Microprocessors and Microcontrollers

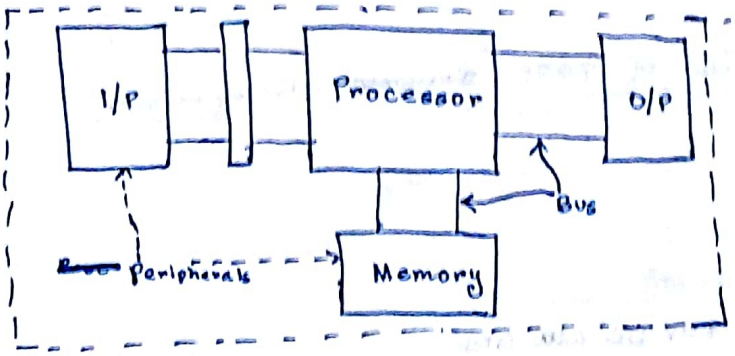
IC 8085
IC 8086

IC 8051 → MCS Family

1971 → Intel → IC 4004 → 4 bit → 740 KHz → PMOS Technology
1971 → Intel → IC 8008 → 8 bit → 500 KHz → PMOS Technology
1975 → Intel → IC 8080 → 8 bit → 2 MHz → NMOS Technology
1976 → Intel → IC 8085 → 8 bit → 3 MHz → NMOS Technology
1978 → Intel → IC 8086 → 16 bit → 8 MHz → NMOS Technology
→ " IC 80386 → 32 bit
⋮
IC 80586
↓
Pentium

Processor: A programmable device which can perform data manipulation, or any other operation. Within a processor data is present in binary.

- **Peripheral Devices**: Devices supporting/surrounding the processor.



Minimal System

- Bus → 1 bit/16 bit ..... → Conducting Lines

- If a processor is 8 bit, it means that at a time, it can process an information 8 bit long.

A | H
1010 0001, 8 bit

- Using 8 bit processor, $2^8 = 256$ different signals can be sent.

~~8 bit~~

Memory

Address Mem	Data Mem
0000H	
0001H	
0002H	
0003H	XY

- To identify the unique memory register XY, the address memory 03H is used.

8 bit bus

16 bit bus (To identify a unique memory location)

$2^{16} \simeq$ 64 KB = 65536 of data handling capacity

A maximum of $2^8 = 256$ I/O devices can be attached.

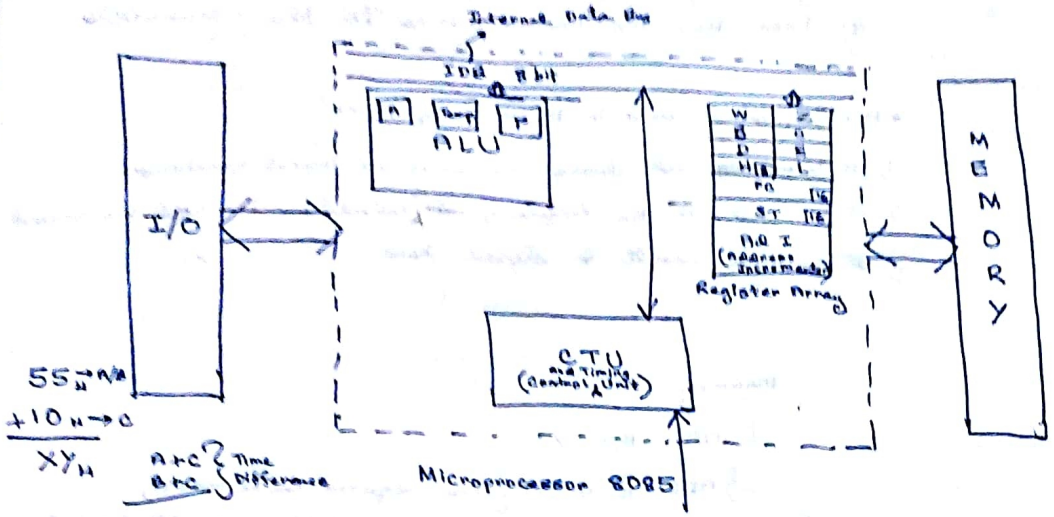
- ~~One of the~~
- 8 bit bus used to send data.
- 16 bit bus used to retrieve from unique memory.

Architecture of 8085 ~~Processor~~ Microprocessor :

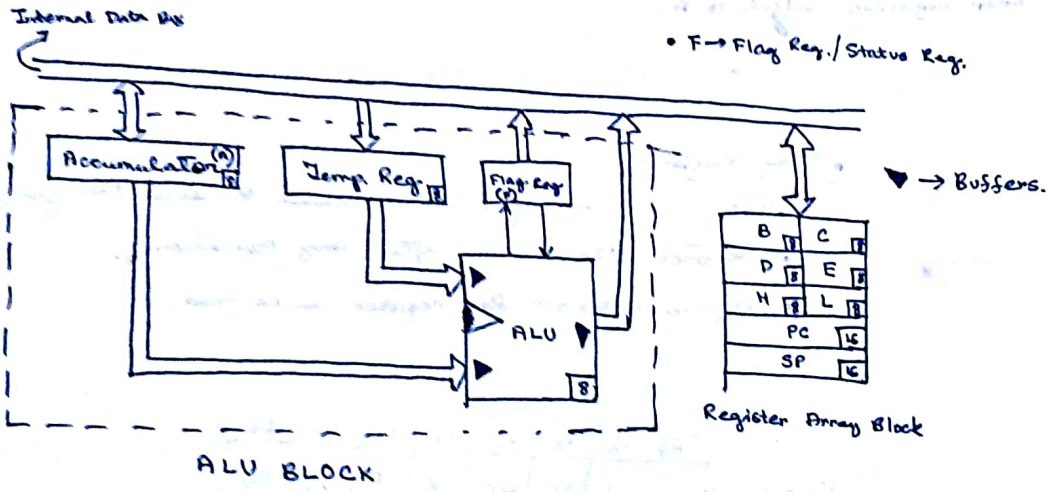
Features

1) 40 pin IC
2) Power Supply: +5V DC with GND
3) Operating Frequency: 3 MHz $T = \frac{1}{f}$ = ... μs
4) Bus: Data Bus, Address Bus, Control Bus
   - Data Bus: 8 bit
   - Address Bus: 16 bit
   - Control Bus: No specific width; may be single line, 2 lines etc; some are bidirectional, some are unidirectional.
5) Register based device: General purpose register → B, C, D, E, H, L → 8 bit, special purpose register (A, F, PC, SP (Stack Pointer)) — A: 8 bit; F, PC, SP: 16 bit, Machine accessible (Temporary) register.
   - W, Z, Temp (8 bit)

- General Purpose Register: To hold data for arithmetic / logical operations.
- Sp. Purpose Register: A, F → can hold 16 bit data ~~(by bit)~~ → Two Registers (Register Pairs) can be grouped to hold 16 bit data.
  PC, SP → cannot hold data.



CTU: Depending upon the internal states, will synchronise the I/P, O/P devices.



- If the ALU is 8 bit, the processor can process a maximum of 8 bit long data.
- If your processor is 8 bit, what does it actually depend on —
  a) Int. Reg.
  b) ALU

Adding:

1) 55 → A
   +10 → C
2) From C the info is copied to Temp. Reg.
3) For addition, both data will move to the ALU they will be added.

Code.

4) From ALU, the result moves to the Accumulator.

- Accumulator is used in the following cases:

1) It is used during arithmetic and logical operations
2) It is used at the beginning of calculation to hold one num
3) The final result is stored here.

Mnemonic

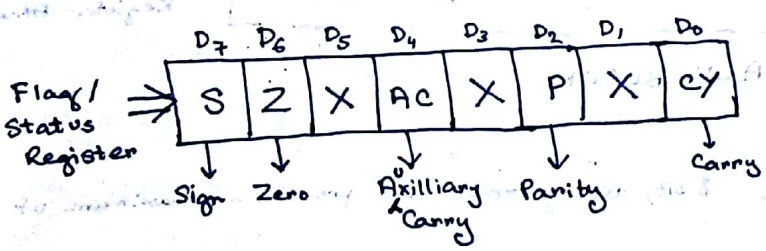
1) ADD B, C ✗

2) ADD B ← Only one register can be written, other register is hidden and adds the conte of the accumulator.

If this is done first, on the contents of A, garbage is added to B. nce, before ②, ③ must e used to move the ter register's content to A.

3) MOV A, C → C's contents are moved to A
↓ ↓
Dest. Source

- Flag Register
  - Only 5 bits out of 8 will be used to serve the purpose.
  - Denotes the status after any operation.
  - Only the status of flag register can be read.



- In Hex, Max data FF_H, $\xrightarrow{+1} 100_H$ ∴ Carry → 1
  Min data 00_H. } Within this range, no carry
- In case of even Parity, if after the last operation, there is even number of 1's, then P is high.
- If the last arithmetic operation has non-zero value, then Z is low, otherwise high
- For the result, Sign (S) will be zero.

- If there is carry-over from D_0, D_2, D_4, Then AC is set.
- Program Status Word (PSW) } The combination of 8 bits of the Flag Reg.
- In the register array block, the registers on the right (C, E, L) hold low, ones on the left hold high.

$A = FF_H$

$B = 01_H$

ADD B

$A = 00_H$

```
  FF
+ 01
 100
```

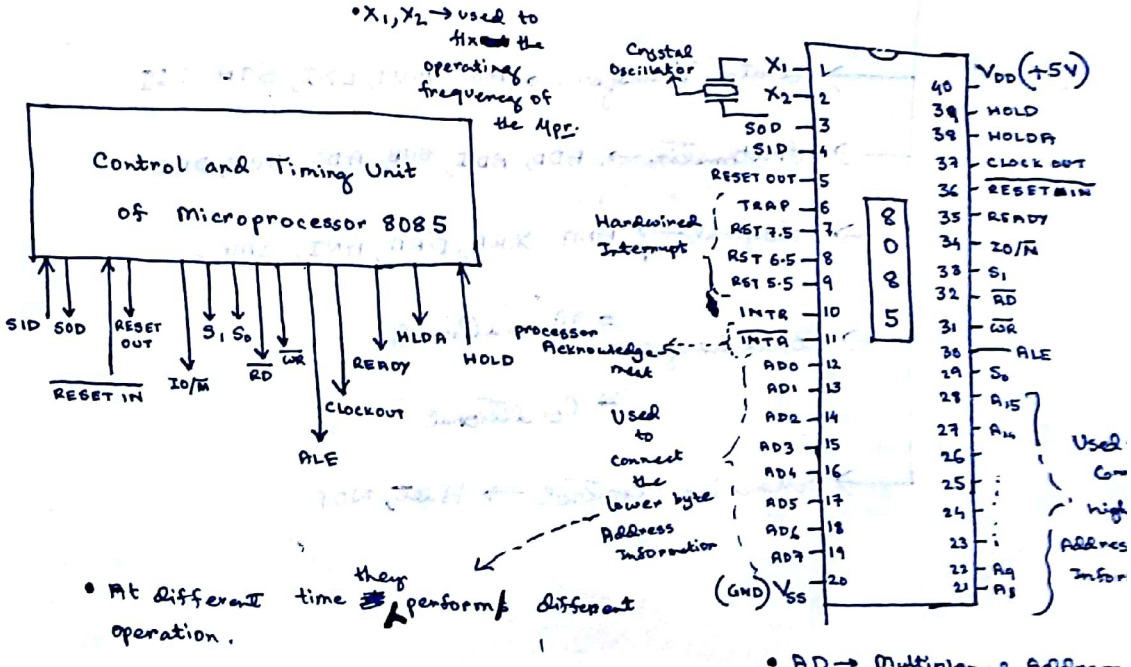
This bit is lost, and 00 is stored in A.

