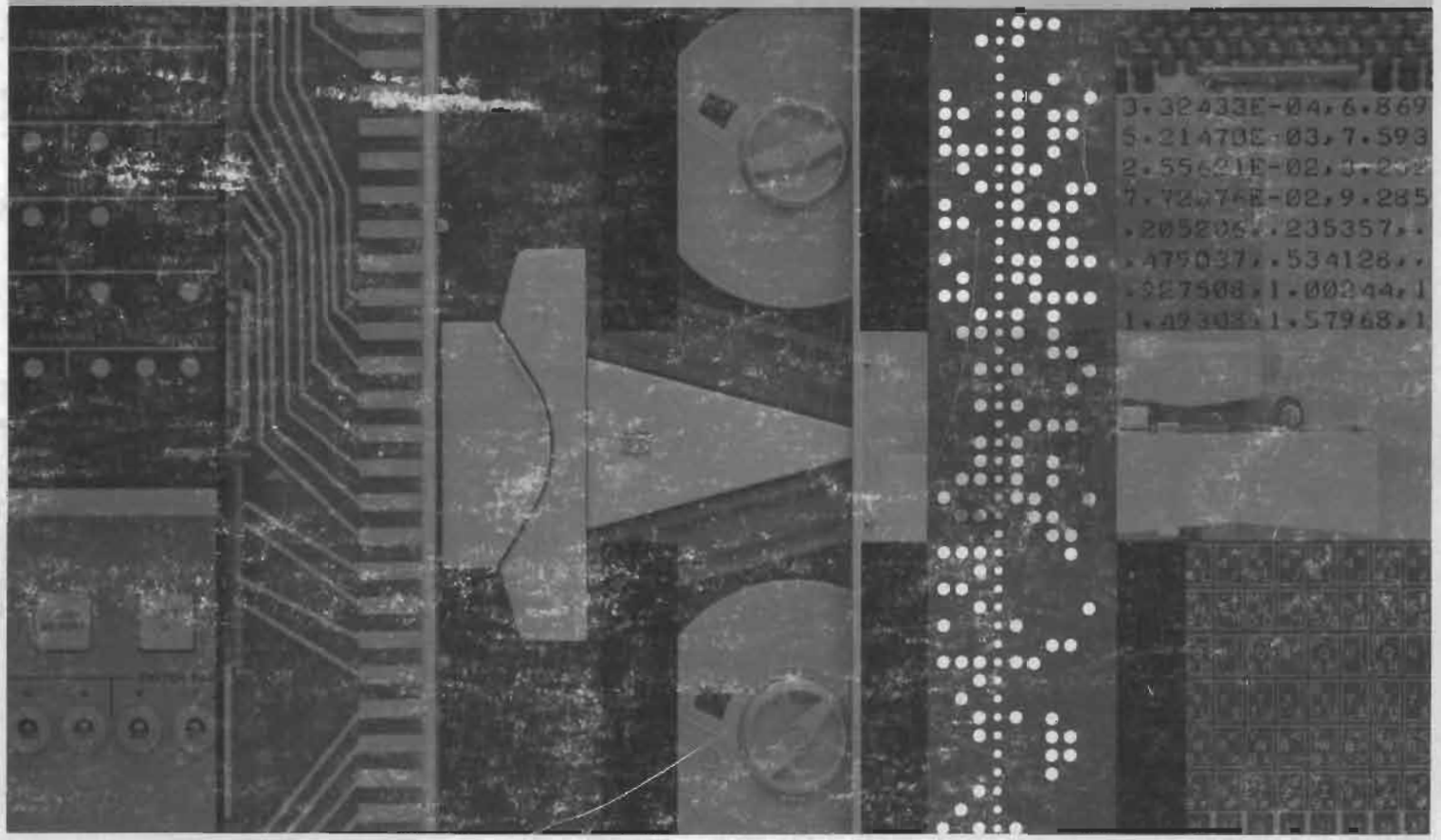
HEWLETT PACKARD

COMPUTER MAINTENANCE COURSE



VOL. I FUNDAMENTALS OF HARDWARE, SOFTWARE AND PROGRAMMING

HEWLETT-PACKARD

COMPUTER MAINTENANCE COURSE

VOLUME I

FUNDAMENTALS OF HARDWARE, SOFTWARE AND PROGRAMMING

(HP STOCK NO. 5950-8703)

-NOTICE-

The information contained in this manual is for training purposes only. Consult the Hewlett-Packard documentation supplied with the computer for current information concerning the specific computer system furnished.

11000 Wolfe Road, Cupertino, California 95014 Area Code 408 257-7000 TWX 910-338-0221

Printed in U.S.A. APR 69

Hewlett-Packard

Fundamentals of Hardware, Software & Programming

OBJECTIVES

1. INTRODUCE THE STUDENT TO THE BASIC ELEMENTS OF HP COMPUTER HARDWARE, SOFTWARE AND PROGRAMMING.
2. TEACH THE STUDENT HOW TO CREATE AND EXECUTE SIMPLE MACHINE AND ASSEMBLY LANGUAGE PROGRAMS.
3. PROVIDE THE STUDENT WITH "HANDS-ON" COMPUTER EXPERIENCE.
4. ACQUAINT THE STUDENT WITH THE COMPUTER USER'S ENVIRONMENT.

LESSON I OBJECTIVES

I. INTRODUCE THE STUDENT TO THE BASIC ELEMENTS OF COMPUTER HARDWARE.

II. INTRODUCE THE STUDENT TO NUMBER SYSTEMS & NUMBER SYSTEM CONVERSION TECHNIQUES.

HP COMPUTERS ARE COMPACT GENERAL PURPOSE COMPUTERS

COMPUTERS OF THIS TYPE ——————

I. CAN DO ARITHMETIC OPERATIONS

II. CAN MAKE LOGICAL DECISIONS

III. CAN RETAIN INFORMATION IN A MEMORY

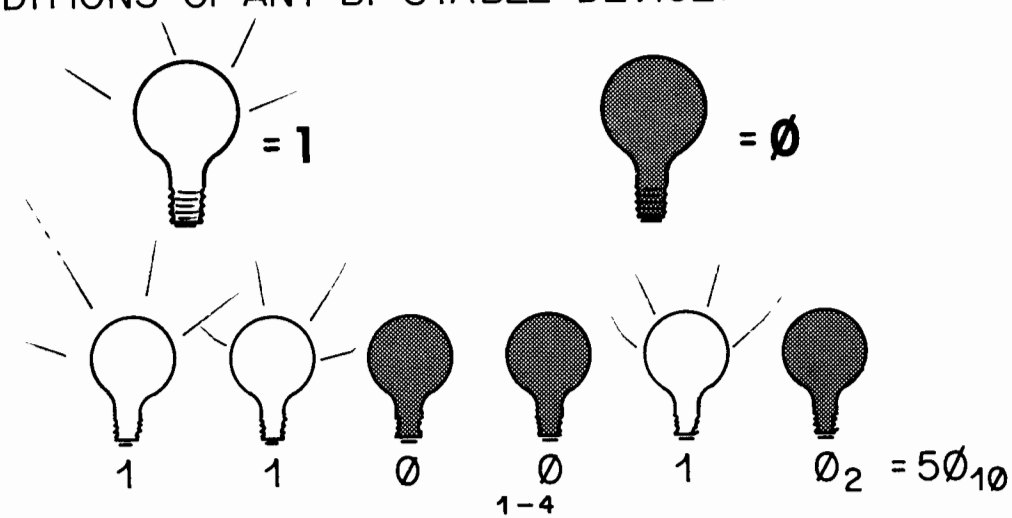
IV. CAN COMMUNICATE WITH THE OPERATOR IN SOME WAY

IN ORDER TO COMMUNICATE WITH A COMPUTER,

We Must Speak It's Language

ALL COMMUNICATION WITH A COMPUTER MUST BE IN BINARY FORM SINCE THE COMPUTER USES BI-STABLE DEVICES TO STORE INFORMATION.

WE CAN USE THE TWO DIGITS Ø & 1, TO REPRESENT THE TWO CONDITIONS OF ANY BI-STABLE DEVICE.



INTRODUCTION TO NUMBER SYSTEMS

HEWLETT-PACKARD computers operate on numbers in binary form; therefore, it is essential that we:

1. REVIEW THE DECIMAL NUMBER SYSTEM
2. INTRODUCE THE BINARY AND OCTAL NUMBER SYSTEMS
3. INTRODUCE BINARY ARITHMETIC
4. INTRODUCE NUMBER SYSTEM CONVERSION METHODS
5. DISCUSS THE LIMITS OF THE COMPUTER'S ABILITY TO HANDLE LARGE NUMBERS

NUMBER SYSTEMS

0,1,2,3,4,5,6,7,8,9 ARE THE TEN NUMERALS OF THE DECIMAL SYSTEM. DECIMAL VALUES LARGER THAN 9 REQUIRE MORE THAN ONE DIGIT. FOR EXAMPLE, THE DECIMAL NUMBER 109 REALLY STANDS FOR:

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

(HUNDRED'S) + (TEN'S) + (ONE'S)

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

IN GENERAL:

ANY NUMBER = $N \times b^n + N \times b^{n-1} + \cdots + N \times b^2 + N \times b^1 + N \times b^0$

WHERE N = DIGIT
b = BASE
$b^0 = 1$ (BY DEFINITION)

BINARY NUMBERS

Ø and 1 are the TWO numerals of the binary system. Binary values larger than 1 require more than one digit. For example, the BINARY number 1101101 really stands for:

$$(1 \times 2^6) + (1 \times 2^5) + (0 \times 2^4) + (1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0)$$

(SIXTY-FOUR'S) + (THIRTY-TWO'S) + (SIXTEEN'S) + (EIGHT'S) + (FOUR'S) + (TWO'S) + (ONE'S)

$$(64) + (32) + (0) + (8) + (4) + (0) + (1) = 109_{10}$$

THEREFORE:

$$1101101_2 = 109_{10}$$

OCTAL NUMBERS

0,1,2,3,4,5,6,7 ARE THE EIGHT NUMERALS OF THE OCTAL SYSTEM. OCTAL VALUES LARGER THAN 7 REQUIRE MORE THAN ONE DIGIT. FOR EXAMPLE, THE OCTAL NUMBER 155 REALLY STANDS FOR:

$$(1 \times 8^2) + (5 \times 8^1) + (5 \times 8^0)$$

SIXTY
(FOUR'S) + (EIGHT'S) + (ONE'S)

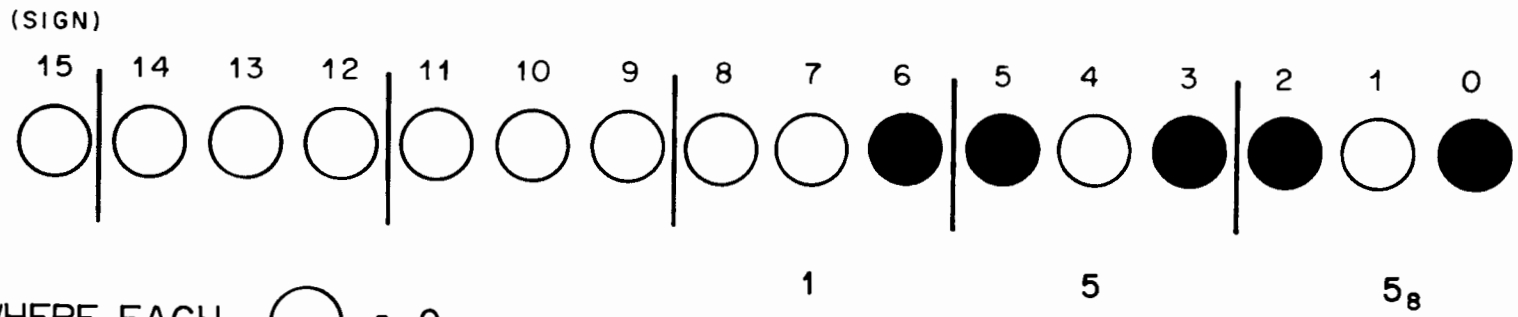
$$(\ 64\ ) + (\ 40\ ) + (\ 5\ ) = 109_{10}$$

THEREFORE —

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

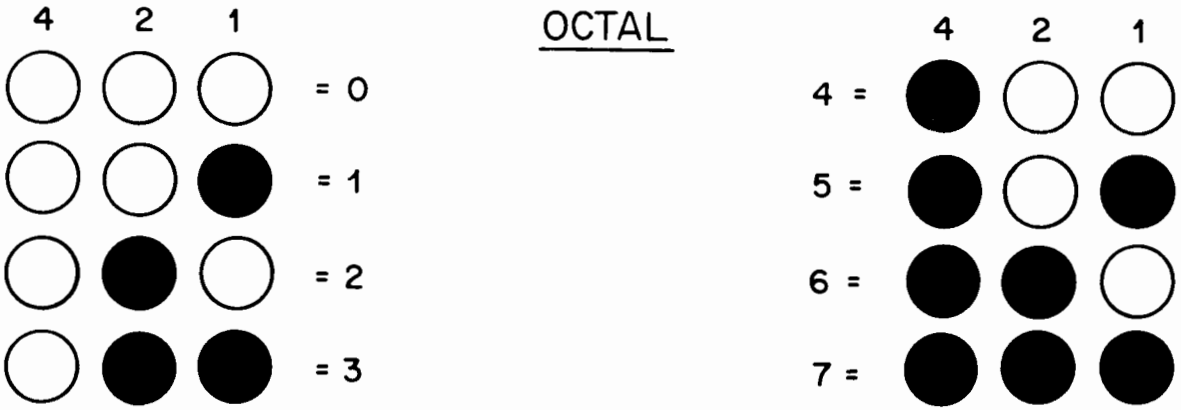
BINARY/OCTAL RELATIONSHIP

HEWLETT-PACKARD COMPUTERS HAVE 16 BINARY DIGITS. (BIT) WHEN BINARY DIGITS (BITS) ARE ARRANGED IN GROUPS OF 3, OCTAL VALUES CAN BE READ DIRECTLY.



WHERE EACH ○ = 0

AND EACH ● = 1



NUMBER SYSTEM CONVERSION METHODS

PROGRAMMERS MUST LEARN THE FOLLOWING NUMBER SYSTEM CONVERSION TECHNIQUES:

CONVERSION	METHOD
BINARY TO OCTAL	BY INSPECTION
OCTAL TO BINARY	BY INSPECTION
OCTAL TO DECIMAL	BY FORMULA
DECIMAL TO OCTAL	BY FORMULA

REMEMBER:

OCTAL IS USED TO REPRESENT BINARY NUMBERS MORE EFFICIENTLY

SIGN ← number value →

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

OCTAL TO DECIMAL CONVERSION

- TO CONVERT THE OCTAL NUMBER 155 TO DECIMAL, PROCEED IN THE FOLLOWING WAY.

1. **Multiply the most significant octal digit by 8**
2. **Add the next least significant octal digit, then multiply the result by 8**
3. **Continue using step 2 above until the least significant digit is reached**
4. **The least significant digit is added to the total but the result is NOT multiplied by 8.**

EXAMPLE:

CONVERT 155_8 TO DECIMAL

```
  155
x 8
  8
+ 5
 13
x 8
104
+ 5
109₁₀   DECIMAL RESULT
```

DECIMAL TO OCTAL CONVERSION

TO CONVERT THE DECIMAL NUMBER 109 TO OCTAL PROCEED IN THE FOLLOWING WAY-

1. DIVIDE THE DECIMAL NUMBER BY 8 AND WRITE DOWN THE REMAINDER. — $8\overline{)109}$ / 13 + 5 REMAINDER
2. DIVIDE THE QUOTIENT OF THE PREVIOUS STEP BY 8 AND WRITE DOWN THE REMAINDER. — $8\overline{)13}$ / 1 + 5 REMAINDER
3. REPEAT STEP 2 UNTIL THE "NEW" QUOTIENT BECOMES ZERO. — $8\overline{)1}$ / 0 + 1 REMAINDER

   ↑ READ OCTAL VALUE

4. THE OCTAL VALUE IS READ AS FOLLOWS:
   - THE LAST REMAINDER IS THE <u>MOST</u> SIGNIFICANT OCTAL DIGIT.
   - THE FIRST REMAINDER IS THE <u>LEAST</u> SIGNIFICANT OCTAL DIGIT

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

BINARY ARITHMETIC

- *In the computer a special logic circuit performs addition using binary arithmetic. Actual computer numbers are 16 "BITS" long, however, for simplicity the following example uses only 6 "BITS."*

EXAMPLE

GENERATED CARRIES

```
        1 1
X       0 0 1 1 1 0
Y       0 0 1 1 0 1
        -----------
sum     0 1 1 0 1 1
```

RULES OF BINARY ADDITION

CARRY (IN)	X	Y
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

SUM	CARRY (OUT)
0	0
1	0
1	0
0	1
1	0
0	1
0	1
1	1

TWO'S COMPLEMENT NUMBERS

HEWLETT-PACKARD COMPUTERS USE THE TWO'S COMPLEMENT ARITHMETIC TECHNIQUE. THE PROCESS OF "TWO'S COMPLEMENTATION" CHANGES A POSITIVE INTEGER VALUE TO NEGATIVE AND VICE-VERSA.

NOTE: IF SIGN = 0, NORMAL FORM (POSITIVE)
IF SIGN = 1, TWO'S COMPLEMENT FORM (NEGATIVE)

FOR EXAMPLE:

SIGN	VALUE					
0	1	1	0	1	1	A NORMAL NUMBER (POSITIVE)
1	0	0	1	0	0	THE ONE'S COMPLEMENT { ALL 1's BECOME 0's, ALL 0's BECOME 1's }
0	0	0	0	0	1	ADD ONE
1	0	0	1	0	1	THE TWO'S COMPLEMENT (NEGATIVE)

COMPLEMENTATION TECHNIQUES

The decimal number 109_{10} when converted to octal appears as 155_8. The example shows the two's complement operation performed on this value.

EXAMPLE:

SIGN	BINARY		OCTAL
0	000 000 001 101 101	(POSITIVE)	0 00155
1	111 111 110 010 010	(ONE'S COMPLEMENT)	1 77622
0	000 000 000 000 001	(ADD ONE)	0 00001
1	111 111 110 010 011	(NEGATIVE TWO'S COMPLEMENT)	1 77623

NOTE THE MOST SIGNIFICANT OCTAL DIGIT REPRESENTS A SINGLE BIT. TO COMPLEMENT WITH OCTAL NUMBERS REMEMBER —

1 — COMPLEMENT THE SIGN DIGIT. (1 or Ø)

2 — TAKE THE EIGHTS COMPLEMENT ON THE REMAINING DIGITS.

NEGATIVE NUMBER CONVERSIONS

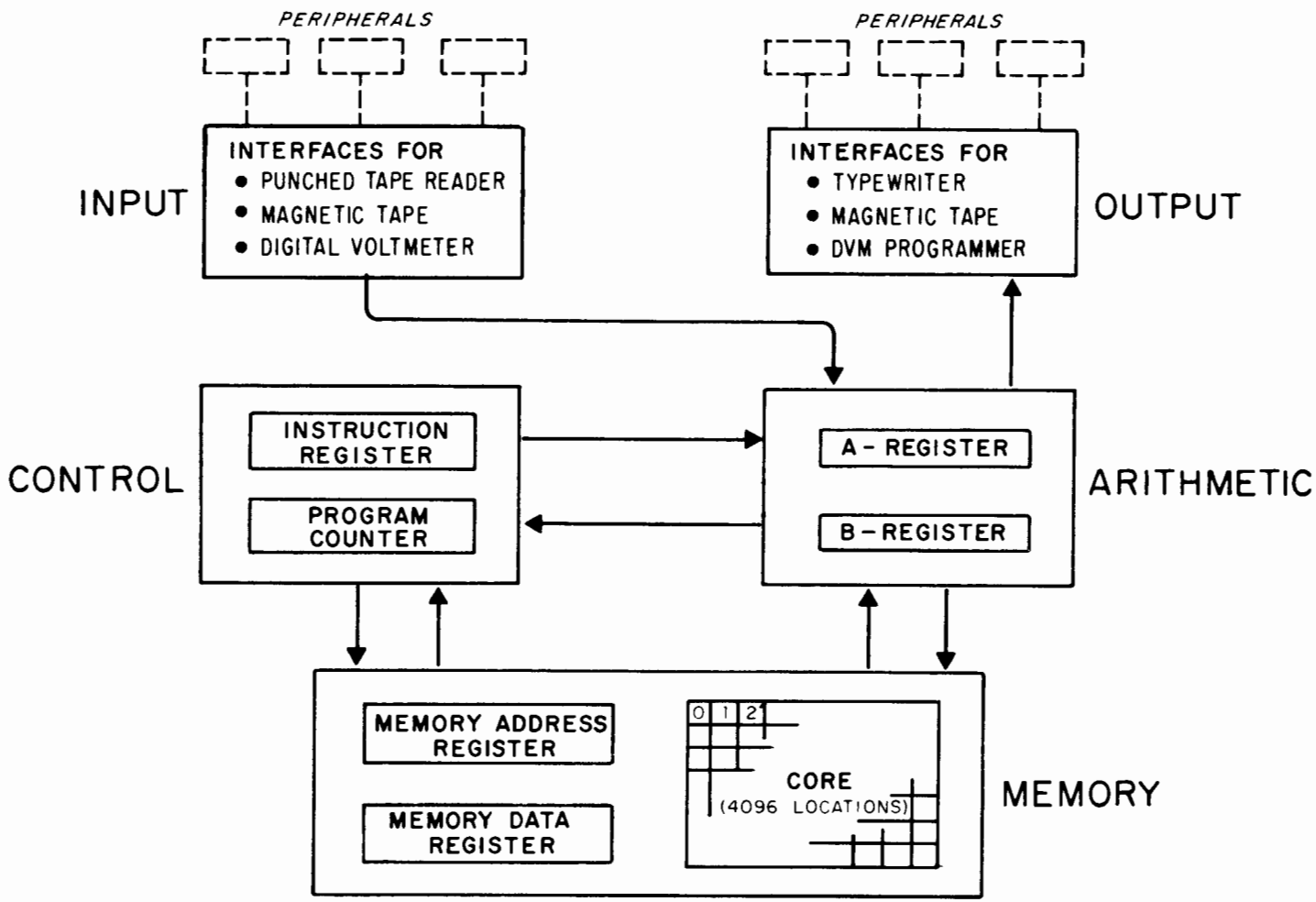
TO CONVERT A NEGATIVE DECIMAL NUMBER TO 16 BIT MACHINE FORM.

1. ASSUME THE DECIMAL VALUE IS POSITIVE
2. CONVERT TO OCTAL FORM
3. TAKE THE TWO'S COMPLEMENT. (OR EIGHT'S COMPLEMENT)

TO CONVERT TWO'S COMPLEMENT NUMBERS TO DECIMAL FORM.

1. TAKE THE TWO'S COMPLEMENT.
2. CONVERT TO DECIMAL
3. AFFIX A MINUS SIGN TO THE DECIMAL RESULT

BASIC ELEMENTS OF COMPUTER HARDWARE



THE COMPUTER WORD

AN -HP- **COMPUTER WORD** IS A GROUP OF 16 **BITS**

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

COMPUTER WORDS ARE USED TO REPRESENT:

1 **DATA** (NUMERIC AND ALPHABETIC)

2 COMPUTER **INSTRUCTIONS**

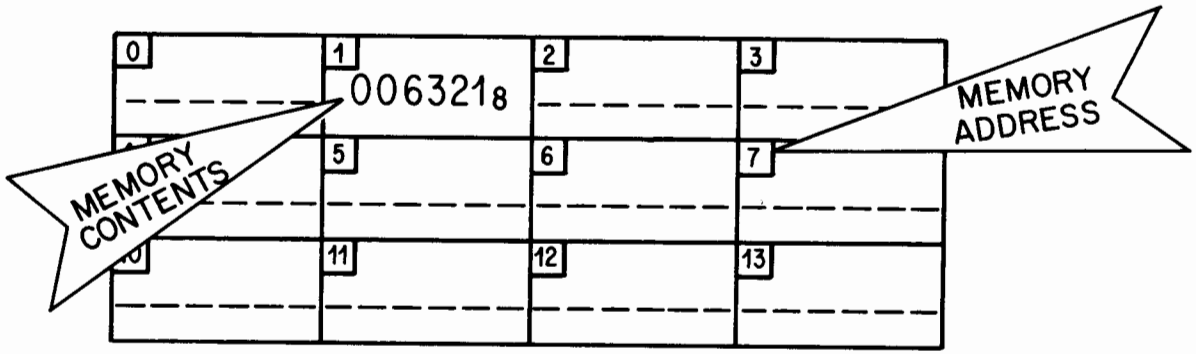
3 COMPUTER MEMORY **ADDRESSES**

MEMORY

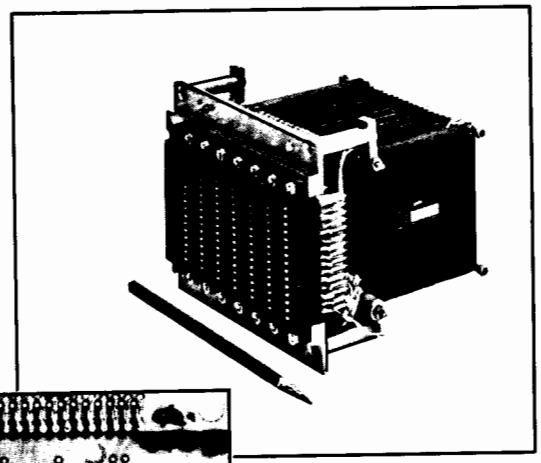
THE MEMORY OF A COMPUTER CONSISTS OF SOME NUMBER OF "MEMORY LOCATIONS"

EACH MEMORY LOCATION IS IDENTIFIED BY A "UNIQUE ADDRESS"

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

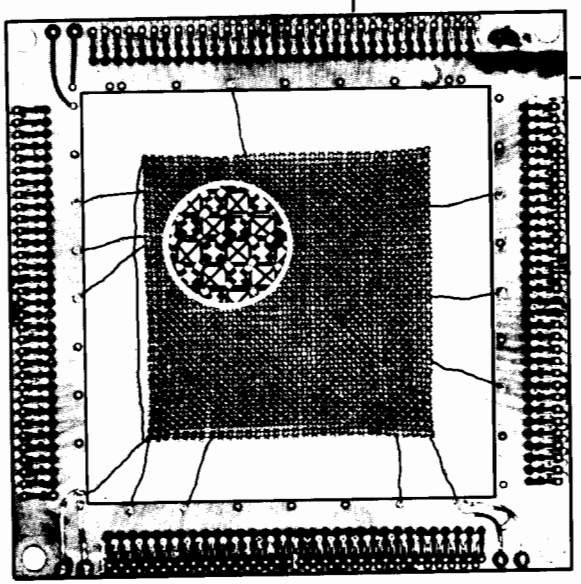


CORE MEMORY



4096 – Word
Core Module

17 CORE PLANES PER MODULE. EACH CORE PLANE SUPPLIES ONE BIT OF THE COMPUTER WORD. (16 DATA BITS + PARITY BIT).



Memory Plane

4096 CORES PER MEMORY PLANE. ONLY ONE CORE ON EACH PLANE IS INTERROGATED WHEN A MEMORY LOCATION IS ADDRESSED.

* THREE KINDS OF COMPUTER WORDS IN A MEMORY LOCATION

Data words — store data used in computations such as: 5, 10, +32767, AB, CD, −32767.

Instruction words — are orders that tell the machine what to do — such as add, shift or store data.

Address words — are used to specify a 15 bit memory address value in the range $0 - 32767_{10}$

TYPES OF COMPUTER INSTRUCTIONS

THERE ARE THREE TYPES OF COMPUTER INSTRUCTIONS —

- *Memory Reference*
- *Register Reference*
- *Input / output*

MEMORY REFERENCE INSTRUCTION

— USED FOR —

READING DATA FROM MEMORY

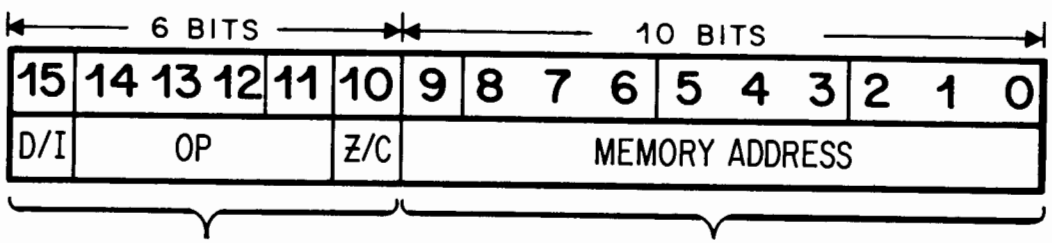
STORING DATA IN MEMORY

ARITHMETIC OPERATIONS

LOGIC OPERATIONS

ALTERATION OF PROGRAM COUNTER

CONTROLLING PROGRAM LOOPS



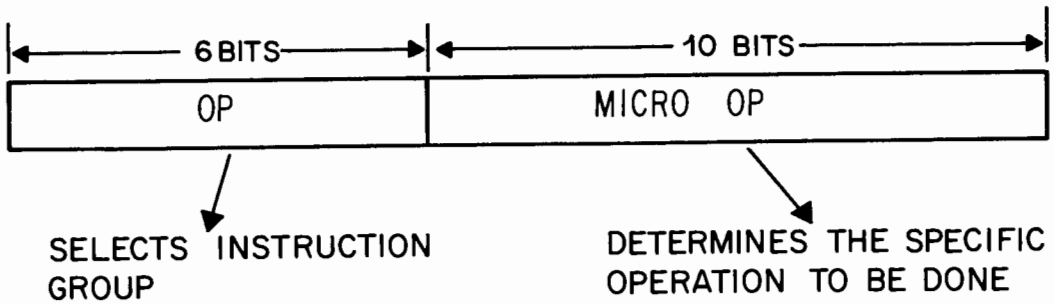
SELECTS 1 OF 14 INSTRUCTIONS AND DETERMINES ADDRESSING MODE

SPECIFIES THE MEMORY WORD ADDRESS

REGISTER REFERENCE INSTRUCTION

- MOVE DATA WITHIN AND BETWEEN ACCUMULATORS
- CLEAR OR COMPLEMENT ACCUMULATORS
- TEST BITS IN ACCUMULATORS

INSTRUCTION FORMAT



INPUT OUTPUT INSTRUCTIONS

- READ DATA FROM DEVICES
- OUTPUT DATA TO DEVICES
- CHECK STATUS OF DEVICES

15 14 13 12 11 10	9 8 7 6	5 4 3 2 1 0
OP	SUB OP	I/O SELECT CODE

SELECTS INSTRUCTION GROUP

DETERMINES SPECIFIC OPERATION TO BE PERFORMED.

IDENTIFIES WHICH INPUT OR OUTPUT DEVICE THE INSTRUCTION REFERS TO.

FIVE BASIC WORD FORMATS –

ARE USED IN HP COMPUTERS TO REPRESENT INSTRUCTIONS, ADDRESSES AND DATA.

1. Memory Reference Instruction

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D/I	OP				Z/C	WORD ADDRESS									

2. Register Reference Instruction

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

3. Input-output Instruction

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

4. Full Address

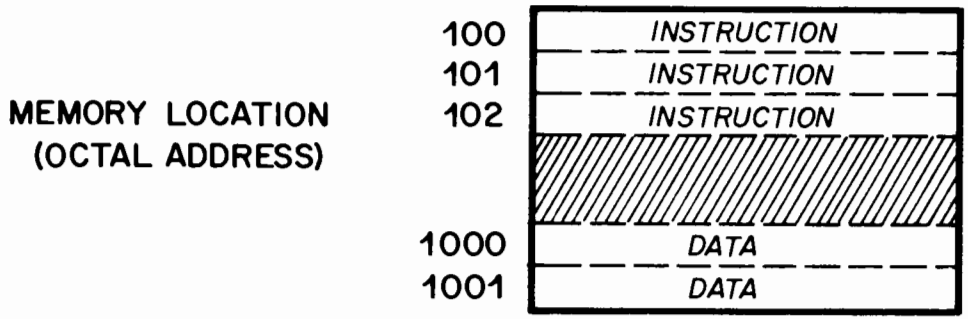
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D/I	PAGE ADDRESS					WORD ADDRESS									

5. Data (single-precision fixed point)

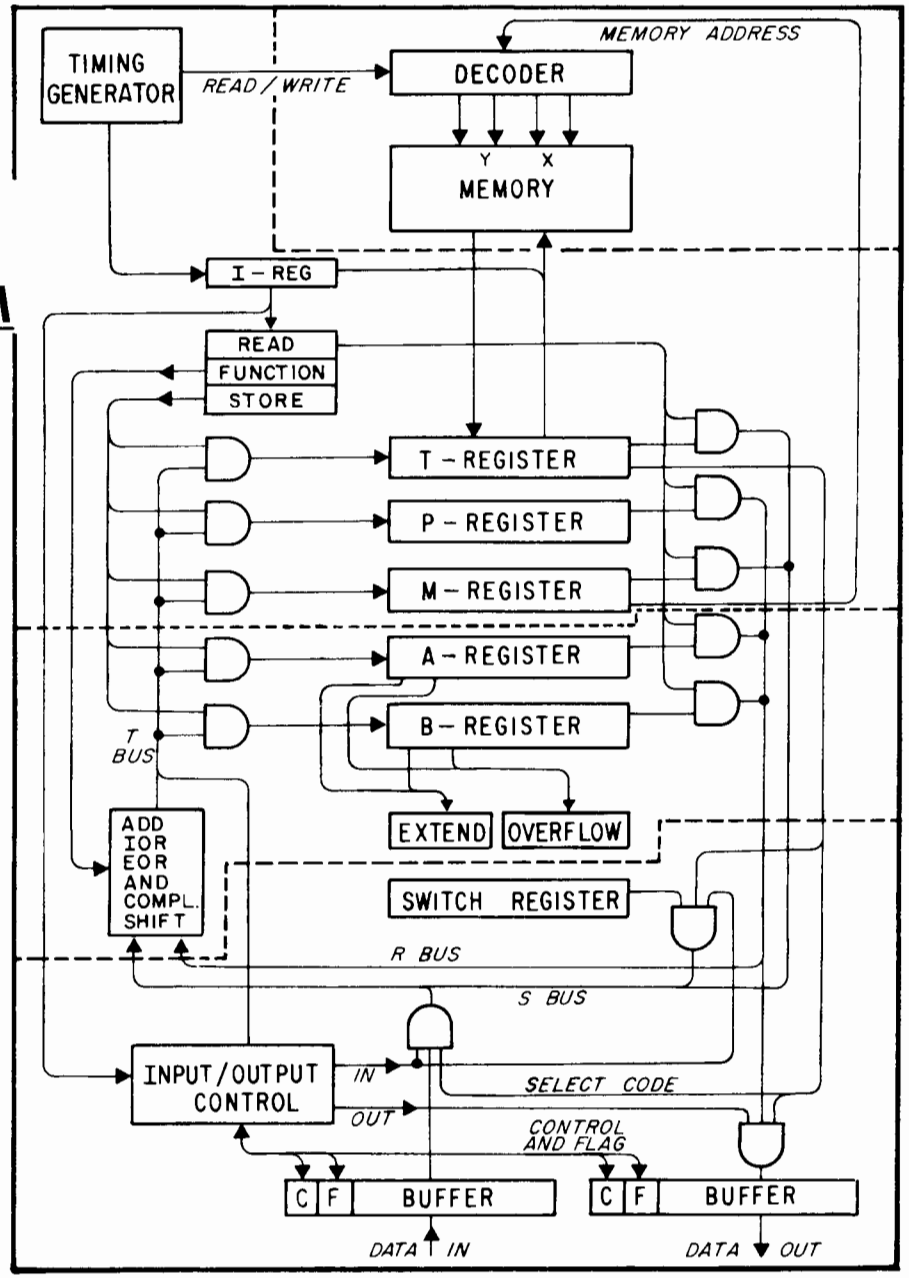
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIGN	INTEGER														

THE COMPUTER –

CANNOT TELL AN INSTRUCTION WORD APART FROM A DATA WORD. THE PROGRAMMER MUST KNOW WHICH LOCATIONS HOLD INSTRUCTIONS AND WHICH LOCATIONS HOLD DATA. THE PROGRAMMER USES ADDRESS WORDS TO SPECIFY A GIVEN MEMORY LOCATION.



