W. ZOLNEROVICH

INTEL MCS-80

MICROCOMPUTER WORKSHOP

—STUDENT STUDY GUIDE—

PREFACE

THIS STUDENT STUDY GUIDE IS FOR PARTICIPANTS IN THE MCS-80 MICROCOMPUTER WORKSHOP.

THE GUIDE CONSISTS OF THE FOLLOWING SECTIONS:

SECTION I - INTRODUCTION

OBJECTIVES, MATERIALS LIST

SECTION II - VISUALS

A COPY OF ALL VISUAL AIDS

SECTION III - EXERCISES AND LABORATORY PROJECTS

SECTION IV - REFERENCE MATERIALS

APPENDICES AND MISCELLANEOUS ITEMS

SECTION II

VISUALS

CONTENTS

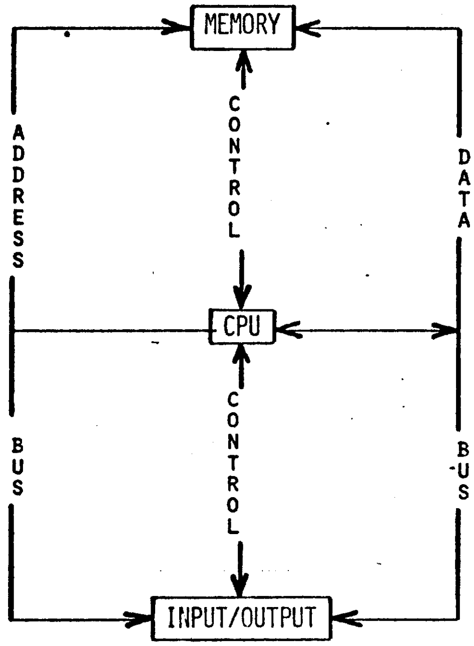
TITLE		PAGE #
Part I	System Introduction	2-1
Part II	Assembly Language Instructions	2-9
	A. •INR/DCR •MOV •LXI,MVI •ORG •HLT •IN/OUT •JMP •END	
	B. •ADD •SUBTRACT •LOGICALS •CARRY •ROTATE •DEFINE STORAGE	
Part III	Microcomputer Development System	2-29
	A. Monitor	
	B. Editor	
	C. Assembler	
Part IV	Basic Hardware	2-41
Part V	Assembly Language Instructions	2-53
	A. •CALL/RET •PUSH/POP •DB	
	B. •RST •DAA/DAD •XCHG/XTHL •STAX •STA,SHLD •DW	
	C. Macros	
Part VI	Peripherals and Design	2-79
	A. 8224	
	B. 8228	
	C. MEMORY	
	D. 8255	
	E. 8251	
	F. 8214	
	G. 8257	
Part VII	ICE 80	2-109

PART I

SYSTEM INTRODUCTION

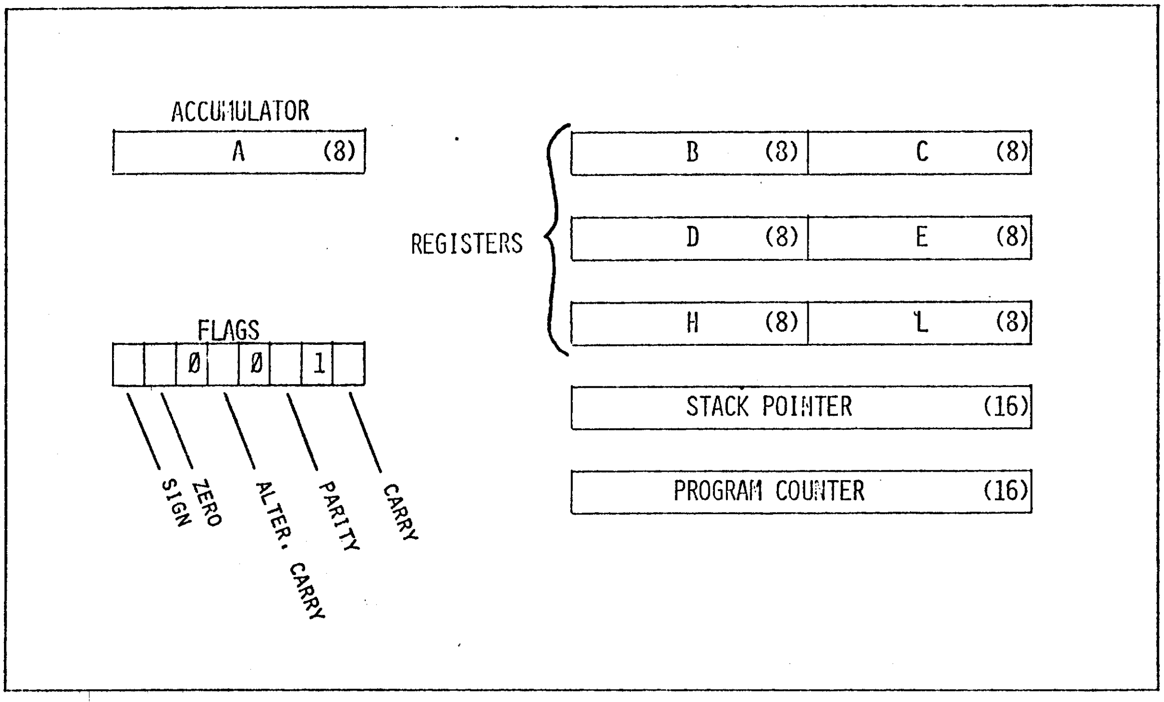
" NOTES "

--- FUNCTIONAL SECTIONS ---



- PROGRAMS
  ADDRESS STACK
  DATA
- OPERATIONS
  DECISIONS
- EXTERNAL
  COMMUNICATION

CPU (PROGRAMMING MODEL)



PROGRAM COUNTER

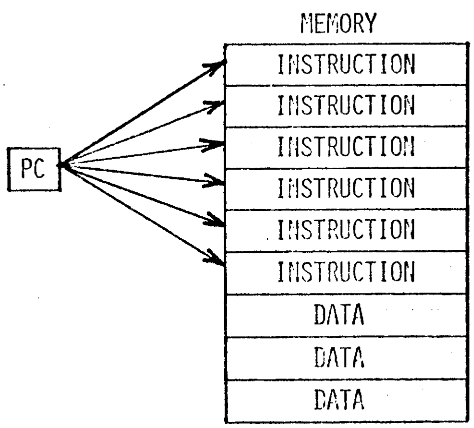
POINTS TO

MEMORY

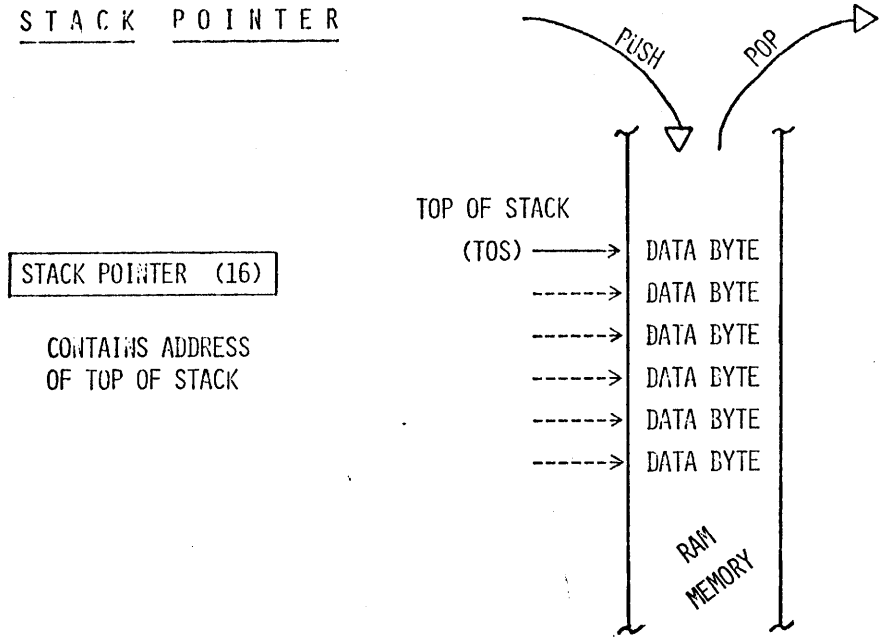
LOCATIONS

CONTAINING

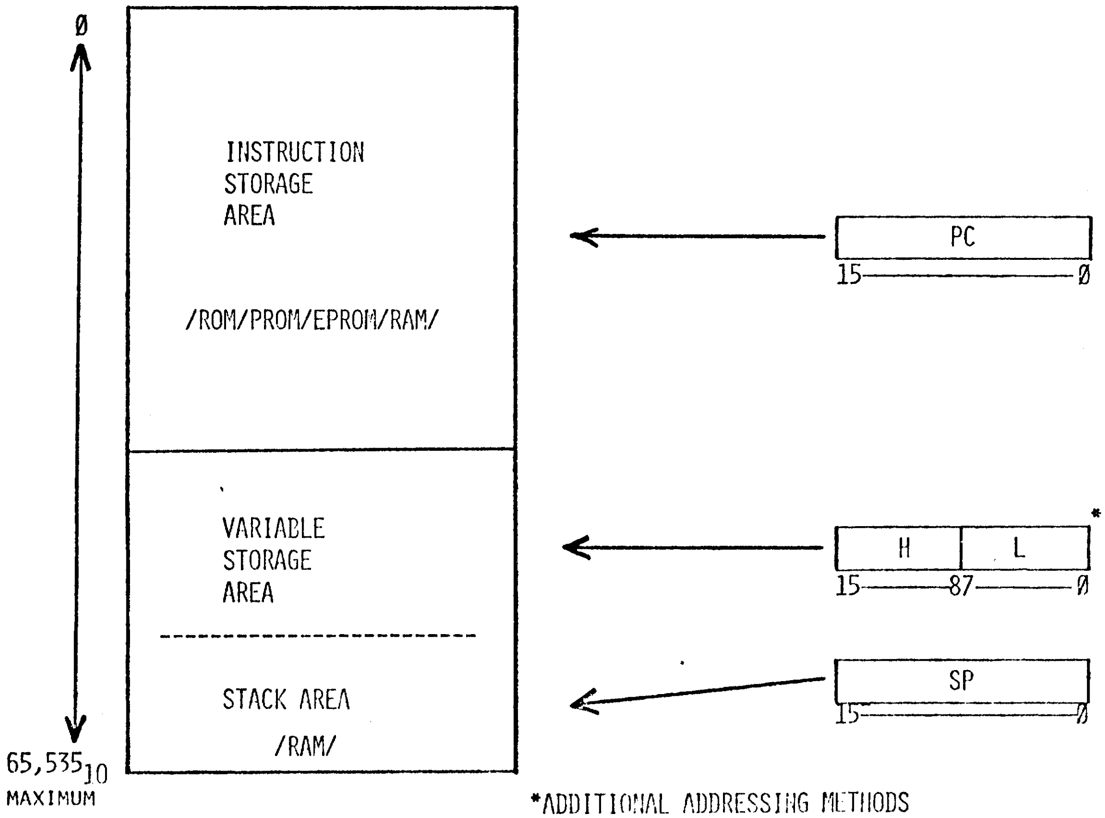
INSTRUCTIONS



STACK POINTER



TYPICAL MEMORY LAYOUT



*ADDITIONAL ADDRESSING METHODS

MACHINE INSTRUCTIONS

TYPES:

- REGISTER
- ARITHMETIC
- LOGICAL
- INPUT/OUTPUT
- CONTROL

INSTRUCTIONS MAY BE ONE, TWO OR THREE BYTES LONG.

EXAMPLES:

MOV C,A

MOV C,A

ADI 7

ADI
7

JMP 4296H

JMP
96
42

EXECUTION SEQUENCE

PROGRAM SEGMENT

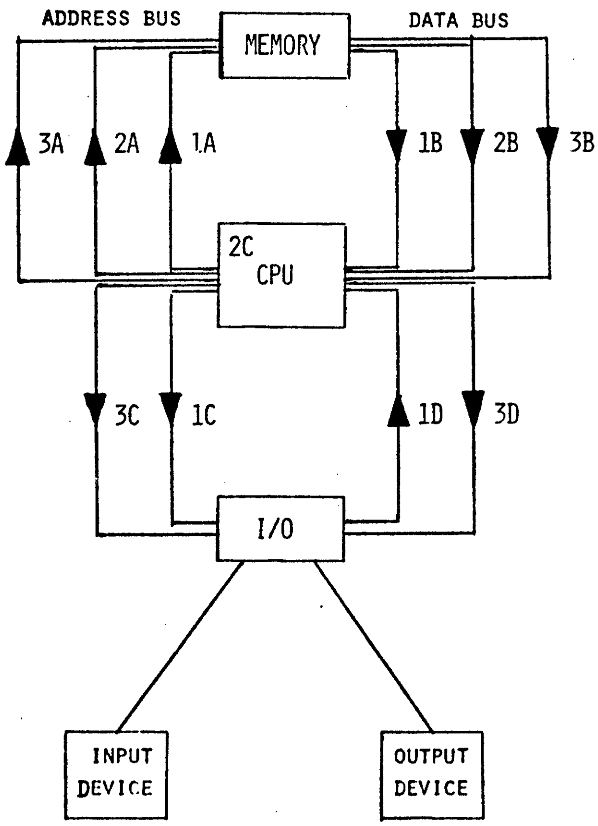
1 - INPUT VALUE

2 - ADD SEVEN

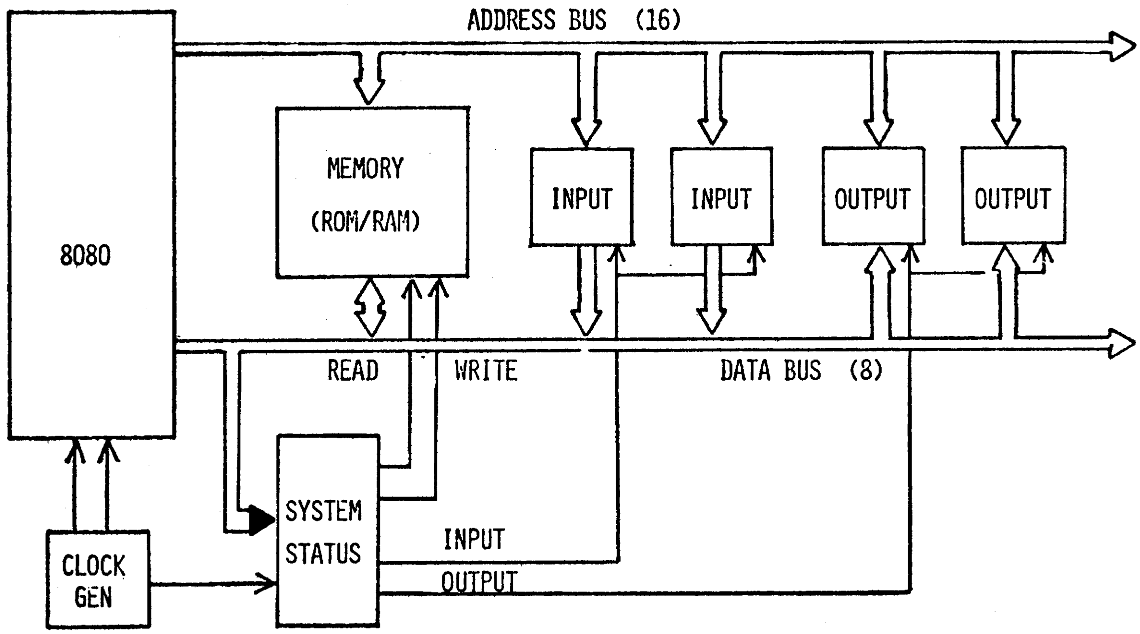
3 - OUTPUT VALUE

STEP NUMBER

1A,B,C,D	IN	4
2A,B,C	ADI	7
3A,B,C,D	OUT	2



SYSTEM INTRODUCTION



" NOTES "

PART II

ASSEMBLY LANGUAGE INSTRUCTION

NUMBER SYSTEMS

- DECIMAL (10 DIGITS, Ø THRU 9)

  109_{10} MAY BE REPRESENTED AS:

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

  $100 + Ø + 9 = 109_{10}$

- BINARY (2 DIGITS, Ø AND 1)

  01101101_2 MAY BE REPRESENTED AS:

  $(0 \times 2^7) + (1 \times 2^6) + (1 \times 2^5) + (0 \times 2^4) + (1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0)$

  $0 + 64 + 32 + 0 + 8 + 4 + 0 + 1$

THUS, $109_{10} = 01101101_2$

HEXADECIMAL SYSTEM

- 16 DIGITS, Ø THRU F

HEX	BINARY	DECIMAL
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15

15B3H MAY BE REPRESENTED AS:

$(1 \times 16^3) + (5 \times 16^2) + (B \times 16^1) + (3 \times 16^0)$

$4096 + 1280 + 176 + 3$

$15B3H = 5555_{10}$

LANGUAGES

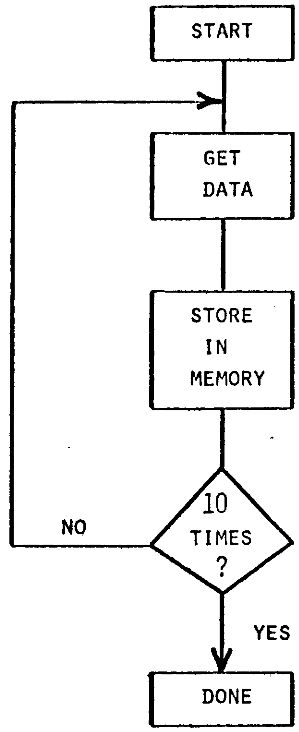
PROBLEM:

SELECT THE LARGER OF TWO NUMBERS STORED IN MEMORY AND STORE IT IN A THIRD LOCATION

```
MACHINE CODE          ASSEMBLY              PL/M

    3A
    FF
    01
    21
    D3                LDA    Y
    00                LXI    H,X
    BE                CMP    M              IF X > Y THEN Z = X ;
    DA                JC     GO                       ELSE Z = Y ;
    43                MOV    A,M
    01           GO:  STA    Z
    7E
    32
    88
    01
```

PROBLEM



```
IN      3
```

```
IN      3
MOV     M,A
```

```
LXI     H,100H
IN      3
MOV     M,A
```

```
LXI     H,100H
IN      3
MOV     M,A
INX     H
```

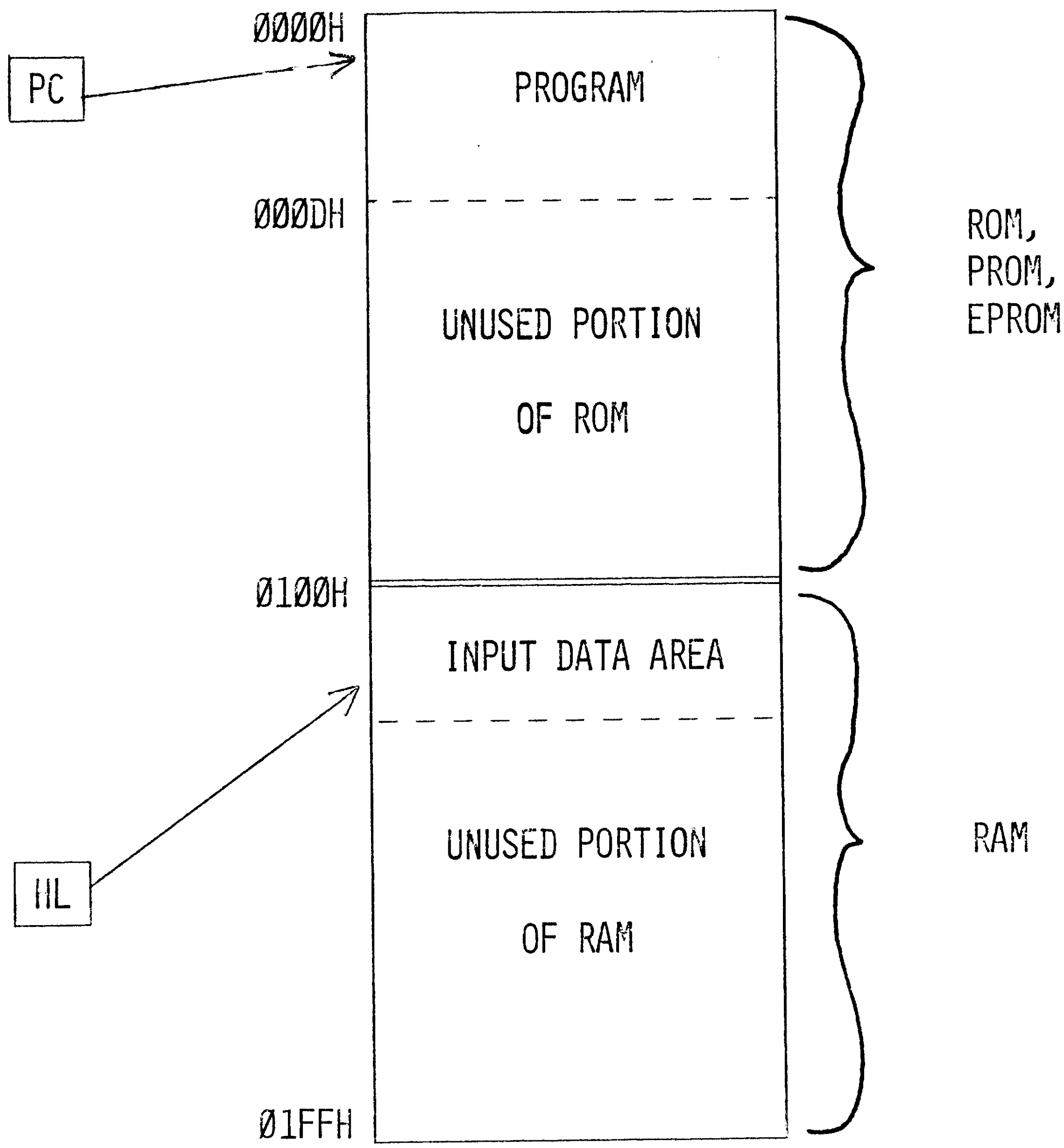
```
        LXI   H,100H
LOOP:   IN    3
        MOV   M,A
        INX   H
        JMP   LOOP
```

```
        MVI   B,0AH
        LXI   H,100H
LOOP:   IN    3
        MOV   M,A
        INX   H
        DCR   B
        JNZ   LOOP
```

```
        ORG   0H
        MVI   B,0AH
        LXI   H,100H
LOOP:   IN    3
        MOV   M,A
        INX   H
        DCR   B
        JNZ   LOOP
        HLT
        END   0H
```

ADRS	HEX	MNEMONIC
00	06	MVI
	0A	
02	21	LXI
	00	
	01	
05	DB	IN
	03	
07	77	MOV
08	23	INX
09	05	DCR
0A	C2	JNZ
	05	
	00	
0D	76	HLT

MEMORY MAP



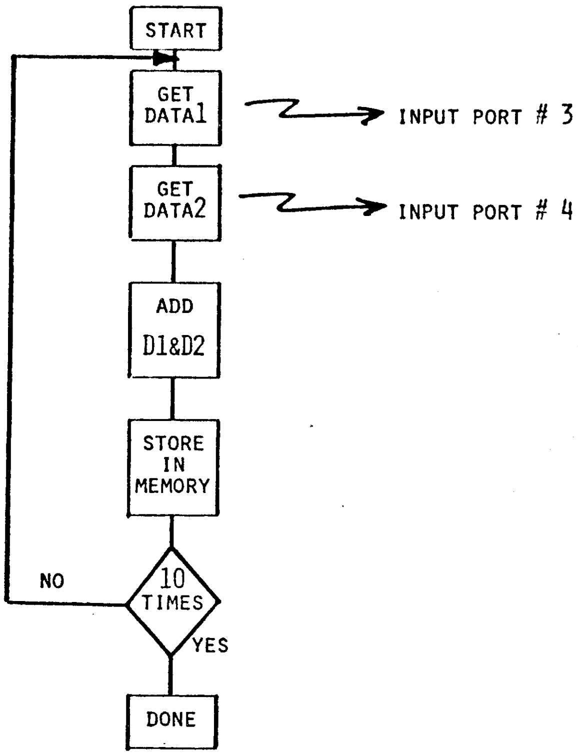
ØØØØH
PC
PROGRAM
ØØØDH
UNUSED PORTION
OF ROM
ROM,
PROM,
EPROM
Ø1ØØH
INPUT DATA AREA
UNUSED PORTION
OF RAM
HL
RAM
Ø1FFH

REVIEW

		8080 ASSEMBLY LANGUAGE PROGRAMMING MANUAL PAGE NUMBER
IN / OUT	PORT #	38
MOV	DESTINATION , SOURCE	16
LXI	DESTINATION , 16 BIT VALUE	26
MVI	DESTINATION , 8 BIT VALUE	26
INR / DCR	8 BIT REGISTER	15
INX / DCX	16 BIT REGISTER	24
JMP	UNCONDITIONAL	32
JNZ / JZ	ZERO	32
JNC / JC	CARRY	32
JPO / JPE	PARITY	33
JP / JM	SIGN	33
HLT	HALT	39
ORG	BEGIN ASSEMBLY	39
END	STOP ASSEMBLY	41

" NOTES "

PROBLEM

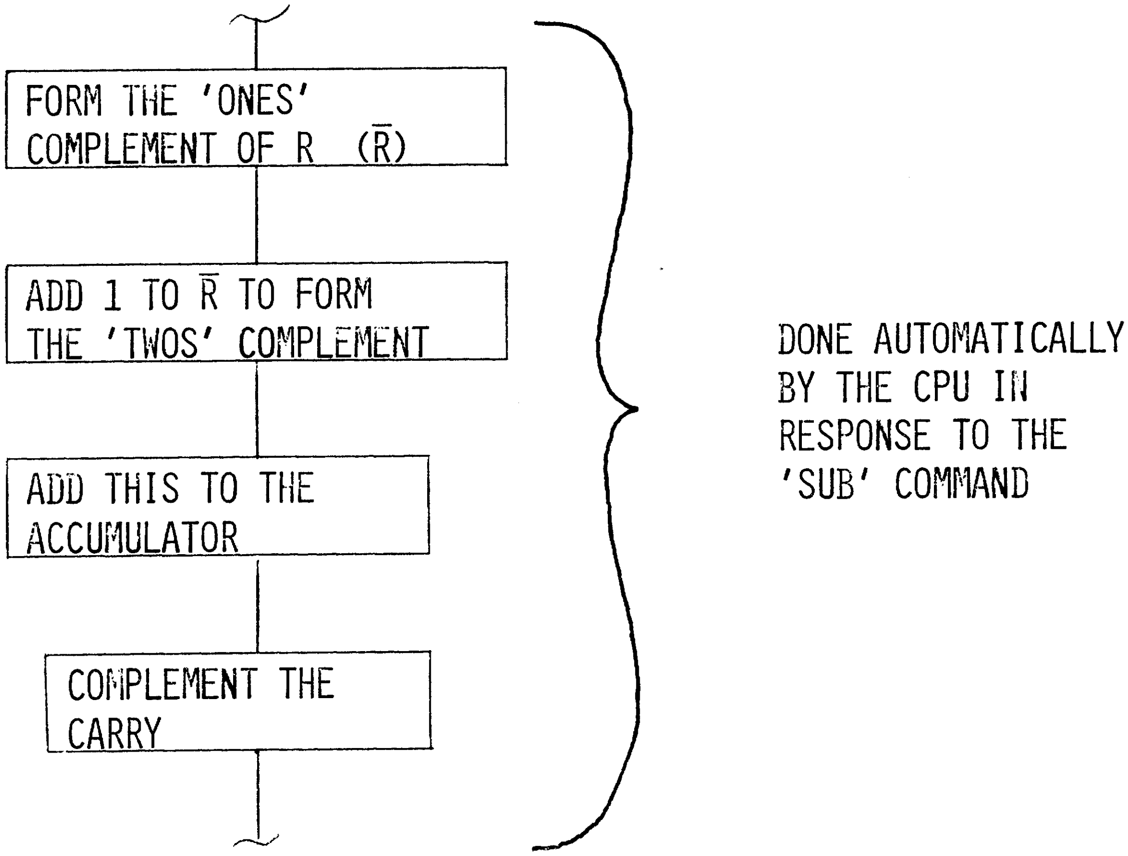



```
        ORG     100H
        MVI     B,0AH
        LXI     H,BUFR
LOOP:   IN      3
        MOV     C,A
        IN      4
        ADD     C
        MOV     M,A
        INX     H
        DCR     B
        JNZ     LOOP
        HLT
BUFR:   DS      10
        END     100H
```

ARITHMETICS

ADD	A <--- A + R
ADC	A <--- A + R + CARRY
SUB	A <--- A + $\overline{R}$ + 1
SBB	A <--- A+($\overline{R+CARRY}$)+1

8080 SUBTRACT OPERATION



EXAMPLES:

A = 97   1001 0111
R = 52   1010 1110 ◁ R = 0101 0010
45    $\boxed{0}$ 0100 0101

$\bar{R}$ = 1010 1101 (ONES COMPLEMENT)
\+ 1
1010 1110 (TWOS COMPLEMENT)

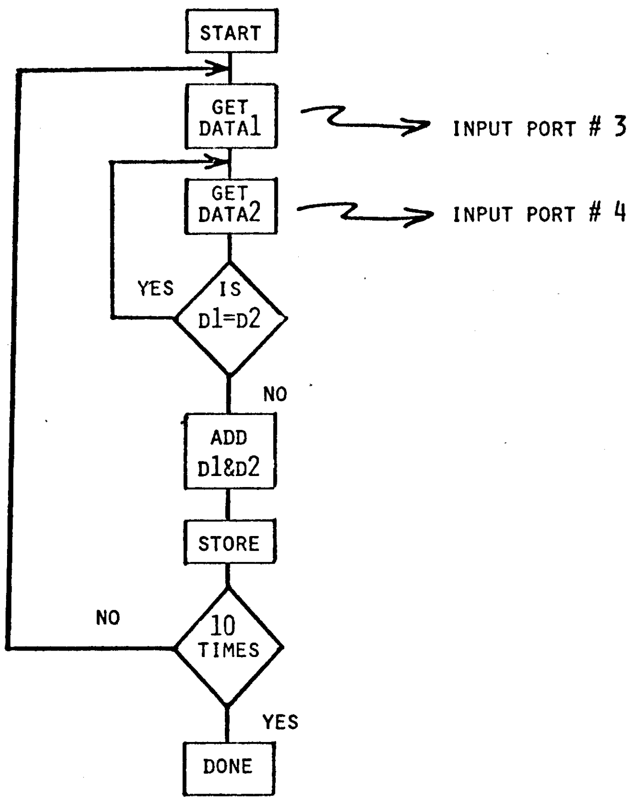
COMPLEMENT

A = 49   0100 1001
R = 76   1000 1010 ◁ R = 0111 0110
$\boxed{1}$ ← D3 $\boxed{1}$ 1101 0011

$\bar{R}$ = 1000 1001 (ONES COMPLEMENT)
\+ 1
1000 1010 (TWOS COMPLEMENT)

COMPLEMENT

P R O B L E M




```
        ORG     100H
        MVI     B,0AH
        LXI     H,BUFR
LOOP:   IN      3
        MOV     C,A
EQUAL:  IN      4
        CMP     C
        JZ      EQUAL
        ADD     C
        MOV     M,A
        INX     H
        DCR     B
        JNZ     LOOP
        HLT
BUFR:   DS      10
        END     100H
```

COMPARE

FLAGS SET BY THE RESULT OF A-R

	CARRY	ZERO
R < A	0	0
R = A	0	1
R > A	1	0

LOGICALS

A <--- A <OPERATOR> R

	ORA	ANA	XRA
ACCUMULATOR	0011	0011	0011
"OTHER ONE"	0101	0101	0101
	0111	0001	0110

RESULT IN ACCUMULATOR

WAIT IF BIT 3 = 0

```
        .
        .
        .
WAIT:   IN    5
        ANI   8
        JZ    WAIT
        .
        .
        .
```

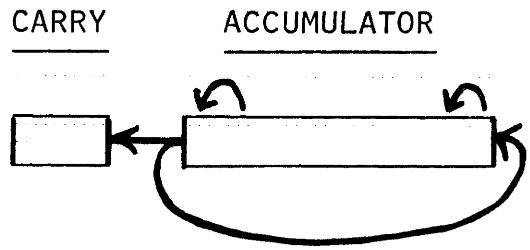
ORA A	CARRY	<---	0
XRA A	CARRY	<---	0
	ZERO	<---	1
	SIGN	<---	0
	PARITY	<---	1
	ACCUMULATOR	<---	0

MISCELLANEOUS

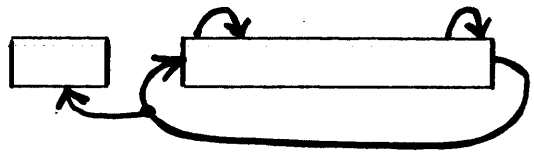
CMA	A	<---	$\overline{A}$
STC	CARRY	<---	1
CMC	CARRY	<---	$\overline{\text{CARRY}}$

ROTATES

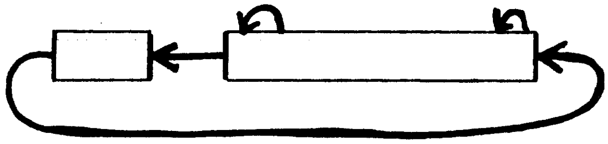
R L C



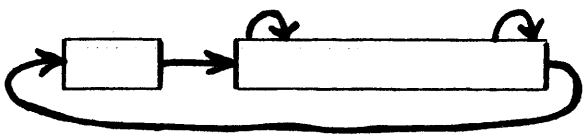
R R C



R A L



R A R



CHECK BIT ZERO

```
        .
        .
WAIT:   IN      5
        RRC
        JNC     WAIT
        .
        .
        .
```

SHIFT LEFT

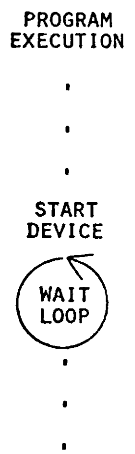
```
        .
        .
        MVI     B,3
SHFT:   ORA     A
        RAL
        DCR     B
        JNZ     SHFT
        .
        .
        .
```

REVIEW

		8080 ASSEMBLY LANGUAGE PROGRAMMING MANUAL PAGE NUMBER
ADD / ADI	ADD	17 / 27
ADC / ACI	ADD WITH CARRY	18 / 27
SUB / SUI	SUBTRACT	18 / 27
SBB / SBI	SUBTRACT WITH BORROW	19 / 28
CMP / CPI	COMPARE	20 / 29
ORA / ORI	LOGICAL OR	20 / 29
ANA / ANI	LOGICAL AND	19 / 28
XRA / XRI	LOGICAL EXCLUSIVE OR	19 / 29
CMA	COMPLEMENT ACCUMULATOR	15
STC	SET CARRY	14
CMC	COMPLEMENT CARRY	14
RLC / RRC	ROTATE ACCUMULATOR	21
RAL / RAR	ROTATE ACCUMULATOR & CARRY	22

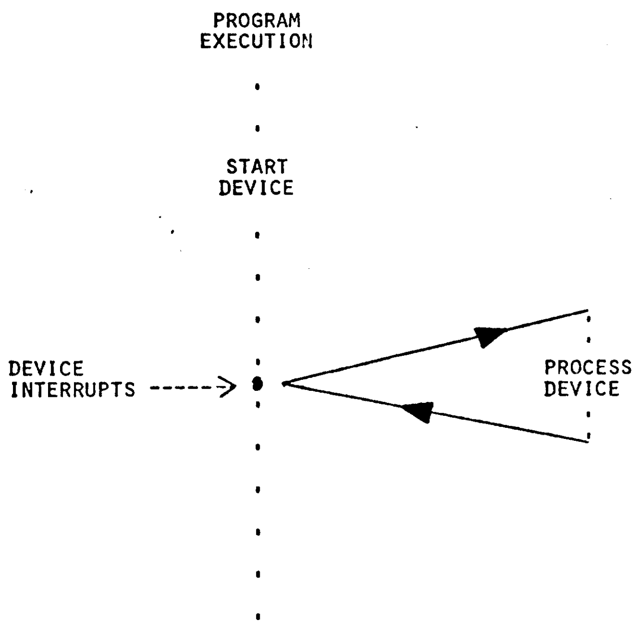
PROGRAMMED INPUT / OUTPUT

START DEVICE AND WAIT FOR COMPLETION



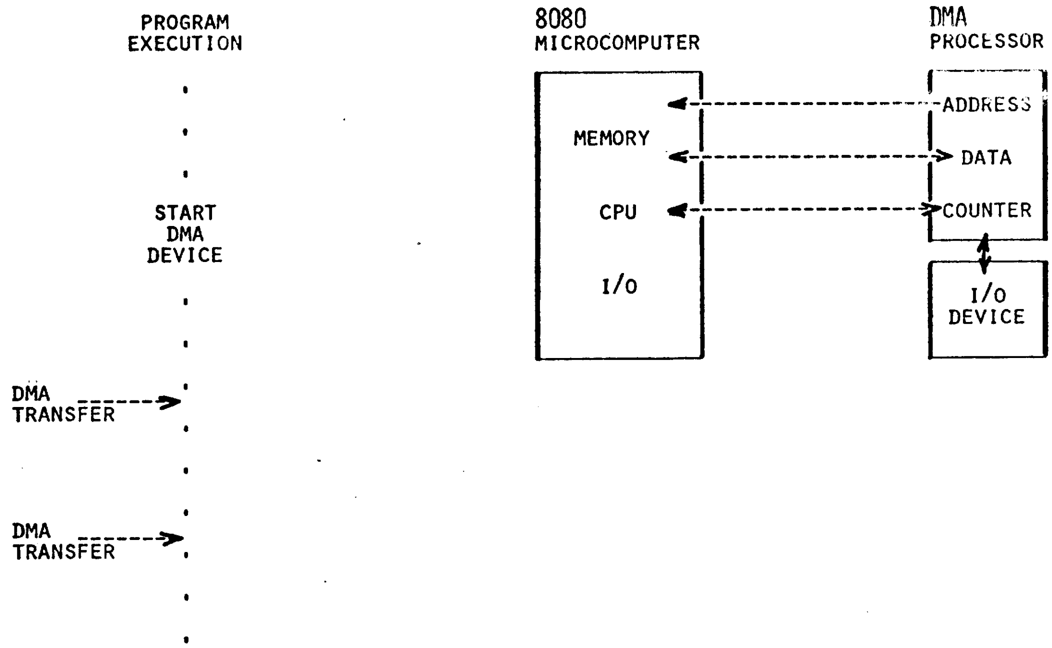
INTERRUPT INPUT / OUTPUT

START DEVICE AND CONTINUE PROGRAM EXECUTION



DIRECT MEMORY ACCESS INPUT / OUTPUT

START DEVICE AND HARDWARE I/O PROCESSOR AND CONTINUE PROGRAM EXECUTION

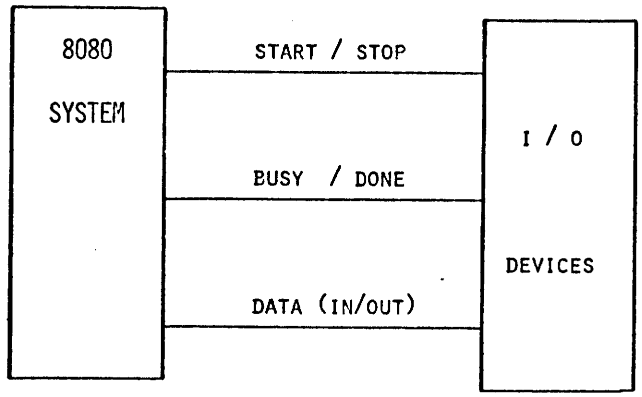


CPU OVERHEAD

(ASSUME 100 TRANSFER PER SECOND DEVICE)