Flag, $F = 55_H$

D_6 0101 — Zero Result

D_0 0100**1** — Even number Zero's — Carry=1

PIN Diagram of Microprocessor 8085

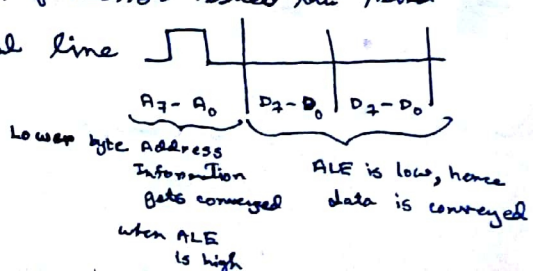


- At different time they perform different operation.

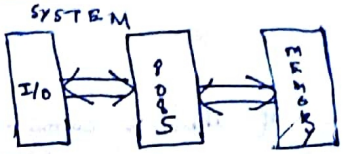
- AD → Multiplexed Address Data Bus

- ALE → Address Latch Enable

Via ALE, processor issues this kind of signal line



Types of Operation —



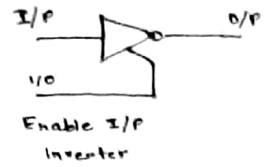
		$I/\overline{R}$	S_1	S_0
i)	opcode fetch	0	1	1
ii)	Memory Read	0	1	0
iii)	Memory Write	0	0	1
iv)	I/O Read	1	1	0
v)	I/O Write	1	0	1
vi)	Interrupt	1	1	1
vii)	Acknowledgement			
viii)	Halt	Z	X	X
ix)	HOLD	Z	X	X

Depending upon type of operation —

- Data Transfer → MOV, MVI, LXI, STA, LDA
- Arithmetic → ADD, ADI, SUB, ADC, DCR, DAD
- Logical → ANA, XRA, ORA, ANI, CMP, ...
- Branching → Unconditional
  - → Conditional
- Machine control → Halt, NOP

Tri-Stated Logic devices

Tristated
Inverter with Active High input enable

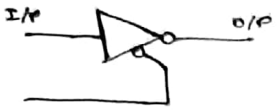


Enable	I/p	O/p
1	1	0
1	0	1
0	X	Z

→ High impedance state

- If enable is 0, then the whole device is deactivated

Inverter with
Tristated & Active Low input enable

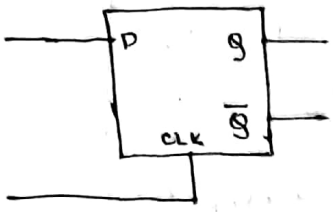


Enable	I/p	O/p
0	1	0
0	0	1
1	X	Z

- In case of buffers, output remains same as input.

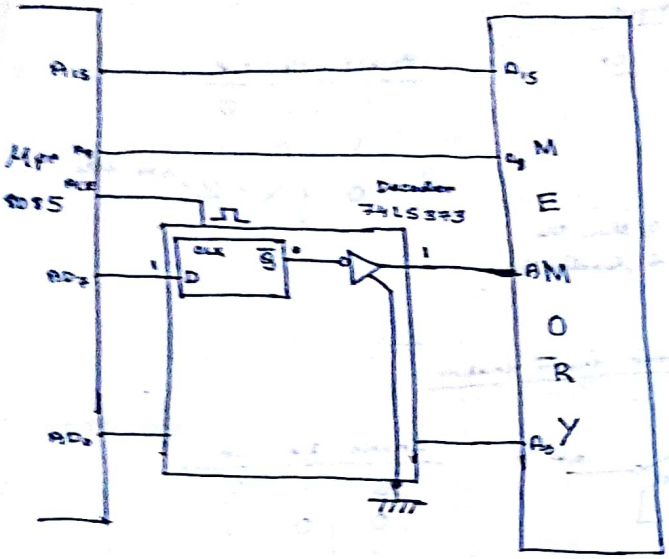
Tristate Buffer with Active low enable I/P

Enable	I/P	O/P
0	1	1
0	0	0
1	X	Z

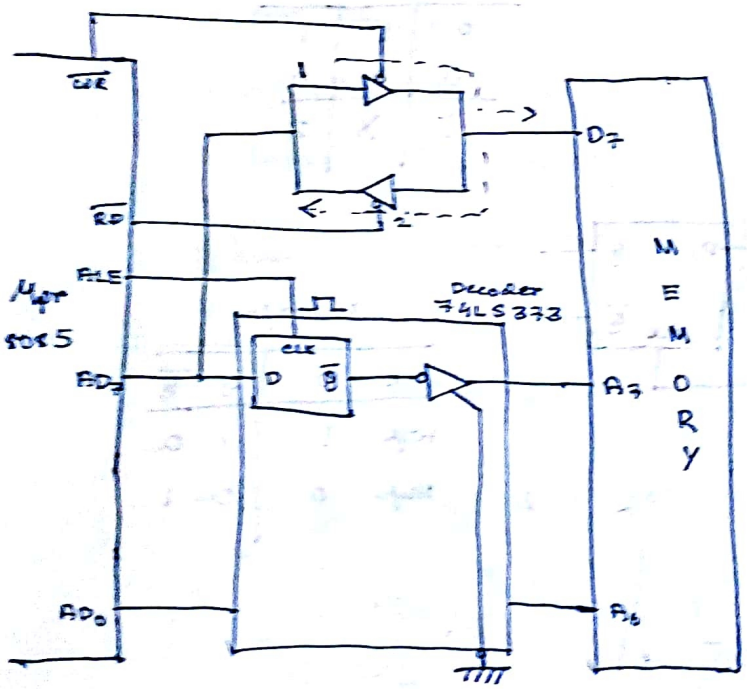


D - F/F

CLK	D	Q	$\overline{Q}$
High	1	1	0
High	0	0	1



- ALE triggers short clock-pulses
- By this system, no data will be directed towards processor.
- When ALE signal line is low, then the AD lines get used as data bus.



- RD & WR are activated one at a time.

- Explanation Required

- Processor requires 3 clock pulses for performing I/O Read, I/O Write, Mem. Read, Mem. Write. This time is called Machine Cycle.
- Machine Cycle is the time required for the processor to perform a specific task.
  - For opcode fetch operation, it is 4 clock cycles.
  - For Interrupt acknowledgement, 6 clock cycles are required.

- Machine Cycle.

  M-Cycle → 3-6 T

- Instruction Cycle → 4-18

8000_H : MVI B, $5A_H$

8000_H : (06H) Opcode Fetch 4

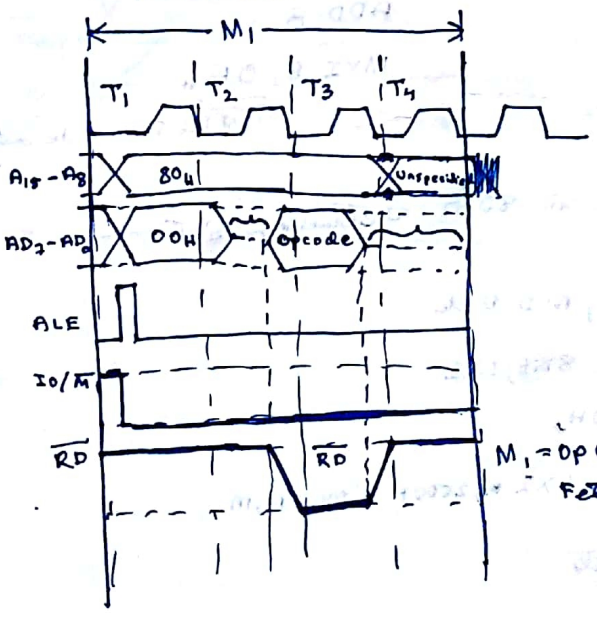
8001_H : $5A_H$ Mem. Read 3

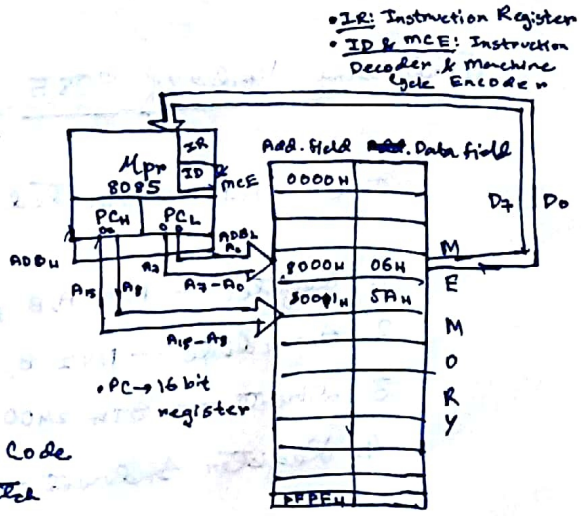
7

~~clock pulses~~

- Timing Diagram → Diagramatic representation where we represent the state changes of control and timing diagram.

- T-States

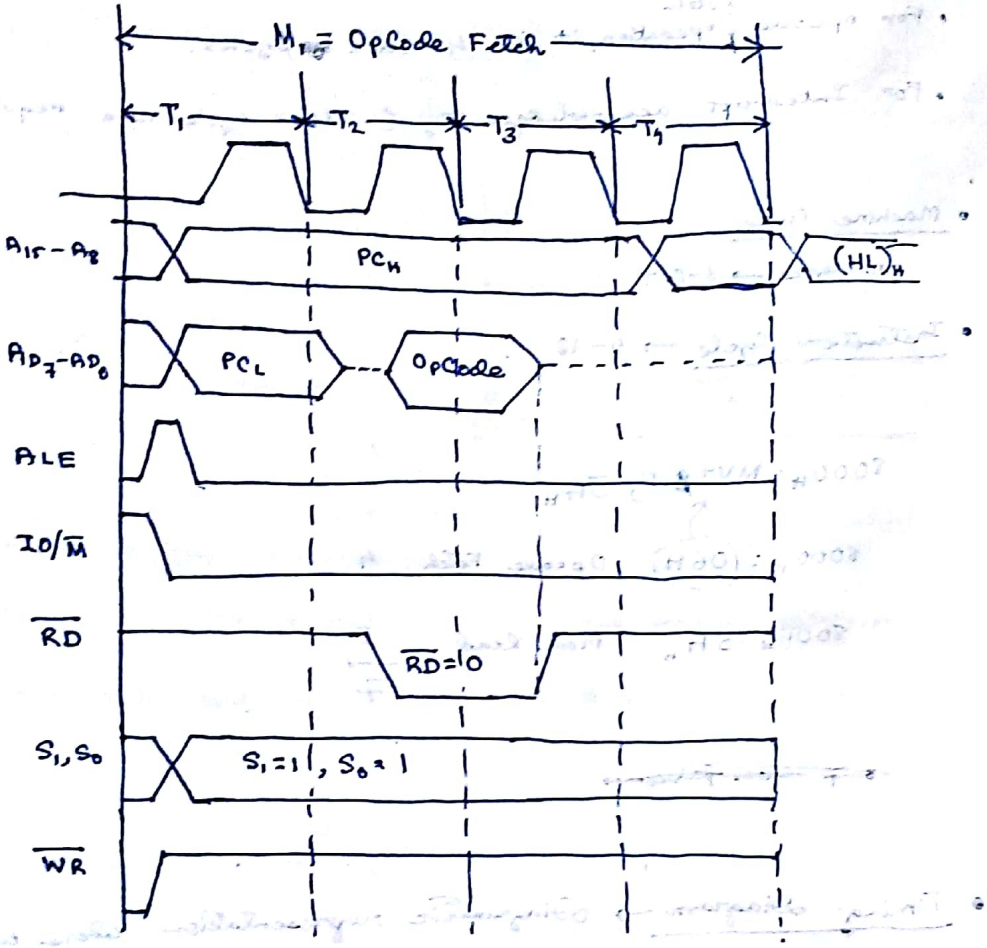




8000_H : MVI B, $5A_H$

OpCode Fetch Diagram

~~T-States~~



PCL (PC+1)