HP COMPUTER BLOCK DIAGRAM

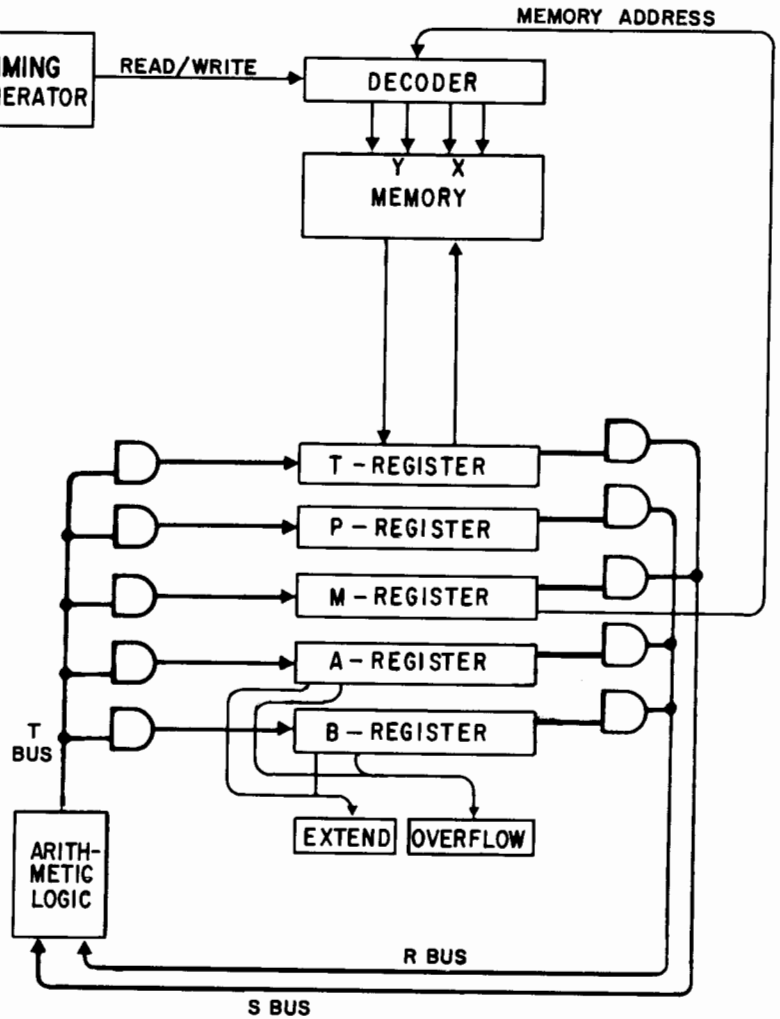


MEMORY UNIT

CONTROL UNIT

ARITHMETIC UNIT

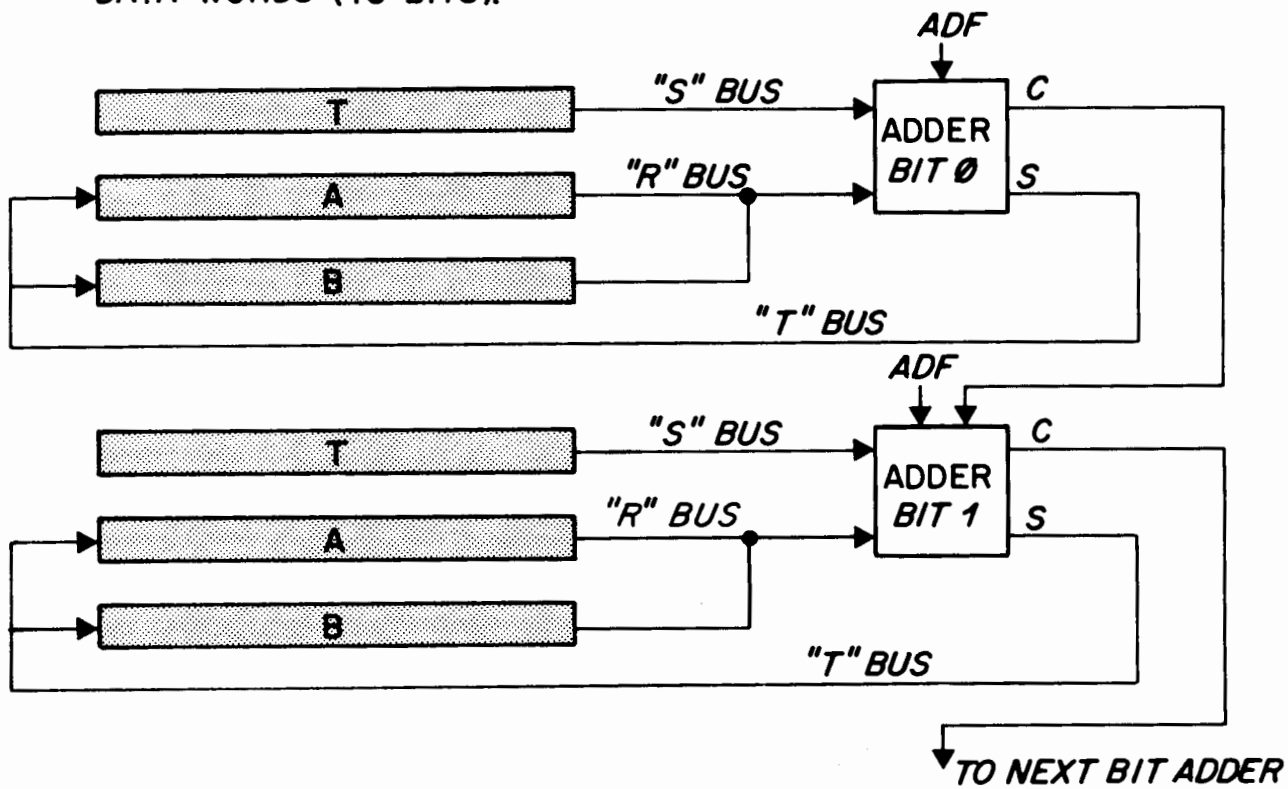
INPUT/OUTPUT UNIT



THE R-S-T BUS SYSTEM IS USED TO TRANSFER DATA AND CONTROL BETWEEN VARIOUS COMPUTER UNITS

APPLICATION OF RST BUS SYSTEM

THE ADDITION HARDWARE. THERE ARE 16 R,S,T BUSES AND 16 ADDERS, ONE BUS AND ONE ADDER FOR EACH BIT IN THE COMPUTER DATA WORDS (16 BITS).



THIS REQUIRES 16 A and B FLIP-FLOPS TO MAKE UP THE A and B ACCUMULATORS.

THE CONTROL UNIT REGISTERS

T - TRANSFER REGISTER - READS AND STORES CONTENTS READ FROM OR WRITTEN INTO MEMORY

I - INSTRUCTION REGISTER - STORES INSTRUCTION CODES FROM T-REGISTER THAT ARE USED TO CONTROL MACHINE OPERATION.

M - MEMORY ADDRESS REGISTER - ADDRESSES A MEMORY LOCATION TO FETCH INSTRUCTION AND/OR DATA

P - PROGRAM COUNTER - HOLDS ADDRESS OF NEXT SEQUENTIAL INSTRUCTION

MEMORY LOCATION	INSTRUCTION	DATA LOCATION
1000	LDA	100
1001	ADA	101
1002	STA	102

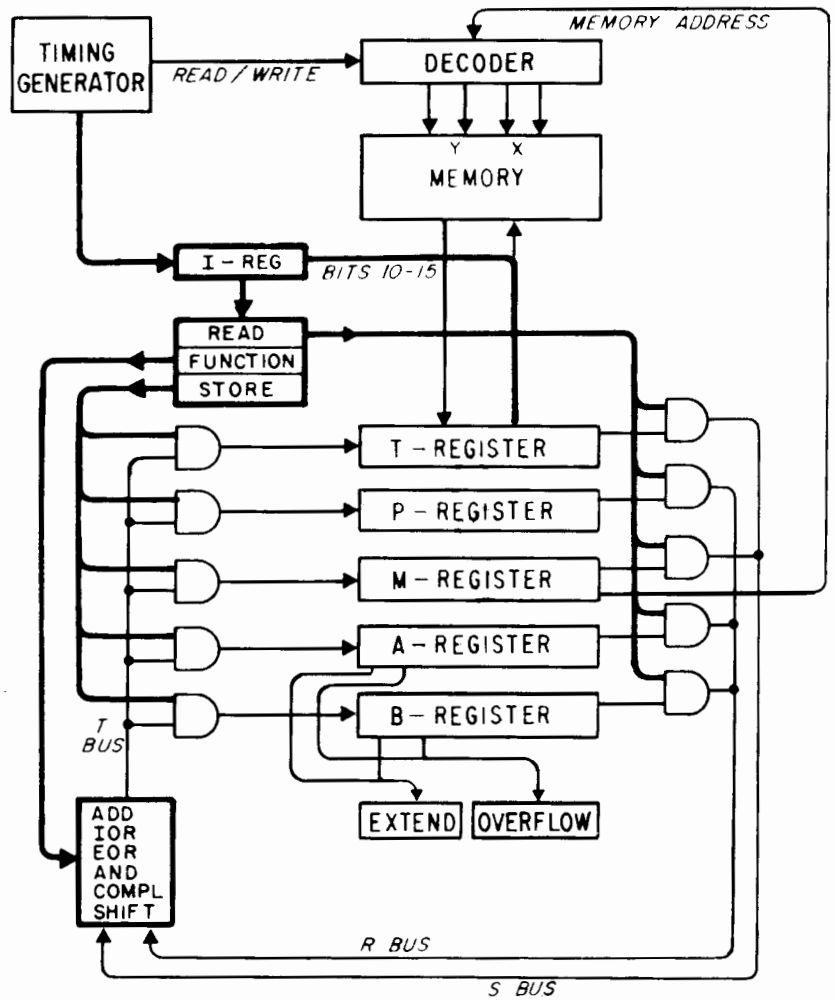
THE ARITHMETIC UNIT

THE ARITHMETIC UNIT CONSISTS OF

1 A AND B REGISTERS (ACCUMULATOR) WHICH ACCUMULATE DATA AND RESULTS OF ARITHMETIC OPERATIONS. THE A REGISTER MAY ALSO PERFORM THE "AND, IOR AND XOR" LOGIC FUNCTIONS.

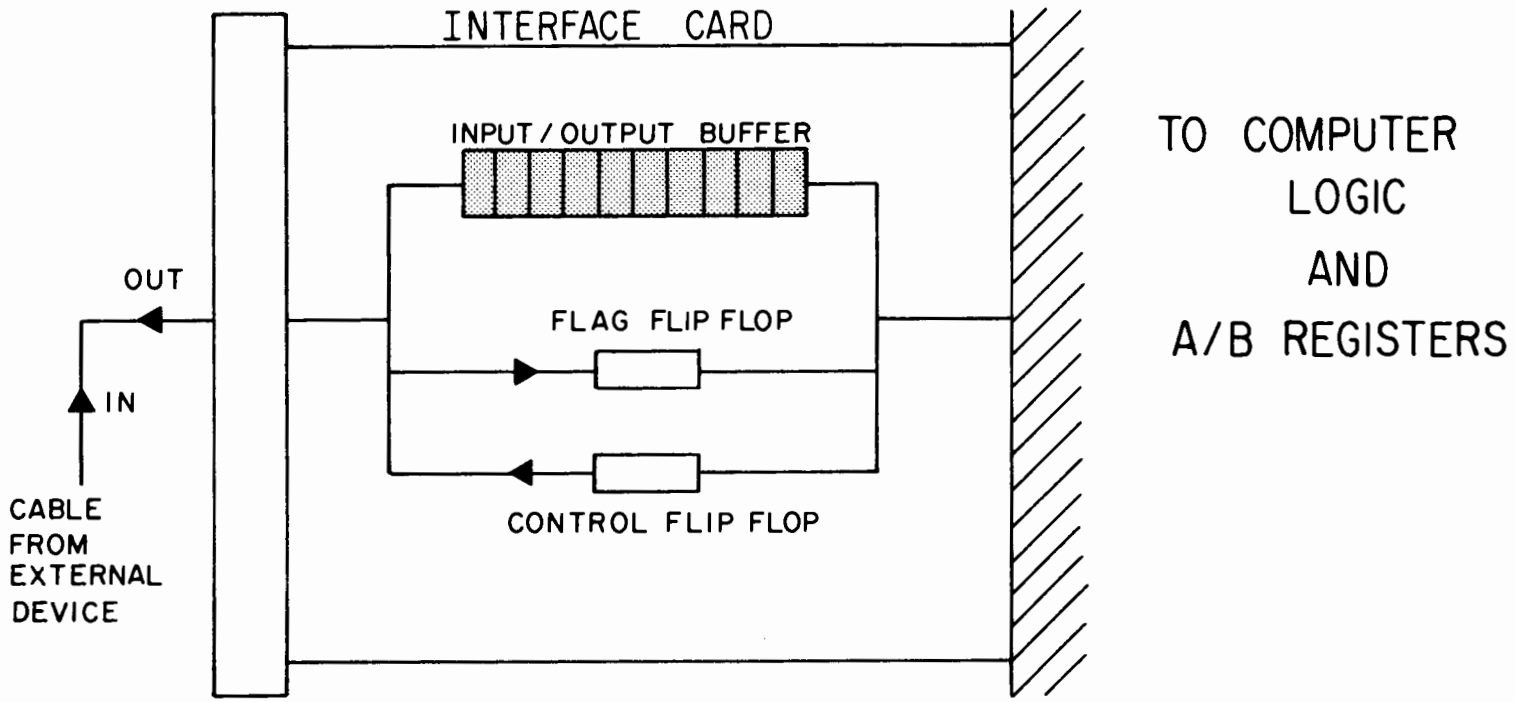
2 LOGIC CIRCUITS WHICH ADD, AND, INCLUSIVE OR, EXCLUSIVE OR, SHIFT, AND COMPLEMENT DATA.

INSTRUCTION LOGIC BLOCK DIAGRAM

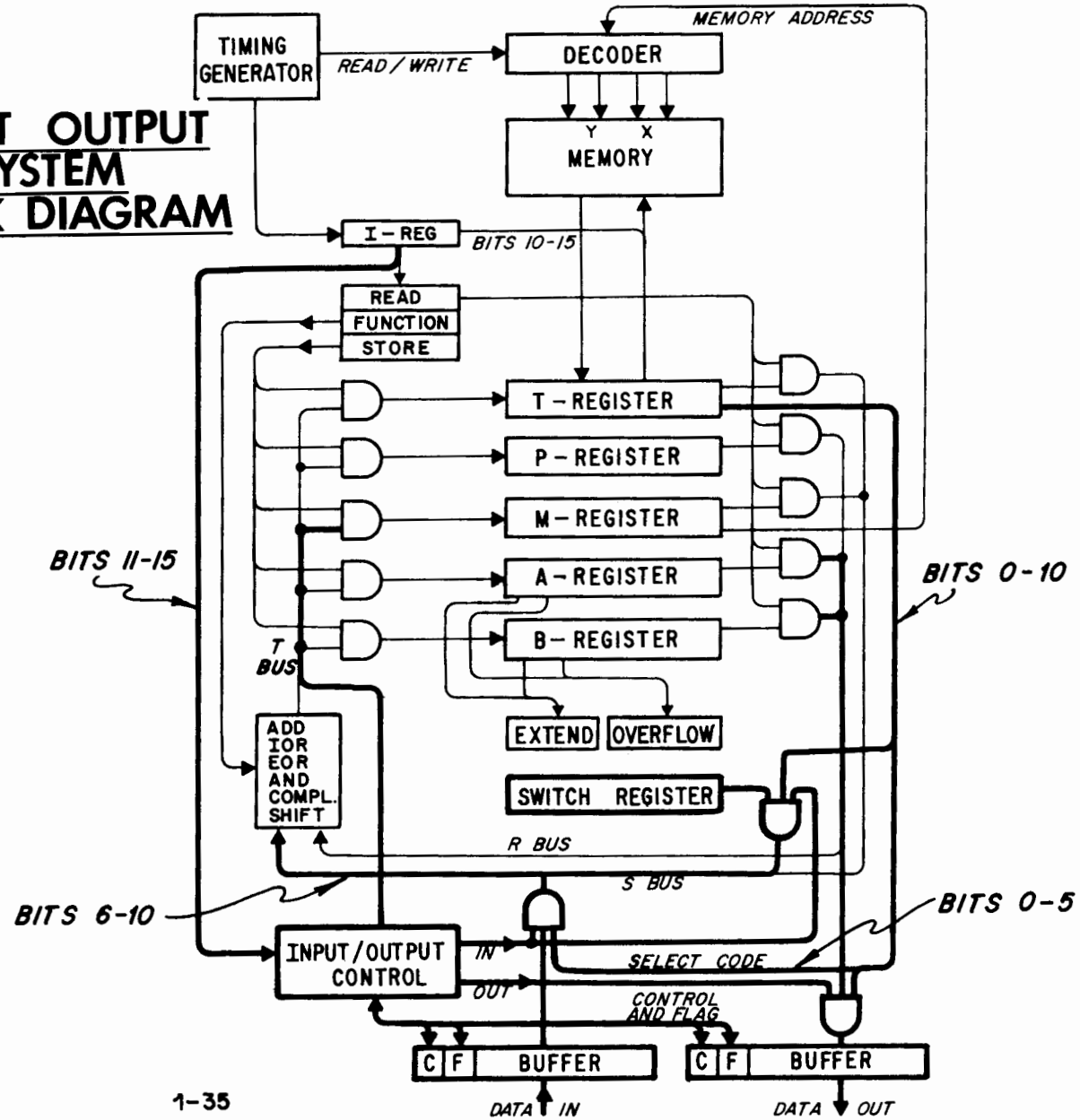


INTERFACE CARDS CONTAIN

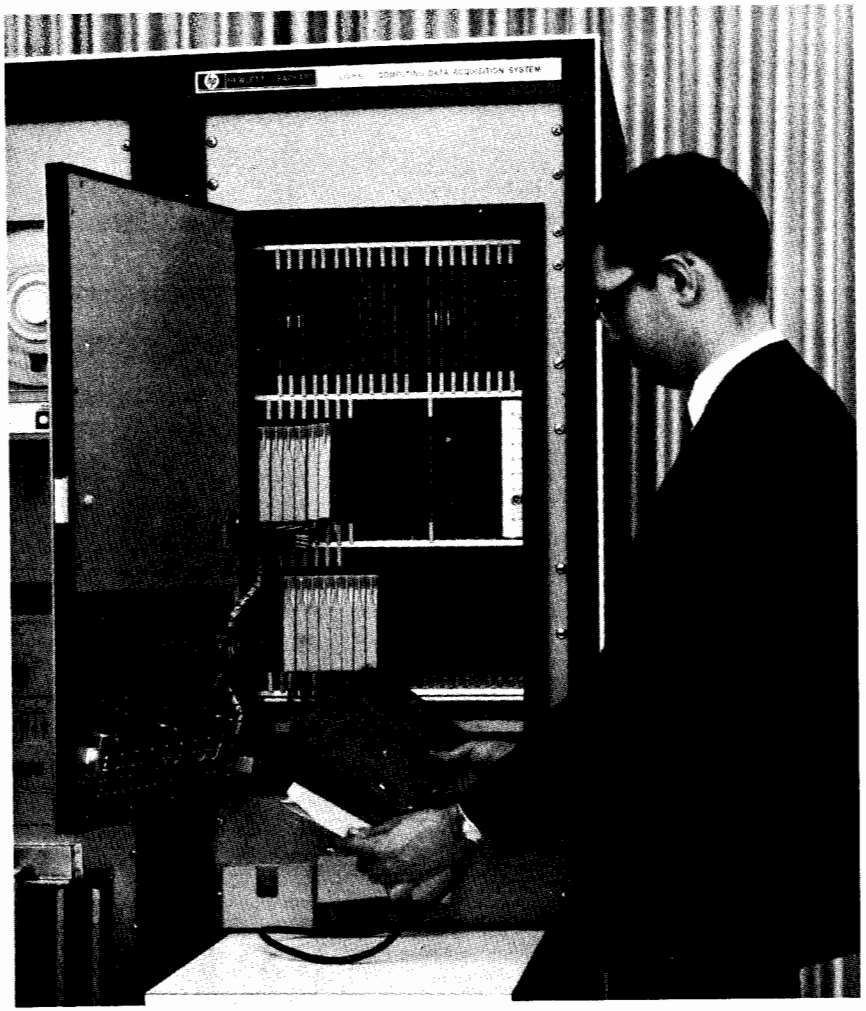
1 *INPUT/OUTPUT BUFFER*
2 *FLAG FLIP FLOP*
3 *CONTROL FLIP FLOP*



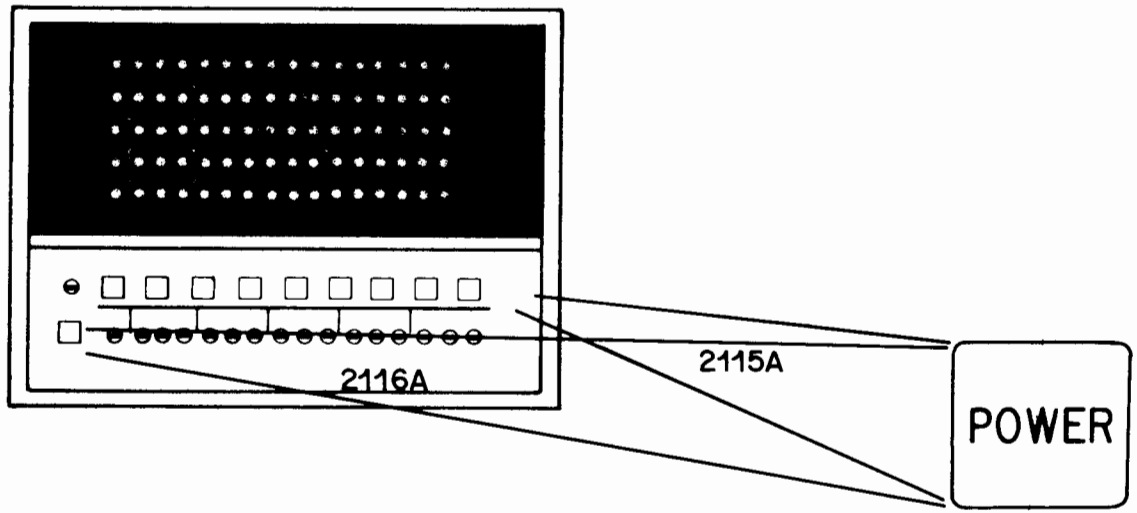
INPUT OUTPUT SYSTEM BLOCK DIAGRAM



HP INTERFACE CARD



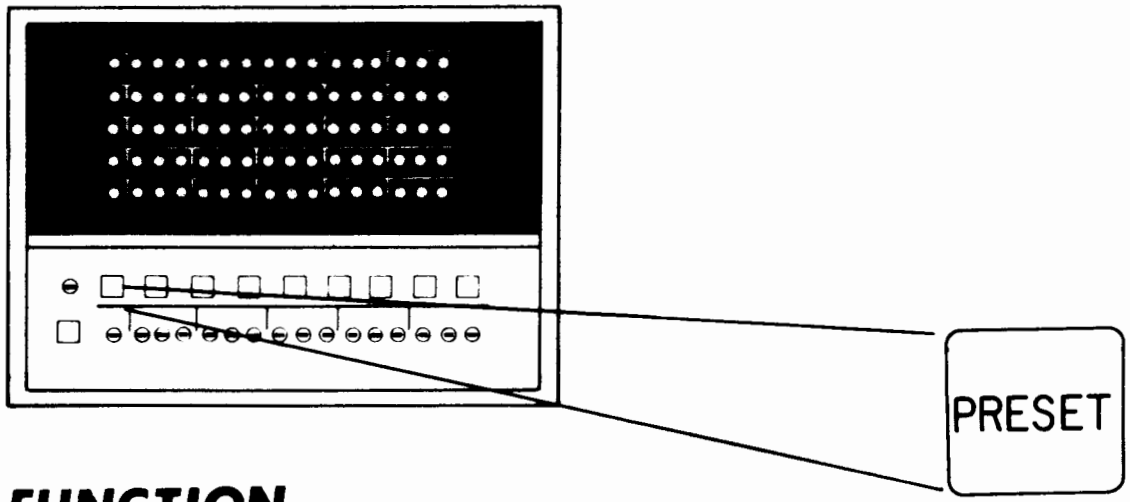
I/O INTERFACE CARDS ARE SIMPLE TO INSTALL OR REARRANGE



FUNCTION

- PUSH-ON PUSH-OFF SWITCH FOR COMPUTER POWER ON-OFF
- CONTENTS OF MEMORY NOT AFFECTED BY SWITCHING POWER OFF AND ON.
- CONTENTS OF WORKING REGISTERS ARE LOST WHEN POWER GOES OFF

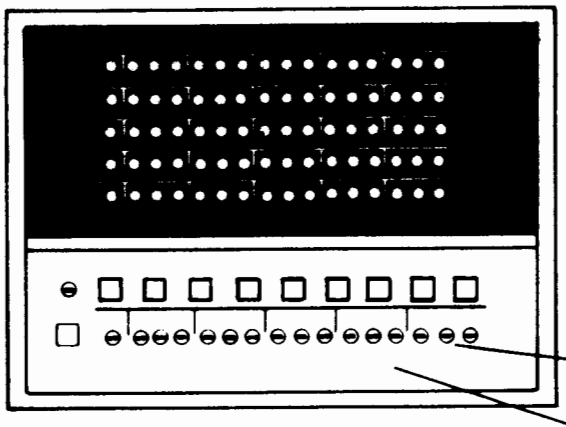
PANEL CONTROLS



FUNCTION

- PRESETS COMPUTER TO FETCH PHASE
- TURNS OFF INTERRUPT SYSTEM
- RESETS ALL INPUT/OUTPUT CONTROL BITS
- SETS ALL INPUT/OUTPUT FLAG BITS
- RESETS PARITY ERROR INDICATION
- INTERNAL PRESET PULSE IS GENERATED, WHEN POWER IS TURNED ON

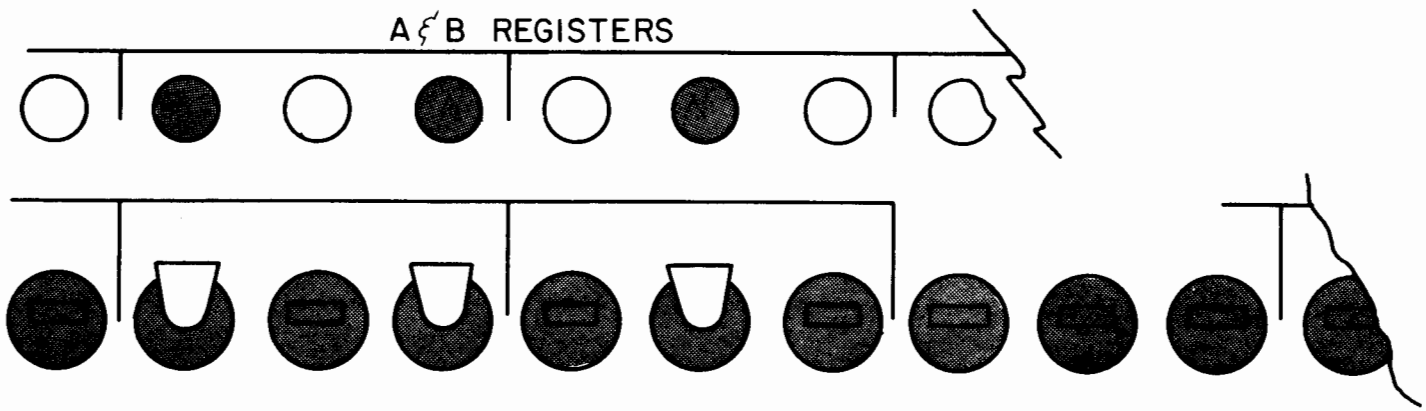
PANEL CONTROLS



FUNCTION

Load switch - register into A or B REGISTERS

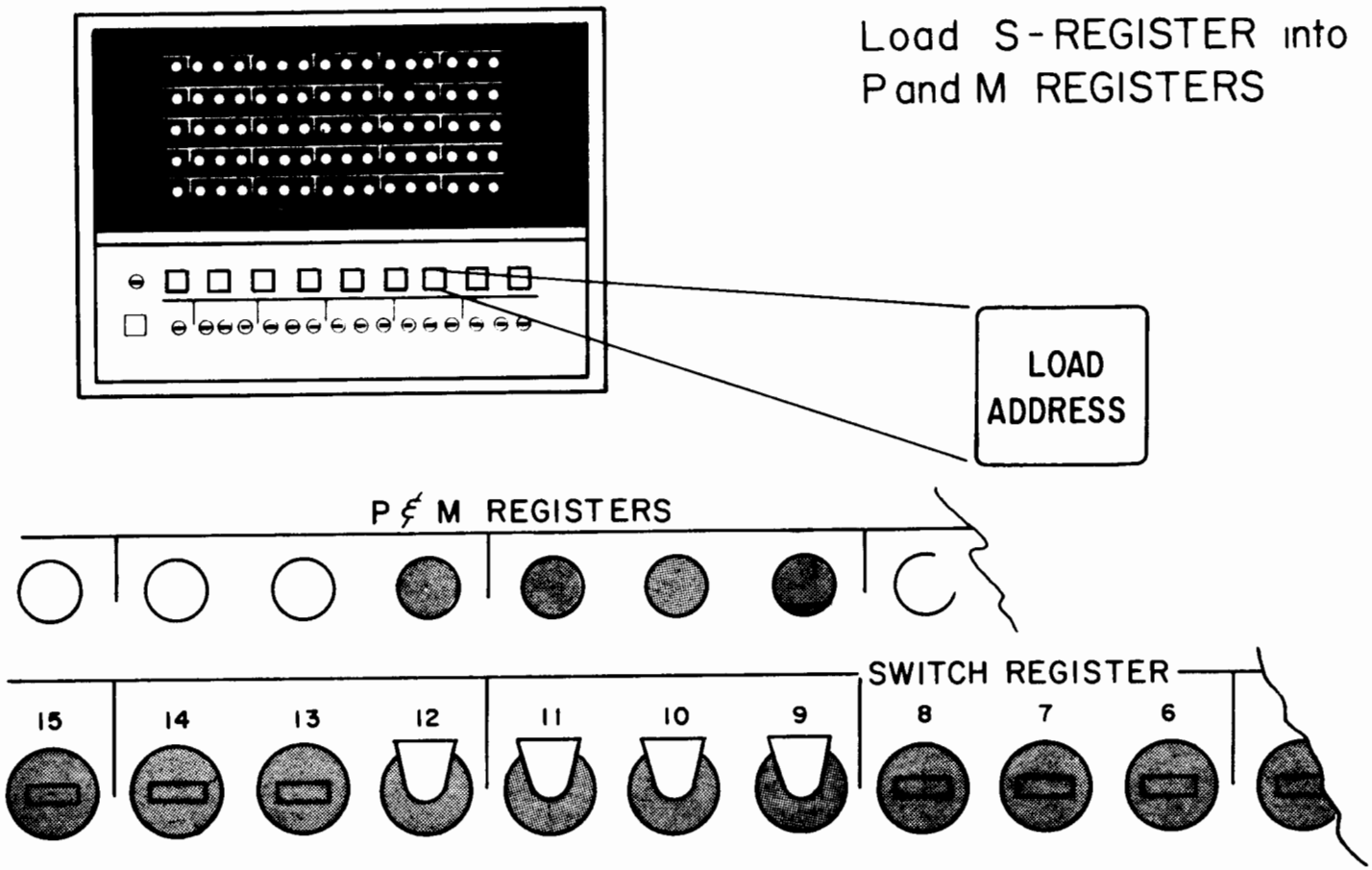
LOAD
A/B



PANEL CONTROLS

FUNCTION

Load S-REGISTER into
P and M REGISTERS



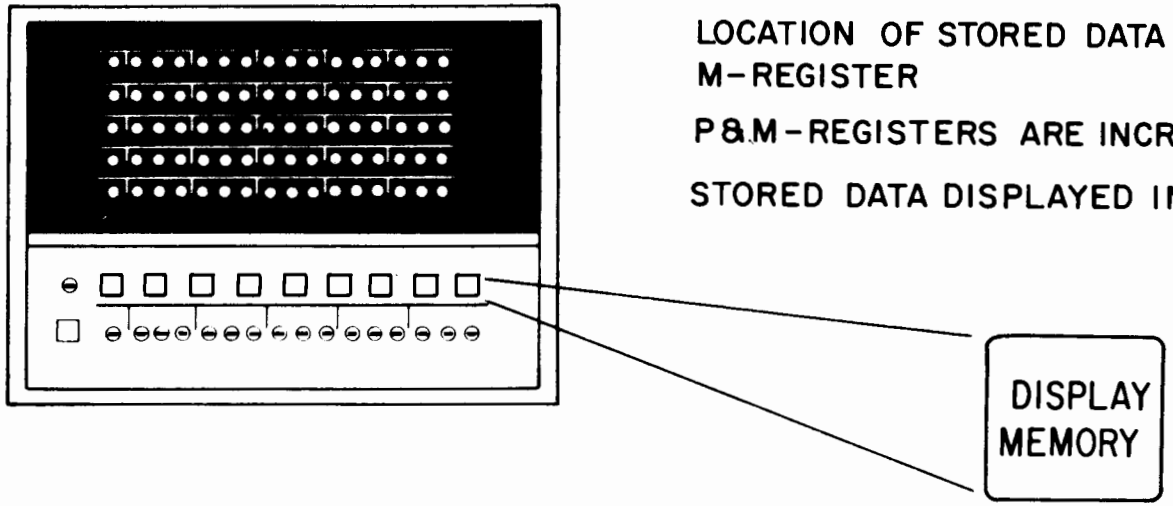
PANEL CONTROLS

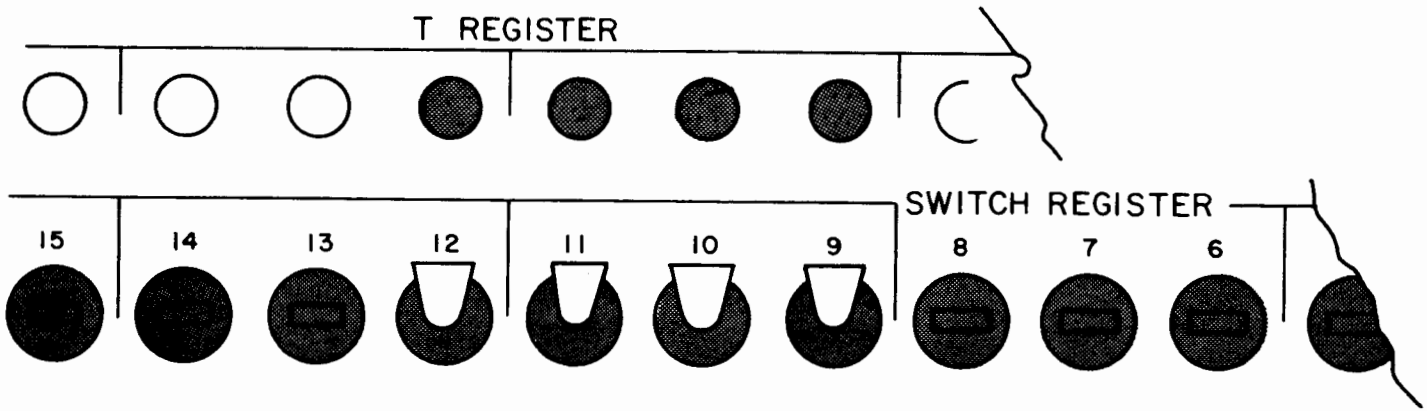
FUNCTION

LOCATION OF STORED DATA SPECIFIED BY M-REGISTER

P&M-REGISTERS ARE INCREMENTED BY ONE

STORED DATA DISPLAYED IN T-REGISTER

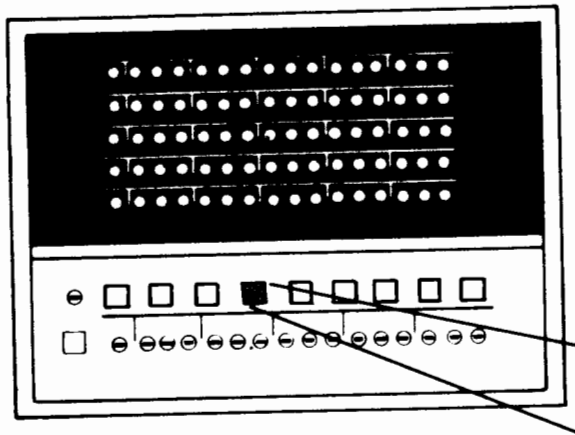




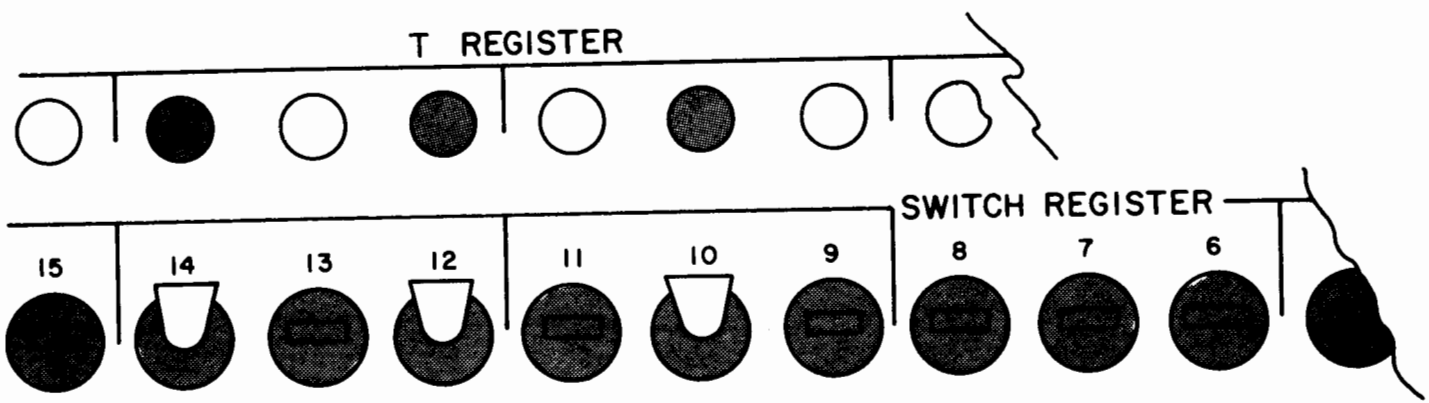
PANEL CONTROLS

FUNCTION

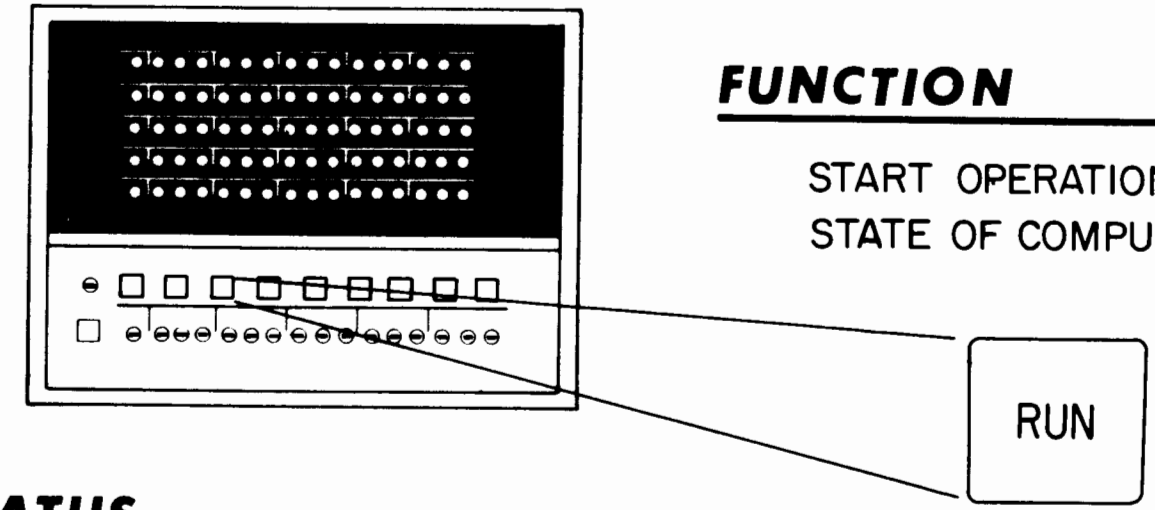
SWITCH REGISTER data loaded into core address indicated by M-REGISTER.
P&M REGISTER are incremented by one
Stored data displayed in T-REGISTER



LOAD MEMORY



PANEL CONTROLS



FUNCTION

START OPERATION AT CURRENT STATE OF COMPUTER.

