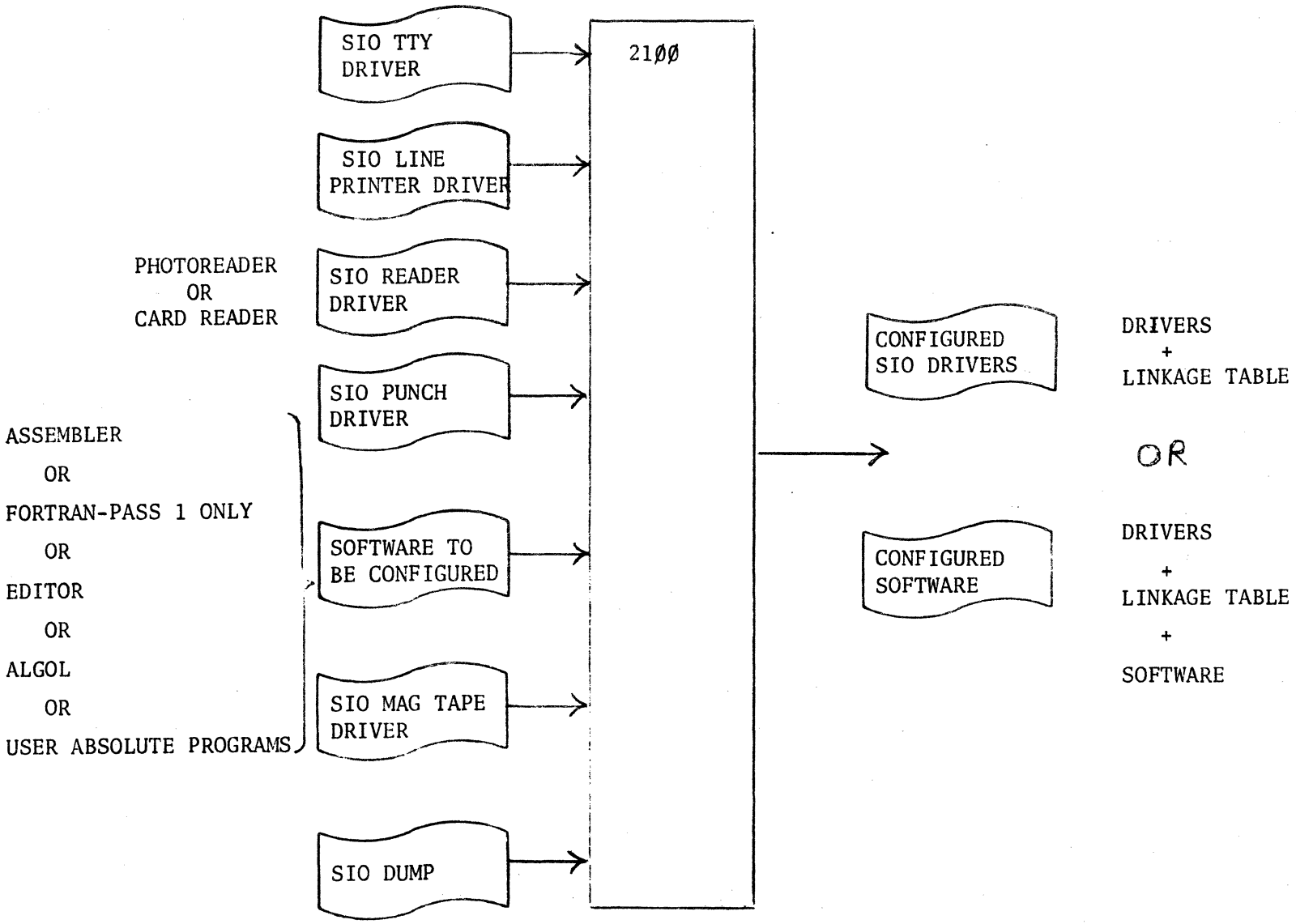
ALF / BLF



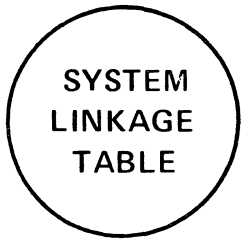
ROTATE THE REGISTER LEFT 4 PLACES. BITS 15, 14, 13, 12 ARE ROTATED INTO BITS 3, 2, 1, Ø, RESPECTIVELY.

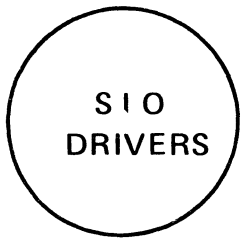

[{ ALS ARS RAL RAR ALR ALF ERA ELA }] [,CLE] [,SLA] [, { ALS ARS RAL RAR ALR ALF ERA ELA }]
[{ BLS BRS RBL RBR BLR BLF ERB ELB }] [,CLE] [,SLB] [, { BLS BRS RBL RBR BLR BLF ERB ELB }]
COMBINING GUIDE
SHIFT-ROTATE INSTRUCTIONS

SYSTEM INPUT-OUTPUT (SIO) CONFIGURATION OF SOFTWARE



SIO MEMORY MAP





4	POWER FAIL HALT
5	PARITY ERROR / MEMORY PROTECT HALT
100	STND SOFTWARE SYSTEM JMP INSTR.
101	INPUT DRIVER ADDRESS
102	LIST OUTPUT DRIVER ADDRESS
103	PUNCH OUTPUT DRIVER ADDRESS
104	KEYBOARD INPUT DRIVER ADDRESS
105	FWA OF AVAILABLE MEMORY
106	LWA OF AVAILABLE MEMORY
107	MAG TAPE DRIVER ADDRESS
	BASE PAGE AVAILABLE MEMORY
2000	PROGRAM AVAILABLE MEMORY
	MAG TAPE DRIVER
	TAPE-PUNCH DRIVER
	PHOTO-READER DRIVER
	TTY DRIVER, LINE PRINTER DRIVER
	ABSOLUTE BINARY LOADER
017777	

SOFTWARE CONFIGURATION PROCEDURE

1. USE THE BBL TO LOAD AN SIO DRIVER.

2. SET P TO 2, SET SWITCH REGISTER BITS Ø-5 TO THE ASSOCIATED SELECT CODE, HIT RUN. COMPUTER WILL HALT WITH 1Ø2Ø77 DISPLAYED. DRIVER IS NOW CONFIGURED.

3. REPEAT ABOVE STEPS FOR EACH SIO DRIVER (EXCEPT MAG TAPE) TO BE INCLUDED. THE SIO DRIVERS MUST BE LOADED IN THE FOLLOWING ORDER:
   TTY, LINE PRINTER, READER (PHOTO OR CARD), PUNCH.

4. USE THE BBL TO LOAD THE SOFTWARE TO BE CONFIGURED.

5. DO STEPS 1 & 2 FOR THE SIO MAG TAPE DRIVER. IN STEP 2 USE THE LOWER NUMBERED MAG TAPE SELECT CODE.

6. USE THE BBL TO LOAD SIO DUMP.

7. SET P TO 2.

8. SET BIT 15 OF THE SWITCH REGISTER TO

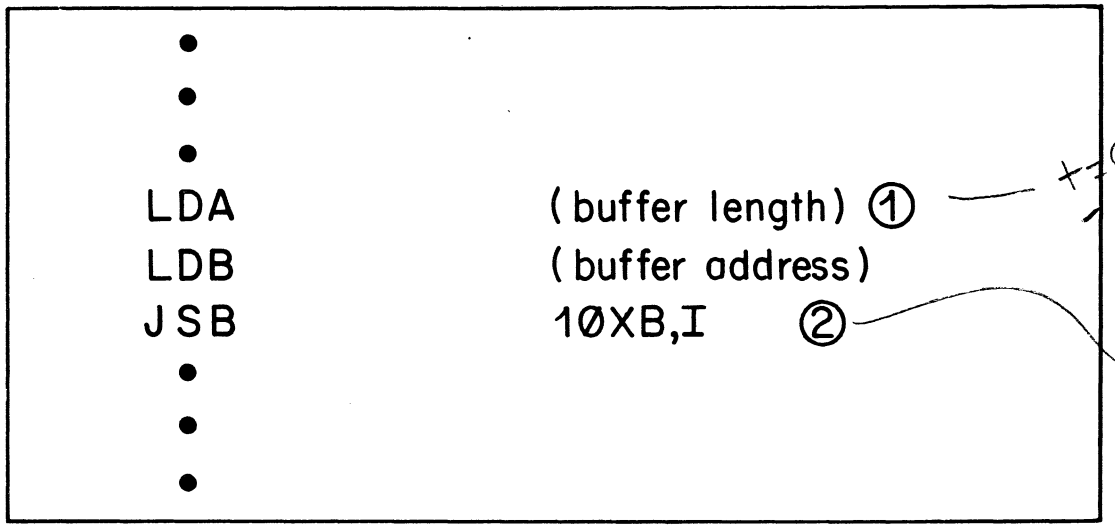
   Ø = OUTPUT LINKAGE TABLE + CONFIGURED SIO DRIVERS
   1 = OUTPUT LINKAGE TABLE + CONFIGURED SIO DRIVERS + SOFTWARE

   PRESS RUN. TAPE PUNCHED AS PER SWITCH REGISTER BIT 15.

9. FOR MULTIPLE COPIES, GO TO STEP 8.

SIO DRIVER USE FOR ABSOLUTE PROGRAMS

THE USER "CALLS" AN S.I.O. DRIVER FROM HIS MAIN PROGRAM AS FOLLOWS:



CONTROL IS RETURNED TO THE MAIN PROGRAM WHEN THE I/O OPERATION IS COMPLETED.

① Use positive numbers for characters, negative numbers for words.

② 10X= the systems linkage table address which contains the address of the proper driver

NOTE: REMEMBER THE DRIVERS MUST BE "CONFIGURED" TO THE PROPER I/O CHANNEL ASSOCIATED WITH THE DEVICE.

EXAMPLE - SIO DRIVER USE

WRITE A PROGRAM WHICH "READS" ASCII RECORDS FROM THE PHOTOREADER AND THEN TYPES THOSE SAME CHARACTERS ON THE TELEPRINTER.

```
ASMB,A,B,L
      ORG  2000B
BGIN  LDA  LN
      LDB  ADRES
      JSB  101B,I
* UPON RETURN, A-REG HAS TX LOG
      LDB  ADRES
      JSB  102B,I
      JMP  BGIN
LN    DEC  72
ADRES DEF  BFFR
BFFR  BSS  36
      END
```

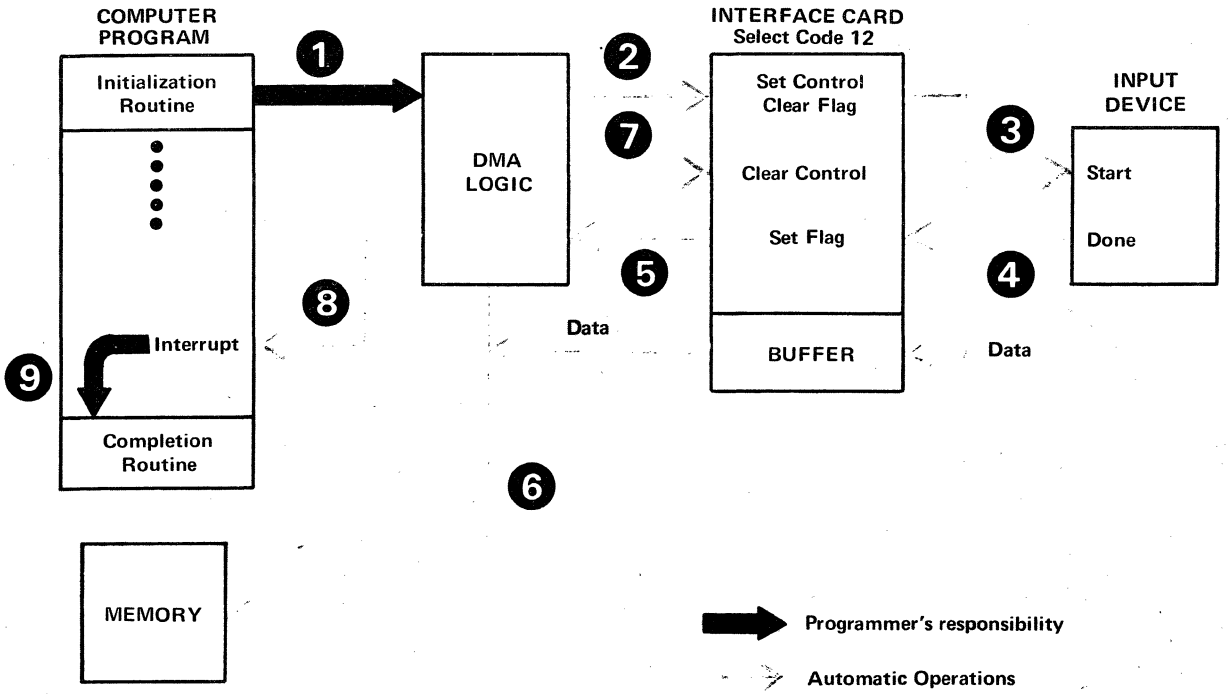
LOADING THE REQUIRED SIO DRIVERS PRIOR TO THIS PROGRAM ASSURES THAT THE LINKAGE WORDS WILL BE IN 101 & 102.

```
20  1037XX   STC  XX,C
21  1023XX   SFS  XX
22  024021   JMP  *-1
23  1025XX   LIA  XX
24  001727   ALF,ALF
25  1037XX   STC  XX,C
26  1023XX   SFS  XX
27  024026   JMP  *-1
30  1024XX   MIA  XX
31  170001   STA  1,I
32  006004   INB
33  024020   JMP  20B
```

SET B-REGISTER ACCORDING TO MEMORY SIZE.