ADD B

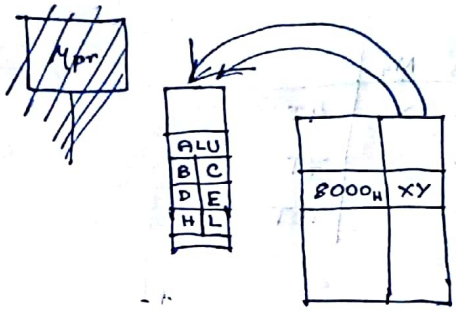
MVI B, 0A$_H$

8 bit immediate data

Addressing Modes of 8085

Five A.M. supported in 8085 Instruction Set —

1. Register — MOV A,B ; ADD E etc
2. Immediate — MVI B, 8bit, LXI
3. Direct — STA 2400H,
4. Register Indirect — LXI H,2500H ; MOV A,M
5. Implied or Implicit



Load direct Accumulator

LDA 8000H ---> Loads data from 8000H to Acc.

↓ memory location

LXI B, 8000H

↓

Gets used as Immediate data

STA 8001H ---> Stored in memory

MOV A, M ---> Memory

↓

Pointed by HL Register Pair

```
LXI  H, 8000H
MVI  B, 02H
MOV  A, M
INX  H
ADD  B
```

HL

80	00

MOV M, A / STA 8001

• XCHG → Exchange

D E

H L

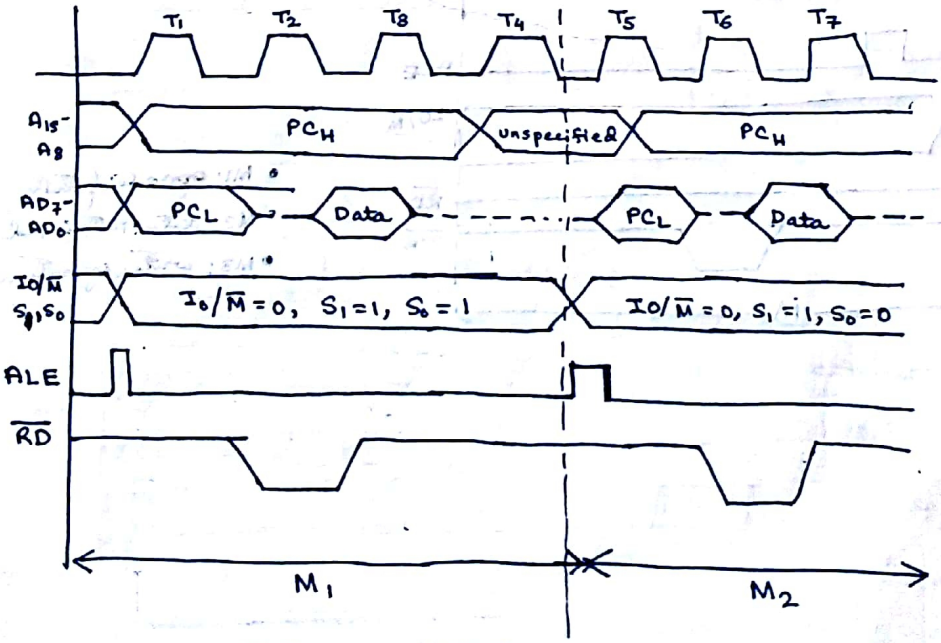
Register Indirect

H, L → used for pointing any address.

H	L
80	00

MOV A, M ← Uses an arbitrary register and doesn't have any physical significance

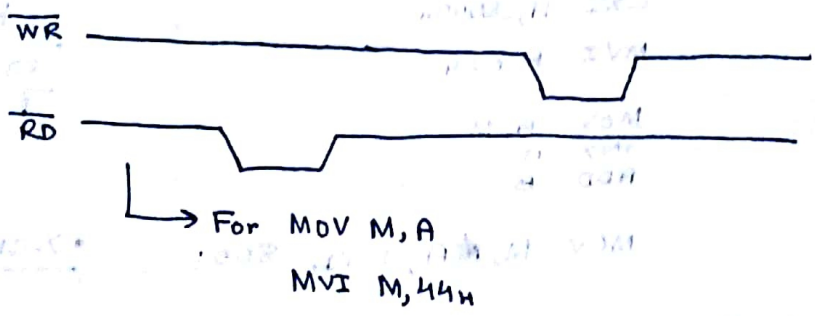
• CMA → means the accumulator automatically complemented

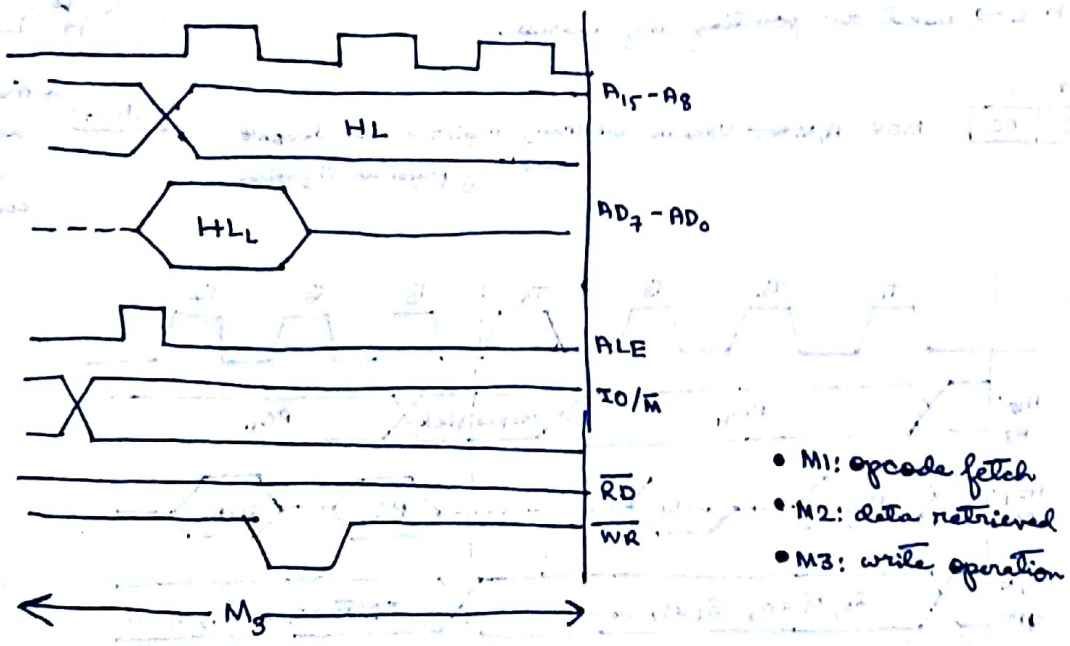


No of bytes	M_1	M_2	M_3	M_4	
1	✓	✗	✗	✗	4T
2	✓	✓	✗	✗	7T
3	✓	✓	✓		10T

Opcode fetch (M_1) = 4T
IO/Mem Read (M_2) = 3T
for 3 byte, ($M_2 = M_3$)

for Add M → In the timing diagram, just replace PC with HL.





Instructions

CMP D

		CY
Data with no carry ← A > D	0	0
Data with some carry ← A < D	0	1
A = D	1	0

↓ Zero Data

A = 0D D = 02

1) CMP D ---> code for comparison with the content of the Acc.

A − D ⇒ non-zero
CY = 01 (∴)
∴ A ← A − D

2) CPI 05H A > Immediate Data
A < Imm Data
A = Imm. Data

3) ~~DAD~~ ADD D ---> A ← A + D

DAD D ---> HL ← HL + ~~DE~~ DE

↓ Double Addition

DA A

↓ Decimal Adjust Accumulator

Data 1 = 0F	09 H
" 2 = 03	03 H
12 H	0C H
	+06
	12

A = 03H
D = 09H
ADD D
DA A

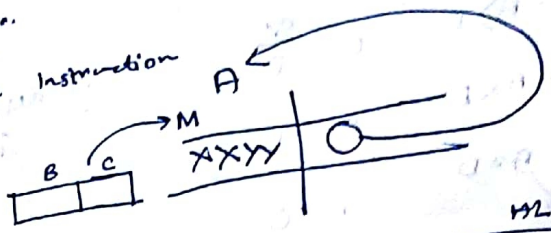
0000 1101
0000 0110
0001 0010

- Auxilliary carry is set when there is a carry generation from lower to higher nibble, i.e. D_3 to D_4

1) LDA X → Load Accumulator Direct, 2 byte instr. B/D → 3 byte instruction

3 byte instr.

2) STA X

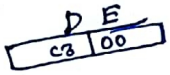


3) LHLD C100H

↓ Load HL Reg. pair direct

LDAX B / MOV A, M

STAX D / MOV M, A



C100

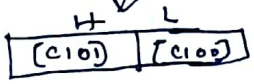
L

H

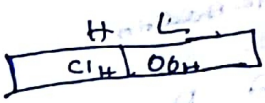
C101

LDA C200 → A

STA C300 ← A

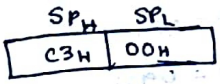


LXI H, C100 →



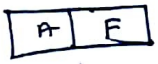
16 bit

LXI SP, C300H



Program Status Word

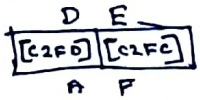
PUSH Rp/PSW



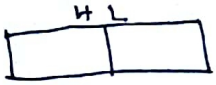
POP Rp

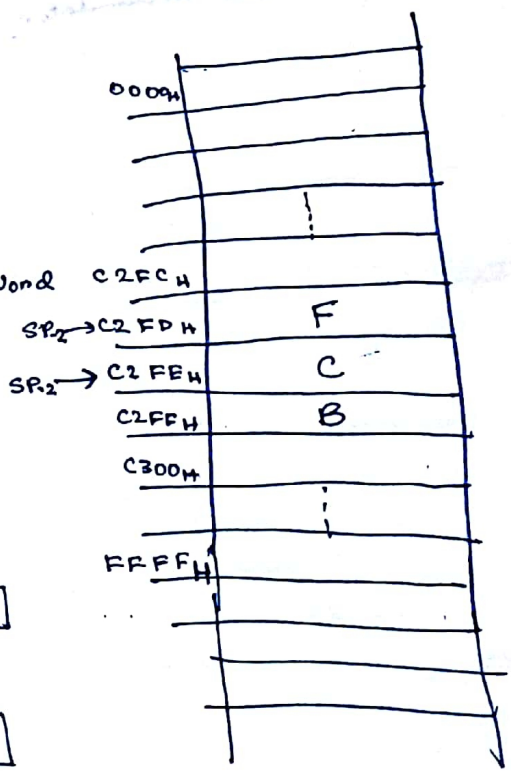
PUSH PSW

POP D



POP H





- If extra POP, it exceeds initial memory location, it gives an error
- PUSH, POP → single byte instruction.