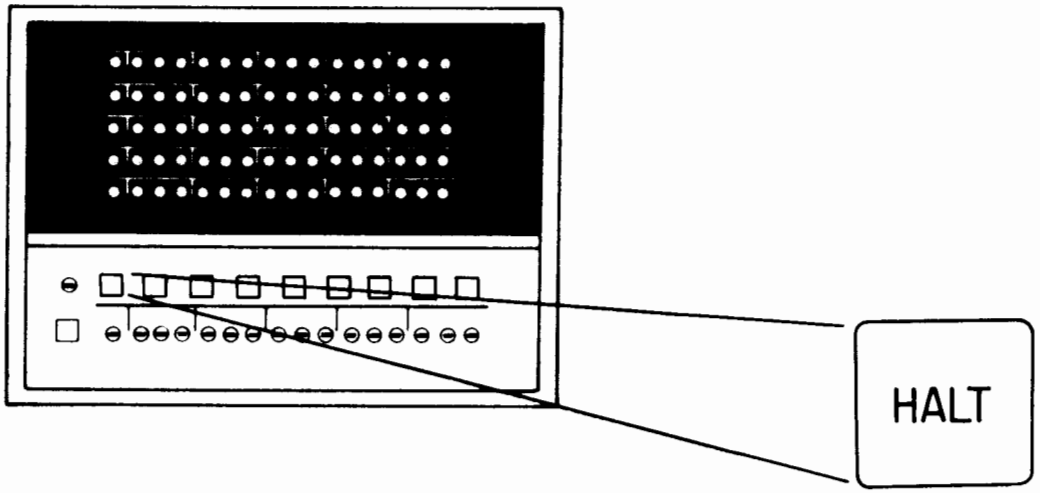
STATUS

LIGHT

ON
- Program is running
- All panel switches (except, Halt, Power, & Loader) disabled

OFF
- Halt is pressed
- HLT instruction is executed
- Parity error occurs (memory parity option)
- Abnormal change of internal power supplies

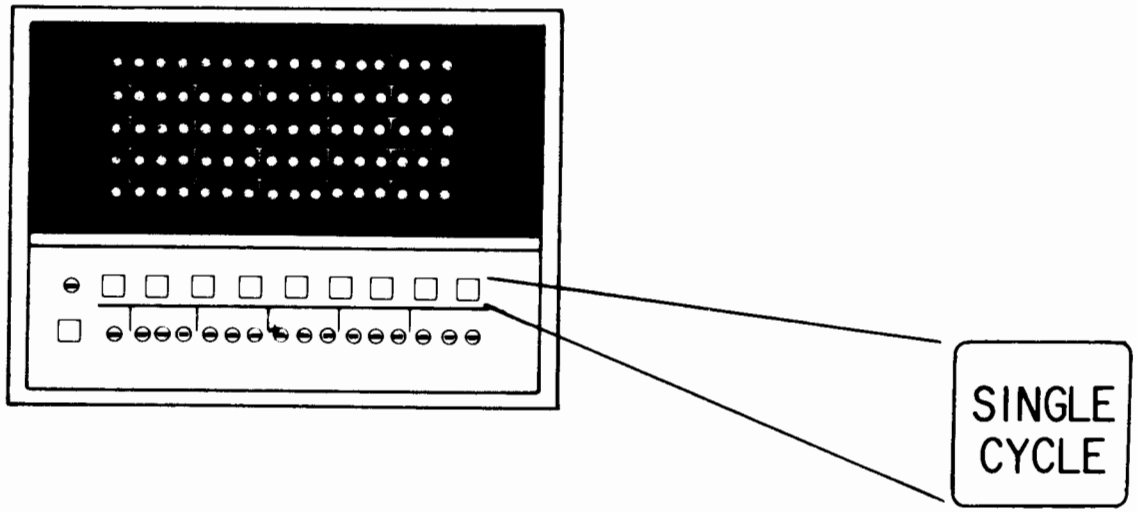
PANEL CONTROLS



FUNCTION

- STOP COMPUTER AT END OF CURRENT PHASE, EXCEPT INTERRUPT
- WHEN HALTED, ALL PANEL CONTROLS ARE ENABLED

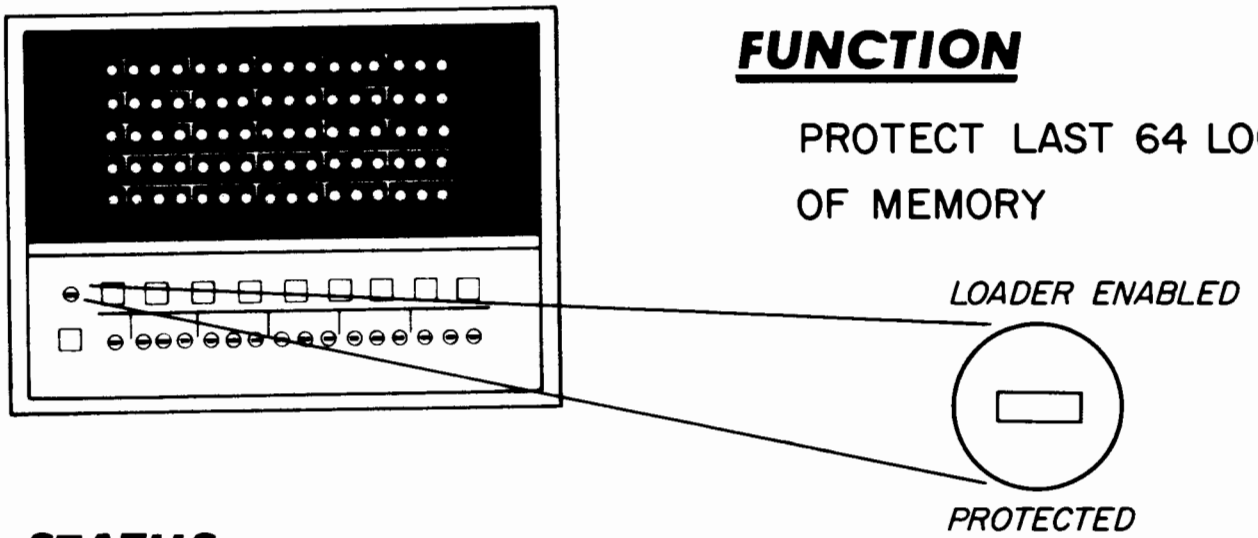
PANEL CONTROLS



FUNCTION

COMPLETES THE INDICATED MACHINE PHASE
EACH TIME THE SWITCH IS DEPRESSED

PANEL CONTROLS



FUNCTION

PROTECT LAST 64 LOCATIONS OF MEMORY

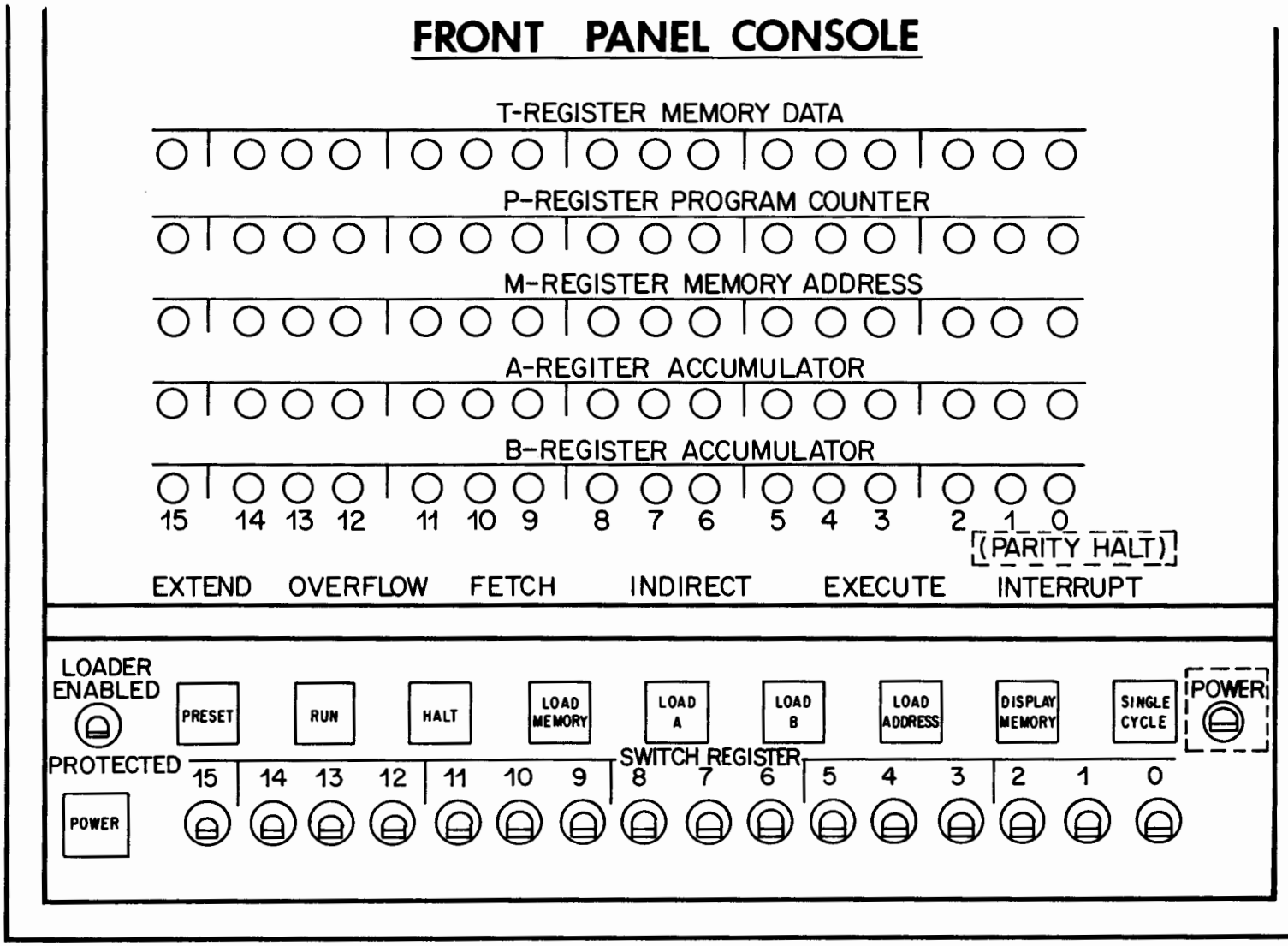
STATUS

SWITCH POSITIONS

- ENABLED-BLOCK OF MEMORY CAN BE READ OR LOADED
- PROTECTED-BLOCK OF MEMORY IS DISABLED

PANEL CONTROLS

FRONT PANEL CONSOLE



INDICATES 2115A FUNCTION

THE PHASES OF MACHINE OPERATION

FETCH → COMPUTER WILL FETCH INSTRUCTION FROM MEMORY.

INDIRECT → COMPUTER WILL OBTAIN A 15 BIT ADDRESS FROM MEMORY.

EXECUTE → COMPUTER WILL OBTAIN A 15 BIT OPERAND FROM MEMORY AND COMPLETE THE REQUIRED COMPUTER INSTRUCTION.

INTERRUPT → INTERRUPT HALTS THE NORMAL PROGRAM SEQUENCE AND FORCES THE COMPUTER TO FETCH ITS NEXT INSTRUCTION FROM ONE OF THE INTERRUPT ADDRESSES.

MACHINE PHASES & TIME PERIODS

THE BASIC MACHINE CYCLE IS 1.6 MICROSECONDS LONG
A MACHINE CYCLE IS FURTHER DEFINED BY THE
PHASE AND TIME PERIOD.

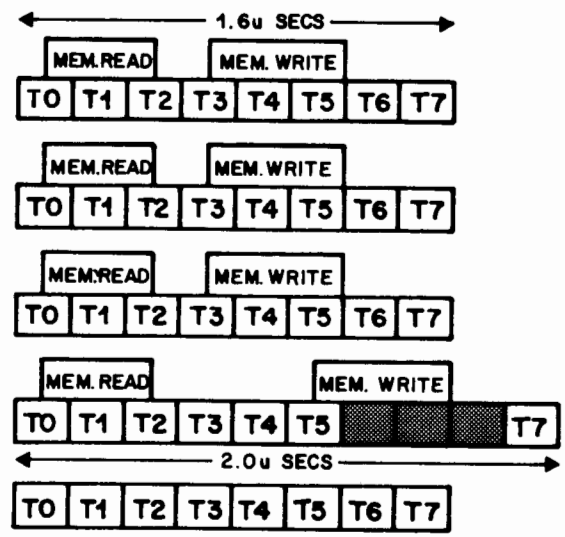
PHASE 1 (FETCH PHASE)

PHASE 2 (INDIRECT PHASE)

PHASE 3 (EXECUTE PHASE) (NOT ISZ)

PHASE 3 (EXECUTE PHASE) (ISZ)

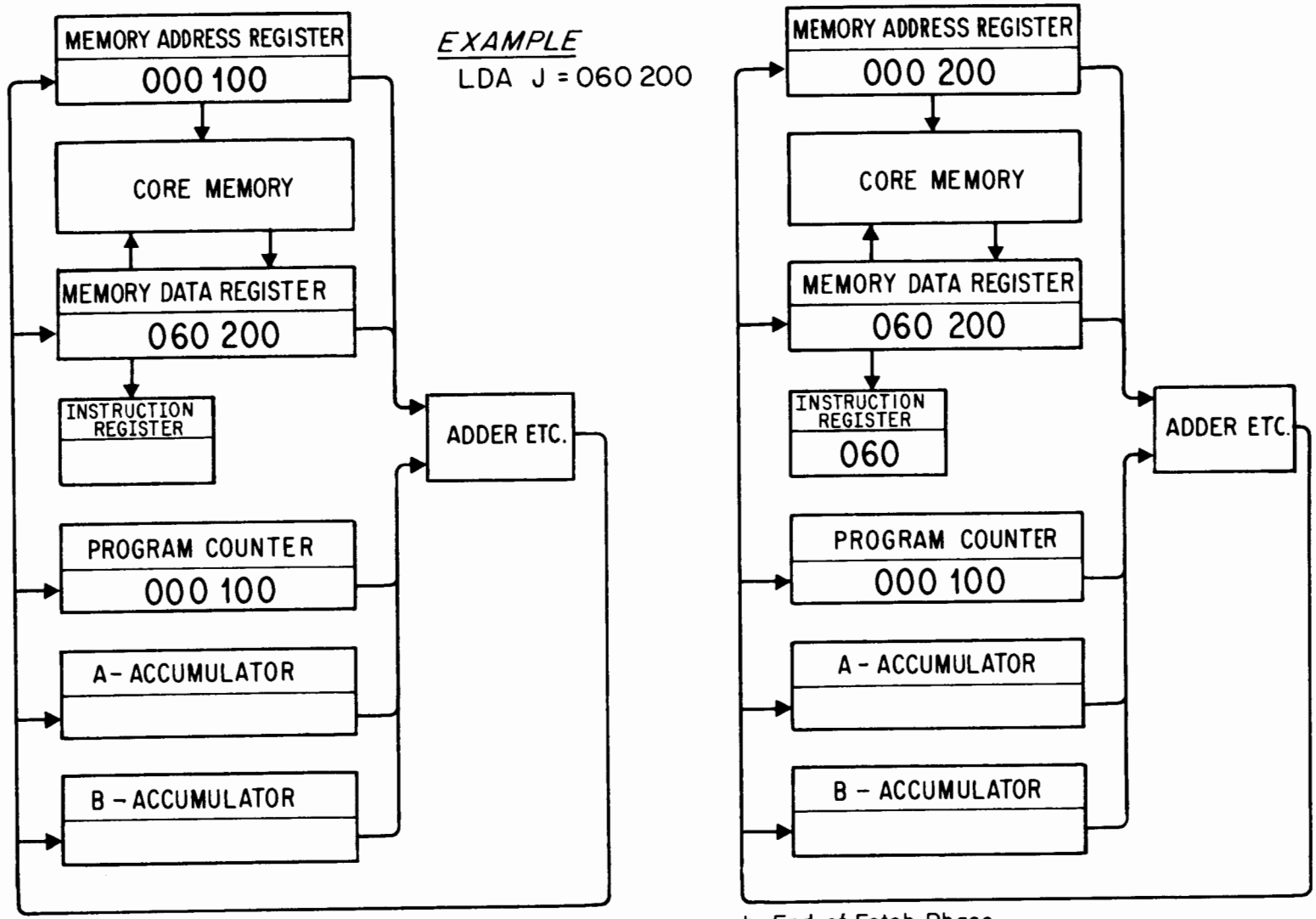
PHASE 4 (INTERRUPT) *



* NO MEMORY CYCLE

THE BASIC COMPUTER WILL ALWAYS BE IN ONE AND
ONLY ONE MACHINE PHASE AT ANY GIVEN TIME.

INSTRUCTION EXECUTION SEQUENCE

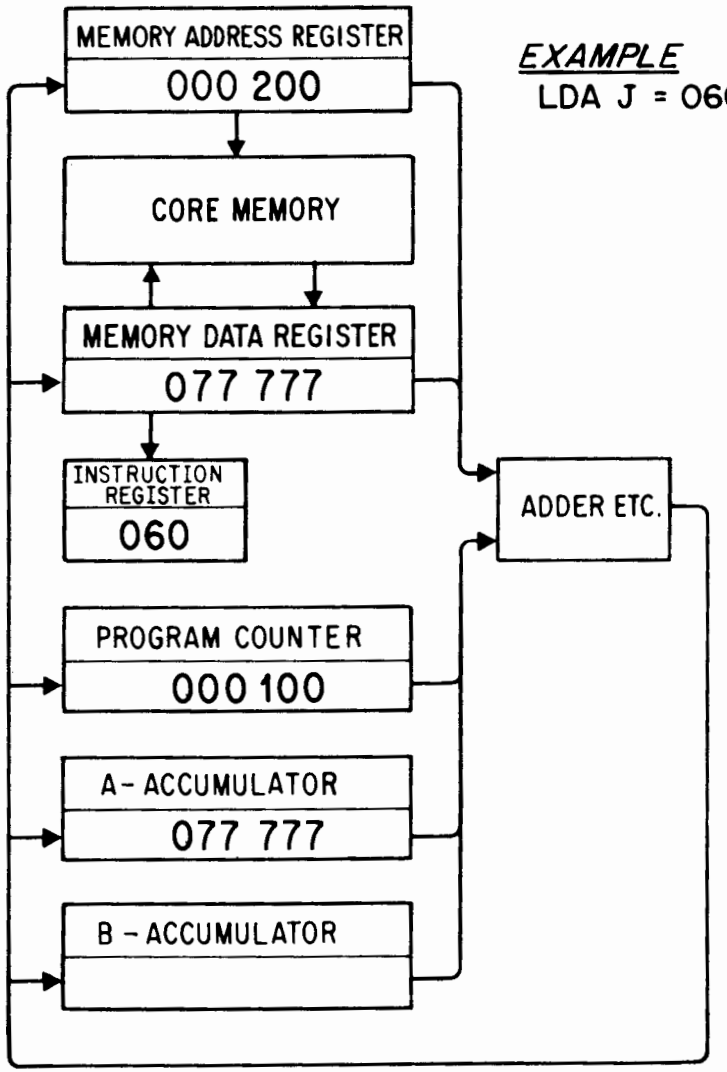


a. Instruction is Read from Memory

b. End of Fetch Phase

INSTRUCTION IN LOCATION 100 - FETCH PHASE

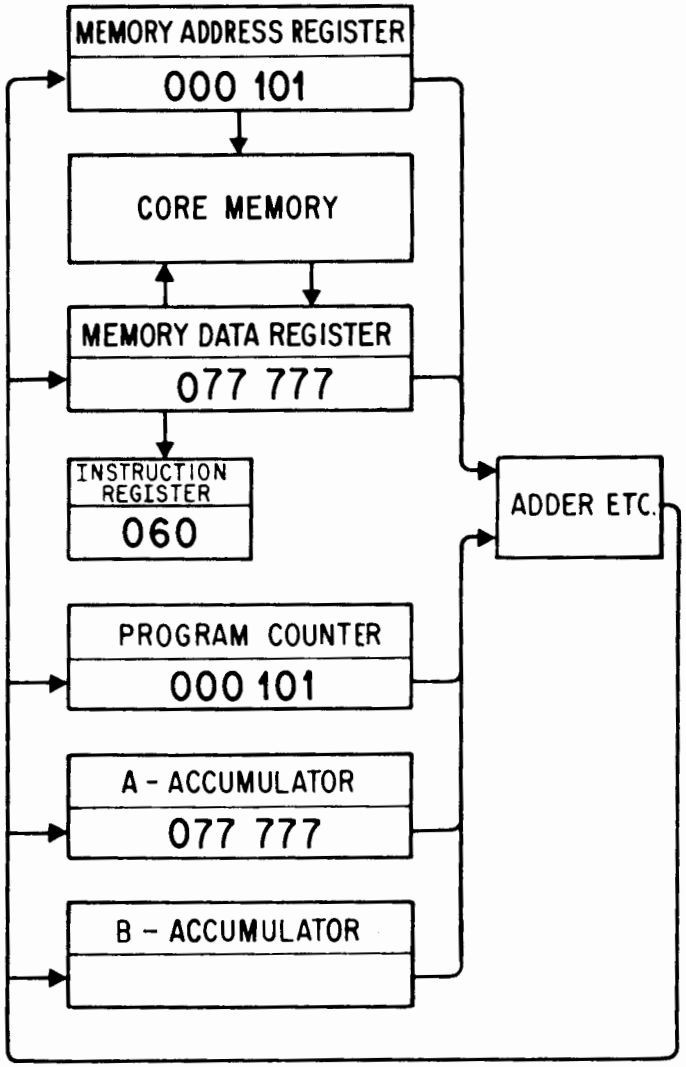
INSTRUCTION EXECUTION Con't



c. Instruction is Executed

EXAMPLE

LDA J = 060200



d. End of Execute Phase

INSTRUCTION IN LOCATION 100 -EXECUTE PHASE

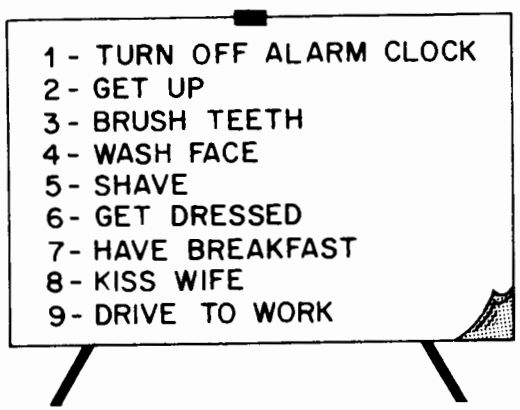
LESSON II OBJECTIVES

I. INTRODUCE THE STUDENT TO THE BASIC ELEMENTS OF MACHINE LANGUAGE PROGRAMMING

II. GIVE THE STUDENT PRACTICE IN THE USE OF MACHINE LANGUAGE INSTRUCTIONS.

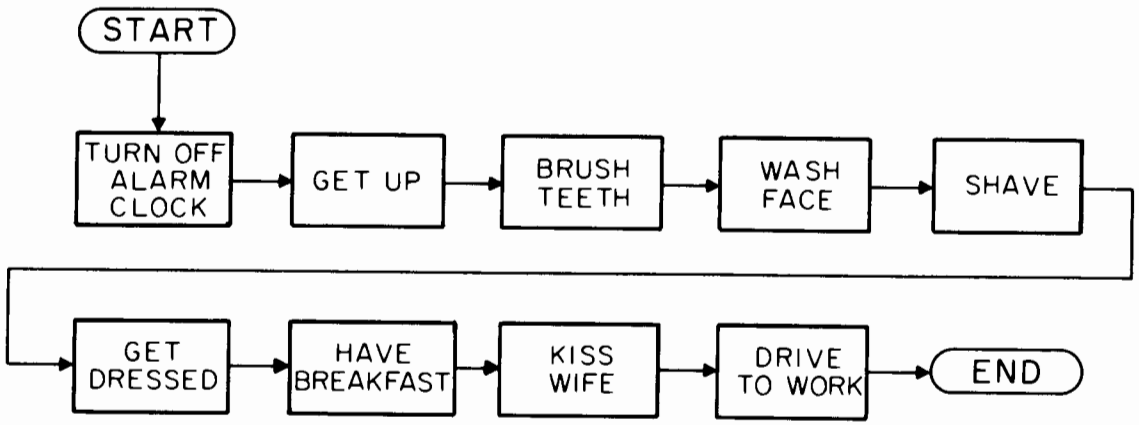
A COMPUTER PROGRAM

FAR FROM BEING A GIANT "BRAIN," A COMPUTER MUST BE TOLD TO DO EVERYTHING. WE TELL COMPUTERS WHAT TO DO BY WRITING PROGRAMS OF INSTRUCTIONS. THE INSTRUCTIONS ARE THEN STORED IN THE COMPUTER'S MEMORY. BY REFERRING TO ITS MEMORY, EACH INSTRUCTION IS EXECUTED IN SEQUENCE UNTIL THE "PROGRAM" IS COMPLETED.

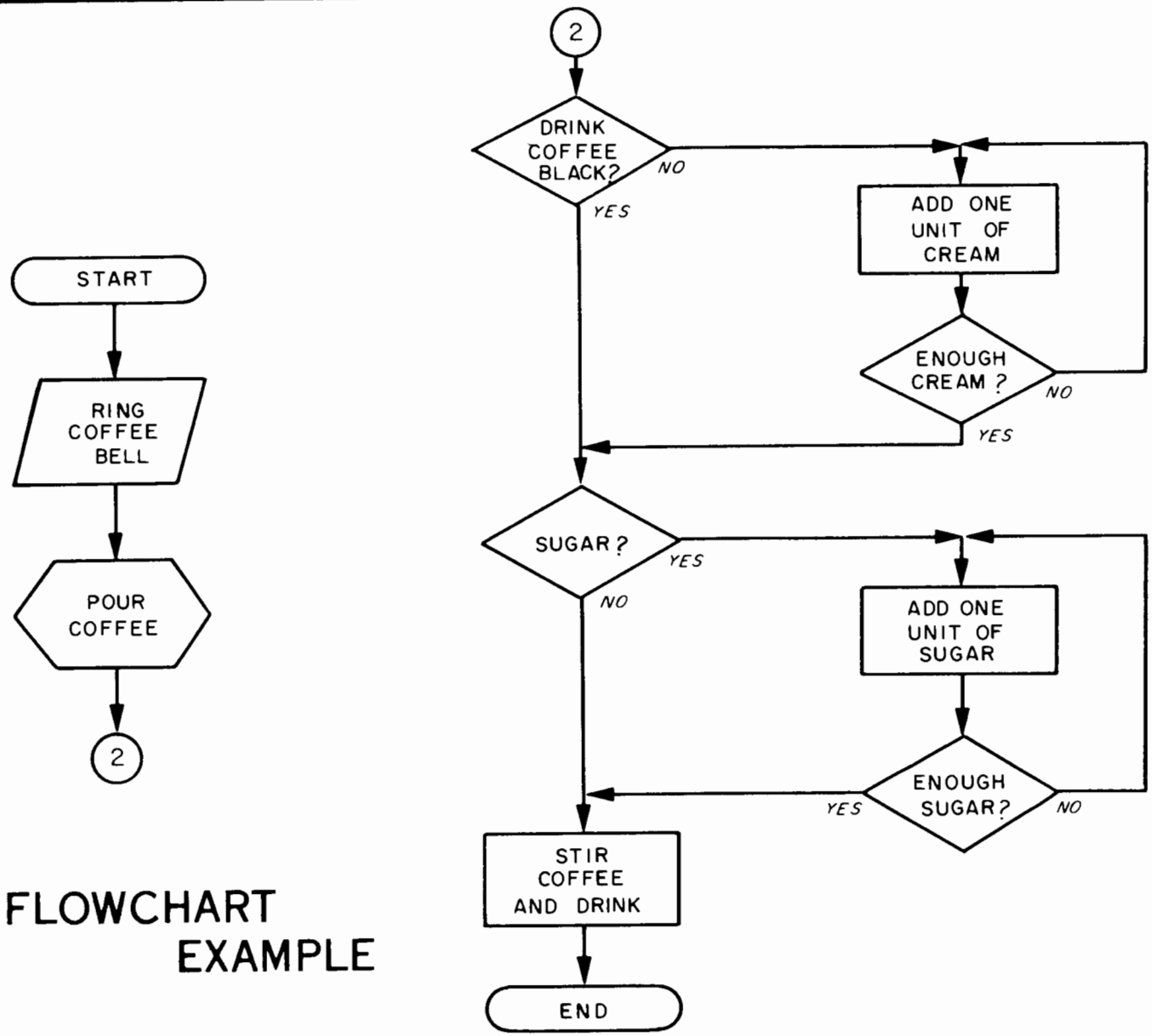


INTRODUCTION TO FLOWCHARTING

THE MOST DIFFICULT PART OF COMPUTER PROGRAMMING IS DEFINING EXACTLY WHAT IS TO BE DONE. FLOWCHARTING HELPS TO REMOVE AMBIGUITY BEFORE THE PROGRAM IS WRITTEN.



FLOWCHARTS CAN BE MADE MORE MEANINGFUL IF STANDARD SYMBOLS AND TECHNIQUES ARE USED.



FLOWCHART EXAMPLE

MACHINE LANGUAGE PROGRAMMING OPERATIONS

THE PROBLEM OF TURNING A FLOWCHART PROBLEM SOLUTION INTO A COMPUTER PROGRAM IS REFERRED TO AS "CODING".

PROBLEM:

"CODING" DIRECTLY IN MACHINE LANGUAGE IS VERY DIFFICULT.

THE ANSWER:

CODE THE PROGRAM IN MNEMONIC SYMBOLS AND, AFTER THE COMPLETE SOLUTION IS "CODED", TRANSLATE THE MNEMONICS INTO MACHINE LANGUAGE (BINARY)

THE MNEMONICS WE WILL USE ARE THOSE THAT CAN BE TRANSLATED BY THE ASSEMBLER SOFTWARE PROGRAM.

THE CODING FORM

HEWLETT-PACKARD MACHINE CODING FORM

PROGRAMMER DATE

PROGRAM PAGE _____ OF _____

LABEL		OP CODE		OPERAND	REMARKS	LOCATION		CONTENTS

ONE INSTRUCTION IS WRITTEN PER LINE.
LOCATION REFERS TO MEMORY ADDRESS.
CONTENTS REFERS TO MEMORY CONTENTS.

MNEMONIC CODES

THE MACHINE INSTRUCTION TO CLEAR THE "A" REGISTER IS

0 000 010 100 000 000

0 0 2 4 0 0

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

SYMBOLIC LABELS

THE TERM LABEL APPEARS ON THE CODING FORM AS:

LABEL	OP CODE	OPERAND

LABEL - A SYMBOLIC LABEL IS AN ALPHANUMERIC SYMBOL, OF 5 CHARACTERS OR LESS, THAT IS USED TO REPRESENT THE MEMORY ADDRESS OF THE INSTRUCTION OR DATA THAT APPEARS ON THE SAME LINE ON THE CODING FORM.

EXAMPLE:

LABEL	OP CODE	OPERAND
	LDA	SAM
	•	
SAM	OCT	7

OP CODE

THE TERM OP CODE APPEARS ON THE CODING FORM AS:

LABEL	OP CODE	OPERAND

OP CODE - THE OP CODE REFERS TO THE MNEMONIC CODE ASSIGNED TO EACH COMPUTER INSTRUCTION. ALL MNEMONIC CODES HAVE 3 LETTERS. THE MNEMONIC CODE "OCT" REFERS TO AN OCTAL CONSTANT.

EXAMPLE:

LABEL	OP CODE	OPERAND
	CLA	
	•	
	OCT	7

SYMBOLIC OPERANDS

THE TERM OPERAND APPEARS ON THE CODING FORM AS:

LABEL	OP CODE	OPERAND

OPERAND – A SYMBOLIC TERM THAT DEFINES THE ADDRESS OF A MEMORY LOCATION, OR A NUMERIC TERM THAT DEFINES A MEMORY CONSTANT VALUE.

EXAMPLE:

LABEL	OP CODE	OPERAND
	LDA	SAM
	•	
SAM	OCT	7

SELECTED INSTRUCTION SHEET

MEMORY REFERENCE INSTRUCTIONS

MNEMONIC	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	D/I	OP-CODE			A/B	Z/C	WORD ADDRESS									
AND	*	0	0	1	0	*	X	X	X	X	X	X	X	X	X	X
JSB	*	0	0	1	1	*										
JMP	*	0	1	0	1	*										
ISZ	*	0	1	1	1	*										
ADA	*	1	0	0	0	*										
ADB	*	1	0	0	1	*										
CPA	*	1	0	1	0	*										
CPB	*	1	0	1	1	*										
LDA	*	1	1	0	0	*										
LDB	*	1	1	0	1	*										
STA	*	1	1	1	0	*										
STB	*	1	1	1	1	*	X	X	X	X	X	X	X	X	X	X

NOTE: D/I, A/B Z/C ARE CODED 0/1

ALTER-SKIP INSTRUCTIONS

CLA	(002400)	CLB	(006400)
CMA	(003000)	CMB	(007000)
INA	(002004)	INB	(006004)
SSA	(002020)	SSB	(006020)
SZA	(002002)	SZB	(006002)
SLA	(002010)	SLB	(006010)

SHIFT-ROTATE INSTRUCTIONS

ARS	(001100)	BRS	(005100)
ALS	(001000)	BLS	(005000)
RAL	(001200)	RBL	(005200)
RAR	(001300)	RBR	(005300)
ERA	(001500)	ERB	(005500)
ELA	(001600)	ELB	(005600)
ALF	(001700)	BLF	(005700)

INPUT-OUTPUT INSTRUCTIONS

MIA	(1024XX)	MIB	(1064XX)
LIA	(1025XX)	LIB	(1065XX)
OTA	(1026XX)	OTB	(1066XX)
HLT	(1020XX)	CLF	(1031XX)
STC	(1027XX)	STC,C	(1037XX)
SFS	(1023XX)	STF	(1021XX)

NOTE: XX DENOTES OCTAL SELECT CODE.

THE LOAD INSTRUCTION

MEMORY

0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	BEFORE

A OR B REGISTER

INSTRUCTION

[LDA/B Y]

LOAD THE SPECIFIED REGISTER WITH THE CONTENTS OF MEMORY LOCATION Y; WHERE Y IS ANY MEMORY LOCATION. THE CONTENTS OF LOCATION Y ARE NOT CHANGED. THE PREVIOUS CONTENTS OF THE SPECIFIED REGISTER ARE LOST.