XX = PAPER TAPE READER SELECT CODE

DMA TRANSFERS



DATA CONTROL WORD FORMATS

CONTROL WORD 1 (DEVICE CONTROL)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	STC		CLC	(NOT USED)							DEVICE SELECT CODE					
0	STC		CLC													

CONTROL WORD 2 (MEMORY CONTROL)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IN / OUT	MEMORY ADDRESS														

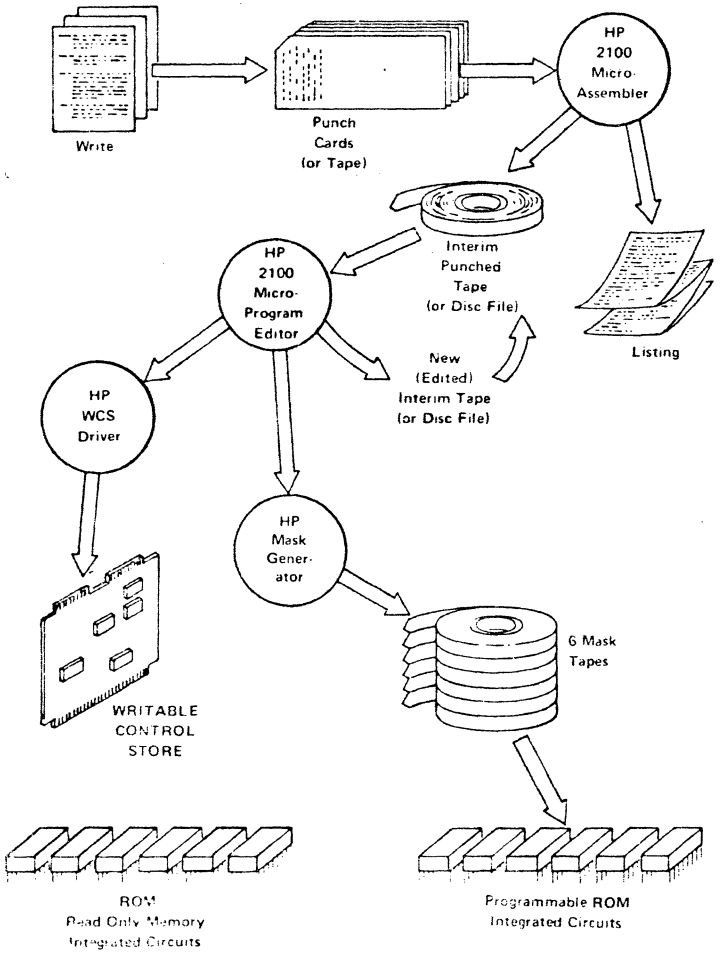
CONTROL WORD 3 (BLOCK LENGTH CONTROL)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WORD COUNT															

PROGRAM TO INITIALIZE DMA

LABEL	OPCODE	OPERAND	COMMENTS
ASGN1	LDA	CW1	Fetches control word 1 (CW1) from memory and loads it in A-register.
	OTA	6	Outputs CW1 to DMA Channel 1.
MAR1	CLC	2	Prepares Memory Address Register to receive control word 2 (CW2).
	LDA	CW2	Fetches CW2 from memory and loads it in A-register.
	OTA	2	Outputs CW2 to DMA Channel 1.
WCR1	STC	2	Prepares Word Count Register to receive control word 3 (CW3).
	LDA	CW3	Fetches CW3 from memory and loads it in A-register.
	OTA	2	Outputs CW3 to DMA Channel 1.
STRT1	STC	10B,C	Start input device.
	STC	6B,C	Activate DMA Channel 1.
	SFS JMP	6 *-1	Wait while data transfer takes place or, if interrupt processing is used, continue program.
. . .	. . .	. . .	
	HLT		Halt
CW1	OCT	120010	Assignment for DMA Channel 1 (ASGN1); specifies I/O Channel select code address (10_8), STC after each word is transferred, and CLC after final word is transferred.
CW2	OCT	100200	Memory Address Register control. DMA Channel 1 (MAR1); specifies memory input operation and starting memory address (200_8).
CW3	DEC	-50	Word Count Register control. DMA Channel 1 (WCR1); specifies the 2's complement of the number of words in the block of data to be transferred (50_{10}).

MICROPROGRAMMING IMPLEMENTATION



ASSEMBLER BLOCK MOVE PROGRAM (DOES NOT USE MICROPROGRAM)

```
0001                    ASMB,A,B,L
0002  04000                     ORG 4000B
0003  04000 000000  START NOP
0004  04001 102501  ENTRY LIA 1
0005  04002 003004        CMA,INA
0006  04003 072024        STA REPS
0007*
0008*
0009  04004 062025  AGAIN LDA NMBR
0010  04005 072026        STA COUNT
0011  04006 062027        LDA FRMDF
0012  04007 072030        STA FRPTR
0013  04010 062031        LDA TODF
0014  04011 072032        STA TOPTR
0015*
0016  04012 162030  NEXT  LDA FRPTR,I
0017  04013 172032        STA TOPTR,I
0018  04014 036030        ISZ FRPTR
0019  04015 036032        ISZ TOPTR
0020  04016 036026        ISZ COUNT
0021  04017 026012        JMP NEXT
0022  04020 036024        ISZ REPS
0023*
0024  04021 026004        JMP AGAIN
0025  04022 102077        HLT 77B
0026  04023 026001        JMP ENTRY
0027  04024 000000  REPS  NOP
0028  04025 177634  NMBR  DEC -100
0029  04026 000000  COUNT NOP
0030  04027 004033  FRMDF DEF FROM
0031  04030 000000  FRPTR NOP
0032  04031 004177  TODF  DEF TO
0033  04032 000000  TOPTR NOP
0034                    SUP
0035                    FROM  REP 5
0036  04033 040000        DEC 1.,2.,3.,4.,5.,6.,7.,8.,9.,10.
0036  04057 040000        DEC 1.,2.,3.,4.,5.,6.,7.,8.,9.,10.
0036  04103 040000        DEC 1.,2.,3.,4.,5.,6.,7.,8.,9.,10.
0036  04127 040000        DEC 1.,2.,3.,4.,5.,6.,7.,8.,9.,10.
0036  04153 040000        DEC 1.,2.,3.,4.,5.,6.,7.,8.,9.,10.
0037                    TO    REP 5
0038  04177 000000        DEC 0.,0.,0.,0.,0.,0.,0.,0.,0.,0.
0038  04223 000000        DEC 0.,0.,0.,0.,0.,0.,0.,0.,0.,0.
0038  04247 000000        DEC 0.,0.,0.,0.,0.,0.,0.,0.,0.,0.
0038  04273 000000        DEC 0.,0.,0.,0.,0.,0.,0.,0.,0.,0.
0038  04317 000000        DEC 0.,0.,0.,0.,0.,0.,0.,0.,0.,0.
0039                    END
**  NO ERRORS*
```

ASSEMBLER BLOCK MOVE PROGRAM (USES MICROPROGRAM)

```
0001                        ASMB,A,B,L
0002  02000                       ORG 2000B
0003  02000 000000  START NOP
0004  02001 102501  ENTRY LIA 1
0005  02002 003004        CMA,INA
0006  02003 072016        STA REPS
0007*
0008  02004 062017  AGAIN LDA NMBR
0009  02005 072011        STA COUNT
0010  02006 062020        LDA FRMDF
0011  02007 066021        LDB TODF
0012  02010 105200        OCT 105200
0013  02011 000000  COUNT NOP
0014*
0015  02012 036016        ISZ REPS
0016  02013 026004        JMP AGAIN
0017  02014 102077        HLT 77B
0018  02015 026001        JMP ENTRY
0019  02016 000000  REPS  NOP
0020  02017 177634  NMBR  DEC -100
0021  02020 002022  FRMDF DEF FROM
0022  02021 002166  TODF  DEF TO
0023                        SUP
0024                  FROM  REP 5
0025  02022 040000        DEC 1.,2.,3.,4.,5.,6.,7.,8.,9.,10.
0025  02046 040000        DEC 1.,2.,3.,4.,5.,6.,7.,8.,9.,10.
0025  02072 040000        DEC 1.,2.,3.,4.,5.,6.,7.,8.,9.,10.
0025  02116 040000        DEC 1.,2.,3.,4.,5.,6.,7.,8.,9.,10.
0025  02142 040000        DEC 1.,2.,3.,4.,5.,6.,7.,8.,9.,10.
0026                  TO    REP 5
0027  02166 000000        DEC 0.,0.,0.,0.,0.,0.,0.,0.,0.,0.
0027  02212 000000        DEC 0.,0.,0.,0.,0.,0.,0.,0.,0.,0.
0027  02236 000000        DEC 0.,0.,0.,0.,0.,0.,0.,0.,0.,0.
0027  02262 000000        DEC 0.,0.,0.,0.,0.,0.,0.,0.,0.,0.
0027  02306 000000        DEC 0.,0.,0.,0.,0.,0.,0.,0.,0.,0.
0028                        END
**  NO ERRORS*
```

BLOCK MOVE MICROPROGRAM

```
 1$ORIGIN=1010
 2$DEBUG
 3*
 4* INTERRUPTABLE BLOCK MOVE MICROPROGRAM
 5*
 6* CALLING SEQUENCE IS,
 7*
 8*         LDA -(NUMBER-OF-WORDS)
 9*         STA  COUNT
10*         LDA  FROM-ADDRESS
11*         LDB  TO-ADDRESS
12*         105200
13*  COUNT NOP
14*
15*
16  1010 375 170757   MOVE      P     IOR  M     RW
17  1011 345 175377             T     IOR  Q
18  1012 137 177407         Q         IOR        RSS  TBZ
19  1013 375 037427                   JMP        OUT
20  1014 375 117431   LOOP            CJMP       INRUP
21  1015 011 170757         A   RRS   IOR  M     RW
22  1016 345 173777             T     IOR  S1
23  1017 077 173377         B         IOR  S2
24  1020 164 060712         F   S2    DEC  M     CW   NMPV
25  1021 375 037427                   JMP        OUT
26  1022 367 171377             S1    IOR  T
27  1023 036 117377         A         INC  A
28  1024 076 116777         B         INC  B
29  1025 136 115367         Q         INC  Q          TBZ
30  1026 375 037414                   JMP        LOOP
31  1027 374 114375   OUT       P     INC  P          EOP
32  1030 377 177777                   IOR
33*
34* UPON INTERRUPT JUMP HERE AND SAVE THE COUNT
35*   NOTE: IT IS THE RESPONSIBILITY OF THE INTERRUPTING PROGRAM
36*         TO SAVE THE A & B REGISTERS
37*
38  1031 375 170716   INRUP     P     IOR  M     CW   UNC
39  1032 377 177777                   IOR
40  1033 111 171377         Q   RRS   IOR  T
41  1034 374 054375             P     SUB  P          EOP
42  1035 374 154377             P     NOR  P
43$END
**NO ERRORS**
```

DAY 4 READING ASSIGNMENT

POCKET GUIDE to the 2100 COMPUTER

2100A Reference

Chapter 1, table 1.1, 1.2

Assembler

Appendix B

BCS

Chapter 4

Chapter 6

LAB 4.1 CONFIGURE ASSEMBLER

USING THE PROCEDURE OUTLINED IN SLIDE 4-30, CONFIGURE AN ASSEMBLER WITH TTY, PHOTOREADER, & PUNCH SIO DRIVERS.