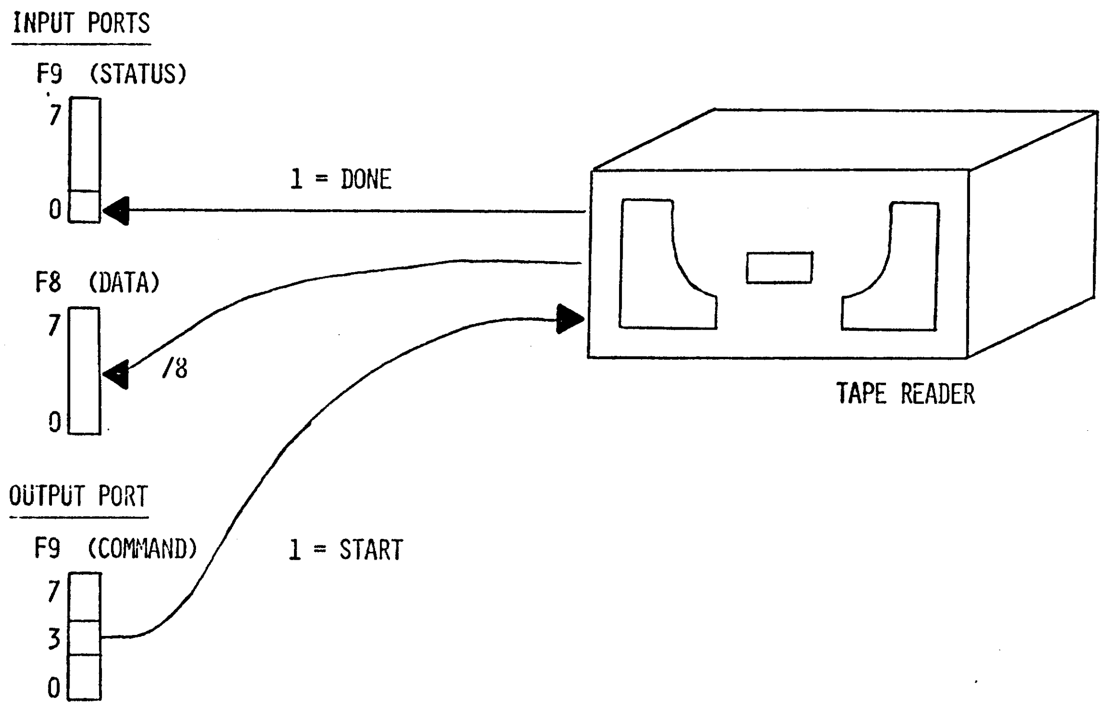
PROGRAMMED	INTERRUPT	DMA
100 %	1 %	0.01 %
WAIT FOR TRANSFERS	100 MICROSECOND SERVICE ROUTINE	1 MEMORY CYCLE PER TRANSFER
1 SECOND FOR 100 CHARACTERS	1/100 SECOND FOR 100 CHARACTERS	NEGLIGIBLE FOR 100 CHARACTERS

INPUT / OUTPUT REQUIREMENTS



- COMMAND
- STATUS
- DATA

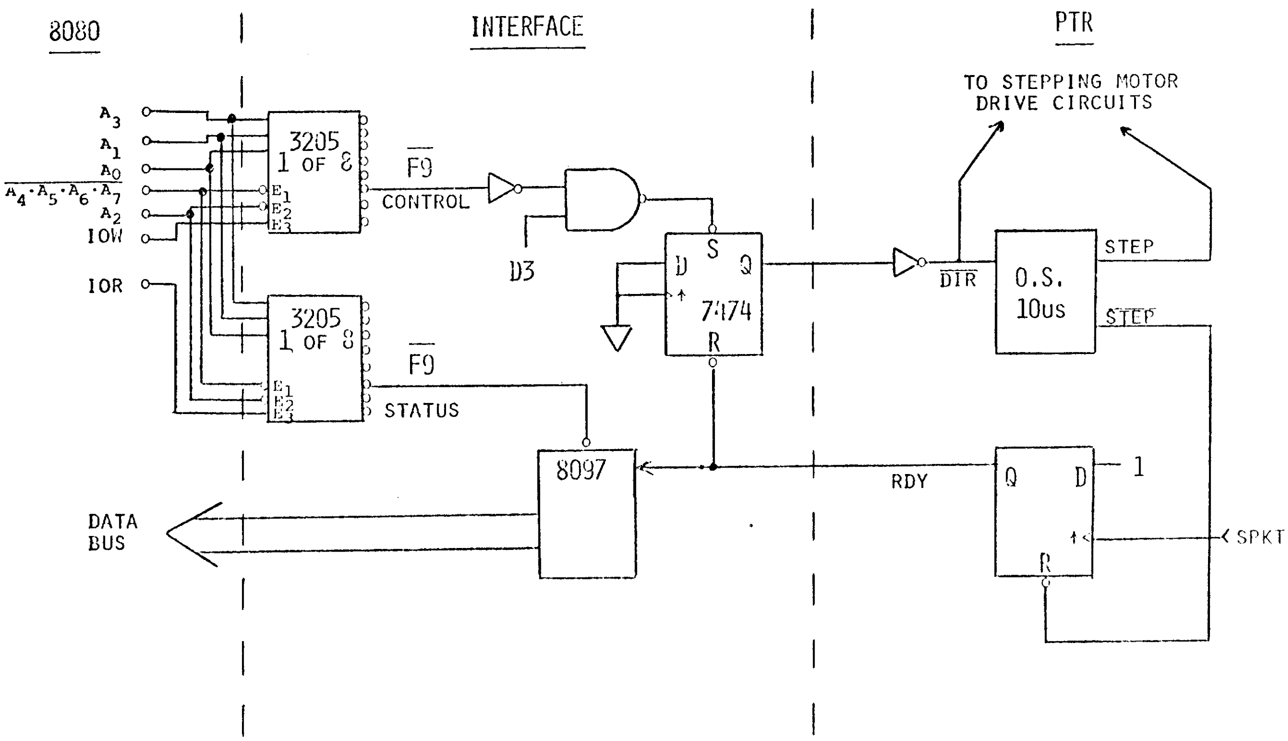
TAPE READER I/O PORT ASSIGNMENT



WAIT - LOOP - I/O

```
TIME IN µSEC          .
                      .
                      .
  3.5     LOOP:   MVI   A,08H    ; DEVICE START
  5               OUT   F9       ; COMMAND
  5       WAIT:   IN    F9       ; GET STATUS
  2               ANI   1H       ; ISOLATE BIT 0
  5               JZ    WAIT     ; IS BIT 0 SET?
  5               IN    F8       ; YES - GET DATA
  3.5             MOV   M,A      ; STORE IN MEMORY
  2.5             INX   H        ; UPDATE ADDRESS
  2.5             DCR   B        ; UPDATE COUNTER
  5               JNZ   LOOP     ; MORE TO GO?
 ----                 .
 39.0                 .
                      .
```

8080 PTR INTERFACE



" NOTES "

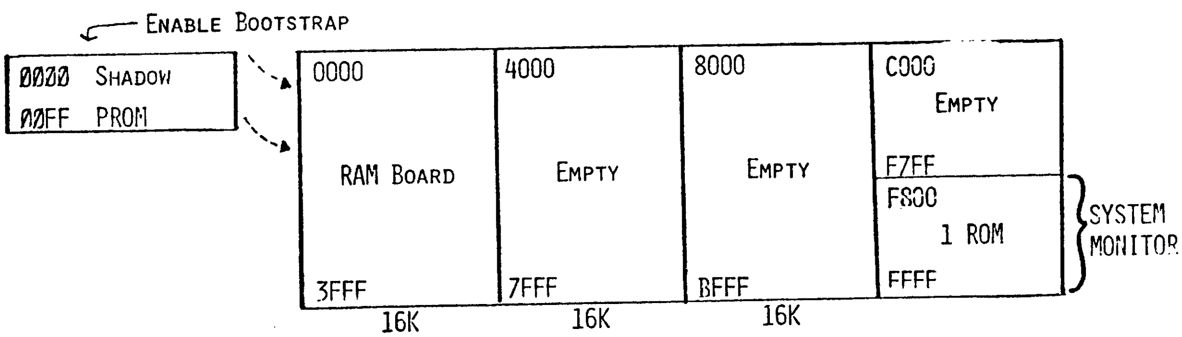
PART III

MICROCOMPUTER DEVELOPMENT SYSTEM

INTELLEC MDS

- SYSTEM DEVELOPMENT (SOFTWARE & HARDWARE)
- DOS OR PTS
- CHIP EMULATION
- EXPANDABLE MEMORY & I/O
- CONTROL PROGRAM (MONITOR)
- RAM RESIDENT MACRO ASSEMBLER
- TEXT EDITOR

INTELLEC MEMORY LAYOUT



- SPACE FOR 64K MEMORY (RAM OR ROM)

SYSTEM MONITOR

- UTILITY / DEBUG PROGRAM
- ROM RESIDENT
- USES RAM LOCATIONS 0 - 6

  318 LOCATIONS AT TOP OF LAST CONTIGUOUS RAM
- COMMUNICATIONS VIA TELETYPE / CRT
- SUPPORTS MANY PERIPHERALS

MONITOR COMMANDS

↲ = CARRIAGE RETURN

- ASSIGN I/O DEVICE

  ALDEV=PDEV — AR=P ↲
- TRANSFER CONTROL

  GADDRESS — G10 ↲
- READ HEX TAPE

  RBIAS ADDRESS — R0 ↲
- DISPLAY AND MODIFY MEMORY

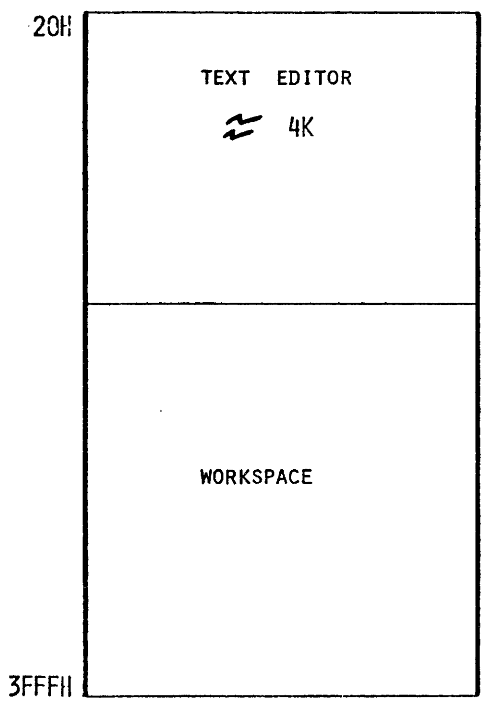
  SADDRESS sp XX- sp XX-ZZ sp XX- — S10 2F- 48-C3 FF-↲

  contents displayed — type in new contents

TEXT EDITOR

- CREATE SOURCE TAPES
- CORRECT SOURCE TAPES
- AUTOMATIC SPACE COMPRESSION

MEMORY MAP



COMMAND FORMAT

- PROMPT SYMBOL IS AN ASTERISK
- TERMINATED BY TWO CONTROL CHARACTERS

  ESC ESC OR ALT MODE ALT MODE
- $$ PRINTED AS ACKNOWLEDGMENT
- CONTROL (C) CANCELS A COMMAND
- RUB OUT OR DEL CANCELS A CHARACTER

TEXT INPUT

- I {TEXT} $$

  AUTOMATIC LINEFEED AFTER CARRIAGE RETURN

  CTRL I TABS TO EVERY EIGHTH COLUMN

  } FROM KEYBOARD

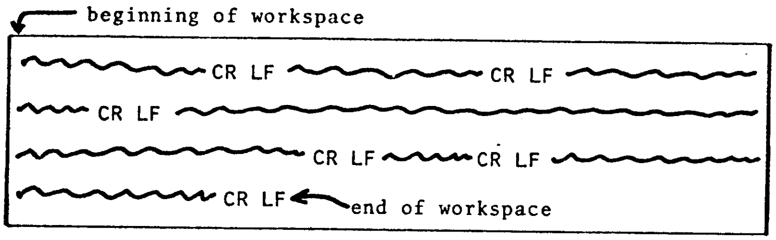
- A$$

  APPEND PAPER TAPE TO WORKSPACE

  } FROM TAPE

WORKSPACE FORMAT

- LINE BY LINE BASIS
- LF DEFINES A LINE



BUFFER (WORKSPACE) POINTER

- POSITION LOCATOR
- POINTS BETWEEN CHARACTERS
- COMMANDS

B - BEGINNING OF WORKSPACE

Z - END OF WORKSPACE

nC - ± N CHARACTERS

nL - ± N LINES (ØL IS BEGINNING OF LINE)

EDIT OPERATION

	ORG	10H	PROGRAM IN WORKSPACE
START:	IN	3	
	OUT	4	
	JMP	START	1 - CHANGE IN 3 TO IN 4
	END	10H	2 - ADD CMA AFTER IN 3

---

1) `B$$` 2) `FINt3$$`	tORGt10H$^{cl}_{rf}$START:tINt3$^{cl}_{rf}$tOUTt4$^{cl}_{rf}$t JMPtSTART$^{cl}_{rf}$tENDt10H$^{cl}_{rf}$
3) `-D$$`	tORGt10H$^{cl}_{rf}$ST<u>A</u>RT:tINt$^{cl}_{rf}$tOUTt4$^{cl}_{rf}$tJ MPtSTART$^{cl}_{rf}$tENDt10H$^{cl}_{rf}$
4) `I4$$` 5) `2C$$`	tORGt10H$^{cl}_{rf}$START:tINt4$^{cl}_{rf}$tOUTt4$^{cl}_{rf}$t JMPtSTART$^{cl}_{rf}$tENDt10H$^{cl}_{rf}$
6) `ItCMA`$^{c}_{r}$ `$$`	tORGt10H$^{cl}_{rf}$START:tINt4$^{cl}_{rf}$tCMA$^{cl}_{rf}$tOU Tt4$^{cl}_{rf}$tJMPtSTART$^{cl}_{rf}$tENDt10H$^{cl}_{rf}$

`B$$`	STEP 1
`S3$4$$`	STEPS 2,3,4
`LItCMA`$^{c}_{r}$`$$`	STEPS 5,6

`BS3$4$LI` `tCMA`$^{c}_{r}$`$$`	STEPS 1,2,3,4,5 & 6

JUST A BIT MORE

- DELETE A LINE

  F {TEXT} $$
  ØL$$
  K$$

- TO CHECK POSITION OF THE BUFFER POINTER

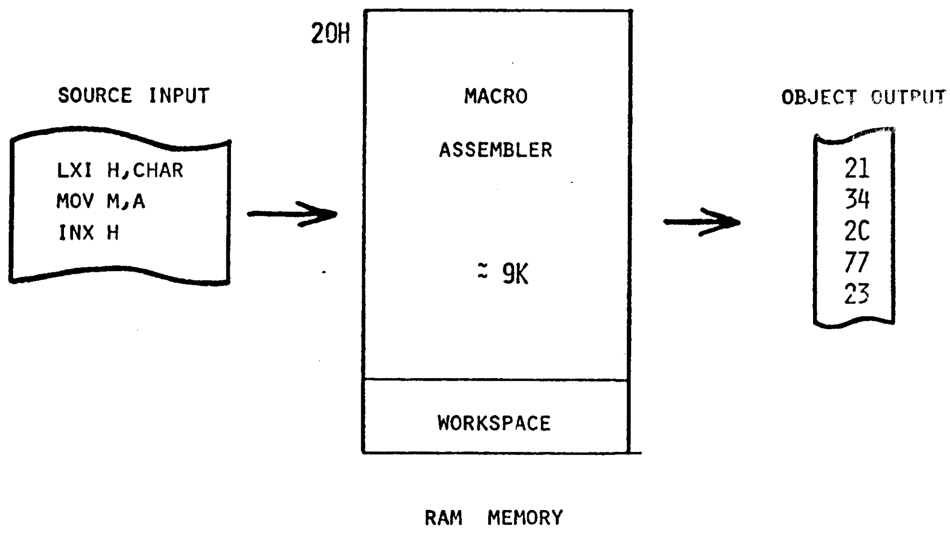
  nT$$ TYPE ± N LINES
  BUFFER POINTER NOT MOVED !

- PUNCH A COPY OF THE WORKSPACE

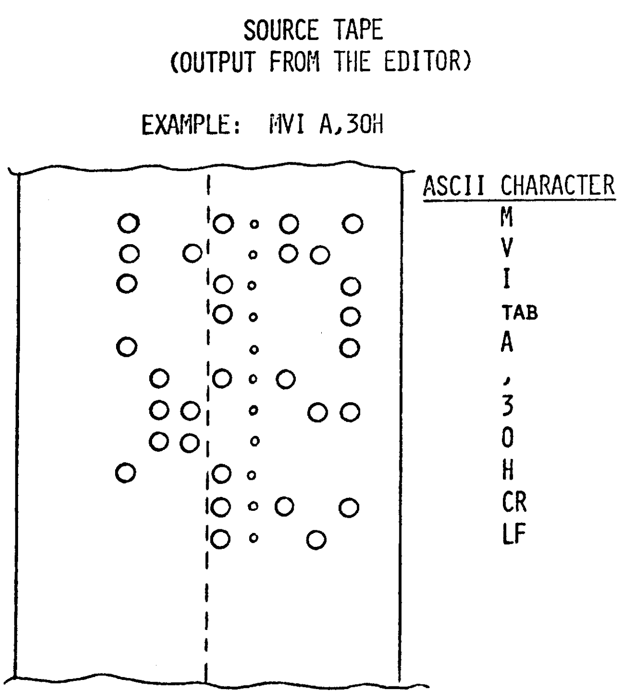
  N$$
  E$$

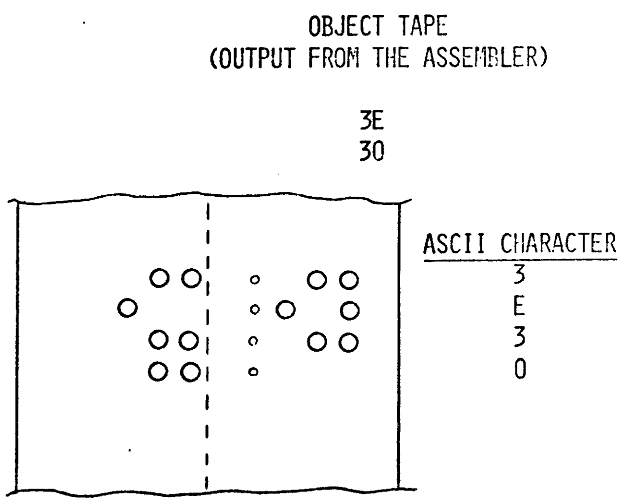
MACRO ASSEMBLER

A PROGRAM WHICH CONVERTS ASSEMBLY LANGUAGE TO BINARY CODE AND CHECKS FOR CERTAIN TYPES OF PROGRAMMER ERRORS.

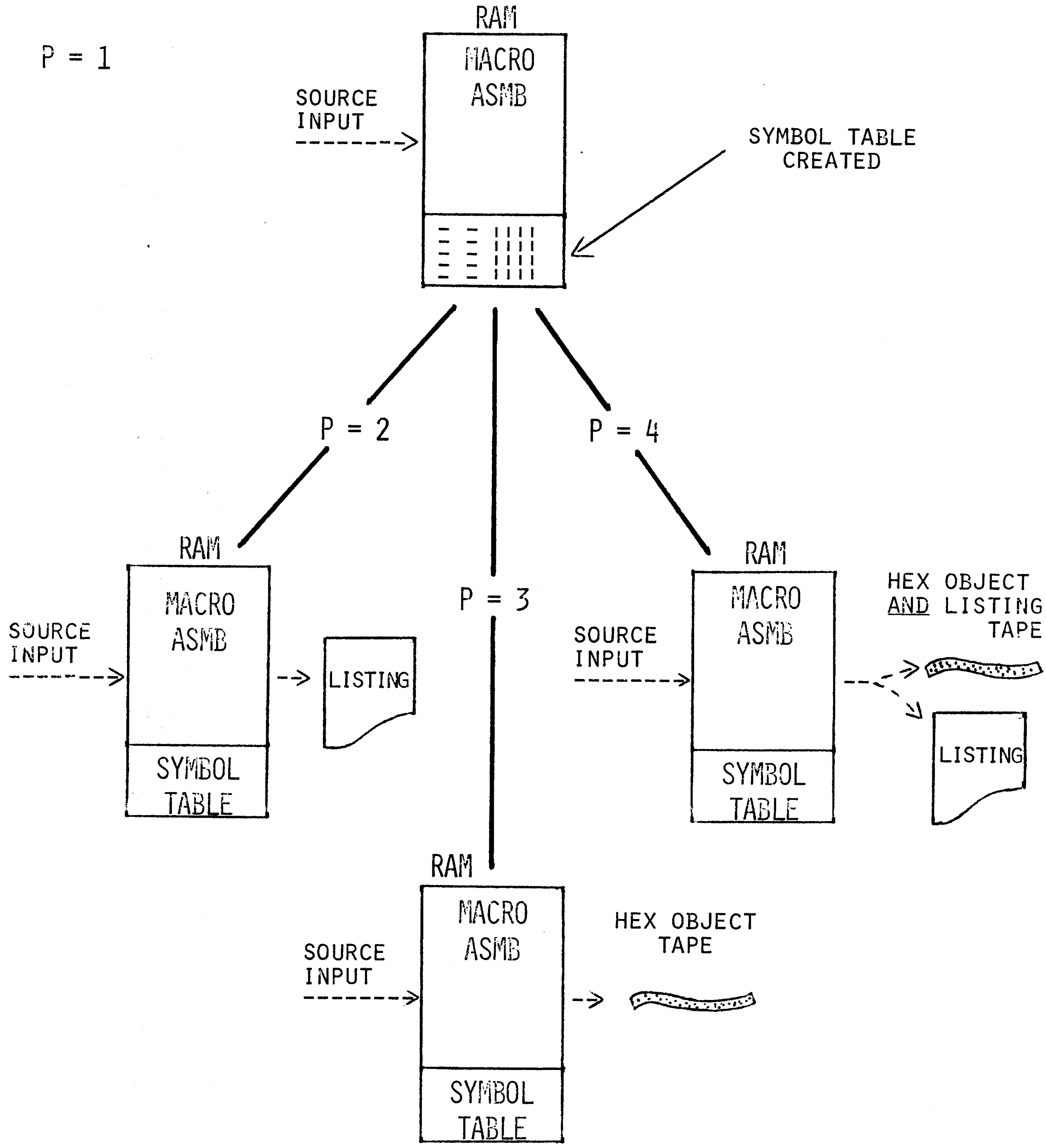


TAPE FORMATS





ASSEMBLER OPERATION



SYMBOL TABLE

ASSEMBLED (CREATED) FROM LABELS AND EQUATE STATEMENTS DURING PASS 1.

TYPICAL LISTING

LOC	CONTENTS	LABEL	CODE	OPERAND
000A		NUM	EQU	0AH
0000			ORG	0
0000	060A		MVI	B,NUM
0002	210001		LXI	H,100H
0005	DB03	LOOP:	IN	3
0007	77		MOV	M,A
0008	23		INX	H
0009	05		DCR	B
000A	C20500		JNZ	LOOP
000D	76		HLT	
			END	0

CORRESPONDING SYMBOL TABLE

NUM	000A
LOOP	0005

FORMAT

FOUR FIELDS 1 OR MORE SPACES IS THE DELIMITER

```
LABEL   CODE   OPERAND  COMMENT
        ORG    10H      ; START ASSEMBLER
BEGIN:  IN     4        ; READ DATA INTO A
        MOV    B,A      ; MOVE TO B
        IN     5        ; READ DATA INTO A
        ADD    B        ; ADD B TO A
        OUT    4        ; WRITE DATA PORT 4
        JMP    BEGIN    ; DO FOREVER
        END    10H      ; STOP ASSEMBLER
```

LABEL - 1 TO 5 CHARACTERS; FIRST CHARACTER IS ALPHABETIC OR @ OR ? TERMINATED BY COLON. CANNOT BE AN INSTRUCTION MNEMONIC OR REGISTER NAME.

CODE - INSTRUCTION OR PSEUDO-INSTRUCTION MNEMONIC

OPERAND - NONE, 1 OR 2 ITEMS (TWO ITEMS SEPERATED BY A COMMA) REGISTERS , IMMEDIATE DATA OR ADDRESSES

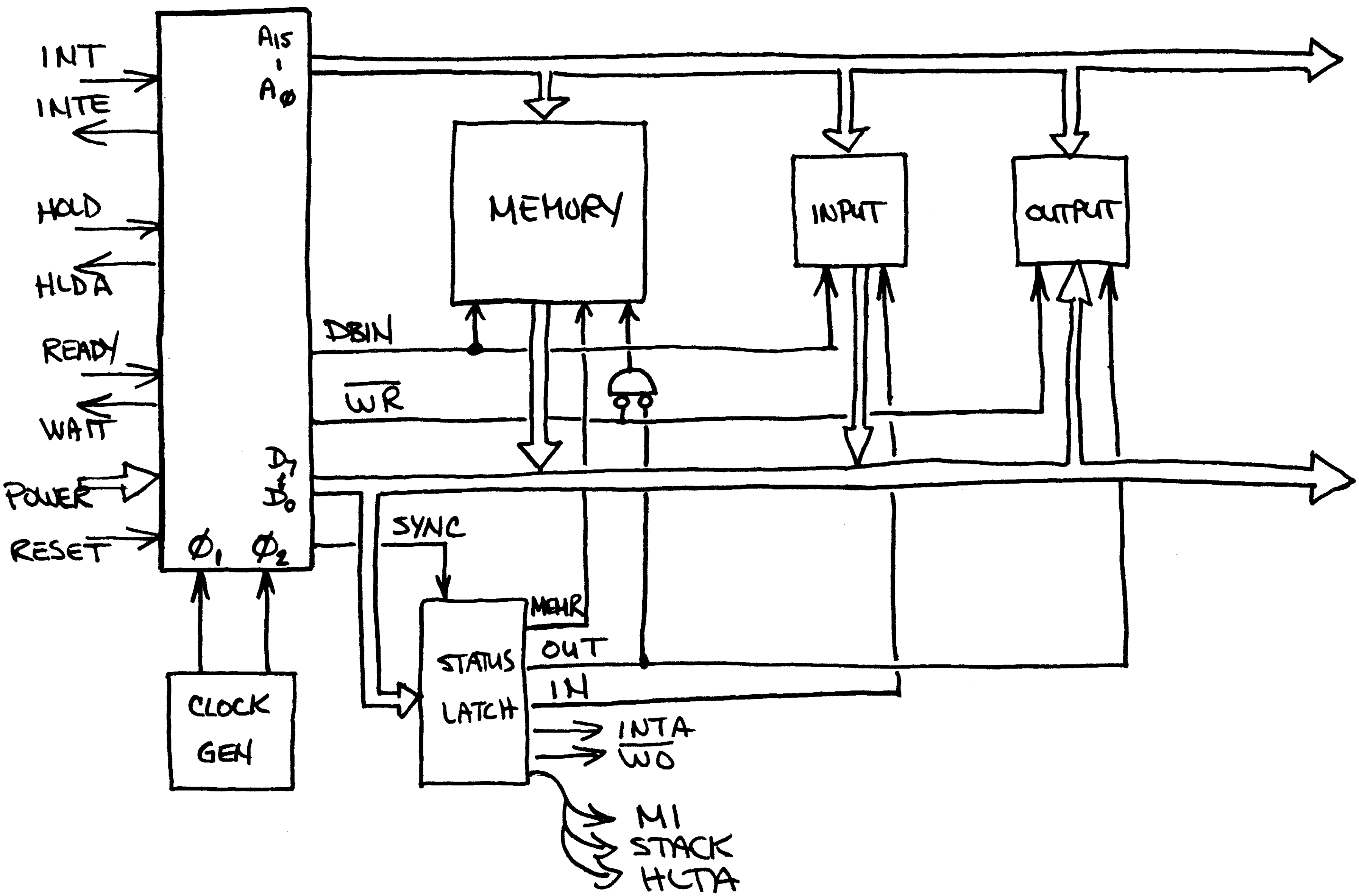
COMMENT - BEGINS WITH SEMICOLON

PART IV

BASIC HARDWARE

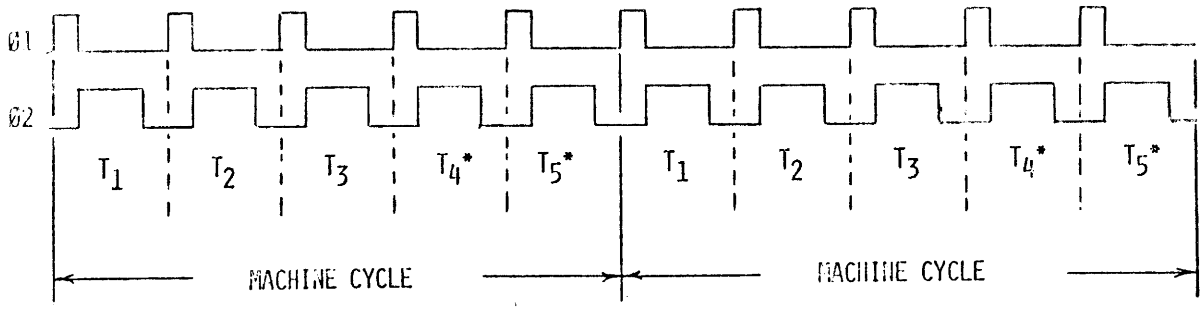
" NOTES "

8080 SYSTEM



INT
INTE
HOLD
HLDA
READY
WAIT
POWER
RESET
A15
A0
D7
D0
ϕ_1
ϕ_2
CLOCK GEN
MEMORY
INPUT
OUTPUT
DBIN
$\overline{WR}$
SYNC
STATUS LATCH
MEMR
OUT
IN
INTA
$\overline{WO}$
MI
STACK
HLTA

BASIC 8080 TIMING

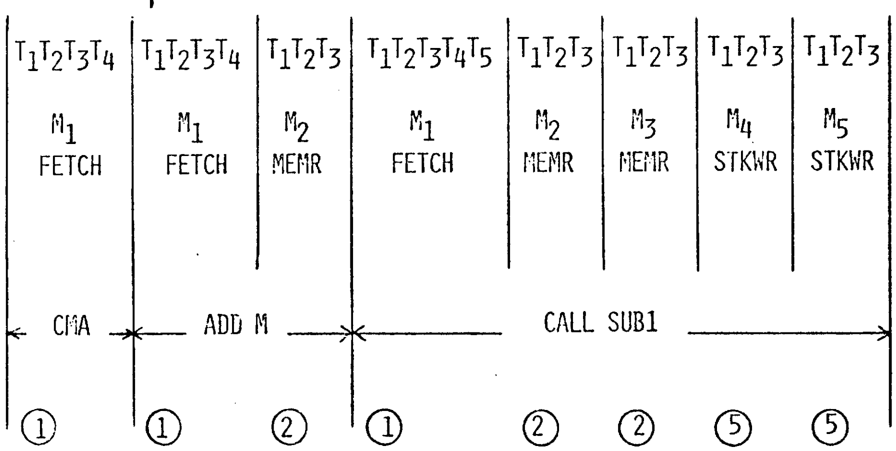


EACH MACHINE CYCLE PERFORMS 1 OR 10 DIFFERENT FUNCTIONS:

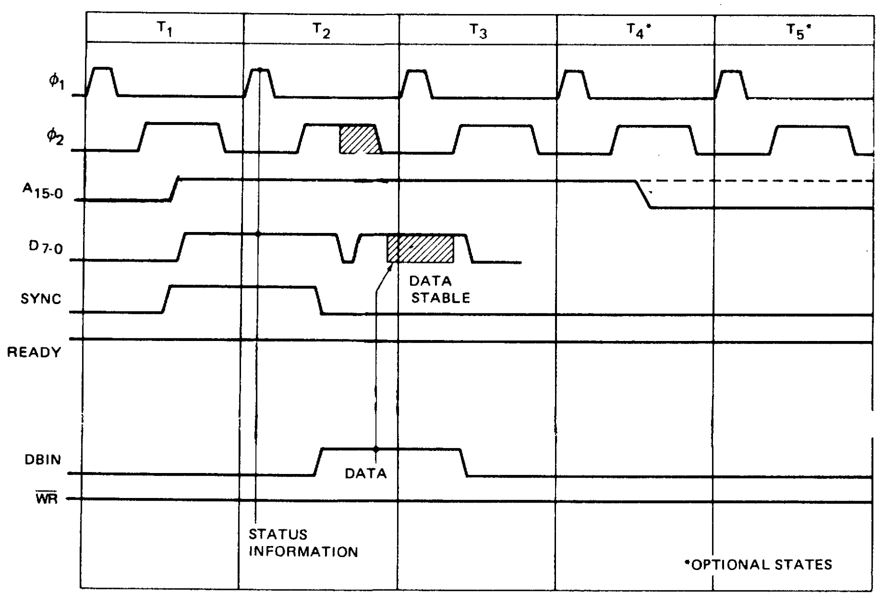
1. FETCH (M1)
2. MEMORY READ
3. MEMORY WRITE
4. STACK READ
5. STACK WRITE
6. INPUT
7. OUTPUT
8. INTERRUPT
9. HALT
10. HALT-INTERRUPT

INSTRUCTION TIMING

```
EXAMPLE:  CMA
          ADD  M
          CALL SUB1
```



STATE (CLOCK PERIOD) DESCRIPTION



T1 - ADDRESS BUS <--- MEMORY ADDRESS OR I/O PORT #
DATA BUS <--- STATUS INFORMATION

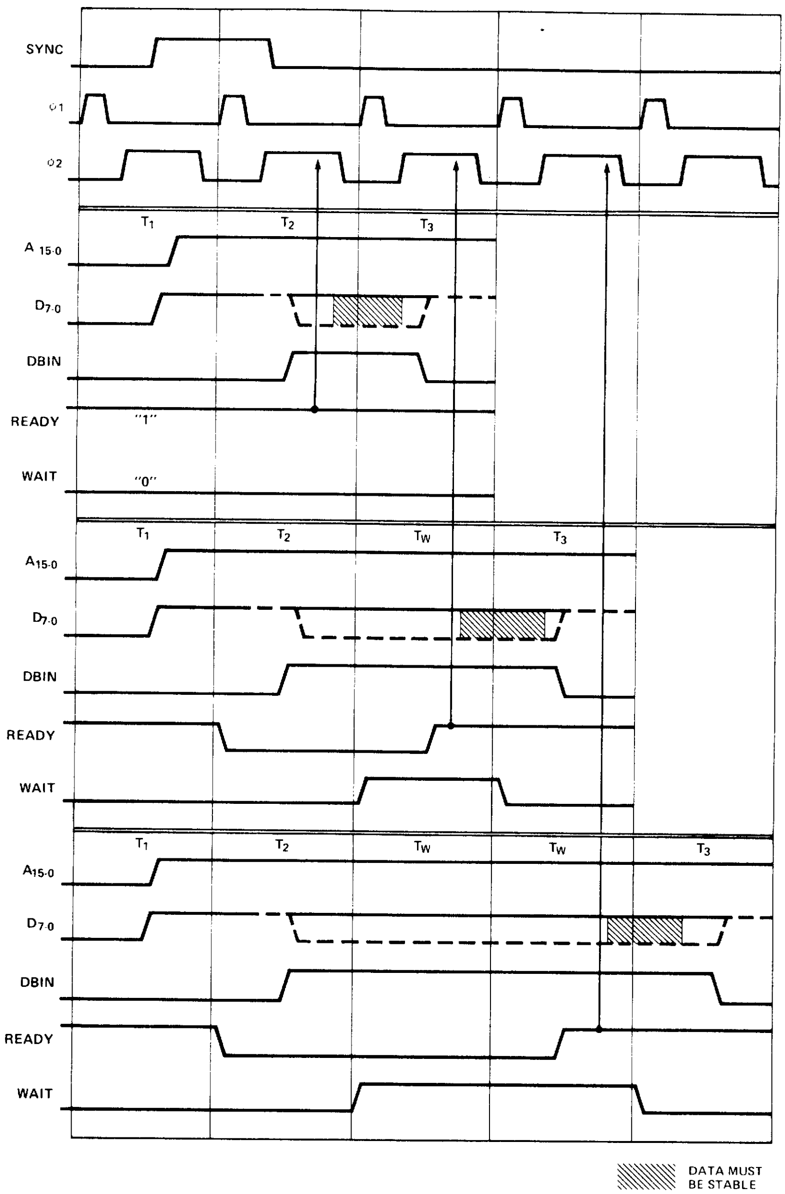
T2 - READY AND HOLD INPUTS SAMPLED
CHECK FOR HALT INSTRUCTION

T3 - DATA BUS <-- DATA FROM MEMORY OR INPUT PORT
DATA BUS <-- DATA FROM CPU FOR MEMORY OF OUPUT PORT

T4 -
T5 - } USED FOR INTERNAL PROCESSOR OPERATIONS IF NEEDED

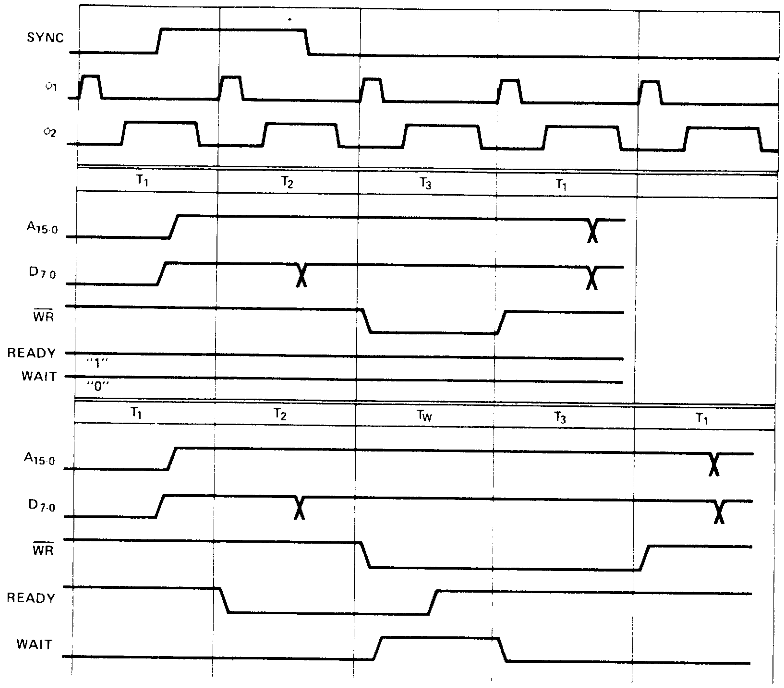
TIMING

Relation between READY and DBIN

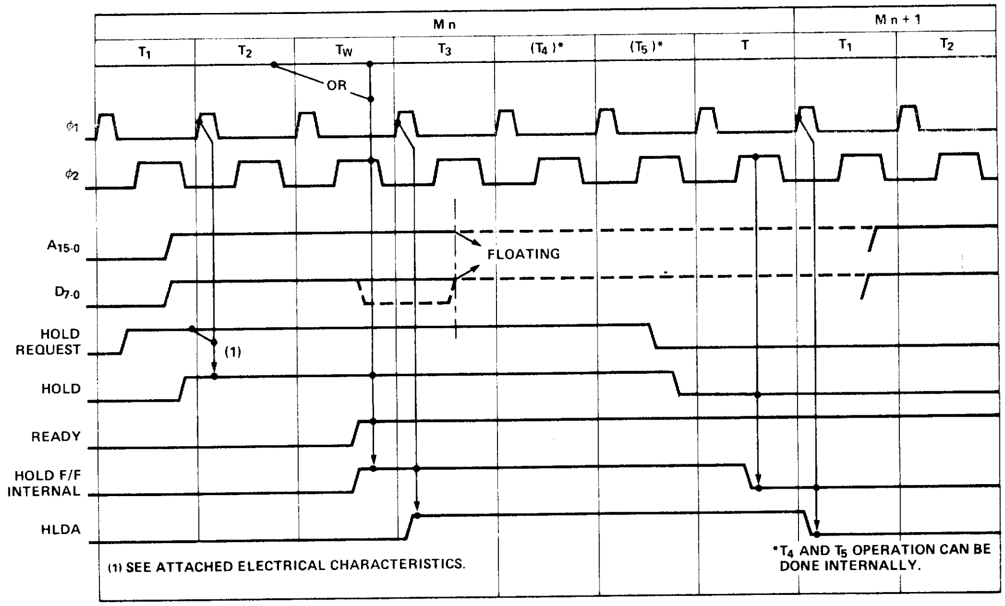


TIMING

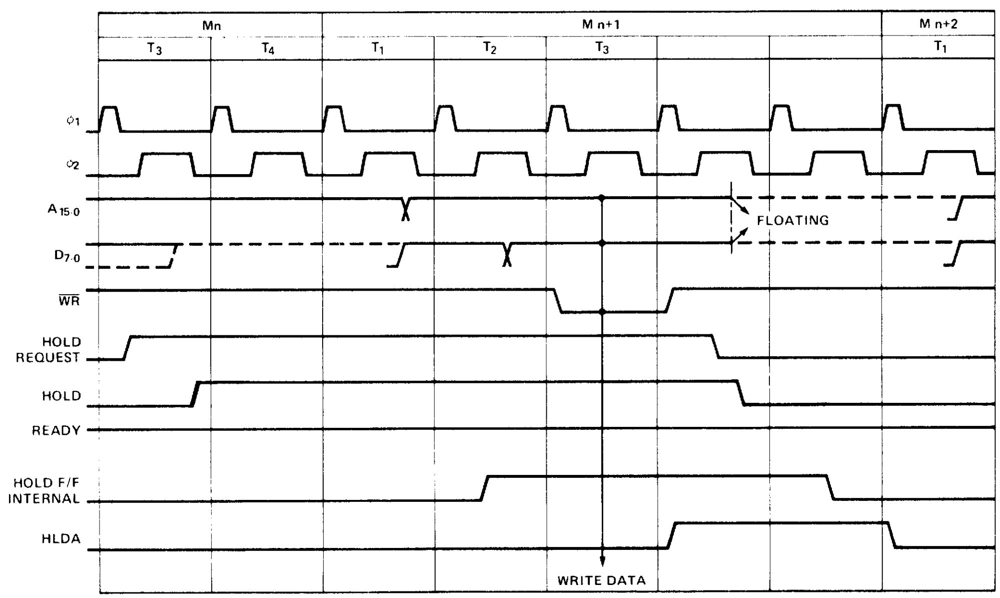
Relation between READY, WAIT and $\overline{WR}$



