EEE - 363

Microprocessor & Interfacing

Section - B

(Nipa Maam)

⇒ **Books**

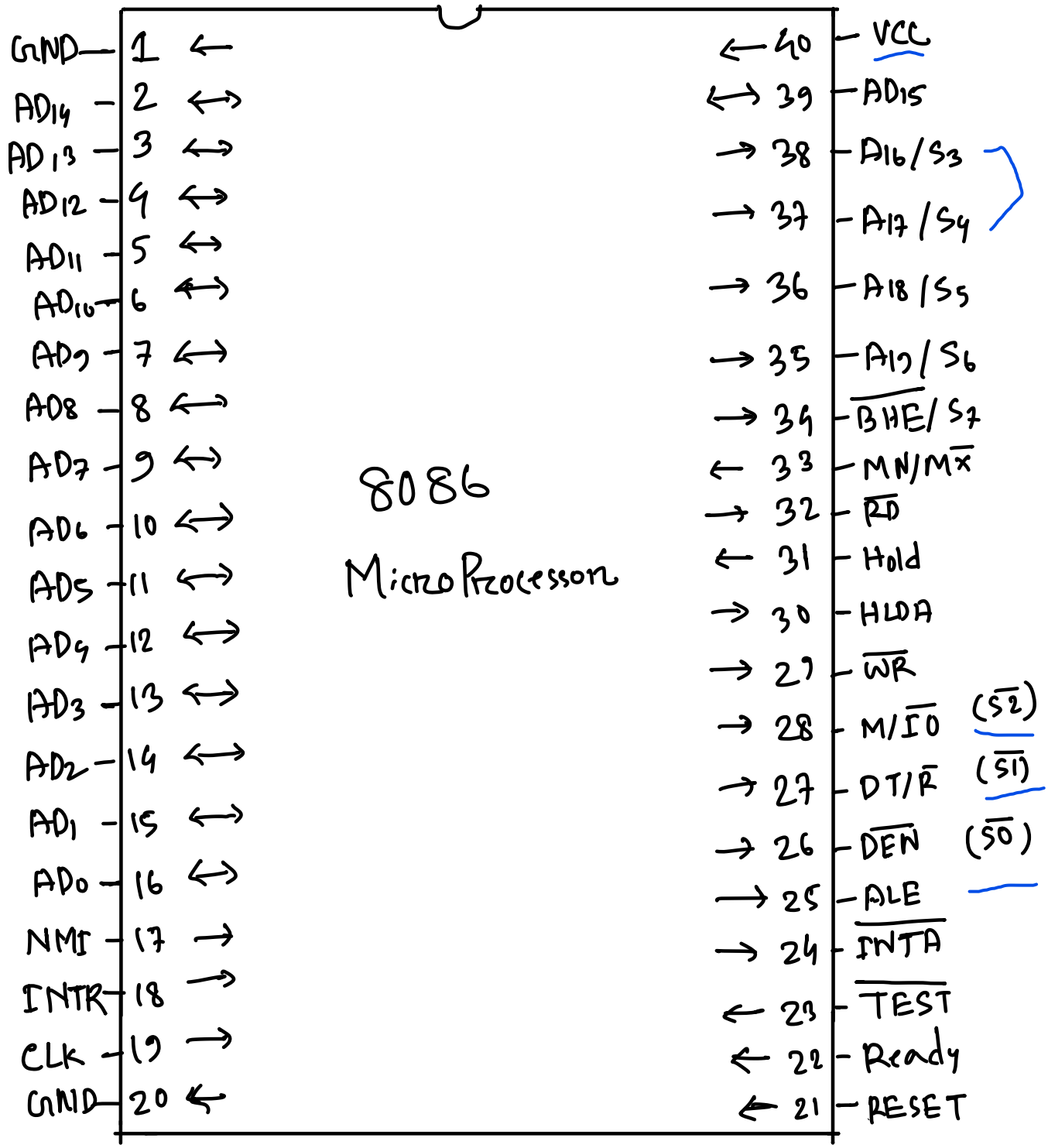
→ Barry B. Brey

→ Ramesh Gaonkar

→ MD Rafiquzzaman

8086 Hardware Specifications

Pin Diagram of 8086



→ 16 bit MP with a 16 bit Data bus.

→ Draws a Maximum supply current 360 mA (5V parts)

Pin out and Pin function:

VCC → Power supply provide +5V signal for MP

GND → Ground connection is the Return for Power Supply

AD_0-AD_{15} → Address / Data bus line composed the multiplexed address or data bus on 8086

ALE = 1 → Address bus → A_{15} - A_0

ALE = 0 → Data bus → D_{15} - D_0

A_{19}/S_6 ... A_{16}/S_3 → Address / status bus bits are multiplexed

Provide Address signal :- A_{19} - A_{16}

" Status bits :- S_6 — S_3

Status bit, S_6 → always zero (0)

Status bit, S_5 → Indicates the condition of IF Flag bits (interrupt flag)

Status bit S_4, S_3 → shows which segment is accessed during current bus cycle.

S_4	S_3	Function	
0	0	ES	Extra Segment
0	1	SS	Stack "
1	0	CS	Code "
1	1	DS	Data "

CLK → Clock Pin Provides time source for Microprocessor clock signal. Must have duty cycle - 33% < High - 1/3 sec, low - 2/3 sec

RESET → Reset input causes the MP to reset itself if the pin is held high for a minimum four clocking Period.

After the RESET instruction, execution begin at FFFF 0H and IF Flag is cleared.

~~BHE/S7~~ → Bus High Enable pin is used to enable the most significant Data bits (D_{15} - D_8) During Read ON. Write Operation → S_7 is always 1

NMI → Non Muskable Interrupt input similar to INTR except that NMI doesn't check if the IF flag is 0 on 1. If NMI activated, this interrupt input gives interrupt vector 2

$\overline{TEST}$ → Send Microprocessor in wait state. During this time, data comes from Co-microprocessor

$\overline{TEST}$ = 0 → Microprocessor waits (WAIT instruction works)

$\overline{TEST}$: 1 → WAIT Instruction wait for $\overline{TEST}$ to become 0.

Maximum Mode Pin:

To achieve maximum mode for use with external coprocessors, $MN/\overline{MX}$ must be connected to Ground.

So $MN/\overline{MX} = 0$

$\Rightarrow \overline{S_2}, \overline{S_1}, \overline{S_0}$ → Status bit indicate the function of current bus cycle

$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	function
0	0	0	Interrupt Acknowledge
0	0	1	I/O Read
0	1	0	I/O write
0	1	1	Halt
1	0	0	Opcode Fetch
1	0	1	Memory Read
1	1	0	Memory write
1	1	1	Passive

Minimum Mode Pin:

→ Obtained by connecting $MN/\overline{MX}$ to +5V

→ It must not be connected through pull-up register.

$\overline{INTA}$ → Interrupt Acknowledgement generated by MP response to Interrupt which causes Interrupt vector to be put on the data bus.

ALE → Address Latch Enable. shows that 8086 address/data bus (A_{15} - A_0) contains Address information.

$\overline{DEN}$ → Data Bus Enable activates external Data Bus Buffering.

$\overline{READY}$ → Ready Pin is related to $\overline{TEST}$. When TEST is activated, MP is not Ready to work, READY = 0

When, function of TEST finishes, MP ready to work, READY = 1

$\overline{TEST}$: 0 ⇒ READY : 0 (MP enter WAIT State)

$\overline{TEST}$: 1 ⇒ READY : 1 (MP operates)

$\overline{RD}$ → $\overline{RD}$ = 0, the MP Data bus is controlled by MP or I/O

$\overline{WR}$ → $\overline{WR}$ = 0, the MP outputs data to Memory or I/O

DT/$\overline{R}$ → DT/$\overline{R}$ = 1 → MP will transmitte data

DT/$\overline{R}$ = 0 → MP " Recive "

M/$\overline{IO}$ → Select Memory or I/O mode. This pin shows MP address bus contains either memory address or I/O port address.

HOLD → Hold input request a Direct Memory Access (DMA)

If Hold = 0 → MP executes software

Hold = 1 → MP stop " "

HLDA → Hold Acknowledgment shows MP enters the Hold state.

$\overline{SSO}$ → This signal is combined with M/$\overline{IO}$ and DT/$\overline{R}$ to decode the function of the current bus cycle.

Difference between minimum and maximum mode operation.

Minimum Mode	Maximum Mode
(I) Only 1 MP available	(I) More than 1 MP available
(II) ALE for the Latch given by 8086	(II) ALE given by different IC 8288
(III) $\overline{DEN}$, DT/$\overline{R}$, M/$\overline{IO}$, $\overline{RD}$, $\overline{WR}$ $\overline{INTA}$, Hold, HLDA control signal generated by MP itself	(III) generated by 8288
(IV) Simple but Slower	(IV) Complex but fast
(V) Performance lower	(V) Performance Higher

8284 A : Clock Generator

Basic Functions of clock Generator :-

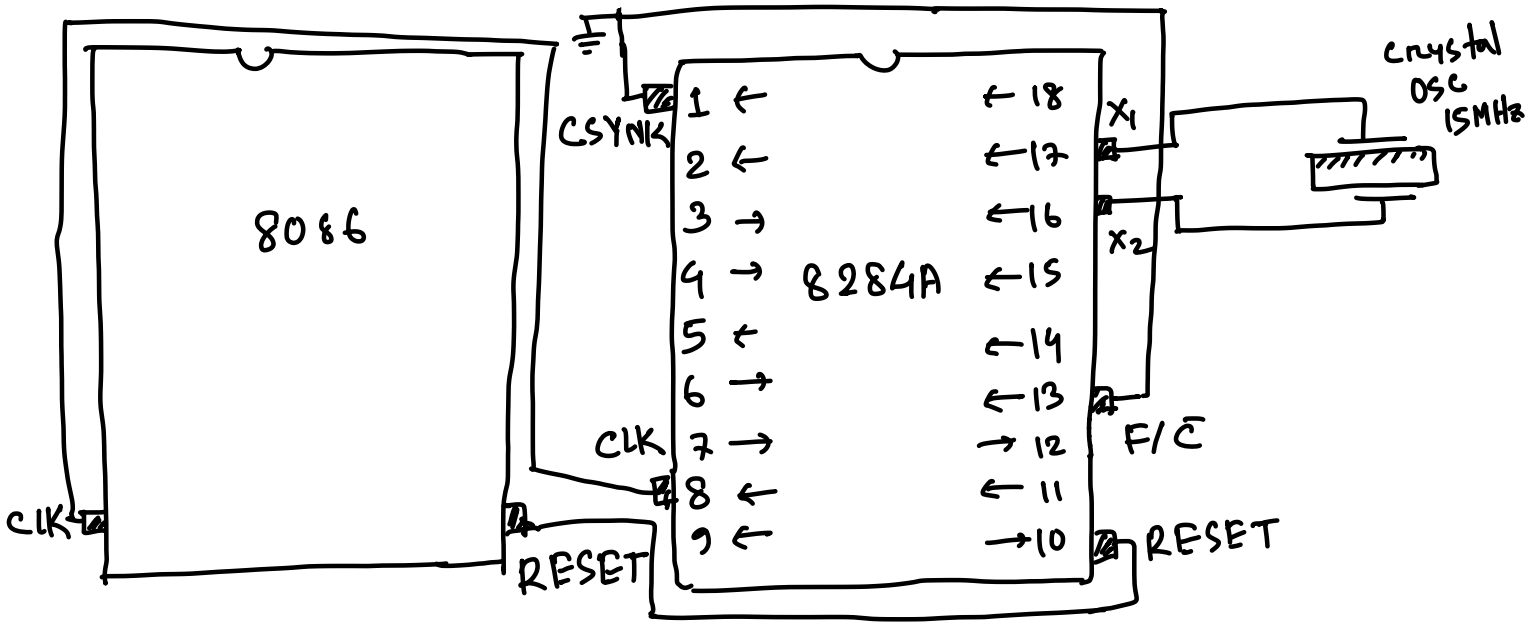
(I) Clock generation

(II) RESET Generation

(III) READY Generation

(IV) Peripheral Clock signal

Connection of 8284A and 8086



F/C̄ → Frequency / Crystal. It selects input chooses for the clocking source for the 8284A.

Clock Generation Process:

→ Crystal is connected to X_1 and X_2

→ XTAL OSC generates a square wave signal at crystal's frequency which feeds:

- An inverting buffer (output OSC) which is used to drive the EFI input of 8284A's
- A 2 to 1 MUX
  F/C̄ selects XTAL on EFI External Input.

→ The MUX drives a divided by 3 counter, which converts 15 MHz to 5 MHz clock signal.

→ This drives :

- The READY Flipflop (READY Synchronization)
- A second divided-by-2 counter which generates a 2.5 MHz CLK for peripheral component (PCLK)
- The RESET flip flop
- CLK which drives the 8086 CLK input

→ When, $\overline{ASYNC} = 0$ → Two stage synchronization

$\overline{ASYNC} = 1$ → One " "

RESET :

→ Negative edge trigger flipflop applies the RESET signal to 8086 on the falling edge

→ The 8086 samples the RESET pin on the Rising edge.

→ CSYNC :- used with Multiple Processors

→ Connect reset timing requires that the RESET input to the MP becomes a logic 1 No Later than 4 clocks after power Up and stay for at least 50us.

Bus Buffering and Latching

There are 3 buses :- (I) Address, (II) Data, (III) Control.

Address and Data bus are multiplexed due to Pin limitation of 8086

Why the Data bus are De-multiplexed / Why not leave the buses multiplexed.

⇒ →The memory and I/O require that the address remain valid and stable throughout a read or write operation.

→ If the buses are multiplexed, the change in address occurs at the memory and I/O, which cause them to read or write in the wrong location.

→ If more than 10 unit loads are attracted to bus Pin, the entire 8086 must be buffered.

→ Latches buffer for $A_0 - A_{15}$

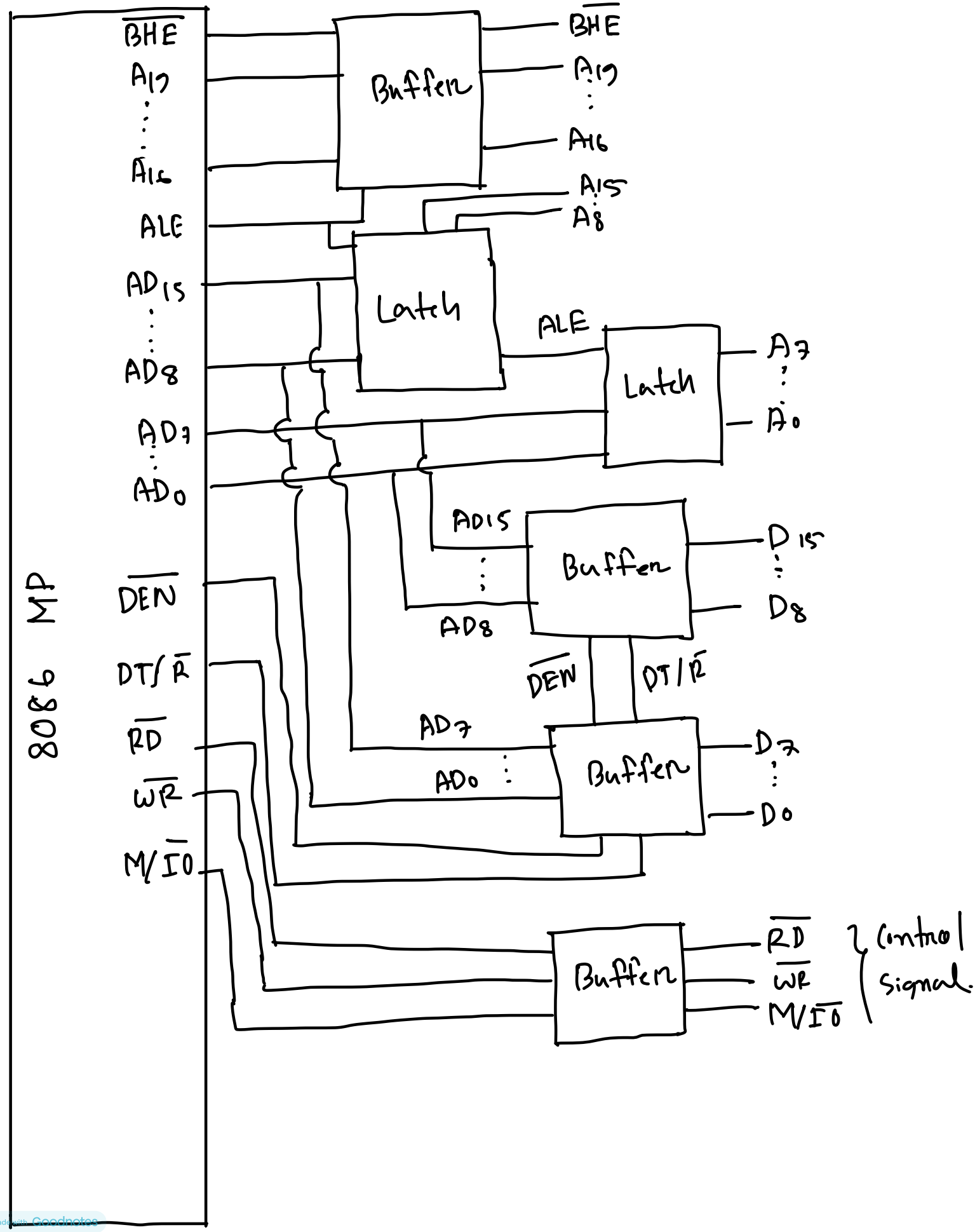
→ Control and $A_{16} - A_{19}$ + $\overline{BHE}$ buffer separately.

→ Data bus Buffering must be bidirectional.