AN UNCONFIGURED ASSEMBLER WILL BE PROVIDED.

TEST YOUR ASSEMBLER BY ASSEMBLING A PREVIOUS LAB SOURCE TAPE.

LAB 4.2 TAPE DUPLICATOR (SFS)

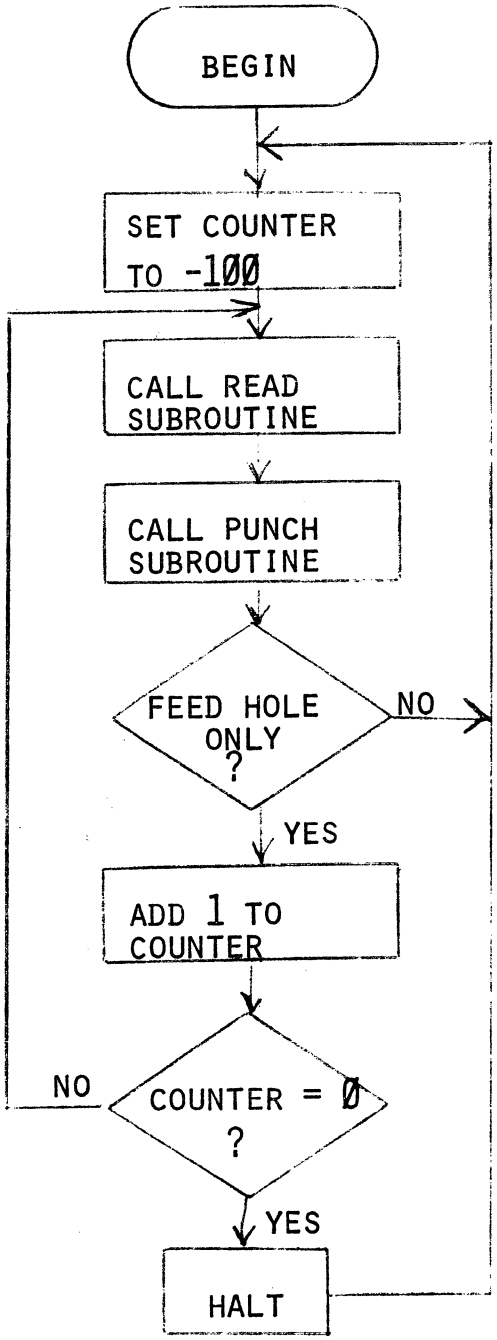
WRITE AN ABSOLUTE PROGRAM THAT WILL READ ANY TAPE ON THE PHOTOREADER AND REPRODUCE THE TAPE ON THE PUNCH.

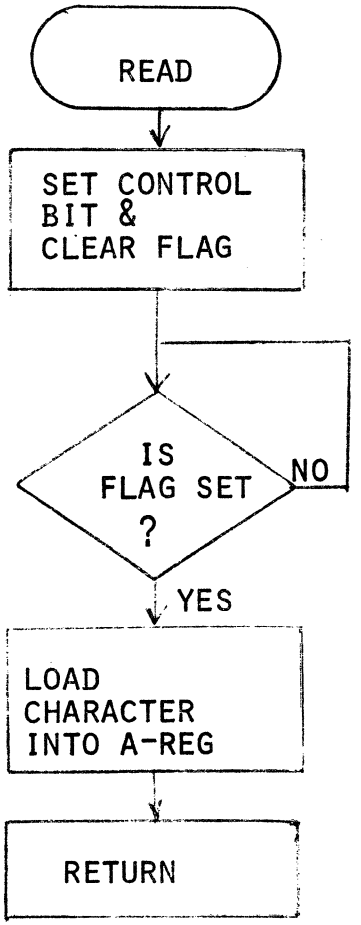
DESIGN THE PROGRAM SO THAT IT WILL HALT IF 100 CONSECUTIVE EMPTY (FEED HOLE ONLY) FRAMES ARE READ.

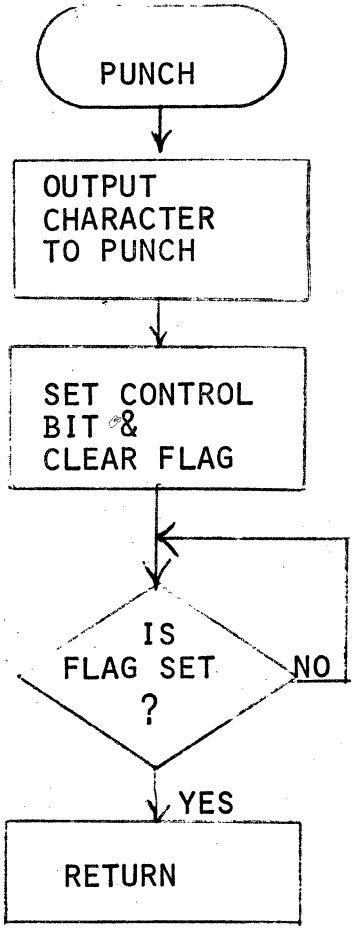
WRITE THE I/O INSTRUCTIONS FOR READING & PUNCHING AS SUBROUTINES.

TEST YOUR PROGRAM BY REPRODUCING YOUR ABSOLUTE PROGRAM TAPE.

LAB 4.2 FLOWCHARTS







LAB 4.3 SOURCE TAPE DUPLICATOR (SIO)

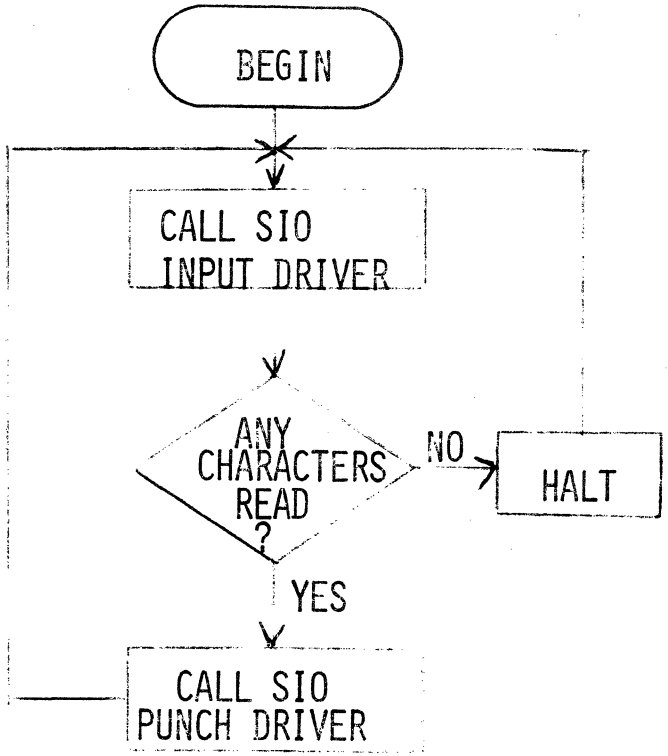
WRITE AN ABSOLUTE PROGRAM THAT WILL READ A SOURCE TAPE ON THE PHOTOREADER AND REPRODUCE THE TAPE ON THE PUNCH.

USE SIO DRIVERS.

TEST YOUR PROGRAM BY REPRODUCING YOUR SOURCE PROGRAM TAPE.

BE SURE TO CONFIGURE YOUR ABSOLUTE PROGRAM TAPE BEFORE ATTEMPTING PROGRAM EXECUTION.

LAB 4.3 FLOWCHART



LAB 4.2 SOLUTION

```
0001                    ASMB,A,B,L
0002*
0003* LAB 4.2 TAPE DUPLICATOR (SFS)
0004*
0005  02000                    ORG 2000B
0006  02000 062026  BEGIN LDA DM100     SET COUNTER TO -100
0007  02001 072027         STA CNTR
0008*
0009  02002 016012  NEXT   JSB READ     CALL READ SUBROUTINE
0010*
0011  02003 016020         JSB PUNCH    CALL PUNCH SUBROUTINE
0012*
0013  02004 002002         SZA          FEED HOLE ONLY ?
0014  02005 026000         JMP BEGIN      NO, RESET COUNTER
0015*
0016  02006 036027         ISZ CNTR       YES, ADD 1 TO COUNTER
0017*                                   COUNTER = 0 ?
0018  02007 026002         JMP NEXT       NO, REPRODUCE NEXT CHARACTER
0019*
0020  02010 102077         HLT 77B        YES, HALT
0021  02011 026000         JMP BEGIN
0022*
0023*
0024*
0025  02012 000000  READ   NOP          READ SUBROUTINE
0026*
0027  02013 103713         STC RDR,C    SET CONTROL BIT & CLEAR FLAG
0028*
0029  02014 102313         SFS RDR      IS FLAG SET ?
0030*
0031  02015 026014         JMP *-1        NO, TEST FLAG AGAIN
0032*
0033  02016 102513         LIA RDR        YES, LOAD CHARACTER INTO A-REG
0034*
0035  02017 126012         JMP READ,I   RETURN
0036  00013           RDR  EQU 13B      PHOTOREADER SELECT CODE
0037*
0038*
0039*
0040  02020 000000  PUNCH NOP          PUNCH SUBROUTINE
0041*
0042  02021 102616         OTA PNCH     OUTPUT CHARACTER TO PUNCH
0043*
0044  02022 103716         STC PNCH,C   SET CONTROL BIT & CLEAR FLAG
0045*
0046  02023 102316         SFS PNCH     IS FLAG SET ?
0047*
0048  02024 026023         JMP *-1        NO, TEST FLAG AGAIN
0049*
0050  02025 126020         JMP PUNCH,I    YES, RETURN
0051  00016           PNCH  EQU 16B     PUNCH SELECT CODE
0052*
0053  02026 177634  DM100 DEC -100
0054  02027 000000  CNTR BSS 1
0055                       END
**  NO ERRORS*
```

LAB 4.3 SOLUTION

```
0001                    ASMB,A,B,L
0002*
0003* LAB 4.3 SOURCE TAPE DUPLICATOR (SIO)
0004*
0005  02000                 ORG 2000B
0006*
0007* CALL SIO INPUT DRIVER
0008  02000 066012  BEGIN LDA MAXLN    MAX # OF CHARACTERS
0009  02001 066013        LDB BUFAD    POINT TO BUFFER
0010  02002 114101        JSB 101B,I
0011*
0012* ANY CHARACTERS READ ?
0013  02003 002002        SZA
0014  02004 026007        JMP PUNCH      YES, PUNCH THEM
0015  02005 102077        HLT 77B        NO, HALT
0016  02006 026000        JMP BEGIN
0017*
0018* CALL SIO PUNCH DRIVER
0019  02007 066013  PUNCH LDB BUFAD    POINT TO BUFFER
0020  02010 114103        JSB 103B,I
0021  02011 026000        JMP BEGIN
0022*
0023  02012 000110  MAXLN DEC 72
0024  02013 002014  BUFAD DEF BUFFR
0025  02014 000000  BUFFR BSS 36
0026                      END
**  NO ERRORS*
```

THE COM PSEUDO INSTRUCTION

for relocatable programs only

- THE COM PSEUDO RESERVES A BLOCK OF STORAGE LOCATIONS THAT MAY BE USED IN COMMON BY SEVERAL SUBPROGRAMS.