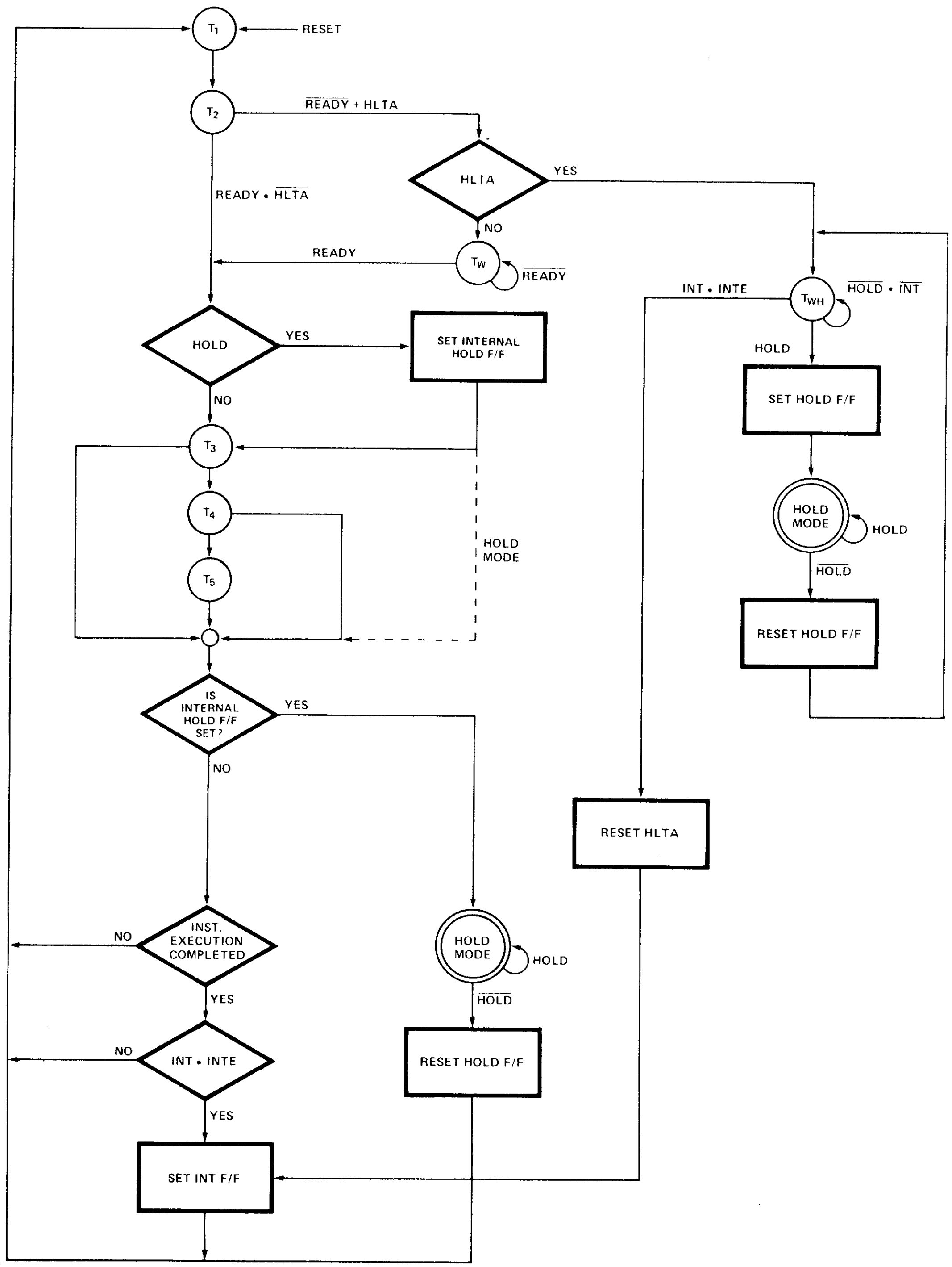
HOLD OPERATION (READ MODE)



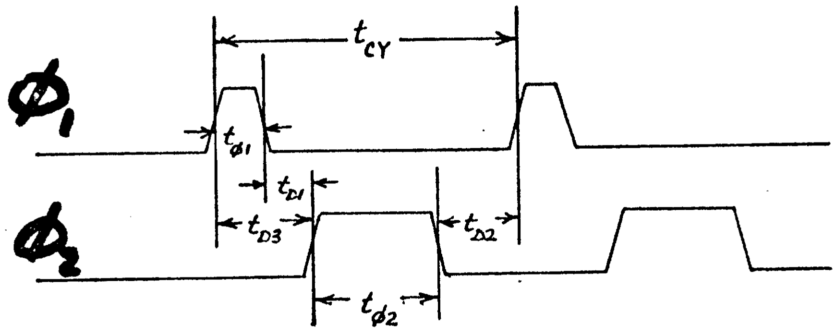
HOLD Operation (Write mode)



CPU STATE TRANSITION DIAGRAM

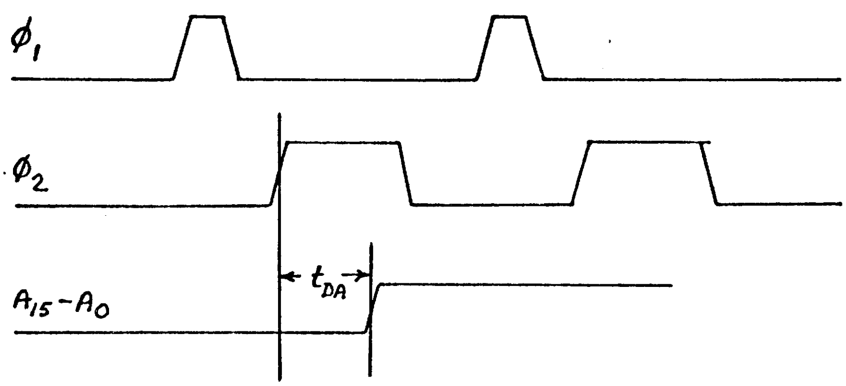


CLOCKS



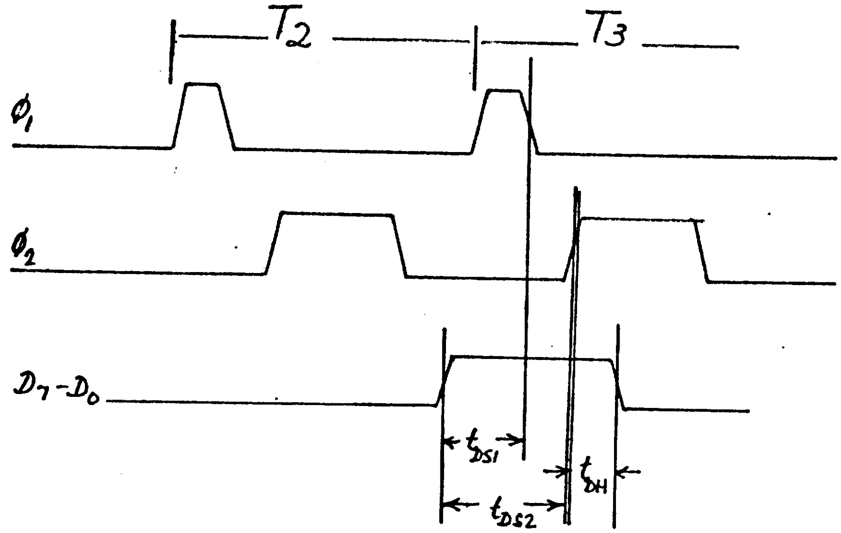
CLOCK PERIOD	480 NSEC $< T_{CY} <$ 2 USEC			
CLOCK RISE & FALL TIME	5 NSEC $< T_R , T_F <$ 50 NSEC	t_{D1}	MIN	0 NSEC
$t_{\phi 1}$ 60 NSEC MINIMUM		t_{D2}	MIN	70 NSEC
$t_{\phi 2}$ 220 NSEC MINIMUM		t_{D3}	MIN	80 NSEC

ADDRESS



T_{DA} = ADDRESS OUTPUT DELAY FROM ϕ_2, MAXIMUM OF 200 NSEC

DATA

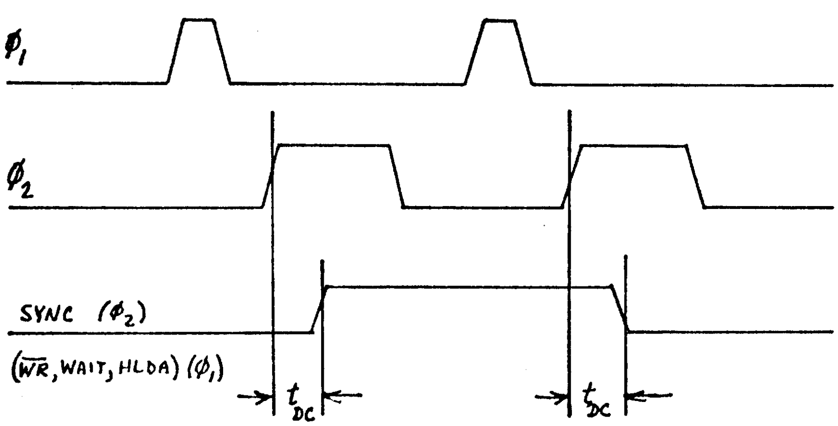


T_{DH} = DATA HOLD TIME FROM $Ø_2$ DURING DBIN, MINIMUM IS T_{DF} (25 - 140 NSEC)

T_{DS1}= DATA SETUP TIME DURING $Ø_1$ AND DBIN, MINIMUM IS 30 NSEC

T_{DS2}= DATA SETUP TIME DURING $Ø_2$ AND DBIN, MINIMUM IS 150 NSEC

CONTROL



T_{DC} = CONTROL SIGNAL OUTPUT DELAY FROM $Ø_1$ OR $Ø_2$, 120 NSEC MAXIMUM

" NOTES "

PART V

ASSEMBLY LANGUAGE INSTRUCTIONS

SUBROUTINES

- SECTIONS OF A PROGRAM THAT ARE CALLed AND RETurned FROM

```
             ORG
              .
              .
              .
             CALL SUBX
              .
              .
              .
             CALL SUBX
              .
              .
              .
     SUBX:   ~
             RET
              .
              .
              .
             END
```

- PC + 3 SAVED AS THE RETURN ADDRESS

THE 8080 STACK

- IMPLEMENTED FOR SUBROUTINE RETURN ADDRESSES

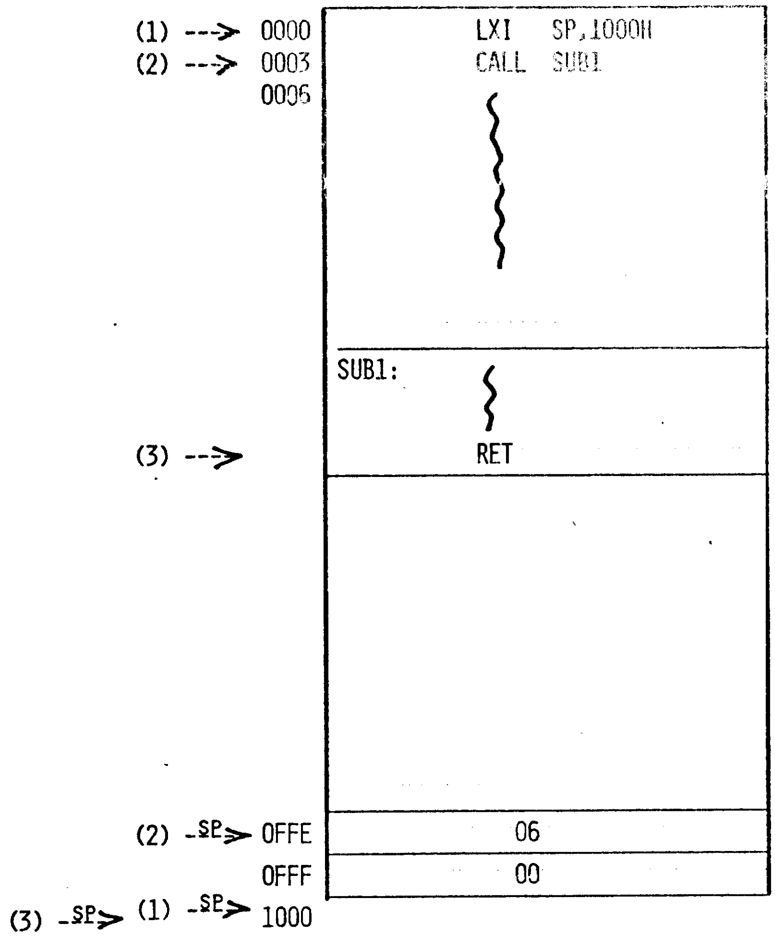
- ADDRESSED BY THE SP REGISTER WITH INITIAL LOCATION SET BY THE USER

  LXI SP,1FC0H (SPHL)

- CALL WRITES THE RETURN ADDRESS INTO THE STACK

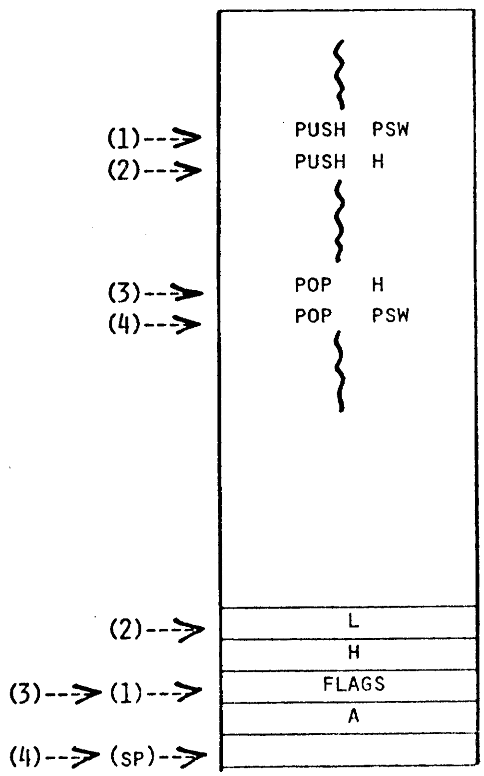
- RET READS THE RETURN ADDRESS FROM THE STACK

STACK OPERATION



PUSH / POP INSTRUCTIONS

- PUSH WRITES
- POP READS
- BITS (15 THRU 8) AT (SP) - 1
- BITS (7 THRU 0) AT (SP) - 2
- SP AUTOMATICALLY ADJUSTED



TEMPORARY REGISTER SAVING

```
; SUBROUTINE UTILIZES H, L, AND A.

SUBX:   PUSH H          ; SAVE HL
        PUSH PSW        ; SAVE A,FLAGS
        LXI  H,BUF1
        IN   1
        MOV  M,A
        POP  PSW        ; RESTORE A,FLAGS
        POP  H          ; RESTORE HL
        RET
```

PARAMETER PASSING

- REGISTERS

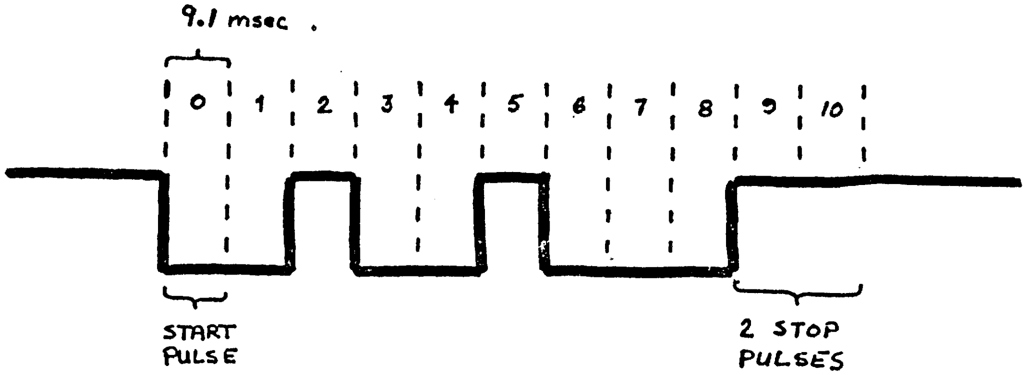
BYTE	ADDRESS
A	HL
B	DE
C	BC
D	
E	
H	
L	

- STACK

- VARIABLE MEMORY

" NOTES "

TTY SERIAL BIT STREAM

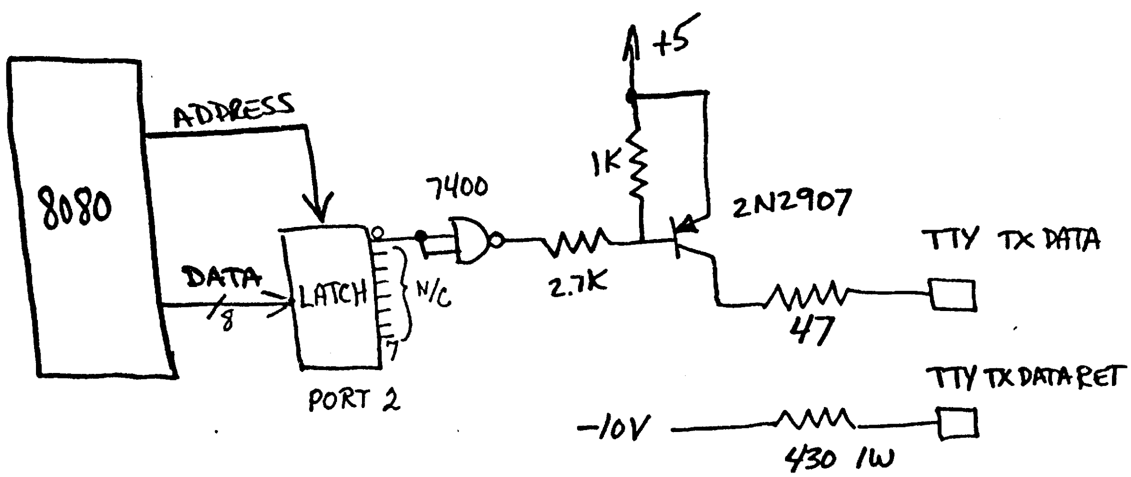


10 CHARACTERS / SECOND

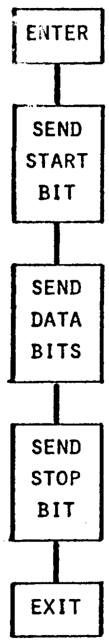
11 BITS/CHARACTER

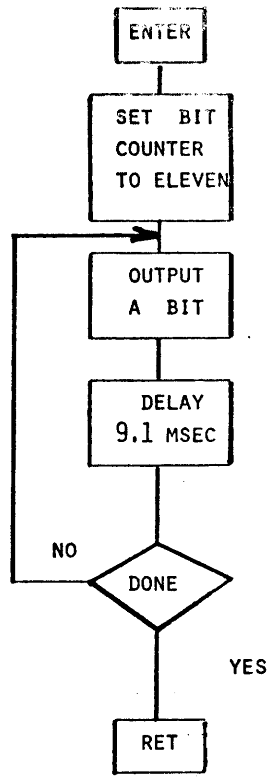
TTY INTERFACE

<TRANSMIT>



PROBLEM (TELETYPE OUTPUT SUBROUTINE)





TELETYPE OUTPUT SUBROUTINE (ASSUME TTY CONNECTED TO PORT 2 BIT 0)

```
;
; THIS SUBROUTINE ENTERED WITH CHARACTER TO BE OUTPUT IN THE C REGISTER
;
TYOUT:  MVI     B,11    ; SET COUNTER FOR 11 BITS
        MOV     A,C     ; CHARACTER TO ACCUMULATOR
        ORA     A       ; CLEAR CARRY - FOR START BIT
        RAL             ; MOVE CARRY TO A(0)
MORE:   OUT     2       ; SEND TO TTY
        CALL    DELAY   ; KILL TIME
        RAR             ; POSITION NEXT BIT
        STC             ; SET CARRY - FOR STOP BITS
        DCR     B       ; DECREMENT BIT COUNTER
        JNZ     MORE    ; DONE ?
        RET             ; YES
;
; 9 MSEC DELAY  ( ASSUME NO WAIT STATES )
;
DELAY:  MVI     D,6
DL0:    MVI     E,200
DL1:    DCR     E       ; 1.5 MSEC
        JNZ     DL1     ; INNER LOOP
        DCR     D
        JNZ     DL0
        RET
```

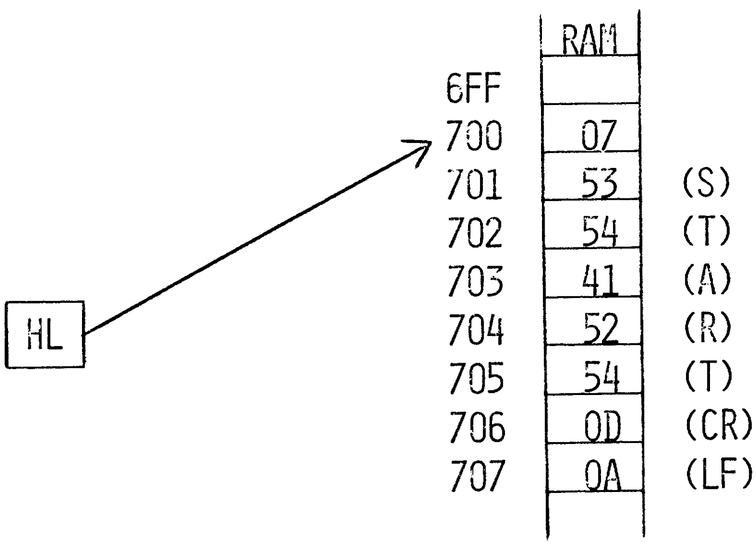
MESSAGE OUTPUT ROUTINE

```
          LXI     H,MSG1
          CALL    PRINT

PRINT:    MOV     D,M
LOOP:     INX     H
          MOV     C,M
          CALL    TTY OUT
          DCR     D
          JNZ     LOOP
          RET

MSG1:     DB      7,'START',ODH,OAH
```

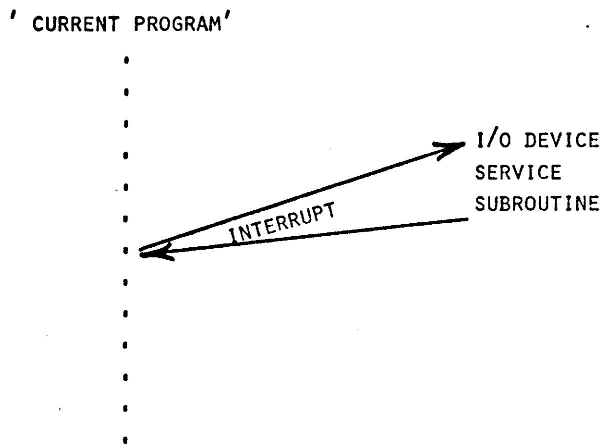
MEMORY CONTENTS AFTER ASSEMBLING ABOVE



REVIEW		8080 ASSEMBLY LANGUAGE PROGRAMMING MANUAL PAGE NUMBER
CALL	UNCONDITIONAL	34
CNZ, CNC, CPO, CP CZ , CC , CPE, CM	ZERO,CARRY,PARITY,SIGN	34 / 35
RET	UNCONDITIONAL	36
RNZ, RNC, RPO, RP RZ , RC , RPE, RM	ZERO,CARRY,PARITY,SIGN	36 / 37
SPHL	SP ←---HL	25
PUSH PSW, B, D, H	STACK WRITE	22
POP PSW, B, D, H	STACK READ	23
DB	DEFINE BYTE (s)	13

INTERRUPT PROCESS

- STOPS CURRENT PROGRAM EXECUTION
- A "SPECIAL" SUBROUTINE CALL INSTRUCTION IS EXECUTED



- PROGRAM COUNTER NOT INCREMENTED DURING INTERRUPT INSTRUCTION

INTERRUPT SPECTRUM

CONVENIENCE <- -> NECESSITY

DEVICES THAT:	DEVICES THAT:
• RARILY REQUIRE SERVICE	• FREQUENTLY REQUIRE SERVICE
• CAN WAIT FOR SERVICE	• MUST BE SERVICED NOW !

BENEFIT

INCREASED PROCESSOR UTILIZATION

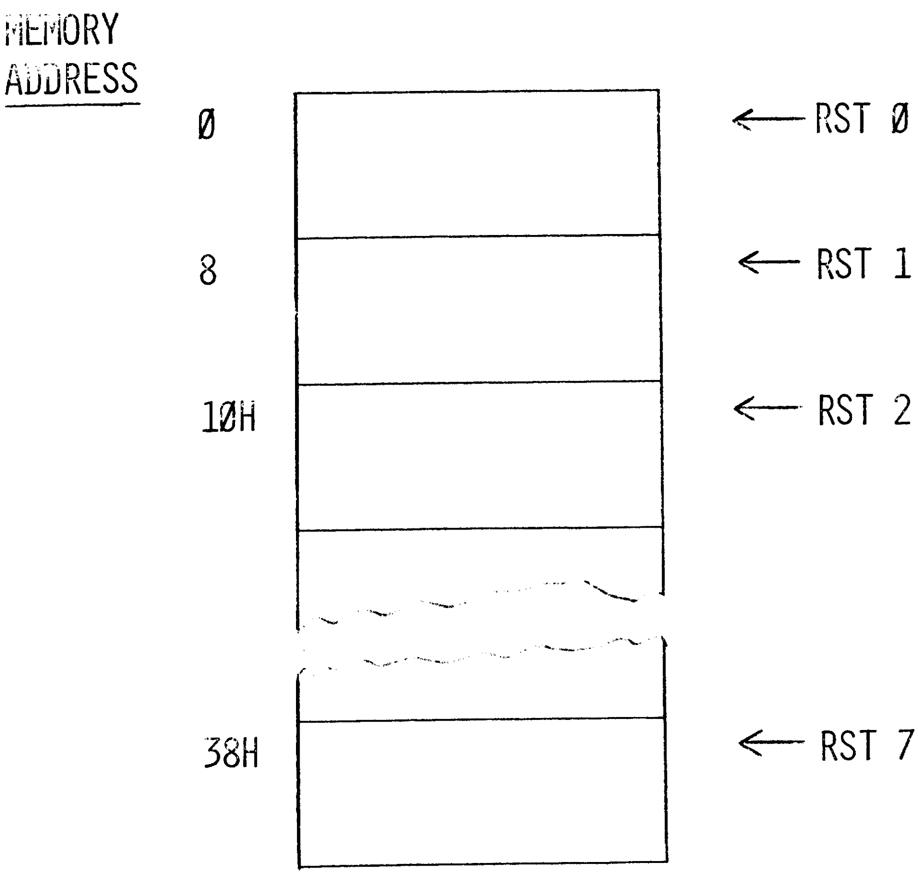
THE SPECIAL INTERRUPT INSTRUCTIONS

- EIGHT 1 BYTE SUBROUTINE CALL'S

  RST Ø THRU RST 7

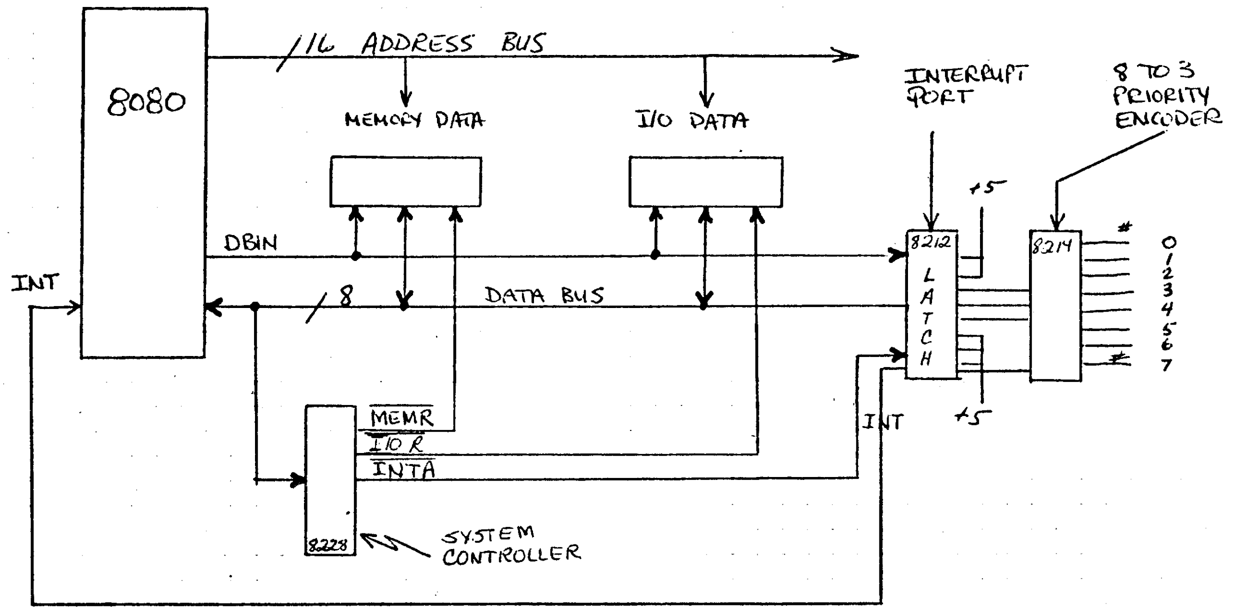
- EXTERNAL HARDWARE MUST PUT THE RST BIT PATTERN ONTO THE DATA BUS WHEN THE INTA STATUS SIGNAL IS PRODUCED (T2)

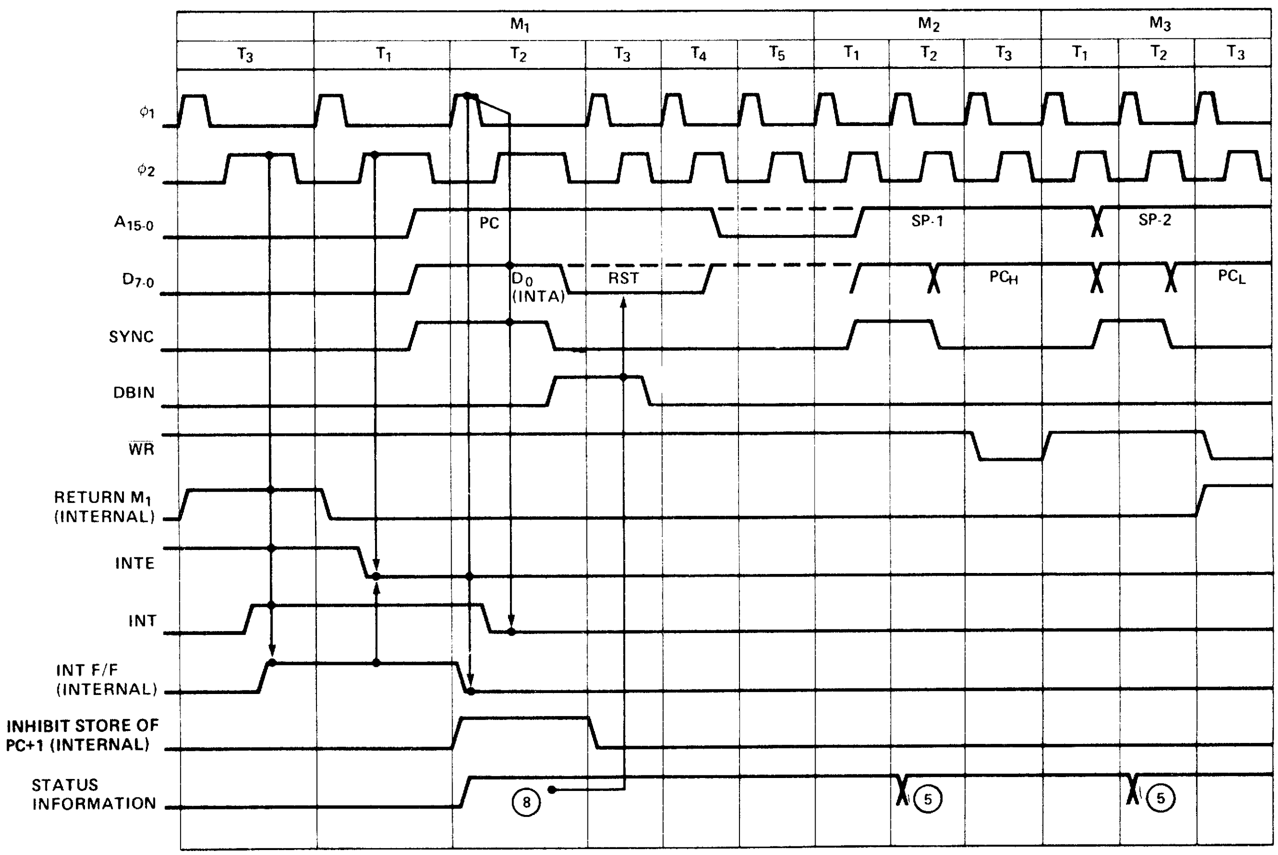
- PROGRAM COUNTER SET TO:
  Ø, 8, 10, 18, 20, 28, 30, 38 (HEX)



- RST'S ALSO USEFUL FOR OFTEN CALLED SUBROUTINES

8080 INTERRUPTS





INTERRUPT PROCESSING

```
SVC:    PUSH    PSW     ; 11 STATES   5.5 MICRO-SEC
        PUSH    B       ;     "             "
        PUSH    D       ;     "             "
        PUSH    H       ;     "             "
         .
         .
         .
         .
         .
         .
        POP     H       ; 10 STATES   5.0 MICRO-SEC
        POP     D       ;     "             "
        POP     B       ;     "             "
        POP     PSW     ;     "             "
        EI              ;  4 STATES   2.0 MICRO-SEC
        RET             ; 10 STATES   5.0 MICRO-SEC
```

WORST CASE TIMING

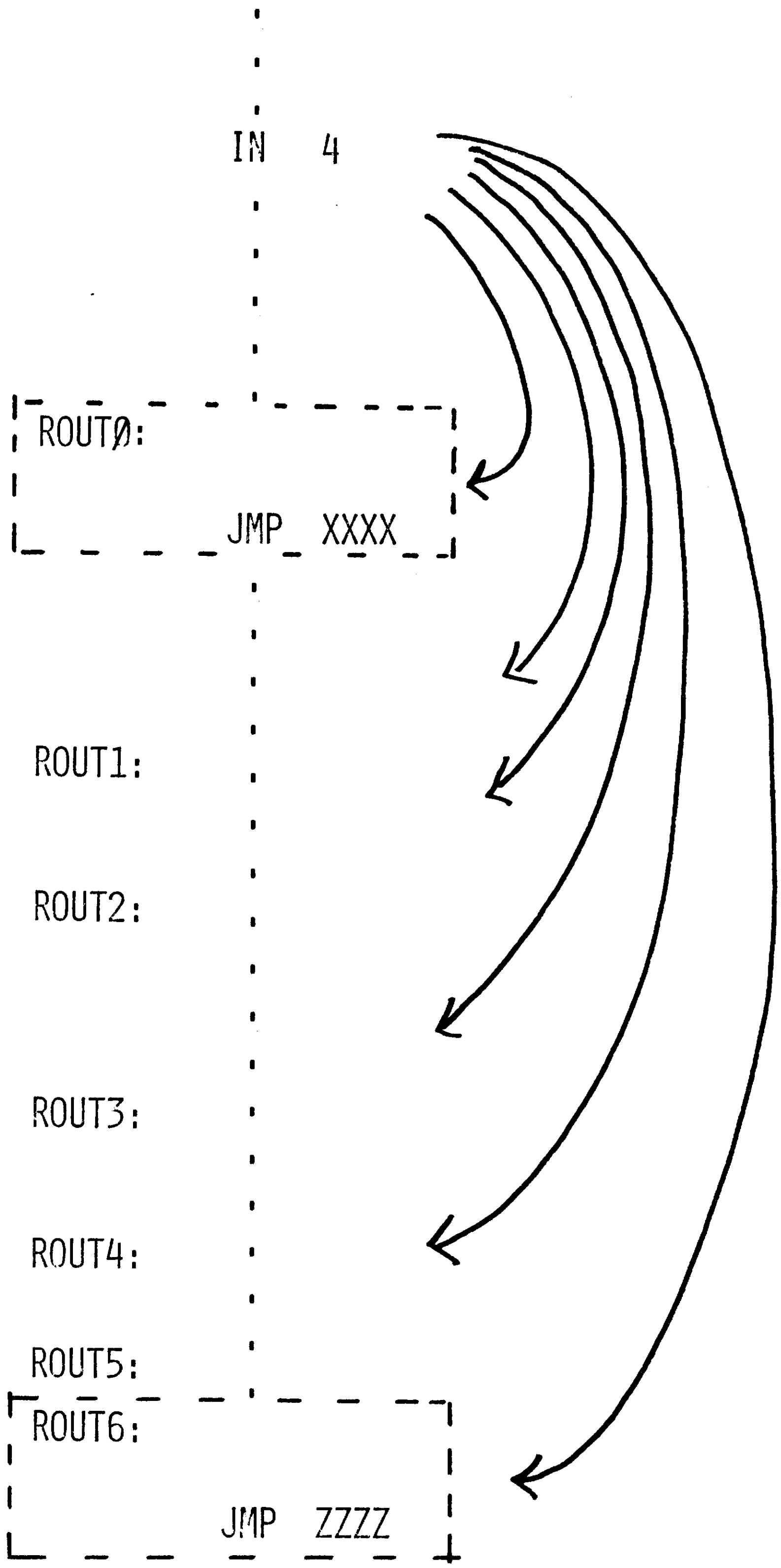
- 9 MICRO-SEC INITIAL RESPONSE ASSUMING XTHL EXECUTION
- 5.5 MICRO-SEC FOR RST INSTRUCTION
- 5 MICRO-SEC FOR JMP INSTRUCTION
- 49 MICRO-SEC FOR SAVE/RESTORE/ENABLE

68.5 MICRO-SEC TOTAL

" NOTES "

LOOK UP TABLE

INPUT VALUE
MUST SELECT
SPECIFIC
ROUTINE



```
                        .
                        .
                        .
                        IN    4
                        .
                        .
MEMORY                  .                      MEMORY
LOCATION                PCHL                   CONTENTS
  700       TABLE:      DW    ROUTØ              800
  702                   DW    ROUT1              850
  704                   DW    ROUT2              888
  706                   DW    ROUT3              943
  708                   DW    ROUT4              9AB
  70A                   DW    ROUT5             1012
  70C                   DW    ROUT6             10FE
                        .
                        .
                        .
                        .
                        .
                        .
                        .
                        .
                        .
  800       ROUTØ:      .
                        .
  850       ROUT1:      .
                        .
  888       ROUT2:      .
                        .
  943       ROUT3:      .
                        .
  9AB       ROUT4:      .
                        .
 1012       ROUT5:      .
                        .
 10FE       ROUT6:      .
```

```
        IN      4
        LXI     H,TABLE




        PCHL
TABLE:  DW      ROUTØ
        .
        .
        .
```

```
        IN      4
        LXI     H,TABLE
        DAD     B



        PCHL
TABLE:  DW      ROUTØ
        .
        .
        .
```

```
        IN      4

        MOV     C,A
        MVI     B,0
        LXI     H,TABLE
        DAD     B




TABLE:  DW      ROUTØ
        .
        .
        .
```

```
        IN      4
        ADD     A
        MOV     C,A
        MVI     B,0
        LXI     H,TABLE
        DAD     B
        MOV     C,M
        INX     H
        MOV     H,M
        MOV     L,C
        PCHL
TABLE:  DW      ROUTØ
        .
        .
        .
```

BYTE MOVE SUBROUTINE

```
;
; ENTER WITH FROM ADDRESS IN  HL
;            TO   ADDRESS IN  BC
;            COUNTER      IN   D
;
MOVE:   MOV     A,M      ; GET FROM BYTE
        STAX    B        ; STORE IT
        INX     H        ; FROM = FROM + 1
        INX     B        ; TO   = TO   + 1
        DCR     D        ; COUNT = COUNT - 1
        JNZ     MOVE     ; DONE ?
        RET              ; YES
```

DECIMAL ADJUST ACCUMULATOR

PURPOSE: CONVERTS RESULT OF BINARY ADDITION TO BCD VALUES.