Bus Buffering and Latching Structure in 8086

→ Address pin transfer through → Latch
→ Buffer

→ Data Pin " "



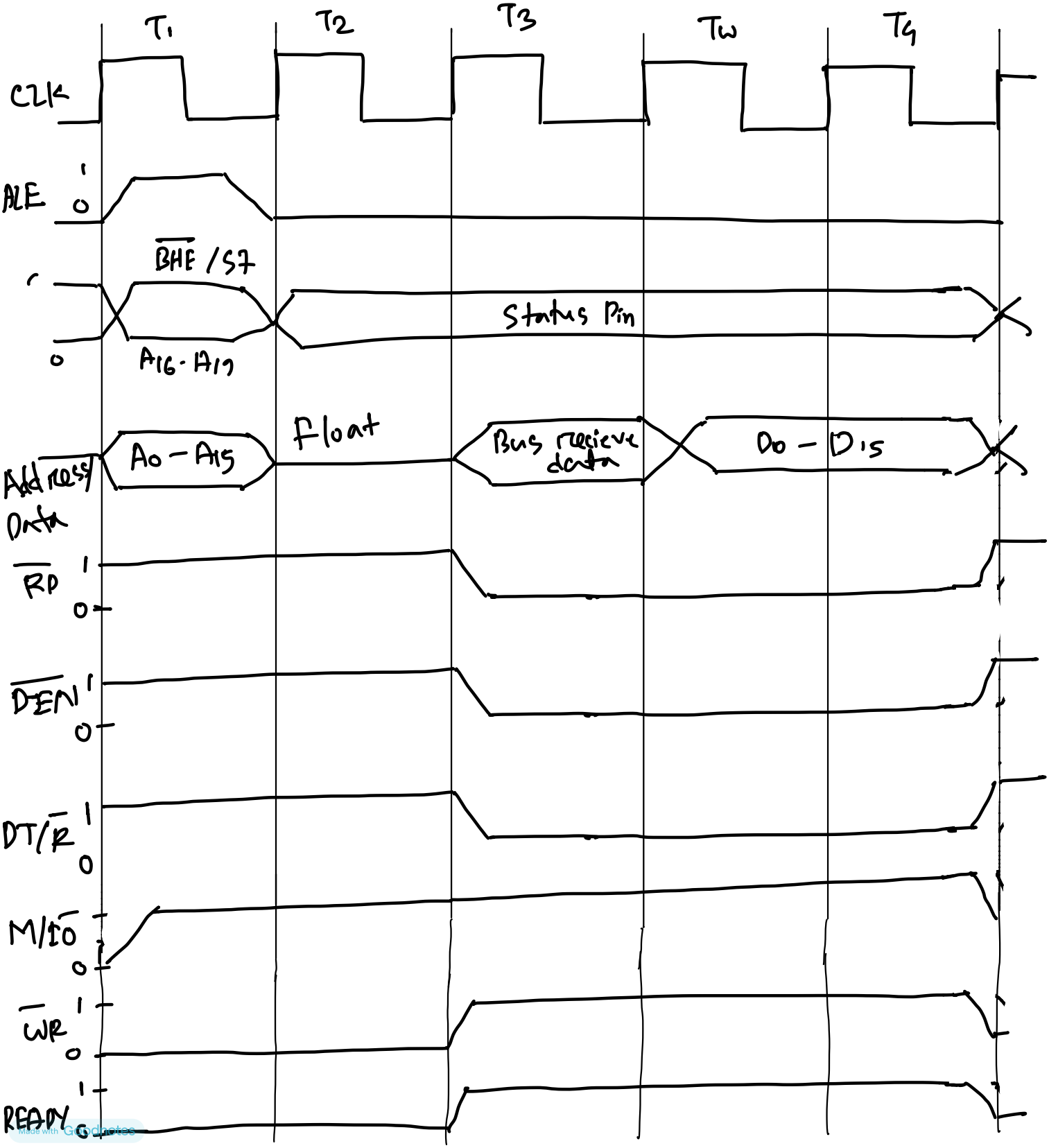
BUS Timing Diagram

(I) Read Timing Diagram (II) Write Timing Diagram

(I) Read Timing Diagram: MOV AX, [BX]

→ Read Data from memory location [BX] to AX

ALE : 1 ; M/$\overline{IO}$: 1 ; DT/$\overline{R}$: 0 ; $\overline{DEN}$: 0 ; $\overline{RD}$: 0 ; $\overline{WR}$: 1



During T_1:

→ During first clock Period, the address of the memory or input-output location is sent through the address bus and address/data bus.

→ Control Signal ALE, $M/\overline{IO}$, $DT/\overline{R}$ latch the address on the address bus and set the data transfer direction on data bus.

During T_2:

→ MP takes some time to setup the data to be read.

→ This stage defined as float stage

During T_3:

→ 8086 issues the read signal.

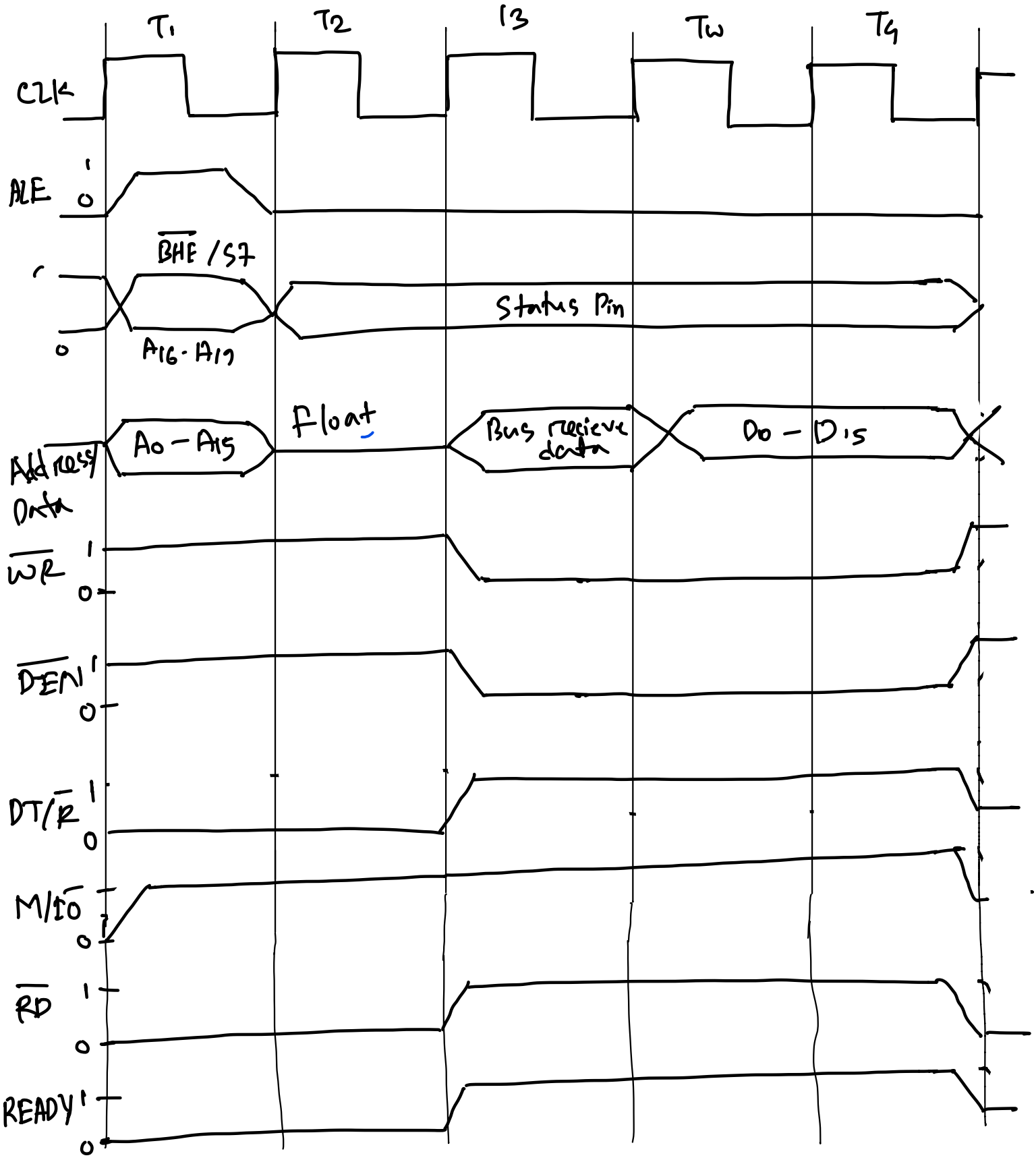
→ MP accepts the data to be read from MP or I/O device

During T_4:

→ All the signals are deactivated for the Preparation of next bus cycle.

(11) Write Bus diagram:

MOV [DX], AX. → Memory Write.



During T1:

→ During first clock Period, the address of the memory on input-output location is sent through the address bus and address/data bus.

→ Control Signal ALE, M/$\overline{IO}$, DT/$\overline{R}$ Latch the addresss on the address bus and set the data transfer direction on data bus.

During T2:

→ 8086 issues some time to write signal. Float

⇒ Data will be written to or recieved by Data bus

→ Memory on I/O Device begin to write.

During T3:

→ clocking Period is provided to allow the memory time to access data.

During T4:

→ D_0-D_{15} data Buses Write data to the memory/I/O device

→. All the buses are deactivated for preparation of Next cycle

Problem: Write down the status of the control Pin of 8086 MP for following Instruction, MOV AX, [BX]

⇒ Here, MOV AX, [BX]

This indicates Read Operation. So the Status of Control Pins are:-

$ALE = 1$ $\quad DT/\overline{R} = 0$

$\overline{DEN} = 0$ $\quad \overline{RD} = 0$ → Read operation

$M/\overline{IO} = 1$ $\quad \overline{WR} = 1$ → Because Memory operation

Data comes from Memory to Register

Problem: Write down the status of the control Pin of 8086 MP for following Instruction, MOV [AX], BX

⇒ Here, MOV [AX], BX

This indicates Write Operation. So the Status of Control Pins are:-

$ALE = 1$ $\quad DT/\overline{R} = 1$

$\overline{DEN} = 0$ $\quad \overline{RD} = 1$ → Write operation.

$M/\overline{IO} = 1$ $\quad \overline{WR} = 0$

Problem: What is the control Pin Status for OUT DX, AX

⇒ OUT DX, AX is a write operation.

So, $ALE = 1$ $\quad DT/\overline{R} = 1$

$\overline{DEN} = 0$ $\quad \overline{RD} = 1$

$M/\overline{IO} = 0$ $\quad \overline{WR} = 0$

↳ Input Output operation

Problem: What is the control Pin status for IN AX, DX

⇒ IN AX, DX is a Read operation.

So, ALE = 1 $DT/\overline{R} = 0$

$\overline{DEN} = 0$ $\overline{RD} = 0$

$M/\overline{IO} = 0$ $\overline{WR} = 1$

↳ Input Output operation

* MOV AX, BX → Not Read, Not Write.

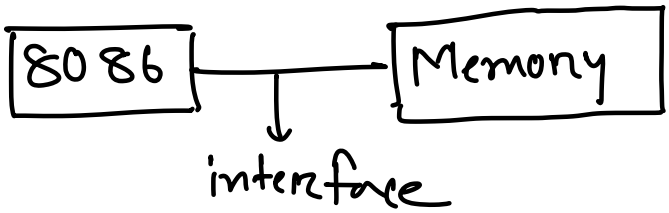
So control Pins are inactive.

Q Difference b/n Maskable Interrupt	vs	Non-maskable Interrupt
1. Can be delayed/rejected		1. Can not be delayed/rejected
2. It will interrupt the processor if it's enabled		2. It will always interrupt the processor
3. Priority 2nd		3. Priority 1st
4. Most of them automatically disable after an interrupt has occured		4. There is no software mechanism to prevent the processor being interrupted by Non-maskable Interrupt
5. Higher Response time		5. It has very Low response time

Differentiate between Buffer and Latch:

Buffer	Latch
i.) Buffer is bidirectional or unidirectional.	i.) Latch is unidirectional.
ii.) Used with data.	ii.) Used with address.
iii.) Sends data quickly to the microprocessor	iii.) Latch keeps the stack until next clock comes.

Memory Mapping Interfacing



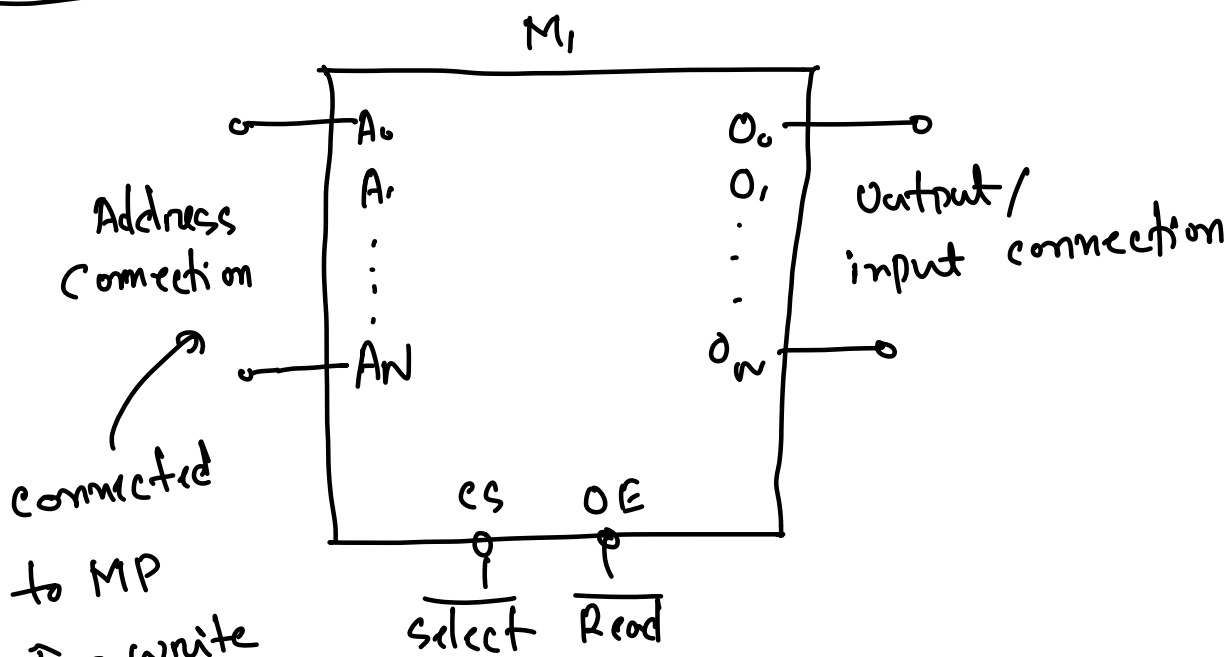
Memory → ROM
RAM.

ROM → System Software, Permanent Data

RAM → Temporary Data, Application Software

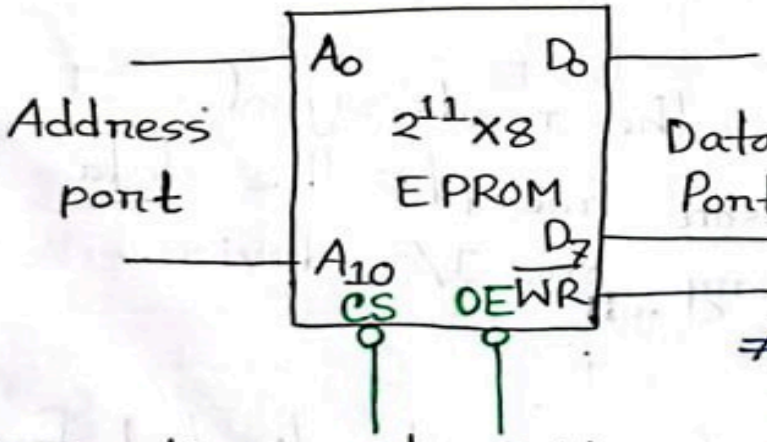
4 Common types of memory : (I) ROM
(II) EEPROM (Flash Memory)
(III) SRAM
(IV) DRAM

Memory Pin Connection :



Memory Size = 2^N → Address Line.

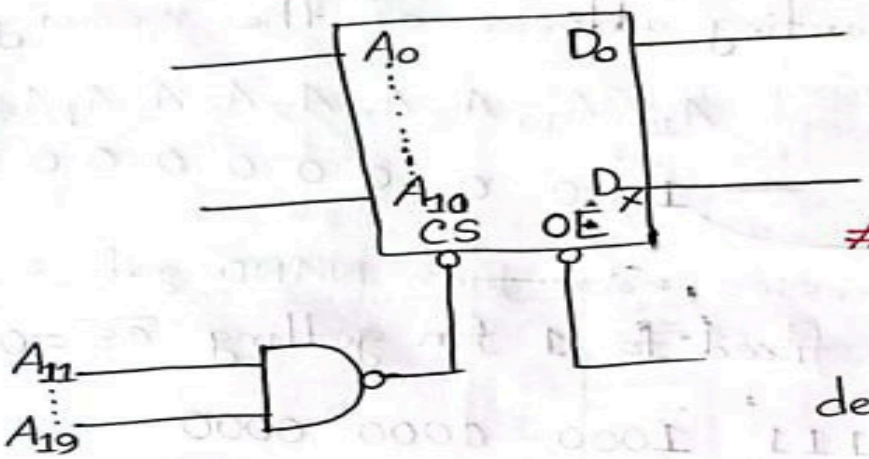
Example: A $2^{11} \times 8$ EPROM:



We know that- 8086 Microprocessor has 20 bit address bus.

But the $2^{11} \times 8$ EPROM (Memory Device) has 11 bit address bus.

So, the rest of the 9-bit address line is provided through a 3-to-8 line decoder or a NAND gate.



To generate more than one address by $A_{11} \ldots\ldots A_{19}$, decoder is used.

Problem For the given memory device-

(i.) Find the size of memory

(ii.) Starting address of memory

(iii.) Last address of memory.

$\Rightarrow$

(i) We know, Size of memory, $= 2^n$.

Here, $n = 11$ So, size of memory: 2^{11}

$= 2048$.

(II) Starting address of memory:

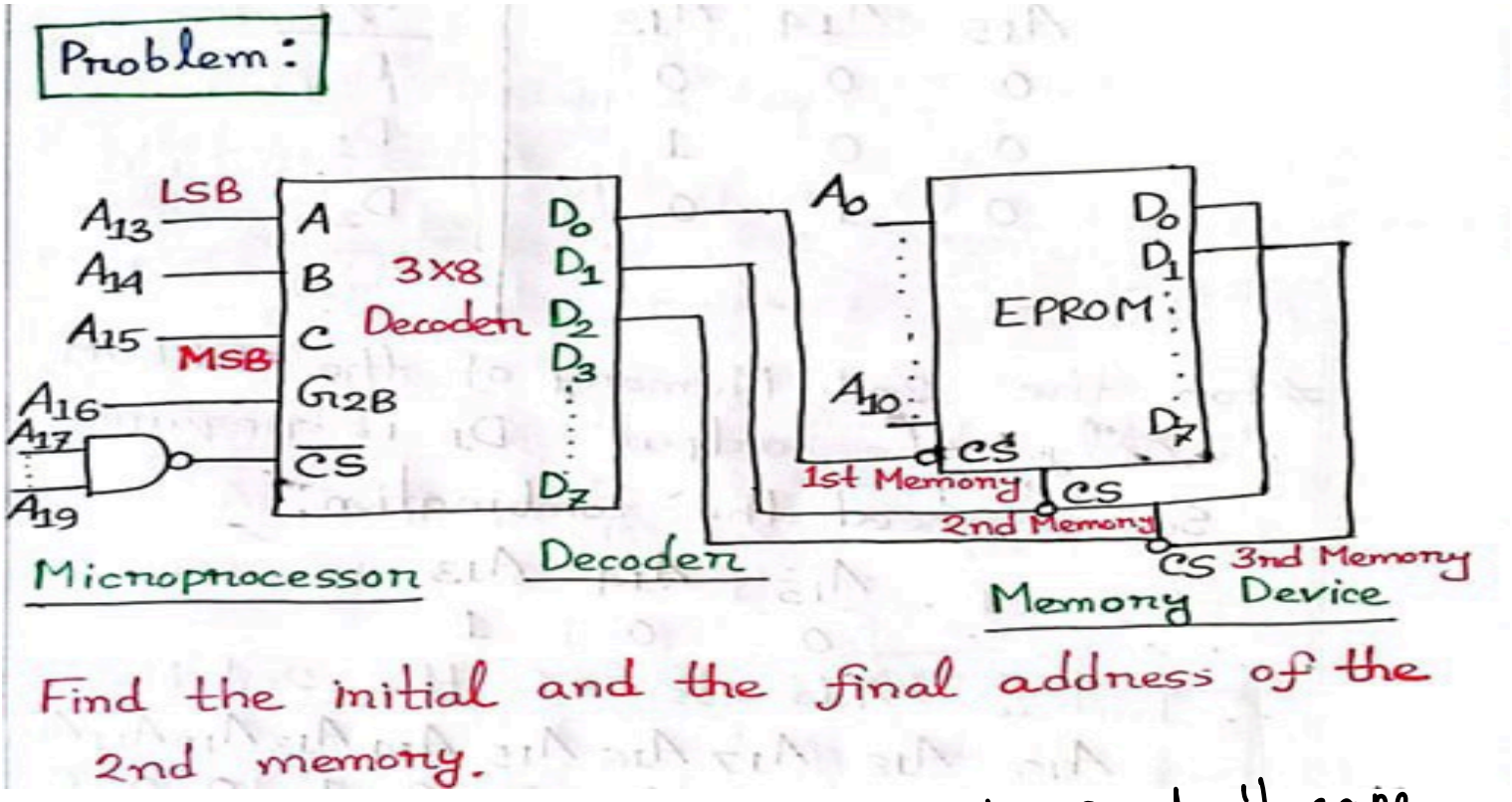
A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
F				F				8				0				0			

∴ Starting address = FF800H

(III) The last address

A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

∴ last address = FFFFFH



Find the initial and the final address of the 2nd memory.