Subroutine

- when CALL instr. is used, internal Stack Memory is used.
- It is better to initialize SP on a location we are not using or at FFFF, otherwise some conflict might take place.

8000: LXI SP, FFFF$_H$ (3 byte)
8005H: CALL (9000$_H$)
8008H: Next Instr.
800F: CALL 9000H
8015: CALL 9000H
801FH: RST 1

Main Program

•CALL → CD$_H$

Start Subroutine ← 9021H!

8000$_H$: ~~Instr~~ LXI H, C100$_H$
MVI B, XX$_H$
Other Instr.
Other Instr.

8008$_H$: MOV A, M
ADD B
JNC L1
INR D
L1 : INX H
MOV M, A
Other Instr.
Other Instr.
Other Instr.

8013$_H$: MOV A, M
Other Instr.

8020

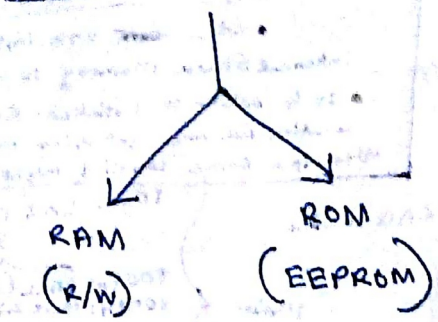
9000H: Location
Subroutine: MOV A, M
ADD B
JNC L1
INR D → CALL
→ RET
L1: INX H
MOV M, A
RET → Return

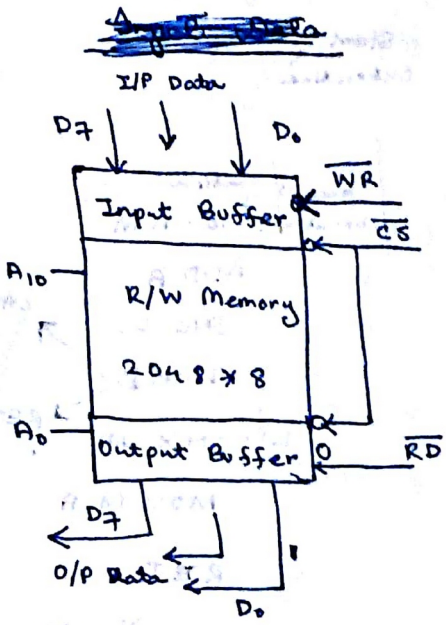
FFFD$_H$: 09$_H$ → lower byte
FFFE$_H$: 80H → higher byte
FFFF$_H$: SP
(Next Instruction)

8005$_H$	CALL
8006$_H$	00H → Z
8007$_H$	90H → W
9000$_H$	

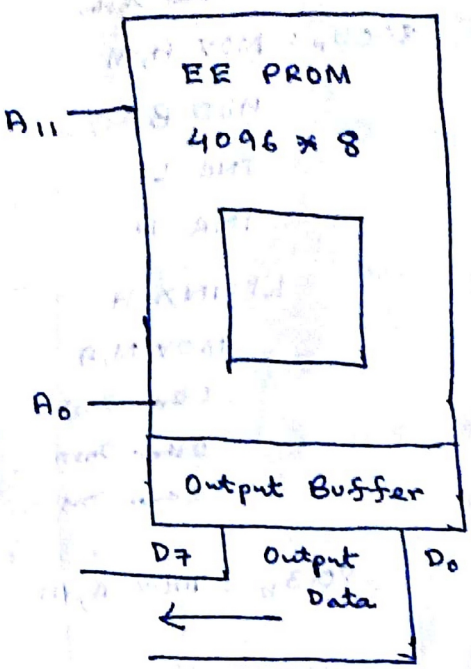
•CALL (XXYY)
↓
— 3 byte instr.
— Requires 5 machine cycles

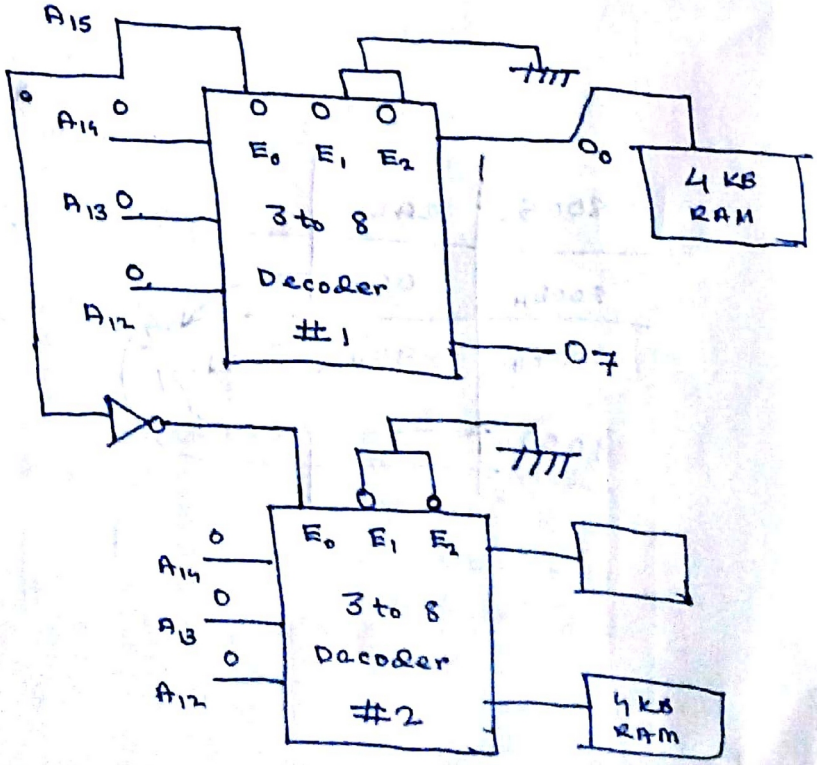
Memory Interfacing





R/W Static Memory





Initial Addr

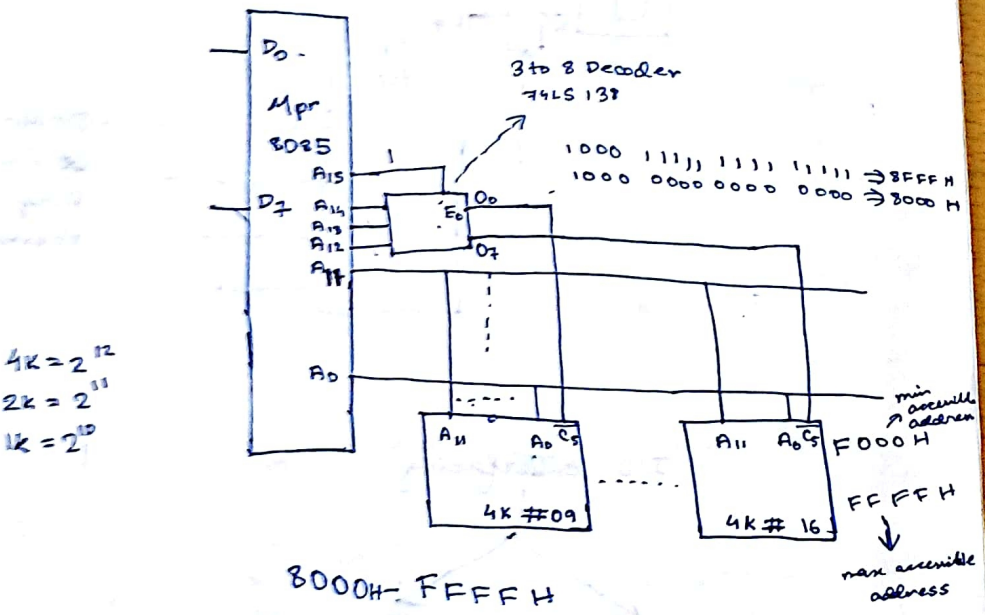
F000 + 2^{12}

= F000 + 0FFF

= FFFF H

ROM { 8000 H → DFFF H ; E000 → FFFF

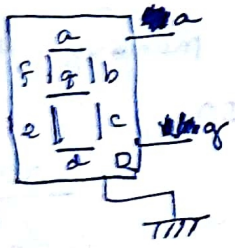
ROM 0000 H → 7FFF H



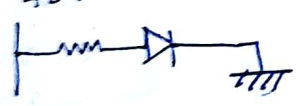
4K = 2^{12}
2K = 2^{11}
1K = 2^{10}

8000H - FFFF H

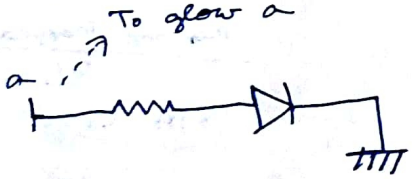
• Talk about input and output buffer drawn in the previous page.



7 Segment Display



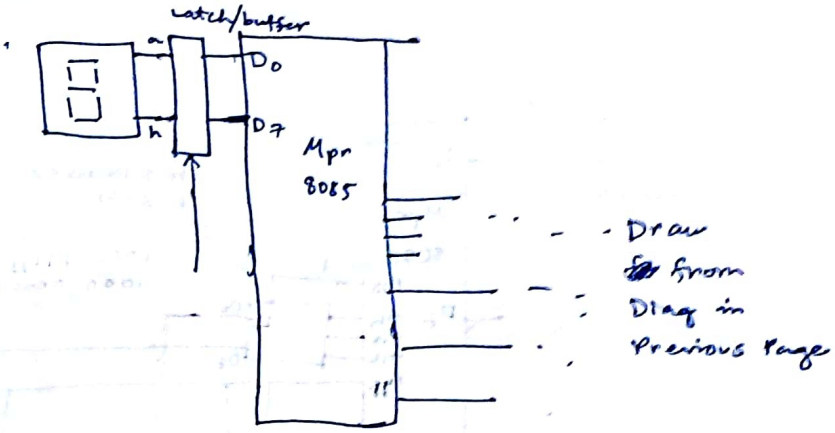
To glow a

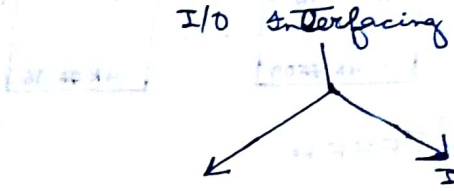


a b c d e f g h

To glow A → 1 1 1 0 1 1 1 0 ⇒ EE H

To enable —





Memory Mapped I/O (16 bit I/O address is accessed)

Mapped I/O (I/O address is 8 bit)

- Compare between memory mapped I/O and I/O mapped IO.