RULE 1: IF $A_{LS4} > 9$ <u>OR</u> IF A.C. = 1 THEN ADD 6.

RULE 2: IF $A_{MS4} > 9$ <u>OR</u> IF C = 1 THEN ADD 60.

EXAMPLES:

DECIMAL		BCD	
29		0010 1001	
+ 1		1	
30		0010 1010	
		0110	(RULE 1)
		0011 0000	
18		0001 1000	
+18		0001 1000	
36		0011 0000	
		0110	(RULE 1)
		0011 0110	
72		0111 0010	
+93		1001 0011	
165	$\boxed{1}$	0000 0101	
		0110 0000	(RULE 2)
	$\boxed{1}$	0110 0101	
94		1001 0100	
+07		0000 0111	
101		1001 1011	
		0110	(RULE 1)
		1010 0001	
		0110 0000	(RULE 2)
	$\boxed{1}$	0000 0001	

DIRECT LOAD / STORE INSTRUCTIONS

8 - BIT

LDA ADDRESS $\quad A \leftarrow M(B_3\ B_2)$

STA ADDRESS $\quad M(B_3\ B_2) \leftarrow A$

3 BYTES

B_1	LDA
B_2	$ADRS_{LSB}$
B_3	$ADRS_{MSB}$

16 - BIT

LHLD ADDRESS

$H \leftarrow M(B_3\ B_2 + 1)$

$L \leftarrow M(B_3\ B_2)$

SHLD ADDRESS

$M(B_3\ B_2 + 1) \leftarrow H$

$M(B_3\ B_2) \leftarrow L$

3 BYTES

B_1	LHLD
B_2	$ADRS_{LSB}$
B_3	$ADRS_{MSB}$

SPECIALS

XTHL: HL ↔ TOS

EXAMPLE: 3 BYTE CALL FOR SINGLE CHARACTER PRINT

```
        ⋮
        CALL    COMC            ;MOV NEXT ADR TO C
        DB      'A'
        ⋮
COMC:   XTHL                    ;GET RET ADDR
        MOV     C,M             ;GET PARAMETER
        INX     H               ;BUMP RET ADDR
        XTHL
        CALL    CO              ;PRINT CHAR IN C REG.
        RET
        ⋮
```

XCHG: HL ↔ DE

EXAMPLE: PERFORM SAME JOB ON 2 BLOCKS OF MEMORY

```
        ⋮
        MVI     B,COUNT         ;COUNT = 2X BYTE COUNT
        LXI     H,ADRX1         ;BEG.ADR X BLOCK
        LXI     D,ADRY1         ;BEG.ADR Y BLOCK
LOOP:   CALL    JOB             ;ON HL ADDRESS
        INX     H
        XCHG
        DCR     B
        JNZ     LOOP
        ⋮
JOB:    MOV     A,M
        ⋮
        RET
```

REVIEW

		8080 ASSEMBLY LANGUAGE PROGRAMMING MANUAL PAGE NUMBER
RST	RESTART	37
EI	ENABLE INTERRUPT	38
DI	DISABLE INTERRUPT	38
DAA	DECIMAL ADJUST ACCUMULATOR	15
PCHL	PC <--- HL	31
DAD	HL <--- HL + RP	24
STAX / LDAX	STORE/LOAD A THRU BC OR DE	17
STA / LDA	STORE/LOAD A THRU MEMORY	30
SHLD / LHLD	STORE/LOAD HL THRU MEMORY	30 / 31
XTHL	HL TOS	25
XCHG	HL DE	24
DW	DEFINE WORDS	14

INSTRUCTION SET

Summary of Processor Instructions

Mnemonic	Description	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Clock[2] Cycles
		Instruction Code[1]								
$MOV_{r1,r2}$	Move register to register	0	1	D	D	D	S	S	S	5
MOV M, r	Move register to memory	0	1	1	1	0	S	S	S	7
MOV r, M	Move memory to register	0	1	D	D	D	1	1	0	7
HLT	Halt	0	1	1	1	0	1	1	0	7
MVI r	Move immediate register	0	0	D	D	D	1	1	0	7
MVI M	Move immediate memory	0	0	1	1	0	1	1	0	10
INR r	Increment register	0	0	D	D	D	1	0	0	5
DCR r	Decrement register	0	0	D	D	D	1	0	1	5
INR M	Increment memory	0	0	1	1	0	1	0	0	10
DCR M	Decrement memory	0	0	1	1	0	1	0	1	10
ADD r	Add register to A	1	0	0	0	0	S	S	S	4
ADC r	Add register to A with carry	1	0	0	0	1	S	S	S	4
SUB r	Subtract register from A	1	0	0	1	0	S	S	S	4
SBB r	Subtract register from A with borrow	1	0	0	1	1	S	S	S	4
ANA r	And register with A	1	0	1	0	0	S	S	S	4
XRA r	Exclusive Or register with A	1	0	1	0	1	S	S	S	4
ORA r	Or register with A	1	0	1	1	0	S	S	S	4
CMP r	Compare register with A	1	0	1	1	1	S	S	S	4
ADD M	Add memory to A	1	0	0	0	0	1	1	0	7
ADC M	Add memory to A with carry	1	0	0	0	1	1	1	0	7
SUB M	Subtract memory from A	1	0	0	1	0	1	1	0	7
SBB M	Subtract memory from A with borrow	1	0	0	1	1	1	1	0	7
ANA M	And memory with A	1	0	1	0	0	1	1	0	7
XRA M	Exclusive Or memory with A	1	0	1	0	1	1	1	0	7
ORA M	Or memory with A	1	0	1	1	0	1	1	0	7
CMP M	Compare memory with A	1	0	1	1	1	1	1	0	7
ADI	Add immediate to A	1	1	0	0	0	1	1	0	7
ACI	Add immediate to A with carry	1	1	0	0	1	1	1	0	7
SUI	Subtract immediate from A	1	1	0	1	0	1	1	0	7
SBI	Subtract immediate from A with borrow	1	1	0	1	1	1	1	0	7
ANI	And immediate with A	1	1	1	0	0	1	1	0	7
XRI	Exclusive Or immediate with A	1	1	1	0	1	1	1	0	7
ORI	Or immediate with A	1	1	1	1	0	1	1	0	7
CPI	Compare immediate with A	1	1	1	1	1	1	1	0	7
RLC	Rotate A left	0	0	0	0	0	1	1	1	4
RRC	Rotate A right	0	0	0	0	1	1	1	1	4
RAL	Rotate A left through carry	0	0	0	1	0	1	1	1	4
RAR	Rotate A right through carry	0	0	0	1	1	1	1	1	4
JMP	Jump unconditional	1	1	0	0	0	0	1	1	10
JC	Jump on carry	1	1	0	1	1	0	1	0	10
JNC	Jump on no carry	1	1	0	1	0	0	1	0	10
JZ	Jump on zero	1	1	0	0	1	0	1	0	10
JNZ	Jump on no zero	1	1	0	0	0	0	1	0	10
JP	Jump on positive	1	1	1	1	0	0	1	0	10
JM	Jump on minus	1	1	1	1	1	0	1	0	10
JPE	Jump on parity even	1	1	1	0	1	0	1	0	10
JPO	Jump on parity odd	1	1	1	0	0	0	1	0	10
CALL	Call unconditional	1	1	0	0	1	1	0	1	17
CC	Call on carry	1	1	0	1	1	1	0	0	11/17
CNC	Call on no carry	1	1	0	1	0	1	0	0	11/17
CZ	Call on zero	1	1	0	0	1	1	0	0	11/17
CNZ	Call on no zero	1	1	0	0	0	1	0	0	11/17
CP	Call on positive	1	1	1	1	0	1	0	0	11/17
CM	Call on minus	1	1	1	1	1	1	0	0	11/17
CPE	Call on parity even	1	1	1	0	1	1	0	0	11/17
CPO	Call on parity odd	1	1	1	0	0	1	0	0	11/17
RET	Return	1	1	0	0	1	0	0	1	10
RC	Return on carry	1	1	0	1	1	0	0	0	5/11
RNC	Return on no carry	1	1	0	1	0	0	0	0	5/11

Mnemonic	Description	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Clock[2] Cycles
		Instruction Code[1]								
RZ	Return on zero	1	1	0	0	1	0	0	0	5/11
RNZ	Return on no zero	1	1	0	0	0	0	0	0	5/11
RP	Return on positive	1	1	1	1	0	0	0	0	5/11
RM	Return on minus	1	1	1	1	1	0	0	0	5/11
RPE	Return on parity even	1	1	1	0	1	0	0	0	5/11
RPO	Return on parity odd	1	1	1	0	0	0	0	0	5/11
RST	Restart	1	1	A	A	A	1	1	1	11
IN	Input	1	1	0	1	1	0	1	1	10
OUT	Output	1	1	0	1	0	0	1	1	10
LXI B	Load immediate register Pair B & C	0	0	0	0	0	0	0	1	10
LXI D	Load immediate register Pair D & E	0	0	0	1	0	0	0	1	10
LXI H	Load immediate register Pair H & L	0	0	1	0	0	0	0	1	10
LXI SP	Load immediate stack pointer	0	0	1	1	0	0	0	1	10
PUSH B	Push register Pair B & C on stack	1	1	0	0	0	1	0	1	11
PUSH D	Push register Pair D & E on stack	1	1	0	1	0	1	0	1	11
PUSH H	Push register Pair H & L on stack	1	1	1	0	0	1	0	1	11
PUSH PSW	Push A and Flags on stack	1	1	1	1	0	1	0	1	11
POP B	Pop register pair B & C off stack	1	1	0	0	0	0	0	1	10
POP D	Pop register pair D & E off stack	1	1	0	1	0	0	0	1	10
POP H	Pop register pair H & L off stack	1	1	1	0	0	0	0	1	10
POP PSW	Pop A and Flags off stack	1	1	1	1	0	0	0	1	10
STA	Store A direct	0	0	1	1	0	0	1	0	13
LDA	Load A direct	0	0	1	1	1	0	1	0	13
XCHG	Exchange D & E, H & L Registers	1	1	1	0	1	0	1	1	4
XTHL	Exchange top of stack, H & L	1	1	1	0	0	0	1	1	18
SPHL	H & L to stack pointer	1	1	1	1	1	0	0	1	5
PCHL	H & L to program counter	1	1	1	0	1	0	0	1	5
DAD B	Add B & C to H & L	0	0	0	0	1	0	0	1	10
DAD D	Add D & E to H & L	0	0	0	1	1	0	0	1	10
DAD H	Add H & L to H & L	0	0	1	0	1	0	0	1	10
DAD SP	Add stack pointer to H & L	0	0	1	1	1	0	0	1	10
STAX B	Store A indirect	0	0	0	0	0	0	1	0	7
STAX D	Store A indirect	0	0	0	1	0	0	1	0	7
LDAX B	Load A indirect	0	0	0	0	1	0	1	0	7
LDAX D	Load A indirect	0	0	0	1	1	0	1	0	7
INX B	Increment B & C registers	0	0	0	0	0	0	1	1	5
INX D	Increment D & E registers	0	0	0	1	0	0	1	1	5
INX H	Increment H & L registers	0	0	1	0	0	0	1	1	5
INX SP	Increment stack pointer	0	0	1	1	0	0	1	1	5
DCX B	Decrement B & C	0	0	0	0	1	0	1	1	5
DCX D	Decrement D & E	0	0	0	1	1	0	1	1	5
DCX H	Decrement H & L	0	0	1	0	1	0	1	1	5
DCX SP	Decrement stack pointer	0	0	1	1	1	0	1	1	5
CMA	Compliment A	0	0	1	0	1	1	1	1	4
STC	Set carry	0	0	1	1	0	1	1	1	4
CMC	Compliment carry	0	0	1	1	1	1	1	1	4
DAA	Decimal adjust A	0	0	1	0	0	1	1	1	4
SHLD	Store H & L direct	0	0	1	0	0	0	1	0	16
LHLD	Load H & L direct	0	0	1	0	1	0	1	0	16
EI	Enable Interrupts	1	1	1	1	1	0	1	1	4
DI	Disable interrupt	1	1	1	1	0	0	1	1	4
NOP	No-operation	0	0	0	0	0	0	0	0	4

NOTES: 1. DDD or SSS – 000 B – 001 C – 010 D – 011 E – 100 H – 101 L – 110 Memory – 111 A.
2. Two possible cycle times, (5/11) indicate instruction cycles dependent on condition flags.

MACROS

1. DEFINITION

```
MOVRT   MACRO
        RRC
        RRC
        RRC
        RRC
        ANI     ØFH
        ENDM
```

2. REFERENCE

```
        MOV     A,M
        MOVRT
        MOV     M,A
```

3. EXPANSION

```
7E        MOV     A,M
0F   +    MOVRT
0F   +
0F   +
0F   +
E6   +
0F   +
77        MOV     M,A
```

MACRO PARAMETERS

DEFINITION -

```
SHV     MACRO   REG,AMT
        MVI     REG,AMT ; REG -- SHIFT COUNT
LOOP:   RRC             ; ROTATE RIGHT
        ANI     7FH     ; CLEAR BIT 7
        DCR     REG     ; DECREMENT SHIFT COUNT
        JNZ     LOOP
        ENDM
```

REFERENCE -

```
        :
        :
        SHV     C,5
        :
        :
        SHV     D,8
        :
        :
        SHV     B,4
        :
        :
        :
```

BREAKPOINTS

- DEBUGGING AID
- OPTIONAL PART OF THE SYSTEM MONITOR'S GO COMMAND

  G ADDRESS,BKPT1 ADDRESS,BKPT2 ADDRESS

- WHEN A BKPT IS ENCOUNTERED:

  ALL REGISTERS/FLAGS SAVED
  BREAKPOINTS CLEARED
  CONTROL TRANSFERRED TO THE SYSTEM MONITOR
  * BKPT ADDRESS (PRINTED BY SYTEM MONITOR)

- X (EXAMINE/MODIFY REGISTERS) COMMAND USED TO INSPECT
- G,BKPT1 ADDRESS,BKPT2 ADDRESS

  STARTS EXECUTION FROM PREVIOUS BKPT ADDRESS

- TWO BKPTS ALLOW BRACKETING OF CONDITIONAL INSTRUCTIONS
- FRONT PANEL MAY BE USED TO PROVIDE RANDOM BREAKPOINT

  INT Ø SWITCH GENERATES RST Ø

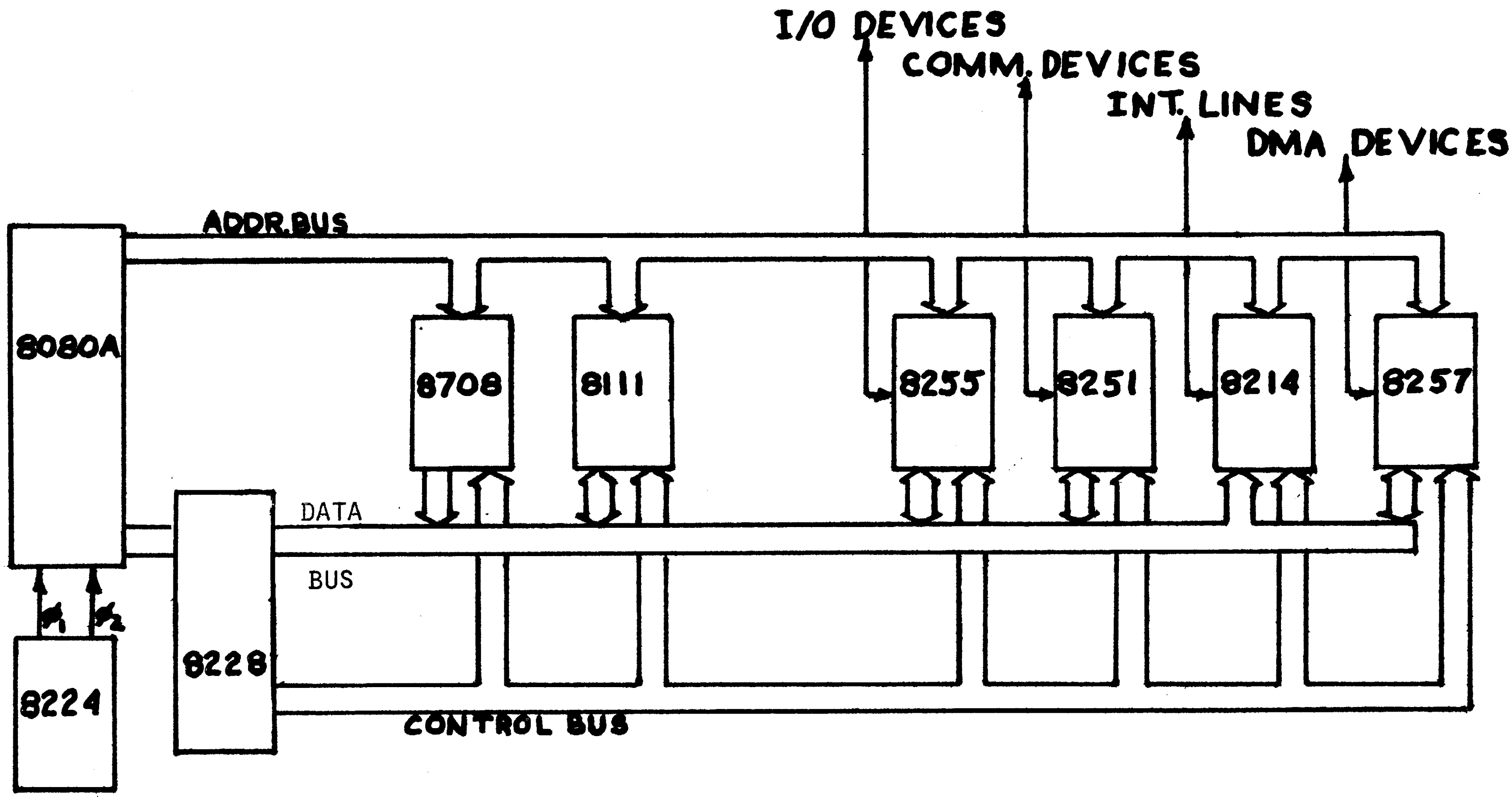
- BKPT1 AND BKPT2 ADDRESSES MUST BE AT FIRST BYTE OF MULTIPLE BYTE INSTRUCTIONS

PART VI

PERIPHERALS AND DESIGN

DESIGN GUIDELINES

• DEFINE SYSTEM PROBLEM	SYSTEM SPECIFICATION BASIC SYSTEM DIAGRAM
• DEFINE PERIPHERAL EQUIPMENT	I/O PORT ASSIGNMENT RAM, ROM MEMORY SIZE FINAL HARDWARE DIAGRAM
• FLOWCHART BASIC SOLUTION	FLOWCHARTS PL/M OR ASBM. LANG. CODING
HARDWARE & SOFTWARE	DEBUG PROGRAMS & PROTOTYPE

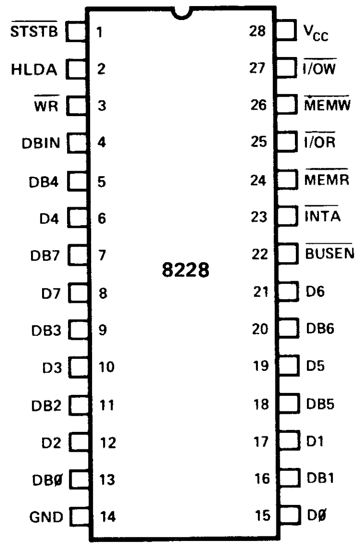

I/O DEVICES
COMM. DEVICES
INT. LINES
DMA DEVICES
ADDR. BUS
8080A
8708
8111
8255
8251
8214
8257
DATA
BUS
8224
8228
CONTROL BUS
ϕ_1
ϕ_2

8080 FAMILY

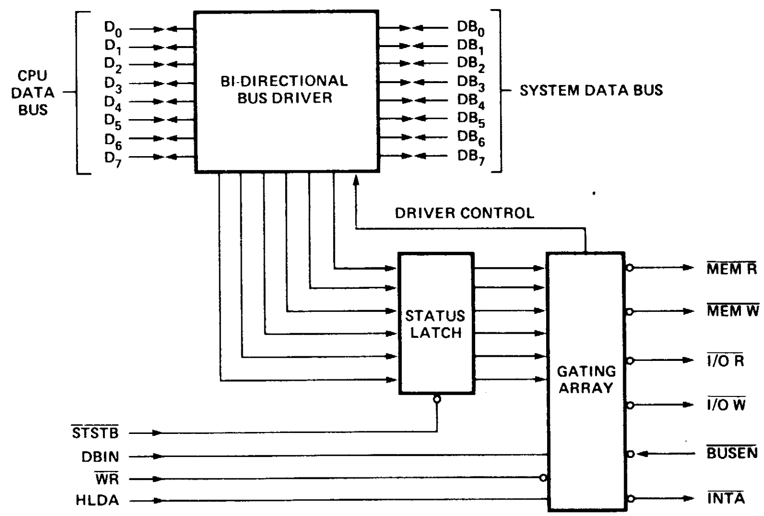
SYSTEM CONTROLLER AND BUS DRIVER FOR 8080A CPU

- **Built-in Bi-Directional Bus Driver for Data Bus Isolation**
- **Allows the use of Multiple Byte Instructions (e.g. CALL) for Interrupt Acknowledge**
- **User Selected Single Level Interrupt Vector (RST 7)**

PIN CONFIGURATION

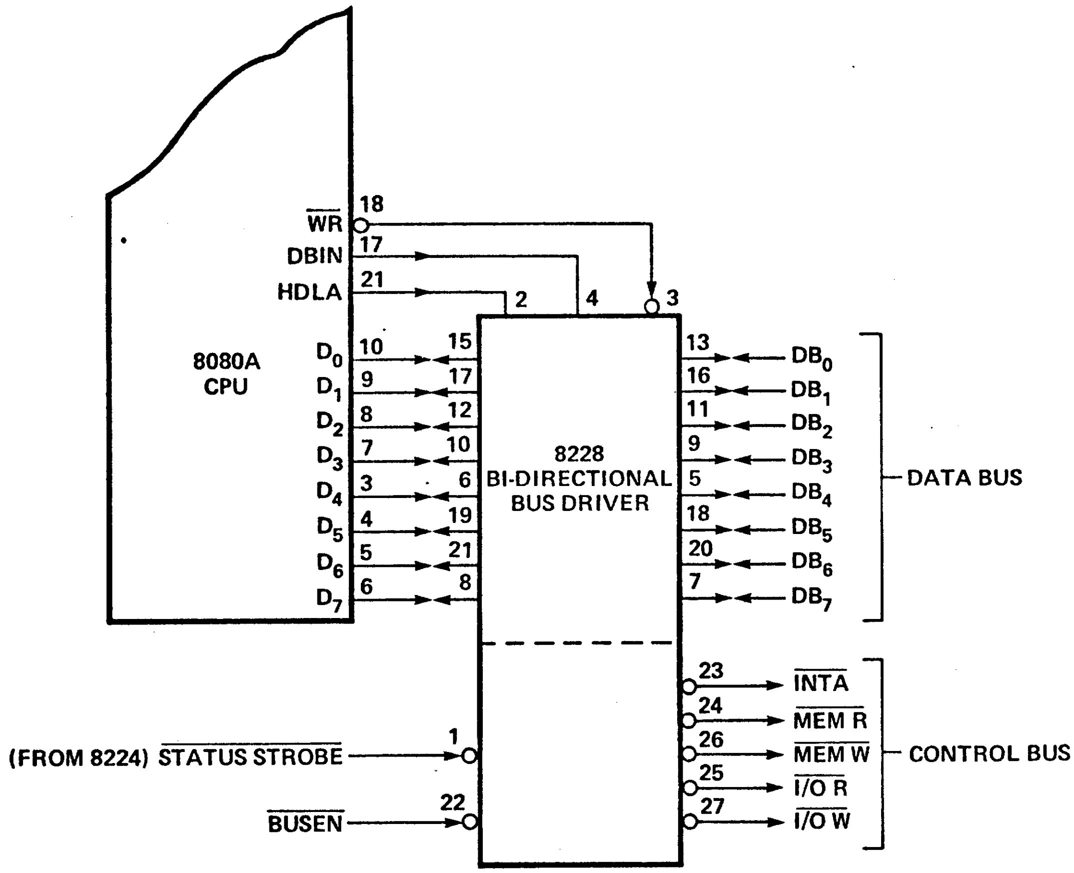


8228 BLOCK DIAGRAM



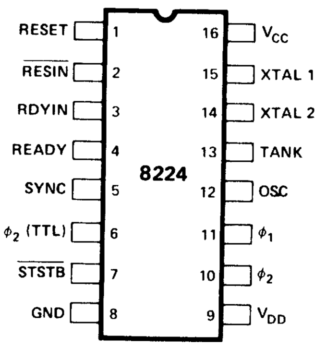
PIN NAMES

D7-D0	DATA BUS (8080 SIDE)	$\overline{\text{INTA}}$	INTERRUPT ACKNOWLEDGE
DB7-DB0	DATA BUS (SYSTEM SIDE)	HLDA	HLDA (FROM 8080)
$\overline{\text{I/OR}}$	I/O READ	$\overline{\text{WR}}$	WR (FROM 8080)
$\overline{\text{I/OW}}$	I/O WRITE	$\overline{\text{BUSEN}}$	BUS ENABLE INPUT
$\overline{\text{MEMR}}$	MEMORY READ	$\overline{\text{STSTB}}$	STATUS STROBE (FROM 8224)
$\overline{\text{MEMW}}$	MEMORY WRITE	V_{CC}	+5V
DBIN	DBIN (FROM 8080)	GND	0 VOLTS

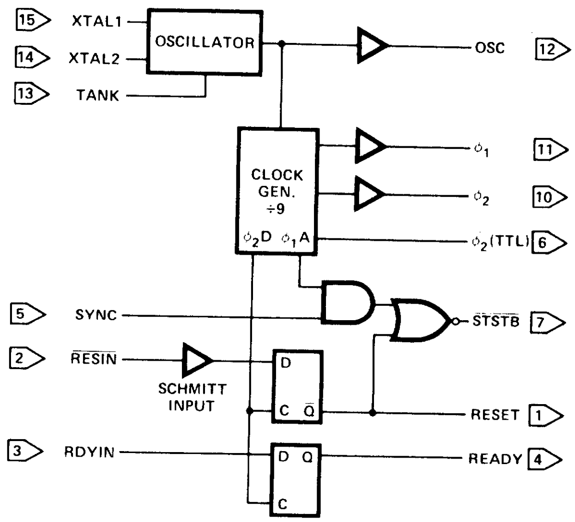

8080A
CPU
WR
DBIN
HDLA
D0
D1
D2
D3
D4
D5
D6
D7
8228
BI-DIRECTIONAL
BUS DRIVER
DB0
DB1
DB2
DB3
DB4
DB5
DB6
DB7
DATA BUS
(FROM 8224) STATUS STROBE
BUSEN
INTA
MEM R
MEM W
I/O R
I/O W
CONTROL BUS

CLOCK GENERATOR AND DRIVER FOR 8080A CPU

PIN CONFIGURATION



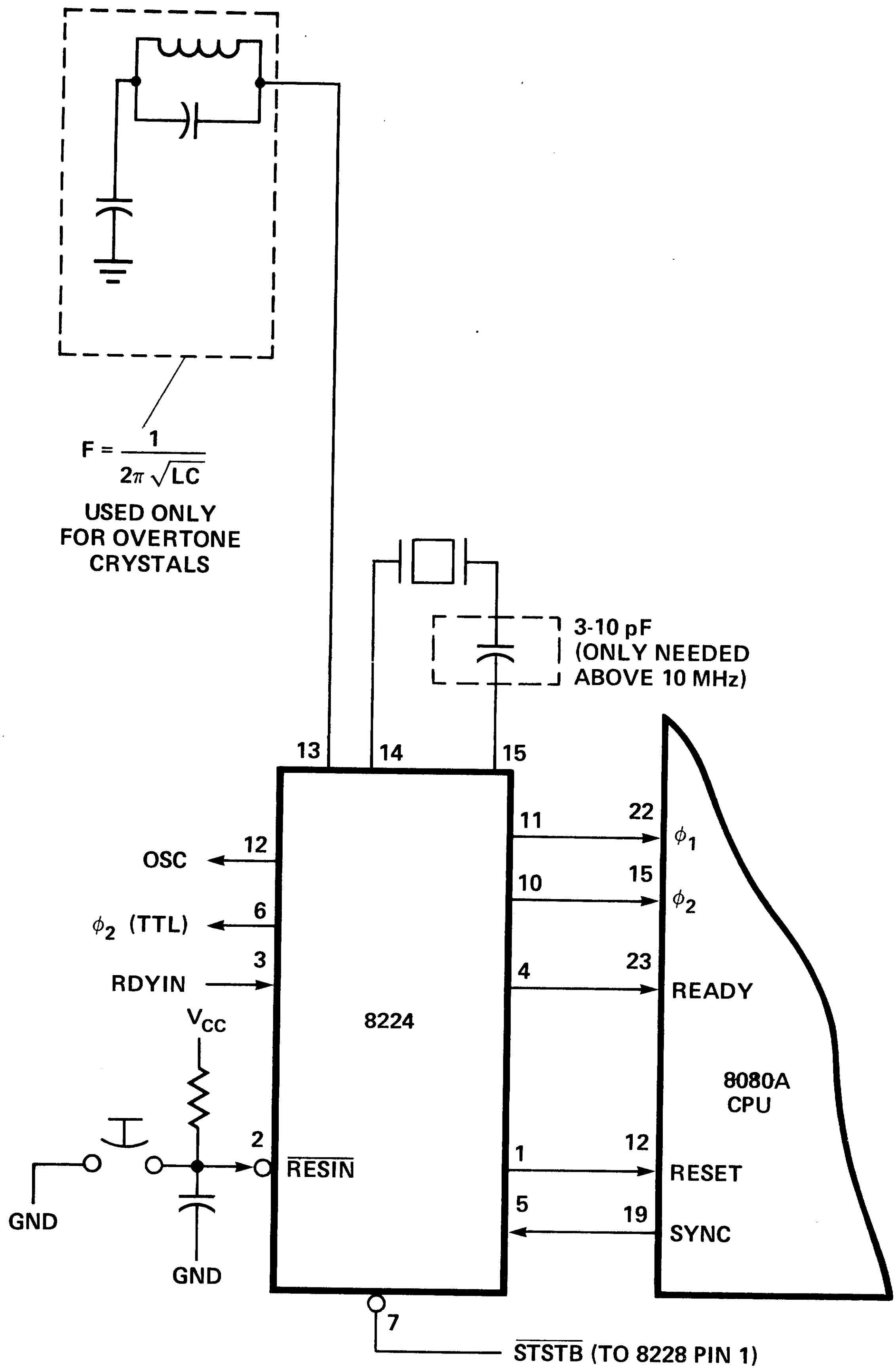
BLOCK DIAGRAM



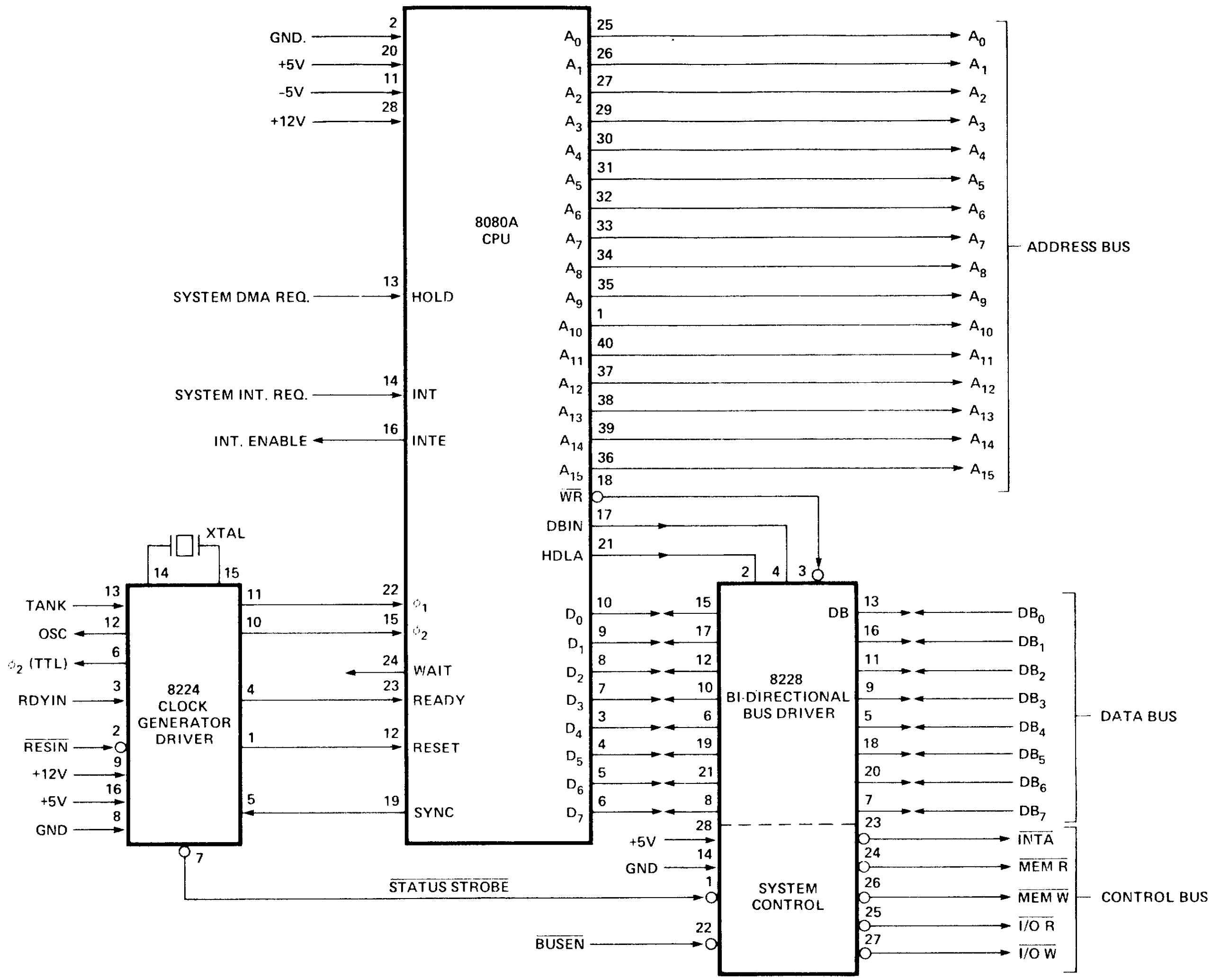
PIN NAMES

$\overline{RESIN}$	RESET INPUT
RESET	RESET OUTPUT
RDYIN	READY INPUT
READY	READY OUTPUT
SYNC	SYNC INPUT
$\overline{STSTB}$	STATUS STB (ACTIVE LOW)
ϕ_1	8080 CLOCKS
ϕ_2	

XTAL 1	CONNECTIONS FOR CRYSTAL
XTAL 2	
TANK	USED WITH OVERTONE XTAL
OSC	OSCILLATOR OUTPUT
ϕ_2 (TTL)	ϕ_2 CLK (TTL LEVEL)
V_{CC}	+5V
V_{DD}	+12V
GND	0V