⇒ Here, A_{11}, A_{12} are not present. So don't care.

Size of 2nd memory $= 2^n = 2^{11} = 2048 = 800\,H$

$n = 11$

Now, Here, for 2nd Memory, input for $\overline{CS}$ is D_1

From, Decoder,

MSB A_{15}	A_{14}	LSB A_{13}	O/P
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
-	-	-	-

So, the combination for 2nd memory, A_{15} A_{14} A_{13} = 0 0 1

Initial Address of 2nd Memory,

A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

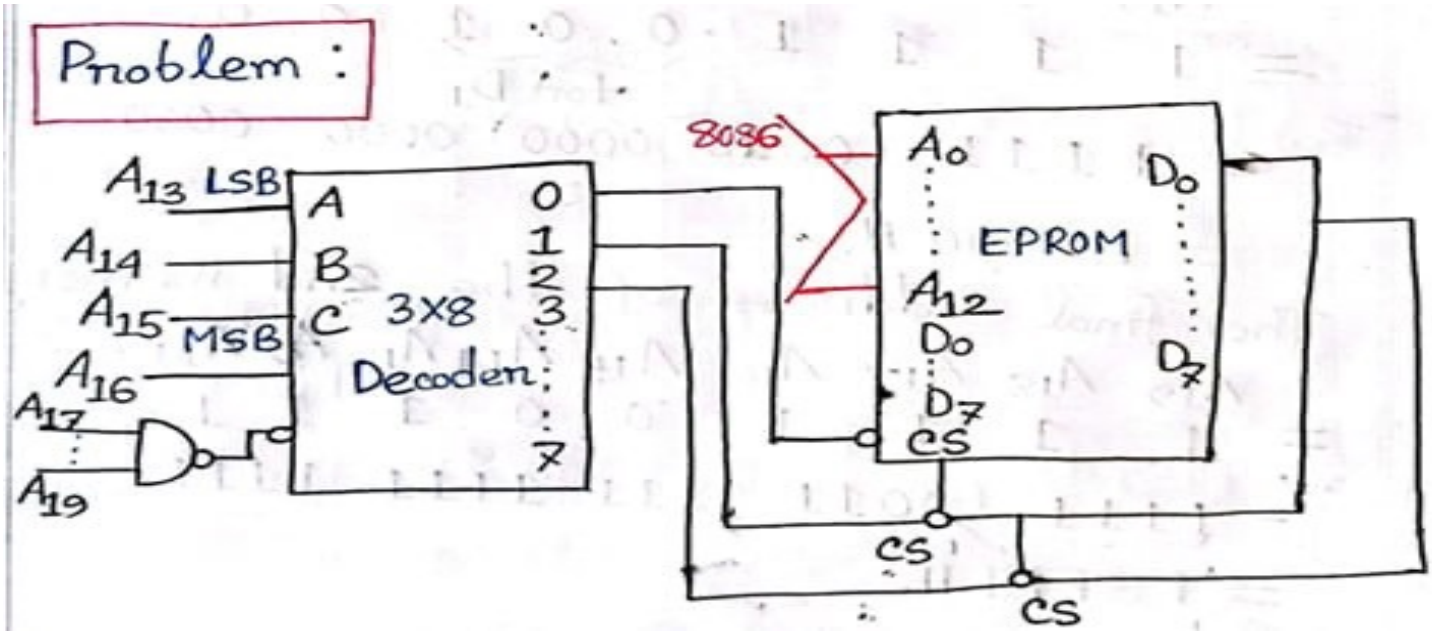
∴ initial Address = F2000H

Final Address of 2nd Memory

A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
1	1	1	1	0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1

∴ final address = F27FF H

Problem :



Question - 01: Find the initial and the final address of 2nd Memory.

Question - 02: Write an instruction to read 2-Byte data from EPROM and write into DX register (from the initial address of 2nd memory 2000H)

Question - 03: What is the size of the memory?

⇒ Here, D_1 goes to 2nd Memory,

So, combination is. A_{15} A_{14} A_{13}
0 0 1

∴ Initial Memory address for 2nd Memory,

A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

= F2000H

Final Memory address

A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1

= F3FFFH

(ii) Here initial physical address: F2000H

= 10X F000H + 2000H

∴ Segment Address = F000H
OFFSET Address = 2000H

Program:

```
MOV AX, F000H
MOV DS, AX
MOV SI, 2000H
MOV DX, [SI]
```

↑ 2 byte data transferred to DX

[For 1 byte transfer alternative

MOV DL, [BX]/[SI]

MOV DH, [BX]/[SI]]

(11) Items, $n = 13$. ∴ Size of 2nd memory : 2^{13}

∴ total Size of memory : 3×2^{13}

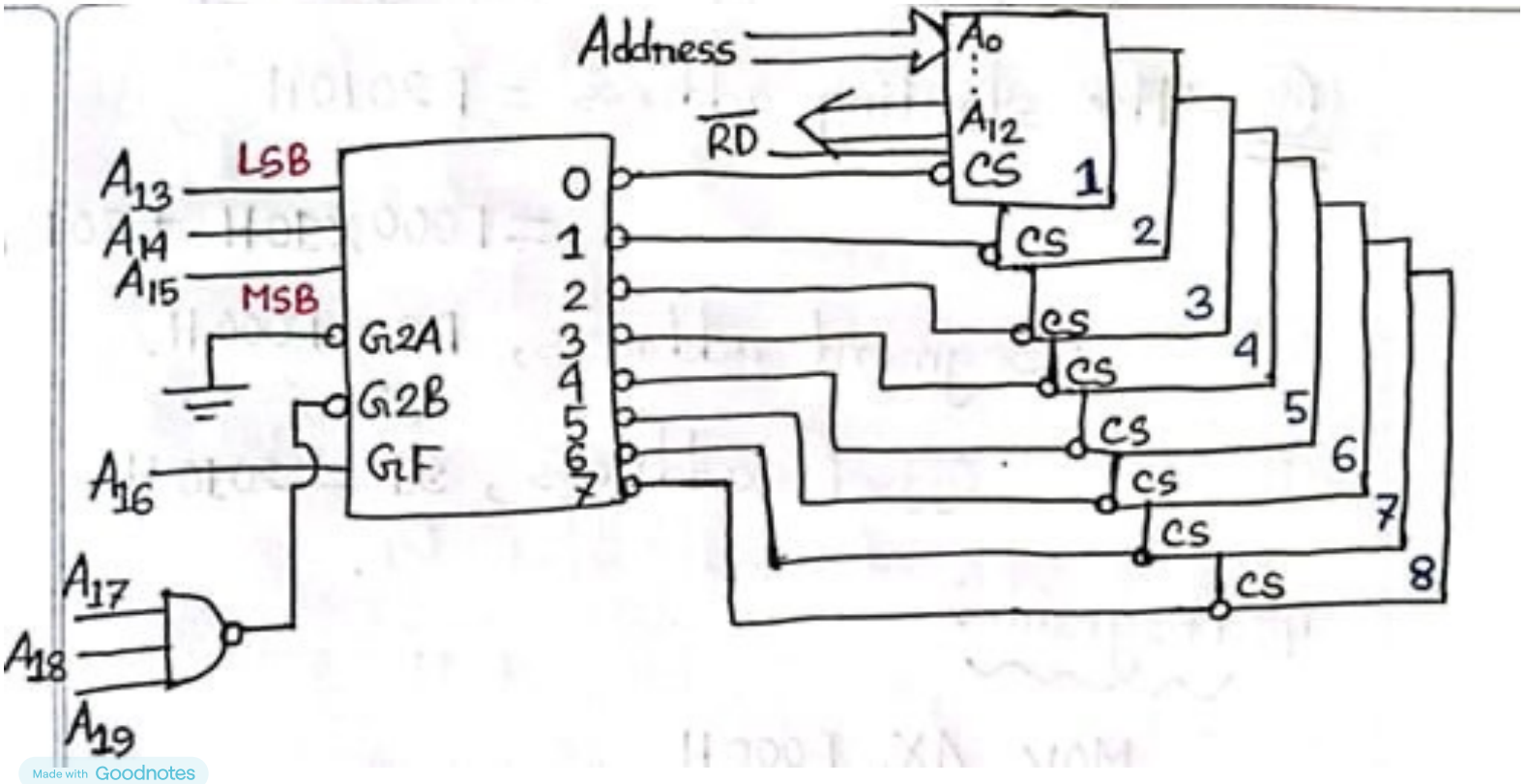
Problem: From the following figure determine

(i.) Initial and final address of the 2nd section.

(ii.) Size of the Memory device.

(iii.) Write an instruction to load a final byte in DL and next 2 byte in DX. The starting address = F2010H.

(iv.) For initial address from the circuit, record 1 byte data.



⇒(i) Here, For the second section, A_{15} A_{14} A_{13}
0 0 1

∴ Initial Address of 2nd Section.

A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

= F2000H

Final Address of 2nd section.

A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1

= F3FFFH

(ii) there are, 13 address port; $n = 13$

So, Size of memory = $2^n = 2^{13}$

(iii) For the starting address = F2010H

= 10 × F000H + 2010H

∴ Segment address = F000H

offset address = 2010H

Program:

```
MOV AX, F000H
MOV DS, AX
MOV SI, 2010H
MOV DL, [SI]
INC SI
MOV DX, [SI]
```

(iv) The initial address : F2000H

: 10 × F000H + 2000H

∴ Segment Address : F000H.

offset Address : 2000H

Program:

```
MOV AX, F000H
MOV DS, AX
MOV SI, 2000H
MOV DX, [SI]
```

I/O Interfacing

IN → Read information from an I/O device to AX

Out → Write " to an " " from AX.

→ I/O Address stored in register DX → Variable address.

→ 8 bit stored in ROM → fix address.

Ex: (i) IN AL, P8 (ii) IN AX, P8, (iii) IN AL, DX

(iv) OUT P8, AL (v) OUT DX, AX

→ Whenever data are transferred by using the IN and OUT instruction, the i/o address after called a port address appear on address line.

8 bit port Number = A_7 — A_0

16 " " " : A_{15} — A_0

There are two method of I/O interfacing :-

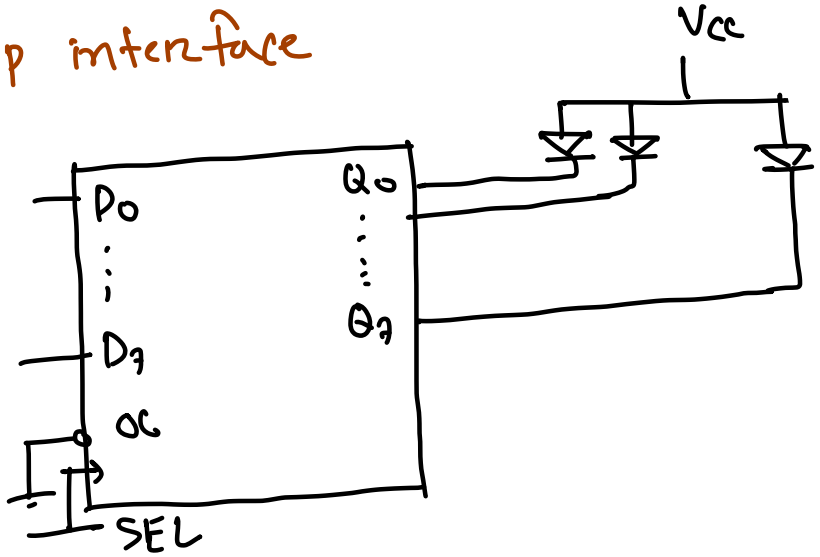
(i) Isolated I/O

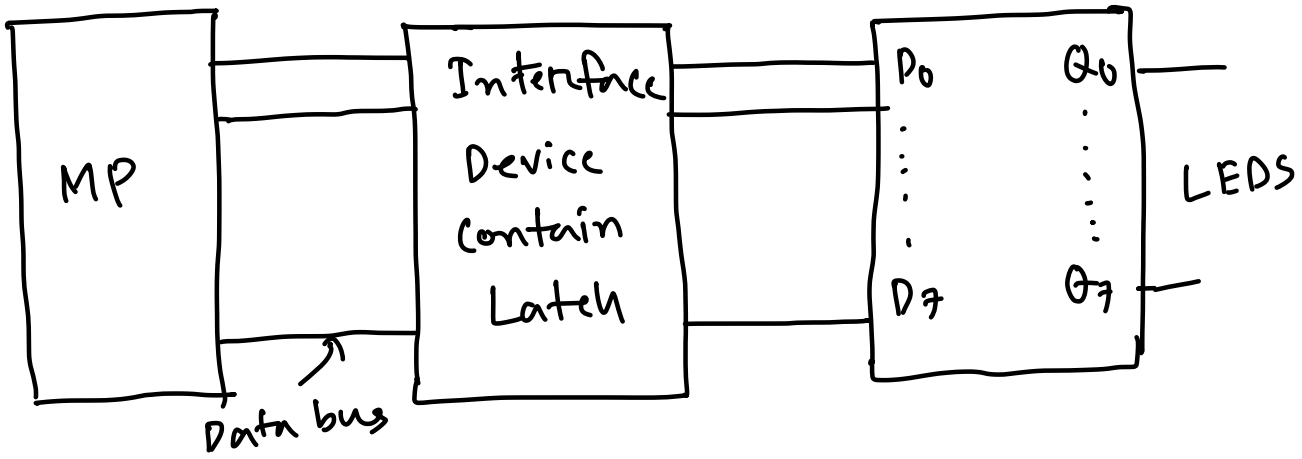
(ii) Memory mapped I/O

Difference between Isolated I/O and Memory-Mapped I/O

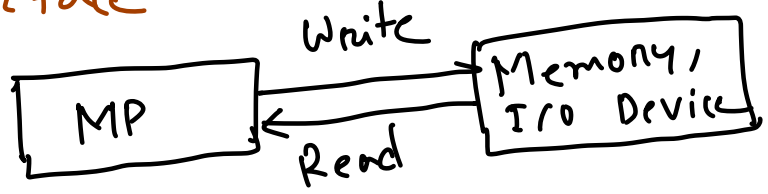
Isolated I/O	Memory Mapped I/O
(I) Use only IN, INS, OUT, OUTS instructions to transfer data.	(I) Doesn't use any instruction that transfer data between MP and memory.
(II) It has separate control signal $\overline{IOWC}$, $\overline{IORC}$	(II) $\overline{IORC}$, $\overline{IOWC}$ has no function here
(III) I/O location are isolated from memory location	(III) Memory-Mapped device is treated as memory location in memory map
(IV) PC use this	(IV) PC doesn't use memory mapped I/O.
(V) Expendable	(V) Non expendable

Basic O/P interface

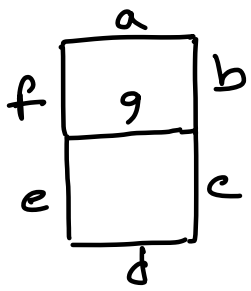




Input Interface

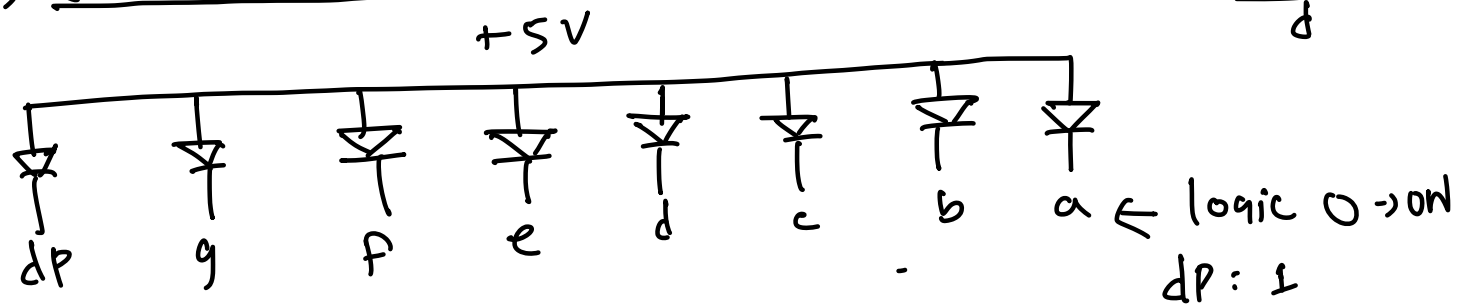


7 Segment Display with MP

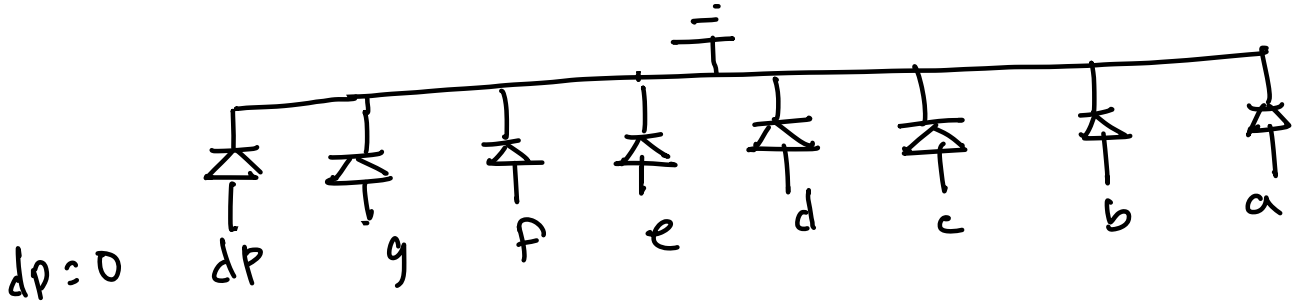


Two Types Connection :-

(1) Common Anode :