MEMORY

0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	AFTER

A OR B REGISTER

THE CODING FORM

HEWLETT-PACKARD MACHINE CODING FORM

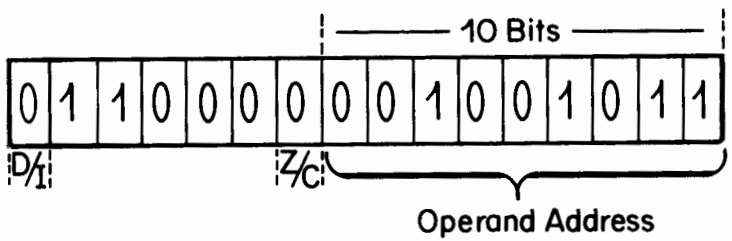
PROGRAMMER JOHN DOE DATE 1-2-68

PROGRAM DEMO # 1 PAGE 1 OF 1

LABEL	OP CODE	OPERAND	REMARKS	LOCATION	CONTENTS
	•			•	•
	•			•	•
	•			•	•
	L.D.A	Y		1.Ø.3	Ø.6.Ø.1.1.3
	•			•	•
	•			•	•
	•			•	•
	•			•	•
	•			•	•
	•			•	•
	•			•	•
Y	O.C.T	7		1.1.3	Ø.Ø.Ø.Ø.Ø.7

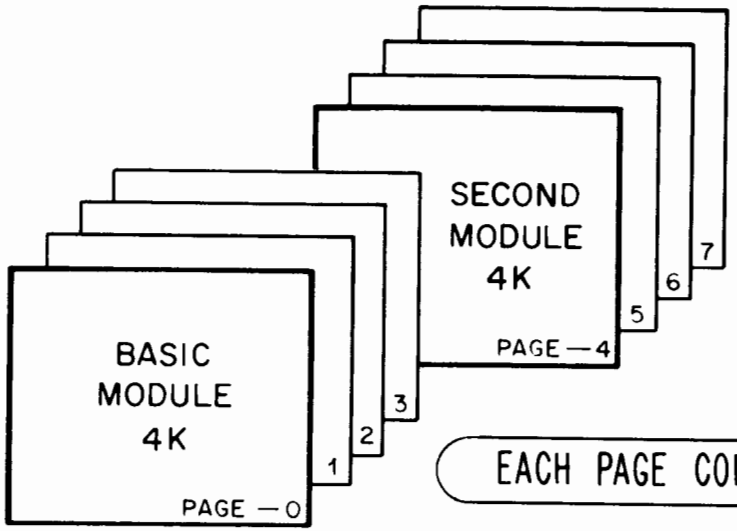
MEMORY REFERENCE INSTRUCTION

OP CODE	OPERAND	LOCATION	CONTENTS
LDA	Y	1Ø3	060113



SINCE ONLY 10 BITS ARE RESERVED FOR THE OPERAND ADDRESS, WE CAN ONLY DIRECTLY ACCESS 1024_{10} WORDS $\left[2^{10} = 1024_{10}\right]$

MEMORY ADDRESSES (8K)



EACH PAGE CONTAINS 1024_{10} LOCATIONS

MODULE	PAGE	OCTAL	
		FROM	TO
BASIC	0	0	01777
"	1	02000	03777
"	2	04000	05777
"	3	06000	07777
SECOND	4	10000	11777
"	5	12000	13777
"	6	14000	15777
"	7	16000	17777

MEMORY ADDRESS REGISTER

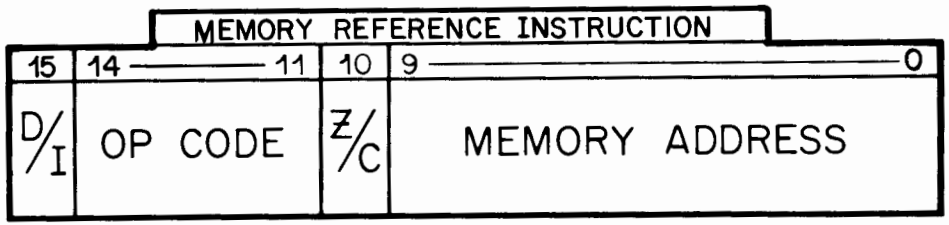
MEMORY

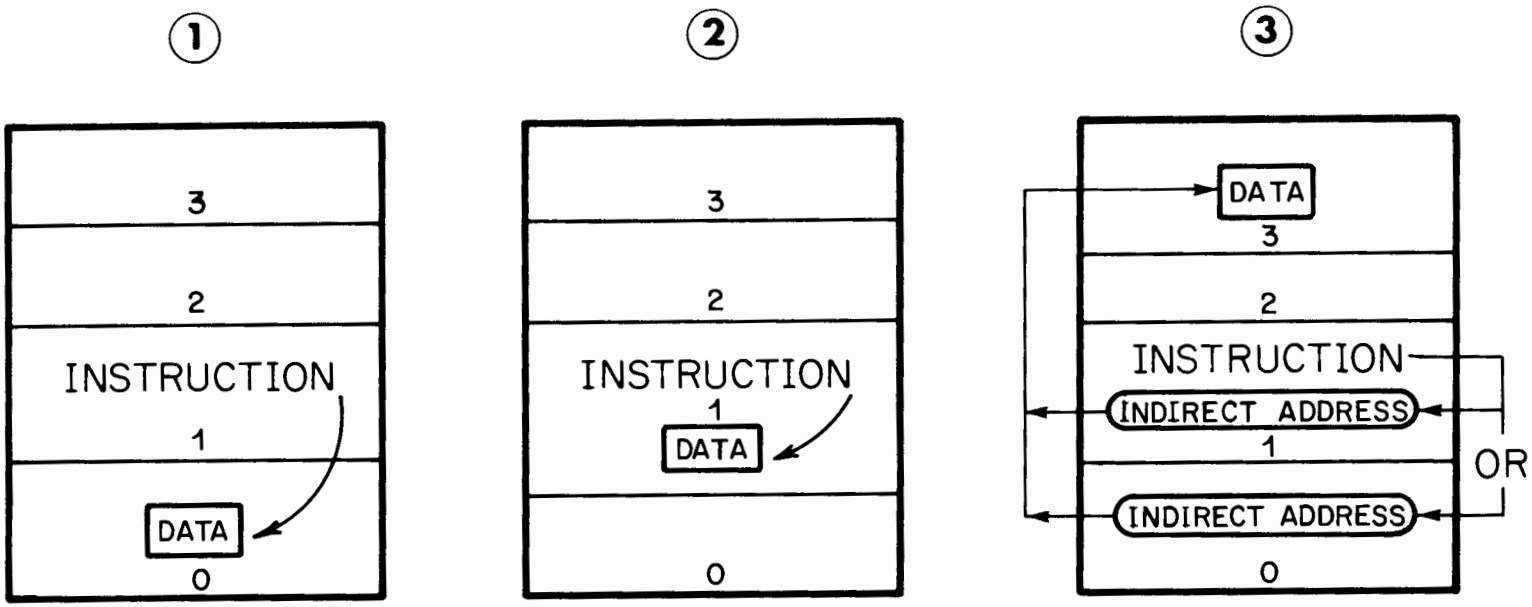
6000 – 7777
4000 – 5777
2000 – 3777
0000 – 1777

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D/I	PAGE ADDRESS					WORD ADDRESS									

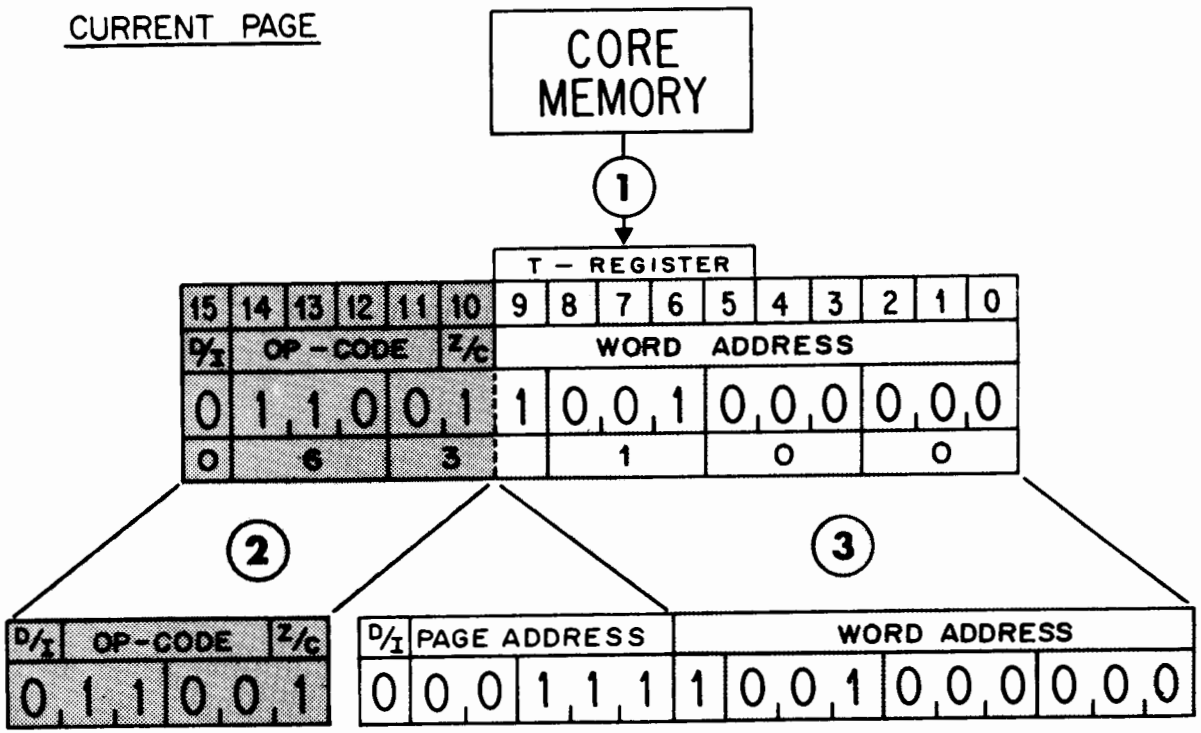
3	0000 – 1777
2	0000 – 1777
1	0000 – 1777
0	0000 – 1777

MEMORY ADDRESSING MODES





MEMORY REFERENCE INSTRUCTION DECODING



(I)NSTRUCTION REGISTER **(M)EMORY ADDRESS REGISTER**

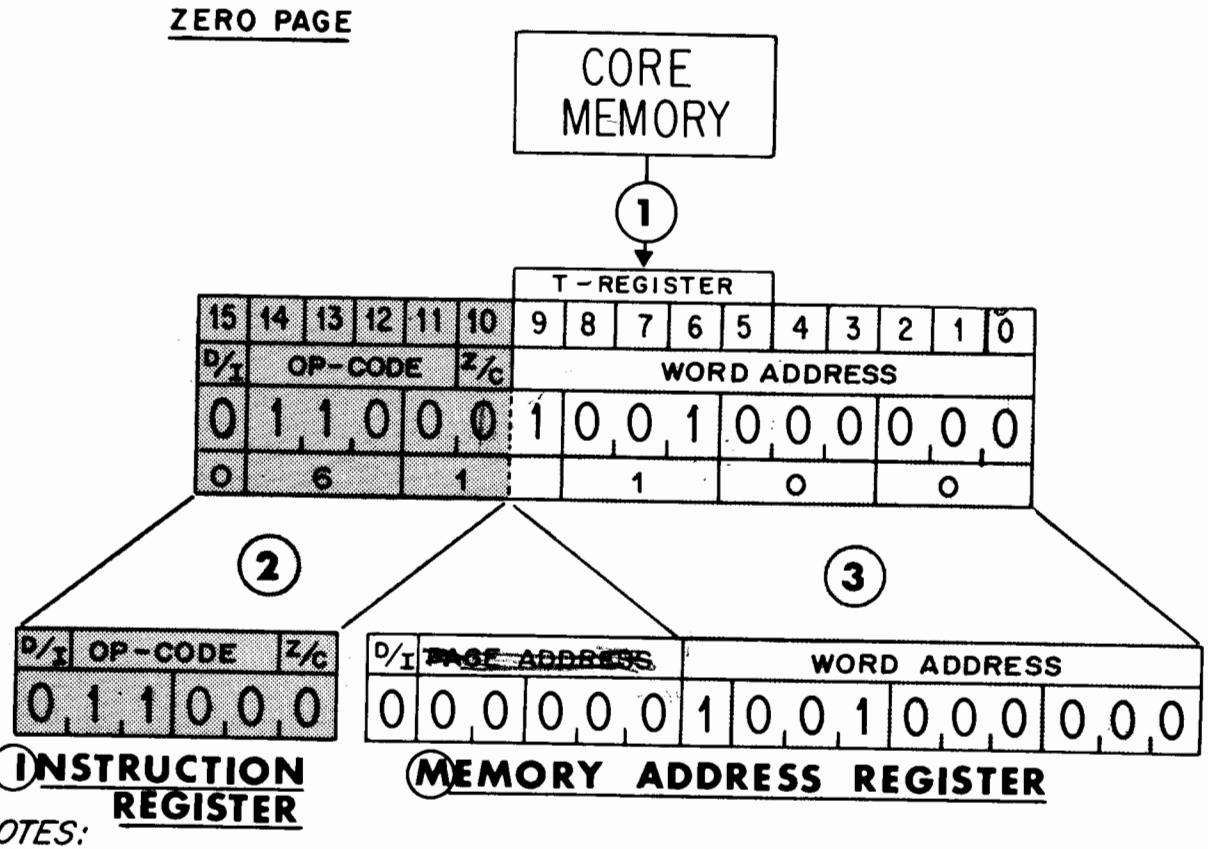
NOTES:

(1) Instruction transferred from memory to the T-REGISTER.

(2) BITS 10-15 transferred from T-REG. to the I-REG.

(3) BITS 0-9 from T-REG are merged with BITS 10-15 of the M-REG.

MEMORY REFERENCE INSTRUCTION DECODING

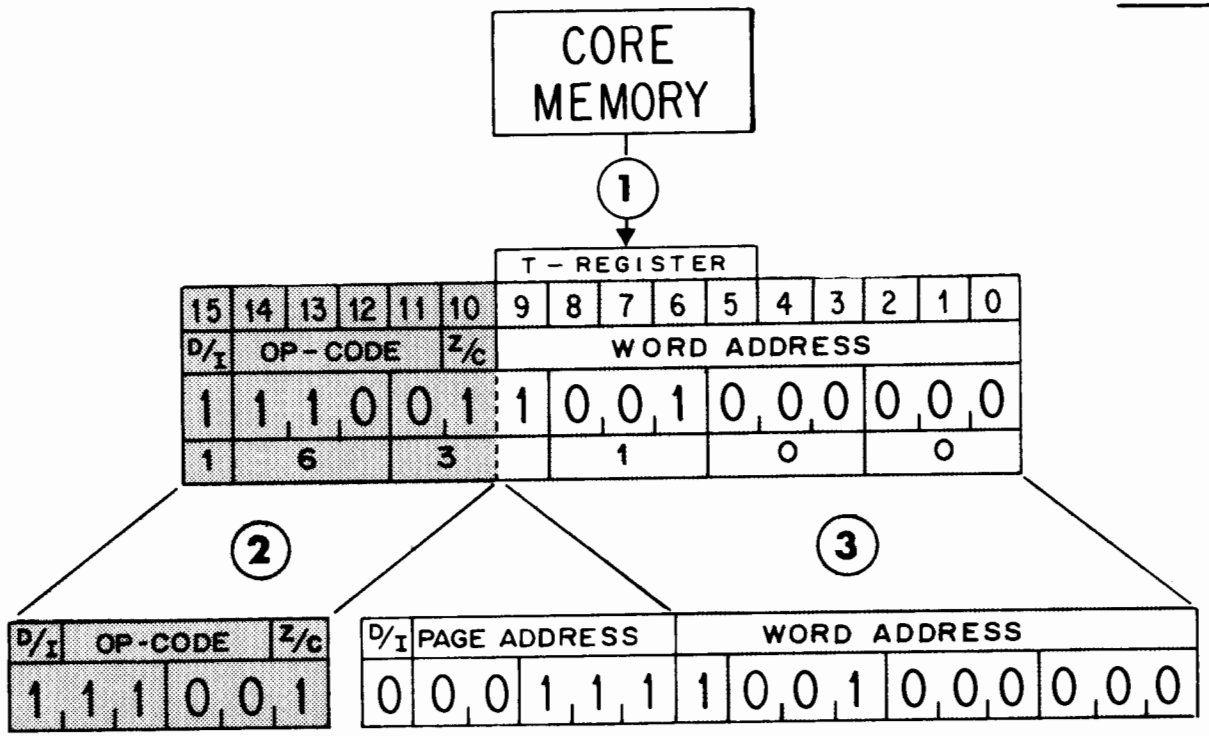


NOTES:

1. Instruction transferred from memory to the T-REGISTER.
2. BITS 10-15 transferred from T-REG to the I-REG.
3. BITS 0-9 transferred from T-REG to the M-REG AND BITS 10-15 of M-REG are cleared to zero.

MEMORY REFERENCE INSTRUCTION DECODING

INDIRECT PART I



(I) INSTRUCTION REGISTER

(M) MEMORY ADDRESS REGISTER

NOTES:

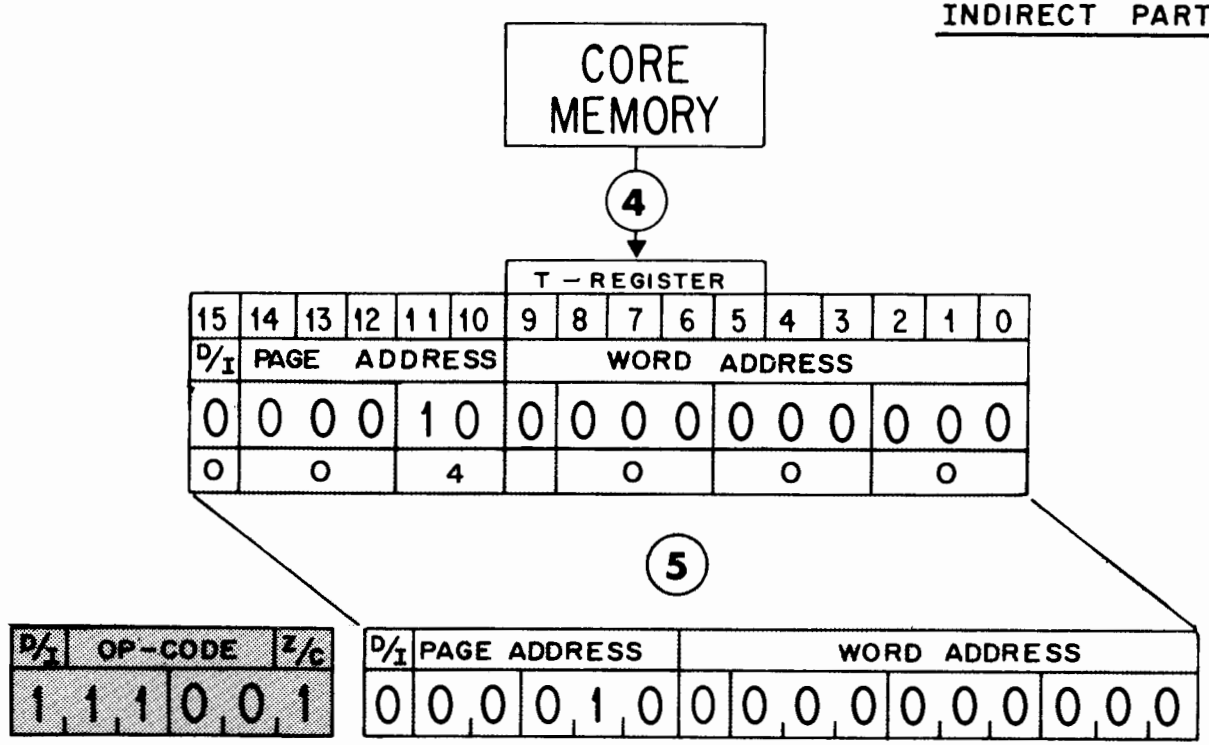
(1) Instruction transferred from memory to the T-REGISTER

(2) BITS 10-15 transferred from T-REG. to the I-REG.

(3) BITS 0-9 from T-REG. are <u>merged</u> with bits 10-15 of the M-REG. <u>BIT 15</u> of <u>I-REG. = 1</u> causes another memory cycle to begin.

MEMORY REFERENCE INSTRUCTION DECODING

INDIRECT PART II



MEMORY ADDRESS REGISTER

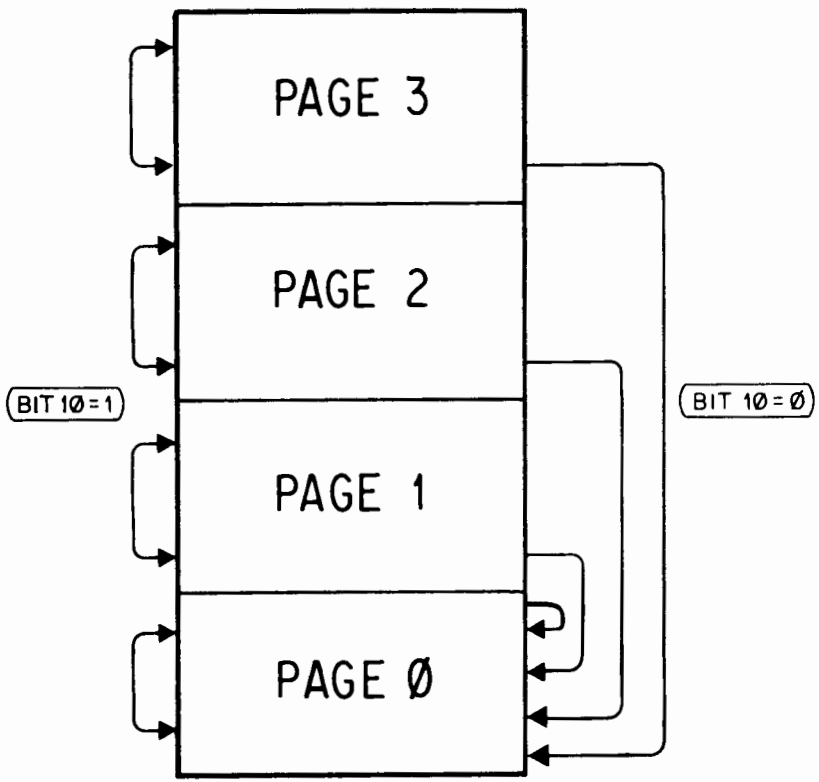
NOTES:

4 The 15 BIT address is transferred from memory to the "T"-REGISTER

5 BITS 0-15 transferred from T-REG. to the M-REG.

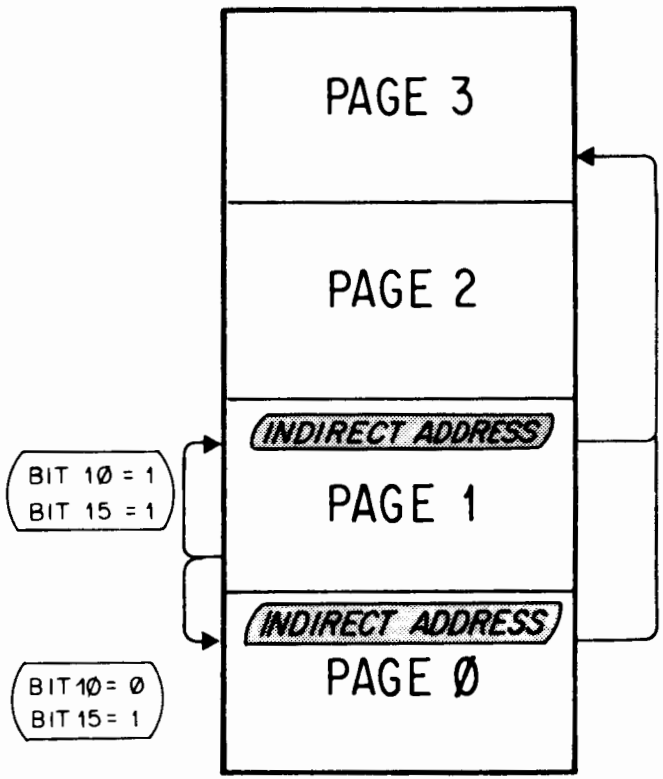
I-REG IS NOT CHANGED

MEMORY ADDRESSING REVIEW



MEMORY ADDRESSING

(DIRECT)



MEMORY ADDRESSING

(INDIRECT)

ADDRESSABLE REGISTER

A UNIQUE FEATURE OF H-P COMPUTERS IS THE ABILITY TO ADDRESS THE "A" OR "B" REGISTERS DIRECTLY. THE METHOD USED TO PROVIDE THIS FEATURE WAS TO MAKE REGISTER "A" SYNONYMOUS WITH MEMORY ADDRESS Ø AND REGISTER "B" SYNONYMOUS WITH MEMORY ADDRESS 1.

THEREFORE

MEMORY ADDRESS Ø IS THE "A" REGISTER.
MEMORY ADDRESS 1 IS THE "B" REGISTER.

EXAMPLE

LOAD THE "A" REGISTER WITH THE CONTENTS OF THE "B" REGISTER.

MNEMONIC	MACHINE CODE
LDA 1	060001

THE SELECTED INSTRUCTION GROUP

THE HP COMPUTERS HAVE A TOTAL OF 68 BASIC INSTRUCTIONS.
THE INSTRUCTIONS ARE CATEGORIZED AS FOLLOWS:

MEMORY REFERENCE	(2 MEMORY CYCLE)	14
REGISTER REFERENCE	(1 MEMORY CYCLE)	41
INPUT-OUTPUT	(1 MEMORY CYCLE)	13
	total	68

IN ORDER TO ELIMINATE CONFUSION AND CONCENTRATE ON PROGRAMMING A SUB-SET OF THE TOTAL INSTRUCTION GROUP WERE SELECTED FOR THIS COURSE. THESE INSTRUCTIONS FORM A GROUP THAT IS REPRESENTATIVE OF THE TOTAL INSTRUCTION SET.

THE SELECTED GROUP BREAKDOWN

MEMORY REFERENCE	12
REGISTER REFERENCE	26
INPUT-OUTPUT	12
total	50

THE STORE INSTRUCTION

MEMORY

X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	BEFORE

A OR B REGISTER

INSTRUCTION

[STA/B Y] STORE THE CONTENTS OF THE SPECIFIED REGISTER IN MEMORY LOCATION Y; WHERE Y IS ANY MEMORY LOCATION. THE CONTENTS OF THE SPECIFIED REGISTER ARE NOT CHANGED. THE PREVIOUS CONTENTS OF MEMORY LOCATION Y ARE LOST.

MEMORY

0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0

0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	AFTER

A OR B REGISTER

THE ADD INSTRUCTION

MEMORY

0	0	0	0	0	0	0	1	0	0	1	1	1	1	1	0

OVFLO	E																
0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	1	BEFORE

A OR B REGISTER

INSTRUCTION

[ADA/B Y]

TO THE CONTENTS OF THE SPECIFIED REGISTER ADD THE CONTENTS OF MEMORY LOCATION Y. THE RESULTS ARE LEFT IN THE SPECIFIED REGISTER. THE CONTENTS OF LOCATION Y ARE NOT CHANGED. THE OVERFLOW REGISTER OR THE EXTEND REGISTER MAY BE SET TO 1 AS A RESULT OF THIS INSTRUCTION.

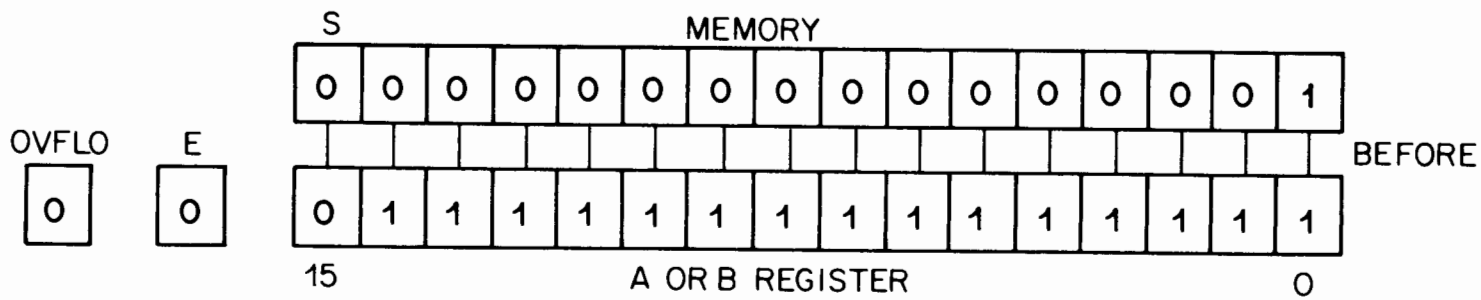
MEMORY

0	0	0	0	0	0	0	1	0	0	1	1	1	1	1	0

OVFLO	E																
0	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0	1	AFTER

A OR B REGISTER

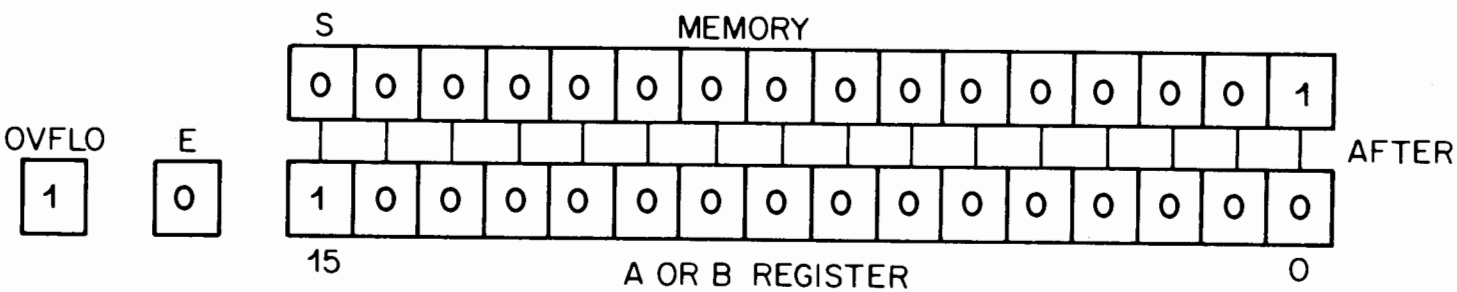
POSITIVE OVERFLOW



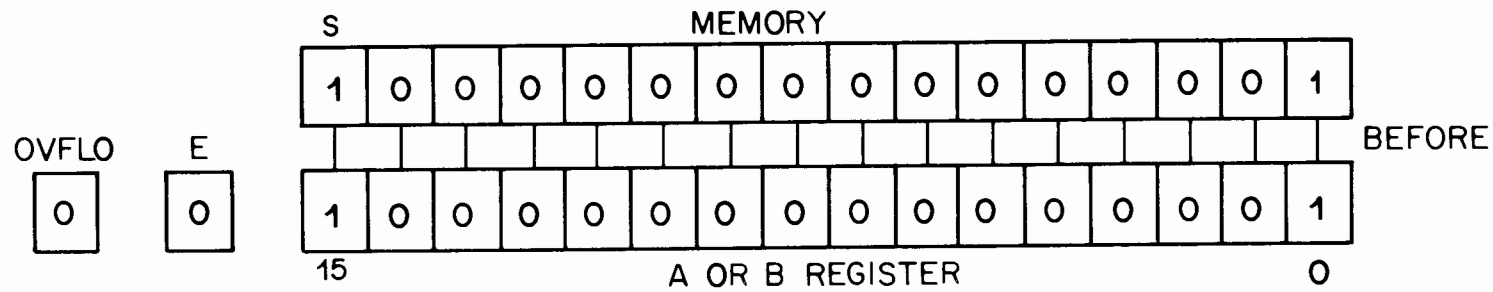
INSTRUCTION

[ADA/B Y]

POSITIVE OVERFLOW CASE. THE CARRY IN TO BIT 15 CHANGES THE SIGN. THE ADDITION OF TWO POSITIVE NUMBERS CANNOT PRODUCE A NEGATIVE RESULT THE OVERFLOW LAMP IS ON.



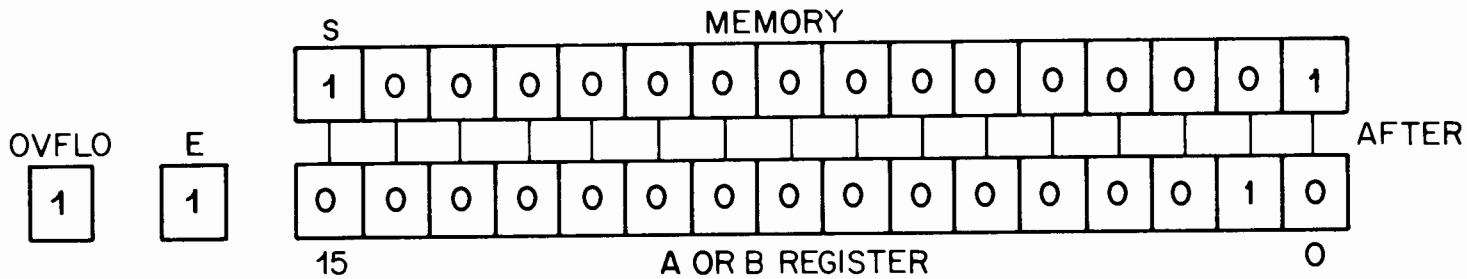
NEGATIVE OVERFLOW



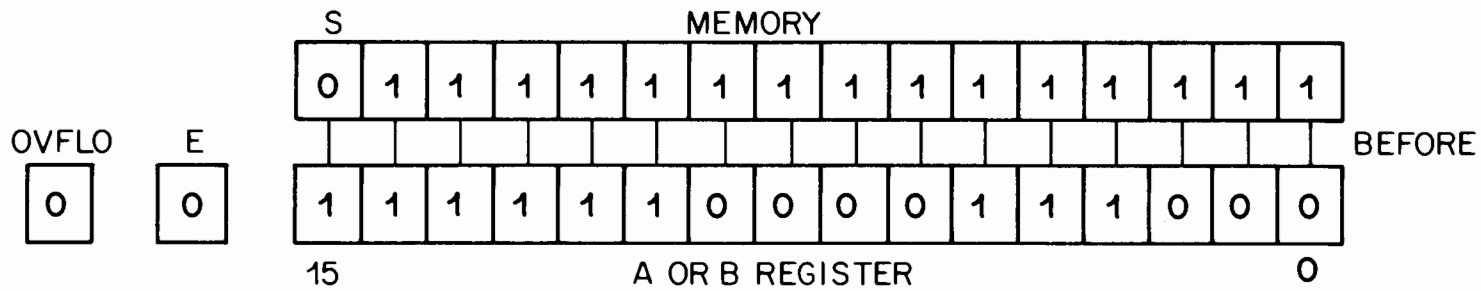
INSTRUCTION

[ADA/B Y]

NEGATIVE OVERFLOW. NO CARRY <u>IN</u> TO BIT 15 CHANGES THE SIGN. THE ADDITION OF TWO NEGATIVE NUMBERS CANNOT PRODUCE A POSITIVE RESULT. THE OVERFLOW LAMP IS ON AND THE CARRY <u>FROM</u> BIT 15 WILL SET E TO 1.



ADDITION OF + AND — NUMBERS



INSTRUCTION

[ADA/B Y]

A POSITIVE NUMBER ADDED TO A NEGATIVE NUMBER (OR THE CONVERSE) WILL NEVER SET THE OVERFLOW CONDITION. IT IS POSSIBLE HOWEVER TO SET "E" TO 1 WITHOUT THE OVERFLOW CONDITION.

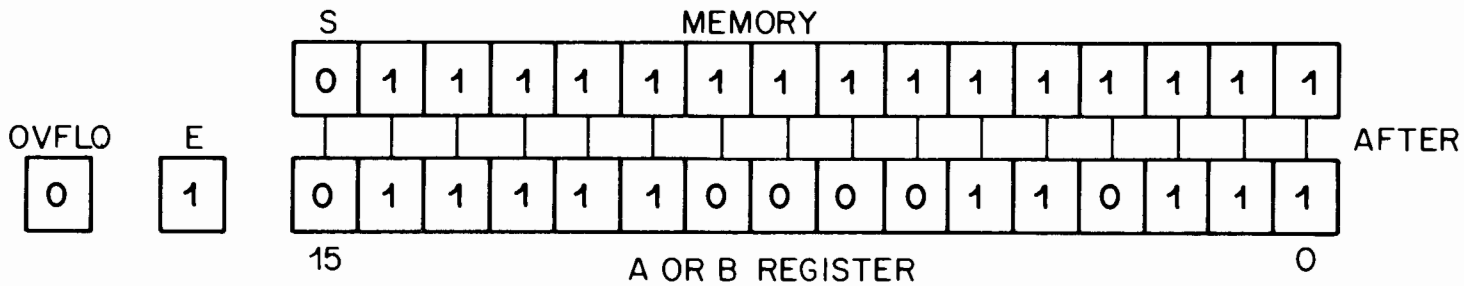


TABLE OF CONDITIONS

(STATUS OF "OVF" & "E" REGISTERS)

MEMORY	A/B REGISTER	RESULT	"OVFLO"	"E" REGISTER
+	+	+	NO	Ø
+	+	−	YES	Ø
+	−	±	NO	1 OR Ø
−	+	±	NO	1 OR Ø
−	−	−	NO	1
−	−	+	YES	1

OVFLO, "E" REGISTERS CAN BE SET BY ADD OR INCREMENT INSTRUCTIONS.

THE HALT INSTRUCTION

1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

①

"T" REGISTER

INSTRUCTION

[HLT (SC)] THIS INSTRUCTION WILL HALT THE COMPUTER. THE INSTRUCTION WILL BE DISPLAYED IN THE "T" REGISTER. THE (SC) OPTION ALLOWS THE SELECTION OF I/O ADDRESSES $0-77_8$.

① SHOWS HALT INSTRUCTION DISPLAY, NO (SC).

② SHOWS HALT INSTRUCTION DISPLAY ($SC=77_8$).

1	0	0	0	0	1	0	0	0	0	1	1	1	1	1	1

②

"T" REGISTER

THE JUMP INSTRUCTION

T REGISTER

0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0

0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	BEFORE

P REGISTER

INSTRUCTION

[JMP Y]

THE JMP INSTRUCTION WILL CAUSE THE COMPUTER TO FETCH ITS NEXT INSTRUCTION FROM MEMORY LOCATION Y. EXECUTION OF THE JMP IS ESSENTIALLY A REGISTER TRANSFER FROM "T" TO "P". IF THE JMP IS DIRECT, THE LOW ORDER 10 BITS (0-9) TRANSFER FROM REGISTER "T" TO REGISTER "P".

T REGISTER(CONTENTS OF Y)

X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	AFTER

P REGISTER

BASIC INSTRUCTION REVIEW

WE HAVE INTRODUCED 5 BASIC MACHINE INSTRUCTIONS AND HOW EACH WORKS INDIVIDUALLY. PROGRAMS ARE GROUPS OF INSTRUCTIONS ARRANGED TO DO A SPECIFIC JOB.

INSTRUCTION REVIEW –

LDA/B Y – LOAD THE A OR B REGISTER FROM MEMORY.
STA/B Y – STORE THE A OR B REGISTER TO MEMORY.
ADA/B Y – ADD TO THE A OR B REGISTER FROM MEMORY.
HLT (SC)– HALT THE COMPUTER.
JMP Y – JUMP OR TRANSFER TO MEMORY LOCATION Y.

A SAMPLE PROBLEM

PROBLEM – WRITE A PROGRAM TO COMPUTE I = J + K WHERE $J = 1372_8$ AND $K = 2347_8$.

SOLUTION

Step 1 – DRAW A SIMPLE FLOW CHART SOLUTION.

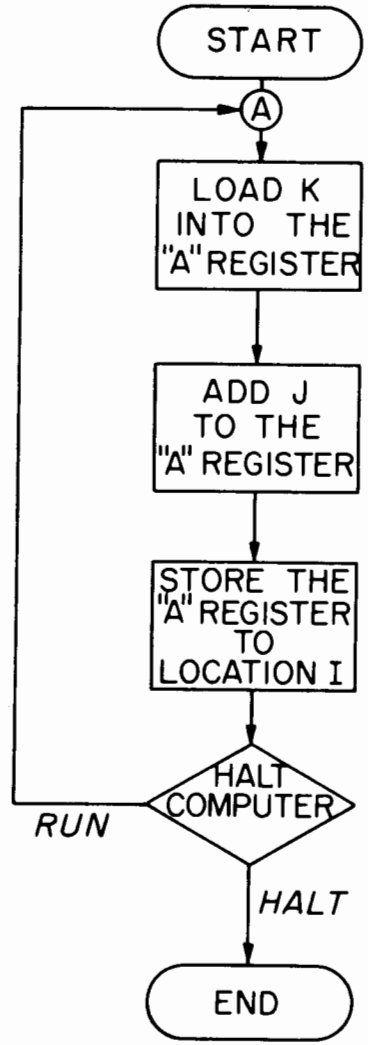
Step 2 – WRITE THE PROGRAM ON THE CODING FORM. MNEMONICS REPRESENT THE INSTRUCTIONS, AND LETTERS (SUCH AS J AND I ABOVE) REPRESENT MEMORY LOCATIONS.

Step 3 – USING THE CODING FORM, CONVERT THE MNEMONICS AND LETTERS FROM STEP 2 INTO THE ACTUAL MACHINE INSTRUCTIONS.

Step 4 – LOAD THE PROGRAM INTO THE COMPUTERS MEMORY.