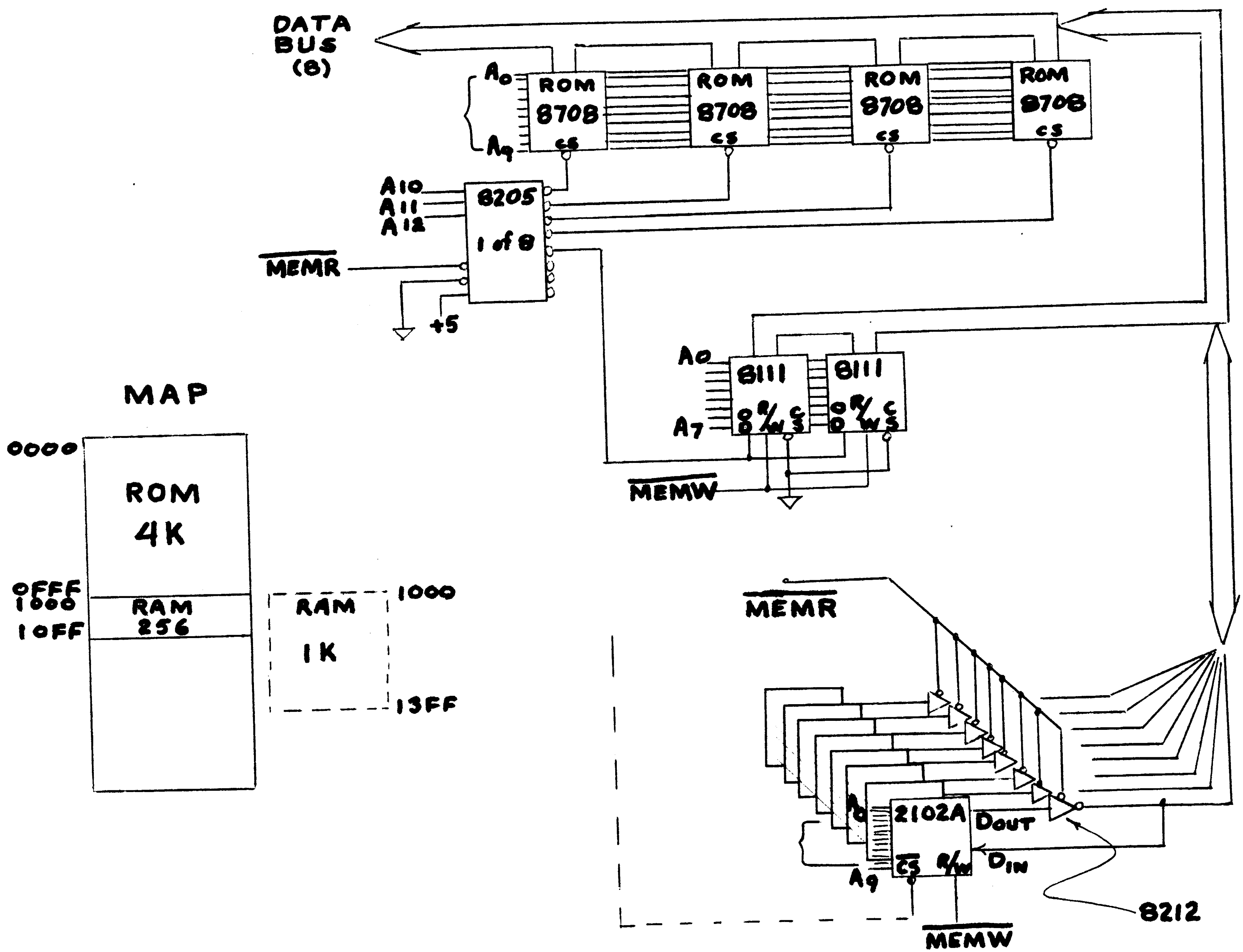


F = 1 / 2π √LC
USED ONLY
FOR OVERTONE
CRYSTALS
3-10 pF
(ONLY NEEDED
ABOVE 10 MHz)
13
14
15
12
OSC
6
φ2 (TTL)
3
RDYIN
VCC
8224
2
RESIN
GND
GND
7
STSTB (TO 8228 PIN 1)
11
22
φ1
10
15
φ2
4
23
READY
8080A
CPU
1
12
RESET
5
19
SYNC

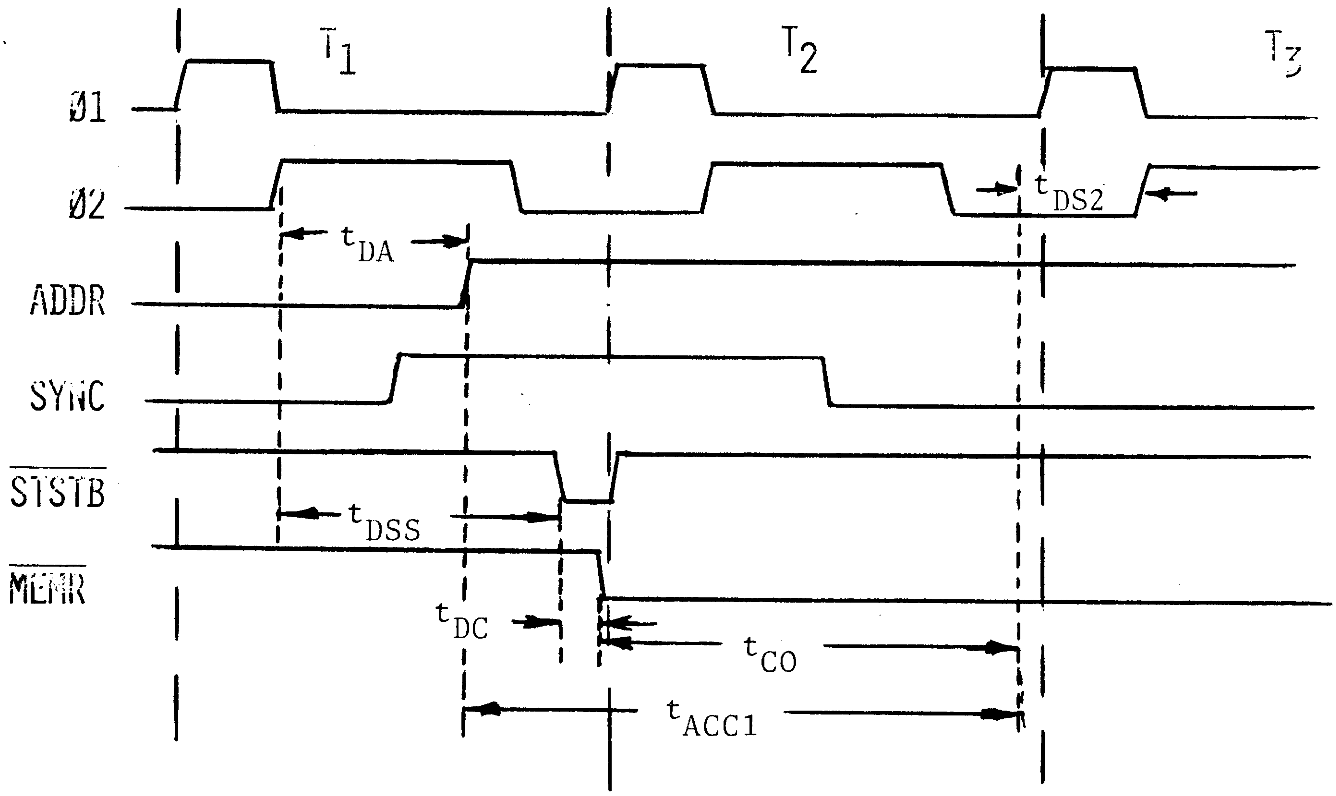


8080A CPU Standard Interface

" NOTES "


DATA
BUS
(8)
ROM
8708
CS
ROM
8708
CS
ROM
8708
CS
ROM
8708
CS
A0
A9
A10
A11
A12
8205
1 of 8
MEMR
+5
A0
A7
8111
OD R/W CS
8111
OD R/W CS
MEMW
MEMR
2102A
DOUT
DIN
A0
A9
CS
R/W
MEMW
8212
MAP
0000
ROM
4K
0FFF
1000
RAM
256
10FF
RAM
1K
1000
13FF

IS A WAIT STATE NEEDED?



$$t_{CO_{MAX}} = 2t_{CY_{MIN}} - t_{DSS_{MAX}} - t_{DC_{MAX}} - t_{DS2_{MIN}}$$

$$= 960 - 326 - 60 - 150$$

$$= 424 \text{ ns}$$

$$t_{ACC1_{MAX}} = 2t_{CY_{MIN}} - t_{DA_{MAX}} - t_{DS2_{MIN}}$$

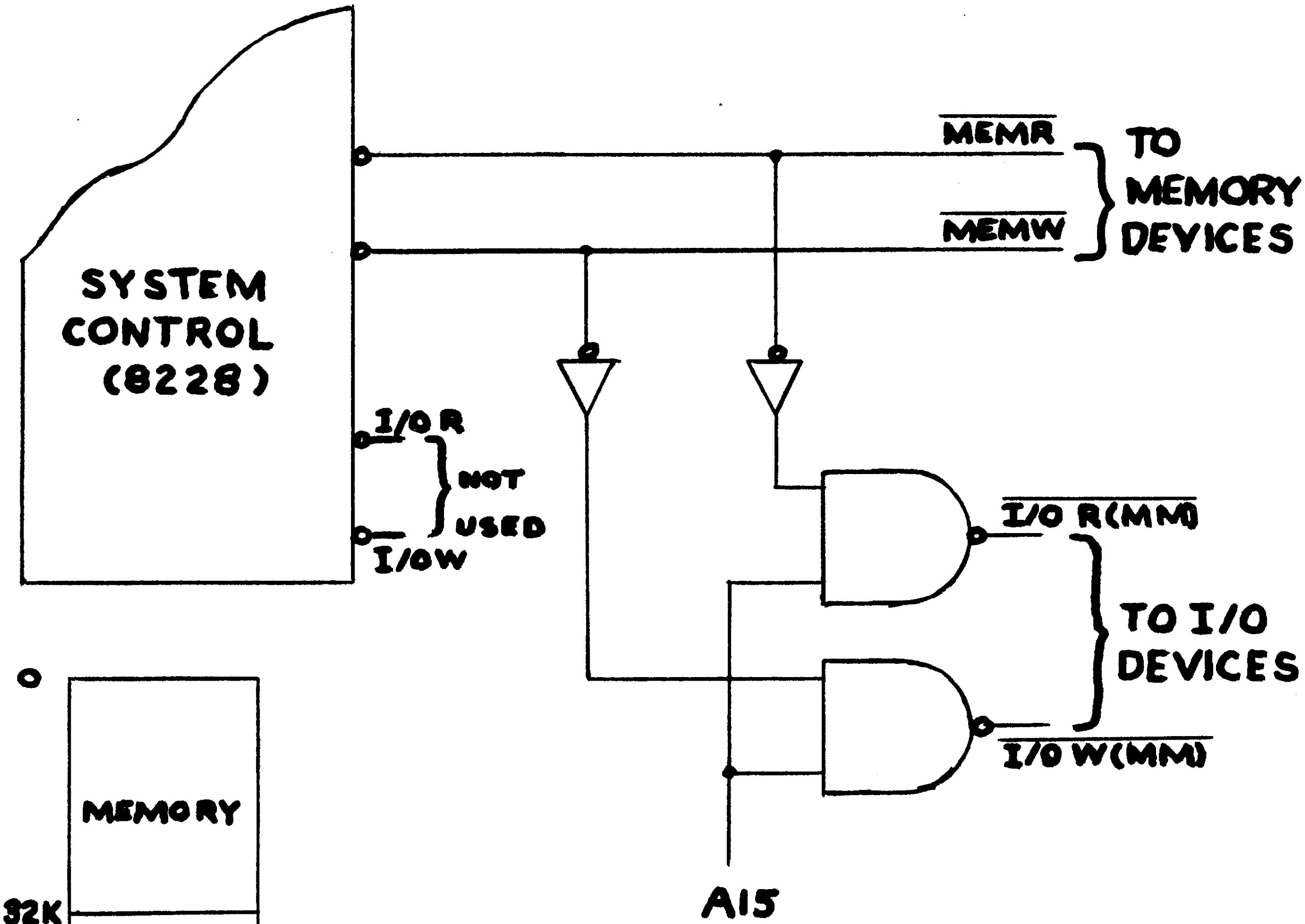
$$= 960 - 200 - 150$$

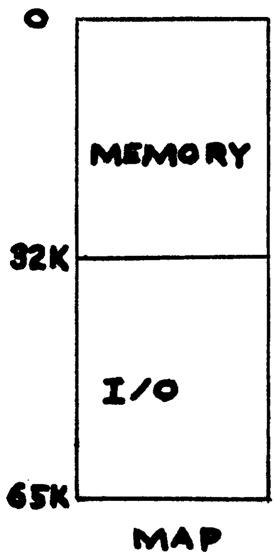
$$= 610 \text{ ns}$$

ACCESS TIMES*	t_{CO}	t_{ACC}	CAT.	TYPE
8080A	424	610		
8080A-2	244	455		
8080A-1	134	340		
1702A	900	1300	PROM	256 x 8
8708	120	450	PROM	1K z 8
8111-2	650	850	RAM	256 STATIC
8102A-4	230	450	RAM	1K STATIC
8107B-4	250	270	RAM	4K DYM

*REF: SEPTEMBER '75 8080 USER'S MANUAL

MEMORY MAPPED I/O



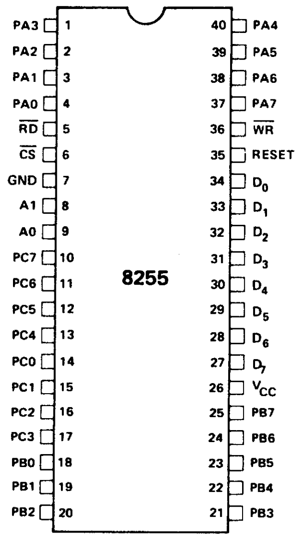


" NOTES "

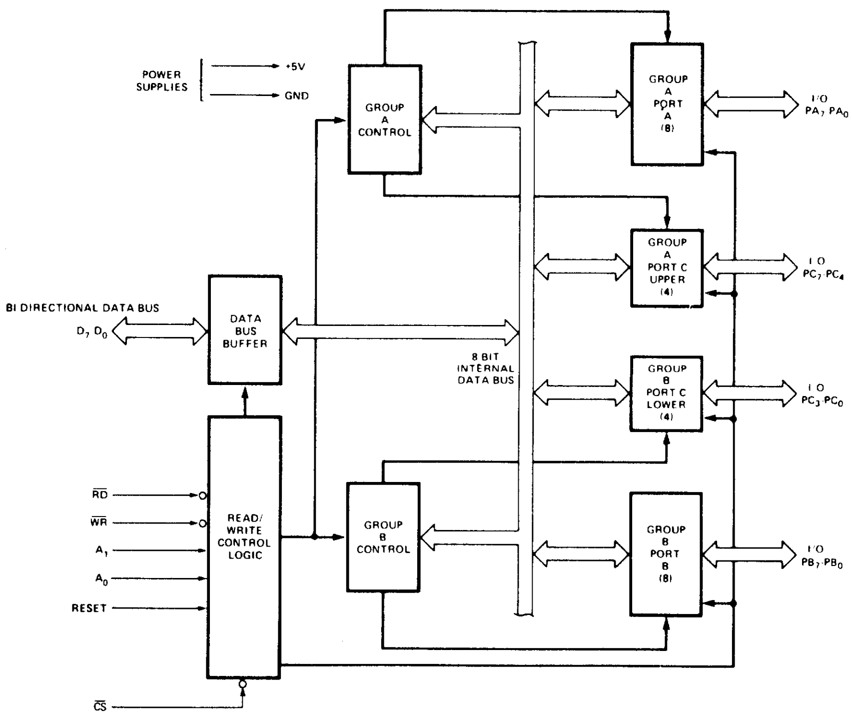
PROGRAMMABLE PERIPHERAL INTERFACE

- **24 Programmable I/O Pins**
- **Completely TTL Compatible**
- **Direct Bit Set/Reset Capability Easing Control Application Interface**

PIN CONFIGURATION



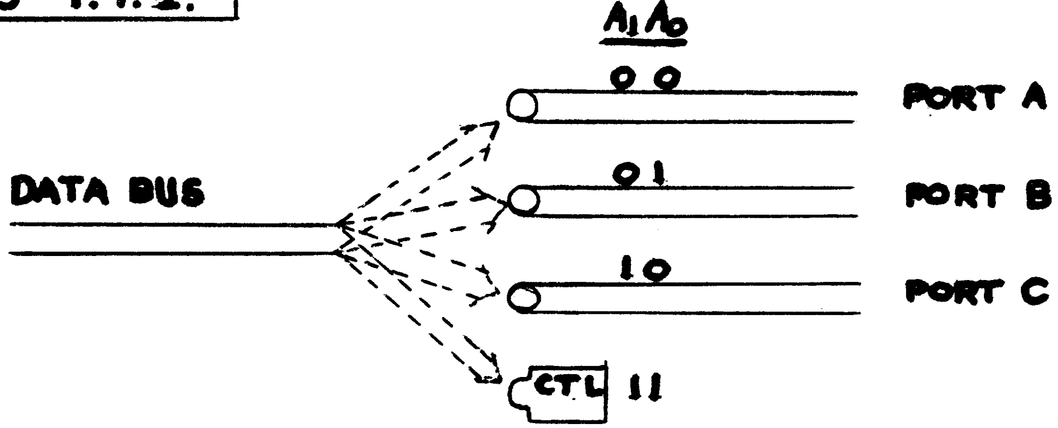
8255 BLOCK DIAGRAM



PIN NAMES

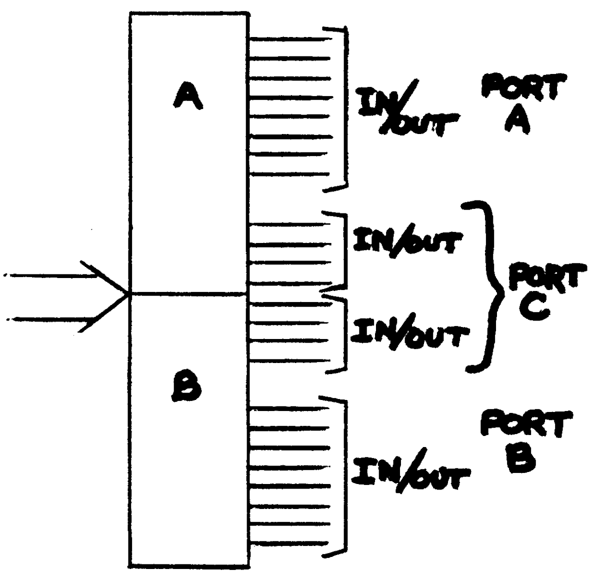
D_7–D_0	DATA BUS (BI-DIRECTIONAL)
RESET	RESET INPUT
$\overline{CS}$	CHIP SELECT
$\overline{RD}$	READ INPUT
$\overline{WR}$	WRITE INPUT
A0, A1	PORT ADDRESS
PA7-PA0	PORT A (BIT)
PB7-PB0	PORT B (BIT)
PC7-PC0	PORT C (BIT)
V_{CC}	+5 VOLTS
GND	Ø VOLTS

8255 P.P.I.

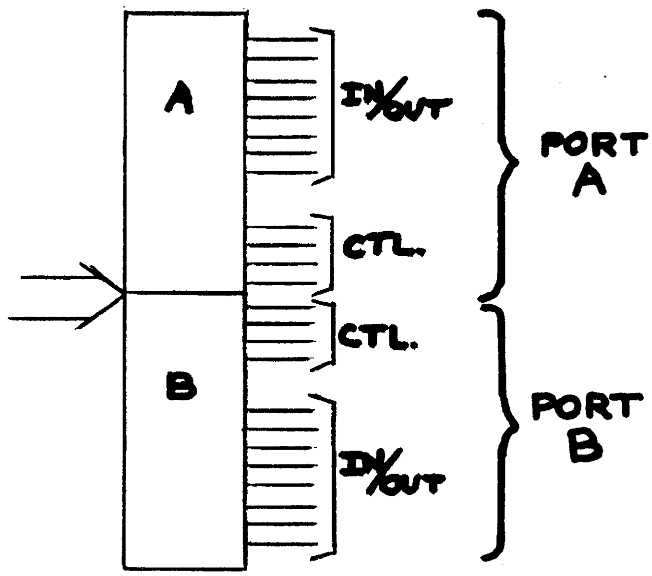


THREE MODES AVAILABLE

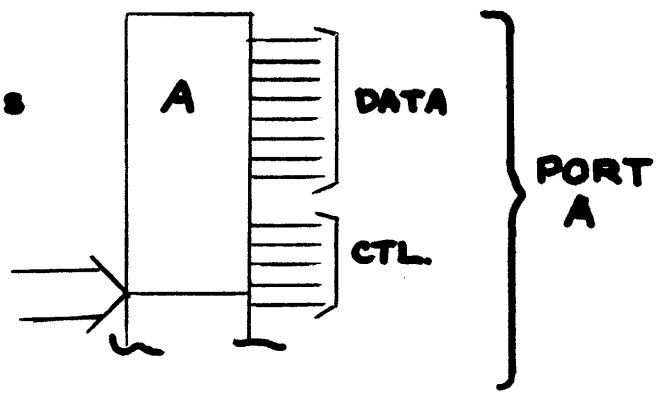
MODE 0:
3 I/O PORTS-NO CTL. LINES



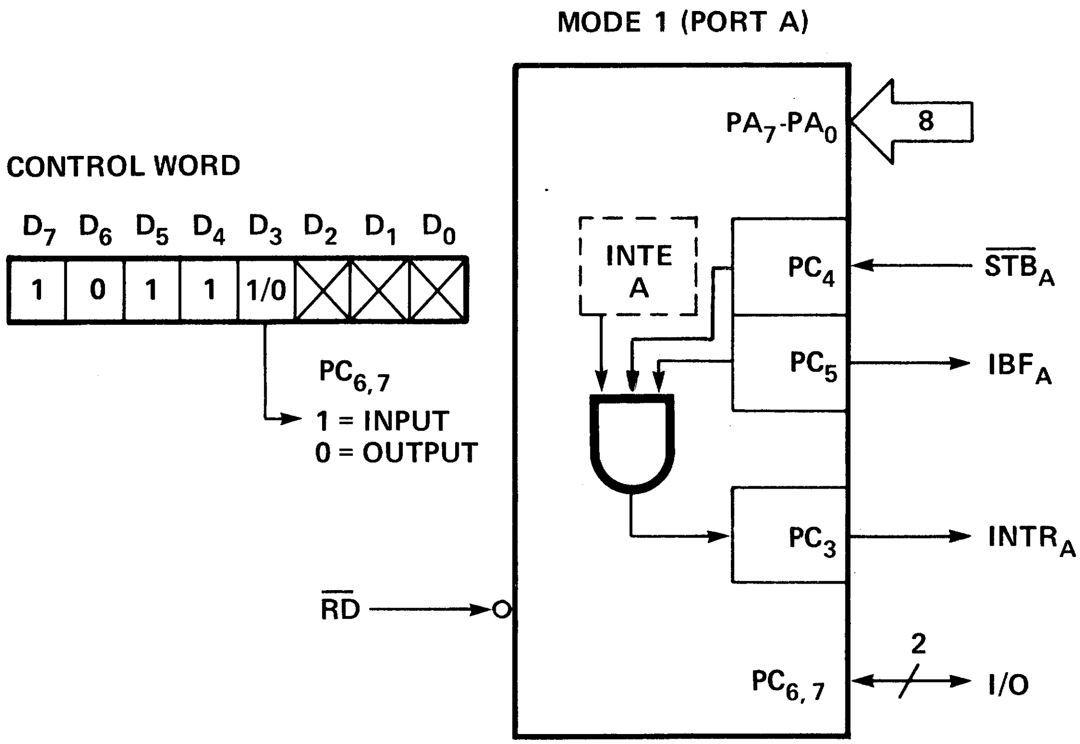
MODE 1:
2 I/O PORTS - 8 CTL. LINES

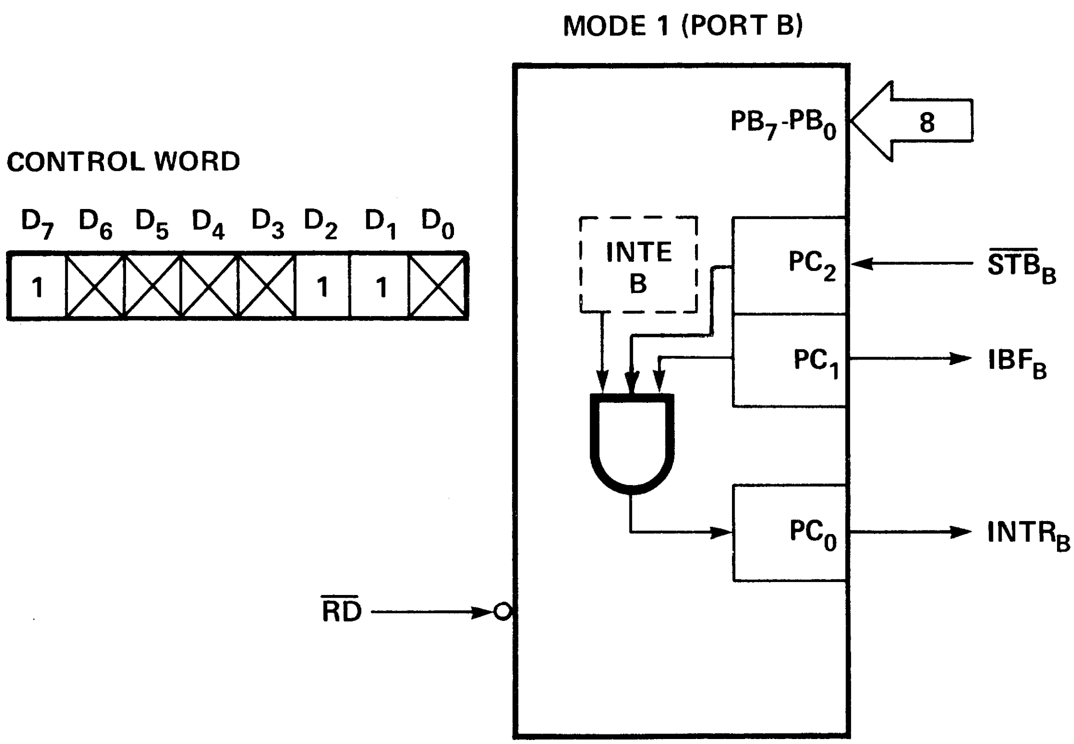


MODE 2:
1 I/O PORT - 5 CTL. LINES



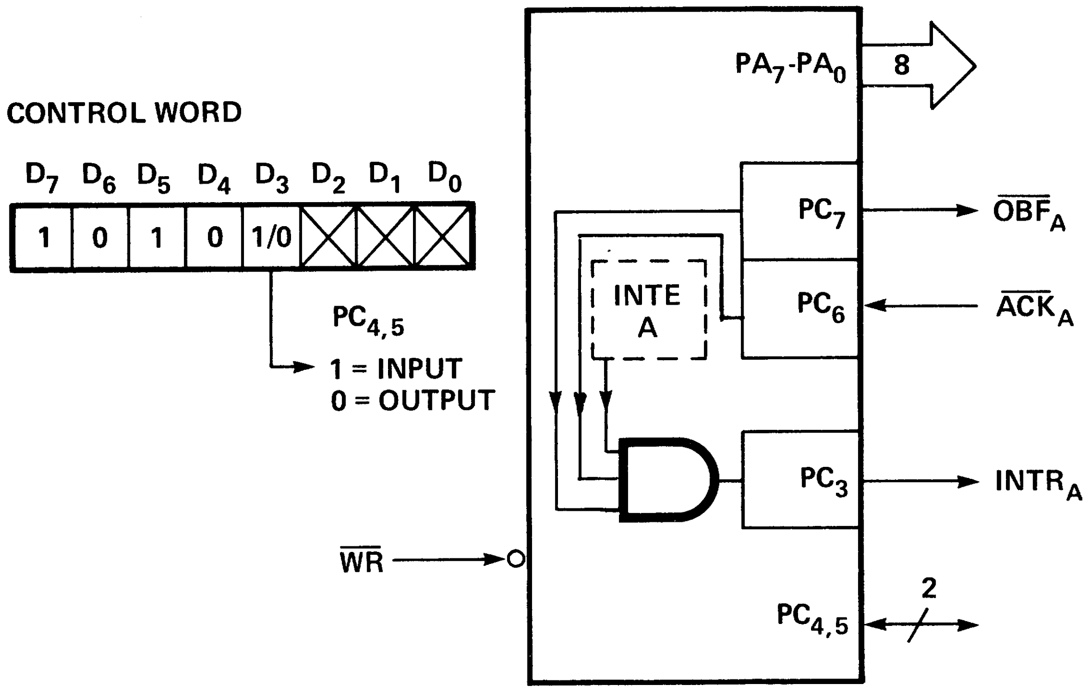
MODE 1 INPUT


MODE 1 (PORT A)
CONTROL WORD
D7 D6 D5 D4 D3 D2 D1 D0
1 0 1 1 1/0
PC6, 7
1 = INPUT
0 = OUTPUT
PA7-PA0
8
INTE
A
PC4
STBA
PC5
IBFA
PC3
INTRA
RD
PC6, 7
2
I/O

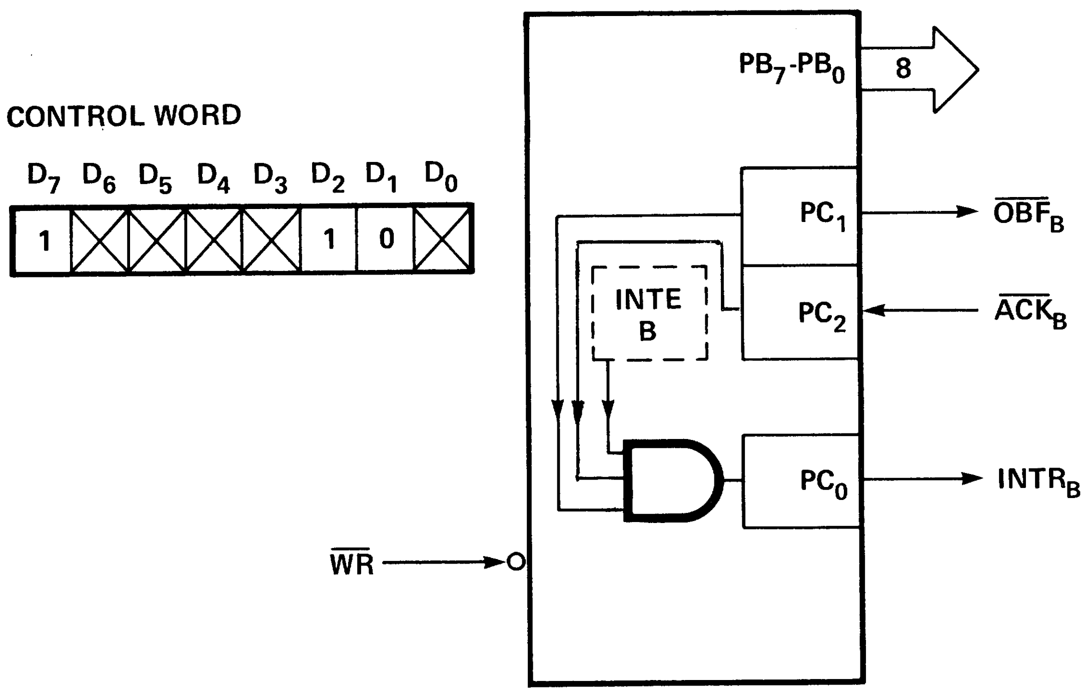

MODE 1 (PORT B)
CONTROL WORD
D7 D6 D5 D4 D3 D2 D1 D0
1 1 1
PB7-PB0
8
INTE
B
PC2
STBB
PC1
IBFB
PC0
INTRB
RD

MODE 1 OUTPUT

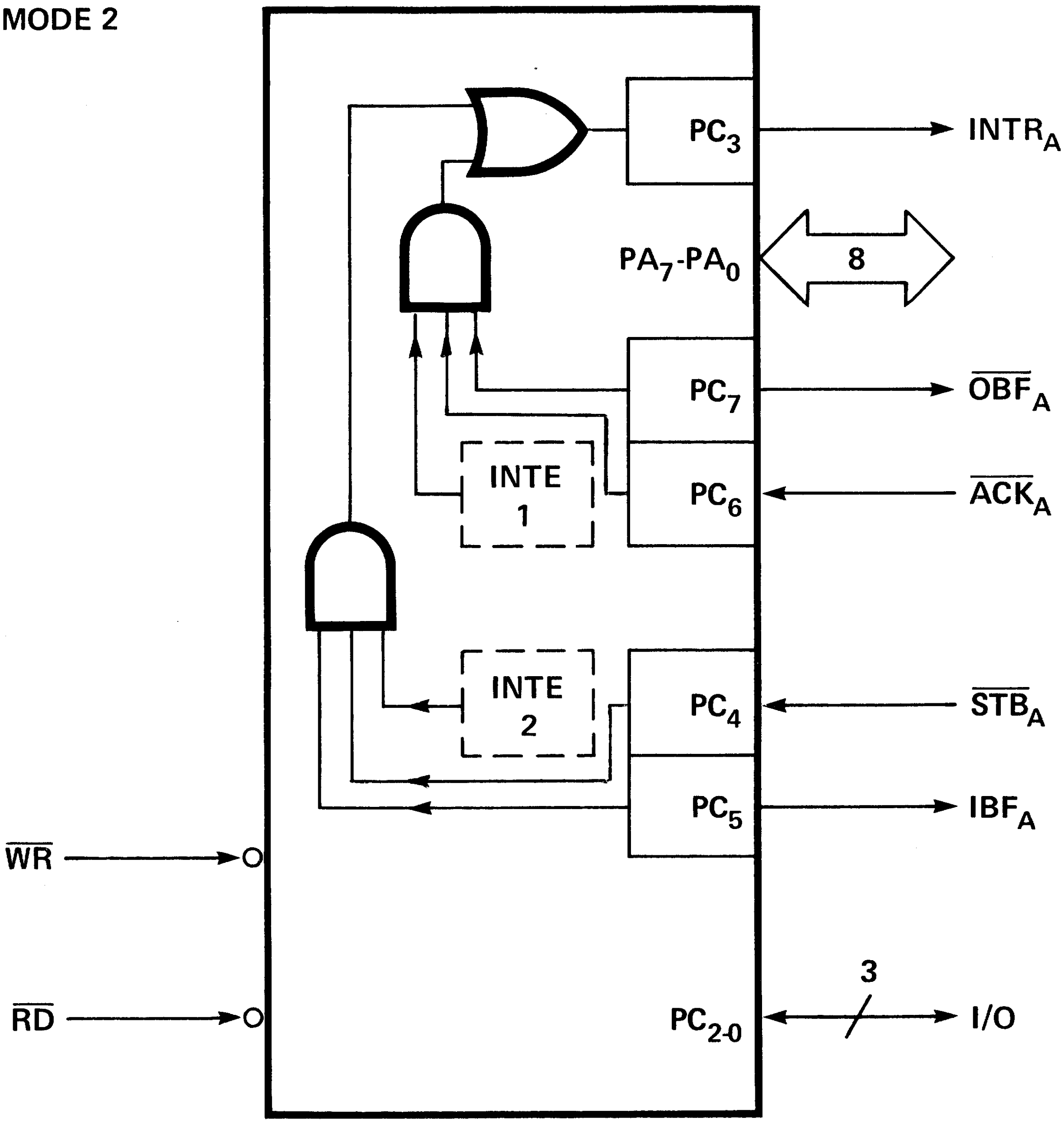
MODE 1 (PORT A)



MODE 1 (PORT B)

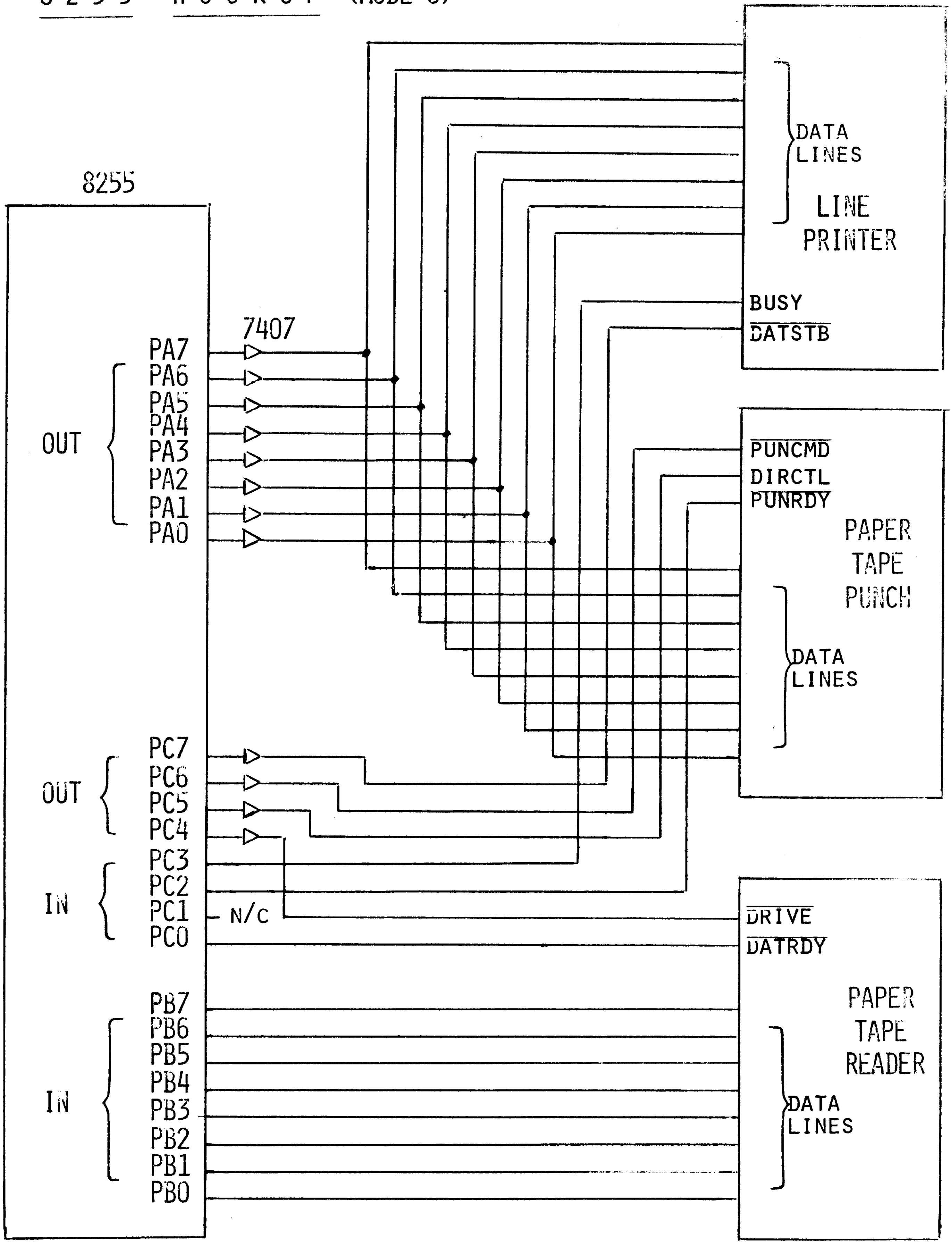


MODE 2



PC_3
$INTR_A$
PA_7-PA_0
8
PC_7
$\overline{OBF}_A$
INTE 1
PC_6
$\overline{ACK}_A$
INTE 2
PC_4
$\overline{STB}_A$
PC_5
IBF_A
$\overline{WR}$
$\overline{RD}$
$PC_{2\text{-}0}$
3
I/O

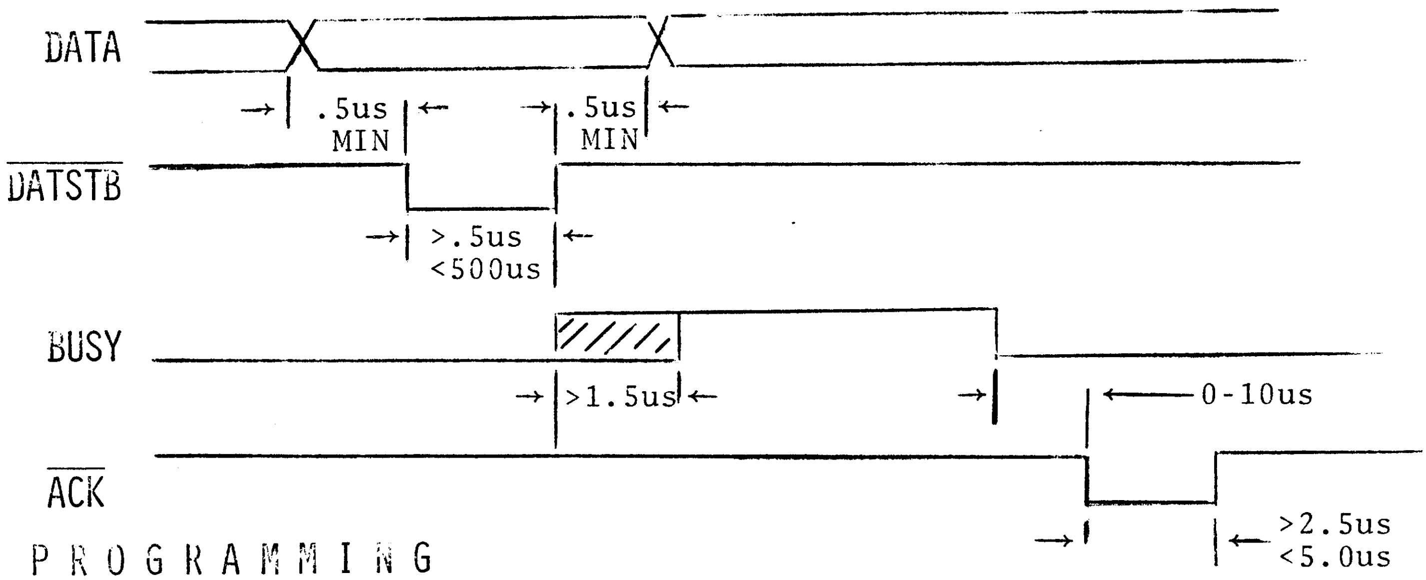
8255 HOOKUP (MODE 0)