(11) Common Cathod :0V



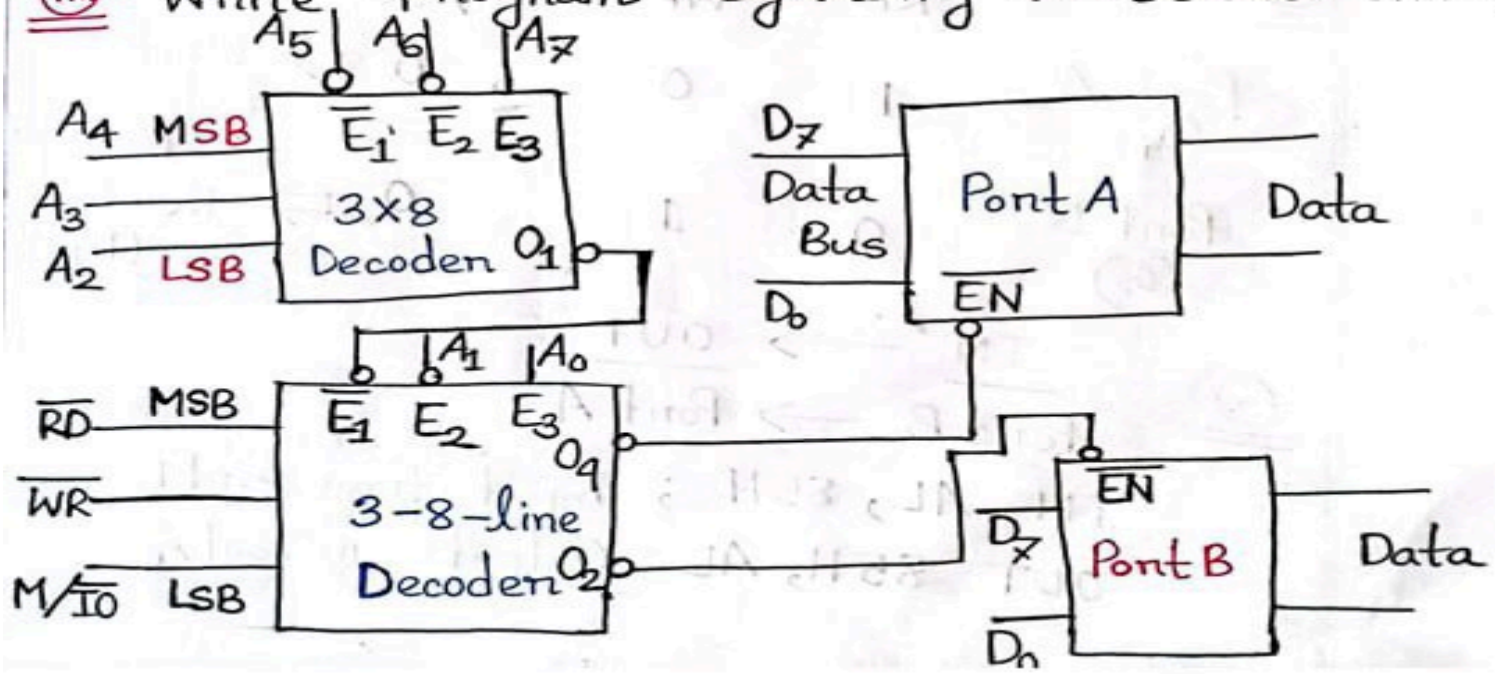
logic 1 → turn ON

Problem:

(i) What are the addresses of port A and portB?

(ii) Identify Port A and Port B as input or output Port

(iii) Write progrom regarding the decoden circuit.



(I) Here, $A_{19} - A_8$ is don't care

O_4 and O_2 both depend on O_1.

and O_1 depend on. A_4, A_3, A_2 which is

A_4	A_3	A_2
0	0	1

So, Port Address,

A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
1	0	0	0	0	1	0	1	= 85 H

Port address for both Port A and Port B Same.

∴ Port A : 85 H ; Port B = 85 H.

(II) Output O_4 goes to enablen of Port A

" O_2 " " " " Port B

and, O_4, O_2 depend on $\overline{RD}$, $\overline{WR}$, $M/\overline{IO}$

So,

	$\overline{RD}$	$\overline{WR}$	$M/\overline{IO}$	
Port A (04) →	1	0	0	⇒ which indicates write
Port B (02) →	0	1	0	⇒ " " read.

∴ Port A → Output port ; Port B → input Port.

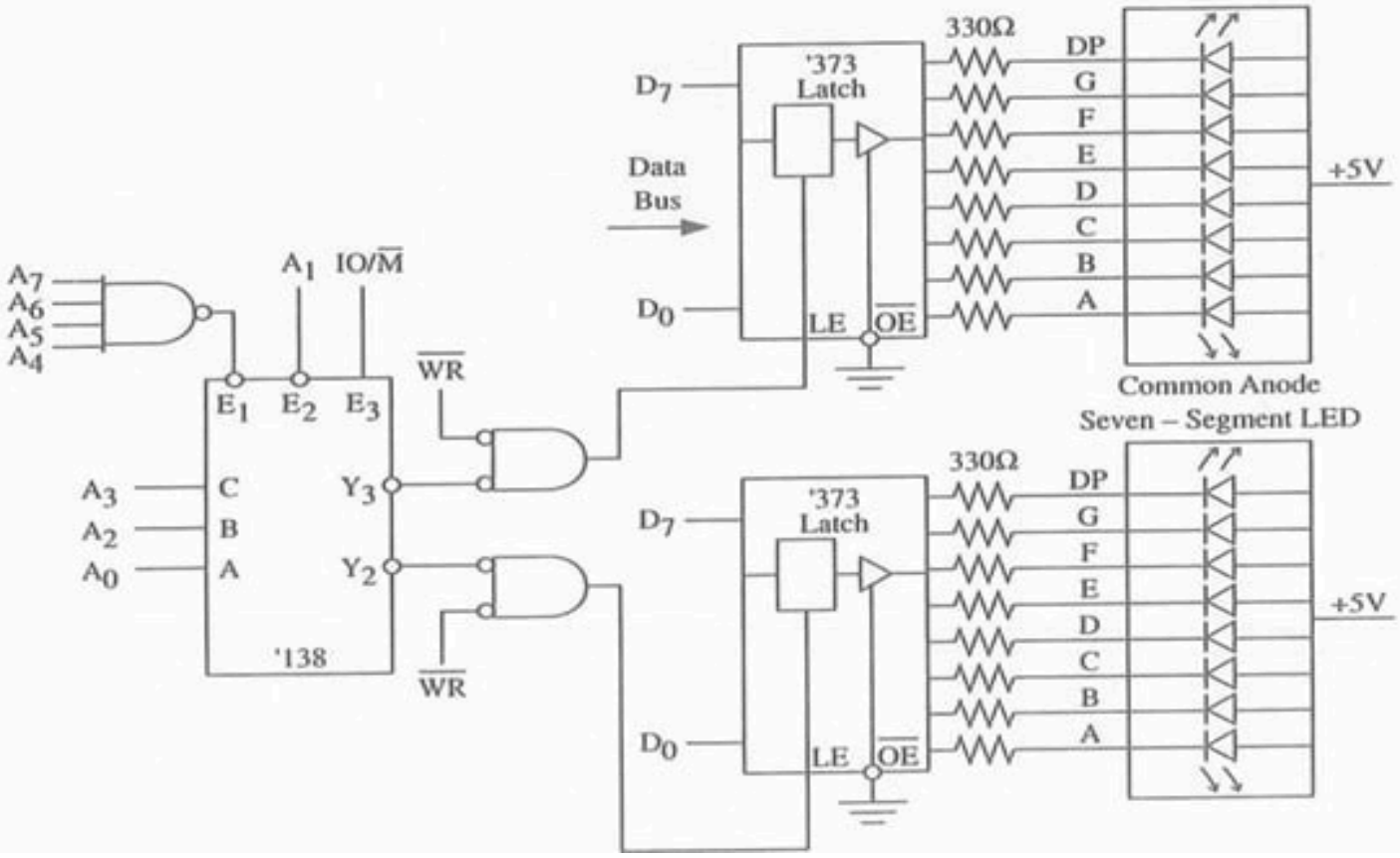
(iii) IN → OUT

Port B → Port A

∴ Program:

```
IN AL, 85H   ← Input from Port B
OUT 85H, AL  ← Output in Port A
```

32. In Figure 5.21, write instructions to display the number "97" at the common-anode seven-segment LED port.



⇒ To display, '97' in common anode 7-segment,

Here, '9' at upper one (Y_3) and '7' at lower one (Y_2)

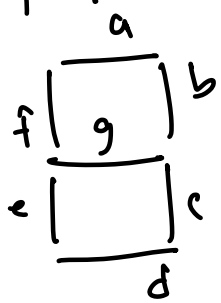
for, Y_3 → A_3 A_2 A_0 = 0 1 1 and Y_2 → A_3 A_2 A_0 = 0 1 0

Now, Port Address for both:-

A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
1	1	1	1	0	1	0	1	= F5 H (for 9)
1	1	1	1	0	1	0	0	= F4 H (for 7)

Now, in Common anode 7 segment display:-

	dp	g	f	e	d	c	b	a	
9:-	1	0	0	1	0	0	0	0	= 90H
7:-	1	1	1	1	1	0	0	0	= F8H



on at logic = 0

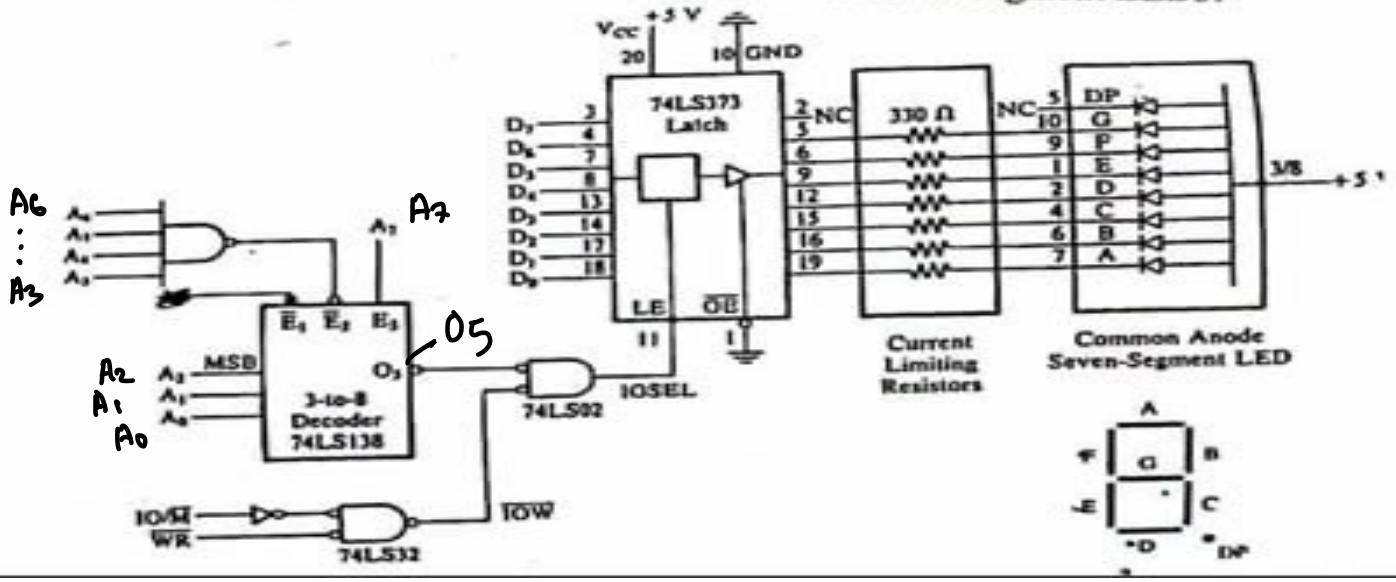
dp : 1

Program:-

```
MOV DX, F5H
MOV AL, 90H      } to display '9'
OUT DX, AL

MOV DX, F4H
MOV AL, F8H      } to display '7'
OUT DX, AL
```

6(a) Fig. 6(a) represents the interfacing of seven segment LED display with 8086 microprocessor. 15

i. Determine whether it is memory mapped I/O device or isolated I/O device.
ii. Write instruction to display digit 7 at the port.
iii. Why latch is used between decoder and seven segment LEDs?



⇒

(I) It is isolated I/O device,
as, inorder to on this, we need instruction like IN and OUT

(II) Here, 7 Segment in common anode. So, logic 0 to turn ON. LED.

Here, Here, for O_5 →

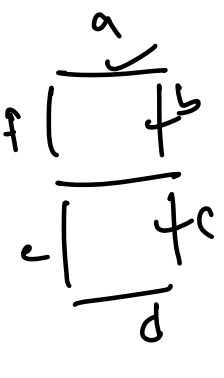
A_2	A_1	A_0
1	0	1

Now, Port Address,

A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
1	1	1	1	1	1	0	1	= FDH

Now, For 7 segment Display,

	dp	g	f	e	d	c	b	a	
7	1	1	1	1	1	0	0	0	= F8H

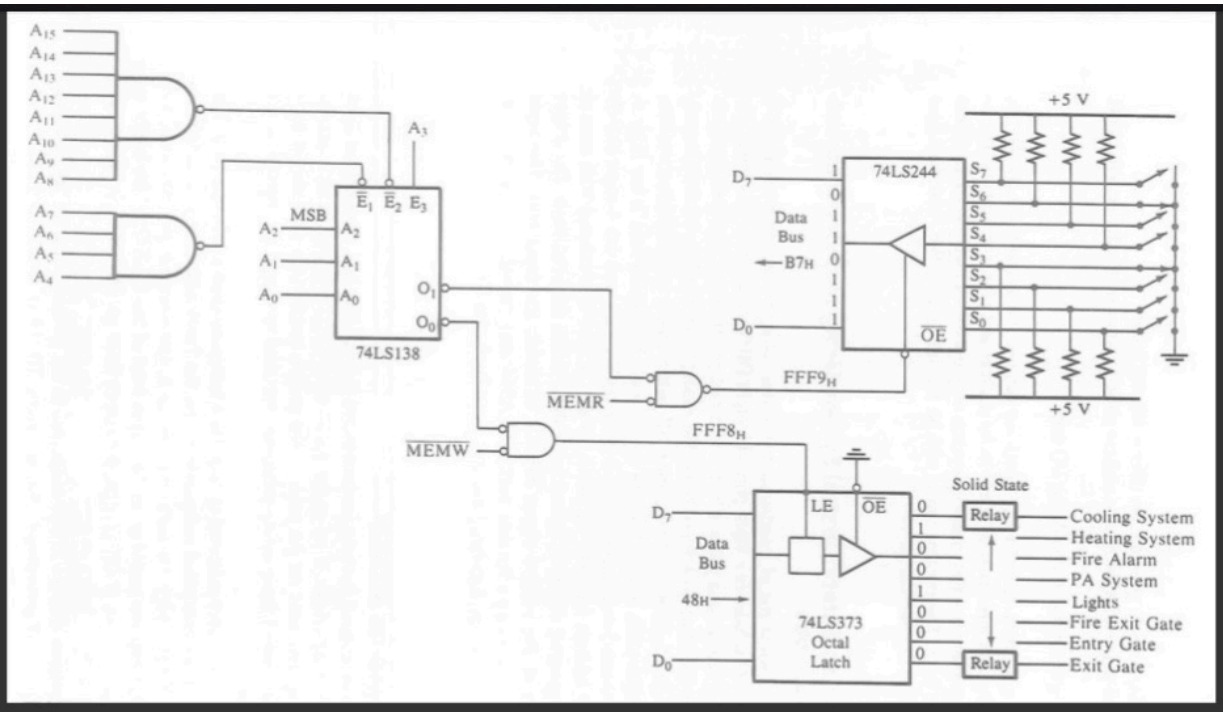


Program:

```
MOV DX, F0H
MOV AL, F8H
OUT DX, AL
```

(iii) The decoder is used to solve the address mismatch problem between microprocessor and input output device.

Microprocessor is very fast device, whereas output device is very slow with respect to microprocessor. So, input output device cannot read data from MP in a short time. To solve this problem Latch is used. When MP send data to address bus, latches hold data and send this to I/O devices it will wait to send data until I/O device read data properly



S_7	S_6	S_5	S_4	S_3	S_2	S_1	S_0	
1	0	1	1	0	1	1	1	= B7H

Read address :

A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	—	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
1	1	1	1	1	—	1	1	1	1	1	1	0	0	1

= FFF9H

Write address : FFF8H

8255 PPI

Programmable Peripheral Interface

Parallel I/O Devices - (i) Displays
(ii) key board
(iii) Printers (old Printers) etc.

Serial I/O Devices - (i) Some Printers (ii) Data Communication

8255

→ 24 I/O Pins

→ 8 bit Parallel Ports A and B

→ C Port :- can be grouped as 4 bit
CU (PC_7 - PC_4) [upper] and CL (PC_3 – PC_0) [lower]

Basically two Modes :-

(i) BSR Mode - Set / Reset

(ii) I/O Mode - Input / output

(i) BSR Mode does set/Reset in Port C

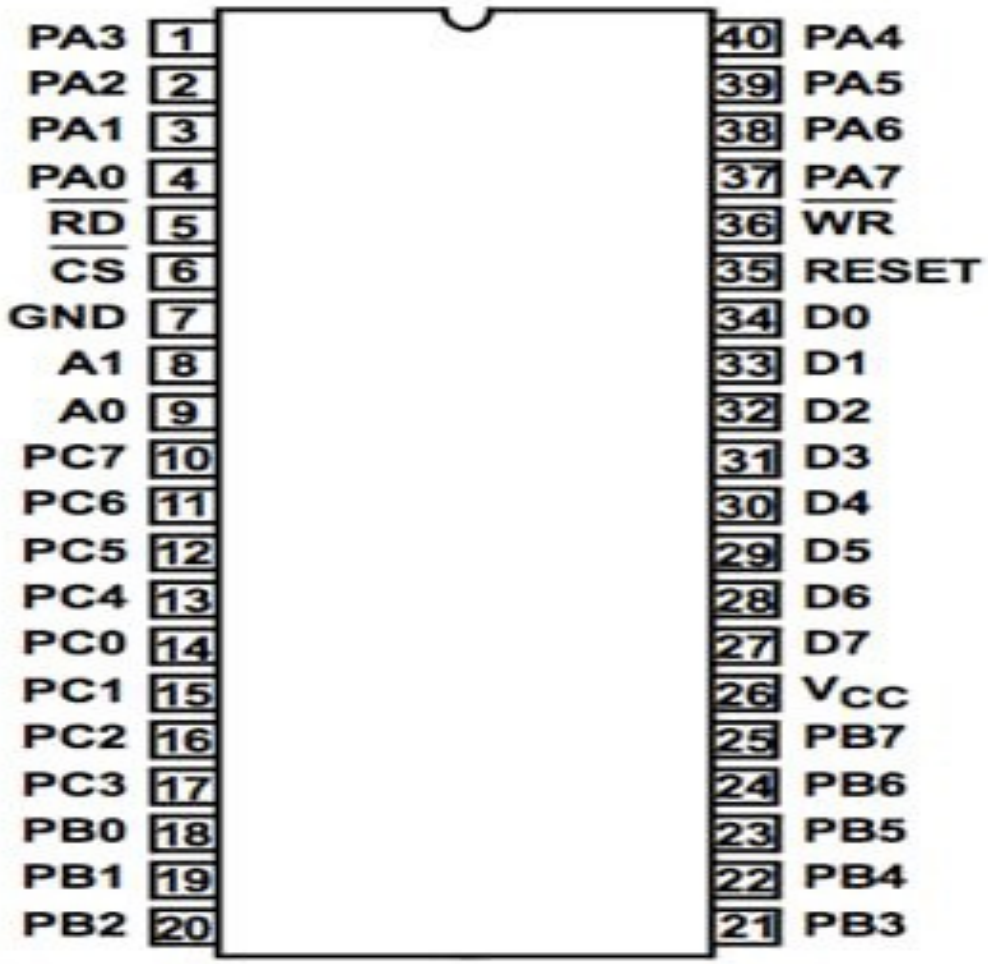
(ii) I/O Modes divided into -

(i) Mode 0 (Simple I/O)

(ii) Mode 1 (Handshake)

(iii) Mode 3 (Bidirectional)

Pin Diagram



Functions of Pins

→ Data Bus (D_0 - D_7) :- 8 bit bidirectional data Buses. are connected to 8086 data bus for transferring data.