GENERAL FORM:

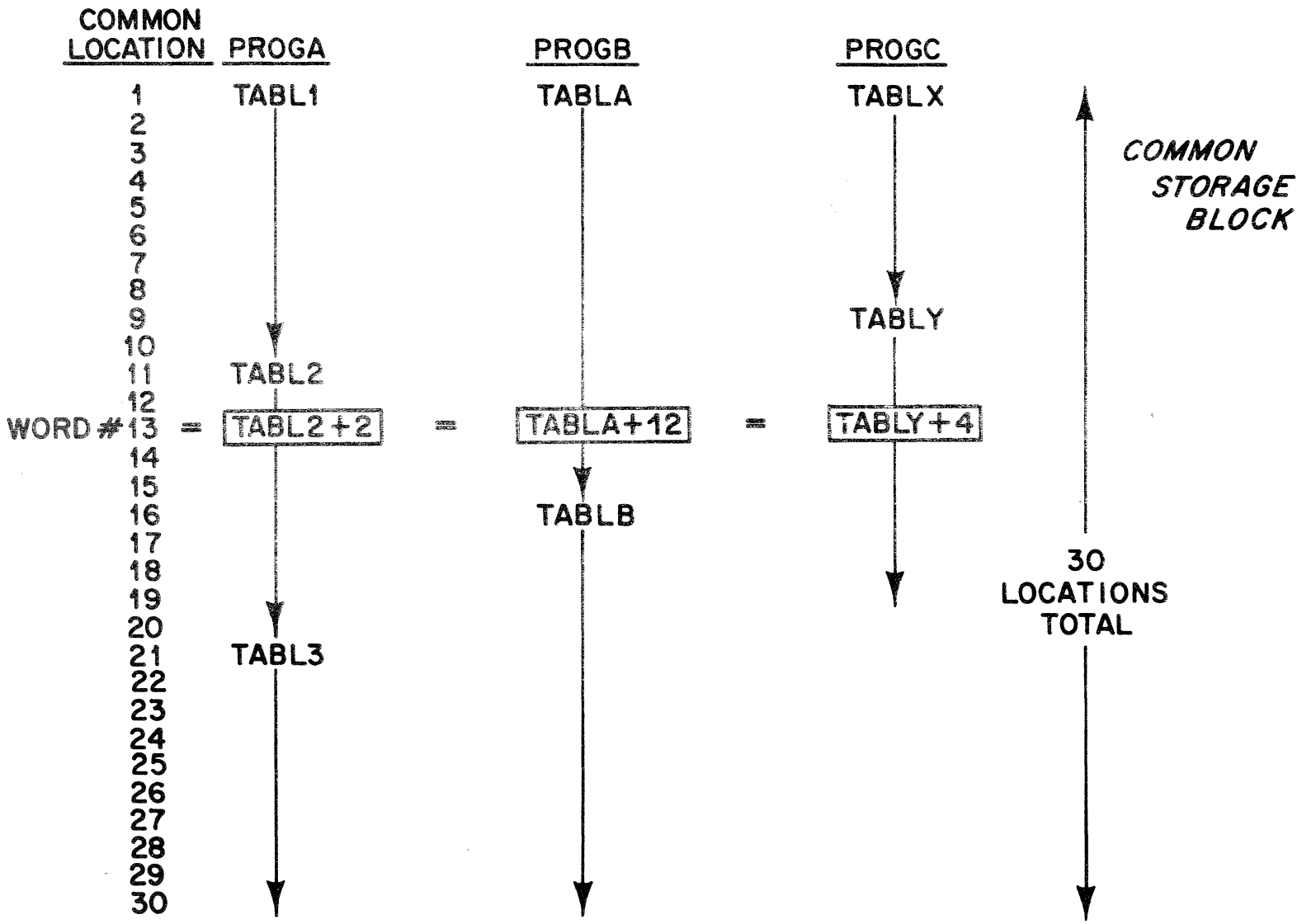
COM	$name_1$ ($size_1$), $name_2$ ($size_2$), . . . , $name_n$ ($size_n$)	REMARKS

- EACH NAME IDENTIFIES A SEGMENT OF THE BLOCK FOR THE SUBPROGRAM IN WHICH THE COM STATEMENT APPEARS.
- STORAGE LOCATIONS ARE ASSIGNED CONTIGUOUSLY
- THE LENGTH OF THE BLOCK IS EQUAL TO THE SUM OF THE LENGTHS OF ALL SEGMENTS NAMED IN ALL COM STATEMENTS IN THE SUBPROGRAM.
- TO REFER TO THE COMMON BLOCK OTHER SUBPROGRAMS MUST ALSO INCLUDE A COM STATEMENT.
- AT LOAD TIME; THE SUBPROGRAM WITH THE GREATEST COMMON DECLARATION MUST BE LOADED FIRST.

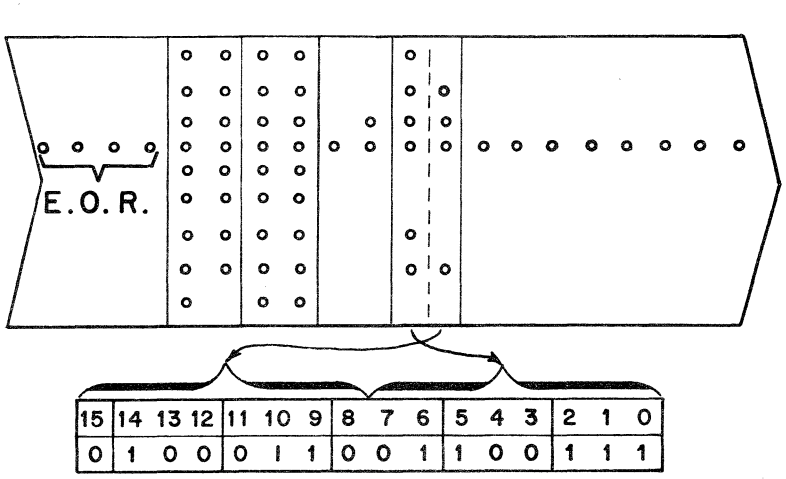
USING THE COM PSEUDO

	LABEL	OP CODE	OPERAND
SEGMENT "A"		NAM	PROGA
	START	NOP	
		LDA	TABL1
		COM	TABL1 (10) ,TABL2 (10), TABL3 (10)
		END	START
SEGMENT "B"		NAM	PROGB
		COM	TABLA (15), TABLB (15)
		END	
SEGMENT "C"		NAM	PROGC
		STA	TABLX
		COM	TABLX (8), TABLY (11)
		END	

COMMON STORAGE

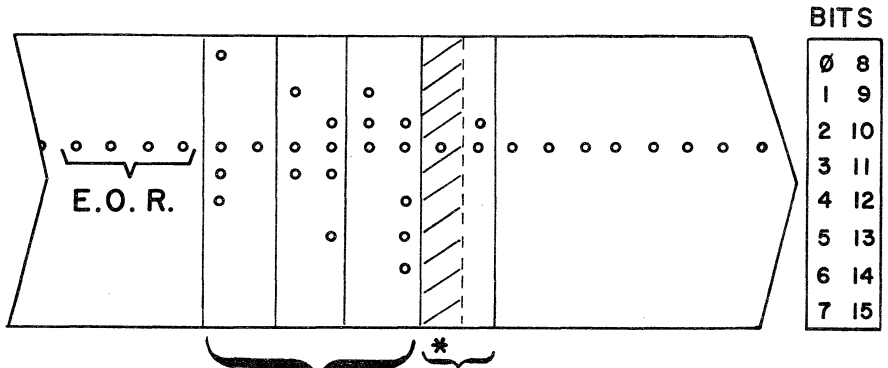


PAPER TAPE (BINARY) DATA



16 BIT BINARY WORD FORMAT

A FIXED LENGTH BINARY RECORD (SHOWN WITH 4 WORDS)



A VARIABLE LENGTH BINARY RECORD (SHOWN WITH 4 WORDS)

1. **Feed frames prior to the first character are ignored by the driver. Therefore the 1st frame of a binary record must be non-zero.**

2. **Four feed frames separate binary records.**

3. **To output a record in variable binary form, the number of words in the data block must be in the range $1 \leq n \leq 255_{10}$. The second frame of the word count * is ignored on input.**

4. **Variable binary input uses the word count, or the specified buffer size to terminate transmission, whichever value is smaller.**

** DEBUG FACILITIES **

ALLOW RELOCATABLE MEMORY REFERENCE

MODIFY MEMORY CONTENTS

MODIFY REGISTER CONTENTS

DUMP MEMORY

BREAKPOINT

TRACE

START EXECUTION AT ANY POINT IN PROGRAM

RESTART

** DEBUG CONTROL STATEMENTS **

M,A	PROGRAM RELOCATION BASE
S,A,V1,V2,...,VN	SET MEMORY
W,R,V	SET REGISTER
D,A,A1,A2	DUMP MEMORY
D,B,A1,A2	
B,I,A	BREAKPOINT
B,O,A	
B,I,Ø	
B,O,Ø	
T,A1[,A2]	TRACE
T,Ø	
R[,A]	START EXECUTION AT ANY POINT IN PROGRAM
A	RESTART

DUMP FORMATS

	ADDR.	WORD1	WORD2	...	WORD8
OCTAL:	AAAAA	000000	000000	...	000000
ASCII:	AAAAA	CC	CC	...	CC

EXAMPLES:

1. DUMP OF RELOCATABLE LOCATIONS 37B-53B, IN OCTAL.

```
D,B,37,53
DUMP--BASE = 02000

 00037                                                          040502

 00040  041504  042506  043510  044512  045514  046516  047520  050522

 00050  051524  052526  053530  054532
```

2. DUMP OF RELOCATABLE LOCATIONS 37B-53B, IN ASCII.

```
D,A,37,53
DUMP--BASE = 02000

 00037                                                  AB

 00040  CD    EF    GH    IJ    KL    MN    OP    QR

 00050  ST    UV    WX    YZ
```

ERROR MESSAGES

ENTRY ERROR

INDIRECT LOOP —— Count indirect refs. to 16 times

PROGRAM TO BE ANALYZED USING DEBUG FACILITY

```
0001                        ASMB,R,B,L
0002  00000                       NAM DEMO DEBUG
0003  00000 000000  START     NOP
0004  00001 062021R AGAIN     LDA MSK
0005  00002 072023R           STA CNTR
0006  00003 062022R           LDA FROM
0007  00004 072024R           STA SRCE
0008  00005 062040R           LDA TO
0009  00006 072037R           STA TRGT
0010  00007 162024R NEXT      LDA SRCE,I
0011  00010 172037R           STA TRGT,I
0012  00011 036024R           ISZ SRCE
0013  00012 036037R           ISZ TRGT
0014  00013 036023R           ISZ CNTR
0015  00014 026007R           JMP NEXT
0016  00015 102501            LIA 1
0017  00016 002011            SLA,RSS
0018  00017 102077            HLT 77B
0019  00020 026001R           JMP AGAIN
0020*
0021  00021 177773  MSK       DEC -5
0022  00022 000025R FROM      DEF BUFI
0023  00023 000000  CNTR      NOP
0024  00024 000000  SRCE      NOP
0025  00025 040502  BUFI      ASC 5,ABCDEFGHIJ
      00026 041504
      00027 042506
      00030 043510
      00031 044512
0026                        SUP
0027  00032 000000  BUFO      OCT 0,0,0,0,0
0028  00037 000000  TRGT      NOP
0029  00040 000025R TO        DEF BUFI
0030                        END START
**  NO ERRORS*
```

You can't trace thru .IOC. because DEBUG uses it.

```
  DEMO

 02000 02040

*LOAD

  DEBUG

 02041 04012

 00077 00077

*LOAD

*LST
•IOC.   16026
•SQT.   16003
•MEM.   15775
•BUFR   16177
HALT    15770
DEBRS   03356
DEBUG   02041

 *LINKS
 01764 01777

*RUN
```