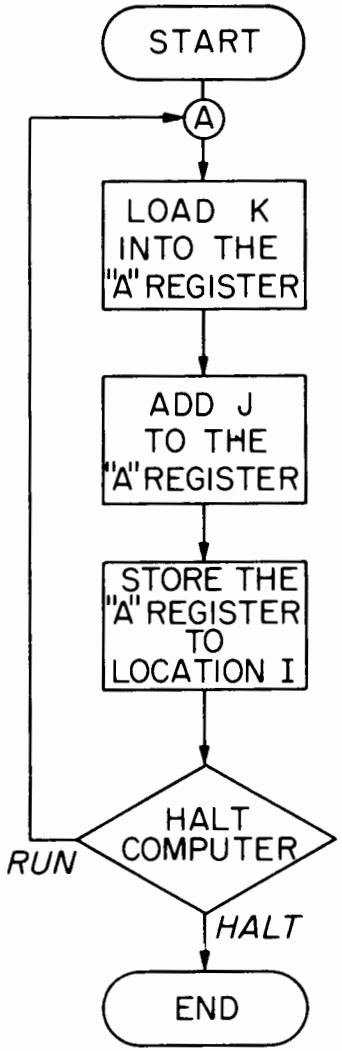
Step 5 – EXECUTE THE PROGRAM.

A SAMPLE FLOW CHART SOLUTION



PROBLEM – SOLVE I = J + K

WRITING THE PROGRAM



LABEL	OP CODE	OPERAND
A	LDA	K
	ADA	J
	STA	I
	HLT	
	JMP	A
K	OCT	0 0 2 3 4 7
J	OCT	0 0 1 3 7 2
I	OCT	0 0 0 0 0 0

ENCODE THE PROGRAM

LABEL	OP CODE	OPERAND	$LOCATION_8$	$CONTENTS_8$
A	LDA	K	06000	062005
	ADA	J	06001	042006
	STA	I	06002	072007
	HLT		06003	102000
	JMP	A	06004	026000
K	OCT	0002347	06005	002347
J	OCT	0001372	06006	001372
I	OCT	0000000	06007	000000

THIS PROGRAM WAS ARBITRARILY ASSIGNED A STARTING ADDRESS OF 6000_8.

ENCODE THE PROGRAM

1 – ASSIGN EACH PROGRAM INSTRUCTION OR DATA VALUE A SEQUENTIAL MEMORY LOCATION.
2 – FIND THE OCTAL EQUIVALENT OF THE OP CODE.
3 – IF MEMORY REFERENCE, DETERMINE THE OCTAL MEMORY ADDRESS.
4 – WRITE THE COMPLETE INSTRUCTION USING 6 OCTAL DIGITS.

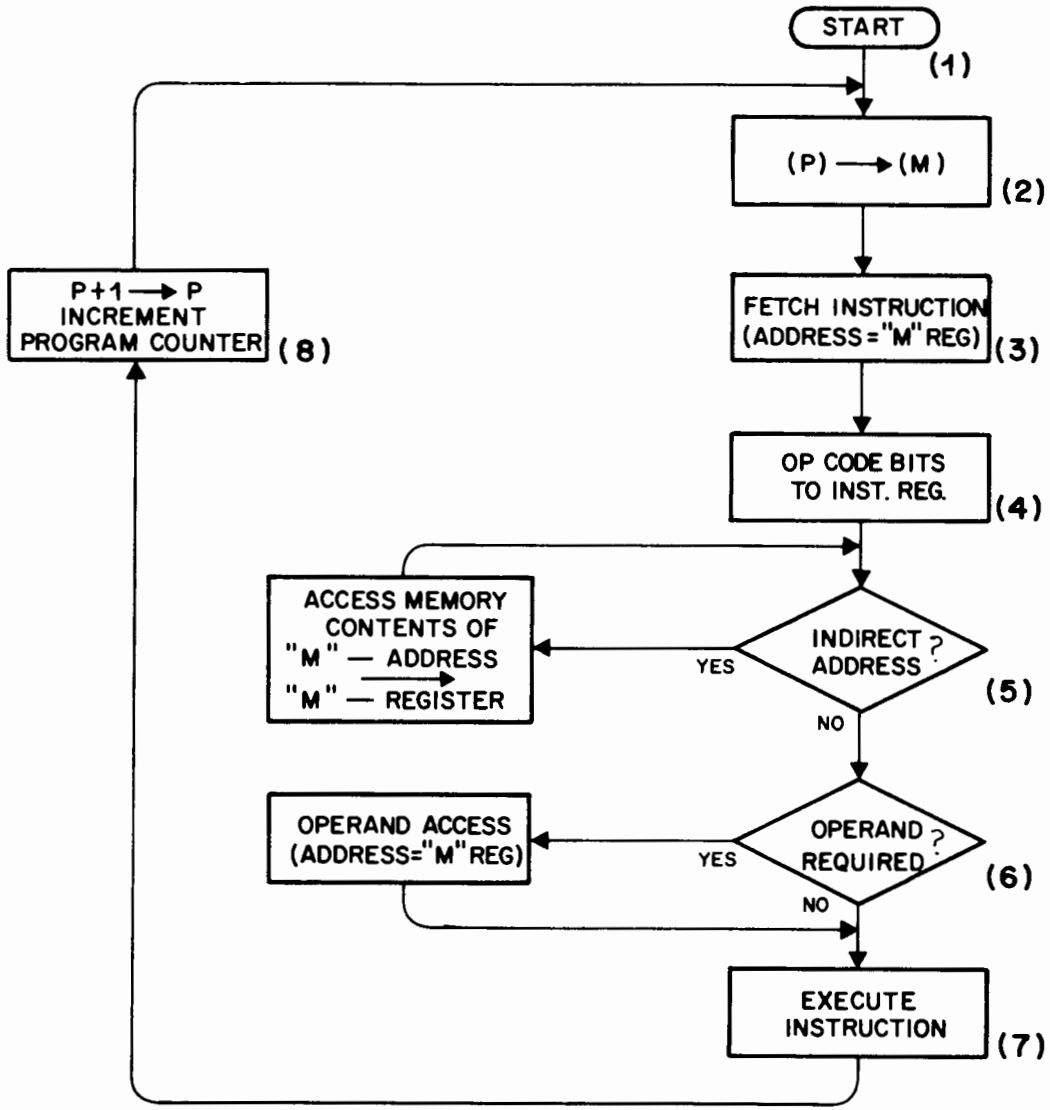
LOADING THE PROGRAM

LABEL	OP CODE	OPERAND	$LOCATION_8$	$CONTENTS_8$
A	LDA	K	06000	062005
	ADA	J	06001	042006
	STA	I	06002	072007
	HLT		06003	102000
	JMP	A	06004	026000
K	OCT	002347	06005	002347
J	OCT	001372	06006	001372
I	OCT	000000	06007	000000

1. SET THE SWITCH REGISTER TO 6000_8 (THE STARTING ADDRESS)
2. PUSH THE "LOAD ADDRESS" BUTTON. (REGISTERS P&M = 6000)
3. SET THE FIRST (NEXT) INSTRUCTION IN THE SWITCH REGISTER.
4. PUSH THE "LOAD MEMORY" BUTTON.
5. REPEAT STEPS 3 AND 4 FOR THE REMAINING INSTRUCTIONS.

NOTE: *EACH TIME THE LOAD MEMORY BUTTON IS DEPRESSED REGISTERS P&M ARE AUTOMATICALLY INCREMENTED BY 1.*

EXECUTING THE PROGRAM



REMARKS

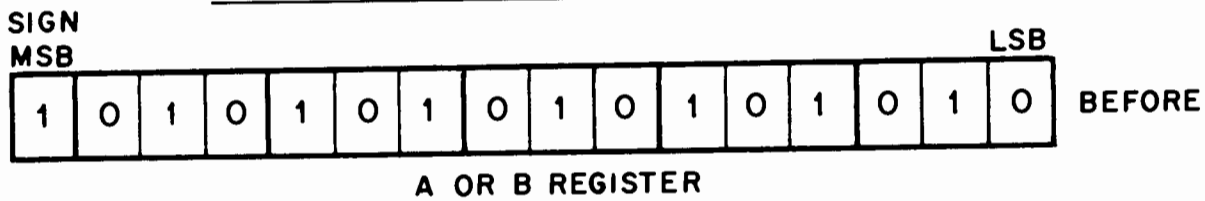
"P" AND "M" ARE ALWAYS EQUAL AT START OF FETCH.

INDIRECT ADDRESS UPDATES THE VALUE OF "M".

CONTENTS OF "M" YIELD THE OPERAND ADDRESS AT THIS POINT.

NO INDIRECT, NO OPERAND, IS A SINGLE PHASE INSTRUCTION.

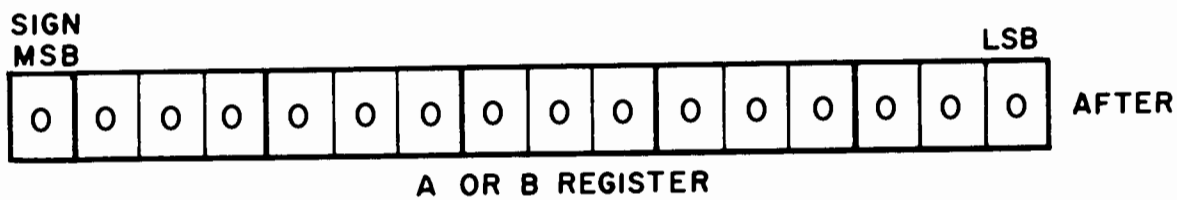
CLEAR ACCUMULATOR



INSTRUCTION

[CLA / CLB]

CLEAR THE INDICATED REGISTER. ALL 16 BITS ARE SET TO Ø. OVFLO, 'E' ARE NOT AFFECTED.



COMPLEMENT ACCUMULATOR

SIGN MSB															LSB	
1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	BEFORE

A OR B REGISTER

INSTRUCTION

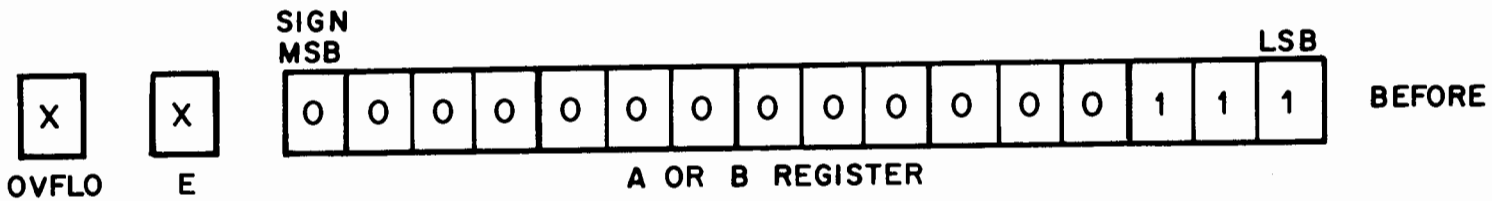
[CMA/CMB]

COMPLEMENT THE CONTENTS OF THE INDICATED REGISTER. THIS IS A 1'S COMPLEMENT. ALL Ø'S BECOME 1'S. ALL 1'S BECOME Ø'S. OVFLO, 'E' ARE NOT AFFECTED.

SIGN MSB															LSB	
0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	AFTER

A OR B REGISTER

INCREMENT THE ACCUMULATOR

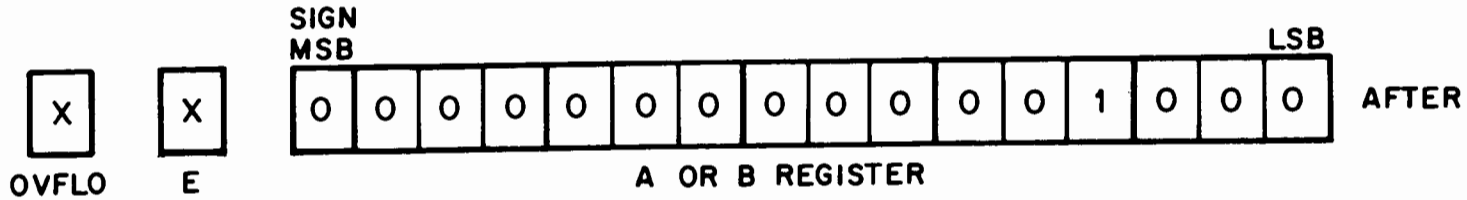


INSTRUCTION

[INA/INB]

X = 1 OR Ø

INCREMENT THE CONTENTS OF THE INDICATED REGISTER BY 1. OVERFLOW CAN BE SET AS A RESULT OF THIS OPERATION. IF A CARRY IS GENERATED FROM BIT 15, THE E REGISTER WILL BE SET TO 1 ALSO.



SKIP SIGN POSITIVE

SIGN MSB															LSB	
0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	SKIP CONDITION

A OR B REGISTER

INSTRUCTION

[SSA/SSB]

X = 1 OR Ø

THIS INSTRUCTION TESTS THE CONTENTS OF BIT POSITION 15. IF 15=Ø (POSITIVE) THE NEXT SEQUENTIAL INSTRUCTION IS SKIPPED. IF BIT POSITION 15=1 (NEGATIVE) THE NEXT SEQUENTIAL INSTRUCTION IS EXECUTED. THE CONTENTS OF A, B, E OR OVFLO ARE NOT AFFECTED BY THIS INSTRUCTION.

SIGN MSB															LSB	
1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	NO SKIP CONDITION

A OR B REGISTER

SKIP ON ZERO

SIGN MSB — LSB

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SKIP CONDITION

A OR B REGISTER

INSTRUCTION

[SZA/SZB]

THIS INSTRUCTION TESTS THE CONTENTS OF THE INDICATED REGISTER. IF THE TEST CONDITION IS PRESENT (16 Ø'S) THE NEXT SEQUENTIAL INSTRUCTION IS SKIPPED. ANY CONDITION OF THE REGISTER OTHER THAN 16 Ø'S CAUSES THE NEXT SEQUENTIAL INSTRUCTION TO BE EXECUTED. THE CONTENTS OF THE A, B, E OR OVFLO REGISTERS ARE NOT AFFECTED BY THIS INSTRUCTION.

SIGN MSB — LSB

1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0

NO SKIP CONDITION

A OR B REGISTER

SKIP ON L.S.B. ZERO

SIGN MSB — LSB

X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0

A OR B REGISTER

SKIP CONDITION

INSTRUCTION

[SLA/SLB]

THIS INSTRUCTION TESTS THE CONTENTS OF BIT POSITION Ø. IF THIS BIT IS Ø THE NEXT SEQUENTIAL INSTRUCTION IS SKIPPED. IF BIT POSITION Ø CONTAINS A 1 THE NEXT SEQUENTIAL INSTRUCTION IS EXECUTED. THE CONTENTS OF A, B, E OR OVFLO ARE NOT AFFECTED BY THIS INSTRUCTION.

SIGN MSB — LSB

X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	1

A OR B REGISTER

NO SKIP CONDITION

INSTRUCTION REVIEW

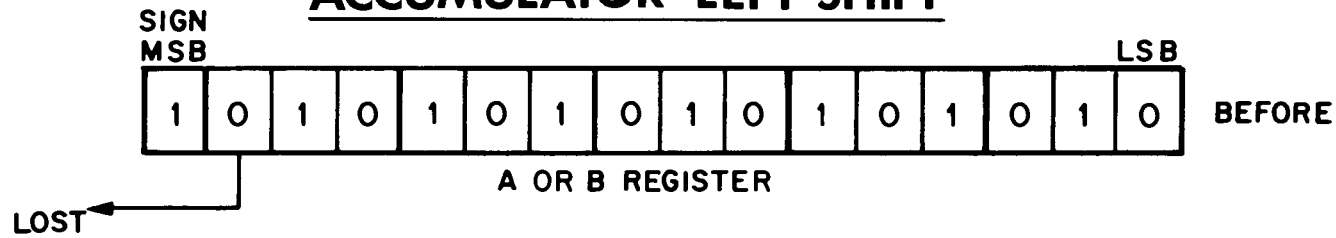
WE CAN NOW ADD 6 REGISTER REFERENCE INSTRUCTIONS TO THE BASIC MEMORY REFERENCE INSTRUCTIONS INTRODUCED PREVIOUSLY.

ALTER-SKIP GROUP
REGISTER REFERENCE INSTRUCTIONS

CLA	———	CLEAR THE "A" REGISTER
CMA	———	COMPLEMENT THE "A" REGISTER
INA	———	INCREMENT THE "A" REGISTER
SSA	———	SKIP IF THE SIGN OF "A" IS POSITIVE
SZA	———	SKIP IF REGISTER "A" IS ZERO
SLA	———	SKIP IF THE L.S.B OF REGISTER "A" IS ZERO

REMEMBER, REGISTER REFERENCE INSTRUCTIONS ARE EXECUTED IN ONE MACHINE CYCLE.

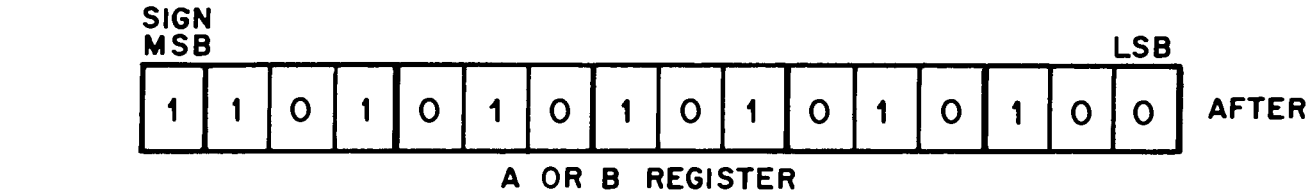
ACCUMULATOR LEFT SHIFT



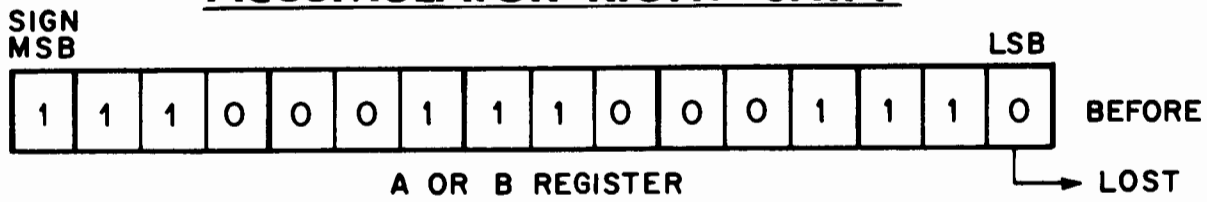
INSTRUCTION

[ALS / BLS]

SHIFT THE INDICATED REGISTER LEFT 1 BIT ARITHMETICALLY, THAT IS, BITS Ø THRU 14 SHIFT LEFT. SIGN, BIT 15, IS NOT AFFECTED. BITS SHIFTED OUT OF BIT POSITION 14 ARE LOST. OVFLO, 'E' REGISTER ARE NOT AFFECTED.



ACCUMULATOR RIGHT SHIFT

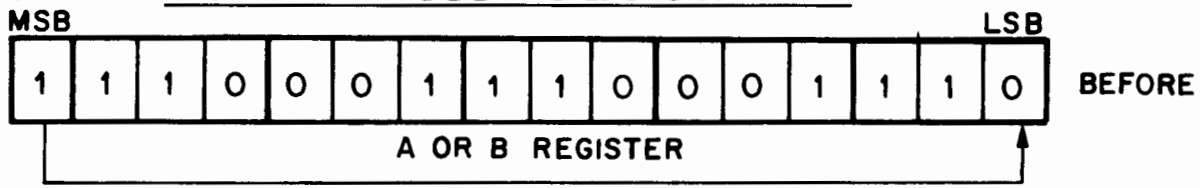


INSTRUCTION

[ARS/BRS]

SHIFT THE INDICATED REGISTER RIGHT 1 BIT ARITHMETICALLY, THAT IS, 14 THRU Ø SHIFT RIGHT. SIGN, BIT 15, IS NOT AFFECTED. BITS SHIFTED OUT OF BIT Ø ARE LOST. A COPY OF BIT 15 (SIGN) IS SHIFTED INTO BIT 14. OVFLO, 'E' REGISTER ARE NOT AFFECTED.

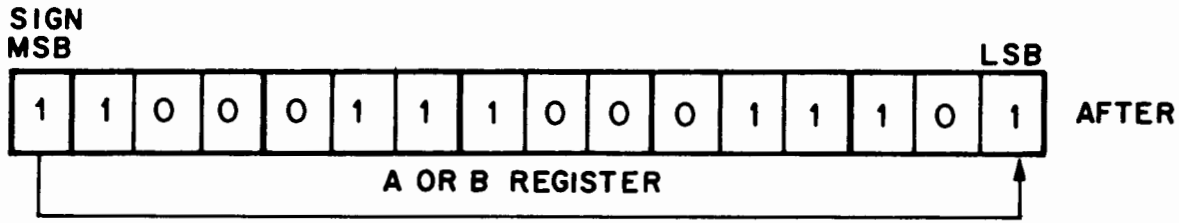
ROTATE ACCUMULATOR LEFT



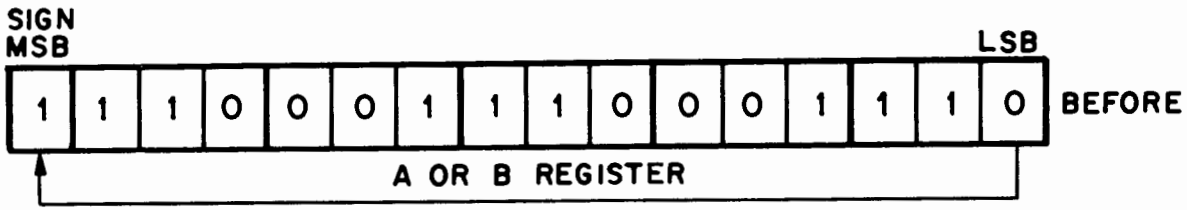
INSTRUCTION

[RAL/RBL]

ROTATE THE INDICATED REGISTER LEFT 1 BIT. BIT 15 IS ROTATED AROUND TO BIT POSITION Ø. NO BITS ARE LOST. OVFLO, 'E' NOT AFFECTED.

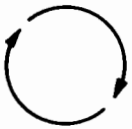


ROTATE ACCUMULATOR RIGHT

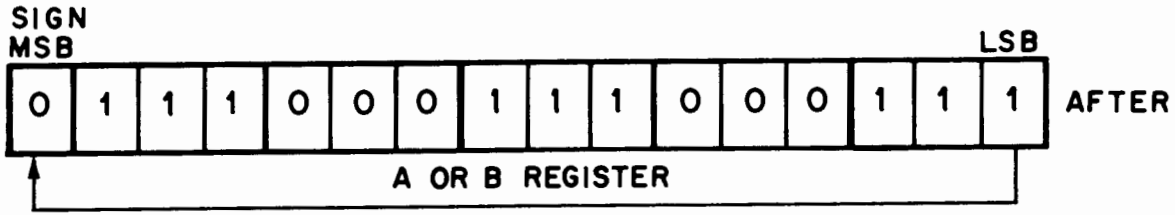


INSTRUCTION

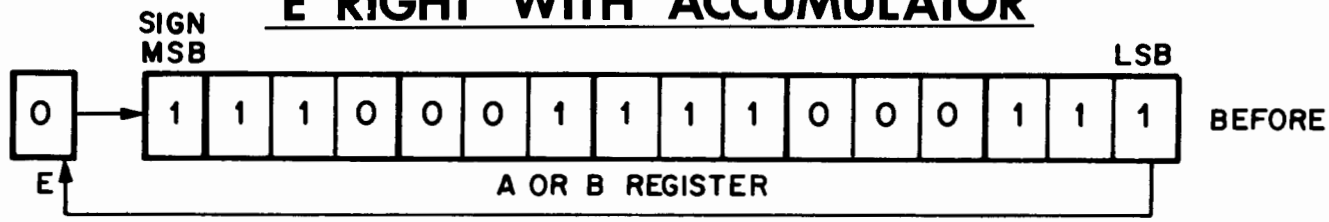
[RAR/RBR]



ROTATE THE INDICATED REGISTER RIGHT 1 BIT. BIT Ø IS ROTATED AROUND TO BIT POSITION 15. NO BITS ARE LOST. OVFLO, 'E' NOT AFFECTED.

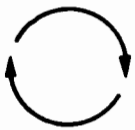


'E' RIGHT WITH ACCUMULATOR

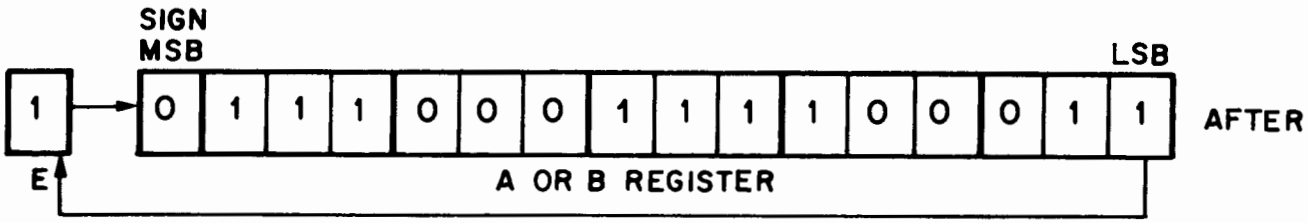


INSTRUCTION

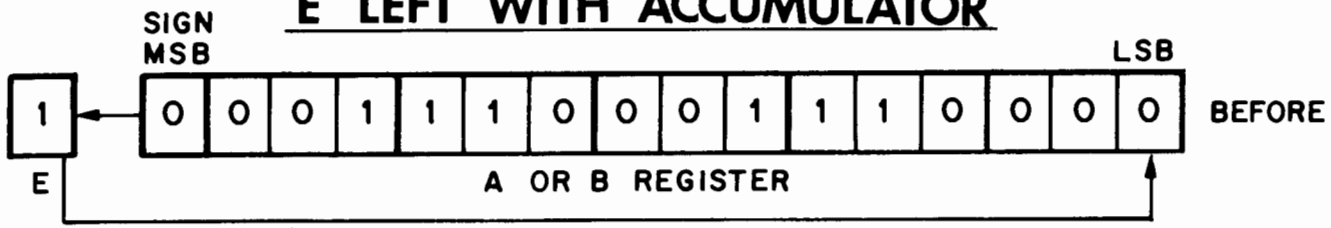
[ERA/ERB]



ROTATE THE INDICATED REGISTER RIGHT, 1 BIT, WITH THE EXTEND REGISTER ('E'). BIT Ø IS ROTATED INTO 'E' AND CONTENTS OF 'E' ARE ROTATED INTO BIT POSITION 15. NO BITS ARE LOST. OVFLO IS NOT AFFECTED.



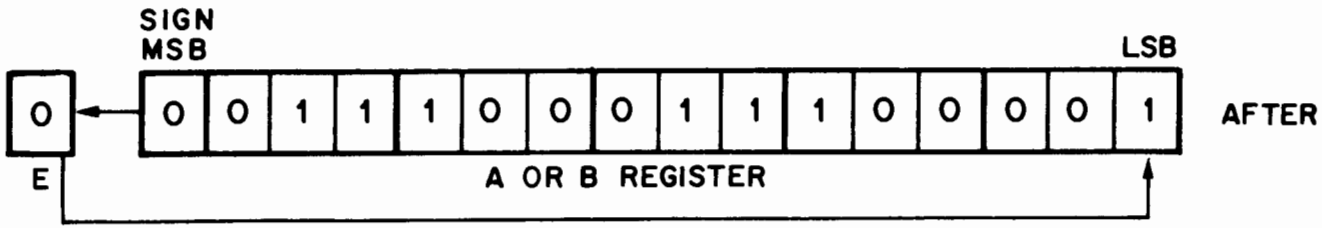
'E' LEFT WITH ACCUMULATOR



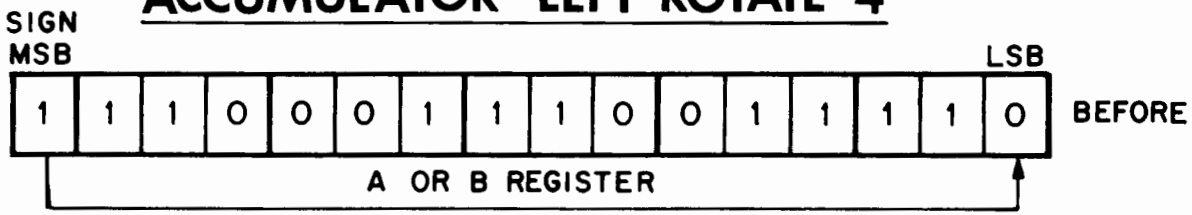
INSTRUCTION

[ELA/ELB]

ROTATE THE INDICATED REGISTER LEFT, 1 BIT, WITH THE EXTEND REGISTER ('E'). BIT 15 ROTATED INTO 'E' AND CONTENTS OF 'E' ARE ROTATED AROUND TO BIT POSITION Ø. NO BITS ARE LOST. OVFLO IS NOT AFFECTED.



ACCUMULATOR LEFT ROTATE 4

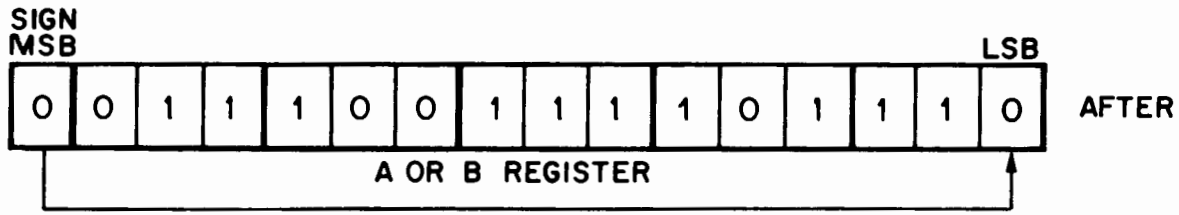


INSTRUCTION

[ALF/BLF]

4

ROTATE THE INDICATED REGISTER LEFT 4 PLACES. NO BITS ARE LOST. BIT 15, 14, 13, 12 ARE ROTATED AROUND TO BIT POSITIONS 3, 2, 1, Ø RESPECTIVELY. OVFLO, 'E' ARE NOT AFFECTED.



SHIFT-ROTATE GROUP REVIEW

ARS – *Arithmetic right shift. A shift right of 1 Bit is equivalent to DIVIDING by 2.*

ALS – *Arithmetic left shift. A shift left of 1 BIT is equivalent to MULTIPLYING by 2.*

RAL – *Rotate left 1 BIT. Used for positioning BITS within the register.*

RAR – *Rotate right 1 BIT.*

ERA – *Rotate "E" right with accumulator 1 BIT.*

ELA – *Rotate "E" left with accumulator 1 BIT. The "E" REGISTER instructions can be used to implement a long rotate operation involving REGISTERS "A" and "B".*

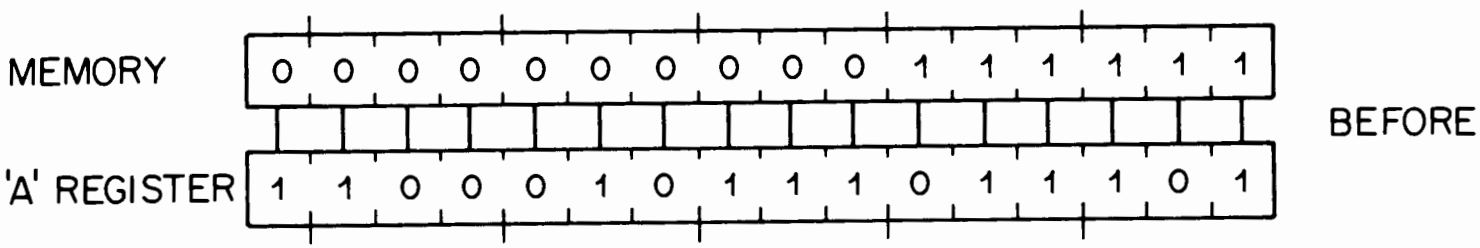
ALF – *The Rotate accumulator left FOUR instruction is used primarily to position 8 BIT alphanumeric characters.*

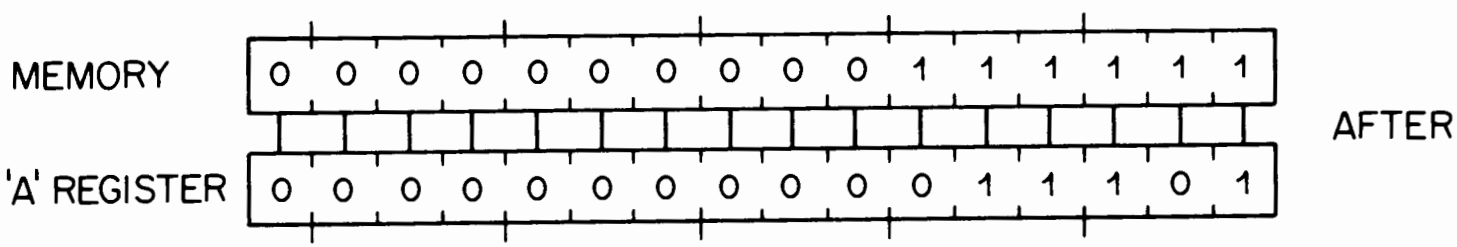
LOGICAL TRUTH TABLE

"A" REGISTER	MEMORY LOCATION	AND	IOR	XOR
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

THE AND INSTRUCTION

LABEL	OP CODE	OPERAND
	AND	MASK
	•	
	•	
MASK	OCT	77





THE COMPARE INSTRUCTION

MEMORY

1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0

1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0

A OR B REGISTER

EQUAL, NO SKIP

INSTRUCTION

[CPA/B Y]

COMPARE THE CONTENTS OF THE SPECIFIED REGISTER AGAINST THE CONTENTS OF MEMORY LOCATION Y. IF ALL 16 BITS COMPARE (EQUAL) THE NEXT SEQUENTIAL INSTRUCTION IS EXECUTED. IF THE COMPARE FAILS, (UNEQUAL) THE NEXT SEQUENTIAL INSTRUCTION IS SKIPPED.

MEMORY

1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0

1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0

A OR B REGISTER

UNEQUAL, SKIP

THE COMPARE INSTRUCTION EXAMPLE

THE CONTENTS OF REGISTER "A" ARE UNKNOWN. DEVISE A PROGRAM SEGMENT THAT WILL TEST THE STATUS OF BITS 3 THROUGH 6. IF THIS FIELD CONTAINS THE OCTAL VALUE 12, TRANSFER TO A LABEL CALLED TRUE. IF THIS FIELD CONTAINS ANY OTHER VALUE THE PROGRAM SHOULD CONTINUE.

REGISTER "A" CONTENTS

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

```
1        5 6 7     10        15        20        25        30        35        40        45        50
          AND   M170                  ISOLATE BITS 3 THROUGH6
          CPA   M120                  DOES"A"COMPARE TO TEST VALUE?
          JMP   TRUE                  YES, JUMP TO TRUE ROUTINE
          JMP   FALSE                 NO,CONTINUE PROGRAM
M170      OCT   170                   OCTAL MASK
M120      OCT   120                   OCTAL TEST VALUE
FALSE     ————
```

LESSON III OBJECTIVES

I. TEACH THE STUDENT HOW TO OPERATE THE HP ASSEMBLER PROGRAM.

II. INTRODUCE ADDITIONAL INSTRUCTIONS

III. TEACH THE STUDENT HOW TO CONFIGURE & USE SIO DRIVERS.

SELECTED INSTRUCTION SHEET

MEMORY REFERENCE INSTRUCTIONS

MNEMONIC	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	D/I	OP-CODE			A/B	Z/C	WORD ADDRESS									
AND	*	0	0	1	0	*	X	X	X	X	X	X	X	X	X	X
JSB	*	0	0	1	1	*										
JMP	*	0	1	0	1	*										
ISZ	*	0	1	1	1	*										
ADA	*	1	0	0	0	*										
ADB	*	1	0	0	1	*										
CPA	*	1	0	1	0	*										
CPB	*	1	0	1	1	*										
LDA	*	1	1	0	0	*										
LDB	*	1	1	0	1	*										
STA	*	1	1	1	0	*										
STB	*	1	1	1	1	*	X	X	X	X	X	X	X	X	X	X

NOTE: D/I, A/B Z/C ARE CODED 0/1

ALTER-SKIP INSTRUCTIONS

CLA	(002400)	CLB	(006400)
CMA	(003000)	CMB	(007000)
INA	(002004)	INB	(006004)
SSA	(002020)	SSB	(006020)
SZA	(002002)	SZB	(006002)
SLA	(002010)	SLB	(006010)

SHIFT-ROTATE INSTRUCTIONS

ARS	(001100)	BRS	(005100)
ALS	(001000)	BLS	(005000)
RAL	(001200)	RBL	(005200)
RAR	(001300)	RBR	(005300)
ERA	(001500)	ERB	(005500)
ELA	(001600)	ELB	(005600)
ALF	(001700)	BLF	(005700)

INPUT-OUTPUT INSTRUCTIONS

MIA	(1024XX)	MIB	(1064XX)
LIA	(1025XX)	LIB	(1065XX)
OTA	(1026XX)	OTB	(1066XX)
HLT	(1020XX)	CLF	(1031XX)
STC	(1027XX)	STC,C	(1037XX)
SFS	(1023XX)	STF	(1021XX)

NOTE: XX DENOTES OCTAL SELECT CODE.

THE 8 STEPS FROM PROBLEM TO PROGRAM

STEP 1 - DEFINE THE PROBLEM

STEP 2 - PREPARE A FLOWCHART SOLUTION

STEP 3 - WRITE AN ASSEMBLY LANGUAGE PROGRAM

STEP 4 - KEYPUNCH THE SOURCE LANGUAGE TAPE USING A TELEPRINTER

STEP 5 - LOAD THE ASSEMBLER PROGRAM INTO THE HP COMPUTER

STEP 6 - ASSEMBLE THE SOURCE PROGRAM

STEP 7 - LOAD THE ASSEMBLER PRODUCED BINARY OBJECT TAPE

STEP 8 - EXECUTE THE OBJECT PROGRAM

ABSOLUTE BINARY LOADER

DEFINITION A 64-WORD PROGRAM USED TO "LOAD" ALL ABSOLUTE BINARY PROGRAM TAPES INTO THE COMPUTER'S MEMORY

CHARACTERISTICS

1. THIS PROGRAM IS CORE RESIDENT IN THE HIGHEST NUMBERED 64 LOCATIONS IN MEMORY
2. THESE 64 LOCATIONS CAN BE "PROTECTED" WHEN NOT IN USE.
3. SHOULD THIS PROGRAM BE ACCIDENTLY DESTROYED, IT MUST BE RELOADED INTO THESE 64 LOCATIONS AGAIN VIA THE CONSOLE SWITCH REGISTER.