→ $\overline{CS}$ (chip select) :- This is active low signal. When, it is low, data is transferred from 8086.

→ $\overline{RD}$: When it is low → Read operation will occur

→ $\overline{WR}$:- " " " " → Write " " "

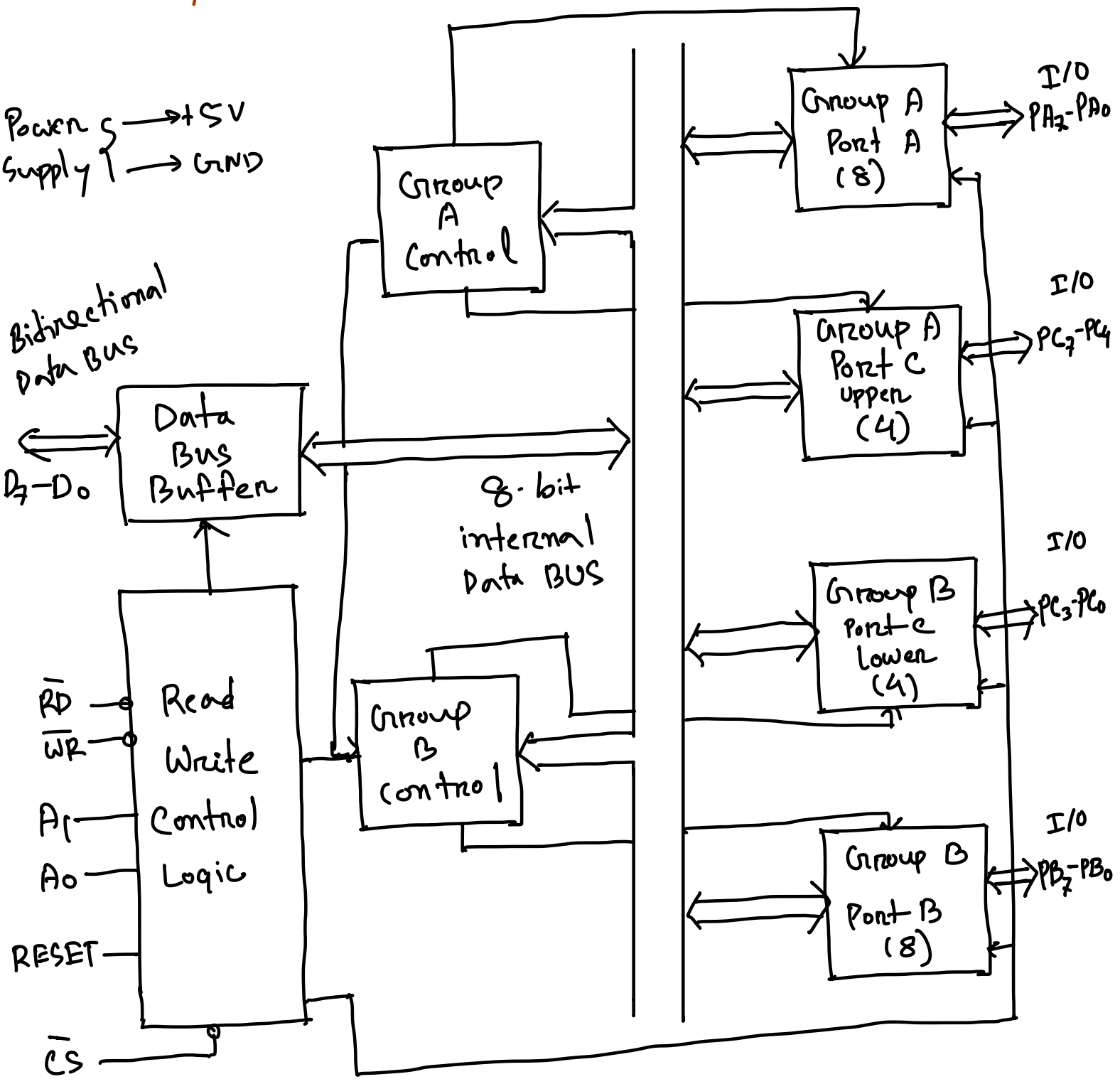
→ Address (A_0-A_1) :- This select the Port of 8255 PPI.

A_0	A_1	
0	0	→ Port A
0	1	→ Port B
1	0	→ Port C
1	1	→ control Register

→ RESET → This is used to reset the device.
This operation cleans control register.

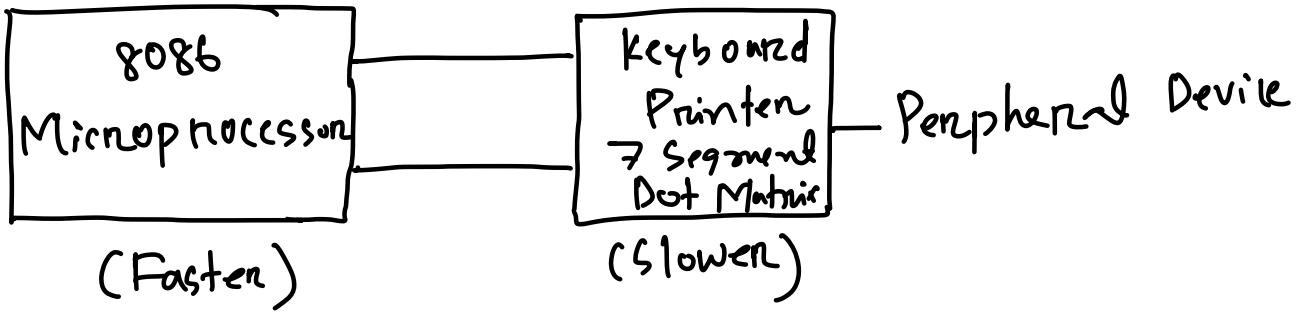
$PA_0 - PA_7$
$PB_0 - PB_7$
$PC_0 - PC_7$
} 8 bit bidirectional Data Bus

Block Diagram of 8086 with 8255 PPI



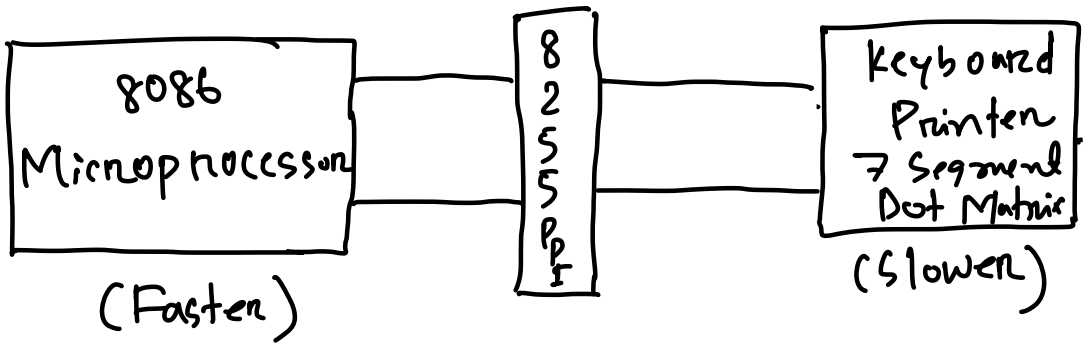
Why PPI are used with 8086 Microprocessor ?

⇒

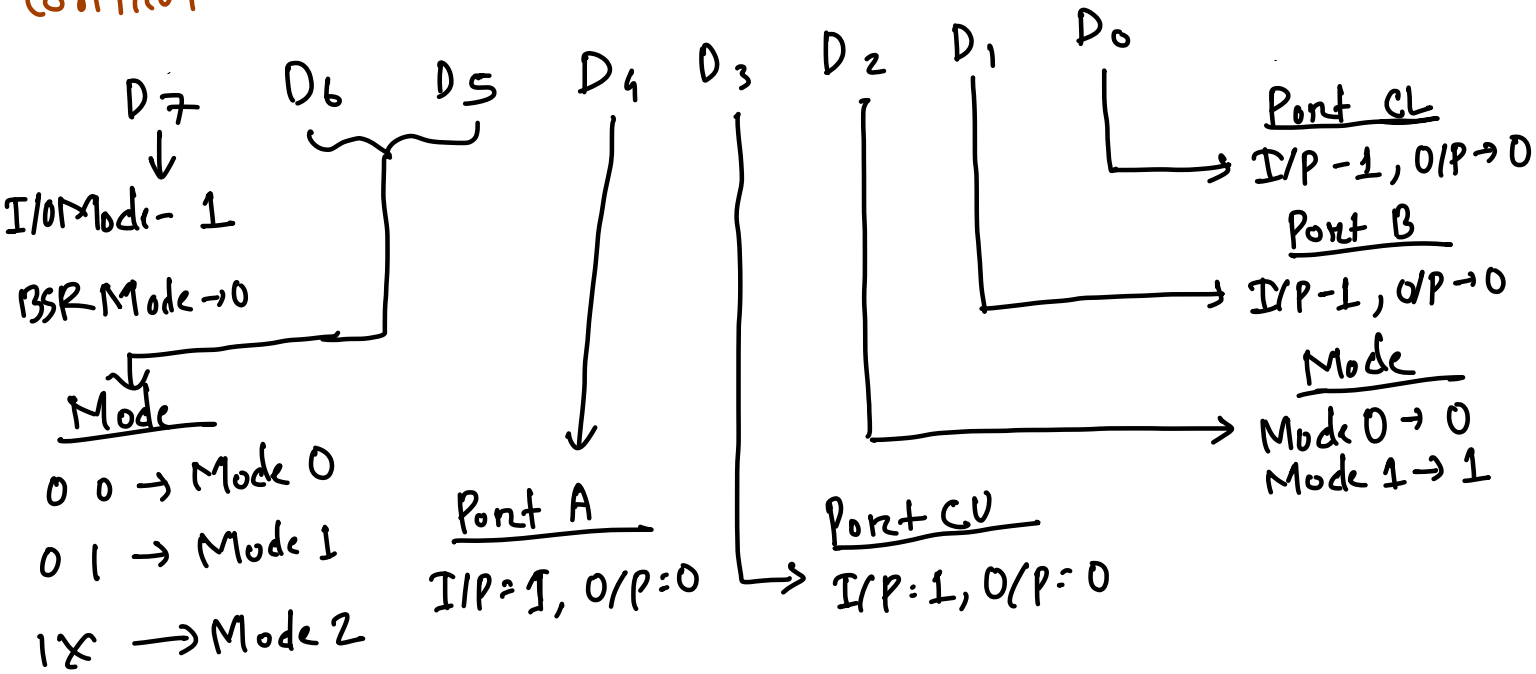


Since the 8086 MP responses faster than the peripheral device, there is a chance of data overwriting.

To solve this problem, we need to add a synchronizing device between the MP and Peripheral Device. which is called Programmable Peripheral Interface. (PPI)



Control Word of 8255 PPI



Mode-0 :- Simple Input Output

→ In this mode, Port A and Port B are used as two simple 8 bit I/O ports and Port C as two 4-bit ports.

→ Each Port can be Programmed to function as simply an input port or output port.

→ The input/output feature in Mode 0 are as follow :-

(i) Output are Latched

(ii) Input are not Latched

(iii) Ports don't have handshake or interrupt capability.

Mode 1 :- Handshake

Handshaking Signal :

The Microprocessor and the peripheral device operate at different speeds. Therefore, the signals are exchanged before data transfer between fast responding MPU and slow responding Peripherals. The signals are called Handshaking signal.

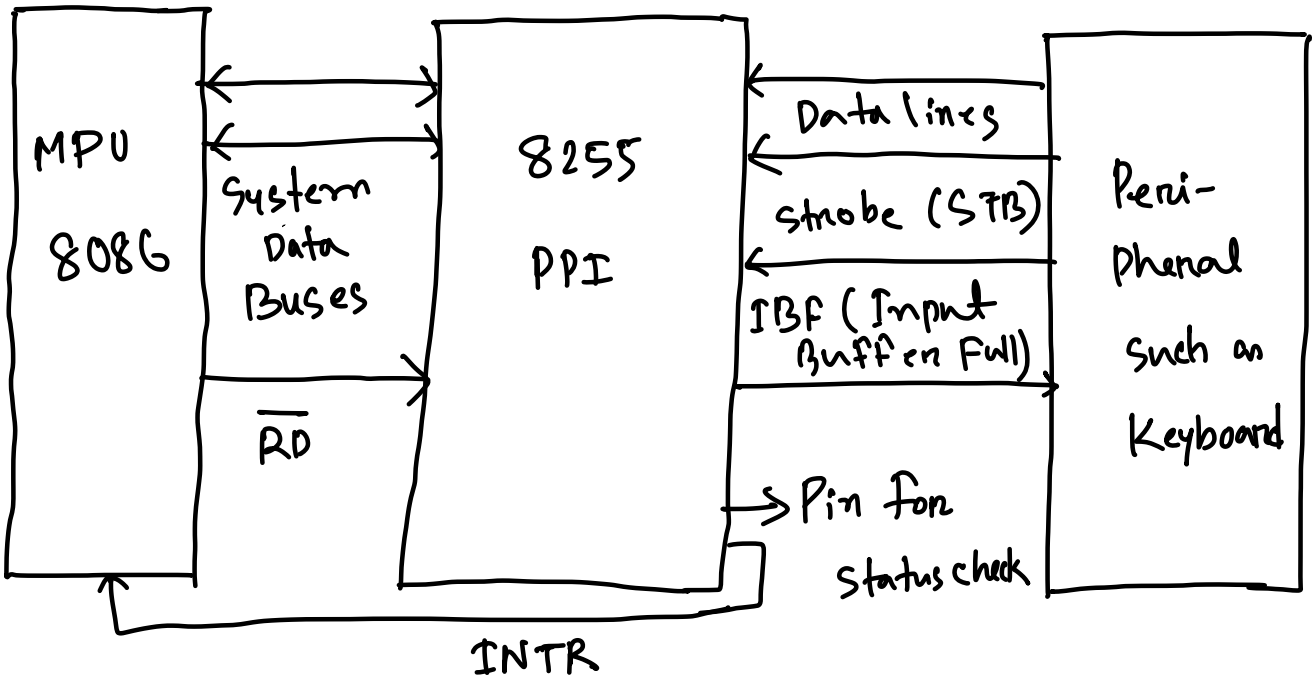
→ Two Ports (A and B) function as 8 bit I/O ports. They can be configured as either input or output ports.

→ Each port uses 3 lines from Port C. as Handshake signal. The remaining two line can be used as simpe I/O operation

→ Input and output Data are latched.

→ Interrupt logic is supported.

Data Input Interfacing with Handshaking Signal (Read Operation)



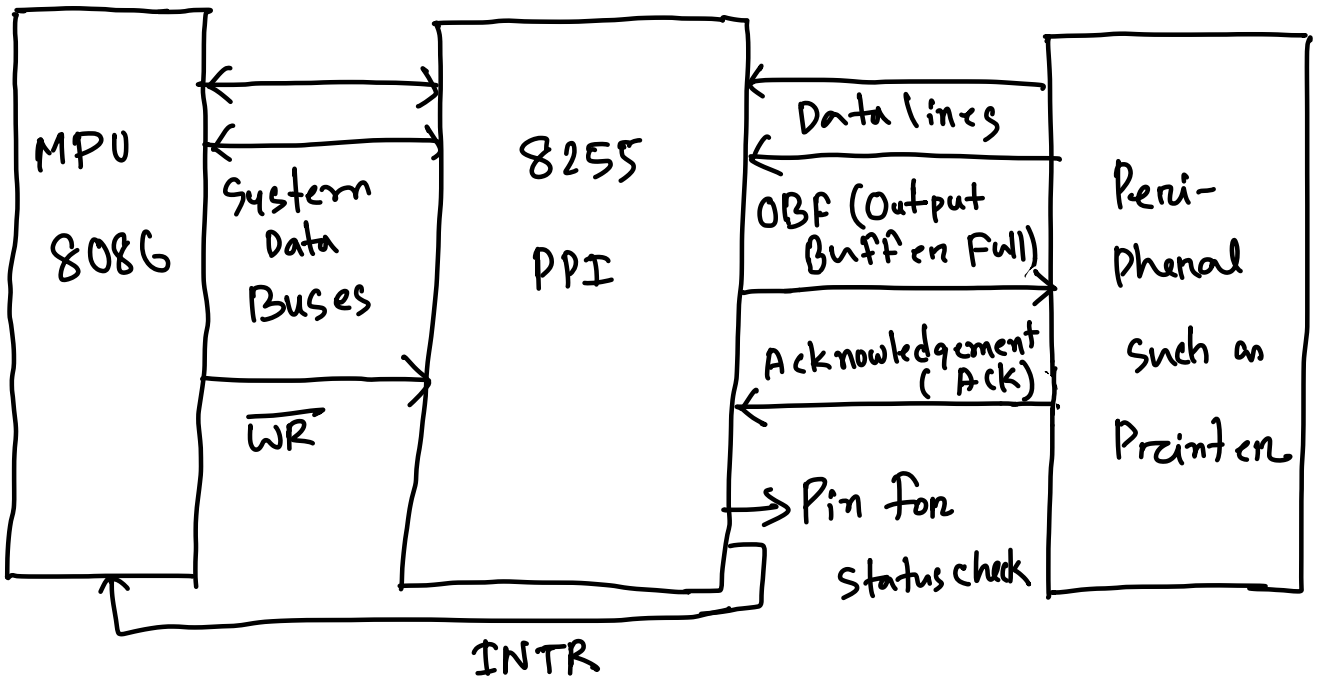
(1) A Peripheral strobe places a data byte in the input port and inform the PPI by sending a handshaking signal STB(strobe)

(2) The PPI device informs the peripheral device that, it's input is full. Don't send the next byte untill this one has been read. This message is sent to peripheral device by handshaking signal IBF (Input Buffer full)

(3) The MPU keep checking the status untill a byte is available. Or the PPI sends an interrupt to MPU to inform that it has a byte to read.