PROGRAMMING

```
MVI A,83H      ;CONTROL WORD
OUT ØF7        ;CTL OUT
```

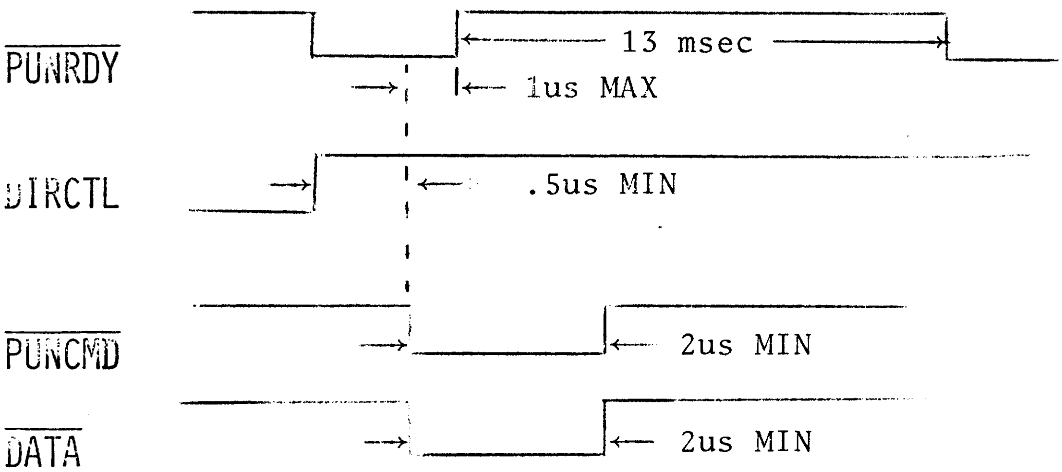
LINE PRINTER (CENTRONIX 500)



PROGRAMMING

```
LPT:  IN   ØF6H        ;READ PORT C (STATUS)
      ANI  8H          ;UNMASK BIT 3
      JNZ LPT
      MOV  A,C         ;C REG HAS DATA
      OUT ØF4H         ;DATA OUT PORT A
      MVI  A,7FH       ;STROBE LOW
      OUT ØF6H         ;STROBE OUT PORT C
      MVI  A,ØFFH      ;STROBE HI
      OUT ØF6H         ;STROBE OUT PORT C
      RET
```

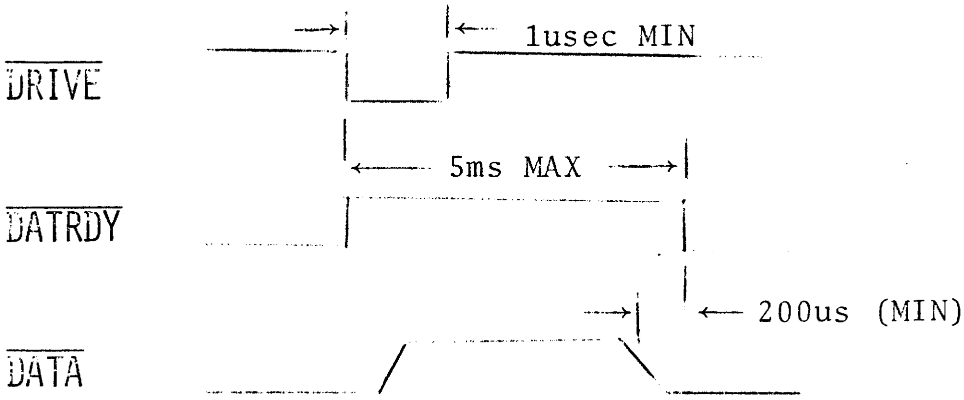
PAPER TAPE PUNCH (REMEX)



PROGRAMMING

```
PTP:  IN   ØF6H        ;READ PORT C (STATUS)
      ANI  4H          ;UNMASK BIT 2
      JNZ  PTP
      MOV  A,C         ;C REG HAS DATA
      CMA              ;COMPLEMENT
      OUT  ØF4H        ;DATA OUT PORT A
      MVI  A,ØFFH      ;DIRCTL & PUNCMD HI
      OUT  ØF6H        ;DIRCTL & PUNCMD OUT
      MVI  A,ØBFH      ;DIRCTL HI - PUNCMD LO
      OUT  ØF6H        ;DIRCTL & PUNCMD OUT
      MVI  A,ØFFH      ;SAME AS ABOVE
      OUT  ØF6H
```

PAPER TAPE READER (REMEX)



PROGRAMMING

```
PTR1: MVI   A,ØEFH    ;DRIVE/LOW
      OUT   ØF6H      ;DRIVE/OUT PORT C
      MVI   A,ØFFH    ;DRIVE/HI
      OUT   ØF6H      ;DRIVE/OUT PORT C
      MVI   H,25      ;TIMEOUT = 25MS
PTR2: IN    ØF6H      ;READ PORT C (STATUS)
      ANI   1H        ;UNMASK BIT Ø
      JZ    PTR3
      CALL  DELAY     ;1MS DELAY
      DCR   H
      JNZ   PTR2
      JMP   TIMOUT
PTR3: IN    ØF5H      ;DATA IN PORT B
      CMA             ;COMPLEMENT
      RET
```

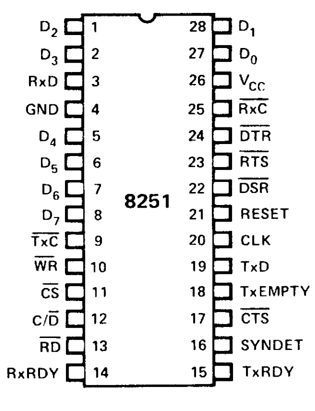
Silicon Gate MOS 8251

PROGRAMMABLE COMMUNICATION INTERFACE

- **Synchronous and Asynchronous Operation**
  - **Synchronous:**
    5-8 Bit Characters
    Internal or External Character Synchronization
    Automatic Sync Insertion
  - **Asynchronous:**
    5-8 Bit Characters
    Clock Rate — 1,16 or 64 Times Baud Rate
    Break Character Generation
    1, 1½, or 2 Stop Bits
    False Start Bit Detection
- **Baud Rate — DC to 56k Baud (Sync Mode) DC to 9.6k Baud (Async Mode)**
- **Full Duplex, Double Buffered, Transmitter and Receiver**
- **Error Detection — Parity, Overrun, and Framing**
- **Fully Compatible with 8080 CPU**
- **28-Pin DIP Package**
- **All Inputs and Outputs Are TTL Compatible**
- **Single 5 Volt Supply**
- **Single TTL Clock**

The 8251 is a Universal Synchronous/Asynchronous Receiver/Transmitter (USART) Chip designed for data communications in microcomputer systems. The USART is used as a peripheral device and is programmed by the CPU to operate using virtually any serial data transmission technique presently in use (including IBM Bi-Sync). The USART accepts data characters from the CPU in parallel format and then converts them into a continuous serial data stream for transmission. Simultaneously it can receive serial data streams and convert them into parallel data characters for the CPU. The USART will signal the CPU whenever it can accept a new character for transmission or whenever it has received a character for the CPU. The CPU can read the complete status of the USART at any time. These include data transmission errors and control signals such as SYNDET, TxEMPT. The chip is constructed using N-channel silicon gate technology.

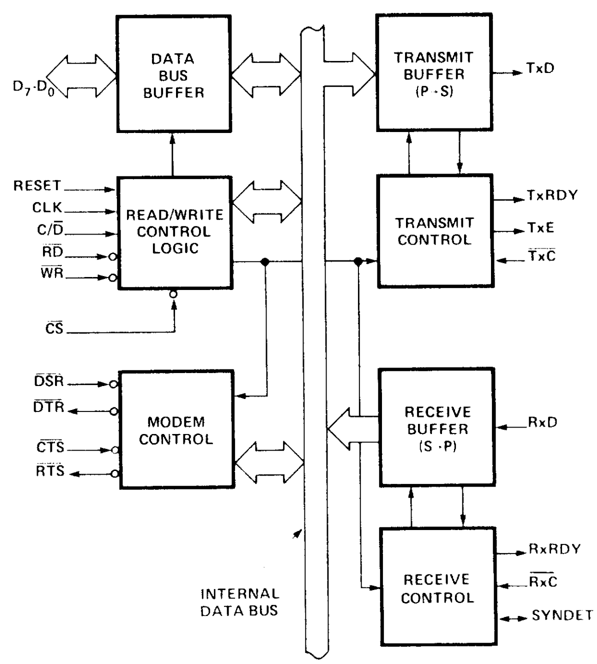
PIN CONFIGURATION



Pin Name	Pin Function
D_7-D_0	Data Bus (8 bits)
C/$\overline{D}$	Control or Data is to be Written or Read
$\overline{RD}$	Read Data Command
$\overline{WR}$	Write Data or Control Command
$\overline{CS}$	Chip Enable
CLK	Clock Pulse (TTL)
RESET	Reset
$\overline{TxC}$	Transmitter Clock
TxD	Transmitter Data
$\overline{RxC}$	Receiver Clock
RxD	Receiver Data
RxRDY	Receiver Ready (has character for 8080)
TxRDY	Transmitter Ready (ready for char. from 8080)

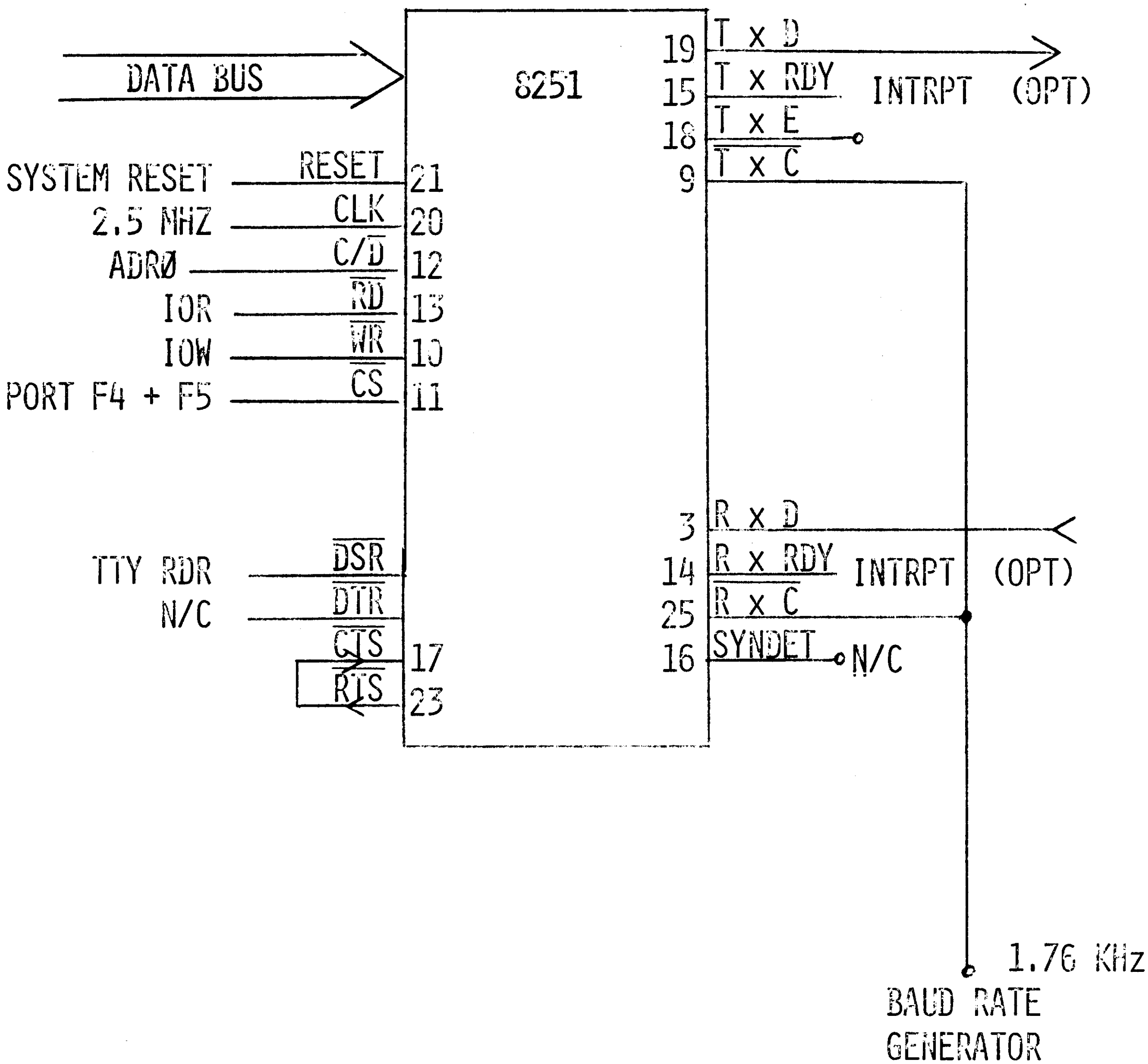
Pin Name	Pin Function
$\overline{DSR}$	Data Set Ready
$\overline{DTR}$	Data Terminal Ready
SYNDET	Sync Detect
$\overline{RTS}$	Request to Send Data
$\overline{CTS}$	Clear to Send Data
TxE	Transmitter Empty
Vcc	+5 Volt Supply
GND	Ground

BLOCK DIAGRAM



8251 CONNECTIONS

(AS DONE IN THE MDS)



8251 PROGRAMMING

	PROGRAMMING	ACCUMULATOR	DEFINITION	FUNCTION
	RESET			
DONE ONCE	OUT ØF5H	11001110	16X	MODE CONTROL WORD
			8 BITS	
			DISABLE PARITY	
			ODD PARITY	
			TWO STOP BITS	
	OUT ØF5H	00100111	TRANSMIT ENABLE	COMMAND CONTROL WORD
			$\overline{DTR}$ OUTPUT TO Ø	
			RECEIVER ENABLE	
			$\overline{RTS}$ OUTPUT TO Ø	
INPUT OPER	IN ØF5H	XXXXXX0X	RCVR BFR NOT READY	STATUS OF INPUT DEVICE
		XXXXXX1X	RCVR BFR READY	
	IN ØF4H	DATA		DATA XFER
OUTPUT OPER	IN ØF5H	XXXXXXX0	TRANSMIT NOT READY	STATUS OF OUTPUT DEVICE
		XXXXXXX1	TRANSMIT READY	
	OUT ØF4H	DATA		DATA XFER

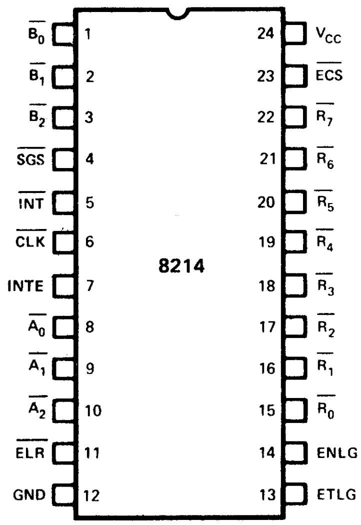
intel

Schottky Bipolar 8214

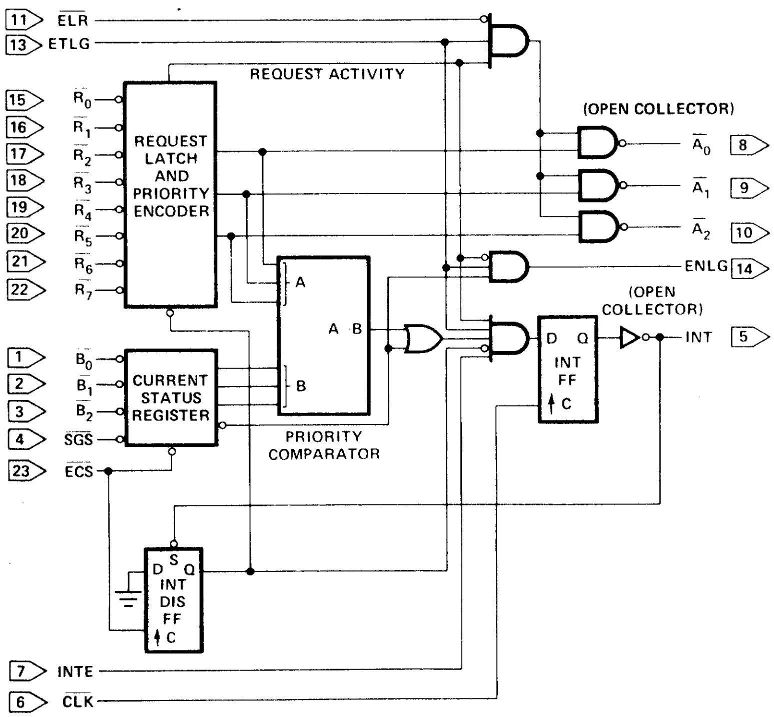
PRIORITY INTERRUPT CONTROL UNIT

- **Eight Priority Levels**
- **Current Status Register**
- **Priority Comparator**
- **Fully Expandable**
- **High Performance (50ns)**
- **24-Pin Dual In-Line Package**

PIN CONFIGURATION



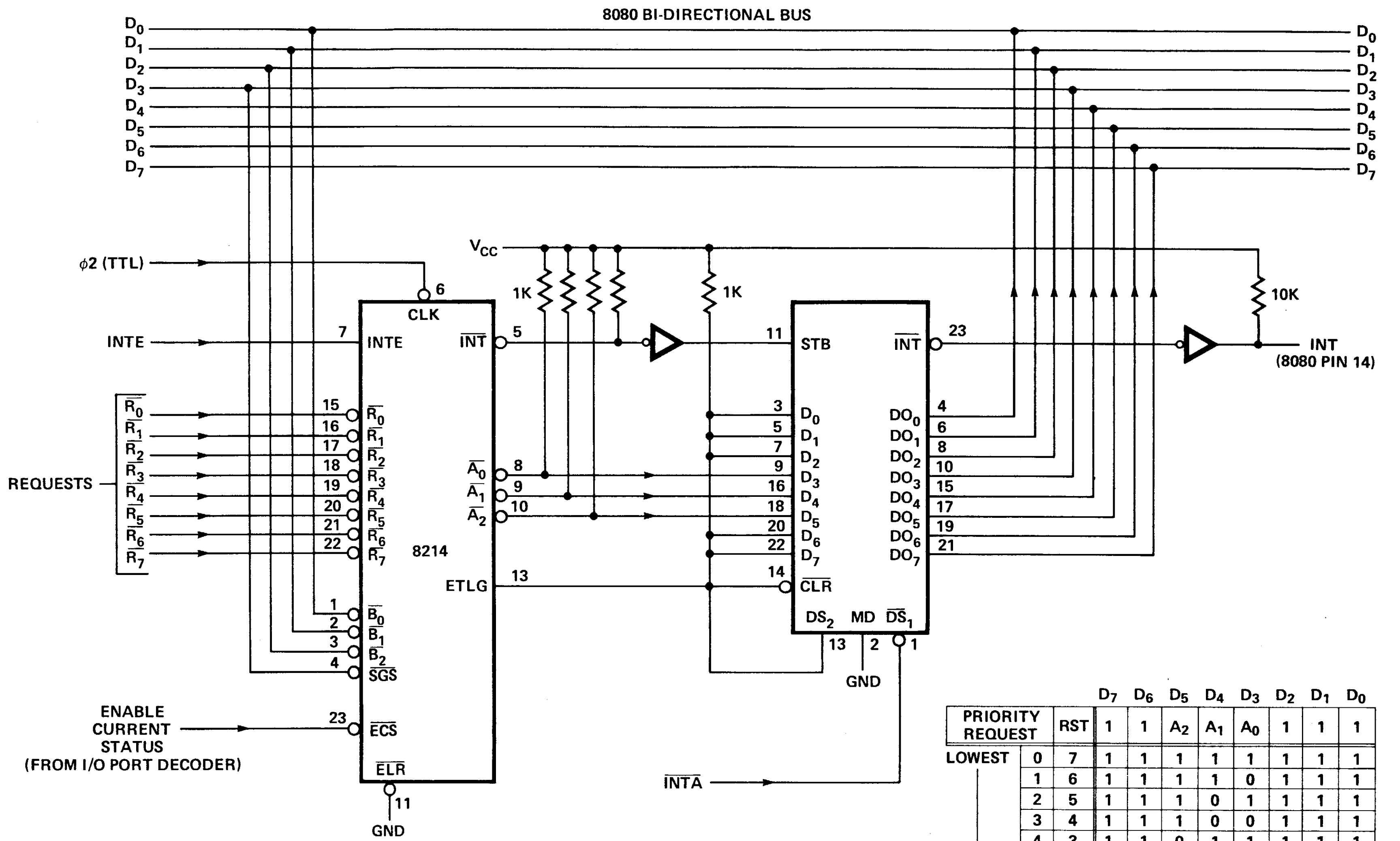
LOGIC DIAGRAM



PIN NAMES

INPUTS		
$\overline{R_0}$-$\overline{R_7}$	REQUEST LEVELS (R_7 HIGHEST PRIORITY)	
$\overline{B_0}$-$\overline{B_2}$	CURRENT STATUS	
$\overline{SGS}$	STATUS GROUP SELECT	
$\overline{ECS}$	ENABLE CURRENT STATUS	
INTE	INTERRUPT ENABLE	
$\overline{CLK}$	CLOCK (INT F-F)	
$\overline{ELR}$	ENABLE LEVEL READ	
ETLG	ENABLE THIS LEVEL GROUP	
OUTPUTS:		
$\overline{A_0}$-$\overline{A_2}$	REQUEST LEVELS	OPEN COLLECTOR
$\overline{INT}$	INTERRUPT (ACT. LOW)	OPEN COLLECTOR
ENLG	ENABLE NEXT LEVEL GROUP	

8 LEVEL CONTROLLER



PRIORITY REQUEST		RST	D_7 1	D_6 1	D_5 A_2	D_4 A_1	D_3 A_0	D_2 1	D_1 1	D_0 1
LOWEST	0	7	1	1	1	1	1	1	1	1
	1	6	1	1	1	1	0	1	1	1
	2	5	1	1	1	0	1	1	1	1
	3	4	1	1	1	0	0	1	1	1
	4	3	1	1	0	1	1	1	1	1
	5	2	1	1	0	1	0	1	1	1
	6	1	1	1	0	0	1	1	1	1
HIGHEST	7	*0	1	1	0	0	0	1	1	1

*RST 0 WILL VECTOR PROGRAM COUNTER TO LOCATION 0 (ZERO) AND INVOKE THE SAME ROUTINE AS "RESET" INPUT TO 8080.
THIS COULD RE-INITIALIZE THE SYSTEM BASED ON THE ROUTINE INVOKED.
(A CAUTION TO SYSTEM PROGRAMMERS.)

Silicon Gate MOS 8257

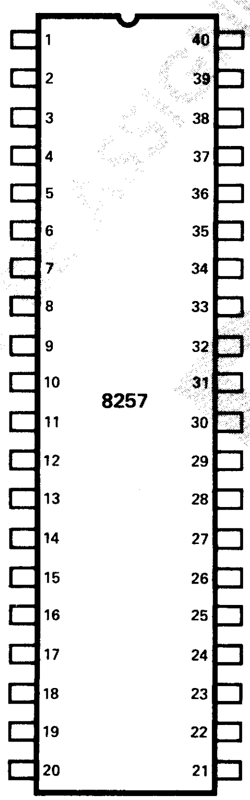
PROGRAMMABLE DMA CONTROLLER

- **Four Channel DMA Controller**
- **Priority DMA Request Logic**
- **Channel Inhibit Logic**
- **Terminal and Modulo 256/128 Outputs**
- **Auto Load Mode**
- **Single TTL Clock (ϕ2/TTL)**
- **Single +5V Supply**
- **Expandable**
- **40 Pin Dual-in-Line Package**

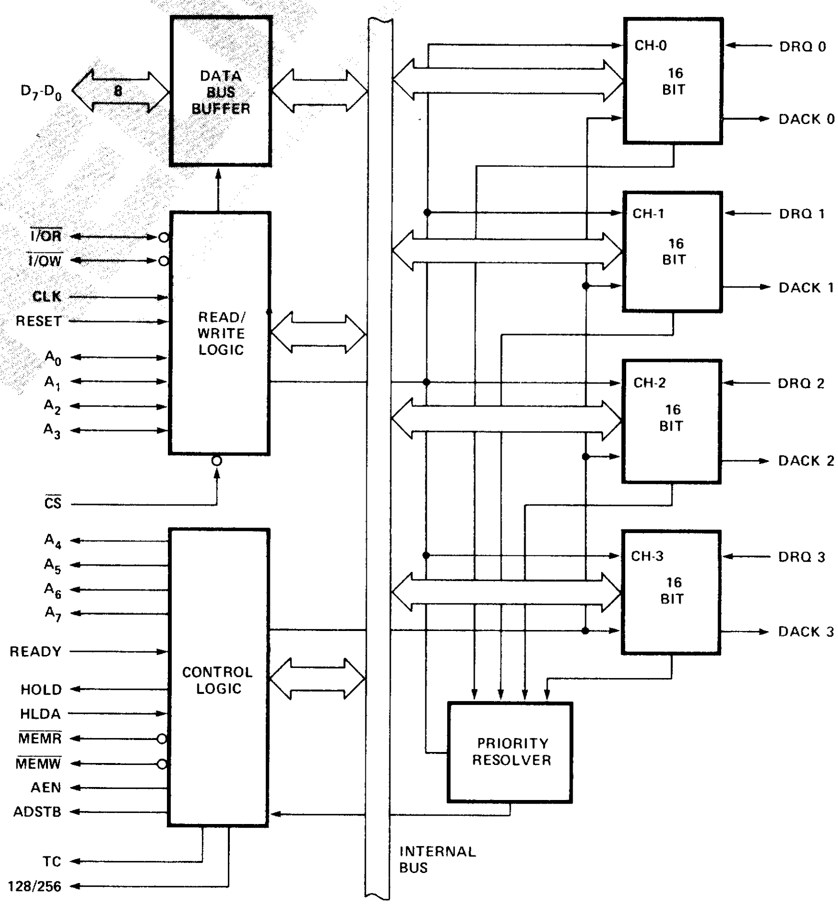
The 8257 is a Direct Memory Access (DMA) Chip which has four channels for use in 8080 microcomputer systems. Its primary function is to generate, upon a peripheral request, a sequential memory address which will allow the peripheral to access or deposit data directly from or to memory. It uses the Hold feature of the 8080 to acquire the system bus. It also keeps count of the number of DMA cycles for each channel and notifies the peripheral when a programmable terminal count has been reached. Other features that it has are two mode priority logic to resolve the request among the four channels, programmable channel inhibit logic, an early write pulse option, a modulo 256/128 Mark output for sectored data transfers, an automatic load mode, a terminal count status register, and control signal timing generation during DMA cycles. There are three types of DMA cycles: Read DMA Cycle, Write DMA Cycle and Verify DMA Cycle.

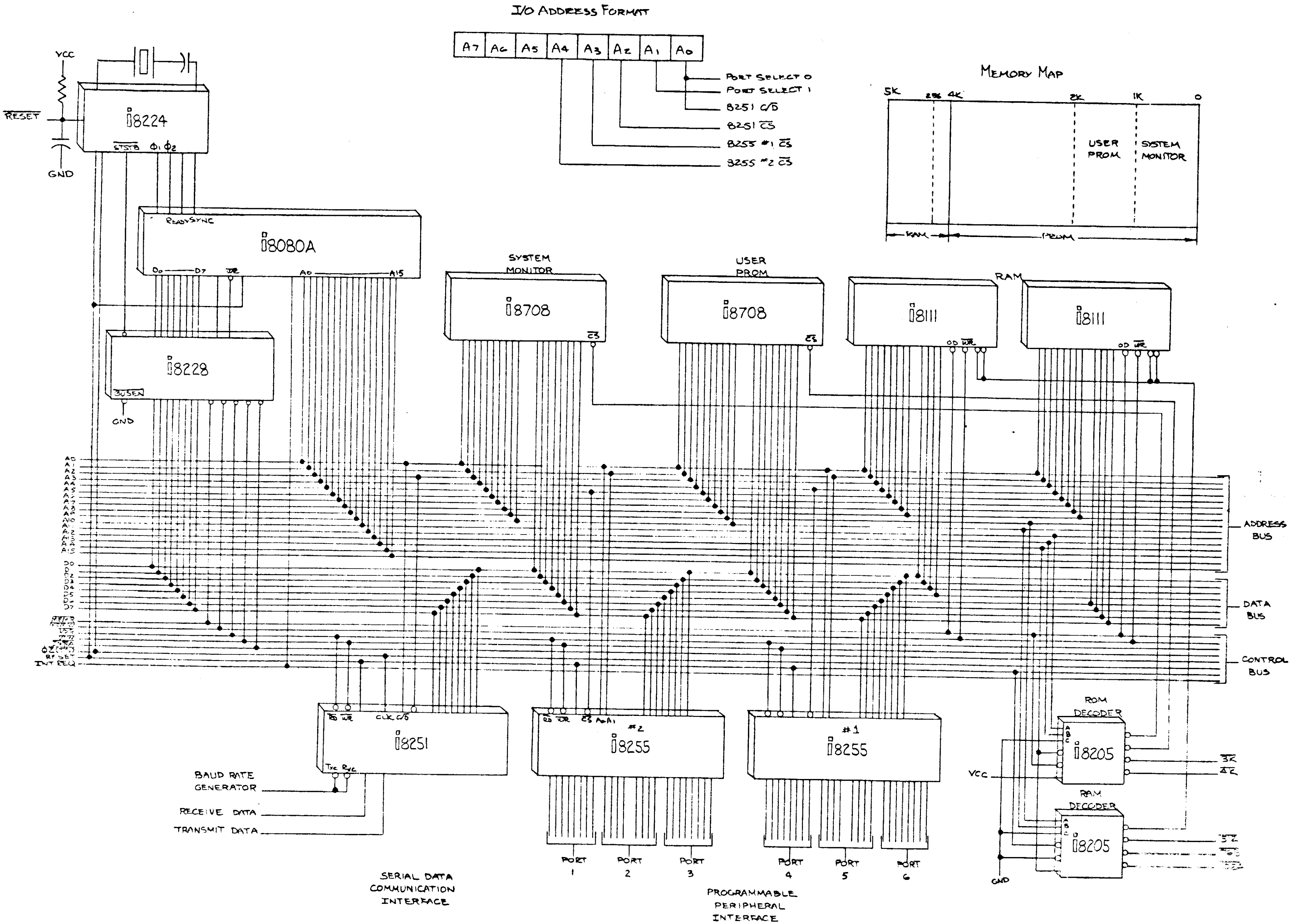
The 8257 is a 40-pin, N-channel MOS chip which uses a single +5V supply and the $\phi 2$ (TTL) clock of the 8080 system. It is designed to work in conjunction with a single 8212 8-bit, three-state latch chip. Multiple DMA chips can be used to expand the number of channels with the aid of the 8214 Priority Interrupt Chip.

PIN CONFIGURATION



BLOCK DIAGRAM




I/O ADDRESS FORMAT
A7 A6 A5 A4 A3 A2 A1 A0
PORT SELECT 0
PORT SELECT 1
8251 C/D
8251 CS
8255 #1 CS
8255 #2 CS
MEMORY MAP
5K
256
4K
2K
1K
0
USER PROM
SYSTEM MONITOR
RAM
PROM
VCC
RESET
GND
8224
STSB
Φ1 Φ2
READY SYNC
8080A
D0 — D7
A0 — A15
8228
BUSEN
SYSTEM MONITOR
8708
CS
USER PROM
8708
RAM
8111
OD WR
ADDRESS BUS
DATA BUS
CONTROL BUS
INT REQ
RESET
8251
CLK C/D
TxC RxC
BAUD RATE GENERATOR
RECEIVE DATA
TRANSMIT DATA
SERIAL DATA COMMUNICATION INTERFACE
#2
8255
#1
8255
PORT 1
PORT 2
PORT 3
PORT 4
PORT 5
PORT 6
PROGRAMMABLE PERIPHERAL INTERFACE
ROM DECODER
8205
RAM DECODER
8205
A B C

" NOTES "

PART VII

ICE 80

"NOTES"

ICE-80

IN-CIRCUIT EMULATION

FOR

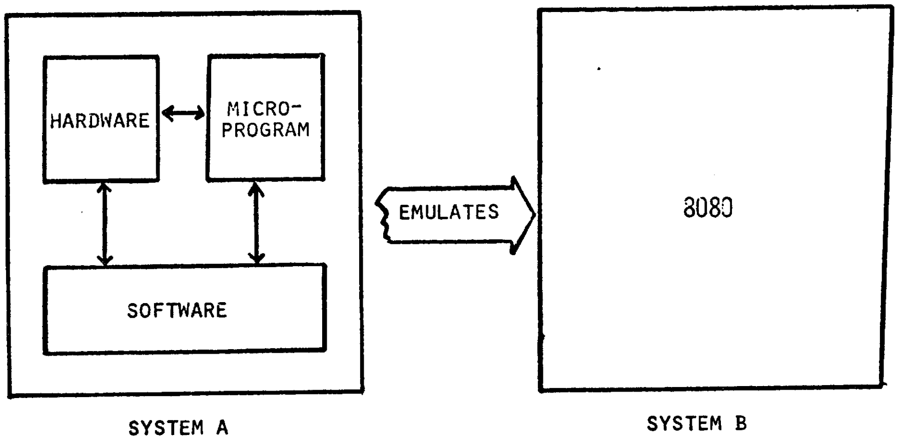
8080 BASED SYSTEM

ICE-80 FEATURES

- EXTEND MDS EXECUTION AND DEBUG CAPABILITIES INTO USER SYSTEM.
- REAL TIME EMULATION OF 8080 SYSTEM.
- SHARED MEMORY AND I/O CAPABILITY.
- DYNAMIC TRACING OF USER PROGRAM.
- SINGLE STEP OR MULTIPLE SINGLE STEP.
- DUAL HARDWARE BREAKPOINT CAPABILITY.
- SYMBOLIC DEBUG CAPABILITY.

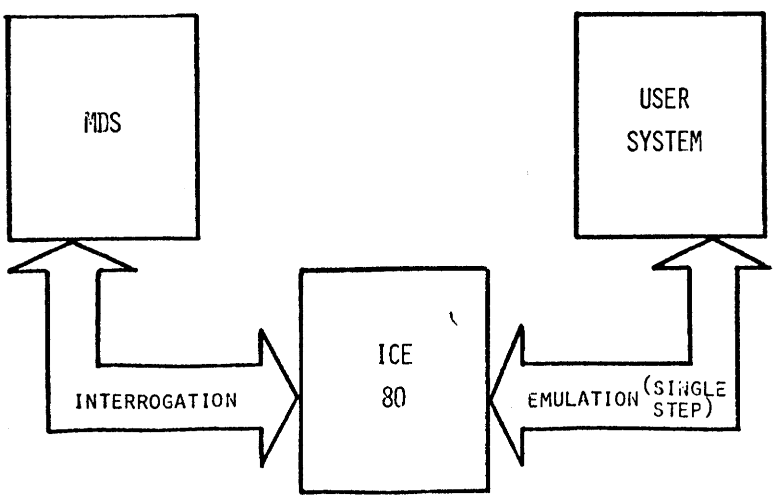
EMULATION

- DEFINITION - HARDWARE, MICROPROGRAMS, AND SOFTWARE ADDED TO ONE SYSTEM TO ALLOW ONE SYSTEM TO IMITATE ANOTHER!!