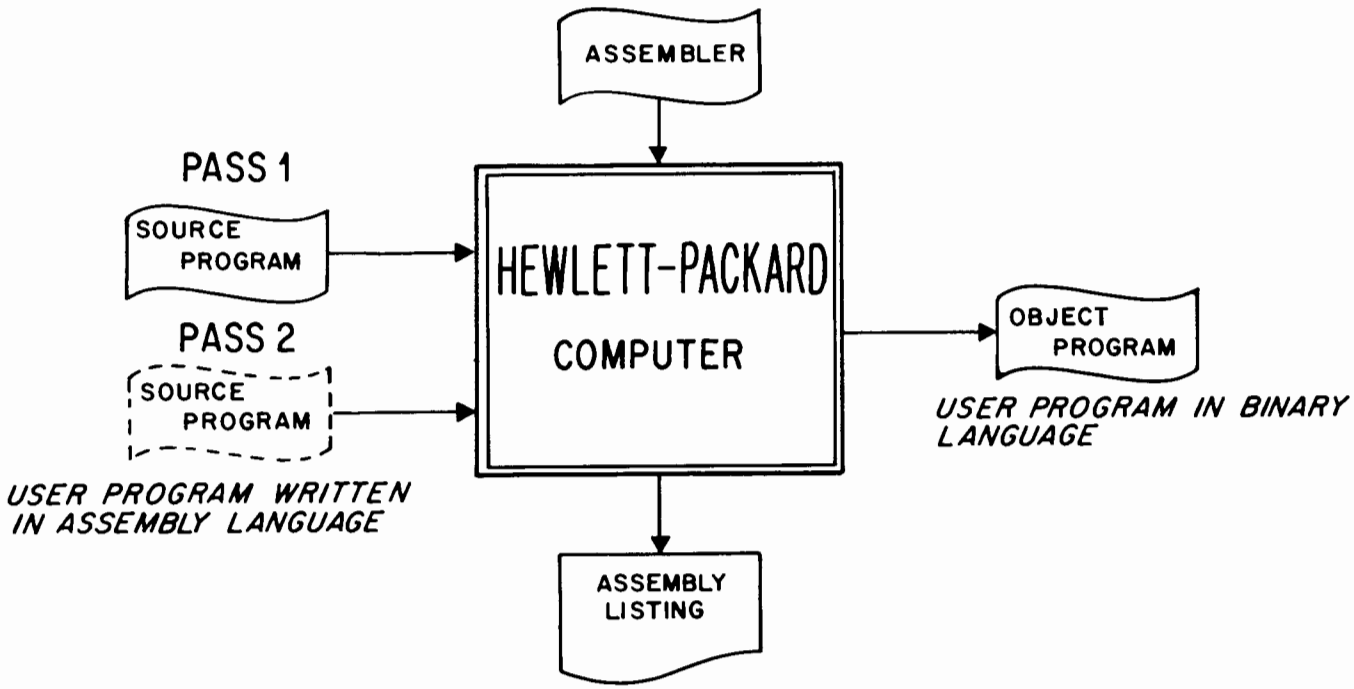
ASSEMBLER CODING FORM

HP ASSEMBLER CODING FORM

Programmer: I.R. SMART | Date: 5-27-68 | Program: SAMPLE PROG.

Label	Operation	Operand	Statement
ASMB,A,B,L,T			
*THIS IS A SAMPLE ASSEMBLY LANGUAGE PROGRAM			
*ASTERISK IN COL 1 INDICATES "COMMENT" STATEMENT			
*PROGRAM TO COMPUTE IANSR=J+K, WHERE J=15726, K=9279			
*			REMARKS FIELD
*			
	ORG	20001B	DEFINE MEMORY ORIGIN
K	DEC	9279	K IS DEFINED AS A DEC CONSTANT
J	DEC	15726	J IS DEFINED AS A DEC CONSTANT
IANSR	OCT	0	RESERVES A MEMORY CELL FOR IANSR
START	NOP		THIS IS THE ENTRY POINT
	LDA	J	LOAD J INTO REGISTER "A"
	ADA	K	ADD CONTENTS OF K TO REG. "A"
	STA	IANSR	STORE J+K TO MEMORY CELL IANSR
	HLT	77B	HALT THE COMPUTER
	JMP	START+1	TRANSFER CONTROL TO START+1
	END		

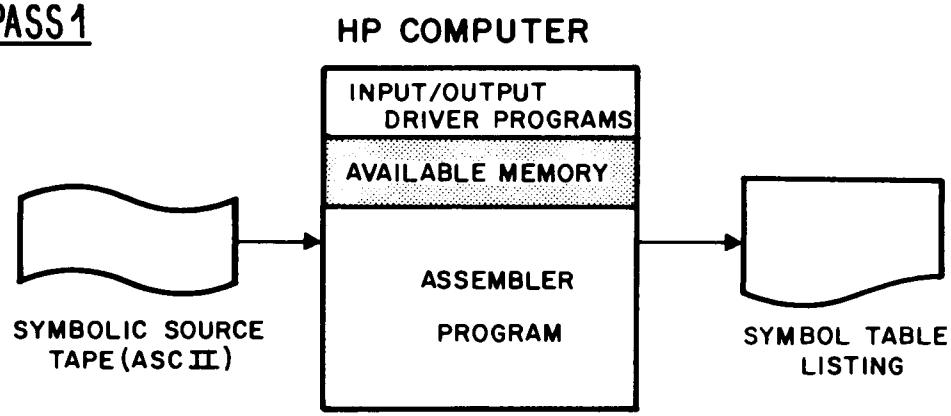
ASSEMBLY PROCESS



1. ASSEMBLER PROGRAM IS LOADED INTO THE COMPUTER.
2. SOURCE PROGRAM IS PROCESSED BY THE ASSEMBLER, PRODUCING THE OBJECT PROGRAM TAPE AND THE ASSEMBLY LISTING IN A TWO PASS OPERATION.

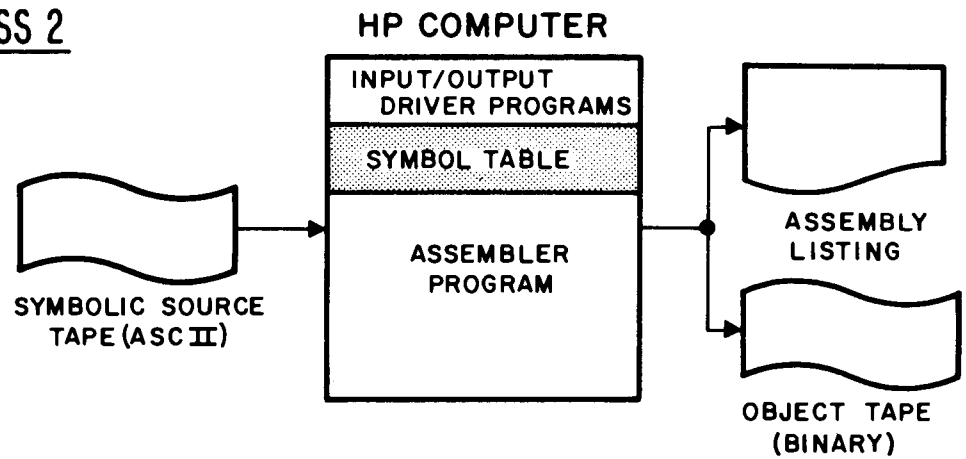
ASSEMBLER PROGRAM OPERATIONS

PASS 1



PASS 1 OPERATIONS CREATE A "SYMBOL TABLE" IN THE AVAILABLE MEMORY AREA.

PASS 2



PASS 2 OPERATIONS RELATE THE SOURCE DATA TO THE SYMBOL TABLE, AND PRODUCE THE BINARY OBJECT TAPE AND THE ASSEMBLY LISTING.

NOTE: ASSEMBLER PRODUCED OUTPUT IS OPTIONAL.

ASSEMBLER PROCESSING

PASS 1

PROGRAM LOCATION COUNTER=

ASSEMBLER SYMBOL TABLE

"K" IS ASSIGNED THE VALUE		2001
"J"	"	"	2002
"IANSR"	"	"	2003
"START"	"	"	2004

PLC,	LABEL	OP CODE	OPERAND
	ASMB,A,B,L,T		
2001		ORG	2001B
2001	K	DEC	9279
2002	J	DEC	15726
2003	IANSR	OCT	0
2004	START	NOP	
2005		LDA	J
2006		ADA	K
2007		STA	IANSR
2010		HLT	77B
2011		JMP	START+1
		END	

NOTE: ONLY STATEMENTS WITH LABELS CREATE SYMBOL TABLE ENTRIES
THE SYMBOL VALUE IS ASSIGNED BY THE PROGRAM LOCATION COUNTER.

ASSEMBLER PROCESSING

PASS 2

LOCATION$_8$	CONTENTS$_8$	LABEL	OP CODE	OPERAND
2001			ORG	2001B
2001	022077	K	DEC	9279
2002	036556	J	DEC	15726
2003	000000	IANSR	OCT	0
2004	000000	START	NOP	
2005	062002		LDA	J
2006	042001		ADA	K
2007	072003		STA	IANSR
2010	102077		HLT	77B
2011	026005		JMP	START+1
			END	

NOTE: MEMORY REFERENCE INSTRUCTIONS SEARCH THE SYMBOL TABLE TO FIND THE PROPER OPERAND VALUE.
MNEMONIC CODES ARE CONVERTED TO THEIR BINARY EQUIVALENT.

ASSEMBLY LISTING

```
 PAGE 0001

0001                    ASMB,A,B,L,T
J        002002
K        002001
IANSR    002003
START    002004
** NO ERRORS*
```

```
 PAGE 0002 #01

0001                    ASMB,A,B,L,T
0002* THIS IS A SAMPLE ASSEMBLY LANGUAGE PROGRAM
0003* ASTERISK IN COL 1 INDICATES "COMMENT" STATEMENT
0004* PROGRAM TO COMPUTE IANSR=J+K, WHERE J=15726, K=9279
0005*                                          REMARKS FIELD
0006  02001                ORG 2001B          DEFINE MEMORY ORIGIN
0007  02001 022077  K      DEC 9279           K IS DEFINED AS A DEC CONSTANT
0008  02002 036556  J      DEC 15726          J IS DEFINED AS A DEC CONSTANT
0009  02003 000000  IANSR  OCT 0              RESERVES A MEMORY CELL FOR IANS
0010  02004 000000  START  NOP                THIS IS THE ENTRY POINT
0011  02005 062002        LDA J              LOAD J INTO REGISTER "A"
0012  02006 042001        ADA K              ADD CONTENTS OF K TO REG. "A"
0013  02007 072003        STA IANSR          STORE J+K TO MEMORY CELL IANSR
0014  02010 102077        HLT 77B            HALT THE COMPUTER
0015  02011 026005        JMP START+1        TRANSFER CONTROL TO START+1
0016                      END
** NO ERRORS*
```

THE ASSEMBLER CHARACTER SET

A	THROUGH Z
Ø	THROUGH 9
.	PERIOD
*	ASTERISK
+	PLUS
–	MINUS
,	COMMA
( )	PARENTHESES
	SPACE

ALL CHARACTERS ARE ASCII CODE

THE CONTROL STATEMENTS

```
LABEL      OP CODE     OPERAND    REMARKS
ASMB,A,B,L,T
           ORG         100B
BEGIN      NOP
           LDA         COUNT
           JMP         GO
NUM        OCT         -12
COUNT      OCT         0
GO         PROGRAM     CONTINUATION
           END
```

THE CONTROL STATEMENT MUST BEGIN IN COLUMN 1, AND IT MUST BE THE FIRST PHYSICAL STATEMENT OF A SOURCE PROGRAM.

ASMB	IDENTIFIES ASSEMBLY INPUT
A/R	ABSOLUTE OR RELOCATABLE PROGRAM.
B	BINARY OBJECT TAPE REQUESTED
L	ASSEMBLY LISTING REQUESTED
T	LISTING OF SYMBOL TABLE REQUESTED
END	MUST BE THE LAST PHYSICAL STATEMENT OF A PROGRAM

USING THE ASSEMBLER CODING SHEET

Label	Operation	Operand
ASMB,A,B,L,T		
	ORG	2ØØØB
	STA	COUNT
	JMP	FINIS
NUM	OCT	-25
COUNT	OCT	Ø
FINIS	HLT	
	END	

1 5 10 15 20 25 30

LABEL FIELD

ONE TO FIVE CHARACTERS, THE FIRST OF WHICH MUST BE ALPHABETIC OR THE PERIOD.

OPERATION FIELD

ALWAYS A THREE LETTER MNEMONIC CODE. COMMA IN COL. 1Ø EXTENDS OPERATION FIELD FOR COMBINABLE INSTRUCTIONS.

OPERAND FIELD

ALPHANUMERIC CHARACTERS, MAY EXTEND TO COL 52.

Ø=ZERO O=ALPHA O 1 OR I= ONE I=ALPHA I LINE TERMINATED BY RETURN/LINE FEED (R/LF)
2= TWO Z=ALPHA Z LINE IS DELETED BY RUBOUT BEFORE R/LF

EXAMPLES OF LABELS

	LABEL					OP CODE	OPERAND	
	1	2	3	4	5			
VALID LABELS	A							
	·	A	B	C	D			
	·	1	2	3	4			
	A	·	1	2	3			
	·							
INVALID LABELS	1	·	A	B				FIRST CHARACTER NUMERIC
	A	B	C	1	2	3		6 CHARS., TRUNCATED TO ABC 12
	A	B	*	C				ASTERISK ILLEGAL
		A	B	C				NO LABEL, FIRST BLANK TERMINATES LABEL FIELD.

SPECIAL USE OF THE ASTERISK IN THE LABEL FIELD

- Asterisk in column 1 identifies a comment statement.
- Positions 2 –80 are available for comments.
- Comments appear in the assembly listing exactly as they appear in the source program.
- Comments are not processed by the assembler and use no storage.

  NOTE: POSITIONS 1 – 68 ONLY WILL BE PRINTED ON THE 2752A TELEPRINTER.

EXAMPLE:

	LABEL	OP CODE	OPERAND
COLUMN	1 2 3 4 5		
	*THIS	IS AN EXAMPLE OF WRITING A COMMENT	
	*STATEMENT		

REMARKS FIELD

Label	Operation	Operand	Remarks
ASMB,A,L			
	ORG	2000B	THE REMARKS FIELD IS
	CLA		SEPARATED FROM THE
	CMA		OPERAND FIELD BY AT
	INA		LEAST ONE SPACE

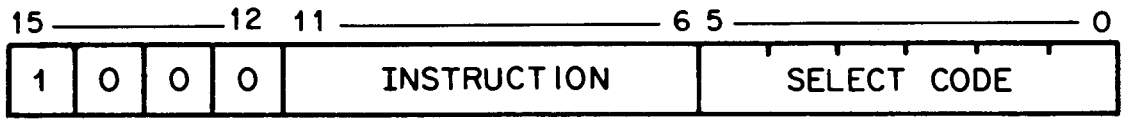
THE REMARKS FIELD EXTENDS FROM THE OPERAND FIELD TO THE 80th CHARACTER. THE ENTIRE STATEMENT LENGTH SHOULD NOT EXCEED 52 CHARACTERS. REMARKS SHOULD BE OMITTED IF THE FOLLOWING STATEMENTS ARE USED WITHOUT OPERANDS: NAM, END, HLT, SOC, SOS.

RELATIVE ADDRESSING

PROGRAM LOCATION COUNTER	LABEL	OP CODE	OPERAND	REMARKS
2001		LDA	COUNT	
2002		STA	COUNT-1	
2003		JMP	*+3	
2004		OCT	Ø	
2005	COUNT	DEC	-25	
2006		HLT		

IN THIS EXAMPLE THE * HAS A VALUE OF 2003 THEREFORE * + 3 = 2006. * EQUALS THE VALUE OF THE P.L.C. WHEN IT IS ENCOUNTERED IN THE ASSEMBLY.

INTRODUCTION TO INPUT/OUTPUT

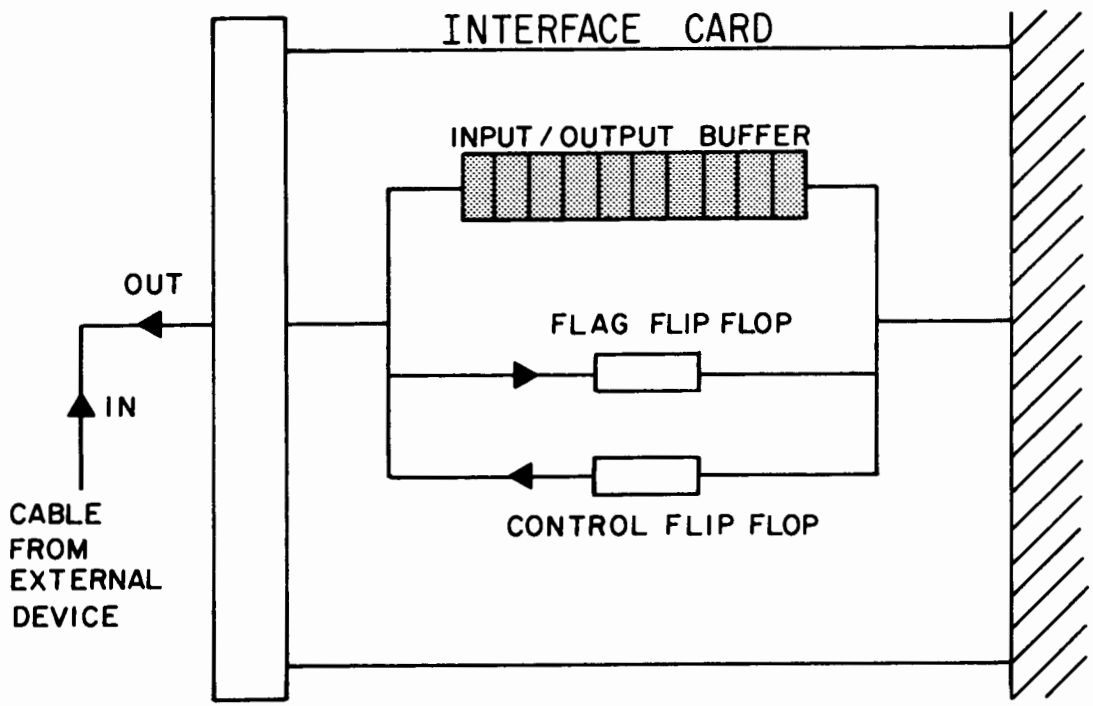


I/O INSTRUCTION FORMAT

INPUT/OUTPUT DEVICE	—	A PHYSICAL DEVICE CAPABLE OF TRANSMITTING AND/OR RECEIVING COMPUTER DATA.
I/O INTERFACE CARD	—	A COMPUTER ELECTRONICS CARD THAT PROVIDES THE PHYSICAL AND ELECTRICAL CONNECTION BETWEEN THE DEVICE AND THE COMPUTER.
I/O CHANNEL	—	THE RECEPTACLE IN THE I/O CARD CAGE THAT HOLDS THE I/O INTERFACE CARD.
SELECT CODE	—	IDENTIFIES A PARTICULAR I/O CHANNEL.

INTERFACE CARDS CONTAIN –

1 *INPUT/OUTPUT BUFFER*
2 *FLAG FLIP FLOP*
3 *CONTROL FLIP FLOP*



TO COMPUTER
LOGIC
AND
A/B REGISTERS

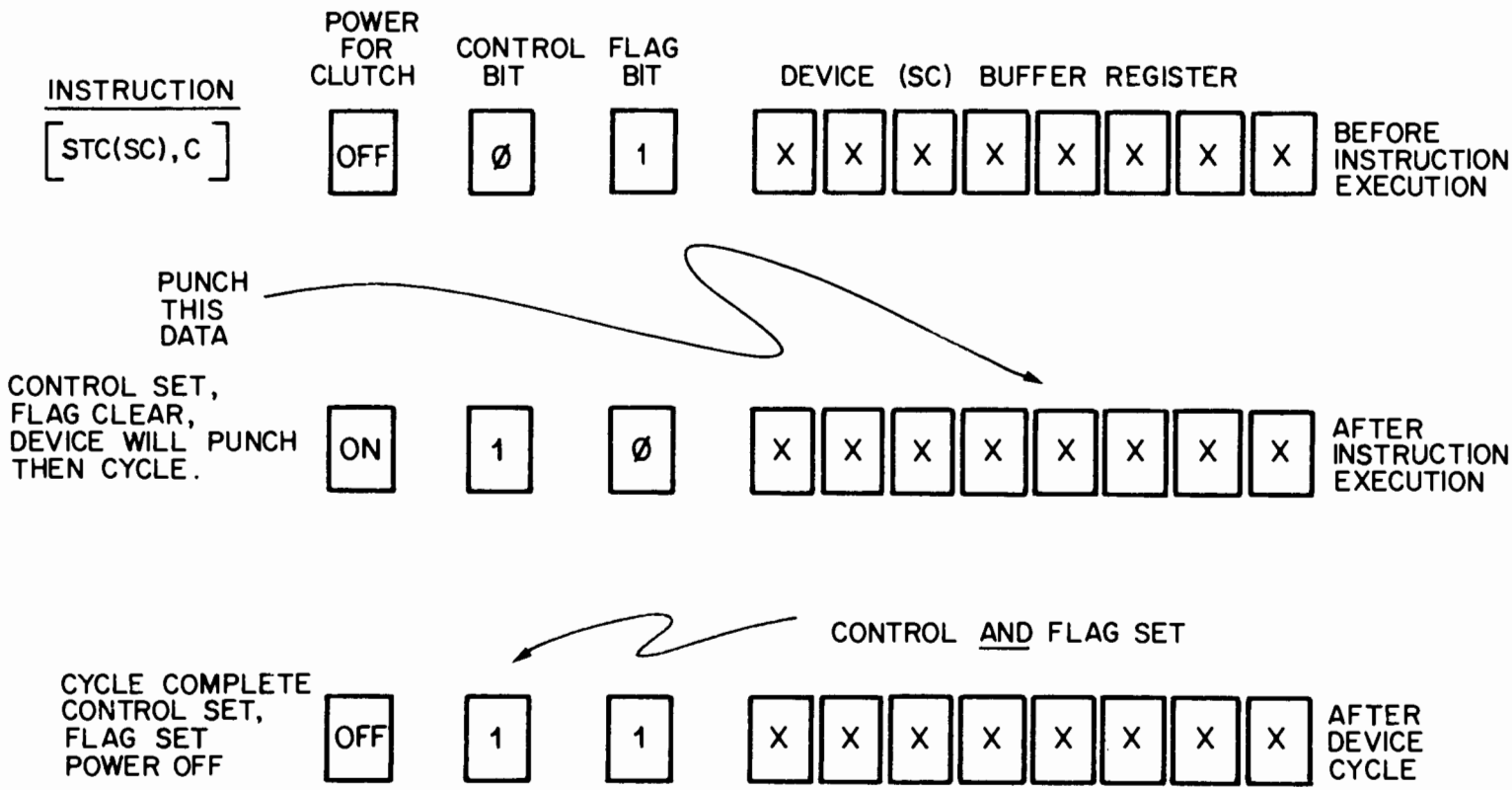
THE STC INSTRUCTION (INPUT DEVICE)

INSTRUCTION	POWER FOR CLUTCH	CONTROL BIT	FLAG BIT	DEVICE (sc) BUFFER REGISTER	
[STC (sc), C]	OFF	Ø	1	X X X X X X X X	BEFORE INSTRUCTION EXECUTION
CONTROL SET, FLAG CLEAR, DEVICE STARTS READ CYCLE	ON	1	Ø	X X X X X X X X	AFTER INSTRUCTION EXECUTION
CYCLE COMPLETE FLAG SET, CONTROL SET, DATA IN BUFFER POWER OFF	OFF	1	1	1 0 1 0 1 0 1 0	AFTER DEVICE CYCLE

CONTROL AND FLAG = 1

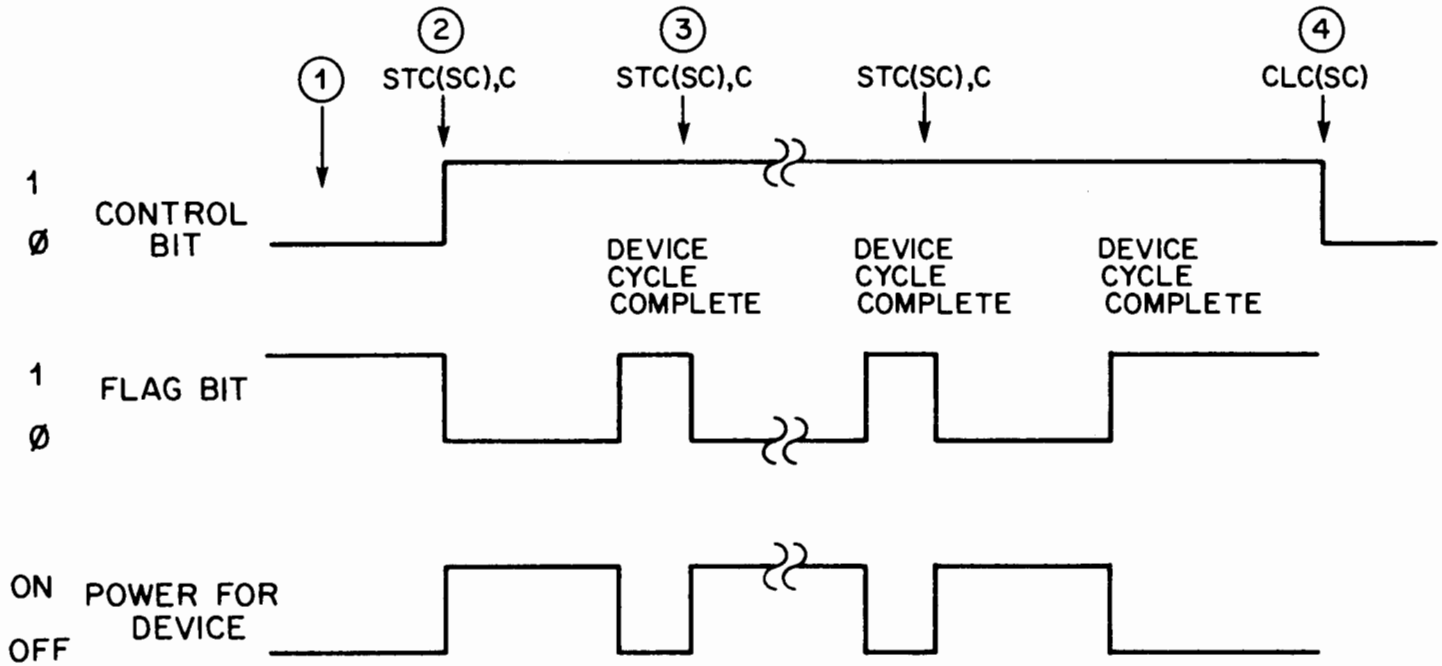
ALTHOUGH THE CONTROL BIT IS SET, ANOTHER STC (SC), C INSTRUCTION MUST BE PROVIDED TO START THE DEVICE (APPLY POWER) AND CLEAR THE FLAG.

THE STC INSTRUCTION (OUTPUT DEVICE)



ALTHOUGH CONTROL BIT IS SET, ANOTHER STC (SC), C INSTRUCTION MUST BE PROVIDED TO START THE DEVICE (APPLY POWER) AND CLEAR THE FLAG. THE CONTENTS OF THE DEVICE BUFFER DO NOT CHANGE DURING A PUNCH CYCLE. THE DEVICE WILL PUNCH, THEN MOVE TAPE, THEN PROVIDE FLAG.

I/O TIMING DIAGRAM



(1) *ASSUME THE INITIALIZED STATE*

CONTROL BIT Ø
FLAG BIT 1
POWER OFF

(2) *FIRST STC (SC) ,C INSTRUCTION*

SETS CONTROL BIT TO 1
CLEAR FLAG BIT TO Ø
APPLIES POWER

(3) *SUBSEQUENT STC (SC) ,C INSTRUCTIONS*

CLEAR FLAG
APPLY POWER
CONTROL BIT REMAINS SET

(4) *AFTER n STC (SC) ,C INSTRUCTIONS THE CLC (SC) INSTRUCTION*

CLEARS CONTROL BIT
FLAG REMAINS SET
POWER REMAINS OFF

THE CLF INSTRUCTION

INSTRUCTION	CONTROL BIT	FLAG BIT								
[CLF (sc)]	X	1	X	X	X	X	X	X	X	X
	X	Ø	X	X	X	X	X	X	X	X

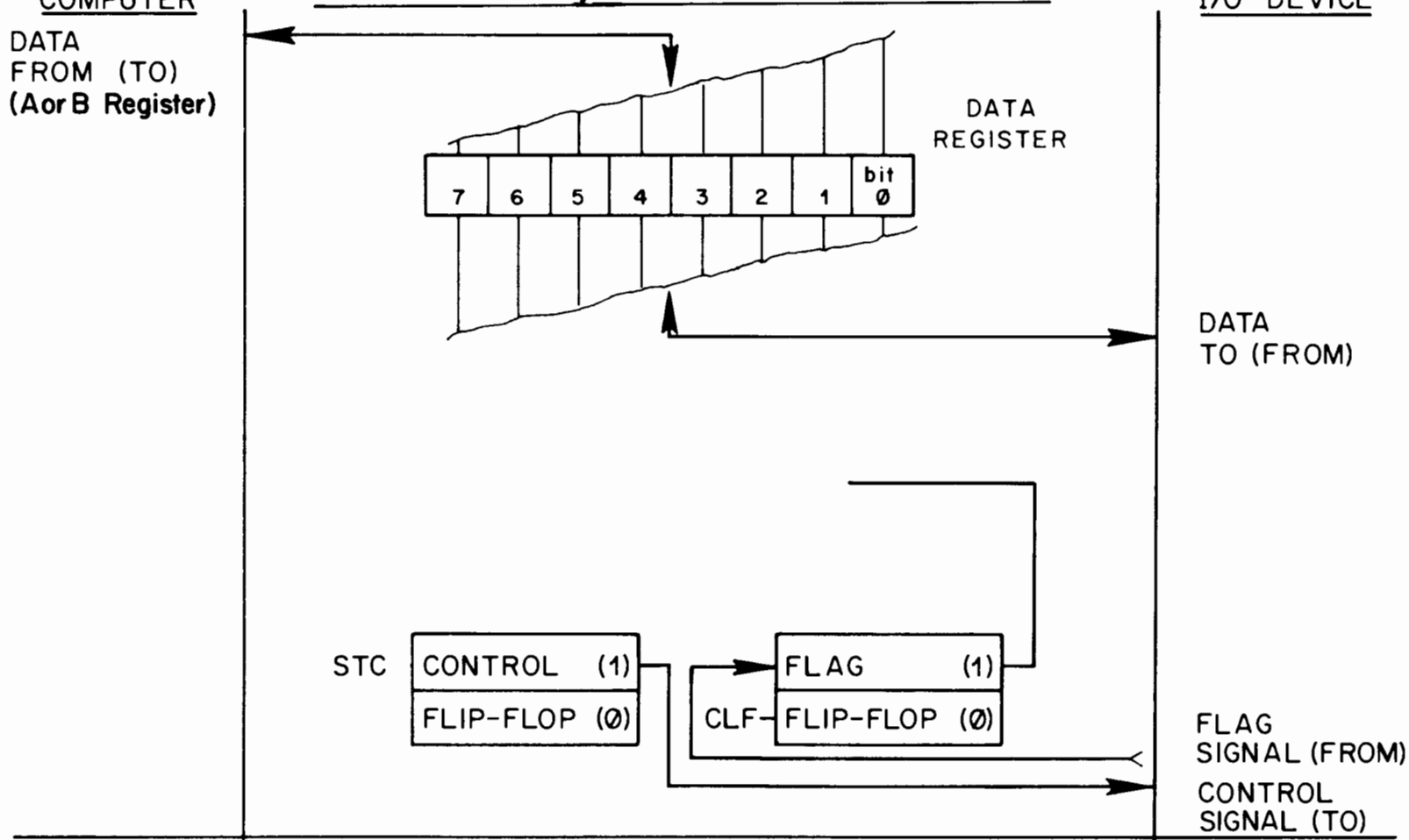
THIS INSTRUCTION CLEARS THE FLAG BIT OF THE SPECIFIED I/O DEVICE. THIS INSTRUCTION CAN BE COMBINED WITH ANY OTHER I/O INSTRUCTION BY USING (,C).

THE SFS INSTRUCTION

INSTRUCTION	CONTROL BIT	FLAG BIT									
[SFS (sc)]	X	1	X	X	X	X	X	X	X	X	SKIP CONDITION
	X	Ø	X	X	X	X	X	X	X	X	NO SKIP CONDITION

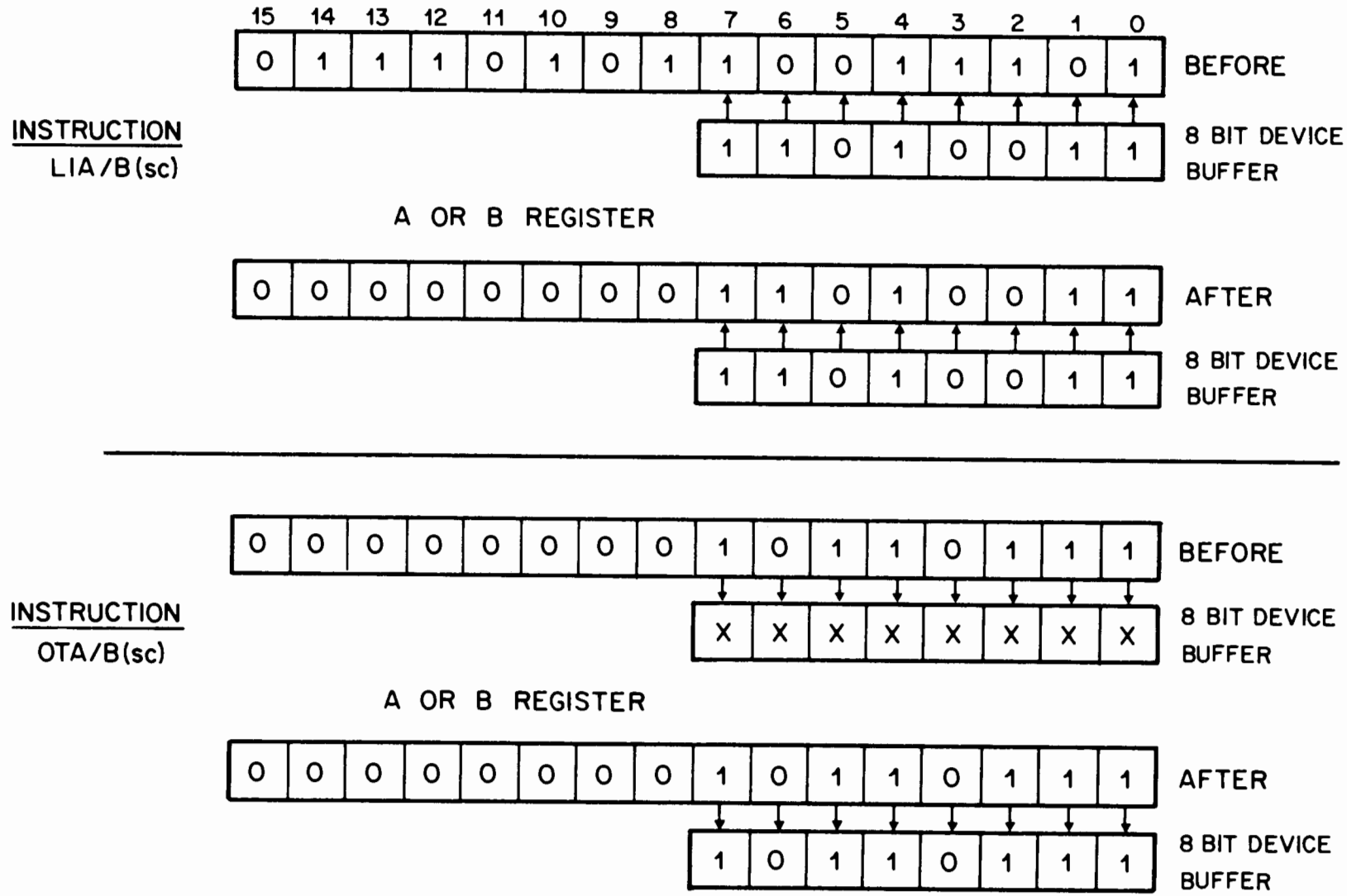
THIS INSTRUCTION WILL TEST THE STATUS OF THE FLAG BIT ON THE SPECIFIED I/O DEVICE. IF THE FLAG BIT IS SET (1), THE NEXT SEQUENTIAL INSTRUCTION IS SKIPPED. IF THE FLAG BIT IS CLEAR (Ø) THE NEXT SEQUENTIAL INSTRUCTION IS EXECUTED.