(4) MPU reads the byte by sending control signal $\overline{RD}$

Data Output Interfacing with Handshaking Signal (Write Operation)



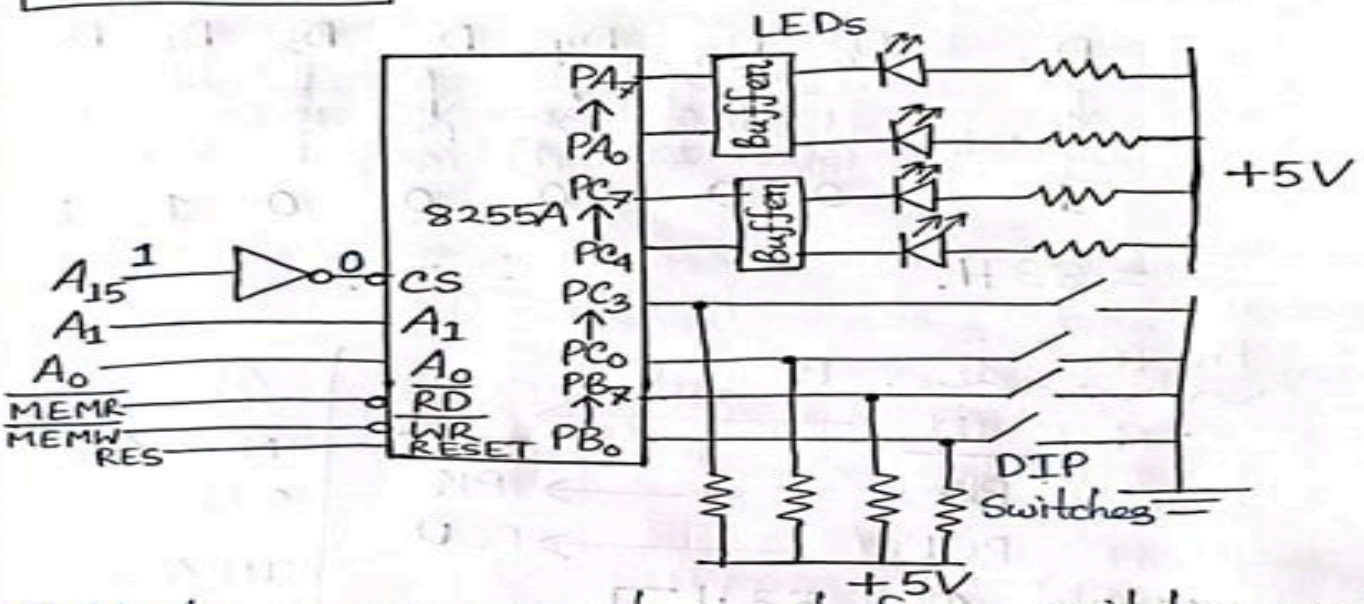
(1) The MPU writes a byte into the output port of the Programmable device by sending a control signal $\overline{WR}$.

(2) The PPI informs the Peripheral, by sending the Handshaking signal OBF, that a byte has to be written.

(3) The peripheral acknowledges the byte by sending back the ACK (Acknowledgment) signal to interfacing device

(4) The PPI ask for the next byte through interrupting the MPU or the MPU finds out that the byte has been acknowledged through the status check

Problem – 01:



Write a program to read from switches and display the reading from port B to port A and from port CL to PCU.

⇒ Here, Port Address:

A_{15}	A_{14}	A_3	A_2	A_1	A_0	
1	0	0	0	0	0	⇒ 8000H → Port A
1	0	0	0	0	1	→ 8001H → Port B
1	0	0	0	1	0	→ 8002H → Port C
1	0	0	0	1	1	→ 8003H → Control Register

Here, Port A, Port CU has LED → So output device

Port B, Port CL " Switch → So input device

and it is Mode 0

Control Word:-

D_7	D_6	D_5	D_4	D_3	D_2	P_1	D_0	
1	0	0	0	0	0	1	1	= 83H

Program :

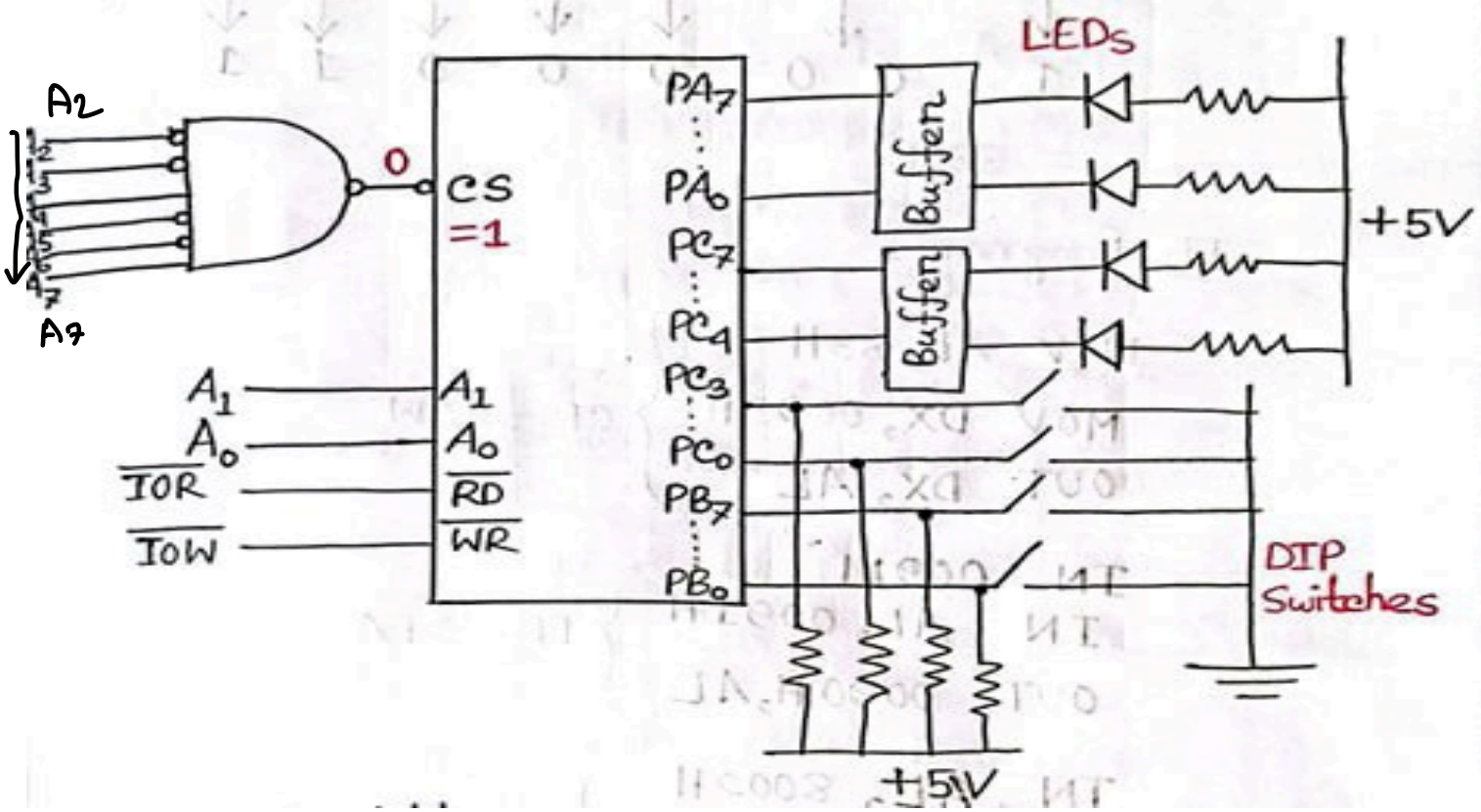
```
PPIA   EQU  8000H
PPIB   EQU  8001H
PPIC   EQU  8002H
PPIC_C EQU  8003H

MOV  AL, 83H
OUT  PPIC_C, AL

IN   AL, PPIB
OUT  PPIA, AL
IN   AL, PPIC
SHL  AL, 04
OUT  PPIC, AL
```

} Defining Port Address

} control word → Control Register

} As per Question.

Problem-02: Write a program to read from switches and display the reading from port B to port A and port PCL to PCU.



⇒ Here, Port Address:

A_{15}	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
0	0	1	0	0	1	0	0	0	0	→ 0090H → Port A
0	0	1	0	0	1	0	0	0	1	→ 0091H → " B
0	0	1	0	0	1	0	0	1	0	→ 0092H → " C
0	0	1	0	0	1	0	0	1	1	→ 0093H → control Register

Control Word :-

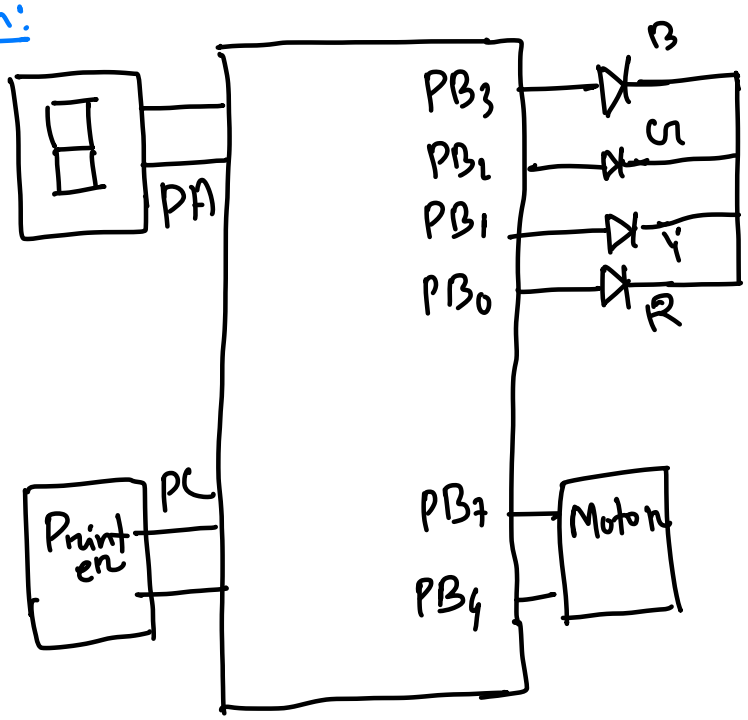
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
1	0	0	0	0	0	1	1	= 83H

Program:

```
PPIA    EQU 0090H
PPIB    EQU 0091H
PPIC    EQU 0092H
PPIC_C  EQU 0093H

MOV  AL, 83H
OUT  PPIC_C, AL
```

```
IN   AL, PPIB
OUT  PPIA, AL
IN   AL, PPIC
SHL  AL, 04
OUT  PPIC, AL
```

Problem:



Write a program to display R → Y → G → B LEDs.

PA : 1BH
PB = 1CH
PC = 1DH
CR = 1EH

⇒ Given, PA = 1BH ; PB = 1CH ; PC = 1DH ; CR = 1EH

PA, PB, PC → all are output connected

For Control Word, D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0

1 0 0 0 0 0 0 0 = 80H

Now, LEDs are connected in $PB_0 - PB_3$

For RED → PB_7 - - - PB_4 PB_3 PB_2 PB_1 PB_0

Yellow → 1 . . . . 1 0 0 0 1

1 . . . . 1 0 0 1 0

Green → 1 . . . . 1 0 1 0 0

Blue → 1 . . . . 1 1 0 0 0

Program:

```
PPIA EQU 1BH
PPIB EQU 1CH
PPIC EQU 1DH
PPIC_C EQU 1EH
MOV AL, 80H
OUT PPIC_C, AL
L1: MOV AL, 11110001B
L2: OUT PPIB, AL
    DELAY
    SHL AL, 01
    TEST AL, 00010000B
    JNZ L1
    OR AL, 11100000B
    JMP L2
    END
```

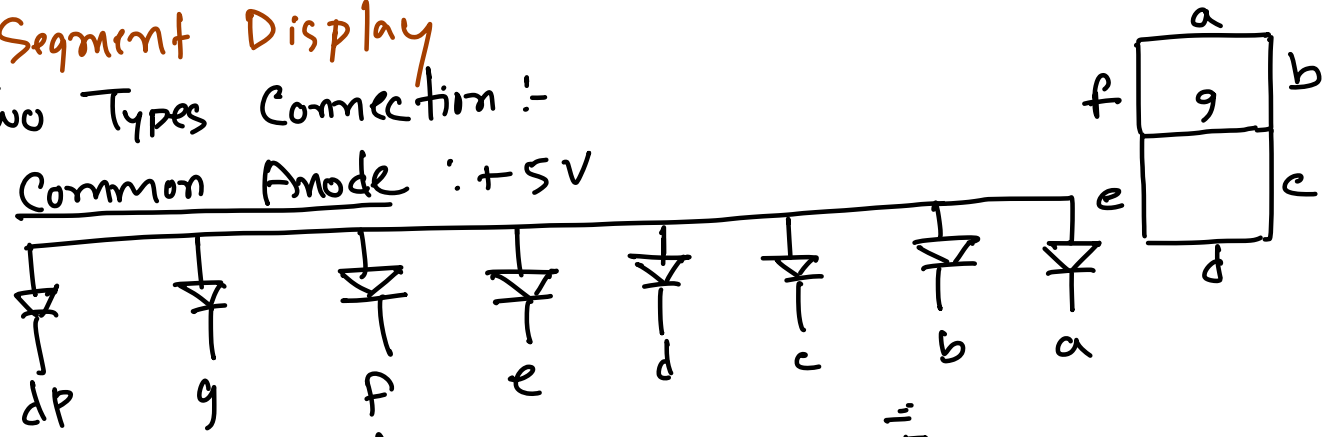
Alternative.

```
PPIA EQU 1BH
PPIB EQU 1CH
PPIC EQU 1DH
PPIC_C EQU 1EH
MOV AL, 80H
OUT PPIC_C, AL
MOV AL, 11110001B   } R
OUT PPIB, AL
MOV AL, 11110010B   } Y
OUT PPIB, AL
MOV AL, 11110100B   } G
OUT PPIB, AL
MOV AL, 11111000B   } B
OUT PPIB, AL
END
```

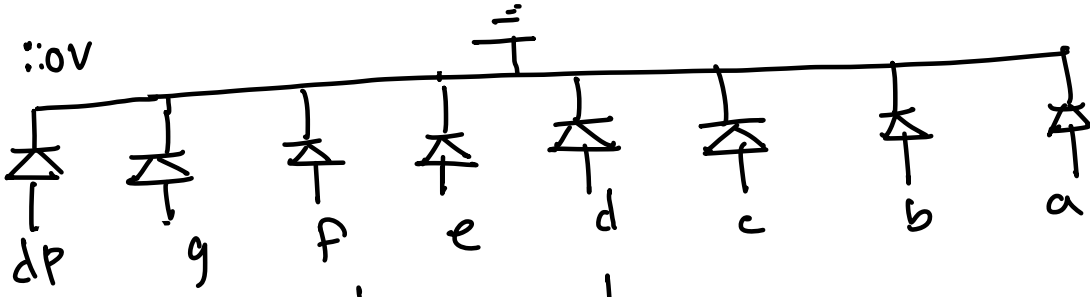
7 Segment Display

Two Types Connection :-

(I) Common Anode : +5V



(II) Common Cathod : 0V

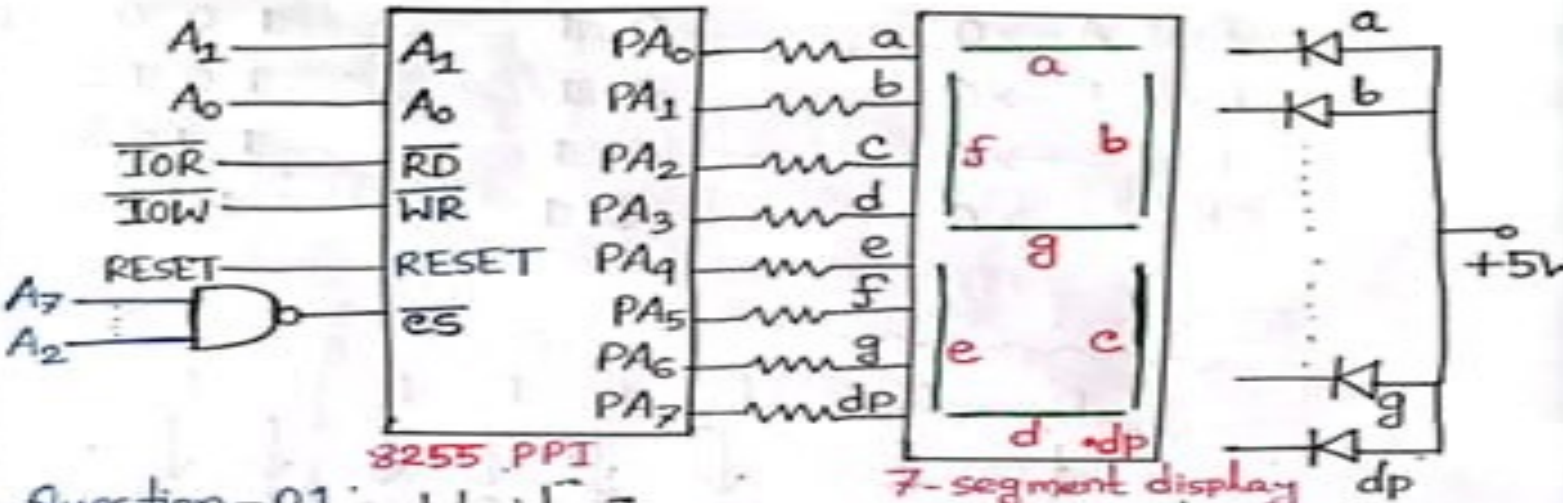


→ To Turn ON Common anode Connection :-
logic '0' will be given to Cathode (dp, g.. a)

→ To Turn ON Common cathod Connection :-
logic '1' will be given to Anode (dp, g.. a)

→ Decimal Pointer (dP) will always be set to = 0 (don't care)

Problem :



Question-01: Write a program to display 0-9 digits using 7-segment display.

Question-02: Write a program to display a specific digit 1/2/3 .....

Question-03: To display a specific alphabet A/B/C/D.