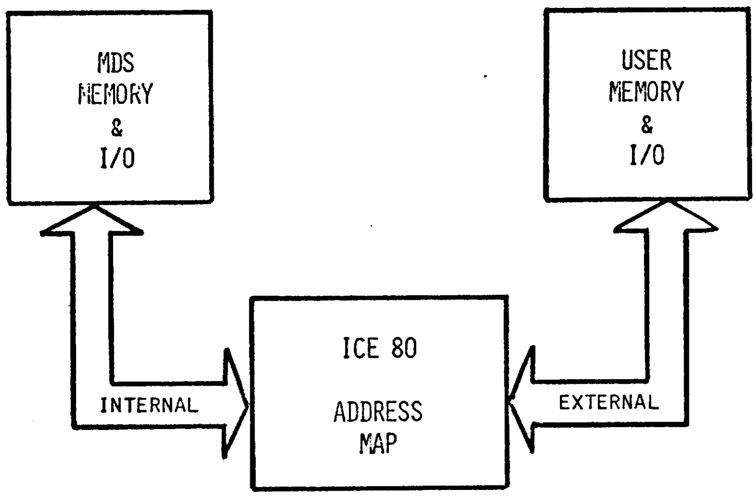
ICE 80 MODES

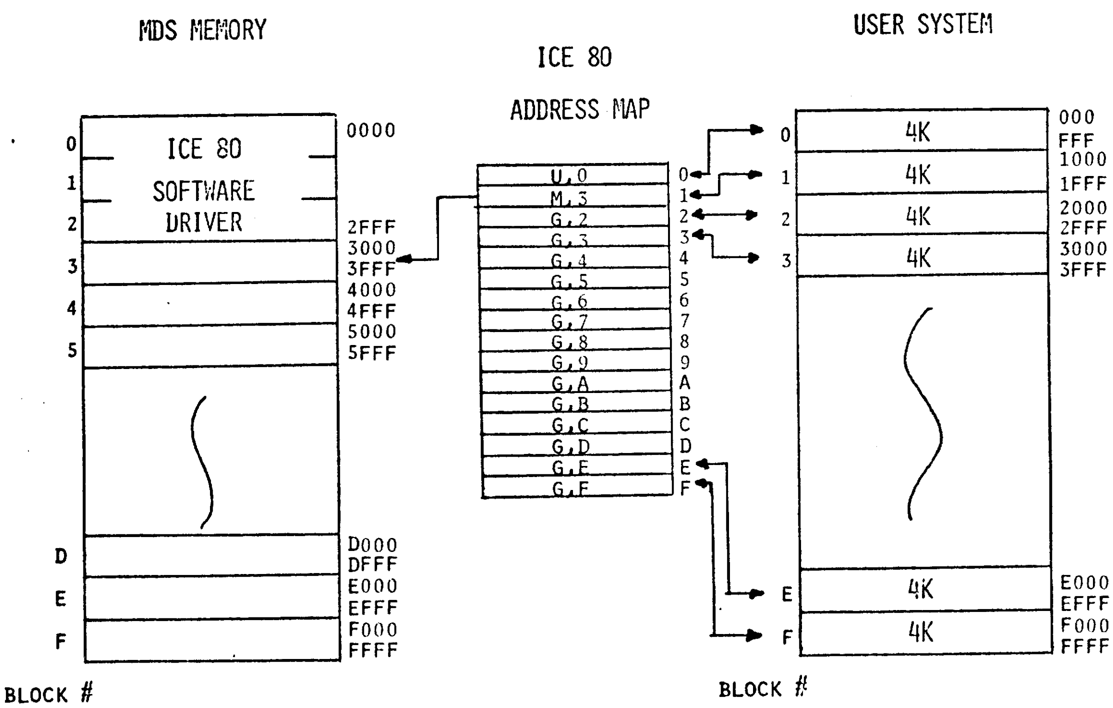
- EMULATION
- INTERROGATION
- SINGLE STEP



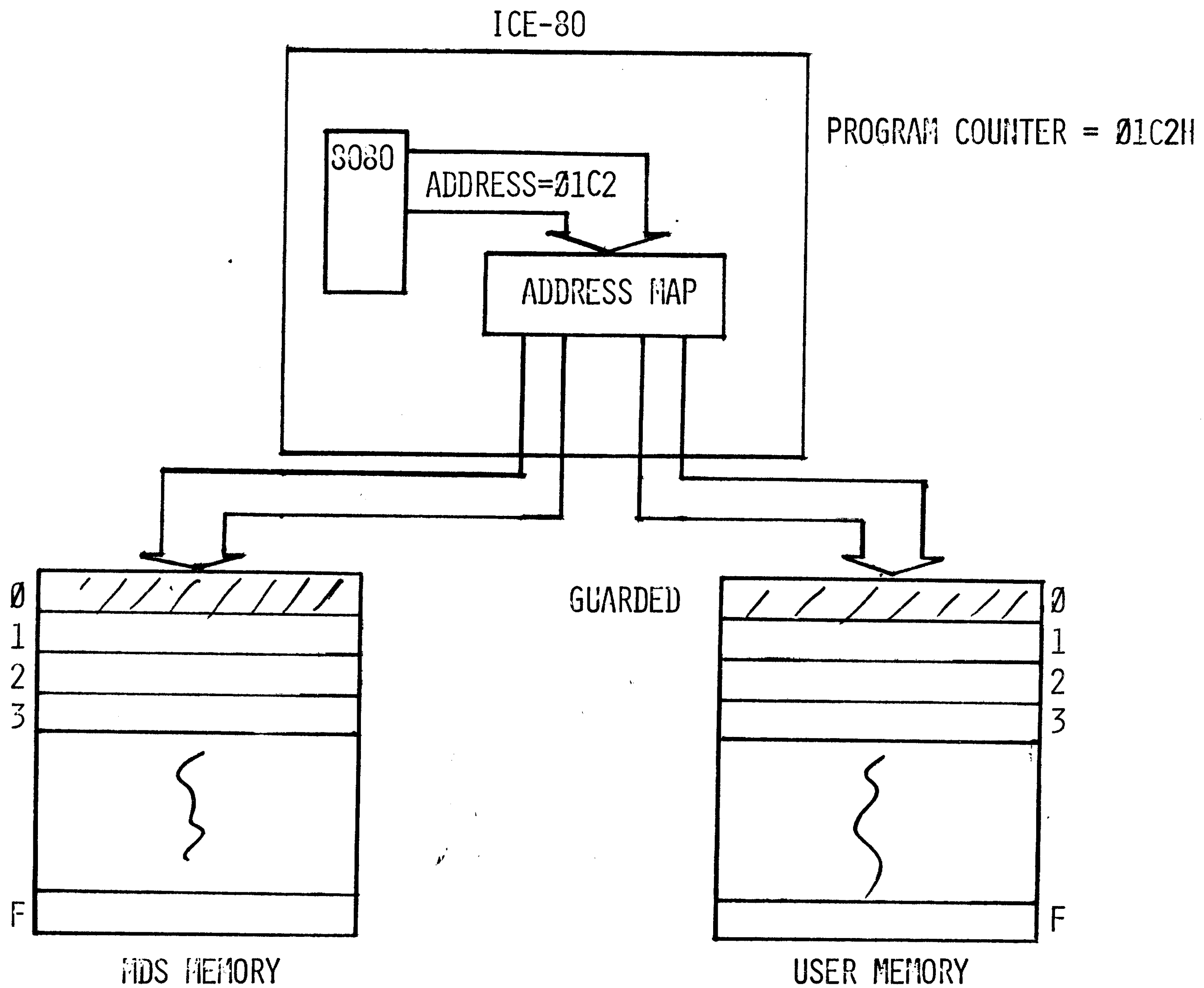
ICE 80 ADDRESS MAP

- MEMORY
- I/O





MEMORY MAP SAMPLE



GUARDED MEMORY

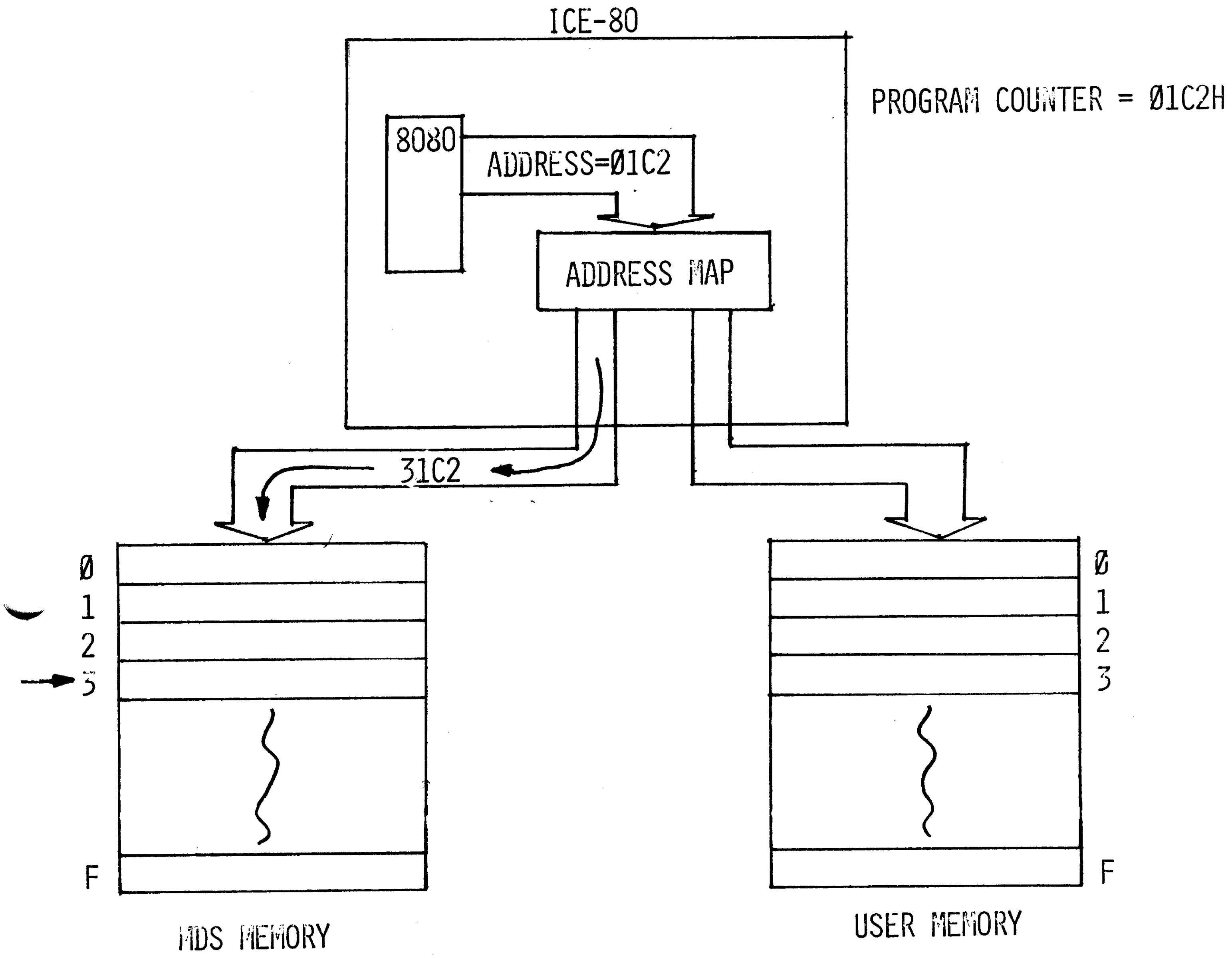
ICE-80 COMMAND

XFORM MEM Ø GUARDED

ICE-80 PATH

PC ADDRESS ——► "GUARDED"

MEMORY MAP SAMPLE



INTERNAL MEMORY

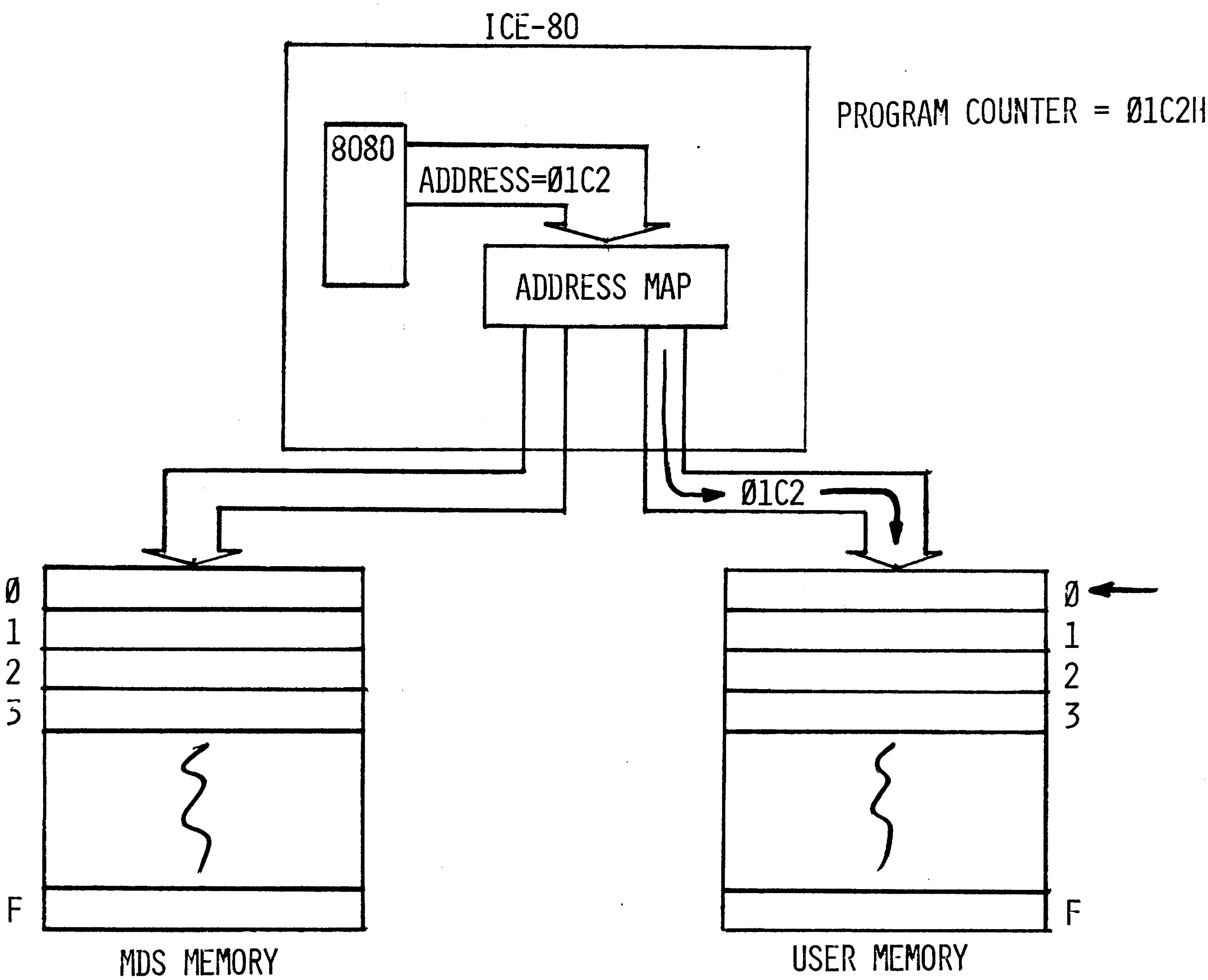
ICE-80 COMMAND

XFORM MEM Ø INTO 3

ICE-80 PATH

PC ADDRESS ——▶ MDS MEMORY

MEMORY MAP SAMPLE

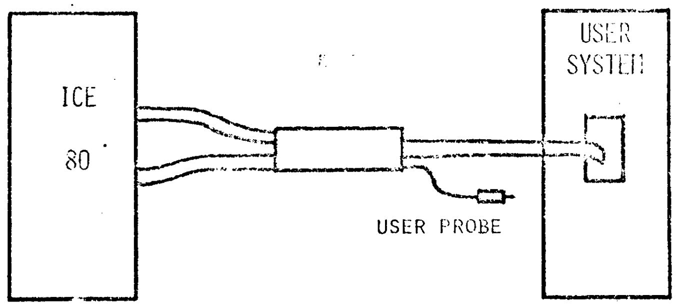


EXTERNAL MEMORY

ICE-80 COMMAND	ICE-80 PATH
XFORM MEM Ø UNGUARDED	PC ADDRESS ——► USER MEMORY

ICE 80 BREAKPOINTS

- DUAL HARDWARE BREAKPOINT REGISTERS
- USER DEFINED SIGNAL BREAK



BREAK ON:

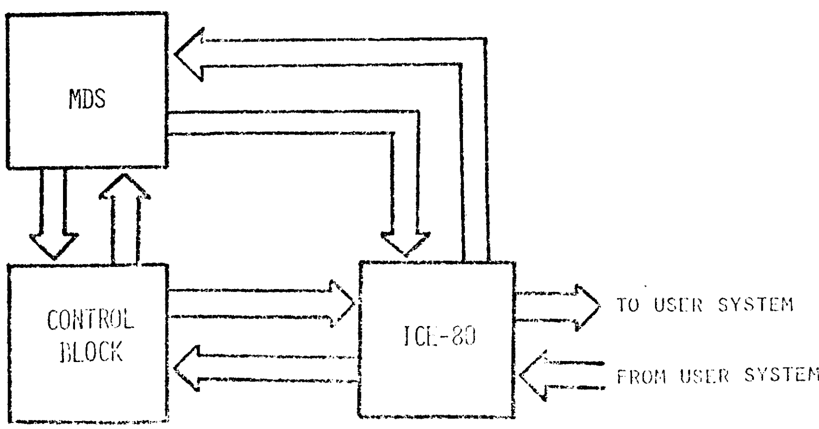
ADDRESS AND

- MEMORY READ
- MEMORY WRITE
- I/O READ
- I/O WRITE
- STACK READ
- STACK WRITE
- M1 FETCH

OR

USER DEFINED SIGNAL=

- FLIP-FLOP
- REGISTER
- COUNTER
- ONE SHOT



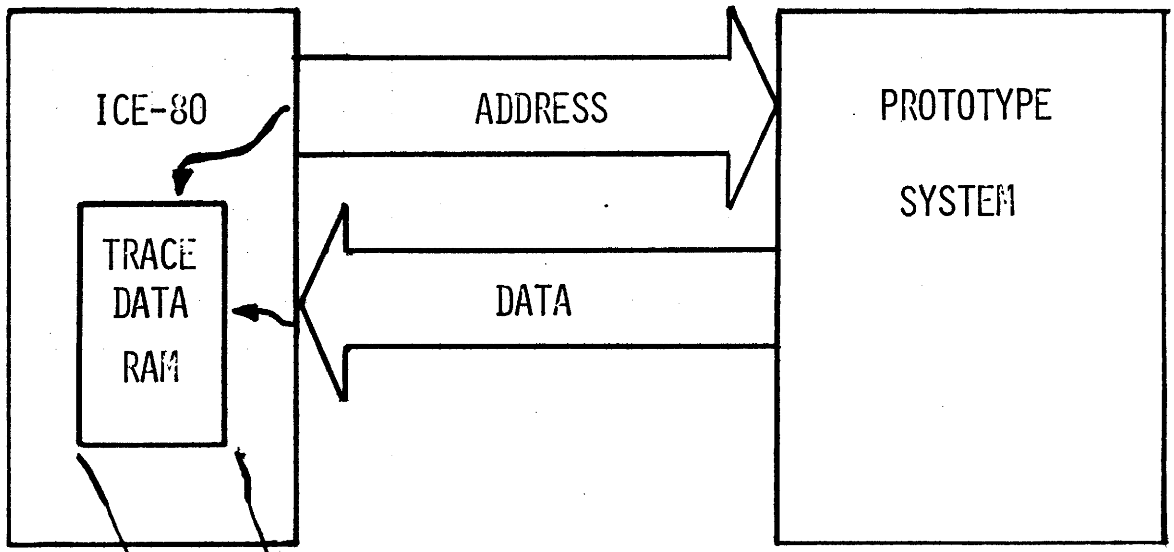
INFORMATION PATHS WITH ICE-80

CONTROL BLOCK SUMMARY

ADDRESS (BASE +)	CONTENTS	COMMENTS
0	Break register	Cause of emulation break
1	Break Status Register	8080 pins at break
2 3 4	Timer, Low Timer, Mid Timer, High	Interval Timer, 20 bits
5 6 7 8 9 A B C D E F 10 11	8080 PC, Low 8080 PC, High 8080 Reg. C 8080 Reg. B 8080 Reg. E 8080 Reg. D 8080 Reg. L 8080 Reg. H 8080 Flags 8080 Reg. A 8080 SP, Low 8080 SP, High 8080 Int. Enable	8080 machine state
12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F	Comp 2 Add., Low Comp 2 Add., High Comp 1 Add., Low Comp 1 Add., High Comp 2 Extension Comp 2 Condition Comp 1 Extension Comp 1 Condision User Condition Enable Comp 2 Enable Comp 2 Ext. Enable Comp 1 Enable Comp 1 Ext. Enable Timeout Enable	Comparator conditions

ADDRESS (BASE +)	CONTENTS	COMMENTS
20-DF	Snap Data	Snap Data Byte 1 = Status Byte 2 = Address, High Byte 3 = Address, Los Byte 4 = Data
E0-EF	Address Map	Address Map
F0 F1 F2	Address, Low Address, High Byte To Be Tranferred	Move Data
F5 F6	Test 1 (=A5H on LCB) Test 2 (=5AH on LCB)	Test Data
F7 F8 F9 FA FB FC	Failed Command Failure Type Failed Add., Low Failed Add., High Good Data Bad Data	Failure Data

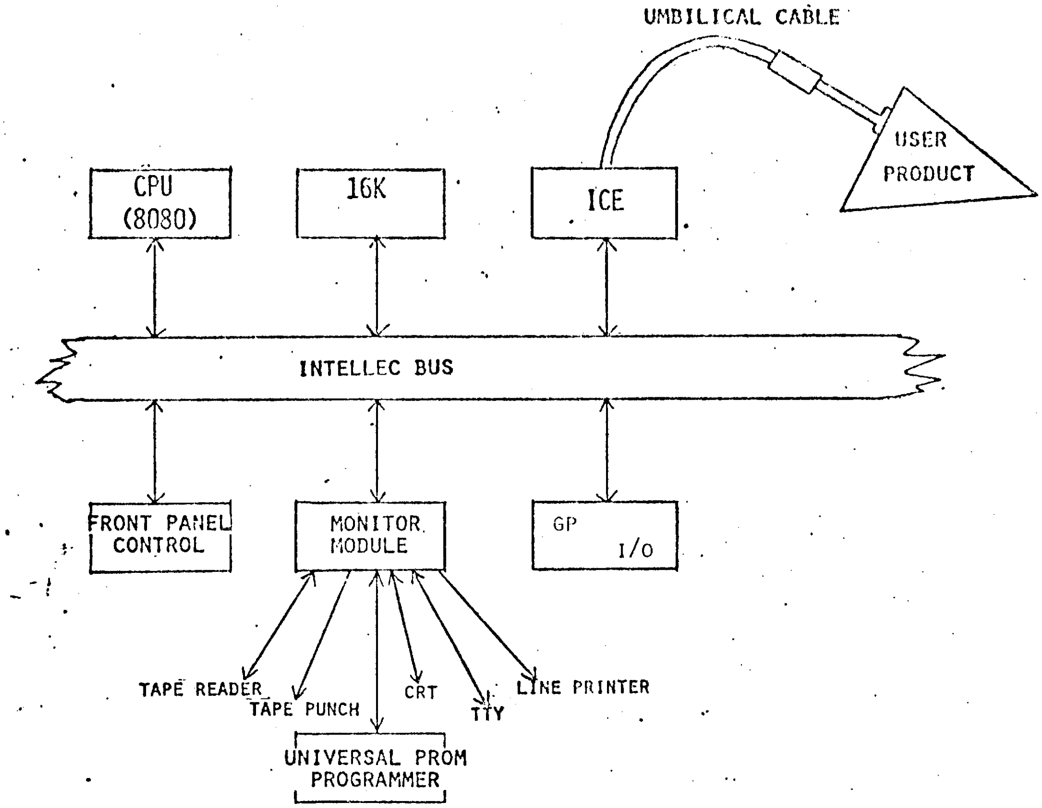
DYNAMIC TRACING

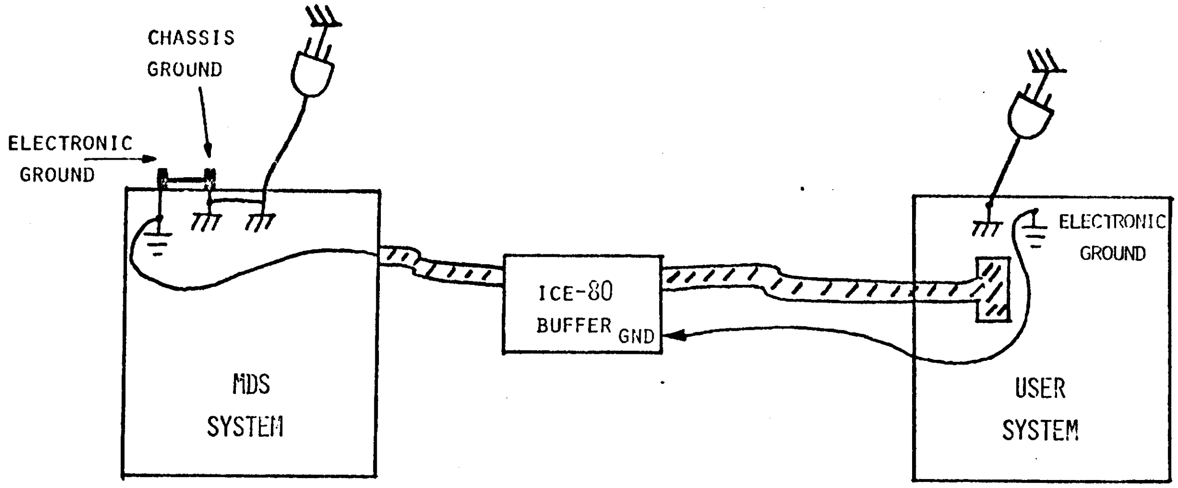


PROGRAM ADDRESS	INSTRUCTION
132Ø	LXI SP,132ØH
1323	LXI H, 13ØØH
1326	MVI M, Ø

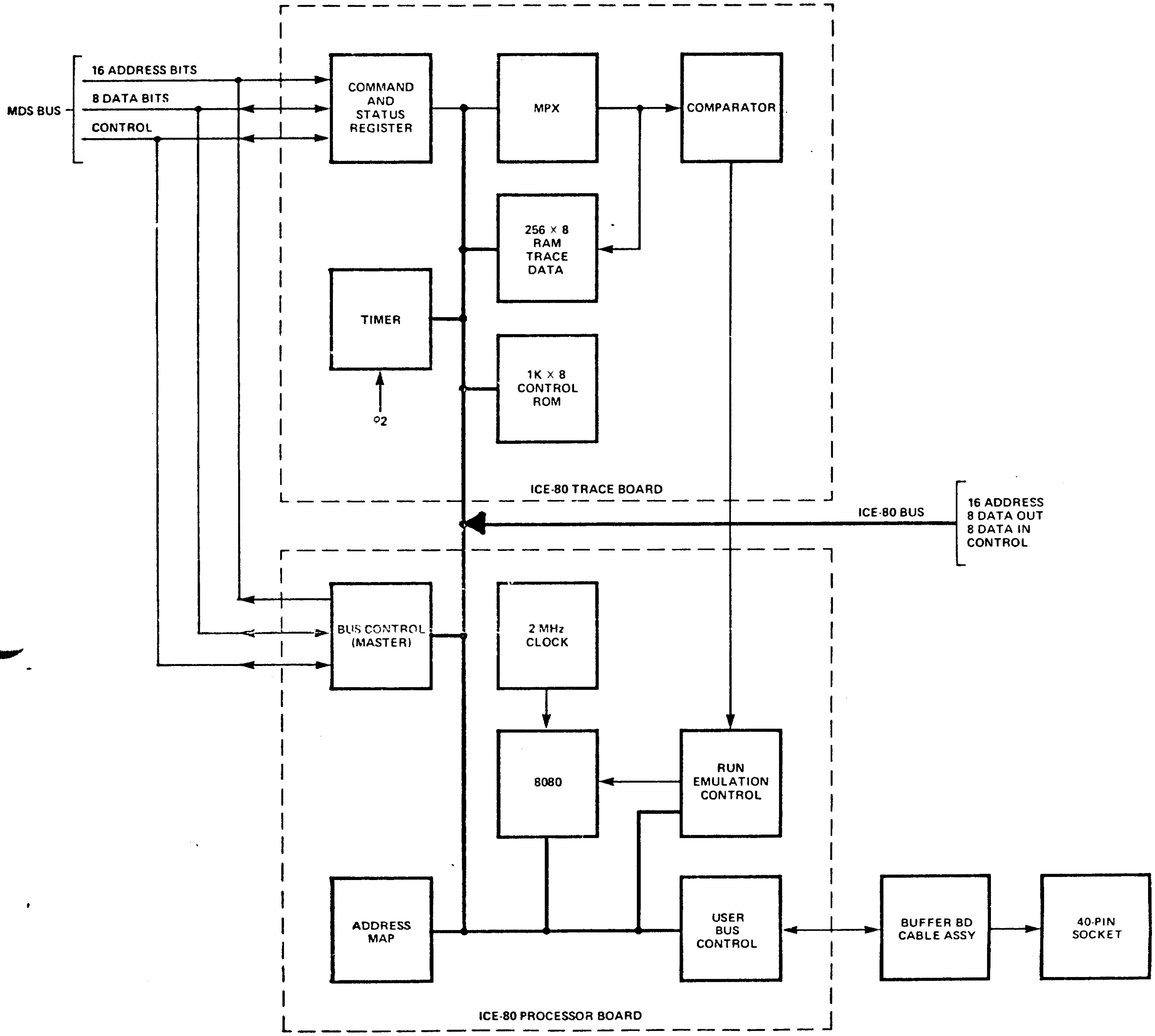
MACHINE CYCLE	STATUS	ADDRESS	DATA
M1	A2	132Ø	31
M2	82	1321	2Ø
M3	82	1322	13
M1	A2	1323	21
M2	82	1324	ØØ
M3	82	1325	13
M1	A2	1326	36
M2	82	1327	ØØ
M3	ØØ	13ØØ	ØØ

MDS RESOURCES





IDEAL SYSTEM GROUNDING



ICE-80 BLOCK DIAGRAM

COMMAND LANGUAGE

INTERFACES USER TO ICE-80 HARDWARE

ENGLISH-LIKE SENTENCES

REFER TO SYMBOLIC NAMES

TYPES OF COMMANDS

EMULATION

- RUN TO BREAKPOINT
- BREAK AFTER EACH INSTRUCTION (SINGLE STEP)
- ACTIONS TO PERFORM AT BREAK
- CONDITIONS FOR AUTOMATICALLY RESUMING

INTERROGATION

- OBTAIN INFORMATION ABOUT STATE OF USER'S SYSTEM
- MODIFY STATE

UTILITY

- LOAD OR SAVE USER'S PROGRAM
- DEFINE NEW SYMBOLS
- RETURN TO MONITOR

EMULATION COMMANDS

GO	FROM <START LOCATION> UNTIL <JACK READ>
	THEN <DUMP> CONTINUE <WHILE MEM JACK <3>
STEP	BY <1 INSTRUCTION> FROM <START LOCATION>
	THEN <DUMP> CONTINUE <FOREVER>
RANGE	<100H TO 300H> , <START LOCATION TO END LOCATION>
CONTINUE	
CALL	<ROUT 1>

INTERROGATION COMMANDS

DISPLAY	<MEMORY 100H TO 10FH>
	<REGISTER A>
	<ALL SYMBOLS>
	<ALL PINS>
BASE	<HEX>
	<DEC>
	<OCT>
CHANGE	<MEMORY 100H=C3H>
	<REGISTER A=25H>
	<FLAG CARRY=1>
XFORM	<MEMORY Ø TO 3 UNGUARDED>
	<MEMORY Ø INTO 3>
	<IO Ø TO 3 UNGUARDED>
SEARCH	<100H TO 300H> <MASK ØFH> FOR <ØFH>

UTILITY COMMANDS

LOAD

LOAD <FILE.HEX> (ISIS ICE-80 ONLY)

SAVE <1000H TO 2000H>

SAVE <FILE.HEX> <1000H TO 2000H> (ISIS ICE-80 ONLY)

MOVE <MEMORY 1000H TO 10FFH> INTO <MDSMEM 7000H>

FILL <MEMORY 1000H TO 10FFH> WITH <FFH>

TIMEOUT ENABLED

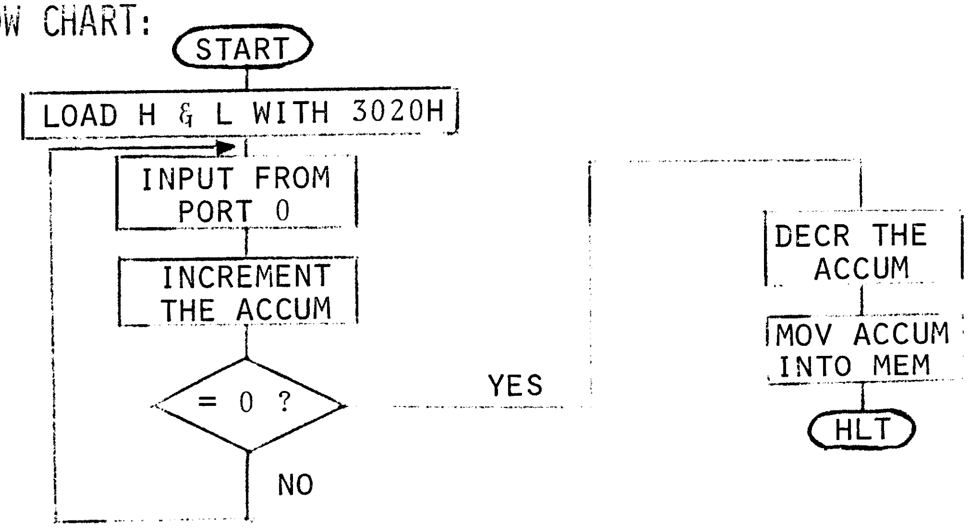
EXIT

WORKSESSION #1

OBJECTIVES: TO WRITE THE ASSEMBLY LANGUAGE PROGRAM DESCRIBED BELOW.

PROGRAM DESCRIPTION: THE PROGRAM IS TO READ A BYTE FROM PORT 0 AND CHECK IT FOR BEING EQUAL TO FFH. IF IT IS FFH THEN FFH IS TO BE STORED IN MEMORY ADDRESS 3020H AND THEN ENTER THE HALT STATE. IF IT IS NOT, THE PORT SHOULD BE READ AGAIN AND THE ABOVE PROCEDURE REPEATED. ORIGIN THE PROGRAM AT 3000H. DON'T FORGET THE END STATEMENT.

FLOW CHART:



SAMPLE CODING FORM:

LABEL	CODE	OPERAND	COMMENTS
			;
			;
			;
			;
			;
			;

WORKSESSION # 2

OBJECTIVES: TO WRITE THE SEQUENCE OF INSTRUCTIONS THAT WILL READ VALUES FROM INPUT PORT NUMBER 3 UNTIL A VALUE WITH BITS 5 AND 3 SET IS ENCOUNTERED. THEN HALT. NOTE: <u>ANY</u> NUMBER THAT HAS A TRUE B5 <u>AND</u> B3 SHOULD CAUSE A HALT. FOR EXAMPLE:

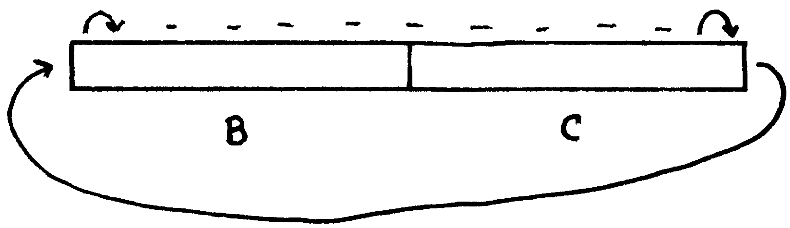
0	0	$\boxed{0}$	1	$\boxed{1}$	0	0	0	INPUT NEW VALUE
0	0	$\boxed{1}$	0	$\boxed{0}$	0	0	0	INPUT NEW VALUE
0	0	$\boxed{1}$	0	$\boxed{1}$	0	0	0	HALT
1	0	$\boxed{1}$	1	$\boxed{1}$	0	1	1	HALT

TO WRITE THE SEQUENCE OF INSTRUCTIONS THAT WILL READ A VALUE FROM INPUT PORT NUMBER 4 AND CHECK IF BIT 7 IS SET.

TO WRITE THE SEQUENCE OF INSTRUCTIONS THAT WILL TEST THE 16 BIT VALUE IN THE D AND E REGISTERS FOR ZERO.

WORKSESSION # 3

OBJECTIVE: To WRITE THE SEQUENCE OF INSTRUCTIONS THAT WILL ROTATE THE VALUE IN THE B AND C REGISTERS RIGHT ONE BIT POSITION.

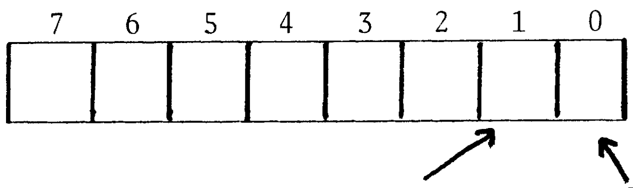


WORKSESSION #4

OBJECTIVE: To write a subroutine that outputs a character from the C register to the teletype that is interfaced to the MDS.

<u>Input Port Assignments</u>

F4	TTY data in (8 bits parallel)
F5	USART (8251) status



RECEIVE	TRANSMIT
1 = USART has data	1 = USART ready to accept data
Ø = no data	Ø = busy

<u>Output Port Assignment</u>

F4	TTY data out (8 bits parallel)

" NOTES "

LABORATORY PROJECT #1

Starting the System Monitor

- Cold Start Procedure

  1) Apply power by turning key to ON position.
  2) Set BOOT switch on.
  3) Press RESET switch.
  4) Type a "space" on device selected to be the system console:

     Monitor then prints on console.

     MDS MONITOR, Vx.x
  5) Set BOOT switch off. Monitor then prompts with a period (.) and is ready to accept a command:

System Monitor Commands

The Display and Substitute Memory commands are used to display and enter data into memory. The format of the Substitute command is:

S $\underbrace{\text{address}}_{\text{Starting Address}}$ sp xx- sp xx- zz CR (sp=space bar)

Starting Address — Current Contents Displayed — Type in New Contents

Using the Substitute command, small programs may be entered into memory in their machine code form and executed using the Go command (discussed below). Such a program is:

Memory Location	Assembly Language	Machine Code
100, 101	IN 0	DB,00
102, 103	OUT 1	D3,01
104, 105, 106	JMP 100H	C3,00,01

This program is entered as:

S100$_{sp}$xx-DB$_{sp}$xx-0$_{sp}$xx-D3$_{sp}$xx-1$_{sp}$xx-C3$_{sp}$xx-0$_{sp}$xx-1$_{sp}$ CR

To look at location 100 through 106 in more legible form use the Display Memory command.

D low address, high address CR

D100, 106 CR

To run the program enter the Program Execute command.

G address CR

G100 CR

Ports Ø and 1 are located on the blue I/O box attached to the MDS.

To return to the system monitor from this program;

Press the INTERRUPT Ø switch on the front panel -- the monitor prints an asterisk and the current program counter value and then prompts with a period (.).

* PC value

To abort a command or operation type:

CONTROL C

USING THE TEXT EDITOR

- Loading the editor into RAM memory.
  1) If not under control of the system monitor start the system monitor. See Cold Start Procedure on page 3-6.
  2) Place the tape into the tapereader and enter:

Command	Description
AR=P (CR)	Assigns paper tape input device to be the high speed reader
R0 (CR)	Reads the tape into memory
REMOVE TAPE FROM READER WHEN DONE!	
G20 (CR)	Go to location 20

(The console prints:
INTELLEC MDS TEXT EDITOR, VERSION x.x)
*

- Creating a Source Tape
  1) Use the I (INSERT) command and enter the following program:
     a) Use the TAB feature when entering the text for legibility. TAB = (CTRL) + (I)
     b) Remember, the editor provides a linefeed upon receipt of the carriage return-character.
     c) Do not press (ESC) (ESC) keys until all source lines are entered.

```
        *I
               ;PROGRAM NAME:  LABEX
                ORG           3000H
LOOP:           IN            0
                MOV           B,A        ;SAVE PORT 0
                IN            1
        'tab'   ADD   'tab'   B          ;PORT 0 + PORT 1
                OUT           0
                JMP           LOOP
                END           3000H  (CR)
(ESC)   (ESC)
```

2) Punch a copy of the workspace

N$$	Punch 6 inches blank tape, the teletype prints: START PUNCH, TYPE CHAR When the leader is punched turn the punch off and type any character
E$$	Punch the workspace text; the teletype prints: START PUNCH, TYPE CHAR
($= ESC)	After text and trailer is punched type any character (turn punch off first)
	NOTICE THAT THE SPACES HAVE BEEN DELETED WHEREVER THE TAB FEATURE WAS USED. THIS SAVES TIME AND TAPE.

- Editing the Source Tape

1. Put the source tape to be edited in the tape reader.
2. Use the A$$ command to read the tape into the editor.
3. Remove the source tape from the reader.
4. Move the buffer pointer to the beginning of the workspace with the B$$ commmand.
5. Use the T command to obtain a listing of the workspace (do not count the lines, just use a large number 20T$$)
6. Delete the instruction 'IN Ø'

FIN 'tab' Ø$$	Find the instruction
ØL$$	Move pointer to beginning of line
K$$	Delete the line

To verify:

-1L$$	Move pointer back 1 line
2T$$	Print two lines

7. Replace the IN Ø instruction

FMOV$$	Find the line to insert in front of
ØL$$	Move pointer to beginning of line

ILOOP: 'tab' IN 'tab' Ø CR

$$

To verify:

-2L$$	Move back two lines
3T$$	Print three lines

8. Change the B,A to C,A

FMOV 'tab' $$	Pointer left in front of the B
D$$	Delete one character (pointer is left in front of the comma)
IC$$	Put C in front of the comma

To verify:

ØL$$	Move pointer to beginning of line
T$$	Print one line

9. Change the C,A back to B,A

SC,A$B,A$$	Search for C,A and when found replace it with B,A

To verify:

ØL$$

T$$

10. Punch a copy of the workspace using the N and E commands.

N$$	Punch 6 inches blank tape; the teletype prints: START PUNCH, TYPE CHAR When the leader is punched turn the punch off and type any character
E$$	Punch the workspace text; the teletype prints: START PUNCH, TYPE CHAR After text and trailer is punched type any character (turn punch off first) NOTICE THAT THE SPACES HAVE BEEN DELETED WHEREVER THE TAB FEATURE WAS USED. THIS SAVES TIME AND TAPE.

USING THE MACRO ASSEMBLER

- Loading the Assembler into RAM memory.