A TYPICAL I/O INTERFACE CARD

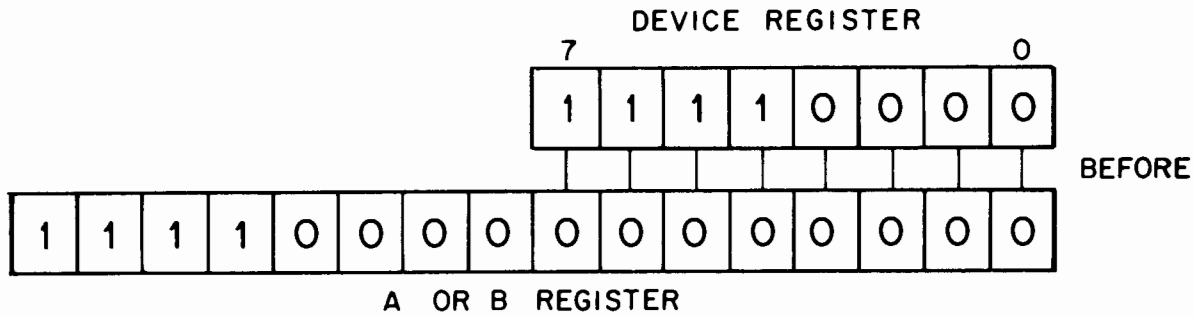


mnemonic	instruction	function
STC (sc)	SET CONTROL FLIP-FLOP	START DEVICE
CLC (sc)	CLEAR CONTROL " "	CLEAR DEVICE
STF (sc)	SET FLAG " "	SET FLAG, "READY"
CLF (sc)	CLEAR FLAG " "	CLEAR FLAG, "BUSY"

8 BIT DATA TRANSFERS



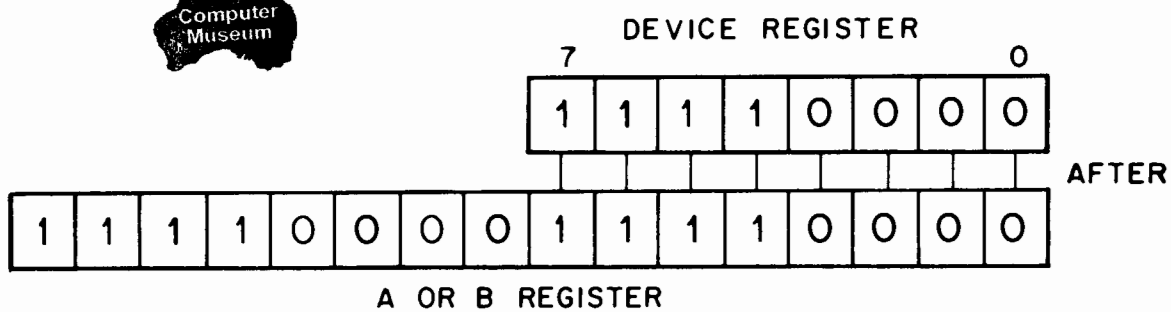
THE MIA/B INSTRUCTION



INSTRUCTION

[MIA/B (sc)]

THIS INSTRUCTION COMBINES THE CONTENTS OF THE SPECIFIED DEVICE REGISTER INTO THE A OR B REGISTER IN AN INCLUSIVE 'OR' OPERATION. OVFLO, E, ARE NOT AFFECTED.



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	LOCATION	CONTENTS
	STC	17B,C	START READER	1001	103717
A	SFS	17B	IS FLAG SET?	1002	102317
	JMP	A	NO, STAY IN LOOP	1003	025002
	LIA	17B	YES, LOAD 1ST 8 BITS	1004	102517
	ALF, ALF		ROTATE TO HI "A"	1005	001727
	STC	17B,C	START READER	1006	103717
B	SFS	17B	IS FLAG SET?	1007	102317
	JMP	B	NO, STAY IN LOOP	1010	025007
	MIA	17B	YES, MERGE 2ND 8 BITS	1011	102417
	HLT		HALT	1012	102000

BOOTSTRAP LOADER PROGRAM

DEFINITION A PROGRAM USED TO LOAD THE ABSOLUTE BINARY LOADER PROGRAM INTO MEMORY

I. ENTER THE FOLLOWING 12 INSTRUCTIONS IN MEMORY BY USING THE FRONT PANEL CONTROLS

ADDRESS OCTAL	CODE OCTAL	INSTRUCTION
00020	1037RA	STC, CLF (READER ADDRESS)
00021	1023RA	SFS (READER ADDRESS)
00022	024021	JMP (21)
00023	1025RA	LIA (READER ADDRESS)
00024	001727	ALF, ALF
00025	1037RA	STC, CLF (READER ADDRESS)
00026	1023RA	SFS (READER ADDRESS)
00027	024026	JMP (26)
00030	1024RA	MIA (READER ADDRESS)
00031	170001	STA (B), I
00032	006004	INB
00033	024020	JMP (20)

RA = OCTAL ADDRESS OF READER

II LOAD THE B-REGISTER WITH 77700

THE INCREMENT -SKIP ZERO INSTRUCTION

MEMORY

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

BEFORE

(NO SKIP CONDITION)

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

AFTER

MEMORY

INSTRUCTION

[ISZ Y]

INCREMENT THE CONTENTS OF MEMORY LOCATION Y AND TEST FOR ZERO. IF Y IS EQUAL TO ZERO THE NEXT SEQUENTIAL INSTRUCTION IS SKIPPED. IF Y IS NOT EQUAL TO ZERO, THE NEXT SEQUENTIAL INSTRUCTION IS EXECUTED.

MEMORY

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

BEFORE

(SKIP CONDITION)

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

AFTER

MEMORY

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
00100			ORG	100B	
00100	000000	BUFFR	BSS	512	SET ASIDE 512 MEMORY LOCATIONS
01100	000000	START	NOP		
01101	002400		CLA		
01102	006400		CLB		
			•		
			•		
			•		

INDIRECT ADDRESSING

THE ASSEMBLER PROGRAM WILL SET THE INDIRECT ADDRESSING BIT (15) FOR ALL MEMORY REFERENCE OPERANDS TAGGED WITH THE ",I" DESIGNATOR.

FOR EXAMPLE:

LOCATION	CONTENTS	LABEL	OPCODE	OPERAND	REMARKS
			•		
2ØØØ	ØØ2Ø21	ADRES	OCT	2Ø21	OCTAL CONSTANT
•			•		
•			•		
•			•		
2Ø1Ø	Ø62ØØØ		LDA	ADRES	PICK UP OCTAL CONSTANT
2Ø11	162ØØØ		LDB	ADRES,I	PICK UP DECIMAL CONSTANT
•			•		
•			•		
•			•		
2Ø21	Ø77777		DEC	32767	DECIMAL CONSTANT
			•		
			•		
			END		

NOTE: AFTER EXECUTION OF CODING
REGISTER "A" = 002021
REGISTER "B" = 077777

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:

LOCATION	CONTENTS	LABEL	OPCODE	OPERAND	REMARKS
00100			ORG	100B	
00100	000114	ADRES	DEF	TABLE	DEF ADDRESS OF TABLE
00101	000000	START	NOP		
00102	066000		LDB	ADRES	GET ADDRESS OF TABLE
00103	160001		LDA	1,I	LOAD "A" THRU "B"
•	•		•	•	(GET FIRST TABLE VALUE)
•	•		•	•	
•	•		•	•	
•	•		•	•	
00114	000000	TABLE	BSS	100	
			END		

ADDRESS MODIFICATION

ADDRESS MODIFICATION IS AN IMPORTANT PROGRAMMING TECHNIQUE.

FOR EXAMPLE: A PROGRAM TO SUM THE CONTENTS OF 10 SEQUENTIAL MEMORY LOCATIONS.

LABEL	OPCODE	OPERAND	REMARKS
*			
*			
	ORG	2000B	
START	NOP		
	LDA	CNT	INITIALIZE
	STA	CNTR	COUNTER
	CLA		CLEAR A
	LDB	PNTR	LOAD ADDRESS OF TABLE
LOOP	ADA	B,I	ADD INDIRECTLY THRU "B"
	INB		ADD 1 TO ADDRESS
	ISZ	CNTR	IS COUNTER ZERO?
	JMP	LOOP	NO, CONTINUE
	HLT	77B	YES, HALT COMPUTER
	JMP	START+1	PROGRAM RESTART
CNT	DEC	-10	COUNT VALUE
CNTR	BSS	1	WORKING COUNTER
PNTR	DEF	TABLE	ADDRESS OF TABLE
TABLE	DEC	10,5,15,23,752,8272,33,84,9,28	
	END		

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.
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

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
2000		SSA	
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

SYSTEM INPUT OUTPUT SUBROUTINES (SIO DRIVERS)

DEFINITION: A "DRIVER" IS A SUBROUTINE PROGRAM USED TO PERFORM I/O OPERATIONS ON A SPECIFIC DEVICE.

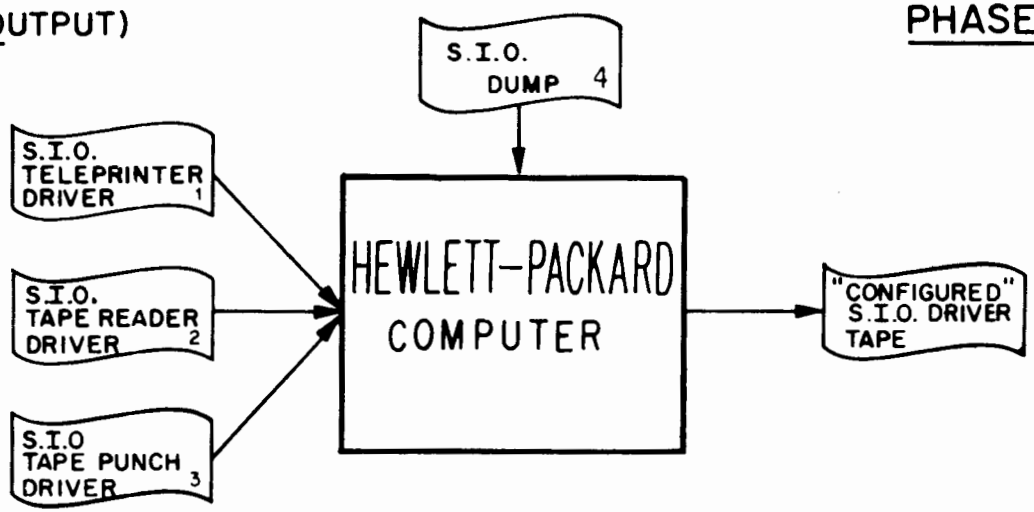
- SIO DRIVERS ARE ABSOLUTE PROGRAMS USED TO PROVIDE THE STANDARD SOFTWARE SYSTEMS WITH I/O CAPABILITY.
- THE "CONFIGURED" ASSEMBLER TAPE INCLUDES BOTH THE ASSEMBLER PROGRAM & THE SIO DRIVERS REQUIRED TO READ THE SOURCE PROGRAM TAPE AND PRODUCE THE LISTINGS AND BINARY OBJECT TAPES.

NOTE: SIO DRIVERS OPERATE WITHOUT INTERRUPT CONTROL

SYSTEM INPUT-OUTPUT DUMP ROUTINE

(SYSTEM INPUT OUTPUT)

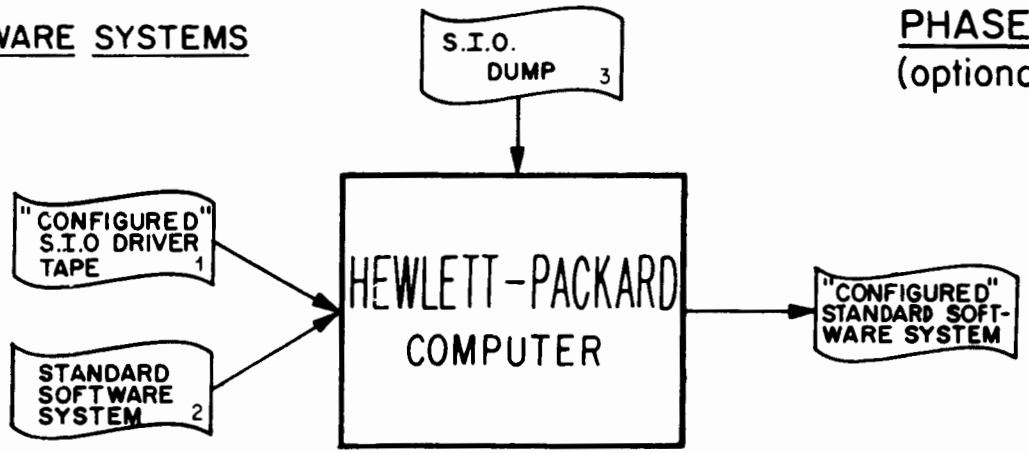
PHASE 1



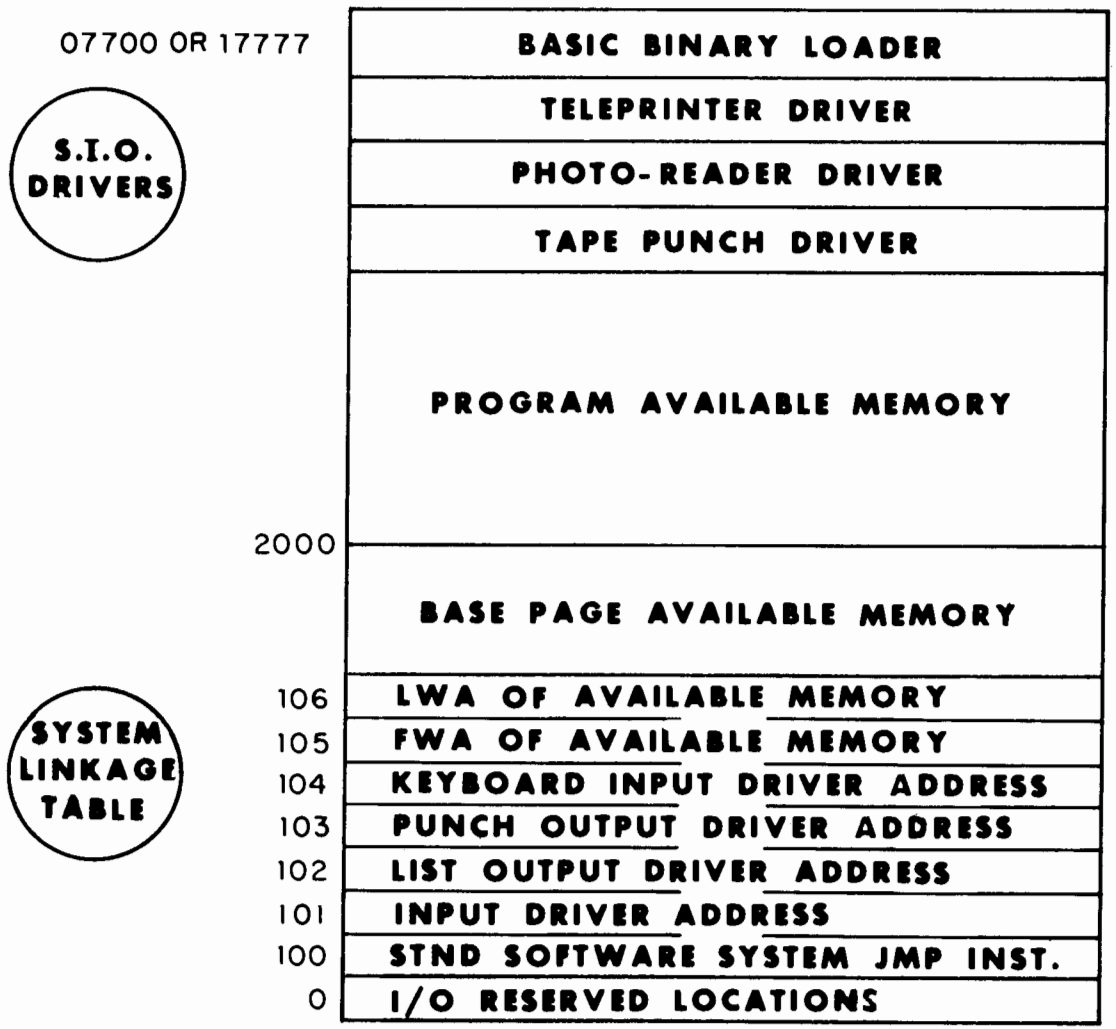
STANDARD SOFTWARE SYSTEMS

PHASE 2
(optional)

FORTRAN
ALGOL
ASSEMBLER
SYMBOLIC EDITOR



S.I.O. MEMORY MAP



CONFIGURING A PROGRAM SYSTEM

THE SYSTEMS TO BE CONFIGURED

- ASSEMBLER SYSTEM
- SYMBOLIC EDITOR SYSTEM
- FORTRAN COMPILER SYSTEM-PASS 1 TAPE ONLY
- ALGOL COMPILER

THE S.I.O. DRIVERS (ONLY PROVIDED WHEN I/O DEVICE ORDERED)

- TELEPRINTER
- TAPE READER
- TAPE PUNCH

THE PROCEDURE (BASIC BINARY LOADER USED FOR ALL MODULE LOADING)

1. LOAD A DRIVER. (THE TELEPRINTER MUST BE LOADED FIRST)(PHOTOREADER SECOND)(PUNCH LAST)
2. PLACE THE ADDRESS 2 INTO THE P-REGISTER; SET SWITCHES 5-0 OF THF SWITCH REGISTER TO THE CHANNEL NUMBER ASSOCIATED WITH THAT DEVICE AND PRESS RUN.
3. REPEAT ABOVE STEPS FOR EACH DRIVER TO BE INCLUDED.
4. LOAD THE PERTINENT PROGRAMMING SYSTEM.
5. LOAD THE S.I.O. DUMP ROUTINE.
6. PLACE THE ADDRESS 2 INTO THE P-REGISTER & SET SWITCH 15 OF THE SWITCH REGISTER TO OBTAIN THE FOLLOWING OPTIONS:

   0 = OUTPUT TO CONTAIN ONLY S.I.O. DRIVERS AND SYSTEM LINKAGE TABLE.
   1 = PROGRAM SYSTEM IS TO BE INCLUDED ON OUTPUT.
7. PRESS RUN TO COMMENCE PUNCH-OUT.
8. MULTIPLE COPIES MAY BE OBTAINED BY REPEATING FROM SWITCH 15 SETTING OF STEP 6.

SIO DRIVER USE FOR ABSOLUTE PROGRAMS

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

```
      •
      •
      •
      LDA        (buffer length) ①
      LDB        (buffer address)
      JSB        10X  B,I  ②
      •
      •
      •
```

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 <u>contains</u> the address of the proper driver

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

EXAMPLE - S.I.O. DRIVER USE

WRITE A PROGRAM WHICH "READS" A 72 CHARACTER ASC II RECORD FROM THE PHOTOREADER AND THEN TYPES THOSE SAME CHARACTERS ON THE TELEPRINTER.

```
        ORG  2000B
BGIN    NOP
        LDA  LN
        LDB  ADRES
        JSB  101,I
        LDA  LN
        LDB  ADRES
        JSB  102,I
        JMP  BGIN+1
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.

LESSON IV OBJECTIVE

INTRODUCE THE STUDENT TO THE USER PROGRAMMING ENVIRONMENT.

YOU WILL LEARN ABOUT:

1. FORTRAN - CONCEPTS AND OPERATING PROCEDURES
2. BASIC CONTROL SYSTEM (BCS) - CONCEPTS AND CONFIGURATION
3. INTERRUPT SYSTEM - HOW AND WHY

PROGRAMMING LANGUAGES

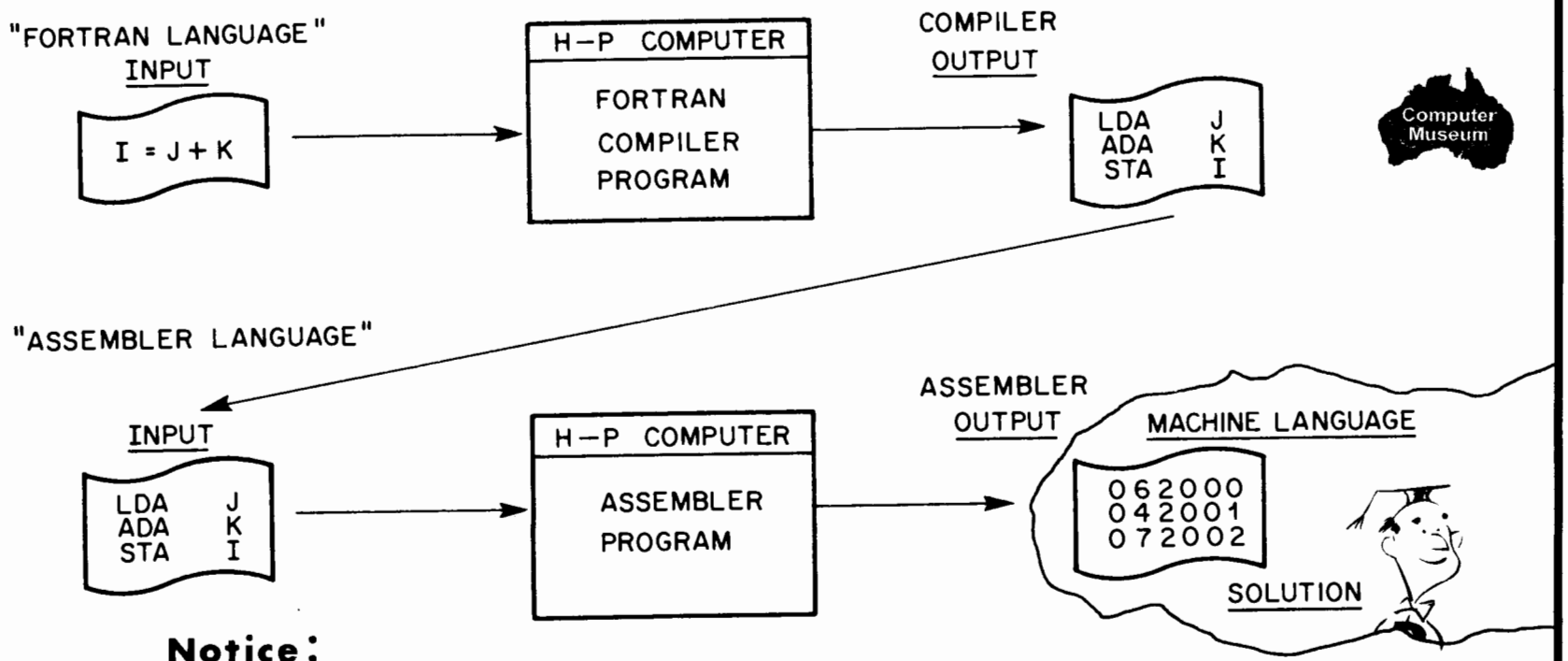
MACHINE LANGUAGE	ASSEMBLY LANGUAGE	COMPILER LANGUAGE
0110001100100001	LDA A	D=A+B+C
0100001100100010	ADA B	
0100001100100011	ADA C	
0111001100100100	STA D	

THE ASSEMBER PERFORMS THIS TRANSLATION *(ONE FOR ONE)*

THE COMPILER PERFORMS THIS TRANSLATION *(MANY FOR ONE)*

COMPILERS AND ASSEMBLERS

THE FORTRAN SOLUTION TO A TRIVIAL PROBLEM IS SHOWN. FORTRAN STANDS FOR FORMULA TRANSLATION.



Notice:

OUTPUT OF THE COMPILER BECOMES THE ASSEMBLER INPUT

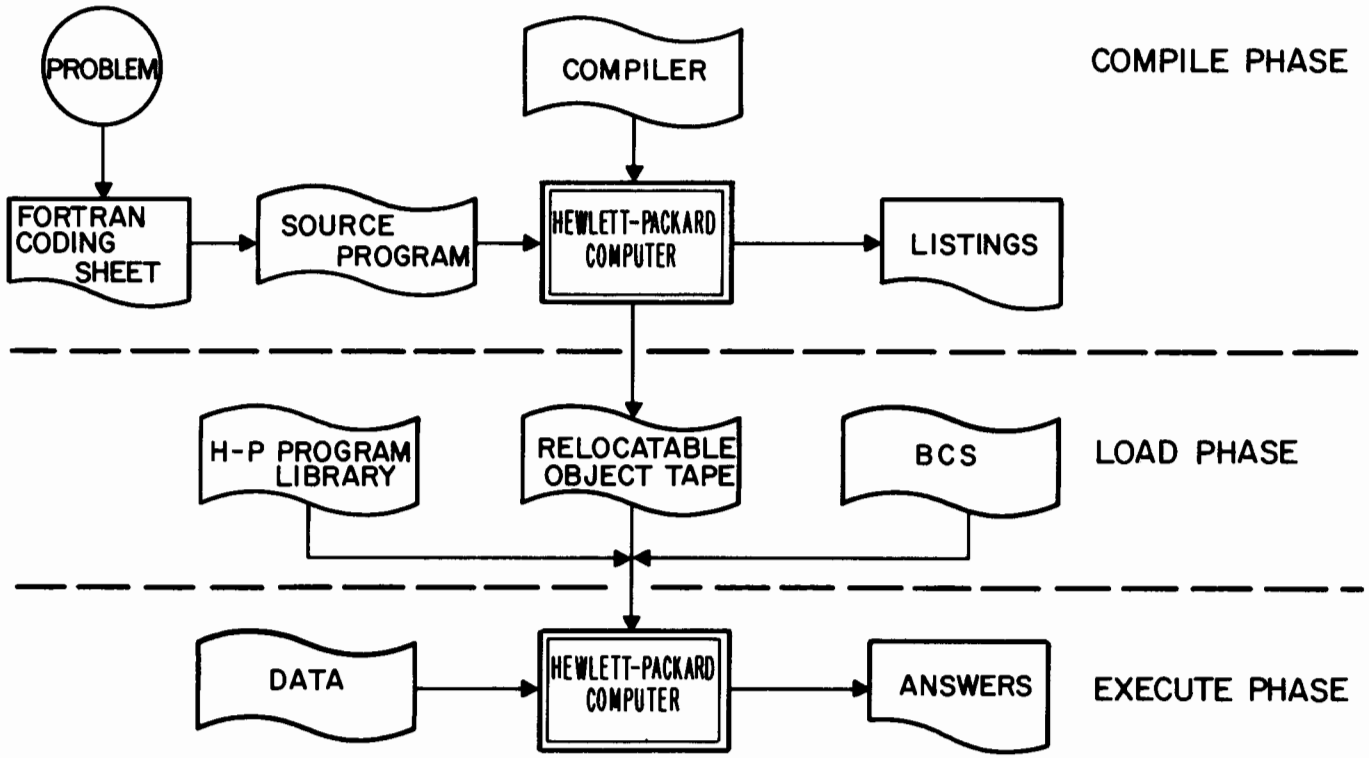
OUTPUT OF THE ASSEMBLER IS IN MACHINE LANGUAGE

A SAMPLE FORTRAN PROGRAM

FORTRAN EXAMPLE – READ IN VALUES FOR SIDES OF A RIGHT TRIANGLE. CALCULATE THE HYPOTENUSE & PRINT OUT THE VALUE ON THE TELEPRINTER.

```
FTN,L,A,B
      PROGRAM RJK
5     WRITE(2,20)
      READ(1,*)A,B
      C=SQRT(A*A+B*B)
      WRITE(2,30)C
30    FORMAT("THE HYPOTENUSE= "F8.3)
20    FORMAT("ENTER VALUES FOR A AND B")
      GO TO 5
      END
      END$
```

FORTRAN OPERATING ENVIRONMENT



BASIC CONTROL SYSTEM (BCS)

WHAT IS IT?

AN ABSOLUTE PROGRAM WHICH "CONTROLS" THE LOADING AND EXECUTION OF RELOCATABLE PROGRAMS

WHAT ARE ITS FUNCTIONS?

1. LOADS AND LINKS RELOCATABLE OBJECT PROGRAMS
2. CREATES INDIRECT & BASE PAGE ADDRESSING WHERE NECESSARY
3. SELECTS & LOADS REFERENCED LIBRARY ROUTINES
4. PROCESSES I/O REQUESTS & SERVICES I/O INTERRUPTS

WHAT ARE ITS PARTS?

1. INPUT/OUTPUT CONTROL SUBROUTINE (IOC)
2. I/O DRIVERS
3. RELOCATING LOADER

RELOCATABLE OBJECT PROGRAM

WHAT IS IT?

A PROGRAM WHICH SPECIFIES RELATIVE RATHER THAN ABSOLUTE MEMORY ADDRESSES FOR ITS INSTRUCTIONS AND DATA.

HOW IS IT LOADED?

THE BASIC CONTROL SYSTEM DETERMINES WHERE A RELOCATABLE OBJECT PROGRAM WILL BE LOADED INTO MEMORY.

THIS PAGE INTENTIONALLY
BLANK

HP PROGRAM LIBRARY

WHAT IS IT?

A GROUP OF RELOCATABLE OBJECT PROGRAM SUB-ROUTINES WHICH THE CUSTOMER MAY USE TO PERFORM SPECIFIC TASKS, THEREBY REDUCING HIS OWN PROGRAMMING EFFORT

EXAMPLES

SQRT ⇨ A Subroutine Which Takes The Square Root Of A Number

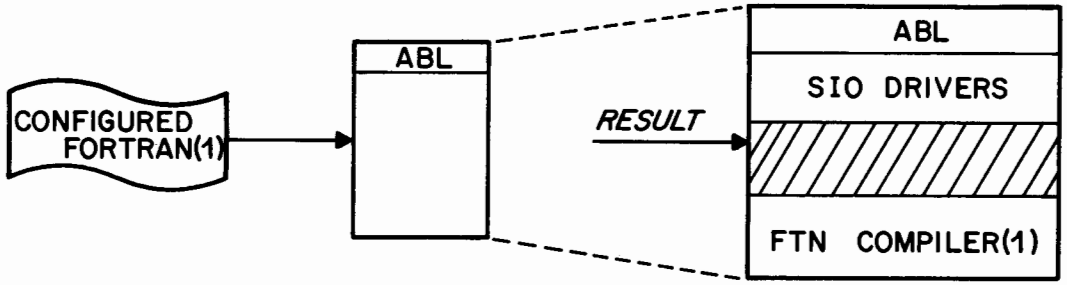
SIN ⇨ A Subroutine Which Finds The Trigonometric Sine Of A Number

ALOG ⇨ A Subroutine Which Calculates The Natural Logarithm Of A Number

THERE ARE OVER 50 OF THESE SPECIAL PURPOSE ROUTINES ON OUR PROGRAM LIBRARY TAPE.

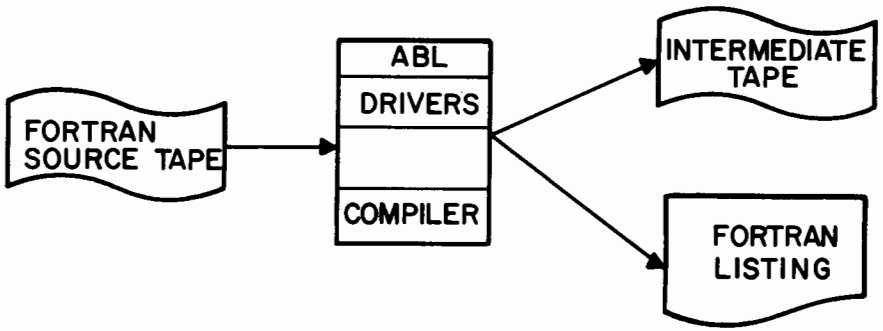
COMPILER OPERATING PROCEDURES (I)

I. LOAD THE FORTRAN COMPILER (PASS 1) TAPE USING THE ABSOLUTE BINARY LOADER.



LOAD

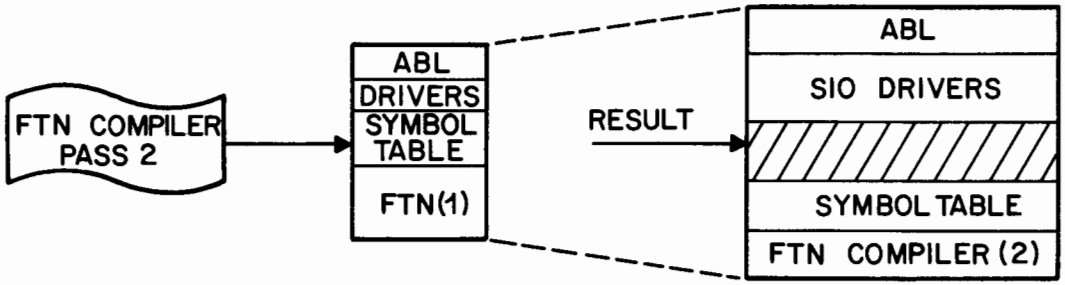
II THE COMPILER THEN PROCESSES OUR FORTRAN SOURCE TAPE AND PUNCHES OUT AN ASSEMBLY LANGUAGE VERSION ON PAPER TAPE.



PROCESS

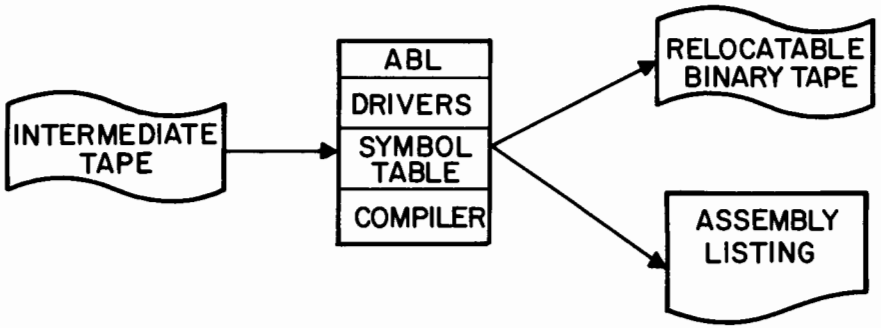
COMPILER OPERATING PROCEDURES(II)

III. LOAD THE FORTRAN COMPILER (PASS 2) TAPE USING THE ABSOLUTE BINARY LOADER



LOAD

IV. THE COMPILER THEN PROCESSES THE INTERMEDIATE TAPE & PUNCHES OUT A RELOCATABLE BINARY TAPE OF OUR PROGRAM.

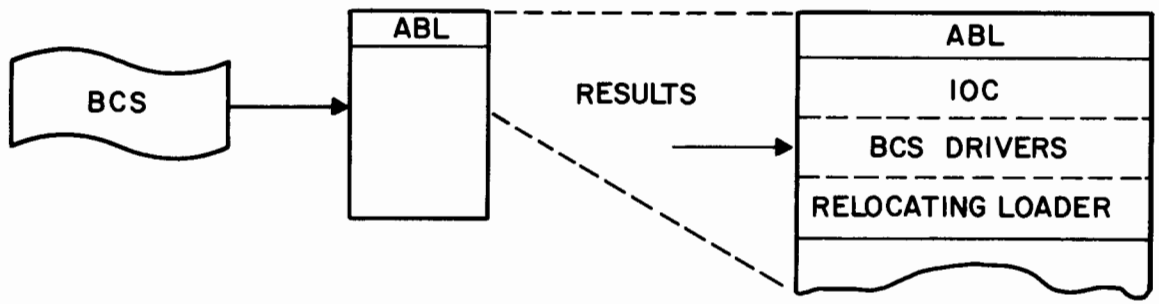


PROCESS

BCS OPERATING PROCEDURES (I)

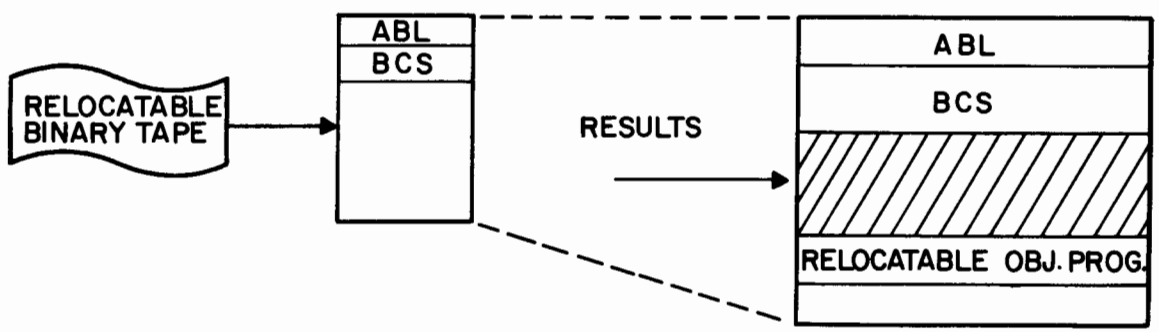
LOADING THE SYSTEM

LOAD THE BASIC CONTROL SYSTEM TAPE USING THE ABSOLUTE BINARY LOADER:



LOADING THE PROGRAM

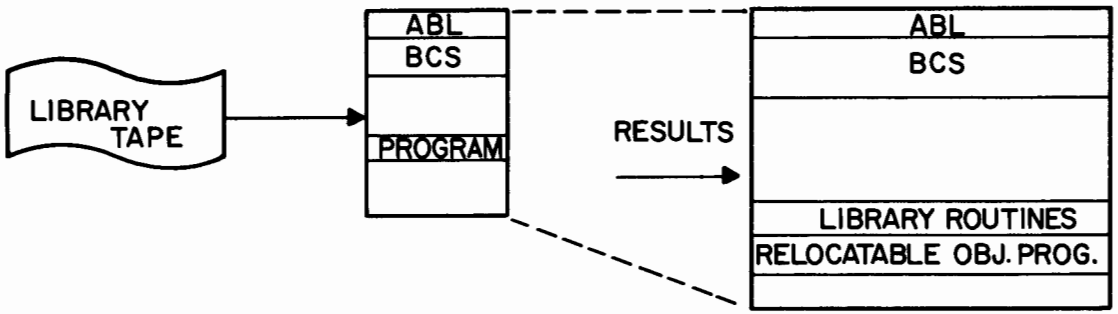
THE RELOCATING LOADER THEN LOADS OUR RELOCATABLE OBJECT TAPE STARTING AT PAGE 1; LOCATION 2000_8.



BCS OPERATING PROCEDURES (II)

LOADING THE LIBRARY ROUTINES

THE RELOCATING LOADER THEN LOADS THE LIBRARY ROUTINES CALLED BY THE USER PROGRAM.



RELOCATABLE ASSEMBLY

ASSEMBLY LANGUAGE PROGRAMS MAY ALSO BE WRITTEN IN "RELOCATABLE MODE"

RELOCATABLE