DEBUG CONSOLE RUN SHEET (cont.)

```
M,2000
R,1
H P= 00017  I=102077  A=000004  B=000000  E=0 O=0

D,B,32,36
DUMP--BASE = 02000

 00032                  000000  000000  000000  000000  000000

T,7,10
R
T P= 00007  I=162024  A=002025  B=000000  E=0 O=0  MA= 00025 MC=040502

T P= 00010  I=172037  A=040502  B=000000  E=0 O=0  MA= 00025 MC=040502

T P= 00007  I=162024  A=040502  B=000000  E=0 O=0  MA= 00026 MC=041504

T P= 00010  I=172037  A=041504  B=000000  E=0 O=0  MA= 00026 MC=041504

T P= 00007  I=162024  A=041504  B=000000  E=0 O=0  MA= 00027 MC=042506

T P= 00010  I=172037  A=042506  B=000000  E=0 O=0  MA= 00027 MC=042506

T P= 00007  I=162024  A=042506  B=000000  E=0 O=0  MA= 00030 MC=043510

T P= 00010  I=172037  A=043510  B=000000  E=0 O=0  MA= 00030 MC=043510

T P= 00007  I=162024  A=043510  B=000000  E=0 O=0  MA= 00031 MC=044512

T P= 00010  I=172037  A=044512  B=000000  E=0 O=0  MA= 00031 MC=044512

H P= 00017  I=102077  A=000004  B=000000  E=0 O=0

T,0
D,B,40
DUMP--BASE = 02000

 00040  002025

S,40,2032
R
H P= 00017  I=102077  A=000004  B=000000  E=0 O=0

D,B,32,36
DUMP--BASE = 02000

 00032                  040502  041504  042506  043510  044512

D,A,32,36
DUMP--BASE = 02000

 00032                   AB      CD      EF      GH      IJ
```

CHARACTER EDITING

/CD,L,C1[,C2] DELETION

/CR,L,C1[,C2] REPLACEMENT

/CI,L,C INSERTION

CHARACTER EDITING EXAMPLE

```
                    HP SYMBOLIC EDITOR

EDIT FILE DEVICE?
/T

*
/L
/E

SYMBOLIC FILE SOURCE DEVICE?
/P

0001  ASMB,BINARY,R,L
0002         NAM LNEDT
0003  START N BEGIN PROG
0004         LXA I
0005         CMA,INA
0006         HLT
0007         END START


**END-OF-TAPE
*
/E

*END
```

CHARACTER EDITING EXAMPLE (CONT.)

EDITS

```
                                HP SYMBOLIC EDITOR


EDIT FILE DEVICE?
/T

*
/CD,1,7,11
/CR,2,11,12
CH
/CI,3,7
OP
/R,4
      LIA 1
/E

SYMBOLIC FILE SOURCE DEVICE?
/P

SYMBOLIC FILE DESTINATION DEVICE?
/P


**END-OF-TAPE
*
/E

*END
```

LISTING OF RESULTS

```
                                HP SYMBOLIC EDITOR


EDIT FILE DEVICE?
/T

*
/L
/E

SYMBOLIC FILE SOURCE DEVICE?
/P

0001  ASMB,B,R,L
0002        NAM CHEDT
0003  START NOP BEGIN PROG
0004        LIA 1
0005        CMA,INA
0006        HLT
0007        END START


**END-OF-TAPE
*
/E

*END
```

LISTING OF TAPES TO BE EDITED

EDITS

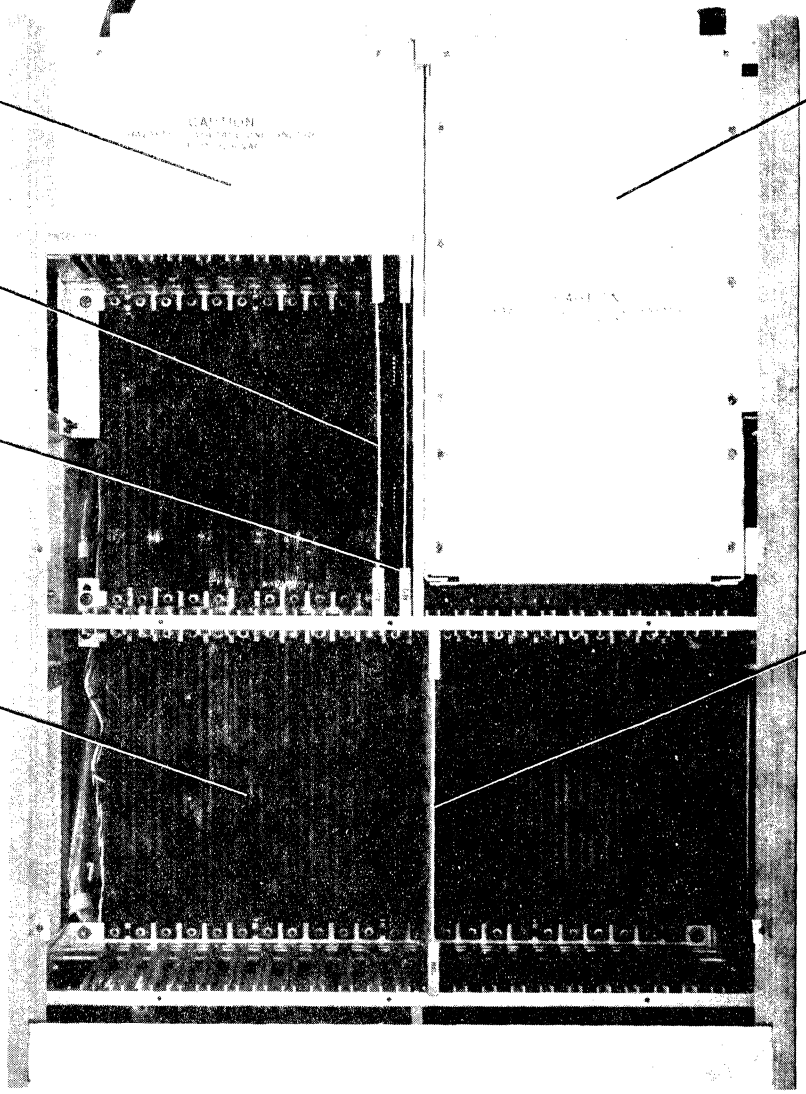
RESULTS

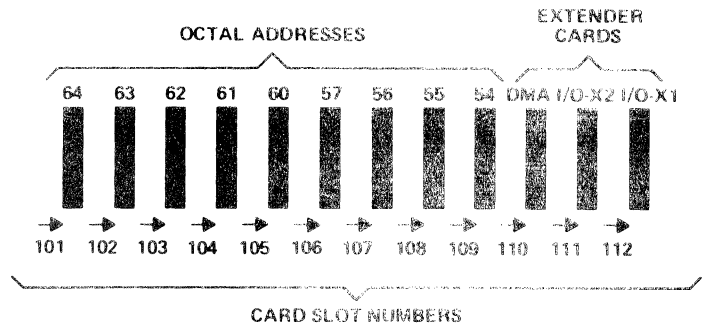
```
                                        HP SYMBOLIC EDITOR

EDIT FILE DEVICE?
/T

*
/L
/E

SYMBOLIC FILE SOURCE DEVICE?
/P

0001  ASMB,R,B,L
0002          NAME SPLIT


**END-OF-TAPE
*
/C
0003  START NOP
0004          HALT
0005          END START


**END-OF-TAPE
*
/E

*END
```

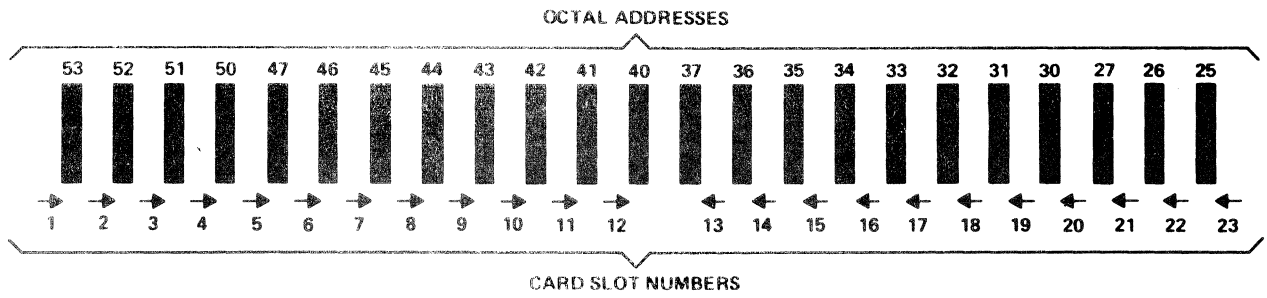
```
                                        HP SYMBOLIC EDITOR

EDIT FILE DEVICE?
/T

*
/CD,3,10
/↑
/CD,3,10
CONTROL STATEMENT DELETED!
*
/CD,2,10
/CD,4,8
/E

SYMBOLIC FILE SOURCE DEVICE?
/P

SYMBOLIC FILE DESTINATION DEVICE?
/P


**END-OF-TAPE
*
/C


**END-OF-TAPE
*
/E

*END
```

```
                                        HP SYMBOLIC EDITOR

EDIT FILE DEVICE?
/T

*
/L
/E

SYMBOLIC FILE SOURCE DEVICE?
/P

0001  ASMB,R,B,L
0002          NAM SPLIT
0003  START NOP
0004          HLT
0005          END START


**END-OF-TAPE
*
/E

*END
```

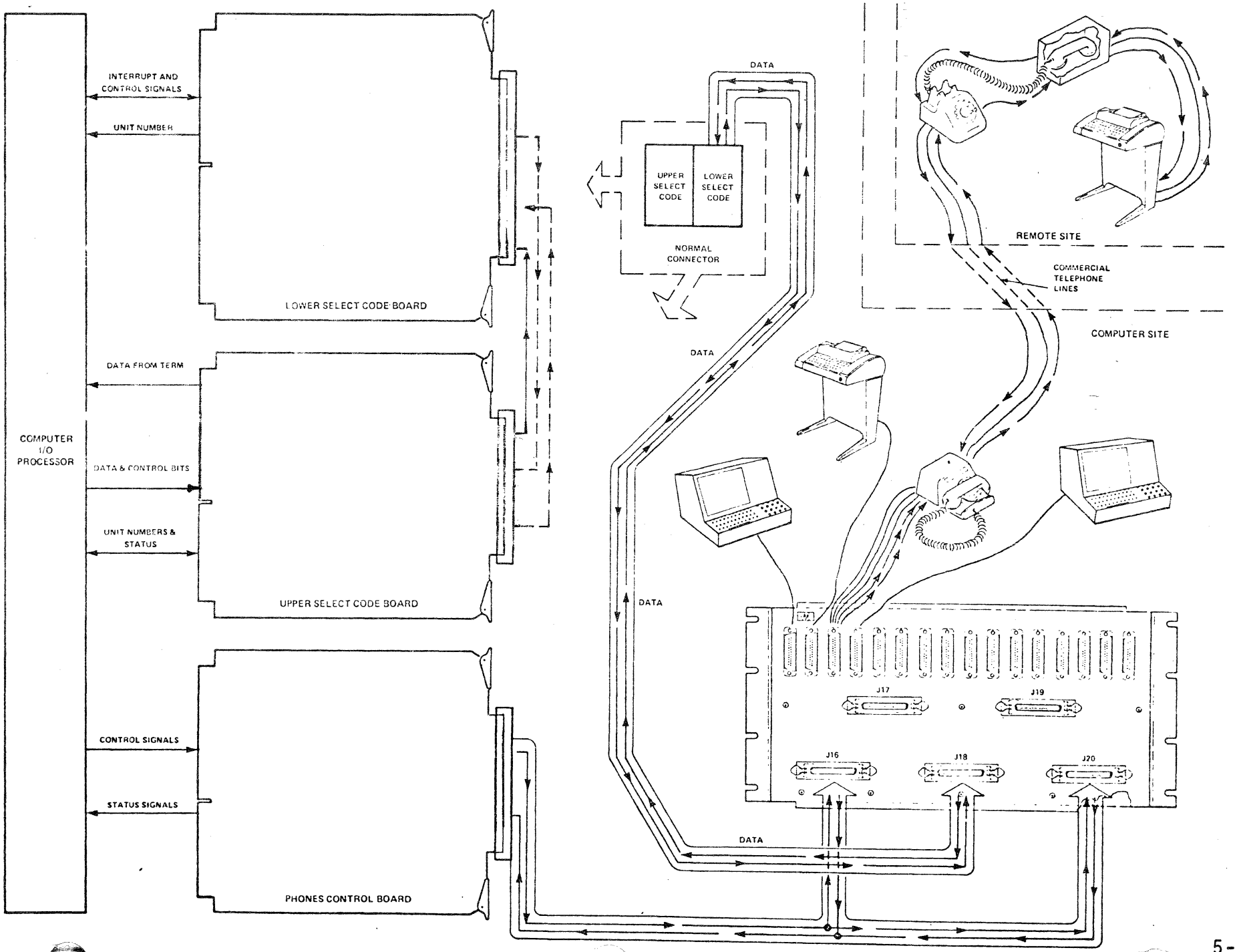
I/O EXTENDER FOR THE 2100







MULTIPLEXER DATA & CONTROL FLOW



GENERAL SPECIFICATIONS

BASIC CHARACTERISTICS

- 16-bit word length; 17th bit for memory parity checking
- Parallel logic
- Power fail interrupt, with automatic restart
- Rack-mountable