  1. Turn the tape reader on.

Omit if previously assigned: AR=P CR — Assigns paper tape input device to be the reader. (if available)

  2. Place the assembler tape into the tape reader and enter the command:

     RØ CR — Reads the tape into memory

- Assembling the source program.

  1. Enter the command: G2Ø CR Console prints:

     8080 MDS MACRO ASSEMBLER, VERSION x.x

     P=

  2. Place the source tape in the reader and type a '1'

     The source tape is read in and the symbol table is created.

     When pass 1 is finished the console prints: P=

  3. Reset the source tape in the reader and type a '2'

     When pass 2 is finished the console prints: P=

  4. Reset the source tape in the reader, turn the punch on and type a '3'

     The hexadecimal object tape is punched. TURN THE PUNCH OFF.

  5. Press INTERRUPT Ø switch. Control is returned to the Monitor.

  6. Place the object tape into Tape Reader and enter command:

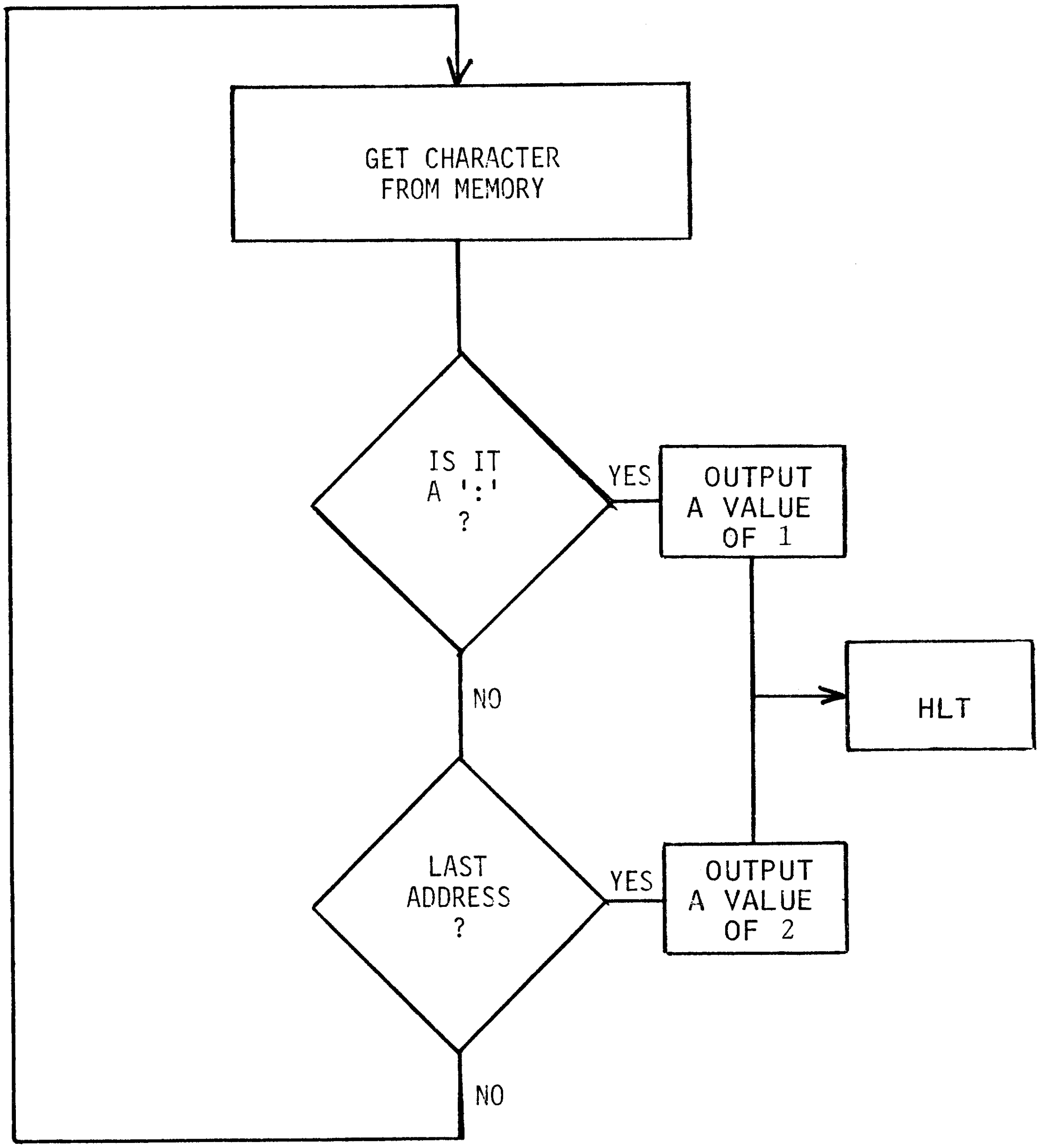
     RØ CR

  7. Enter the command G3000 CR

The Program is Executing!!

LABORATORY PROJECT #2

Objective: To write a program that searches a string of characters in memory locations 3100H through 3109H for a colon. If a colon is found, output a 1 to port number Ø. If a colon is not found, output a 2 to port number Ø. In either case, enter the halt state upon completion. Origin the program at 3000H. Use an END statement argument (in the opened field) of 3000H.



LABORATORY PROJECT #3

OBJECTIVE: Alter the program written in Laboratory Program Project #1 such that actual messages are output on the TTY instead of lighting lites on port 0. Sample messages are:

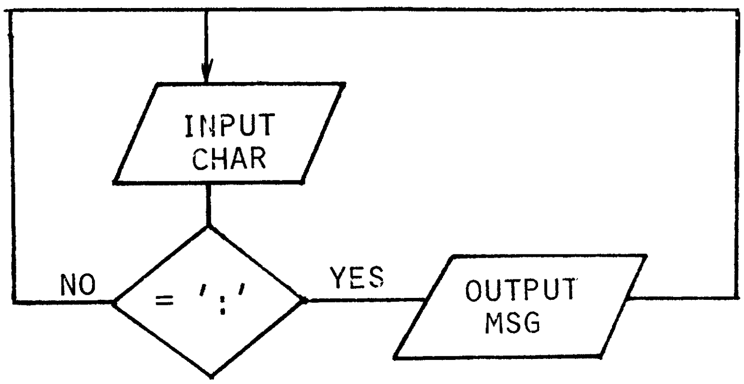
I FOUND IT!

or

I GIVE UP!

LABORATORY PROJECT #4

OBJECTIVE: Alter the program above such that instead of searching memory for a colon, the program checks what is input from the TTY. If a colon is input, a message is typed out. If any other character is input, no action is taken and the program waits for the next input character.



After the messages is printed, return to checking the next input character.

LABORATORY PROJECT #5

Starting the Systems Implementation Supervisor

- Cold Start Procedure

  1. Apply power by turning key to ON position.
  2. Turn on Diskette.
  3. Insert a "system" diskette in the drive 0 opening with the read/write access slot first and the index hole, which is slightly off center, toward the bottom of the drive. Close the door.
  4. Set BOOT switch on.
  5. Press RESET switch (Interrupt light 2 should go on.)
  6. Type a "space" on device selected to be the system console (Interrupt light 2 should go off.)
  7. Set BOOT switch off: Supervisor then prints on console:

     ISIS, Vx.x

     and prompts with a dash (-). It is now ready to accept a command.

Editing Command Inputs

RUBOUT - This key operates the same as on the text editor, i.e., it deletes the character while, repeat-printing it.

CONT-R - This retypes the current command string. This is valuable when many rubouts were used.

CONT-X - This cancels the current line and types a number sign (#) and a carriage return.

Using the Text Editor

- Procedures for calling up the text editor vary according to whether:

  a. a source file is to be created,
  b. a source tape exists and is to be read and changed into a disc file,
  c. a source file exists now on the disc and is to be edited.

Each situation is described below.

A. Create a new source file called LABEX

1. Enter the command:

   EDIT LABEX (CR)

2. The console prints:

   ISIS TEXT EDITOR, Vx.x
   NEW FILE

   and prompts with the editor asterisk (*). It is now ready to receive the same text editor commands as covered on the paper tape operating system except for the N$$ command which is no longer needed.

3. Abort and return back to ISIS by using the Quit command. Enter:

   Q$$

   and an ISIS prompt will result.

B. A source tape exists

1. Load the source tape onto the tape reader.
2. Enter the command:

   EDIT :HR: TO LABEX (CR)

   NOTE: Other I/O file designations are:

   :TP:TTY PUNCH OUTPUT
   :TR:TTY TAPE READER INPUT
   :HP:HIGH SPEED PUNCH OUTPUT
   :LP:LINE PTR OUTPUT
   :VI:CRT INPUT
   :VO:CRT OUTPUT
   :TO:TTY LIST OUTPUT

3. The console prints:

   ISIS TEXT EDITOR, Vx.x

   and prompts with the editor asterisk (*).
4. Enter:

   A$$

   The tape will read in and a prompt is printed. The source program is now in the workspace and is ready for a normal edit.
5. Make some kind of an edit then output the file onto the disc by entering:

   E$$
6. Verify the file has been added by entering

   DIR CR

   The directory prints out with the new filename added to it.
7. Verify it is a correct file by entering:

   COPY LABEX TO :CO: CR

   The console prints out the hexidecimal file in the same format as the source tape which was read in.

C. Source file already exists on the disk and needs to be edited.

1. Enter the command:

   EDIT LABEX CR

   The console prints out:

   ISIS TEXT EDITOR, Vx.x

   and an (*).
2. Enter:

   A$$

   This causes the file to be read from the disk and be put in the editor workspace.
3. Do an edit of some kind.
4. Enter:

   E$$

   to output the file.
5. Enter:

   DIR CR

   The console prints out the directory. Notice there is now 2 files, LABEX and LABEX.BAK. The latter is a back up version that does not have the last edit.

6. Files can be copied and/or deleted. Enter:

    COPY LABEX.BAK TO BACKUP CR

   Examine the directory (a fast version) by entering:

    DIR$F CR where $ = SHIFT 4

   The file BACKUP is now a copy of LABEX.BAK so we can now delete LABEX.BAK. Enter:

    DELETE LABEX.BAK CR

   The console responds with:

    LABEX.BAK,DELETED

Using the Macro Assembler

- Calling up the assembler.

1. Enter:

   ASM8Ø LABEX (CR)

   (Note: The source program must be a disk file)

2. The console prints out:

   ISIS 8080 MACRO ASSEMBLER, Vx.x

   ASSEMBLY COMPLETE, NO PROGRAM ERRORS

   and a prompt (-) if there are no errors. Otherwise it prints the number of errors.

3. Enter:

   DIR (CR)

   A hexadecimal object file and a list file has been created corresponding to the object tape and listing as done on the paper tape system.

4. Change the name of the hexfile by entering

   RENAME LABEX.HEX TO LAX (CR)

   Examine the directory to see the results.

5. To print out the listing, enter:

   COPY LABEX.LST TO :CO: (CR)

6. To execute the object code it must be changed from hex to binary. Enter:

   HEXBIN LAX TO LAX.BIN (CR)

7. Enter:

   DIR (CR)

   Notice the new binary file you just created.

8. To execute the program now enter:

   LAX.BIN (CR)

   The program is now executing! Now put ISIS back in control by hitting Interrupt 1.

9. Other ISIS commands not covered in this section are:

   ATTRIB - used to assign WRITE PROTECT and INVISIBILITY attributes to any file.

   FORMAT - used to initialize a new diskette.

   Consult chapter 4 in the DOS Operator's Manual for details.

LABORATORY PROJECT #6

1. If the MDS was just powered up, perform the bootstrap procedure as discussed in the preceding sections.

2. Paper Tape Operating System

   - Enter - AR = P (CR)
   - Load the object tape on the tape reader
   - Enter - RØ (CR)

   Proceed to debug techniques below.

   Disk Operating System

   - Enter - DEBUG (FILENAME) (CR)

   The console will print an asterisk followed by the starting address (as specified by the 'END' statement argument" then a monitor prompt.

   Debug Techniques

   1. Using the program listing, choose one or two breakpoint instructions and locate the addresses where the breakpoint instructions are located. REMEMBER TO SELECT ONLY THE 1ST BYTE OF MULTIPLE BYTE INSTRUCTIONS.
   2. Enter the following commands for execution up to the breakpoints:

      If no breakpoint desired-
      - G(ST.ADDR) (CR)

      If one breakpoint desired-
      - G(ST.ADDR),(BRKPT ADDR) (CR)

      If two breakpoints desired-
      - G(ST.ADDR),(BRKPT ADDR),(BRKPT ADDR) (CR)
   3. Execution should now begin. If either breakpoint is reached, the system will print the breakpoint address and a monitor prompt.
   4. Display all registers by entering:

      X (CR)

5. Change the contents of a register by entering:

    • X (register)'space'

   Contents of register will be printed. Index new contents and hit space. The contents of next register in sequence is printed (as in the monitor 'Substitute' command). Again, new contents may be entered. Return to a 'prompt' by CR. Restore original contents to all registers.

6. Using the 'Substitute' command, display the contents of the top of stack by entering the stack pointer displayed in no. 4 above. (S = SP)

    • S (STACK POINTER),-------- CR

   Also display the contents of 'M' (as pointed at by the H and L registers:

    • S (H and L CONTENTS),-------- CR

7. Continue execution by entering 'G' and new breakpoints:

    • G,BRKPT1,BRKPT2 CR

8. Using above techniques, debug your program!!

Helpful Hints

9. When errors are located, use the substitue command to change the wrong machine code in memory.

10. If an instruction needs to be completely wiped out, use the NOP command.

11. If a group of instructions need to be inserted, use RST7:

PROGRAM WITH A BUG

LOC	CODE
3000	3E
3001	Ø8
3002	D3
3003	F9
3004	DB
3005	F9

73
23
FE
04

This code needs to be inserted here.

CORRECTED PROGRAM

LOC	CODE	LOC	CODE
3000	3E	38	D3
3001	ø8	39	F9
3002	FF(RST7)	3A	73
3003	øø	3B	23
3004	DB	3C	FE
3005	F9	BD	04
		3E	C9(RET)

12. Re-execute after a bug has been fixed to make sure it is correct. After <u>all</u> bugs have been found and corrected, edit the corresponding errors from the <u>source</u> program.

LABORATORY PROJECT #7

UNIVERSAL PROM PROGRAMMER

Using the PROM Programmer.

1. Set POWER switch on to ON position.
2. Erase PROMs using Ultraviolet light.
3. Start System Monitor.
4. Insert PROM (24-pin) into Socket 2 and lock.
5. Place the hexadecimal object tape from laboratory project #2 into the reader and enter the following commands:

   AR=P CR
   R0 CR

6. When the tape has been read enter the command:

   PTX3000,30FF,0 CR

   TRUE PROGRAM — SOCKET #2 — STARTING MEMORY ADDRESS — ENDING MEMORY ADDRESS — STARTING PROM ADDRESS

7. Remove PROM following a prompt from the Monitor and turn PROM programmer power off.
8. Optionally, if a PROM board is available, turn MDS power OFF and insert the PROM into the PROM Module at position A1.
9. Ensuring PROM Module is selected for Address 3ØØØ - 3FFF replace module into the MDS.
10. Restart the Monitor and enter G3ØØØ CR — the program is now running.

NOTE: The special construction of the 8704, 8708 family of proms requires a slightly different programming algorithm from the one normally used to program 1702As. Therefore, a special software interface is needed. The program is read in and starts in location 20 just like the editor and assembler. Once read in, the commands are the same as above.

ICE 80 EXERCISES

ICE LAB EXERCISES

Introduction: Prior to Power Up of the Intellec MDS, ensure that ICE-80 cables are disconnected and the clock select jumpers are configured for internal clock.

It is our intention to begin the lab exercises with basic command sequences to familiarize you with the ICE-80 command language before an actual debug session.

EXERCISES	REFERENCE & NOTES
I. Power up and Initialization	
STEP	
1) Power ON Intellec MDS and Console Device.	REFERENCE: Section 2, pages 10-16 of the ICE-80 Operator's Manual. MCS 822-1075
2) Power Diskette System and load System Diskette into Drive Ø position.	NOTE: The Interrupt 2 indicator should be illuminated on the Intellec MDS.
3) Press Boot and Reset the Intellec MDS.	
4) Press "Space Bar" on System Console and Release Boot Switch.	NOTE: The MDS responds with the message: "ISIS Ver x.x" and prompts with (-) hyphen.
5) Enter, via console: "ICE-80" followed by carriage return.	NOTE: When ICE-80 is initialized properly the console prompts with: "ISIS ICE 80 Ver x.x" NOTE: ICE-80 now goes through an automatic system checkout to verify presence of clock and interface connection. If test is successful the following prompt is issued: "**" (double asterisk)

EXERCISES	REFERENCE & NOTES

II. Interrogation

STEP

1) In order to familiarize you with the commands of the ICE-80 Software Driver, the following exercises should be performed.

REFERENCE: Section 3, page 43 of ICE-80 Operator's Manual.

2) Enter, via console:

"XFORM MEMORY Ø to FH"

followed by carriage return.

NOTE: The console responds with:

ØH=G,ØH
1H=G,1H
2H=G,2H
3H=G,3H
4H=G,4H
5H=G,5H
6H=G,6H
7H=G,7H
8H=G,8H
9H=G,9H
AH=G,AH
BH=G,BH
CH=G,CH
DH=G,DH
EH=G,EH
FH=G,FH

3) Enter, via console:

"XFORM IO Ø to FH"

followed by carriage return.

NOTE: The console responds with same message as above.

4) Enter:

"BASE HEX"

followed by carriage return.

REFERENCE: BASE and DISPLAY commands in Section 3, page 37 & 38 of ICE-80 Operator's Manual.

NOTE: The command BASE HEX initializes ICE-80 to display values in Base 16.

5) Enter:

"DISPLAY ALL REG"

followed by carriage return.

NOTE: This command displays all 8080 registers in the following manner:

B= C= D= E= H= L= F= A= P= *= S=

<u>*</u> references last instruction executed

<u>F</u> references Flag Byte

EXERCISES	REFERENCE & NOTES
II. Interrogation (cont.)	
STEP	
6) Enter: "DISPLAY ALL PINS" followed by carriage return.	NOTE: Pins are displayed: HLT= HLD= INT= RDY= RST=
7) Enter: "XFORM MEMORY Ø INTO 6" followed by carriage return.	NOTE: This command assigns user memory block Ø to by physically located in MDS Memory block 6.
8) Enter: "CHANGE MEMORY 100H= 3EH, AAH, C3H, 00H, Ø1H" followed by carriage return.	NOTE: This command has stored in memory location Ø thru 4 the short routine: Ø,1 MVI A,ØAAH 2,3,4 JMP Ø1ØØH
9) Enter: "SEARCH MEMORY Ø1ØØH to Ø1ØFH MASK ØFFH for AAH" followed by carriage return.	NOTE: This command searches User memory 1ØØH to 1ØFH for value AAH and will print location of each occurance.
10) We will now initialize ICE-80 to operate with internal MDS Memory and I/Ø. This is in preparation for executing the program shown in Figure 1.	
11) The program is origined at 3ØØØH and utilizes some of the Intellec MDS Monitor I/O. In order to set up User memory for storage of the program enter: "XFORM MEMORY 3 INTO 7" followed by carriage return.	NOTE: This command assigns User memory block 3 to be located in MDS Memory block 7.
12) Because Monitor I/O is being used we must now enter: "XFORM MEMORY FH INTO FH" followed by carriage return.	NOTE: The Monitor is located in MDS Memory at F8ØØH to FFFFH. This command allows program to access the FH block of memory.

EXERCISES	REFERENCE & NOTES

II. Interrogation (cont.)

STEP

13) Enter:

"XFORM MEMORY Ø INTO 6"

followed by carriage return.

NOTE: These commands permit the Monitor to access the dedicated RAM associated with normal MDS operation.

14) Enter:

"MOVE MDS MEM Ø TO FH INTO MEM Ø"

followed by carriage return.

NOTE: We have essentially moved locations Ø to FH of MDS Memory into MDS Memory 6000H to 600FH.

15) Enter:

"XFORM IO FH INTO FH"

followed by carriage return.

NOTE: This command allows our exercise program to access MDS IO at block FH.

16) Mount diskette:

"TNGØØØ. 2Ø5"

in Drive Ø and closed drive door.

17) Enter:

"LOAD WAYNE3. DEM"

followed by carriage return.

NOTE: This command loads the Hex file WAYNE3 DEM into User memory under direction of address map we have set up.

REFERENCE: Figure 1 for program listing.

18) Enter:

"DISPLAY ALL SYMBOL"

followed by carriage return.

NOTE: This command displays symbols loaded with Hex file.

BLOCKØØ1	ØØØØH
CHECK	303EH
CI	FCA2H
DBYTE	FE51H
LCRLF	FEB6H
LINE	3028H
LØ	FDA1H
LOOPC	ØØØ2H
NDIGIT	ØØØ5H
NEXTC	302DH
OUTCT	3000H

19) Enter:

"DISPLAY MEMORY 3Ø2ØH TO 3Ø2FH"

followed by carriage return.

NOTE: This command displays contents of memory in this format:

3Ø2ØH= 31H 20H 30H 21H - - - - -

EXERCISES	REFERENCE & NOTES
III. Emulation	
STEP	
1) In order for the program to execute, the CRT or TTY console will be used. Enter: "GO FROM 3Ø2ØH" followed by carriage return.	NOTE: The console will print: "EMULATION BEGUN" and then print: Ø1 Ø2 Ø3 Ø4 Ø5
2) With each depression of Space Key, the program continues by incrementing a counter and listing values. Enter successive: "SPACE KEY TO VERIFY!"	
3) To exit from emulation, press the Interrupt 4 switch on the front panel of the MDS.	NOTE: The console then prints: "PROCESSING ABORTED" EMULATION TERMINATED AT XXXX"
4) Enter: "DISPLAY ALL REGISTERS: followed by carriage return.	NOTE: The "*=" indicates last instruction executed when Interrupt entered. The "P=" indicates next instruction to be fetched.
5) Enter: "DISPLAY CYCLES 10" followed by carriage return.	NOTE: This command displays the last 10 machine cycles completed up to the interrupt. Format is:

Status	Address	Data	
--	----	--	2 cycles
--	----	--	
--	----	--	
--	----	--	
--	----	--	5 cycles
--	----	--	
--	----	--	
--	----	--	
--	----	--	3 cycles
--	----	--	
			last instruction

EXERCISES	REFERENCE & NOTES
III. Emulation (cont.)	
STEP	
6) We will now set a breakpoint condition for this program. The "OUTCT" is the memory location that determines the value to be output. So we will set a breakpoint on the read of that location and continue emulation while OUTCT≠ FH.	
7) Enter: "GO from 3020H until OUTCT READ THEN DUMP continue while Memory OUTCT<>FH." followed by carriage return.	REFERENCE: Section 3, page 27-31 ICE-80 Operator's Manual. NOTE: THEN DUMP means display all registers - For this entry may not fit on one line, prior to entering carriage return, enter: "&" and the carriage return. This allows the command to be continued on next line.
8) Upon the break occuring each time you will see the DUMP of all 8080 registers until Memory OUTCT=FH	
9) Emulation terminates with message - Emulation terminated at 3031H. Enter: "DISPLAY CYCLES 10" followed by carriage return.	NOTE: The last machine cycle printed is the last instruction occuring with the breakpoint.
10) Enter: "DISPLAY MEMORY OUTCT" followed by carriage return.	NOTE: This command displays location 3000H=FH. The condition for emulation break.
11) Enter: "DISPLAY ALL PINS" followed carriage return.	NOTE: Pins HLD, HLTA, INT, READY and RESET are displayed.

EXERCISES	REFERENCE & NOTES
III. Emulation (cont.)	
STEP	
12) Enter: "DISPLAY ALL FLAGS" followed by carriage return.	NOTE: The AUX CARRY, CARRY, PARITY, SIGN, ZERO, and INTE are displayed. REFERENCE: Section 3, page 22 & 23 of ICE-80 Operator's Manual-for Pins & Flags.
13) Enter: "EXIT" followed by carriage return.	NOTE: This returns control back to Intellec MDS Monitor. You are now ready to perform the following exercises with ICE-80!

EXERCISES	REFERENCE & NOTES
Exercise 1	
After Bootstrapping the system, initialize ICE-80 and load Wayne3.DEM.	REFERENCE: Figure 1 and ICE-80 Operator's Manual.
Use internal MDS memory and MDS Monitor.	
Map User memory block 3 to be MDS block 6 and User Memory block Ø to be MDS block 7.	NOTE: Don't forget to move location Ø to FH of MDS memory to User Memory.
Execute the program!	
Exercise 2	
Emulate Wayne3.DEM. with a breakpoint set for instruction at location 303FH being executed.	REFERENCE: Figure 1 NOTE: Instruction at 303FH is DCR LOOPC.
What is value stored in LOOPC??	
How many machine cycles for last two instructions?	
Exercise 3	
Enter the parameters to ICE-80 that will cause program to continually output the 1st line repetitively!!	REFERENCE: Page 27-30 of ICE-80 Operator's Manual.

```
                              ;
                              ;
                              ;
                              ;THIS IS A DEMØNSTRATIØN PRØGRAM FØR THE
                              ;                  INTELLEC MDS/ICE 80
                              ;
                              ;
0005                NDIGT     EQU 5
0002                LØØPC     EQU D
                              ;
                              ;ADDRESSES ØF MØNITØR FUNCTIØNS
                              ;
FEB6                LCRLF     EQU 0FEB6H
FE51                DBYTE     EQU 0FE51H
FCA2                CI        EQU 0FCA2H          ;CØNSØLE INPUT
FDA1                CØ        EQU 0FDA1H          ;CØNSØLE OUTPUT
                              ;
3000                ØRG       3000H
3000                ØUTCT:    DS 1                ;ØNE BYTE FØR ØUTPUT CØUNTER
                              ;
3020                ØRG       3020H
3020 312030                   LXI SP,3020H
3023 210030                   LXI H,ØUTCT         ;ØUTPUT CØUNTER
3026 3600                     MVI M,0             ;INIT ØUTPUT VALUE TØ ZERØ
3028 CDB6FE         LINE:     CALL LCRLF          ;PRINT ØNE LINE
302B 1605                     MVI LØØPC,NDIGT
302D 210030         NEXTC:    LXI H,ØUTCT         ;NEXT CHARACTER
3030 34                       INR M
3031 7E                       MØV A,M             ;FØR ØUTPUT SUBRØUTINE
3032 CD51FE                   CALL DBYTE          ;CØNVERT TØ ACSII
3035 0E20                     MVI C,' '           ;SET UP BLANK
3037 CDA1FD                   CALL CØ             ;PRINT BLANK
303A 15                       DCR LØØPC
303B C22D30                   JNZ NEXTC
303E CDA2FC         CHECK:    CALL CI             ;WAIT FØR "SPACE"
3041 E67F                     ANI 7FH             ;STRIP ØFF PARITY BIT
3043 FE20                     CPI ' '             ;SPACE
3045 C23E30                   JNZ CHECK
3048 C32830                   JMP LINE
0000                          END
```

FIGURE 1

SYMBOL TABLE

CHECK	303E	CI	FCA2	DBYTE	FE51	LCRLF	FE86
LINE	3023	LO	FDA1	LOOPC	0002	NDIGI	0005
NEXTC	302D	QUICI	3000				

FIGURE 2

"NOTES"

EXERCISES/ LABORATORY PROJECT

SOLUTIONS

WORKSESSION #1 - SOLUTION

```
        ORG     3000H
        LXI     H,3020H     ;DESTINATION ADDRESS
LOOP:   IN      0
        INR     A           ;IF FFH, A NOW = 0
        JNZ     LOOP
        DCR     A           ;RESTORE TO FFH
        MOV     M,A         ;MOV A TO 3020H
        HLT
        END     3000H
```

WORKSESSION # 2 SOLUTION

Part # 1

```
LOOP:   IN      3       ; input
        ANI     28H     ; isolate bits
                        ; 5 and 3
        CPI     28H     ; 5 and 3 set?
        JNZ     LOOP    ; no loop!
```

Part # 2

```
        IN      4       ; input
        ORA     A       ; set flags
        JM      SET     ; check bit 7
```

Part # 3

```
        MOV     A,E     ; copy E to A
        ORA     D       ; merge in D
        JZ      ZERO    ; jump zero
```

WORKSESSION # 3 SOLUTION

```
MOV     A,C     ; copy C to A
RRC             ; copy C(0) to Carry flag
MOV     A,B     ; copy B to A
RAR             ; Carry to B(7), B(0) to Carry
MOV     B,A     ; copy A to B
MOV     A,C     ; copy C to A
RAR             ; Carry to C(7)
MOV     C,A     ; copy  A to C
```

WORKSESSION #4 SOLUTION

```
WAIT:   IN      0F5H    ;READ IN STATUS
        ANI     1       ;ISOLATE BIT Ø
        JZ      WAIT    ;IS IT A ZERO?
        MOV     A,C     ;NO - CHARACTER TO A
        OUT     0F4H    ;OUTPUT FROM A
        RET             ;DONE
```

- Since the USART (8251) is programmable, it must be initialized prior to use.

```
; Set 8251 mode for - 16x baud rate factor
; 2 stop bits, parity disabled, 8 bit character
;

        MVI     A,0CEH
        OUT     0F5H

; Set 8251 command for
; Receive and transmit enable

        MVI     A,5
        OUT     0F5H
```

- This initialization is done by the System Monitor program in the MDS.

LABORATORY PROJECT #2

SOLUTION:

```
        ORG     3000H
        LXI     H,3100H ; ADDRESS 3100
LOOP:   MOV     A,M     ; READ CHARACTER
        CPI     ':'     ; IS IT A COLON
        JZ      EXIT    ; YES - OUTPUT A  1
        INX     H       ; ADDRESS NEXT CHARACTER
        MOV     A,L     ; GET CURRENT ADDRESS
        CPI     0AH     ; IS IT LAST + 1 ?
        JNZ     LOOP    ; NO
        MVI     A,3BH   ; YES - OUTPUT A  2
EXIT:   ADI     0C7H    ; ":" + 0C7H = 01
        OUT     Ø       ;
        HLT
        END     3000H
```

ALTERNATE SOLUTION:

```
        ORG     3000H
        LXI     H,3100H ;START SEARCH ADDR.
        MVI     B,10    ;SEARCH 10 LOCATIONS
LOOP:   MOV     A,M
        CPI     ':'     ;IS IT A COLON?
        JZ      FOUND
        INX     H       ;UPDATE MEMORY POINTER
        DCR     B       ;UPDATE LOC. COUNT
        JNZ     LOOP
        MVI     A,2
        OUT     0       ;OUTPUT A 2
        HLT
FOUND:  MVI     A,1
        OUT     0       ;OUTPUT A 1
        HLT
        END     3000H
```

LABORATORY PROJECT #3 - SOLUTION

(BASED ON 'ALTERNATE' SOLUTION OF #2)

```
;PROGRAM NAME:  FIND COLON
          ORG     3000H
          LXI     SP,3B00H
          LXI     H,3100H    ;START SEARCH ADDR
          MVI     B,10       ;SEARCH 10 LOCATIONS
LOOP:     MOV     A,M
          CPI     ':'        ;IS IT A COLON?
          JZ      FOUND
          INX     H          ;UPDATE MEM POINTER
          DCR     B          ;UPDATE LOC COUNT
          JNZ     LOOP
          LXI     H,MSG1     ;ADDR OF 'NO FIND'
          JMP     OUTPT
FOUND:    LXI     H,MSG2     ;ADDR OF 'FOUND'
OUTPT:    CALL    PRINT      ;PRINT MESSAGE
          HLT

;MESSAGES

MSG1:     DB      13,'NO CAN FIND',ODH,OAH
MSG2:     DB      12,'I FOUND IT',ODH,OAH

;PRINT SUBROUTINE

PRINT:    MOV     B,M        ;LOAD CHAR COUNT
LOOP1:    INX     H
          MOV     C,M        ;LOAD A CHAR
          CALL    CO         ;PRINT CHAR IN C
          DCR     B          ;UPDATE CHAR COUNT
          JNZ     LOOP1
          RET

;EQUATE STATEMENTS

  CO      EQU     ØF809H

          END     3000H
```

LABORATORY PROJECT #4 - SOLUTION

```
;PROGRAM NAME:  TTY COLON MONITOR
        ORG     3000H
        LXI     SP,3B00H
LOOP:   CALL    CI          ;TTY CHAR TO ACCUM
        CPI     ':'         ;IS IT A COLON?
        JNZ     LOOP
        LXI     H,MSG1
        MOV     B,M         ;LOAD CHAR COUNT
LOOP1:  INX     H
        MOV     C,M         ;LOAD A CHAR IN C
        CALL    CO          ;TYPE CHAR
        DCR     B
        JNZ     LOOP1
        JMP     LOOP

;MESSAGE

MSG1:   DB      19,'YOU TYPED A COLON',0DH,0AH

;EQUATE STATEMENTS

CO      EQU     ØF809H
CI      EQU     ØF803H
        END     3000H
```

SECTION IV

REFERENCE MATERIALS

CONTENTS

PAGE #

APPENDIX A 4-2

MESSAGE OUTPUT SUBROUTINE

APPENDIX B 4-4

TIME OF DAY ROUTINE

APPENDIX C 4-6

8080 SYSTEM CONTROL

" NOTES "

MESSAGE OUTPUT SUBROUTINE

```
MEMORY
ADDRESS   CONTENTS

400,1,1                    CALL    PRINT
403,4       500            DW      MSG1     ; ADDRESS OF MESSAGE
405                    <CONTINUE>
                             .
                             .
                             .
                             .
                             .
                             .
                             .
500-50C   0C...   MSG1:    DB      (FINI-MSG1-1),'START READER'
50D               FINI:    DB      1
                             .
                             .
                             .
                  PRINT:   XTHL             ; HL <--> TOS
                           MOV     E,M      ; E <--- MEMORY
                           INX     H
                           MOV     D,M      ; D <--- MEMORY
                           INX     H
                           XTHL             ; HL <--> TOS
                           XCHG             ; HL <--> DE
                           MOV     B,M      ; B <--- MEMORY
                  MORE:    INX     H
                           MOV     C,M      ; C <--- MEMORY
                           CALL    CO       ; GO TO MONITOR
                           DCR     B        ; B <--- B - 1
                           JNZ     MORE
                           RET
                             .
                             .
                             .
```

XTHL REVEALED

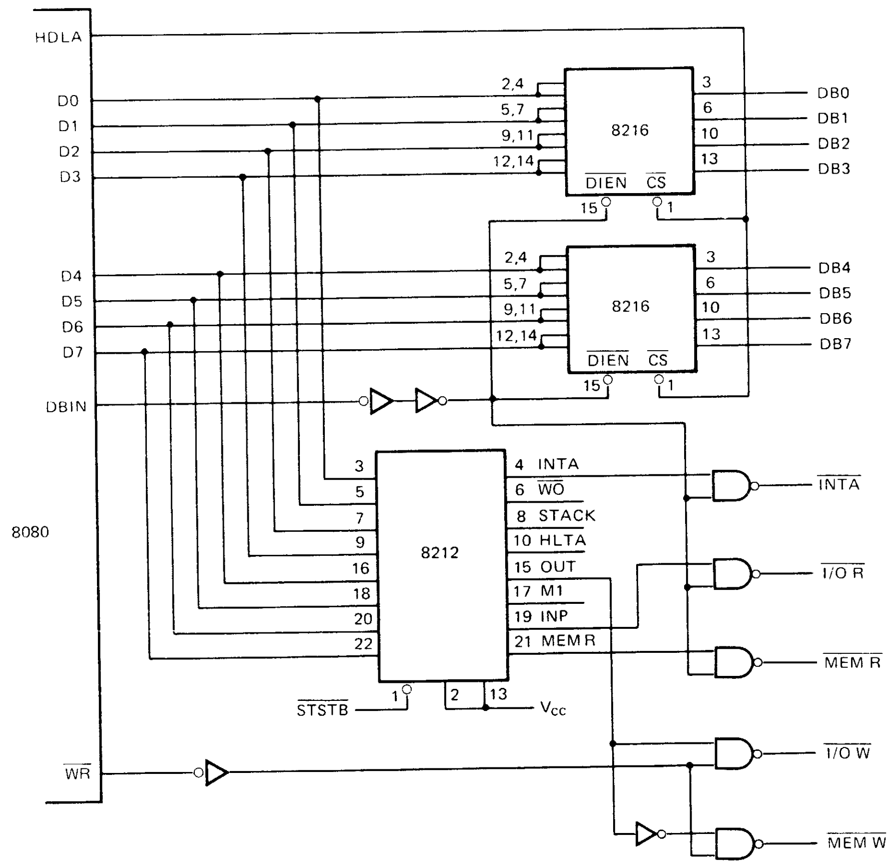
Label	Instruction	Registers	Stack
	.		
	.		
	CALL PRINT		SP → 03 / 04
	.		
	.		
	.		
PRINT:	XTHL	H 04, L 03	SP → ? / ?
	MOV E,M	E 00	
	INX H	H 04, L 04	
	MOV D,M	D 05	
	INX H	H 04, L 05	
	XTHL	H ?, L ?	SP → 05 / 04
	XCHG	H 05, L 00 D ?, E ?	
	MOV B,M	B 0C	
	.		
	.		
	.		

TIME OF DAY CLOCK

- ASSUME A 1 SECOND COUNTER INTERRUPTING AND FORCING A RST 2 INSTRUCTION
- 3 BCD COUNTERS (SECONDS, MINUTES, HOURS) IN RAM MEMORY

```
        ORG     10H
        JMP     TIME
         .
         .
         .
        ORG     "XXXX"  ; SOMEWHERE IN RAM MEMORY
SEC:    DS      1       ; COUNTER
MIN:    DS      1       ; COUNTER
HRS:    DS      1       ; COUNTER
         .
         .
         .
        ORG     "YYYY"  ; IN ROM MEMORY
;
;  INITIALIZE COUNTERS TO ZERO & START TIMER
;
        XRA     A       ; SET A TO ZERO
        LXI     H,SEC   ; GET SECONDS ADDRESS
        MOV     M,A     ; SET SECONDS TO ZERO
        INX     H       ; ADDRESS MINUTES
        MOV     M,A     ; SET MINUTES TO ZERO
        INX     H       ; ADDRESS HOURS
        MOV     M,A     ; SET HOURS TO ZERO
        MVI     A,1     ; TIMER START COMMAND
        OUT     2       ; ASSUME TIMER ATTACHED BIT 0
        EI              ; INTERRUPT SYSTEM ON
         .
         .
         .
        DI              ; DISABLE INTERRUPT SYSTEM
                        ; BEFORE READING TIME
```

```
TIME:   PUSH    H        ; SAVE H AND L
        PUSH    PSW      ; SAVE A AND FLAGS
; ------------------------------------------------
        LXI     H,SEC    ; GET SECONDS COUNTER
        MOV     A,M
        INR     A        ; INCREMENT IT
        DAA              ; REMAKE INTO BCD
        MOV     M,A      ; PUT IT BACK INTO MEMORY
        CPI     60H      ; 60 SECONDS YET ?
        JNZ     EXIT     ; NO
        MVI     M,0      ; YES, SET SECONDS TO ZERO
; ------------------------------------------------
        INX     H        ; GET MINUTES COUNTER
        MOV     A,M
        INR     A
        DAA
        MOV     M,A
        CPI     60H      ; 60 MINUTES YET ?
        JNZ     EXIT     ; NO
        MVI     M,0
; ------------------------------------------------
        INX     H        ; GET HOURS COUNTER
        MOV     A,M
        INR     A
        DAA
        MOV     M,A
        CPI     24H      ; 24 HOURS YET ?
        JNZ     EXIT     ; NO
        MVI     M,0
; ------------------------------------------------
EXIT:   POP     PSW      ; RESTORE FLAGS AND A
        POP     H        ; RESTORE L AND H
        EI               ; ENABLE INTERRUPTS
        RET
```



8080 System Control