Memory Mapped I/O

O/P

STA C100

I/P

LDA C101

IO/$\overline{M}$

13/7 T-states

IO Mapped I/O

I/O

LDA C101

OUT FF

IN 12

IO/$\overline{M}$

10 T States

8 bit IO Port Address Range → FF H, 00 H

STA C100 → A

OUT 11 H → A — IO WRITE

LDA CD00 → A

IN 12 → A — IO READ

opcode fetch + memory read

4 + 3 + 3 = 10 T

- IOADR → IO Address
- IOSEL → IO Select

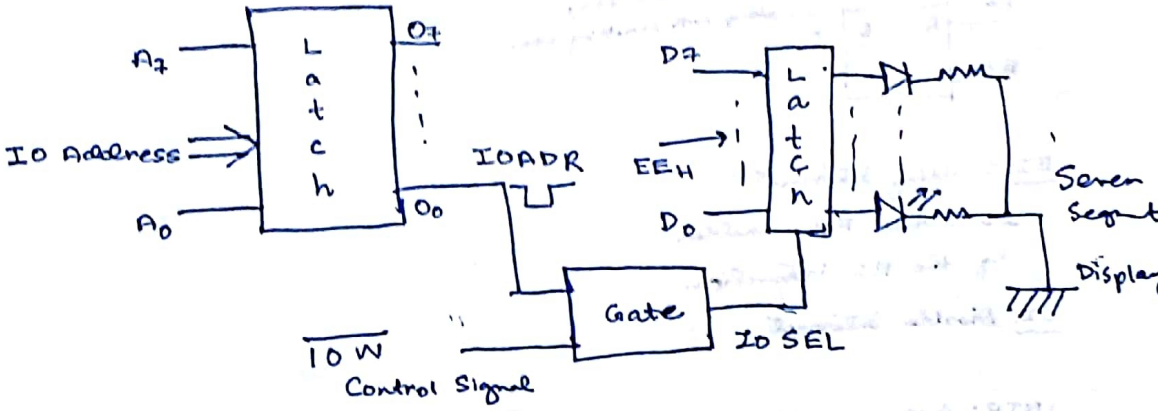
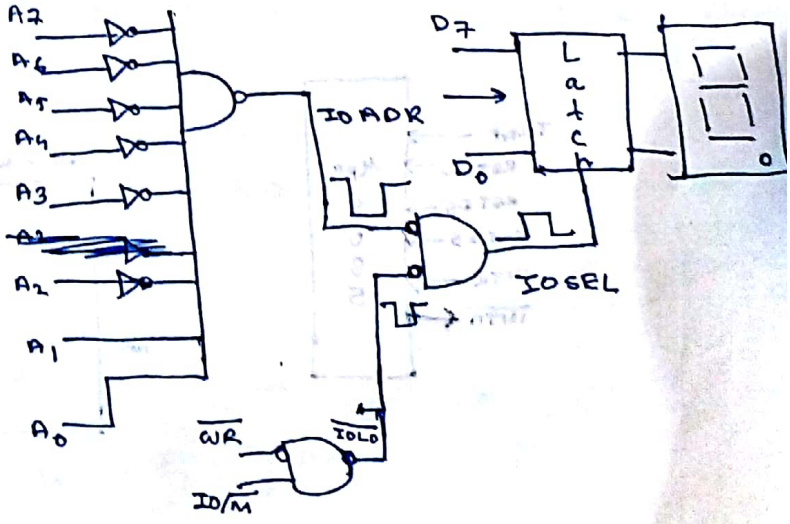


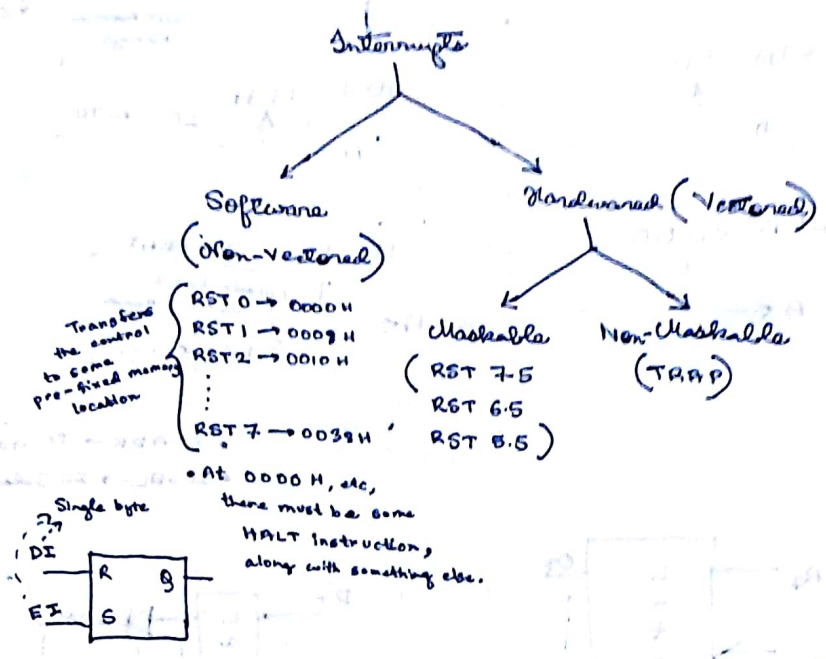
Fig-(1): generalised IO Connection Diagram

A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	I/O
1	1	1	1	1	1	X	X	0/0
0	0 → FC							0/0
0	1 → FD							0/0
1	0 → FE							0/0
1	1 → FF							
								0/0
								1/1
								1/1



Fig(2): IO Interfacing circuit with NAND gate

- If you are provided with 8 Kbyte chip, and initial address is given, what will be the final address.
- Draw the circuit and Explain.



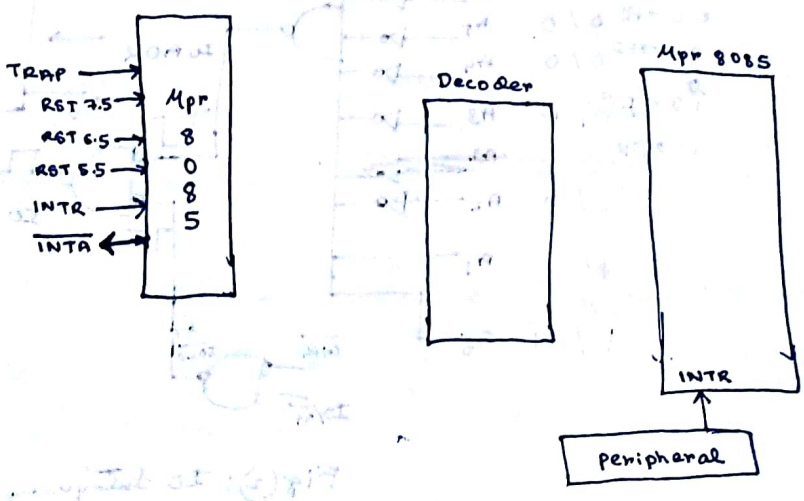
EI: Enabling Interrupt

The above FF is enabled by the EI instruction.

DI: Disable Interrupt

INTR: Interrupt Request

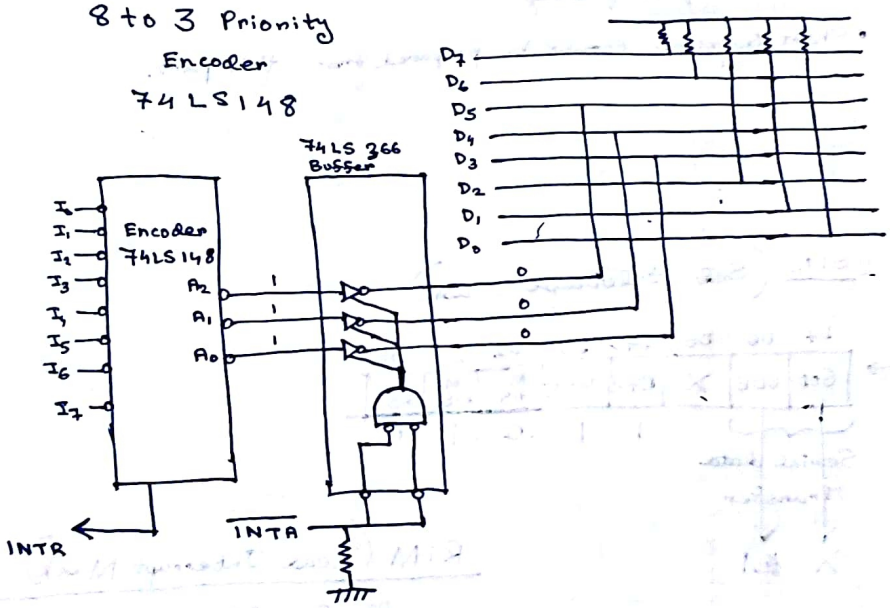
$\overline{\text{INTA}}$: Interrupt Acknowledgement



Call Loc	Code	Hex Code		D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0000H	RST 0	C7	→	1	1	0	0	0	1	1	1
0008H	RST 1	CF	→	1	1	0	0	1	1	1	1
0010H	RST 2	D7	→	1	1	0	1	0	1	1	1
0018H	RST 3	DF	→	1	1	0	1	1	1	1	1
0020H	RST 4	E7	→	1	1	1	0	0	1	1	1
0028H	RST 5	EF	→	1	1	1	0	1	1	1	1
0030H	RST 6	F7	→	1	1	1	1	0	1	1	1
0038H	RST 7	FF	→	1	1	1	1	1	1	1	1

Circuit Not Required for Sem

8 to 3 Priority Encoder 74LS148



Vectored Interrupt

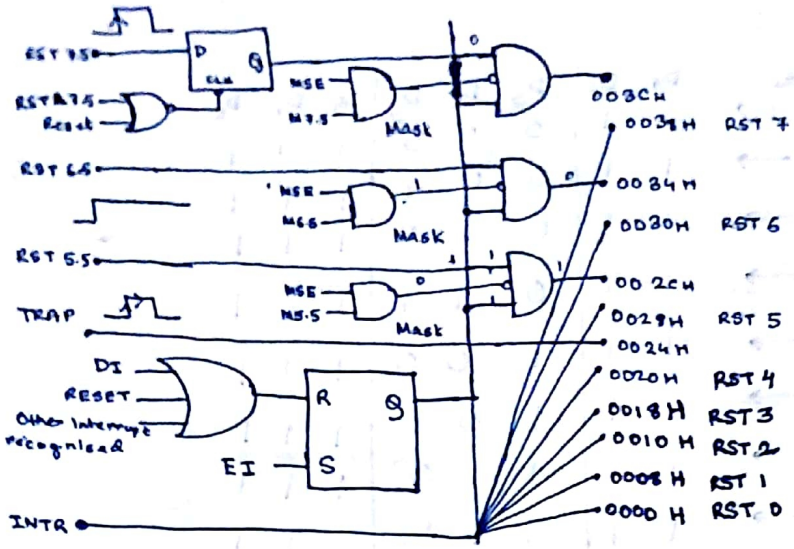
Automatically program control is transferred

RST 1 CFH 0008H

Vectored Interrupt

		mem. loc.
TRAP	6	0024H
RST 7.5	7	003CH
RST 6.5	8	0034H
RST 5.5	9	002CH

INTR



- TRAP has highest priority
- Short Questions should be prepared from this part.