MEMORY

- Magnetic core storage
- 980 nanosecond cycle time
- Parity generation and checking is standard in all units
- Six memory sizes available, 4,096 to 32,768 words; field-expandable by plug-in cards
- 1024-word page size
- Protected 64-word block for stored loader

PROCESSOR

- 80 basic instructions, including extended arithmetic
- Up to eight instructions may be combined into one word (register reference group)
- Two accumulators, addressable as memory locations
- Unlimited levels of indirect addressing allowed
- Six working registers, may be selected for display and instant modification (A, B, T, P, M, S)
- Illuminated control pushbuttons allow simultaneous display and control of internal functions
- All instructions fully executed in 1.96 microseconds, except ISZ and extended arithmetic (2.94 to 16.7 microseconds)
- Only 980 nanoseconds added for each level of indirect addressing
- Memory and I/O protection is standard

SOFTWARE

- FORTRAN, FORTRAN IV, ALGOL, and BASIC languages
- Extended Assembly language
- Editor, subroutine library, Formatter, and Debug routine
- Several operating systems, including
  - Basic Control System
  - Magnetic Tape System
  - Disc Operating System
  - Time-Shared BASIC System

INPUT/OUTPUT SYSTEM

- 14 internal I/O channels, externally expandable to 45
- Optional multiplexed I/O extends capacity to 56 channels; may be plugged into any slot
- All channels buffered and bi-directional
- Multilevel priority interrupt for device servicing
- Peripherals interfaced simply with plug-in cards
- Optional dual-channel direct memory access, can transfer 1,020,400 words per second
- General-purpose interface cards available

PERIPHERALS

- Magnetic Tape
  Read and write 9-track IBM-compatible magnetic tape, 800 BPI, at speeds of 25, 37.5, or 45 inches per second; also read and write 7-track IBM-compatible magnetic tape at speeds of 25, 37.5, or 45 inches per second with switch-selectable densities of 200, 556, and 800 BPI
- Disc/Drum Memory
  Fixed head-per-track design for rapid access, capacities range from 262,144 to 1,048,576 words
- Cartridge Disc
  Moving-head disc for low-cost mass storage; capacities from 1.2 million to 4.8 million words
- Disc File
  Moving-head mass storage; 11.7 million words per drive, 8 drives maximum
- Card Reader
  Reads punched 80-column cards, 12 bits in parallel, at 1000 cards per minute
- Mark Reader
  Reads punched and pencil-marked cards at 200 cards per minute
- Line Printers
  Print 120 or 132 columns per line at 300 lines per minute; ASCII 64-character set. Also from 356 lines per minute (80 columns) to 1110 lines per minute (20 columns); 64-character set
- Keyboard Display Terminal
  CRT screen displays 25 lines, 72 characters per line; standard teleprinter keyboard plus 10-key numerical keyboard; speeds of 10 to 200 characters per second, switch selectable
- Tape Readers
  Read 5- and 8-level punched paper tape at up to 500 characters per second; with or without automatic reroller
- Tape Punch
  Punch 5- and 8-level code at 120 characters per second; also 5- and 8-level code at 75 characters per second
- Teleprinter
  Keyboard with punch, read, and print capabilities; 10 characters per second; ASCII character set
- Data Communications
  Asynchronous data set interface, 75 to 2400 bits per second, full- and half-duplex; also synchronous receive and transmit data set interfaces, 1200 to 9600 bits per second; automatic dialing capability available

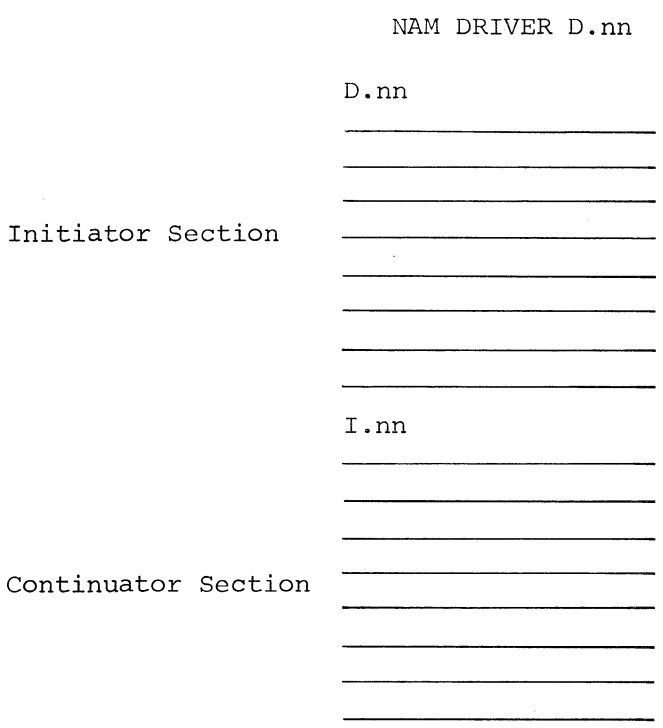
SAMPLE POWER FAIL SUBROUTINE

LABEL	OPCODE	OPERAND	COMMENTS
PFAR	NOP		Power Fail/Auto Restart Subroutine
	SFC	4B	Skip if interrupt was caused by a power failure
	JMP	UP	Power is being restored, reset state of computer system
DOWN	STA	SAVA	Save A-register contents
	CCA		Set switch indicating that the computer was running when
	STA	SAVR	power failed
	STB	SAVB	Save B-register contents
	ERA,ALS		Transfer E-register content to A-register bit 15
	SOC		Increment A-register if Overflow
	INA		is set
	STA	SAVEO	Save E- and O-register contents
	LDA	PFAR	Save contents of P-register at time of
	STA	SAVP	power failure
	LIA	1B	Save contents of
	STA	SAVS	S-register
	⋮		Insert user-written routine to save I/O device states
	CLC	4B	Turn on restart logic so computer will restart when power is restored after momentary power failure
	HLT		Shutdown
UP	LDA	SAVR	Was computer running when
	SZA,RSS		power failed?
	JMP	HALT	No
	CLA		Yes, reset computer Run switch to
	STA	SAVR	initial state
	LDA	FENCE	Restore the memory protect
	OTA	5B	fence register contents
	⋮		Insert user-written routine to restore I/O device states
	LDA	SAVEO	Restore the contents
	CLO		of the
	SLA,ELA		E-register and
	STF	1B	O-register
	LDA	SAVS	Restore the contents of the
	OTA	1B	S-register
	LDA	SAVA	Restore A-register contents
	LDB	SAVB	Restore B-register contents
	STC	4B	Reset power fail logic for next power failure
	STC	5B	Turn on memory protect
	JMP	SAVP,I	Transfer control to program in execution at time of power failure
HALT	HLT		Return computer to halt mode
FENCE	OCT	2000B	Fence address storage (must be updated each time fence is changed)
SAVEO	OCT	0	Storage for E and O
SAVA	OCT	0	Storage for A
SAVB	OCT	0	Storage for B
SAVS	OCT	0	Storage for S
SAVP	OCT	0	Storage for P
SAVR	OCT	0	Storage for Run switch

SAMPLE MEMORY PROTECT/PARITY ERROR SUBROUTINE

LABEL	OPCODE	OPERAND	COMMENTS
MPPE	NOP		Memory Protect/Parity Error Subroutine
	CLF	0	Turn off interrupt system to inhibit I/O devices
	CLF	5	Turn off P.E. interrupt during subroutine
	STA	SVA	Save A-register contents
	STB	SVB	Save B-register contents
	LIA	5	Get contents of violation register in MP logic
	SSA		Check bit 15 to determine kind of error
	JMP	PERR	If a 1, go to parity error routine
	JMP	MPTR	If a 0, go to memory protect routine
MPTR	—		User's routine in case of memory protect violation
	—		
	—		
	etc.		
	—		
	—		
	—		
	LDA	SVA	Restore A-register
	LDB	SVB	Restore B-register
	STF	0	Enable interrupt system
	STF	5	Turn on parity error interrupt
	STC	5	Turn on memory protect interrupt
	JMP	MPPE,I	Exit the subroutine
PERR	—		User's routine in case of parity error
	—		
	—		
	etc.		
	—		
	—		
	—		
	JMP	PERR-6	Restore accumulators, turn on interrupts, exit

I/O DRIVERS, STRUCTURE



I/O DRIVERS, INITIATOR

ON ENTRY TO D.NN,

(A) = ADDRESS OF EQT ENTRY

(B) = ADDRESS OF WORD 2 OF I/O REQUEST

INITIATOR FUNCTIONS:

1. REJECT THE I/O REQUEST IF BUSY CONDITIONS EXIST OR THE REQUEST PARAMETERS ARE ILLEGAL.

2. SAVE THE I/O REQUEST PARAMETERS WITHIN DRIVER STORAGE.

3. CONFIGURE THE I/O INSTRUCTIONS IN THE DRIVER.

4. SET BUSY FLAGS.

5. INITIALIZE OPERATING CONDITIONS & ACTIVATE THE DEVICE.

6. RETURN TO .IOC. WITH A & B REGISTERS SET TO INDICATE INITIATION OR REJECTION.

ENTERED BY DEVICE INTERRUPT.

CONTINUATOR FUNCTIONS:

1. SAVE REGISTERS

2. IF DATA TRANSMISSION NOT COMPLETE,
   INITIATE THE NEXT DATA TRANSFER,
   RESTORE REGISTERS,
   RETURN CONTROL TO PROGRAM.