Ans :

⇒ (1) Here, Port Address :-

A_{15} ... A_8	A_7 ..... A_2	A_1	A_0	
0 0	1 1	0	0	→ 00FCH → PA
0 0	1 1	0	1	→ 00FDH → PB
0 0	1 1	1	0	→ 00FEH → PC
0 0	1 1	1	1	→ 00FFH → CR
0 0	1			

Control word,

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
1	0	0	0	0	0	0	0	: 80H

Program :-

```
PPIA EQU 00FCH
PPIB EQU 00FDH
PPIC EQU 00FEH
PPIC_C EQU 00FFH
MOV AL, 80H
OUT PPIC_C, AL
L2: MOV SI, OFFSET DATA
L1: MOV AL, BYTE PTR DS:[SI]
    CMP AL, 00H
    JZ L2
    OUT PPIA, AL
    INC SI
    JMP L1
```

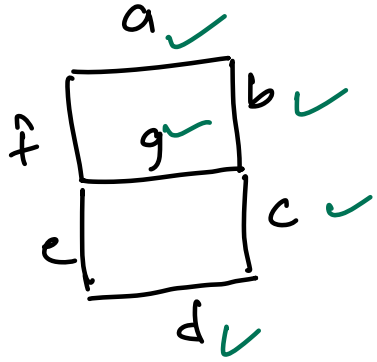
[7 Segment in Common Anode. So logic '0' to turn on a, b, ... f]

g f e d c b a

Data :- DB 1100 0000 B
DB 1111 1001 B
DB 1010 0100 B
DB 1011 0000 B
DB 1001 1001 B
DB 1001 0010 B
DB 1000 0010 B
DB 1111 1000 B
DB 1000 0000 B
DB 1001 0000 B

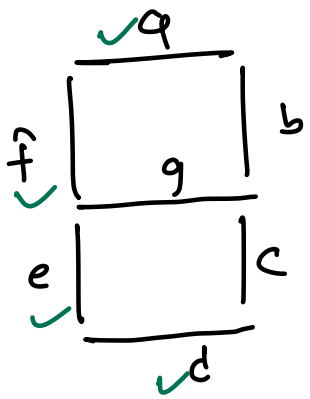
sequence of 0-9

(II) To display only 3




```
PPIA EQU 00FC H
PPIB EQU 00FD H
PPIC EQU 00FE H
PPIC_C EQU 00FF H
MOV AL, 80H
OUT PPIC_C, AL
MOV AL, 10110000 B
OUT PPIA, AL
```

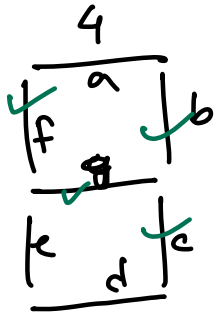
(iii) To Display C in 7-Segment




```
PPIA EQU 00FC H
PPIB EQU 00FD H
PPIC EQU 00FE H
PPIC_C EQU 00FF H
MOV AL, 80H
OUT PPIC_C, AL
MOV AL, 1100 0110B
OUT PPIA, AL
```

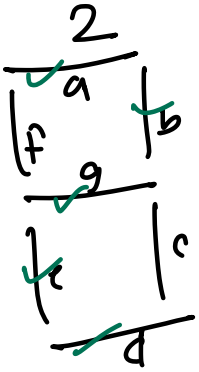
Problem: Write an instruction to display the number '42' at Common Cathode 7 Segment Display

⇒



→ (common cathod
→ dp = 0
→ a/b/.. g ON → '1'

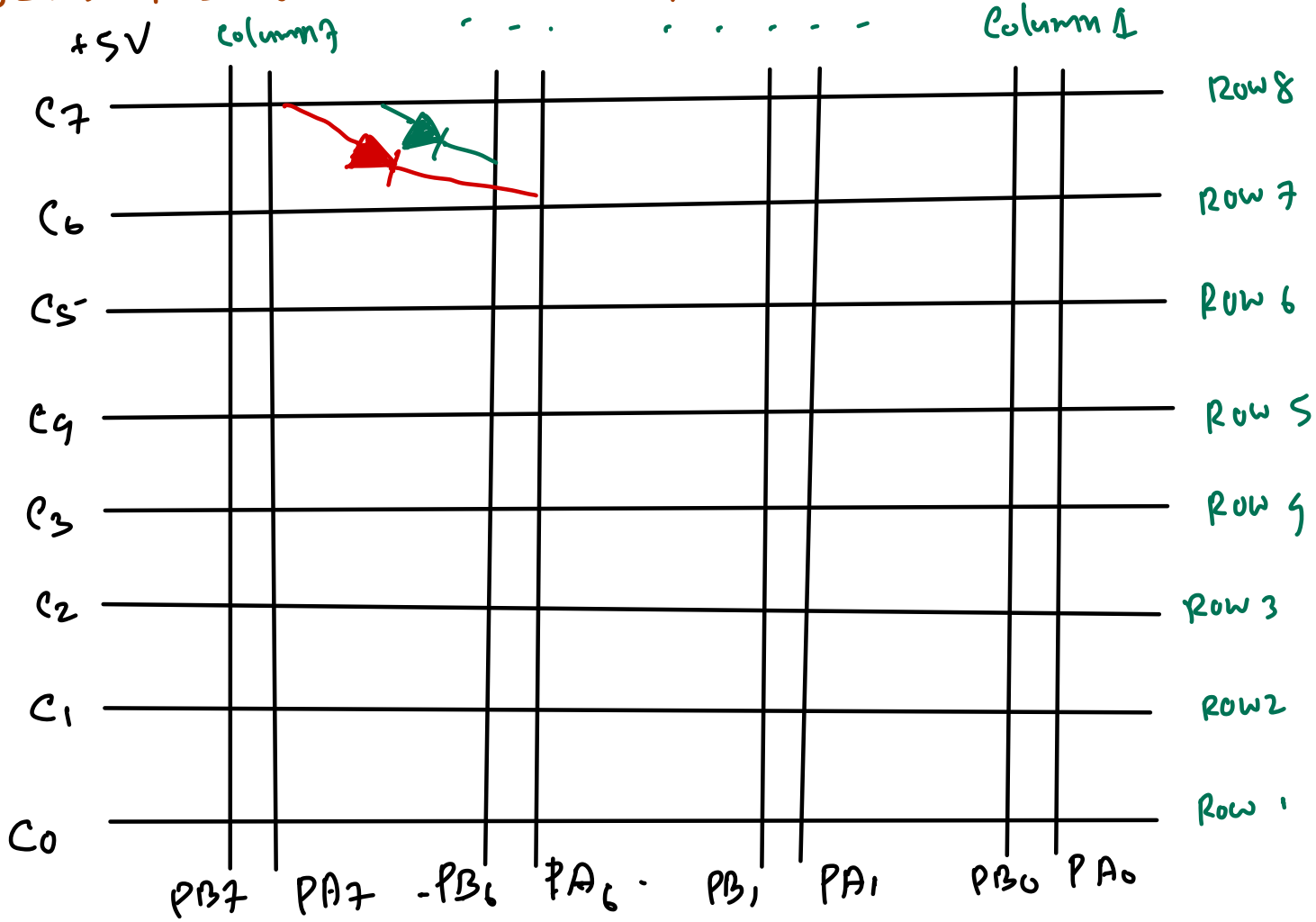
⇒ 0110 0110 B



⇒ 0101 1011 B

```
PPIA EQU 00FC H
PPIB EQU 00FD H
PPIC EQU 00FE H
PPIC_C EQU 00FF H
MOV AL, 80H
OUT PPIC_C, AL
MOV AL, 0110 0110B
OUT PPIA, AL
MOV AL, 0101 1011B
OUT PPIA, AL
```

8255 PPI with DOT Matrix



Port C → Common Anode (RED, Green) → ROW

Port A → RED LED cathode
Port B → Green LED cathod
} → Column

To turn on the LEDs. $PC_7 : \ldots\ldots PC_0 = 1$

$P_{B7} \ldots\ldots PB_0 = 0$

$P_{A7} \ldots\ldots PA_0 = 0$

Problem: Write a Program to sequentially generate the 1st ROW of DOT Matrix with RED LED.

Given, PA = 18H; PB = 1AH; PC = 1CH; CR = 1EH

⇒ Control Word,

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
1	0	0	0	0	0	0	0	= 80H

Program :

```
PPIA   EQU 18H
PPIB   EQU 1AH
PPIC   EQU 1CH
PPIC_C EQU 1EH
MOV AL, 80H
OUT PPIC_C, AL
MOV AL, FFH      } Anode ON (Port C)
OUT PPIC, AL     }
MOV AL, FFH      } Green off (Port B)
OUT PPIB, AL     }
L1 : MOV AL, 11111110 B
L2 : OUT PPIA, AL
     SHL AL, 1
     CMP AL, 00H
     JZ L1
     JMP L2
```

Problem: Program for generating one point (one LED in a row sequentially) at a time.

=>

Program:

```
L1 : MOV AL, 11111110 B
L2 : OUT PPIA, AL
     ROL AL, 1
     CMP AL, 00H
     JZ L1
     JMP L2
```

Problem: Write a Program to turn Green LED from 0C → 1C → 7C | Given, PA: 1CH, PB: 1DH, PC: 1EH; CR: 1FH

⇒

```
PPIA    EQU 1CH
PPIB    EQU 1DH
PPIC    EQU 1EH
PPIC_C  EQU 1FH
        MOV AL, 80H
        OUT PPIC_C, AL
        MOV AL, FFH
        OUT PPIC, AL
        MOV AL, FFH
        OUT PPIA, AL
L1:     MOV AL, 1111 1110B
L2:     OUT PPIB, AL
        CALL TIMER
        STC
        ROL AL, 1
        JC L1
        JMP L2
```

Control word: 1000 0000 B

= 80H

0C → 1111 1110 B

1C → 1111 1101

⋮

07C → 0111 1111 B

7(a) Fig. 7(a) illustrates a 8×8 dual color (Red and Green) dot matrix display. Using EQUATE (EQU) directive, write an assembly language program to illustrate the dots shown in Fig. 7(a) using green LED (solid dots). Assume that the port addresses of PA, PB and PC are 18H, 1AH, and 1CH respectively, where PC is connected to anode. 18

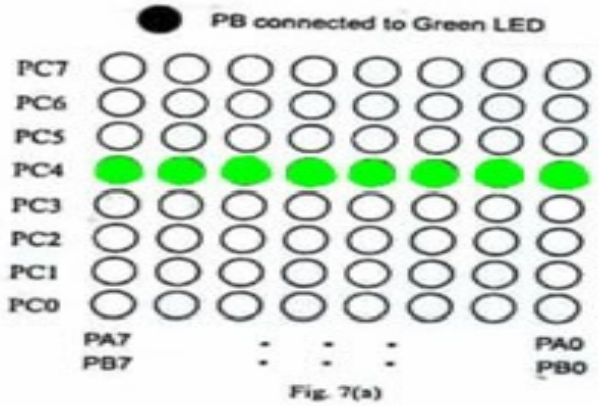


Fig. 7(a)

⇒ Given, PA = 18H, PB = 1AH, PC = 1CH, CR = 1EH

Control Word,

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
1	0	0	0	0	0	0	0	= 80H

Program to on green light in 5th Row:-

```
PPIA EQU 18H
PPIB EQU 1AH
PPIC EQU 1CH
PPIC_C EQU 1EH
MOV AL, 80H
OUT PPIC_C, AL
MOV AL, 11111111B
OUT PPIA, AL
MOV AL, 00010000B
OUT PPIC, AL
MOV AL, 00000000B
OUT PPIB, AL
```

Problem: Write a program to show generate 'A' using RED LED in the dot matrix.

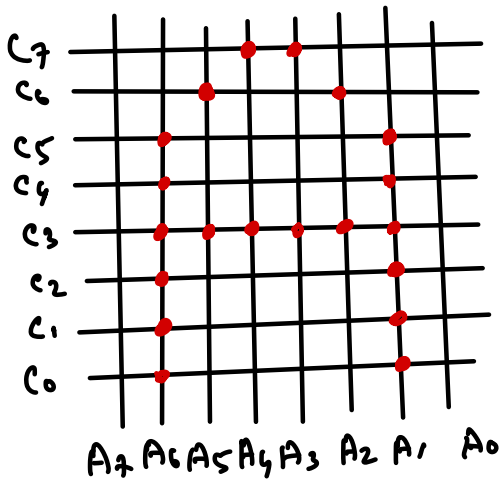
⇒ Program:

```
PPIA EQU 18H
PPIB EQU 1AH
PPIC EQU 1CH
PPICC EQU 1EH
```

```
MOV AL, 80H
OUT PPICC, AL
MOV AL, FFH
OUT PPIB, AL
L1 :- MOV SI, OFFSET Point
MOV AH, 11111110B ; Column scanning
L2 :- MOV AL, BYTE PTR DS:[SI]
OUT PPIC, AL
MOV AL, AH  ; Row Scanning
OUT PPIA, AL
Call Timer
INC SI
JC L1
JMP L2
END
```

Point :

```
DB  0 000 0000 B
DB  1 1 1 1  1 1 00  B
DB  0 00 1 0 0 1 0  B
DB  00 0 1 0 0 0 1  B
DB  000 1  0 0 0 1  B
DB 0 0 0 1  0 0 1 0 B
DB 11 11  1 1 0 0  B
DB 00 00 0 0 0 0  B
```



Problem: To illuminate green LED row wise

4 → 3 - 2 → 1 → 0 [PA: 1CH, PB = 1DH, PC = 1EH, CR = 1FH, CW: 80H]

⇒

Code:

```
PPIA EQU 1CH
PPIB EQU 1DH
PPIC EQU 1EH
PPIC_C EQU 1FH
MOV AL, 80H
OUT PPIC_C, AL
MOV AL, FFH
OUT PPIA, AL
MOV AL, 00H
OUT PPIB, AL
L1 : MOV AL, 00010000B
L2: OUT PPIC, AL
  DELAY
  ROR AL, 1
  CMP AL, 00000001B
  JE L1
  JMP L2
  END
```

Problem: To illuminate RED LED Column Wise

5 → 4 → 3 → 2

[PA: 1CH, PB = 1DH, PC = 1EH,
CR = 1FH, CW = 80H]

⇒

Code:

```
PPIA EQU 1CH
PPIB EQU 1DH
PPIC EQU 1EH
PPIC_C EQU 1FH
MOV AL, 80H
OUT PPIC_C, AL
MOV AL, FFH
OUT PPIB, AL
MOV AL, FFH
OUT PPIC, AL
L1: MOV AL, 11011111B
L2: OUT PPIA, AL
DELAY
ROR AL, 1
CMP AL, 11111101B
JE L1
JMP L2
END
```

Mode-1 : Input Output With Handshake

→ In this mode, handshake signal are exchanged between the MPU and Peripheral before data transfer

→ Two Port (A and B) function as 8 bit I/O Port can be configured as Input and Output Port

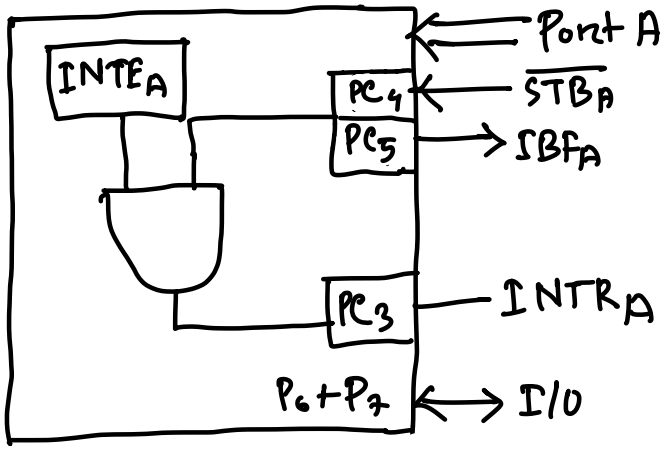
→ Each port uses three lines from port C as Handshake Signal. Other two ports can be considered as simple I/O operation.

→ Input and Output data are latched

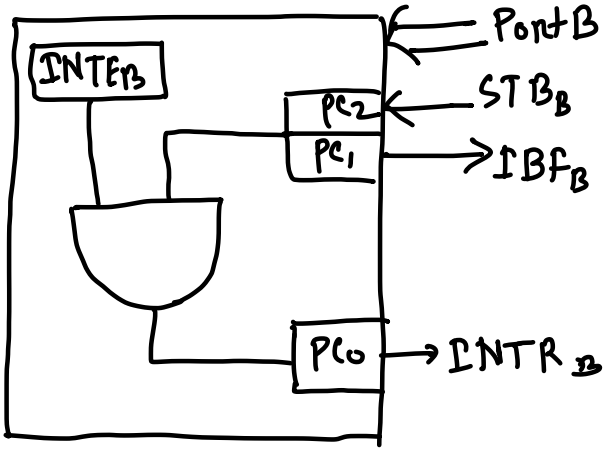
→ Interrupt logic is supported

Input Handshaking (Read Operation) in Mode 1

Mode 1, Port A



Mode 1, Port B

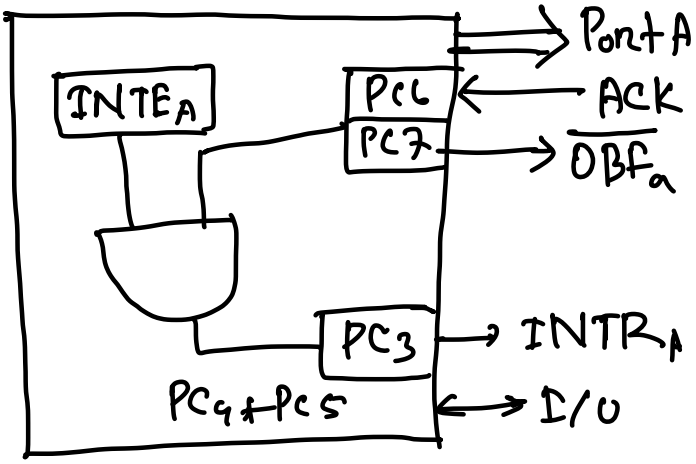


Status Word—Mode 1 Input

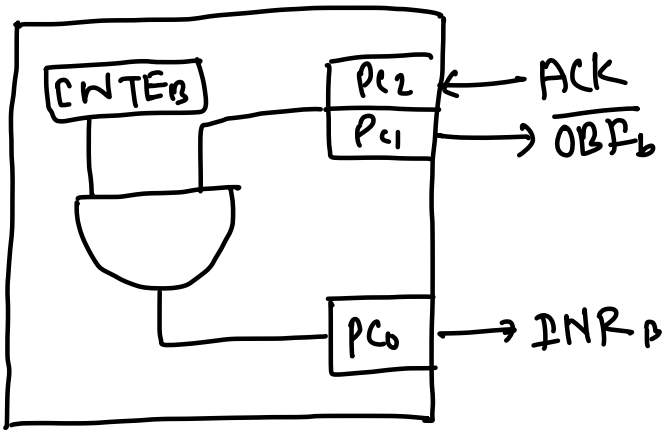
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
I/O	I/O	IBF_A	$INTE_A$	$INTR_A$	$INTE_B$	IBF_B	$INTR_B$

Output Handshaking (writing Operation) in Mode 1

Mode 1 Port A



Mode 1 Port B



Status Word

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
$\overline{OBF}_A$	$INTE_A$	I/O	I/O	$INTR_A$	$INTE_B$	$\overline{OBF}_B$	$INTR_B$

Control Word

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	1	I/O	I/O	1	I/O	X(0)