SIM (Set Interrupt Mask)

	D7	D6	D5	D4	D3	D2	D1	D0
A →	SOD	SDE	X	R7.5	MSE	M7.5	M6.5	M5.5
				1	1	0	1	0

Serial data Transfer

X X1

RIM (Read Interrupt Mask)

	D7	D6	D5	D4	D3	D2	D1	D0
A =	SID	I7.5	I6.5	I5.5	IE	M7.5	M6.5	M5.5

Pending Interrupt

Mask Interrupt

Serial I/P Data

Interrupt Enable F/F Status

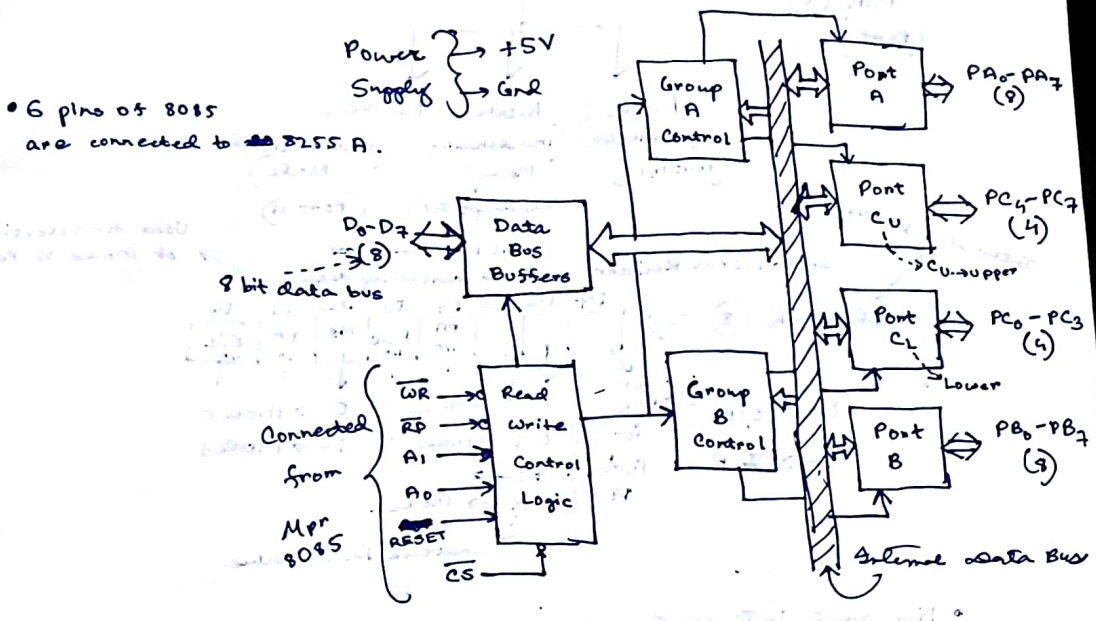
```
RIM   A=
ANI   18H=
JNZ   L1
JMP   L2
L1: MVI A,00H
    SIM
    .
    .
    .
L2: HLT
```

- Draw the hardware diagram

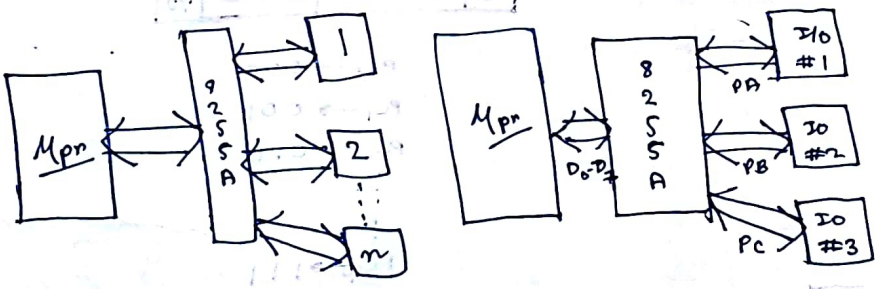
~~• Draw the hardware diagram.~~

Programmable Peripheral Interface (PPI) 8255 A

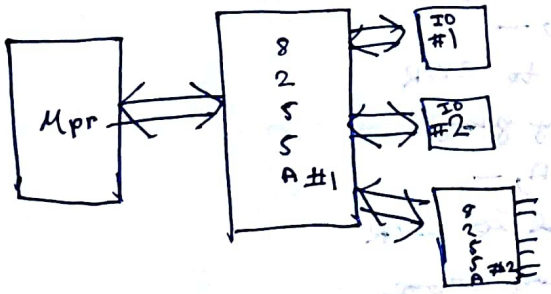
Advanced Version



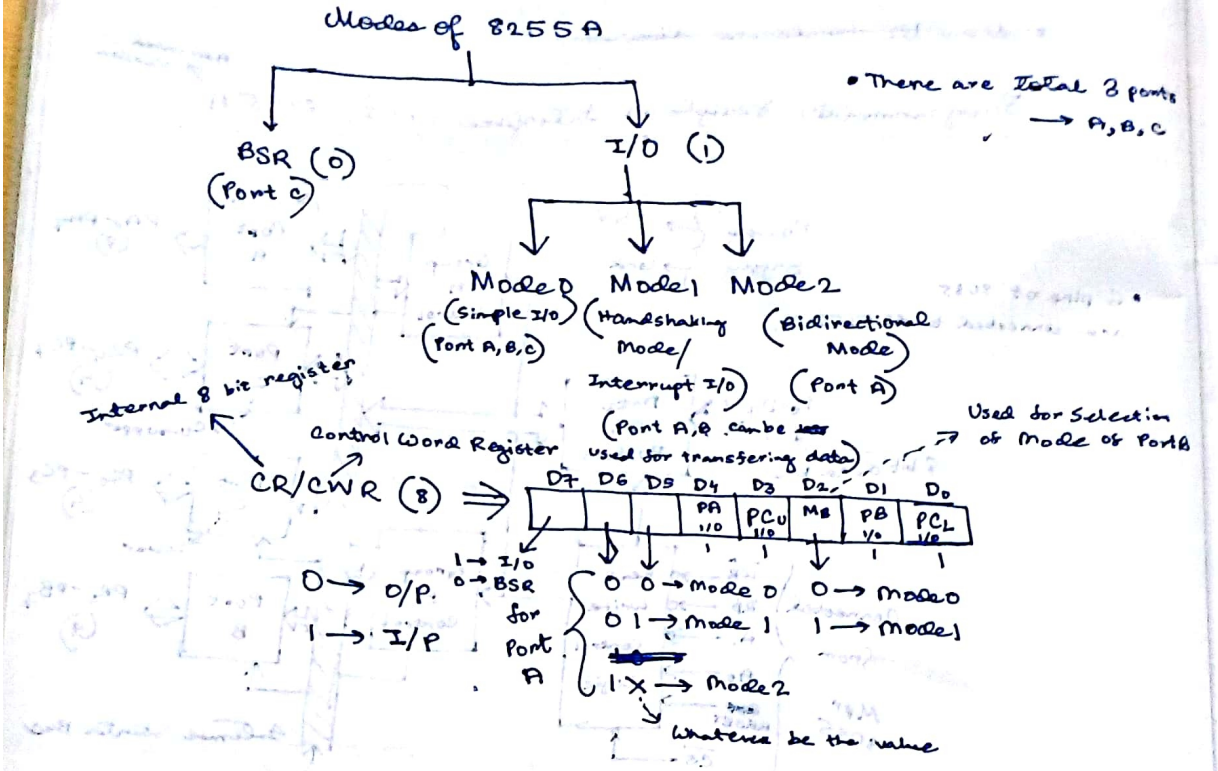
Block Diagram of PPI ~~8228~~ 8255 A



• 8255A has no memory



Modes of 8255A

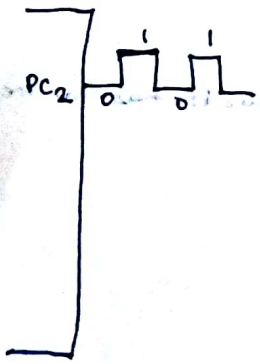


- There are total 3 ports → A, B, C
- For port initialization, D_7 is always 1.

BSR.CWR →

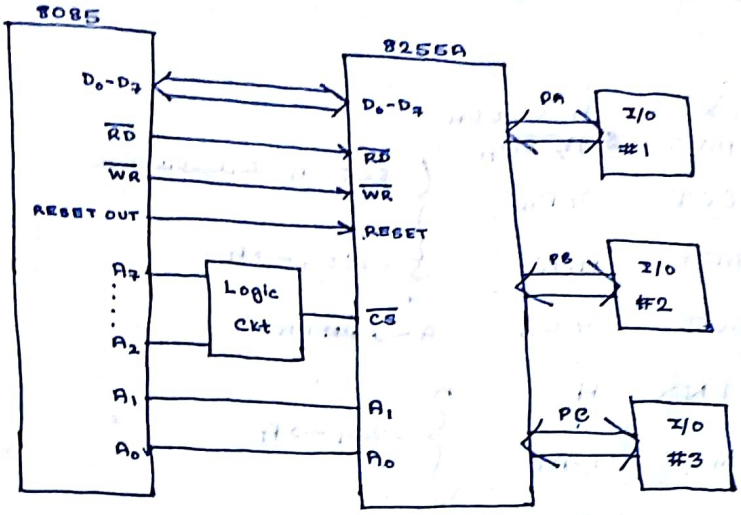
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	X	X	X	0	1	0	1/0

$PC_0 \rightarrow 000$
$PC_1 \rightarrow 001$
$PC_2 \rightarrow 010$
⋮
$PC_7 \rightarrow 111$




```
MVI A, —
Send to CWR
Delay prog.
MVI A, —
Send To CWR
Delay Prog
```

PPI 8255A

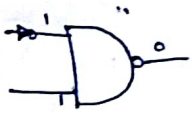


- 8 bit address used to identify I/O here.

A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
0	1	1	1	1	1	0	0	→ Port A ($7C_H$)
						0	1	→ Port B ($7D_H$)
						1	0	→ Port C ($7E_H$)
						1	1	→ CWR ($7F_H$)

To make $\overline{CS}$ = 0

Logic Ckt



CW → 1 0 0 1 0 0 0 0 = 90H
D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0

Mode 0

CWR → 1 | | PA I/O | PC_U I/O | . . | PB I/O | PC_L I/O

I/O | Mode of PA | Mode of PB

To Store data in 8 bit Memory Loc.

MVI A, 90H
STA XXYYH

Same (Stored data)

MVI A, 9
OUT FF

MVI A, 90H
OUT $7F_H$
IN $7C_H$
OUT $7D_H$
RST 1

Transfers data in the input port to the Acc.