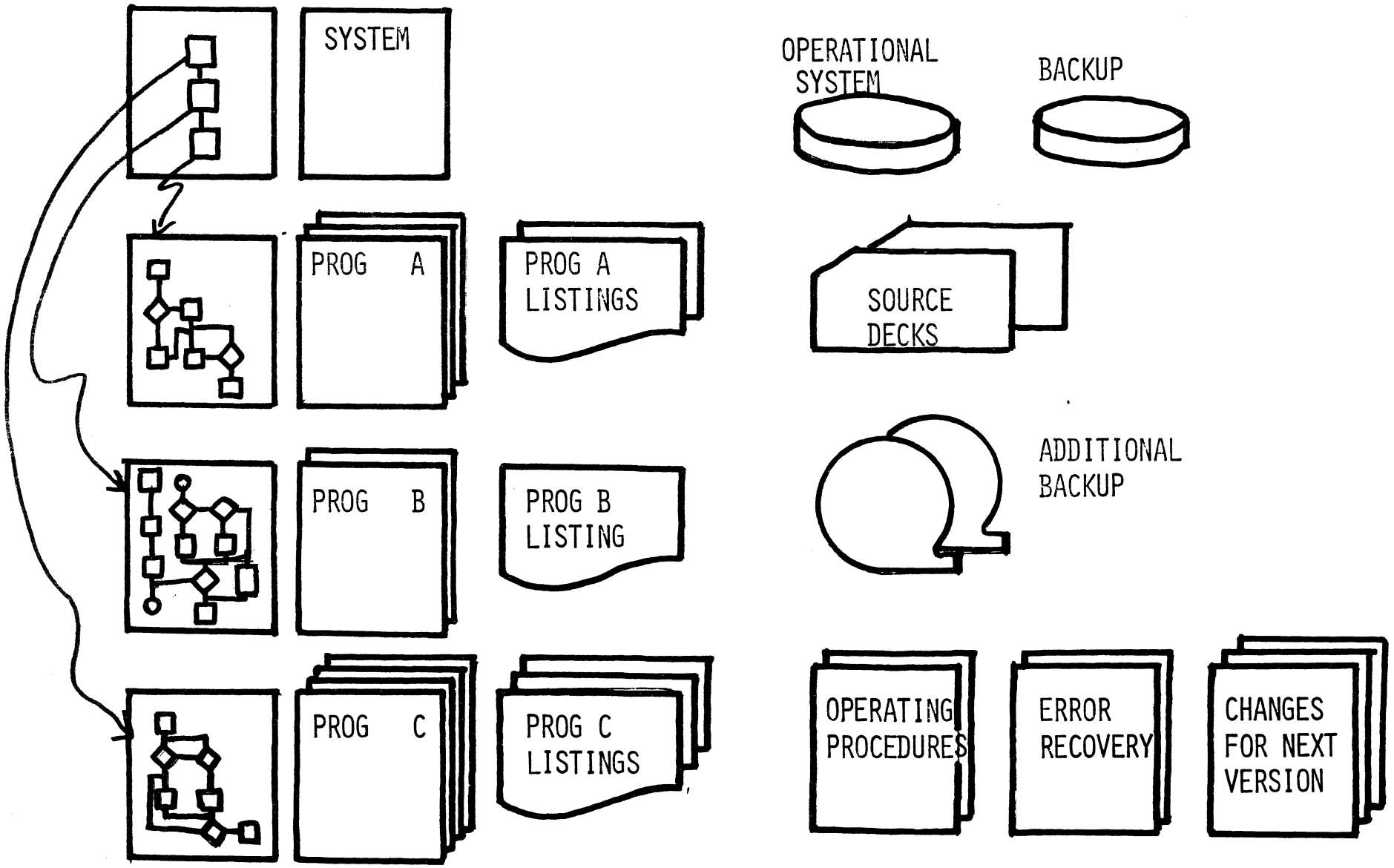
3. IF DATA TRANSMISSION COMPLETE, OR IF A MALFUNCTION OCCURS,
   UPDATE THE EQT.

4. CLEAR BUSY FLAGS.

5. RESTORE REGISTERS

6. RETURN VIA JMP I.NN,I.

A DOCUMENTATION TECHNIQUE



2100 BIBLIOGRAPHY

5950-9226	HP 2100 PROGRAM CATALOGUE
5951-4498	A POCKET GUIDE TO INTERFACING THE HP 2100 COMPUTER
02116-9017	BASIC CONTROL SYSTEM
02100-90136	DOS-III DISC OPERATING SYSTEM
0200-90002	A GUIDE TO TIME-SHARED BASIC
02005-90002	REAL-TIME EXECUTIVE SYSTEM
5951-3028	2100 COMPUTER MICROPROGRAMMING GUIDE
5951-4423	A POCKET GUIDE TO THE 2100 COMPUTER

B A S I C B I N A R Y L O A D E R P R O C E D U R E

1. HALT CPU (IF RUNNING).

2. PLACE ABSOLUTE TAPE (I.E. ASSEMBLER, OR BCS, ... ETC.) INTO PHOTOREADER, NARROW SIDE OF THE TAPE TOWARD CPU.

3. SET P TO X77ØØ, WHERE X = Ø FOR 4K, 1 FOR 8K, 2 FOR 12K, 3 FOR 16K, 5 FOR 24K
   7 FOR 32K.

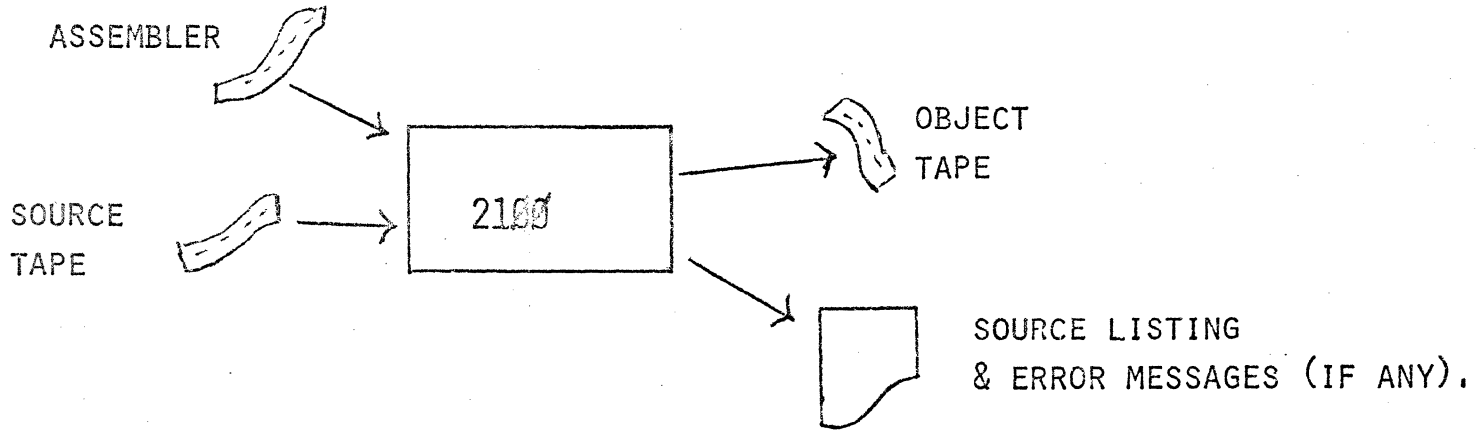
4. CLEAR S, PRESS EXTERNAL PRESET, INTERNAL PRESET, LOADER ENABLE, RUN.

5. TAPE READ, CPU THEN HALTS. 1Ø2Ø77 = OK,

   1Ø2Ø55 = ADDRESS TOO HIGH,

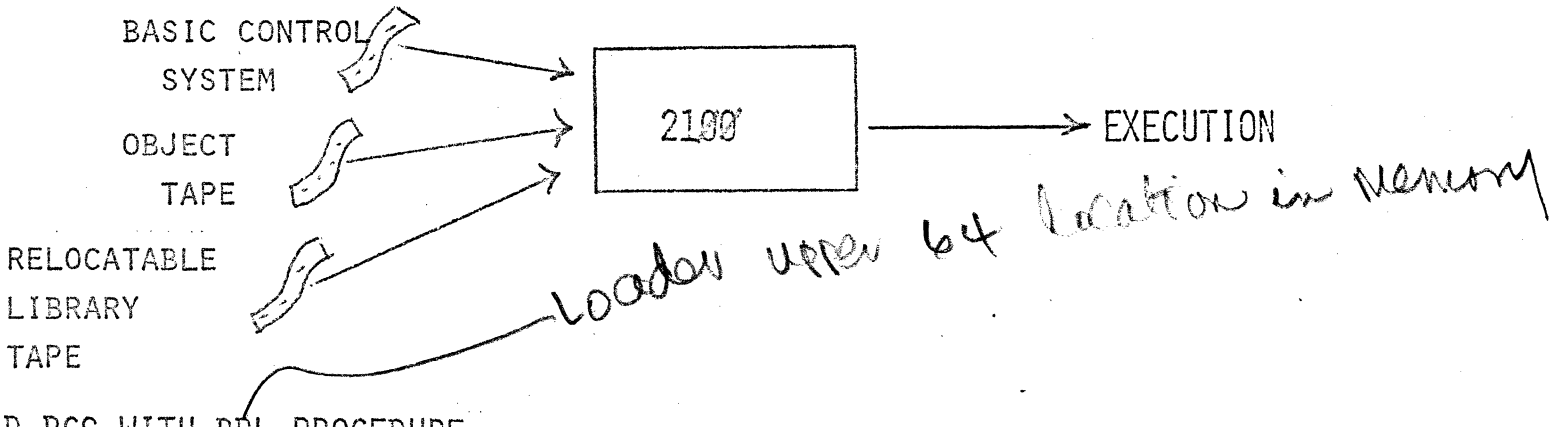
   1Ø2Ø11 = CHECKSUM (WRONG OR DAMAGED TAPE).

TYPICAL ASSEMBLY PROCEDURE



1. LOAD ASSEMBLER USING BBL PROCEDURE.
2. SET P TO 100.
3. PLACE SOURCE TAPE IN PHOTOREADER, PRESS RUN.
4. SOURCE TAPE READ, COMPUTER HALTS, 102011 IN DISPLAY REGISTER.
5. REWIND SOURCE TAPE & PLACE BACK IN PHOTOREADER, PRESS RUN.
6. SOURCE TAPE READ, OBJECT TAPE PUNCHED, SOURCE LISTED.
7. COMPUTER HALTS, 102077 IN DISPLAY REGISTER.

TYPICAL EXECUTION PROCEDURE



1. LOAD BCS WITH BBL PROCEDURE.
2. SET P TO 2, SET SWITCH REGISTER BIT 15 (S15) TO 1.
3. PLACE OBJECT TAPE IN PHOTOREADER, PRESS RUN.
4. OBJECT TAPE READ.
5. *LOAD PRINTS ON SYSTEM CONSOLE, COMPUTER HALTS WITH 102001.
6. IF MORE OBJECT TAPES, THEN GO TO STEP 3, ELSE NEXT STEP.
7. SET S2 TO 1, PLACE RELOCATABLE LIBRARY TAPE IN PHOTOREADER, PRESS RUN.
8. RELOCATABLE LIBRARY READ (IF NECESSARY).
9. *LST PRINTS ON SYSTEM CONSOLE, COMPUTER HALTS WITH 102000.
10. PRESS RUN.
11. *RUN PRINTS ON SYSTEM CONSOLE, COMPUTER HALTS WITH 102000.
12. PRESS RUN, PROGRAM EXECUTES.

EDIT COMMANDS

/D,M,N DELETE LINE. IF M ONLY ENTERED, DELETE LINE M ONLY.
IF M & N ENTERED, DELETE LINES M THRU N

E.G. /D,10 LINE 10 DELETED
/D,15,30 LINES 15,16,17,...,29,30 DELETED.

/R,M,N REPLACE LINE. IF M ONLY ENTERED, REPLACE LINE M ONLY.
IF M & N ENTERED, REPLACE LINES M THRU N.

E.G. /R,10 LINE 10 REPLACED BY
START NOP
LDA Z

/R,15,30 LINES 15 THRU 30 REPLACED BY
JMP X

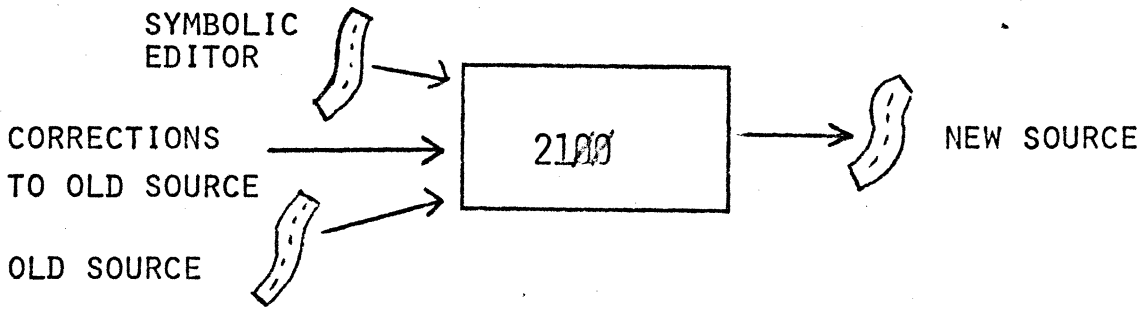
/I,M INSERT LINE. INSERT WHATEVER FOLLOWS THE /I STATEMENT AFTER LINE M

E.G. /I,10 OLD LINE 10 RETAINED & NOW FOLLOWED BY
IOR 1
XOR MASK

OLD LINE 11 NOW FOLLOWS XOR MASK LINE.