```
ASMB,R,B,L,T
      NAM RJK
        •
        •
        •
        •
        •
```

ABSOLUTE

```
ASMB,A,B,L,T
      ORG 2ØØB
        •
        •
        •
        •
        •
```

THE PROGRAM ON THE LEFT WILL PRODUCE A RELOCATABLE BINARY TAPE THAT MAY REFERENCE LIBRARY ROUTINES.

USERS WILL PRIMARILY USE THE RELOCATABLE MODE WHEN PROGRAMMING IN ASSEMBLY LANGUAGE.

BCS DRIVERS

THE INPUT/OUTPUT DRIVERS ASSOCIATED WITH THE BASIC CONTROL SYSTEM MAKE USE OF THE COMPUTER'S PRIORITY INTERRUPT FEATURE.

IN ORDER TO UNDERSTAND MORE ABOUT THE OPERATION OF THE BASIC CONTROL SYSTEM, IT IS ESSENTIAL THAT WE EXAMINE THE INTERRUPT STRUCTURE IN SOME DETAIL.

NON-INTERRUPT METHOD REVIEW

THE USER COMMANDS THE I/O DEVICE TO CYCLE AND THEN PROGRAMS A LOOP THAT "WAITS" FOR THE DEVICE CYCLE TO COMPLETE.*

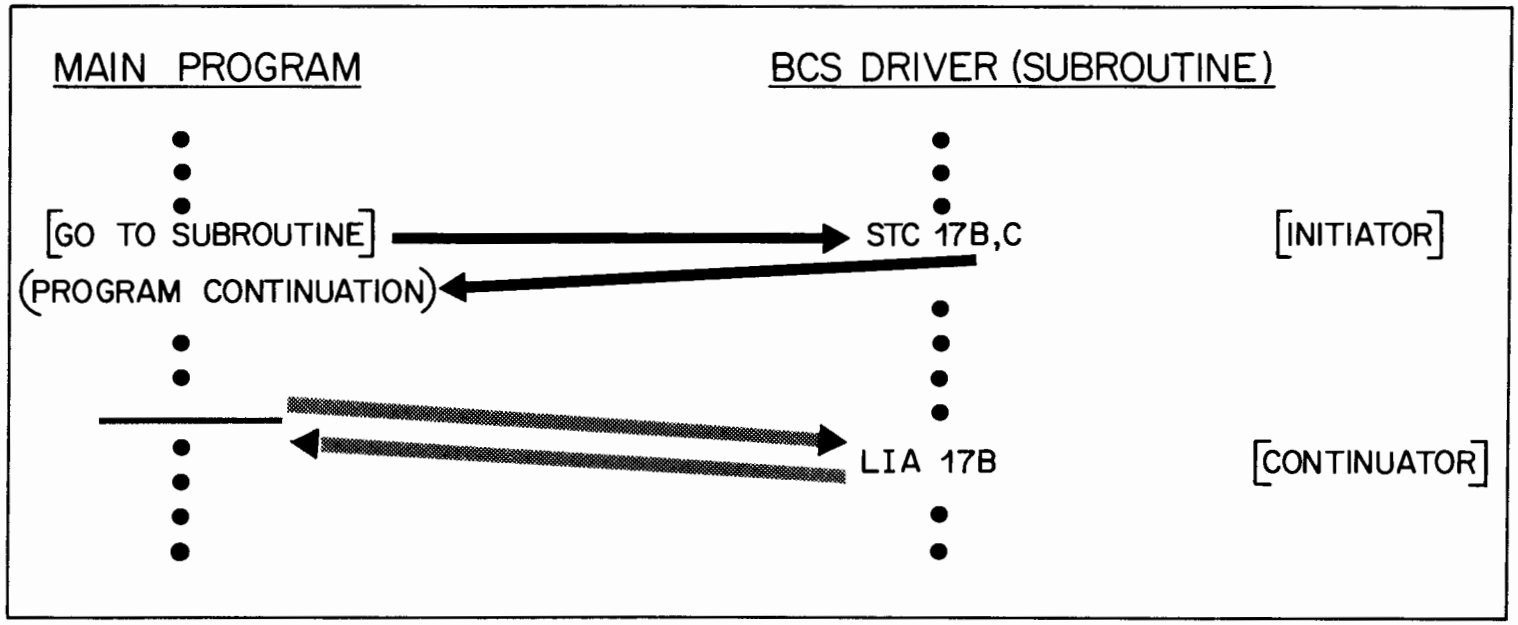
EXAMPLE:

```
STC  17B, C      START READER
SFS  17B         IS THE FLAG SET
JMP  *-1         NO STAY IN LOOP
LIA  17B         YES LOAD IN DATA
```

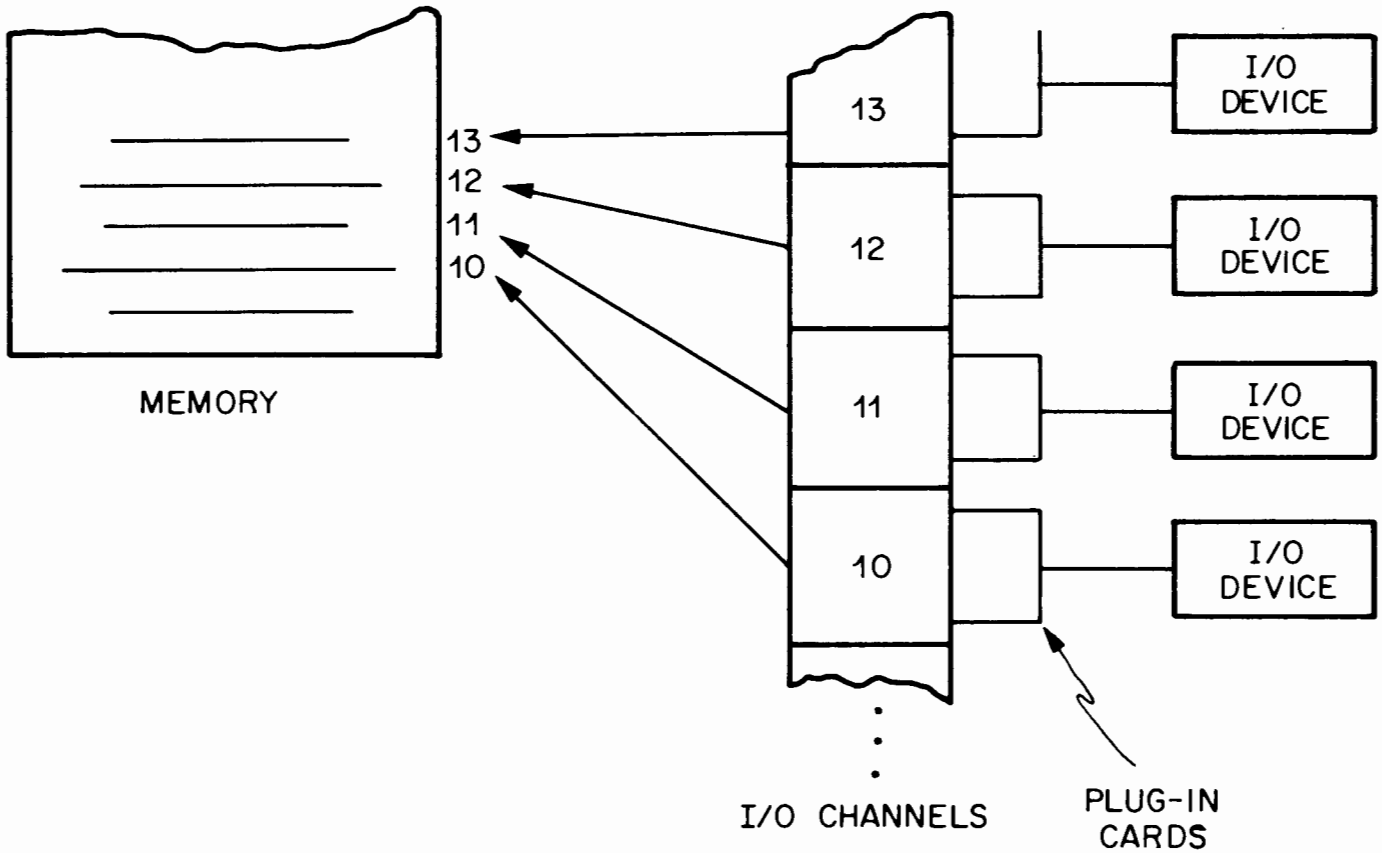
REMEMBER, THIS IS THE METHOD USED BY SIO DRIVERS

INTERRUPT METHOD

THE USER COMMANDS THE I/O DEVICE TO CYCLE AND THEN CONTINUES EXECUTION OF THE MAIN PROGRAM. THE COMPLETION OF THE DEVICE CYCLE WILL INTERRUPT THE MAIN PROGRAM AND TRANSFER CONTROL TO A SUBROUTINE THAT WILL HANDLE THE ACTUAL DATA TRANSFER.



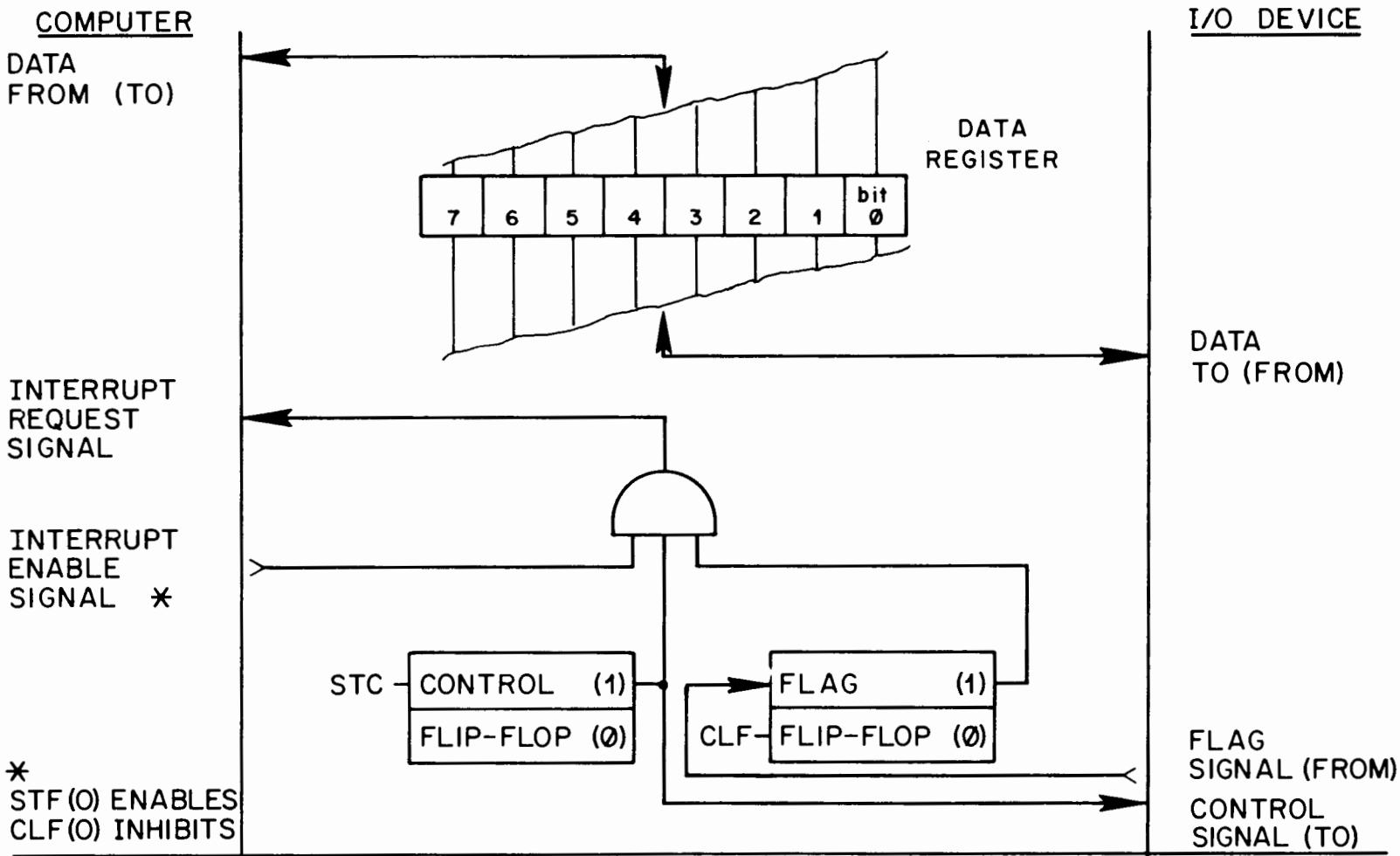
SELECT CODES AND INTERRUPT ADDRESSES



SELECT CODE ASSIGNMENTS

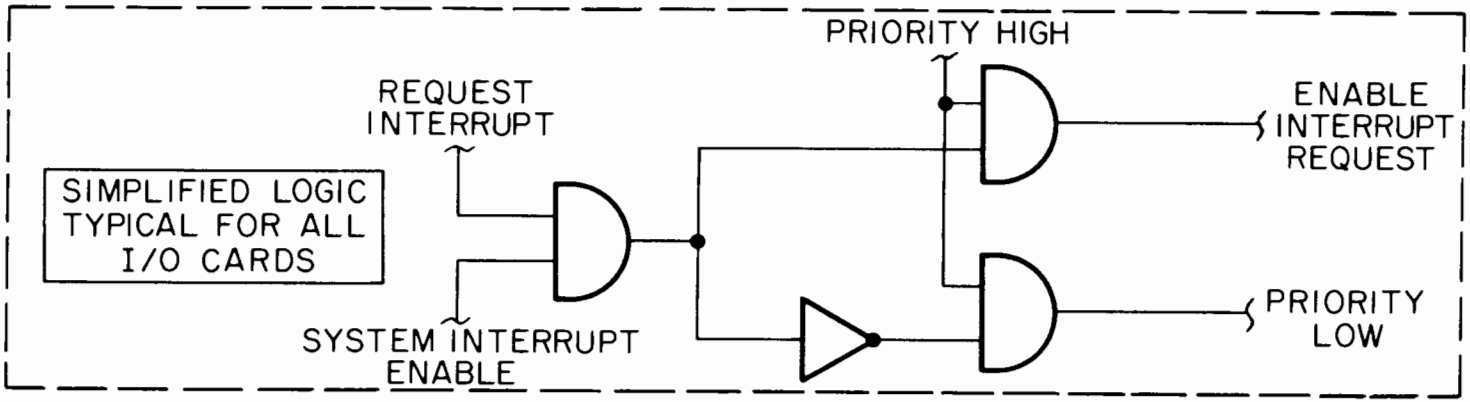
SELECT CODE	INTERRUPT LOCATION	FUNCTIONAL ASSIGNMENTS
0	NONE	ENABLE/DISABLE I/O AND INT. SYST.
1	NONE	SWITCH REGISTER
2	NONE	DMA CH 1
3	NONE	DMA CH 2
4	4 –	POWER FAIL
5	5 –	MEMORY PROTECT
6	6 –	DMA CH1
7	7 –	DMA CH2
10	10 –	I/O DEVICE HIGHEST PRIORITY
.	.	. .
.	.	. .
.	.	. .
.	.	. .
77	77	I/O DEVICE LOWEST PRIORITY

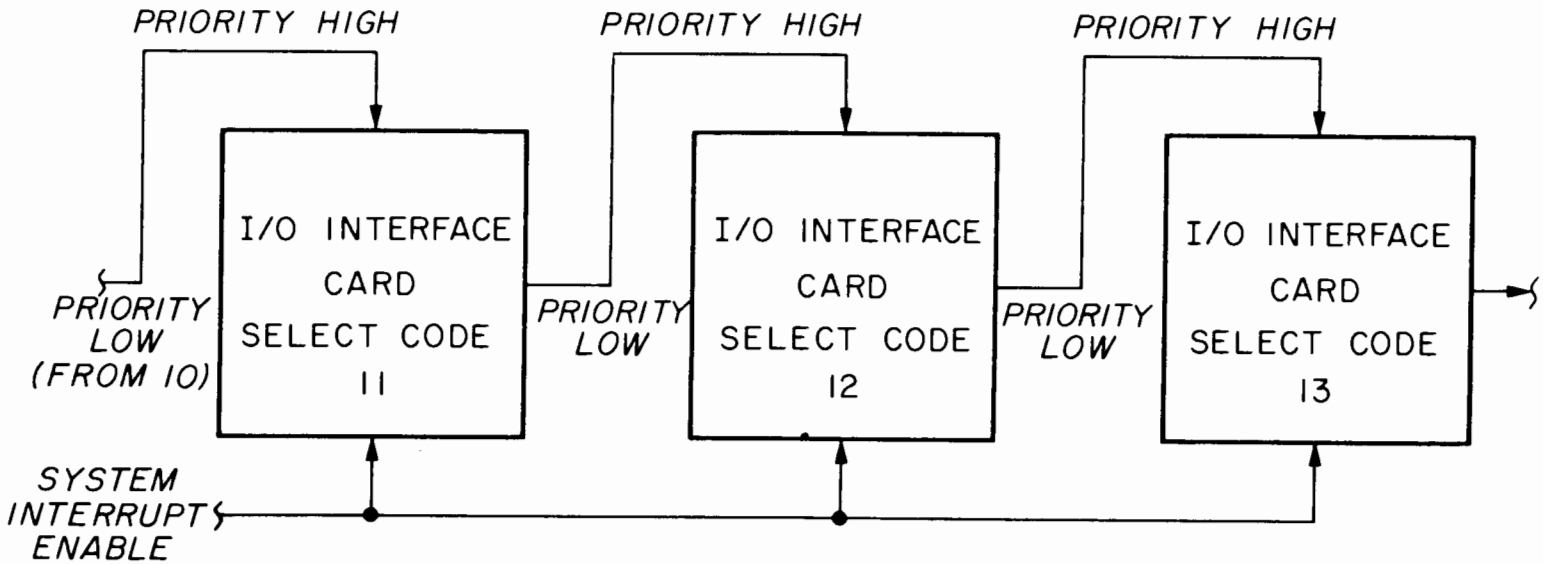
A TYPICAL I/O INTERFACE CARD



mnemonic	instruction	function
STC (sc)	SET CONTROL FLIP-FLOP	START DEVICE
CLC (sc)	CLEAR CONTROL " "	CLEAR DEVICE
STF (sc)	SET FLAG " "	SET FLAG, "READY"
CLF (sc)	CLEAR FLAG " "	CLEAR FLAG, "BUSY"

SIMPLIFIED PRIORITY LOGIC





INTERRUPT PHASE - HOW IT WORKS

I. WHEN AN INTERRUPT SIGNAL IS ACKNOWLEDGED BY THE PROCESSOR, PHASE 4 (INTERRUPT PHASE) IS SET, CAUSING THE FOLLOWING TO HAPPEN:

1. P-REGISTER IS DECREMENTED (P-1)
2. M-REGISTER IS CLEARED
3. M-REGISTER IS SET TO THE SELECT CODE OF THE INTERRUPTING DEVICE

II. AT THE COMPLETION OF THE INTERRUPT PHASE THE FETCH PHASE IS SET, CAUSING THE COMPUTER TO FETCH THE INSTRUCTION STORED IN MEMORY LOCATION SPECIFIED BY THE M-REGISTER [INTERRUPT LOCATION]

INTERRUPT PHASE-EXAMPLE

```
④ 0010      INB

① 1400      STF  0
  1401      STC  10B,C

② 2000      CLB
  2001      LDA  X
  2002      ADA  Y

  2500  X   OCT  30
  2501  Y   OCT  20
```

INTERRUPT

③

AT MOMENT OF INTERRUPT

P = 2001
M = 2500
B = 0000

IN THE INTERRUPT PHASE

P→2000
M→10
FETCH PHASE IS THEN SET

AT CONCLUSION OF FETCH PHASE

B = 0001
P = 2001
M = 2001

---

① *PROGRAM STARTING POINT*
② *POINT OF INTERRUPT*
③ *REGISTER CONTENTS*
④ *INTERRUPT LOCATION*

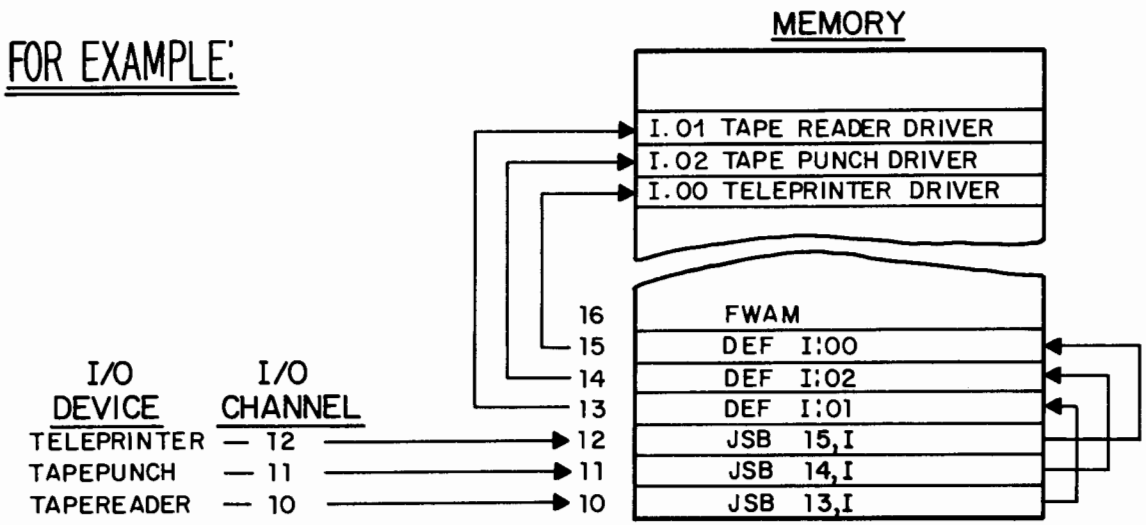
INTERRUPT OPERATION COMPLETE

THE COMPUTER THEN FETCHES THE INSTRUCTION IN 2001 AND CONTINUES THE EXECUTION OF THE MAIN PROGRAM.

INTERRUPT LINKAGE

THE INSTRUCTION STORED IN THE INTERRUPT LOCATION IS ONE WHICH WILL TRANSFER CONTROL TO THE CONTINUATOR SECTION OF THE I/O DRIVER ASSOCIATED WITH THE DEVICE. SINCE ALL INTERRUPT LOCATIONS ARE ON THE BASE PAGE & THE I/O DRIVERS ARE IN HIGH MEMORY, THE TRANSFER TO THE DRIVER MUST USE INDIRECT ADDRESSING.

FOR EXAMPLE:



PREPARE CONTROL SYSTEM (P.C.S.)

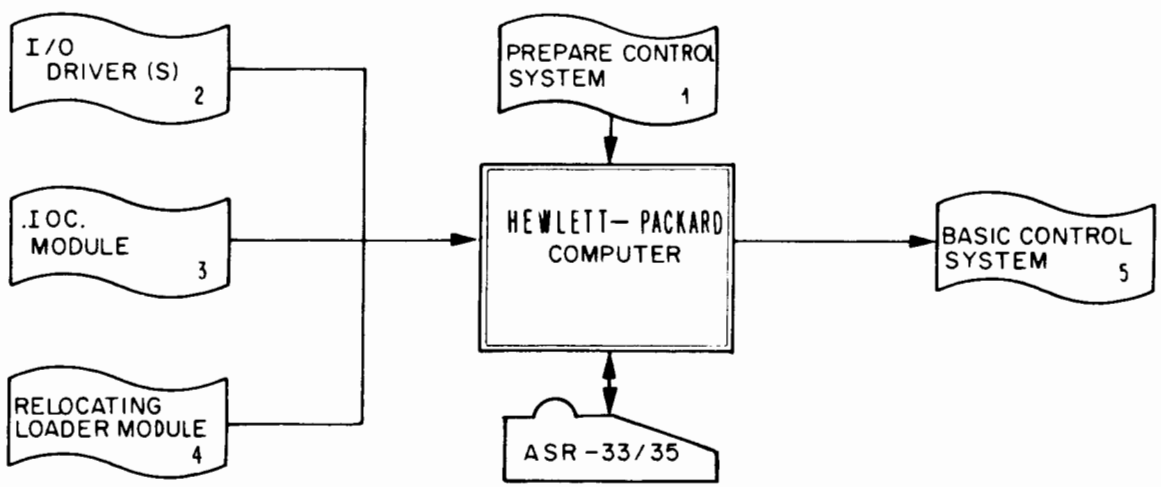
WHAT IS IT?

A COMPUTER PROGRAM WHICH PROCESSES RELOCATABLE MODULES OF THE BASIC CONTROL SYSTEM AND PRODUCES AN ABSOLUTE VERSION OF B.C.S. TAILORED TO THE SPECIFIC HARDWARE CONFIGURATION.

WHAT DOES IT DO?

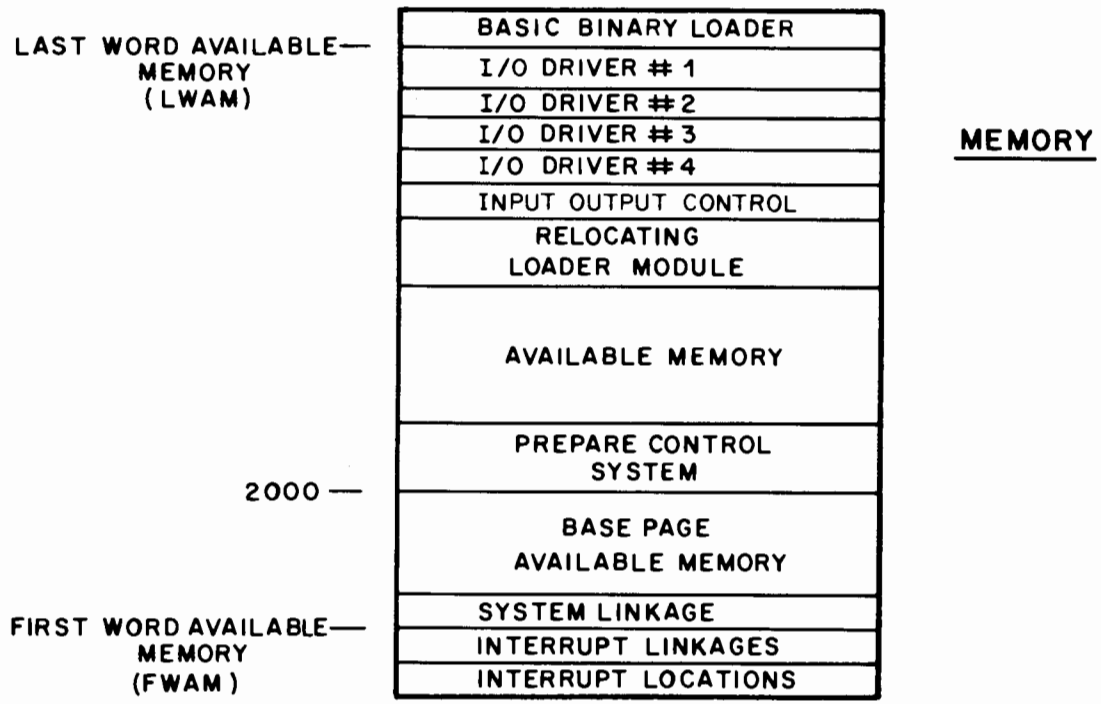
IT CREATES AN OPERATING SYSTEM CONSISTING OF THE INPUT/OUTPUT SUBROUTINE (I.O.C.), THE RELOCATABLE LOADER (LDR) AND THE REQUIRED PERIPHERAL EQUIPMENT INPUT/OUTPUT DRIVER SUBROUTINES.

PROCESSING ENVIRONMENT



P.C.S. OVER VIEW

P.C.S PROVIDES THE CAPABILITY OF CREATING A COMPLETE BASIC CONTROL SYSTEM IN THE COMPUTERS MEMORY.



WHEN ALL INDIVIDUAL ELEMENTS ARE PRESENT IN MEMORY. P.C.S. WILL PUNCH AN ABSOLUTE BINARY VERSION OF THE COMPLETE BASIC CONTROL SYSTEM.

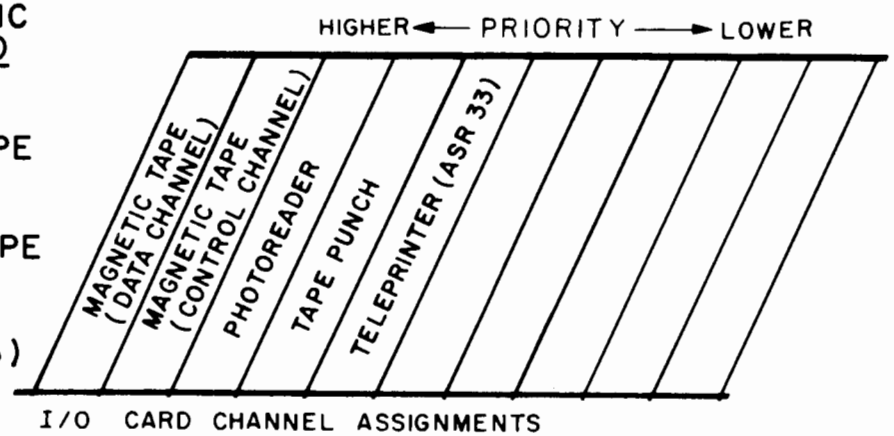
PLANNING THE SYSTEM

THE FIRST CONSIDERATION TO BE MADE IS THE PHYSICAL PLACEMENT OF THE I/O INTERFACE CARDS. CHANNEL #10 HAS THE HIGHEST PRIORITY, #11 NEXT HIGHEST, ETC. GENERALLY, THE DEVICE THAT GENERATES THE GREATEST NUMBER OF INTERRUPTS PER UNIT OF TIME IS ASSIGNED THE HIGHEST PRIORITY.

FOR EXAMPLE:

ASSUME A COMPUTER SYSTEM IS MADE UP OF THE FOLLOWING UNITS:

1. - READ/WRITE MAGNETIC TAPE (REQUIRES TWO INTERFACE BOARDS)
2. - HIGH-SPEED PAPER TAPE READER
3. - HIGH-SPEED PAPER TAPE PUNCH
4. - TELEPRINTER (ASR-33)



I/O CARD CHANNEL ASSIGNMENTS

EQUIPMENT TABLE NUMBERS

EQUIPMENT TABLE NUMBERS BEGIN WITH 7. EACH DEVICE IS ASSIGNED A SEQUENTIAL OCTAL NUMBER. WITHIN THIS FRAMEWORK THE INITIAL NUMBER ASSIGNMENTS ARE ARBITRARY.

FOR EXAMPLE:

EQT

1st ENTRY - PHOTOREADER
2nd ENTRY - TAPE PUNCH
3rd ENTRY - TELEPRINTER
4th ENTRY - MAG TAPE

	MAGNETIC TAPE (DATA CHANNEL)	MAGNETIC TAPE (CONTROL CHANNEL)	PHOTOREADER	TAPE PUNCH	TELEPRINTER (ASR 33)				
EQUIPMENT TABLE →	12		7	10	11				

INTERRUPTS, LINKAGE, DRIVER I.D.

INTERRUPT LOCATION – P.C.S. WILL CAUSE A COMPUTER INSTRUCTION TO BE STORED HERE. (USUALLY A JSB, I)

LINKAGE LOCATION – P.C.S. WILL CAUSE THE ADDRESS OF THE CONTINUATOR SECTION OF THE I/O DRIVER TO BE STORED HERE.

DRIVER IDENTIFICATION – THE SYMBOLIC NAME OF THE I/O DRIVER INITIATOR SECTION ENTRY POINT.

INTERRUPT IDENTIFICATION – THE SYMBOLIC NAME OF THE I/O DRIVER CONTINUATOR SECTION ENTRY POINT.

FWAM – THE FIRST WORD OF AVAILABLE MEMORY.

NOTE: DRIVER AND INTERRUPT ID CODES ASSIGNED BY H-P. THE SYMBOLS USED MUST BE UNIQUE.

	MAG TAPE (DATA CHANNEL)	MAG TAPE (CONTROL CHANNEL)	PHOTOREADER	TAPE PUNCH	TELEPRINTER (ASR 33)
DRIVER IDENTIFICATION	D.21	–	D.01	D.02	D.00
INTERRUPT LOCATION	10	11	12	13	14
LINKAGE LOCATION	15	16	17	20	21
INTERRUPT IDENT.	I.21	C.21	I.01	I.02	I.00
EQUIPMENT TABLE →	12		7	10	11

(22) FWAM

STANDARD UNIT NUMBERS

The standard unit numbers are simply pointers to the appropriate equipment table entries.

To assign standard units place a checkmark at the intersection of the standard unit table number (x-axis), and the correct equipment table number (y-axis)

	MAG TAPE (DATA CHANNEL)	MAG TAPE (CONTROL CHANNEL)	PHOTOREADER	TAPE PUNCH	TELEPRINTER (ASR 33)
DRIVER IDENTIFICATION	D.21	–	D.Ø1	D.Ø2	D.ØØ
INTERRUPT LOCATION	10	11	12	13	14
LINKAGE LOCATION	15	16	17	20	21
INTERRUPT IDENT.	I.21	C.21	I.Ø1	I.Ø2	I.ØØ
EQUIPMENT TABLE →	12		7	10	11

	STANDARD UNIT TABLE					
1.	KEYBOARD INPUT					✓
2.	TELEPRINTER OUTPUT					✓
3.	PROGRAM LIBRARY			✓		
4.	PUNCH OUTPUT				✓	
5	INPUT			✓		
6.	LIST OUTPUT					✓

P.C.S. OPERATIONS

THE NEXT FEW CHARTS WILL DESCRIBE A SIMPLE B.C.S. CONFIGURATION. THE SYSTEM WILL CONSIST OF A COMPUTER SYSTEM WITH 8K OF MEMORY AND THE FOLLOWING PERIPHERALS:

1. READ/WRITE MAGNETIC TAPE — I/O CHANNELS 10,11
2. PHOTOELECTRIC PUNCHED PAPER TAPE READER — I/O CHANNEL 12
3. HIGH SPEED PAPER TAPE PUNCH — I/O CHANNEL 13
4. TELEPRINTER (ASR 33) — I/O CHANNEL 14

THE ACTUAL CONFIGURATION PROCESS MAY BE DESCRIBED IN FIVE PHASES.

PHASE 1– INITIALIZATION

PHASE 2– LOADING THE I/O EQUIPMENT DRIVER

PHASE 3– LOADING THE IOC MODULE

a. CREATING THE EQUIPMENT TABLE
b. CREATING THE STANDARD UNIT TABLE

PHASE 4– LOADING THE RELOCATING LOADER MODULE

a. ESTABLISH THE INTERRUPT LINKAGES

PHASE 5– PUNCH THE ABSOLUTE OUTPUT TAPE

INITIALIZATION PHASE

THE P.C.S. PROGRAM INITIALIZATION PHASE

COMMUNICATIONS	REMARKS
HS INP?	Is H.S. input unit available ?
17	Channel number of photo-reader
HS PUN?	Is H.S punch available ?
20	Channel number of tape punch
FWA MEM?	Request first word address of available memory
22	First word following required interrupt locations
LWA MEM?	Request last word address of available memory
17677	Word preceding basic loader (8K memory)
* LOAD	Request to load first BCS module

THESE ENTRIES REFER TO THE "CONFIGURING" SYSTEM.

LOADING THE I/O EQUIPMENT DRIVERS

COMMUNICATIONS		REMARKS
D.21 16220 17677	→	MAGNETIC TAPE DRIVER PROCESSED* MEMORY BOUNDS OF THE DRIVER
* LOAD	→	REQUEST TO LOAD NEXT MODULE
D.Ø1 15661 16217		PHOTO-READER DRIVER PROCESSED
* LOAD		
D.Ø2 15351 15660		TAPE PUNCH DRIVER PROCESSED
* LOAD		
D.ØØ 14615 15350		TELEPRINTER DRIVER PROCESSED
* LOAD		

** WHEN PRESENT, THIS DRIVER SHOULD BE LOADED FIRST DUE TO ITS LARGE SIZE*

LOADING THE IOC MODULE

COMMUNICATIONS		REMARKS
IOC 14376 14614	→	IOC MODULE PROCESSED
* TABLE ENTRY	→	EQUIPMENT TABLE
EQT?		UNIT #
12,D.01	→	7
13,D.02	→	10
14,D.00	→	11
10,D.21	→	12
/E		
SQT?	→	STANDARD UNIT TABLE
-KYBD? 11		
-TTY? 11		
-LIB? 7		
-PUNCH? 10		
-INPUT? 7		
-LIST? 11		
DMA? 0	→	DIRECT MEMORY ACCESS OPTION. 0 INDICATES DMA NOT AVAILABLE.
* LOAD		

	MAGNETIC TAPE (DATA CHANNEL)	MAGNETIC TAPE (CONTROL CHANNEL)	PHOTOREADER	TAPE PUNCH	TELEPRINTER (ASR 33)
DRIVER IDENTIFICATION	D.21	—	D.01	D.02	D.00
INTERRUPT LOCATION	10	11	12	13	14
LINKAGE LOCATION	15	16	17	20	21
INTERRUPT IDENT.	I.21	C.21	I.01	I.02	I.00
EQUIPMENT TABLE →	12		7	10	11

	STANDARD UNIT TABLE					
1.	KEYBOARD INPUT					✓
2.	TELEPRINTER OUTPUT					✓
3.	PROGRAM LIBRARY			✓		
4.	PUNCH OUTPUT				✓	
5	INPUT			✓		
6.	LIST OUTPUT					✓

THE RELOCATING LOADER MODULE

DRIVER IDENTIFICATION	D.21	–	D.Ø1	D.Ø2	D.ØØ
INTERRUPT LOCATION	1Ø	11	12	13	14
LINKAGE LOCATION	15	16	17	2Ø	21
INTERRUPT IDENT.	I.21	C.21	I.Ø1	I.Ø2	I.ØØ

COMMUNICATIONS

LOADR
12115 14346

INTERRUPT LINKAGE?

10,15,I.21
11,16,C.21
12,17,I.01 (ERROR 0≠0)
*UN NAME
12,17,I.01
13,20,I.02
14,21,I.00
/E → LIST TERMINATOR

MEANING	ENTRY POINT LIST	
ADDRESS OF SYSTEM TABLE	.SQT.	14347
ADDRESS OF EQUIP. TABLE	.EQT.	14355
I/O DRIVER – INITIATOR AND CONTINUATOR ENTRY POINTS	D.21	16220
	I.21	17216
	C.21	17130
	D.Ø1	15661
	I.Ø1	15776
	D.Ø2	15351
	I.Ø2	15465
MAINTAINS COMPATABILITY BETWEEN BUFFERED AND UNBUFFERED VERSIONS OF I.O.C.	.BUFR	14544
	D.ØØ	14615
	I.ØØ	14771
ADDRESS OF .IOC. ENTRY POINT	.IØC.	14376
DMA STATUS WORD CH #1	DMAC1	14613
DMA STATUS WORD CH #2	DMAC2	14614
ADDRESS OF I/O ERROR HALT	IOERR	14572
SYSTEM TABLE LINK WORD	XSQT	14611
EQUIPMENT TABLE LINK WORD	XEQT	14612
RELOCATING LOADER ENTRY POINT	.LDR.	13601
ADDRESS OF MEMORY TABLE*	.MEM.	14342
LOADER SYMBOL TABLE ADDRESS	LST	12141

*SYSTEM LINK
ØØØ22 ØØ153

* MEMORY TABLE
FWABP _____
LWABP _____
FWAM _____
LWAM _____

*BCS ABSOLUTE OUTPUT