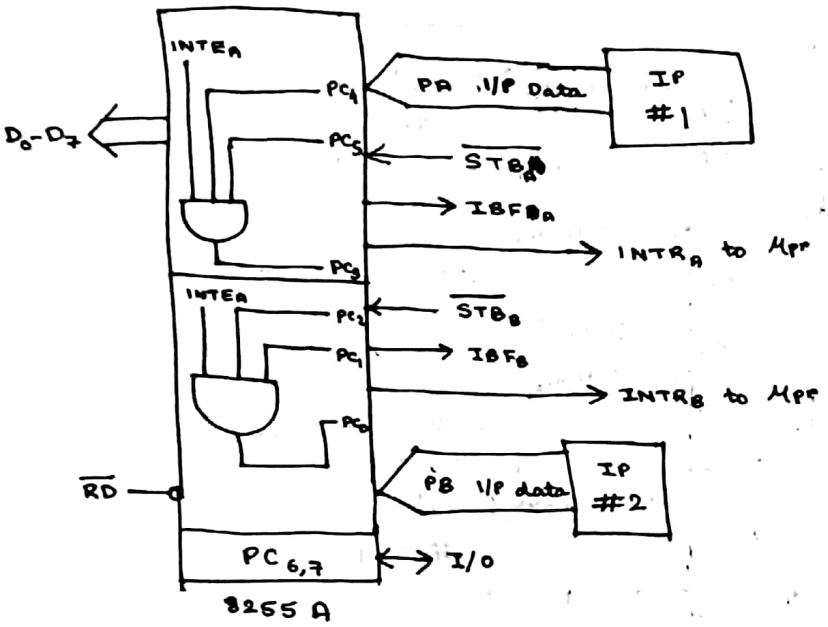
LDA ⇔ IN
↓
Load data in A

If all the ports are made O/P ports

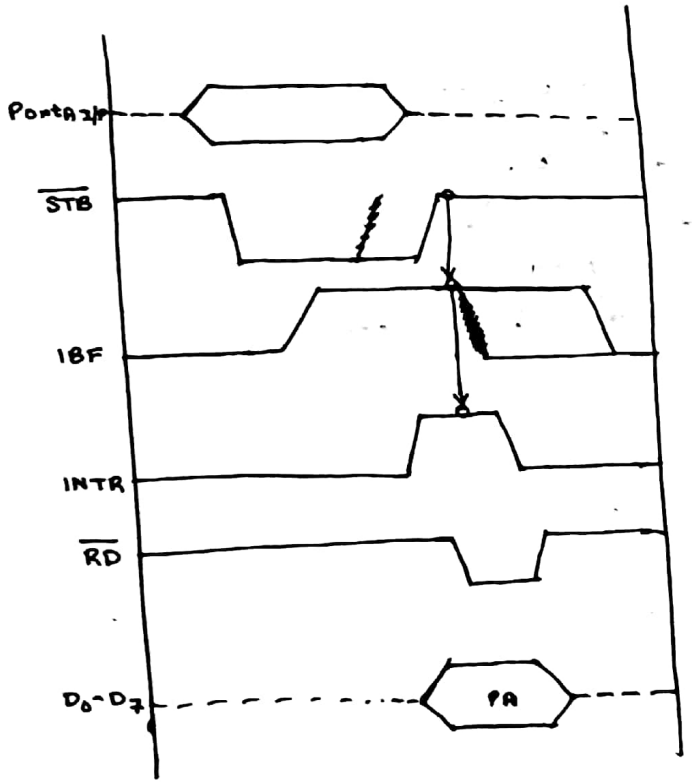
CW → 1 0 0 0 0 0 0 0 = 80H D_0-D_7

```
LXI   H, C200H
MVI   A, 80H      } 8255A Initialization
OUT   7FH         }
MOV   A, M        } C200H → A
OUT   7CH         A → PA
INX   H           } C201 → A
MOV   A, M        }
OUT   7DH         A → PB
INX   H           } C202 → A
MOV   A, M        }
OUT   7EH         A → PC
RST   1
```

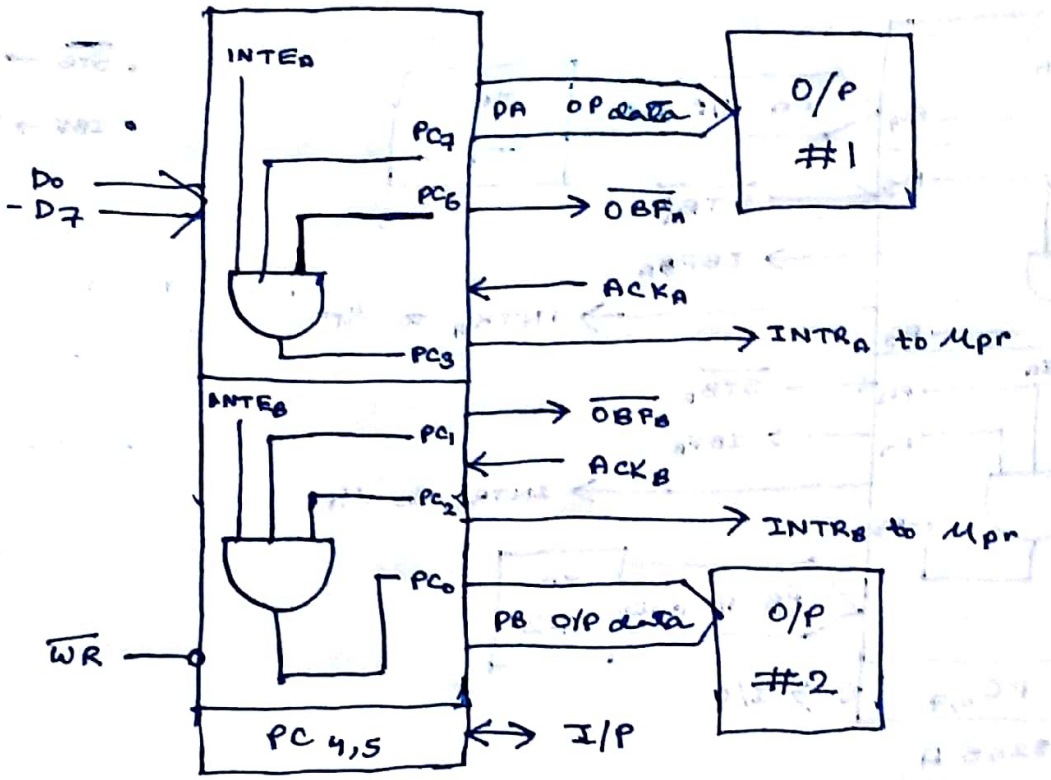
Mode 1 PA/PB (I/P)



- $\overline{STB}$ → Strobe Input
- IBF → Input Buffer

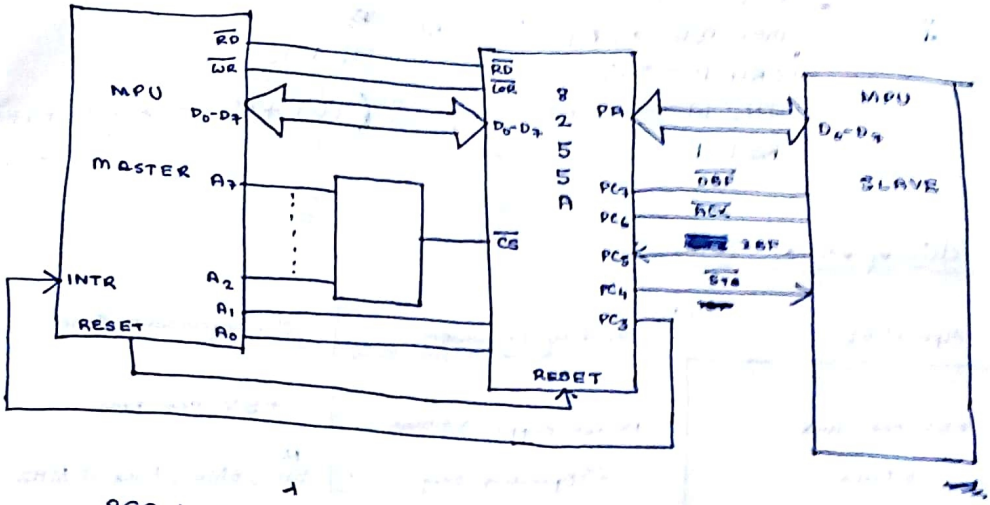


Mode 1 (O/P)

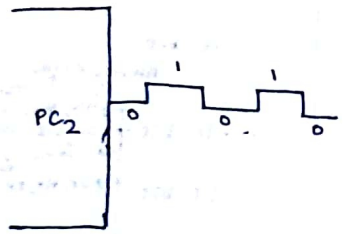


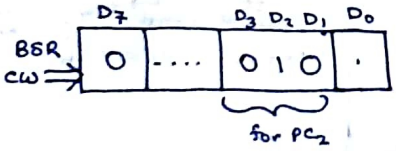
• OBF → O/P Buffer Full

Bidirectional Mode (Mode 2) of 8255 A



BSR Mode





$1\,0\,1\,0 \Rightarrow 08_H$

for PC_4 $\Rightarrow 09_H$

Main Program for eg. ..... via PC_4

```
LI: MVI  A, 08H
    OUT  FFH
    CALL DELAY
    MVI  A, 09H
    OUT  FFH
    CALL DELAY
    JMP  LI
```

DELAY

```
DELAY: MVI D, FFH (7T)
                XY
LI: DCR D  (4T)
    JNZ LI (10T)/(7T)
    RET
```

$7 + \{(4T + 10) \times 256 - 3T\}$

5 MHz

$T = \frac{1}{5} = 0.2\,\mu s$

65536

LXI D, FFFF$_H$

DCX D → 6T

MOV A,E → 4T

ORA D → 4T

JNZ L1 → 10/7T

RST 1

Total Delay

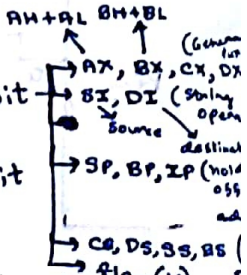
$= T_H + T_L$

$= (10T + 65536 * (6+4+4+ \ldots -3T$

$=$

Microprocessor 8086

Mpr 8085	Point of Comparison	Microprocessor 8086
+5V ref. Gnd	Power supply Voltage	+5V ref Gnd
3 MHz	Operating freq	5MHz, 8MHz, Max 10 MHz
40 pin	IC Package	40 pin
8 bit processor (ALU 8 bit)	ALU specification	16 bit
8 bit	Data Bus	16 bit
16 bit	Address Bus	20 bit
2^{16} = 64 KB (0000H – FFFFH)	Memory Limit	2^{20} = 1 MB (00000H – FFFFFH)



- BX → takes part before operand
- AX → Accumulator
- CS → Code Seg.
- DS → Data Seg.
- SS → Stack Seg
- ES → Extra Seg

Flag Reg (16)

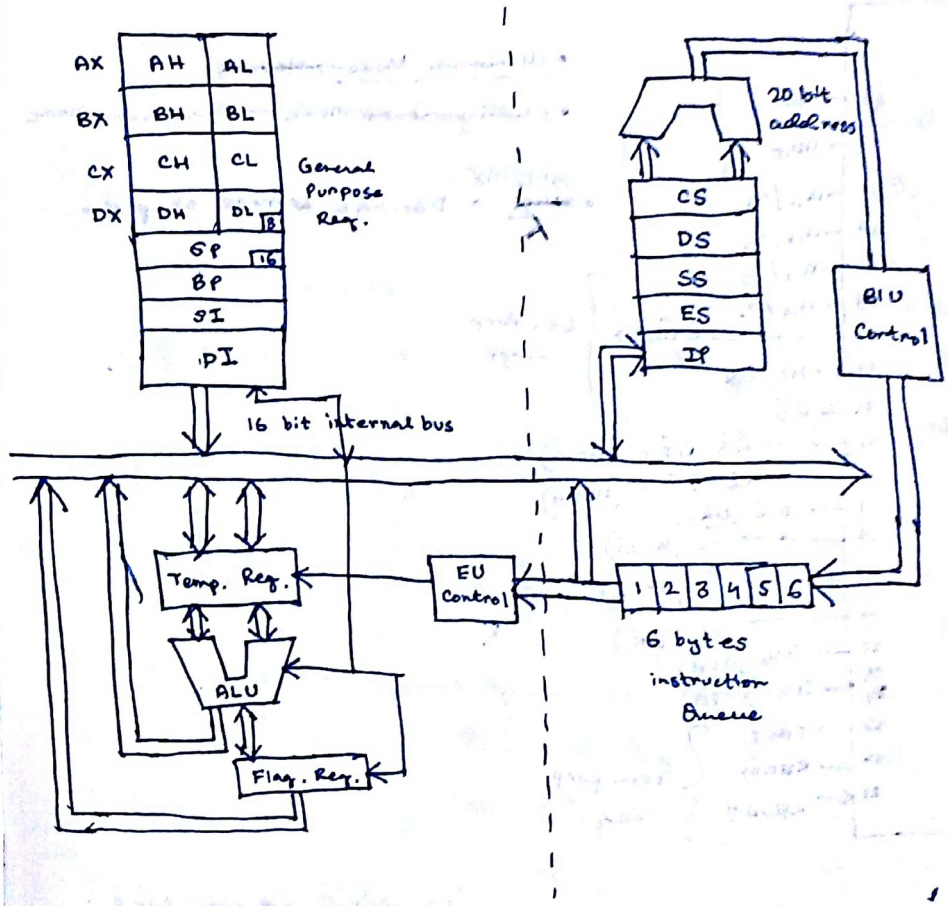
D_{15}	D_{14}	D_{13}	D_{12}	D_{11}	D_{10}	D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
X	X	X	X	O	D	I	T	S	Z	X	AC	X	P	X	CY

8000H to 7FFFH (-32768 to +32767)

O → Overflow
D → Direction (CLD, STD) — 80H–FFH to 00H–7FH (-128 to 127) — 2 bit
I → Interrupt (CLI, STI)
T → Trap?