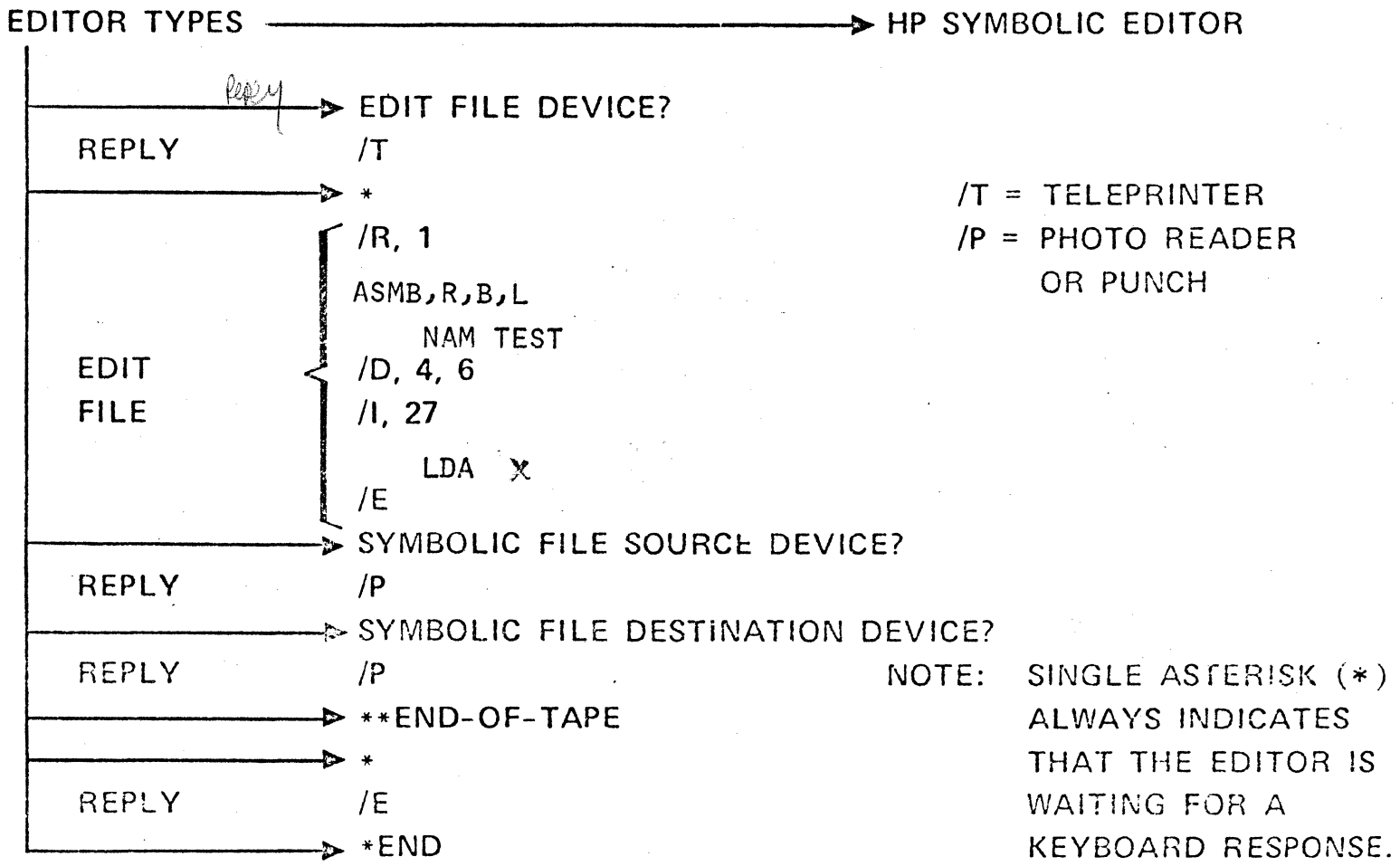
ALWAYS USE OLD LINE NUMBERS AS REFERENCE.
LINE NUMBERS CAN ONLY BE REFERRED TO ONCE, & MUST BE IN ASCENDING ORDER.
WHEN ALL DELETES, REPLACES, & INSERTS HAVE BEEN ENTERED, /E MUST BE ENTERED.

TYPICAL EDIT PROCEDURE



1. LOAD SYMBOLIC EDITOR WITH BBL PROCEDURE.
2. SET P TO 100, RUN.
3. "HP SYMBOLIC EDITOR" PRINTS ON LIST DEVICE.
4. "EDIT FILE DEVICE?" PRINTS ON LIST DEVICE (NEXT LINE).
5. KEY IN "/T" ON SYSTEM CONSOLE
6. "*" PRINTS ON LIST DEVICE.
7. KEY IN REQUIRED EDIT STATEMENTS. LAST EDIT STATEMENT MUST BE "/E".
8. "SYMBOLIC FILE SOURCE DEVICE?" PRINTS ON LIST DEVICE.
9. PLACE OLD SOURCE TAPE IN PHOTOREADER.
10. KEY IN "/P" (FOR PHOTOREADER).
11. "SYMBOLIC FILE DESTINATION DEVICE?" PRINTS ON LIST DEVICE.
12. KEY IN "/P" (FOR PUNCH)
13. NEW SOURCE TAPE PUNCHED.
14. "** END-OF-TAPE" PRINTS ON LIST DEVICE.
15. "*" PRINTS ON LIST DEVICE.
16. KEY IN "/E".
17. NEW SOURCE TAPE TRAILER PUNCHED.
18. "* END" PRINTS ON LIST DEVICE, EDIT PROCESS COMPLETE.

SAMPLE EDIT


EDITOR TYPES
HP SYMBOLIC EDITOR
EDIT FILE DEVICE?
REPLY
/T
*
/R, 1
ASMB,R,B,L
NAM TEST
EDIT
/D, 4, 6
FILE
/I, 27
LDA X
/E
SYMBOLIC FILE SOURCE DEVICE?
REPLY
/P
SYMBOLIC FILE DESTINATION DEVICE?
REPLY
/P
**END-OF-TAPE
*
REPLY
/E
*END
/T = TELEPRINTER
/P = PHOTO READER
OR PUNCH
NOTE: SINGLE ASTERISK (*)
ALWAYS INDICATES
THAT THE EDITOR IS
WAITING FOR A
KEYBOARD RESPONSE.

EDITING ON A TTY (NO CPU AVAILABLE)

1. PLACE OLD SOURCE TAPE IN TTY TAPE READER.

2. PLACE TTY IN LOCAL MODE.

3. PUSH TTY PUNCH ON BUTTON.

4. PUNCH LEADER BY DEPRESSING HERE-IS KEY (3X).

5. PLACE TTY STOP/START/FREE SWITCH IN START POSITION.

6. TTY WILL NOW LIST & DUPLICATE OLD SOURCE TAPE.

7. WHEN THE TTY IS DUPLICATING THE LINE TO BE CHANGED, PLACE STOP/START/FREE SWITCH TO STOP POSITION.

8. KEY IN RUBOUT (CR) (LF), THE CORRECT LINE (CR) (LF), AND ANOTHER RUBOUT (DO NOT FOLLOW THIS SECOND RUBOUT WITH A (CR) (LF)).

9. PLACE STOP/START/FREE SWITCH IN START POSITION.

10. CONTINUE STEPS 6 THRU 10.UNTIL ENTIRE TAPE EDITED.

DAY 1 QUIZ (QUESTIONS 1 - 4)

1. Convert 32677_{10} to binary; -1000_{10} to octal.

   A. $32677_{10} = 0\ 011\ 010\ 110\ 111\ 111_2$; $-1000_{10} = 177266_8$

   B. $32677_{10} = 0\ 111\ 111\ 110\ 100\ 101_2$; $-1000_{10} = 176030_8$

   C. $32677_{10} = 0\ 011\ 010\ 110\ 111\ 111_2$; $-1000_{10} = 176030_8$

   D. $32677_{10} = 0\ 111\ 111\ 110\ 100\ 101_2$; $-1000_{10} = 177266_8$

2. Which instruction will put the contents of the B-register into the A-register?

   A. OTB 0
   B. LDB 0
   C. STB 0
   D. LIB 0

3. Which set of codes will load the A-register from the switch register and HLT if A-register contents = B-register contents.

   A.
   ```
   HERE  LDA  1
         CPA  B
         HLT
         JMP  HERE
   ```

   B.
   ```
   HERE  LIA  1
         CPA  B
         HLT
         JMP  HERE
   ```

   C.
   ```
   HERE  LDA  1
         CPB  0
         HLT
         JMP  HERE
   ```

   D.
   ```
   HERE  LIA  1
         CPB  0
         HLT
         JMP  HERE
   ```

4. Which of the following best describes the ASSEMBLY process? The assembler program is loaded into computer memory and:

   A. The source tape is processed in two passes producing an assembly listing and the object tape.

   B. The object tape is processed in two passes producing an assembly listing and a new object tape.

   C. The old source tape is processed, producing a new source tape.

   D. The source tape is processed in one pass producing an assembly listing and the object tape.

DAY 1 QUIZ (QUESTIONS 5-7)

5. Which of the following statements best describes the loading of an object program using the basic control system? The BCS tape is loaded into the computer and:

   A. The loader listing is produced, then the object program is loaded, then the relocatable library routines are loaded.

   B. The object tape is loaded, then the relocatable library routines are loaded.

   C. The relocatable library routines are loaded, then the object tape is loaded.

   D. The source tape is loaded, then the relocatable library routines are loaded.

6. Which of the following best describes the edit process? The editor is loaded into the computer and:

   A. The old source is read, the edits keyed in, the new source is punched.

   B. The edits are keyed in, the old source is punched, the new source is read.

   C. The edits are keyed in, the old source is read, the new source is punched.

   D. The old source is punched, the edits keyed in, the new source read.

7. When the following code executes (from START), and then halts. what will be in the T,A,M, & P registers.
   Assume START is location 2ØØØB.
   Note: T-register is another name for the MEMORY DATA register.

```
2ØØØ        START   NOP
                    LDA VALUE
                    AND MASK
                    HLT
MASK                OCT 17Ø763
VALUE               OCT 135Ø26
```

M= address of the instruction in memory

   A. T=1Ø2Ø77, A=175767, M=2ØØ4, P=2ØØ3
   B. T=1Ø2ØØØ, A=175767, M=2ØØ4, P=2ØØ3
   C. T=1Ø2Ø77, A=13ØØ22, M=2ØØ3, P=2ØØ4
   D. T=1Ø2ØØØ, A=13ØØ22, M=2ØØ3, P=2ØØ4

EXPANDED BCS PROCEDURE

(INCLUDES LOAD MAP, ENTRY POINT LIST & ABSOLUTE TAPE OPTIONS)

1. LOAD BCS WITH BBL PROCEDURE.
2. SET P TO 2.

3. IF ABSOLUTE TAPE DESIRED, SET S14 TO 1, PRESS EXTERNAL & INTERNAL PRESETS.

4. IF LOAD MAP DESIRED, SET S15 TO Ø.

5. PLACE OBJECT TAPE IN PHOTOREADER, PRESS RUN.
6. OBJECT TAPE READ.
7. *LOAD PRINTS ON SYSTEM CONSOLE, COMPUTER HALTS WITH 1Ø2ØØ1.
8. IF MORE OBJECT TAPES, GO TO STEP 4.

9. SET S2 TO 1.
10. IF LOAD MAP DESIRED, SET S15 TO Ø.
11. PLACE RELOCATABLE LIBRARY IN PHOTOREADER, PRESS RUN.
12. RELOCATABLE LIBRARY READ (IF NECESSARY!)

loader symbol table

13. *LST PRINTS ON SYSTEM CONSOLE. COMPUTER HALTS WITH 1Ø2ØØØ.
14. IF ENTRY POINT LIST DESIRED, SET S15 TO Ø.
15. PRESS RUN. IF S15 = Ø, ENTRY POINT LIST PRINTS ON SYSTEM CONSOLE

16. IF IN STEP 3, S14 WAS SET TO 1, *END PRINTS ON SYSTEM CONSOLE, COMPUTER HALTS WITH 1Ø2Ø77, ABSOLUTE TAPE HAS BEEN PUNCHED. TO EXECUTE, LOAD ABSOLUTE TAPE WITH BBL, SET P TO 2, PRESS RUN, PROGRAM EXECUTES.

    IF IN STEP 3, S14 WAS SET TO Ø, *RUN PRINTS ON SYSTEM CONSOLE, COMPUTER HALTS WITH 1Ø2ØØØ. PRESS RUN, PROGRAM EXECUTES.