7E + 02 ⇒ 80H

Internal Architecture of 8086



$2^{20} \Rightarrow$ 00000H ↓ FFFFFH (1 MB)

20400H →

Seg Reg	Default Offset Reg.
CS	IP
DS	BX, DI, SI
SS	SP
ES	DI

20 bit physical address = Segment Add × 10_d + Offset Address

= 2040 × 10 + Offset Address

= 20400H + 0000H

CS: 2000 H →

DS: 2040H →

IP: 3000 H

Physical Add = 2000 X 10 + 3000 H

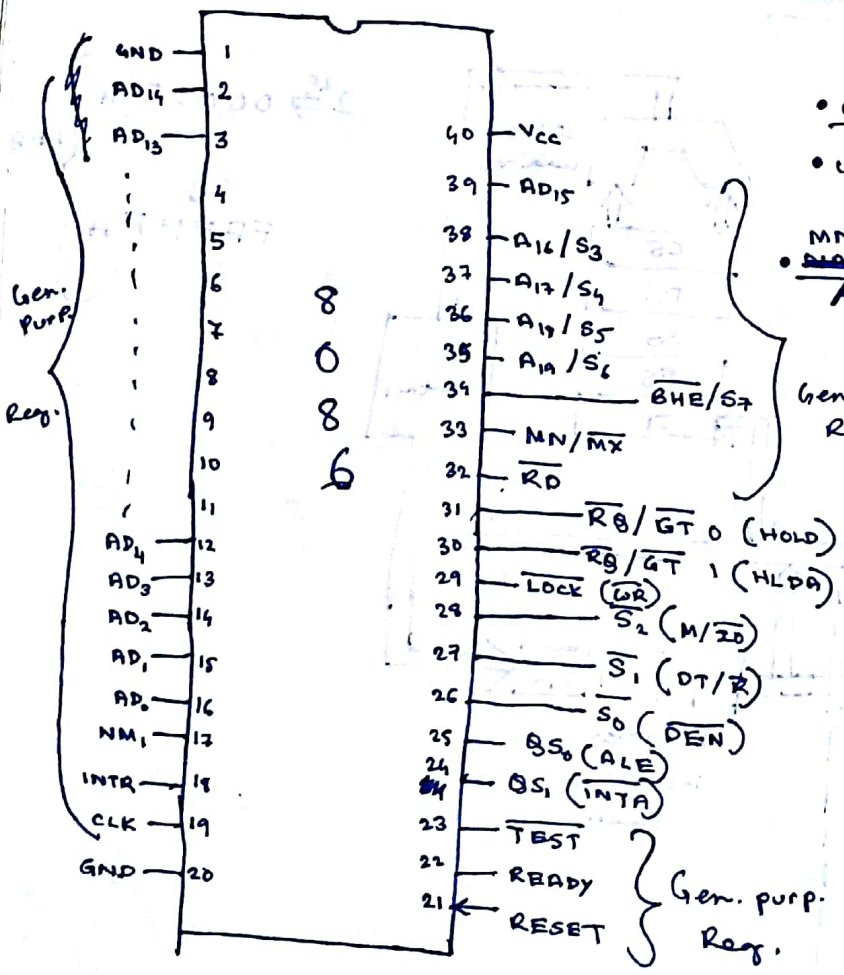
= 23000 H

03FF H → 1 KB

23 3FF H

Next Segment → 23400

PIN DIAGRAM



- Minimum Mode
- Multiprocessor Mode → Maximum Mode
- MN/$\overline{MX}$ → Decides mode of Mpr.

- NMI (Non-Maskable Interrupt) → Correlate with TRAP in 8085
- INTR (Interrupt Request):

There is an interrupt flag (IF) in the flag register. Only if it is enabled, then INTR can be set.

- If ALE (Address Latch Enable) is low, then AD_0-AD_{15} will be used as data bus. Else, they are used as address bus.

- S_3, S_4, S_5, S_6 → Status Signal Lines

S_4	S_3	
0	0	→ ES (Processor is interacting with ES)
0	1	→ SS
1	0	→ CS or none
1	1	→ DS

S_5 indicates the IF status
S_6 is always high
S_7 is normally high

- BHE (Bus high Enable): When enabled, then D_{8}-D_{15} is enabled (higher byte is enabled)

- RESET: When high, sets all the internal registers with 0; only CS: set to FFFF.
  - CS: Code Segment Register
  - Physical Address from which code segment is selected = FFFF0

- READY → Issues low signal to wait for a certain time; when high it indicates that the external device is ready to interact.

- $\overline{TEST}$ → Used by a specific instruction called wait. WAIT → NOP

When $\overline{TEST}$ is low, wait has no operation.
When $\overline{TEST}$ is high, wait makes the processor wait till the particular pin is set to low

- Via the HOLD pin, the DMA controller requests.
- M/$\overline{IO}$: High for Memory; Low for I/O
- DT/$\overline{R}$ (Data Transmit / Receipt)
- $\overline{DEN}$ (Data Enable)

$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	
0	0	0	→ $\overline{INTA}$
0	0	1	→ IO/Read
0	1	0	→ IO Write
0	1	1	→ Halt
1	0	0	→ Opcode Fetch
1	0	1	→ Memory Read
1	1	0	→ Memory Write
1	1	1	→ Passive (Inactive)

- QS_0, QS_1 → Helps to identify various instruction queue status

QS_1	QS_0	
0	0	→ Queue is idle
0	1	→ First byte is read
1	0	→ Queue is empty
1	1	→ Subsequent byte is read

- $\overline{RQ}/\overline{GT}$ 0 (Request/Grant)

Instructions of 8086

MOV BX, CX → • Transfers content from 16 bit Reg. to 16 bit Reg.
• No MVI, LXI etc in 8086

MOV DX, $23A5_H$

MOV AL, [BX] → • BX points to some memory location containing 8 bit data

MOV [SI], 2300_H

ADD BX, DX → BX = BX + DX

ADD CL, 05_H → 8 bit Addition ; CL ← CL + 05H

MUL

MUL ~~BX~~ BX ([AX] ← [AX] × [BX])

for 16 bit

([DX][AX] ← [AX] × [BX]) ← for 32 bit
(higher 16 bit, lower 16 bit)

([AH][AL] ← [AL] × [BL]) → for 8 bit
(for 16 bit: Not used, 8 bit)

DIV BL / DL / Immediate Data

8 bit / 16 bit / 32 bit ÷ 8 bit / 16 bit → Quotient: 8 bit / 16 bit; Remainder: 8 bit

AX / DL → AL/AX (8 bit) (16 bit); AH/DX (8 bit) (16 bit)

Block Shifting Operation:

```
MOV DS, 1000H        → Segment memory DS=1000H
MOV [SI], 2300H      → Offset memory location
MOV [DI], 2400H
MOV SB  → Byte of data, available on the source string to the destination string
MOV SW  → Sequential Word / 2 Sequential byte (2300, 2301 data transferred)

MOV CL, 05H   }
L1: MOV SW    }  If we run a loop, then total 10 bytes will get transfer
LOOP L1
```

```
MOV [SI], 2300H
MOV [DI], 2400H
MOV CL, 05H
L1: MOV SW
INC SI
INC SI
INC DI
INC DI
LOOP L1
```

OR

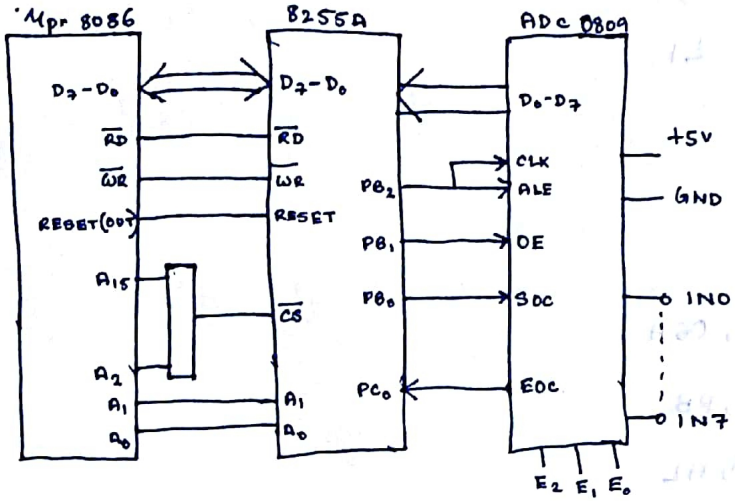
```
MOV [SI], 2300H
MOV [DI], 2400H
MOV CL, 05H / 0AH
REPE MOV SW / SB
CMP
```

To Match To Strings use

INT 3 / HALT

- DAA is also available in 8086 like in 8085.
- AAA is specially for 8086, used for ASCII
- ROR AL, n
  ROL AL, n
  } Rotate n times

Interfacing ADC0809 with Mpr 8086 through 8255A



- SOC → Start of Conversion
- EOC → End of Conversion
- OE → Output Enable
- ADC 0809 is a clock based device

Fig: Interfacing circuit of Mpr 8086 with ADC 0809 through PPI 8255A

- In place of port Address we will write → PA, PB, PC, CWR

CW → 1001 1001 ⇒ 99H

8255A Initialisation

MOV AL, 99H / MOV AX, 0099H

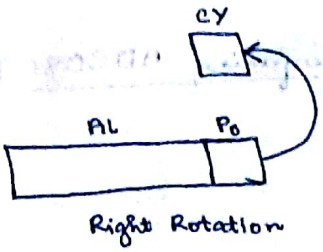
MOV DX, PA
↓
Holds Port Address

OUT DX, AL → The content of AL is shifted to the memory location pointed by DX.

	PB_2	PB_1	PB_0	
00000	1	0	1	⇒ 05H
00000	1	0	0	⇒ 04H
00000	1	1	0	⇒ 06H

SOC Generation

```
MOV AL, 05H
MOV DX, PB
OUT DX, AL
MOV AL, 04H
OUT DX, AL
```



Monitoring EOC

```
MOV  DX, PC
L1: IN   AL, DX
    ROR  AL, 1
    JNC  L1
```

Right Rotation

OE

```
MOV  AL, 06H
MOV  DX, PB
OUT  DX, AL
```

Receiving digital data

```
MOV  DX, PA
IN   AL, DX
```

Ending code

```
INT 3
```