I/O

Mode 1 G

Port A
input = 1
output : 0

Port B
input = 1
output = 0

Port B Mode 1

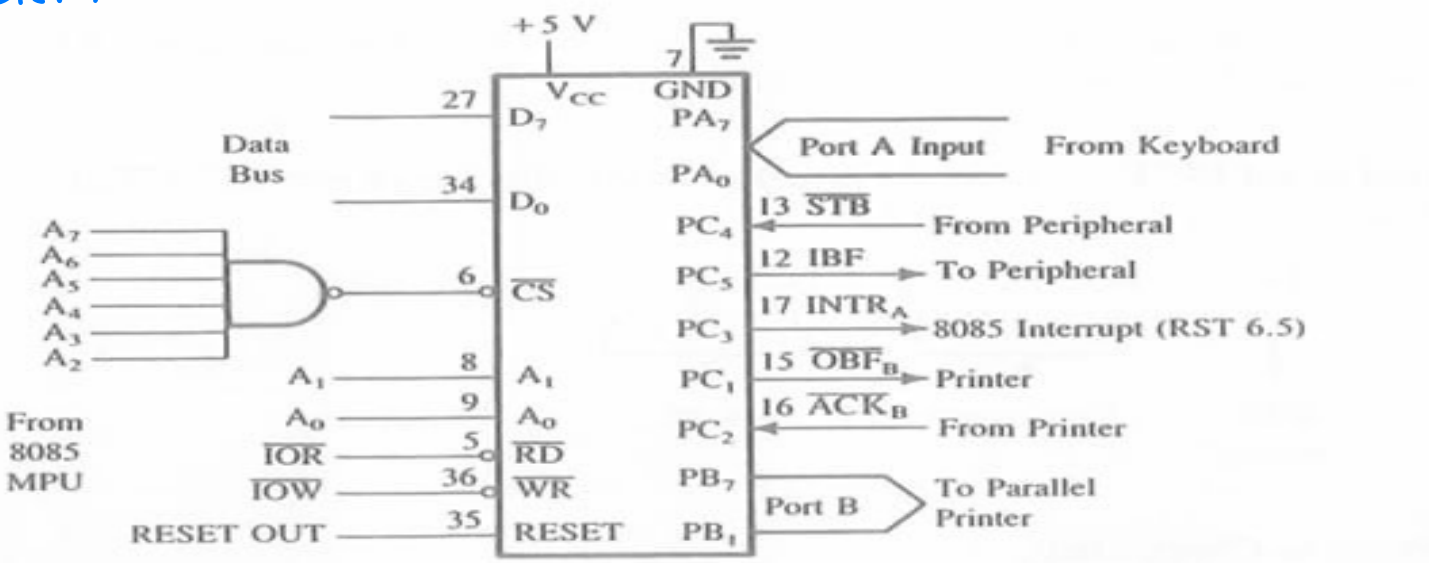
input mode ($\overline{RD}$)

$PC_{6,7}$
input = 1
output = 0

Output mode ($\overline{WR}$)

$PC_{4,5}$
input = 1
output = 0

Problem



Find, (I) Port Address, (II) Control Word, (III) BSR Word (IV) Status word. to enable INTEA to check $\overline{OFB_B}$

⇒

(I) Port Address :-

A_{15} .... A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
0 0	1	1	1	1	1	1	0	0	= 00FCH Port A
0 0	1	1	1	1	1	1	0	1	= 00FDH Port B
0 0	1	1	1	1	1	1	1	0	= 00FEH Port C
0 0	1	1	1	1	1	1	1	1	= 00FFH CR

(II) Control Word :-

Set up Port A as input and Port B as Output.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
1	0	1	1	0	1	0	0	= B4H

(Port A i/p) ← 1 → (PC 6,7 don't care) 0 → (Port B o/p) 0

∴ Control Word = B4H

(III) BSR Word To enable INTEA (Port A)

→ To enable or generate INTEA, Flipflop must be set by 1. This can be done by using BSR mode to set PC4

∴ BSR Word:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	0	0	0	1	0	0	1	= 09 H

D_0 → set

1 0 0 → PC4

Program, MOV AL, 09 H

OUT 00FF H, AL

(iv) Status Word to check $\overline{OFB_B}$

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
x	x	x	x	x	x	$\overline{OFB_B}$	x	= 02H

∴ Musking byte = 02H

Bit Set Reset Mode / BSR Mode

→ This mode concerned only with 8 bit of Port C

This Port can be set and Reset by writing appropiate control word in Control Register

→ C.W with $D_7 = 0$ recognize this mode.

→ Dosn't chang previously transmitted C.W at $D_7 = 1$

So, I/O connection of Port A ; Port B is unaffected.

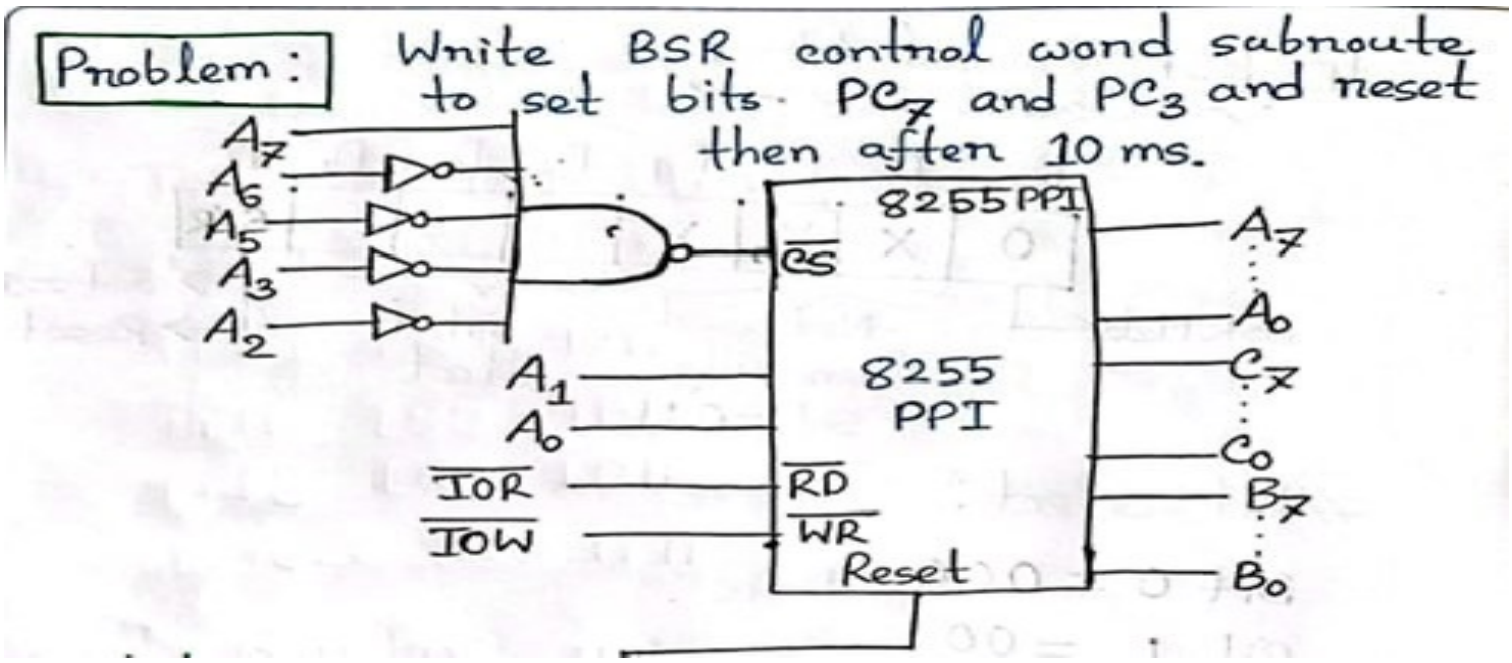
Control word :

	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
BSR Mode ←	0	x	x	x	Bit select.			I/O → Set = 1, Reset = 0

Bit select:

$PC_0 \rightarrow 000$
$PC_1 \rightarrow 001$
$PC_2 \rightarrow 010$
$PC_3 \rightarrow 011$
$PC_4 \rightarrow 100$
$PC_5 \rightarrow 101$
$PC_6 \rightarrow 110$
$PC_7 \rightarrow 111$

Problem: Write BSR control word subroute to set bits PC_7 and PC_3 and reset then after 10 ms.



⇒ Here, Port Addresses,

A_{15} ..... A_8	A_7	A_6 ... A_2	A_1	A_0	
0 0	1	0 0	0	0	= 0080H → PA
0 0	1	0 0	0	1	= 0081H → PB
0 0	1	0 0	1	0	= 0082H → PC
0 0	1	0 0	1	1	= 0083H → CR

BSR control word:

	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
To Set PC_7	0	0	0	0	1	1	1	1	= 0F H
To Reset PC_7	0	0	0	0	1	1	1	0	= 0E H
To Set PC_3	0	0	0	0	0	1	1	1	= 07 H
To Reset PC_3	0	0	0	0	0	1	1	0	= 06 H

Program:

```
PPI A   EQU  80H
PPI B   EQU  81 H
PPI C   EQU  82 H
PPI C_C EQU  83 H

MOV AL, 0F H   } set PC7
OUT PPI C_C, AL

MOV AL, 07H    } set PC3
OUT PPIC_C, AL

DELAY          } Timer of 10ms
MOV AL, 0E H   } Reset PC7
OUT PPIC_C, AL

MOV AL, 06 H   } Reset PC3
OUT PPIC_C, AL
```

Programmable Interval Timer 8254

Why 8254 used with 8086 MP?

→ It is not possible for 8086 to generate highly accurate time delay's due to processing variation, interrupt handling and other factors.

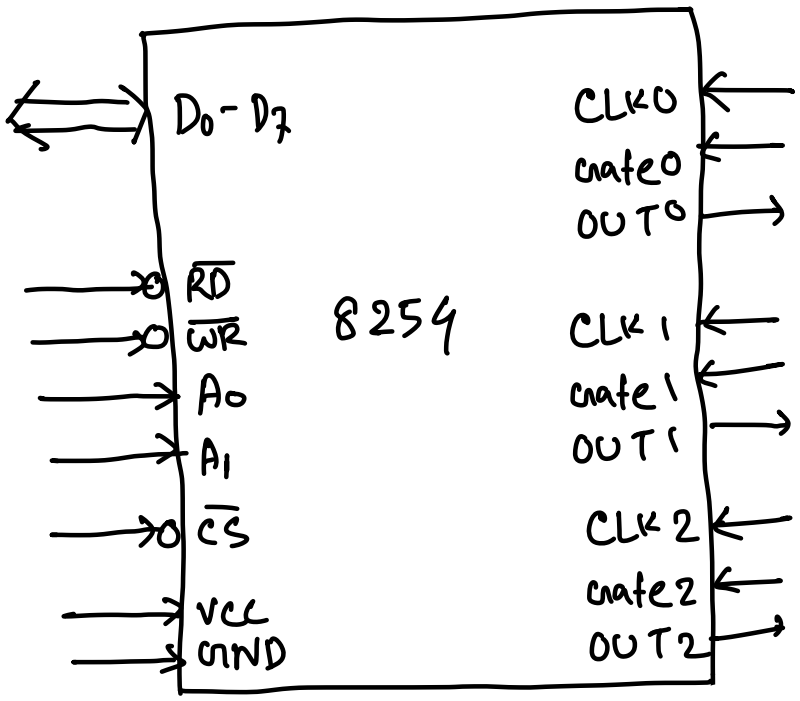
→ Intel 8254 is introduced to handle time delay's more accurately and efficienty with some key features:-

- Accurate time delay
- Minimizing load on MP 8086 from handling time delay manually
- Can function as a real time clock, providing consistent timing for several application
- Can work as an event counter
- Can work as digital one shot (create a single output pulse)
- Generate square wave of different frequencies.
- Can also generate complex wave generaton.

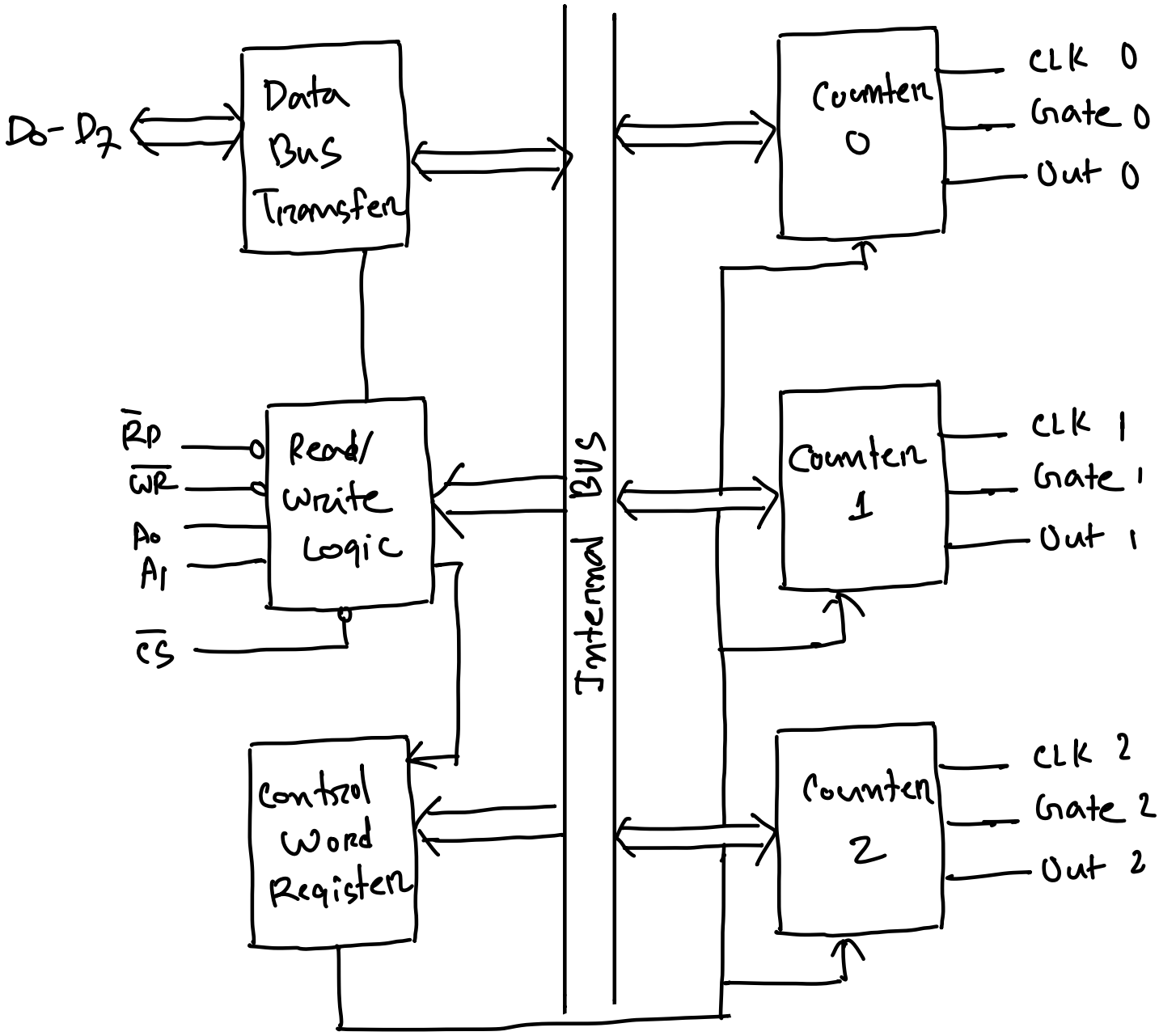
Difference between 8254 and 8253

8254	8253
(i) Can operate with compatively higher clock frequency range (Dc to 8MHz and 10MHz)	(i) Operate in comparitively lower clock frequency range (Dc to 2MHz)
(ii) Includes a status-Read-Back Command that can latch the count and status of the counter.	(ii) No status-Read-Back command
(iii) Provide accurate result	(ii) Doesn't provide accurate result
(iv) A new version of PJT	(iv) Old version PJT

Block Diagram and Pin Internal structure.



Block Diagram



→ 3 counters (C_0, C_1, C_2) works in 6 Modes.

Control Word Format

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SC_1	SC_0	RW_2	RW_1	M_2	M_1	M_0	BCD

SC_1 SC_0 — Select Counter

SC_1	SC_0	
0	0	→ Counter 0
0	1	→ " 1
1	0	→ " 2
1	1	→ Read Back Command

RW_1 RW_0 — Read/Write

RW_1	RW_0	
0	0	→ Counter Latch Command
0	1	→ Read/Write LSB
1	0	→ Read/Write MSB
1	1	→ Read/Write first LSB then MSB

M_2 M_1 M_0 — Mode

BCD — Use BCD or not

M Mode:

M_2	M_1	M_0	
0	0	0	→ Mode 0
0	0	1	→ Mode 1
X	1	0	→ Mode 2
X	1	1	→ Mode 3
1	0	0	→ Mode 4
1	0	1	→ Mode 5

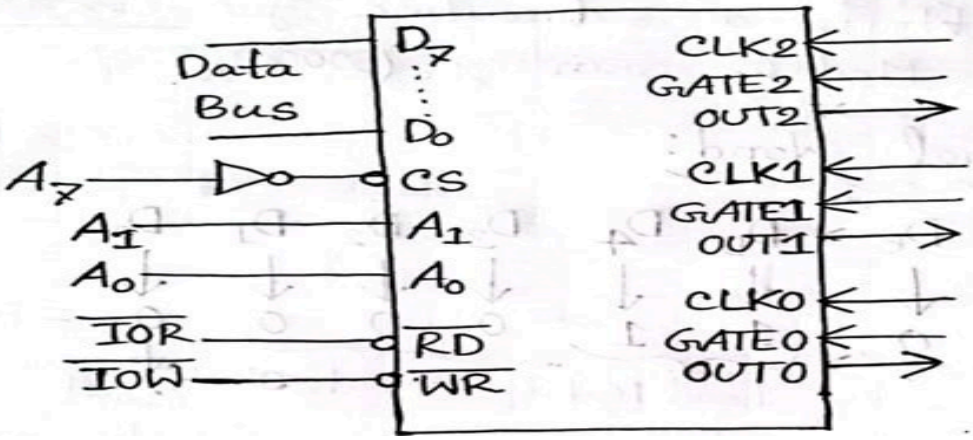
BCD

- 0 → Binary Counter (16 bit)
- 1 → Binary Coded Decimal Counter (BCD)

Port Address:

A_1	A_0	
0	0	→ Counter 0
0	1	→ " 1
1	0	→ " 2
1	1	→ Control Register

Question : The 8254 as a Counter.



(I) Identify the Port Address for Control Register and Counter-2?

(II) Write a program for counter-2 in Mode 0, count for $(50000)_{10}$ or Write a subroutine for counter-2 in Mode 0, counting $(50000)_{10}$

⇒ Port Address :-

A_{15} ... A_8	A_7	A_6 ... A_2	A_1	A_0		
0 ... 0	1	0 ... 0	0	0	→ 80H	→ Counter 0
0 ... 0	1	0 ... 0	0	1	→ 81H	→ " 1
0 ... 0	1	0 ... 0	1	0	→ 82H	→ " 2
0 ... 0	1	0 ... 0	1	1	→ 83H	→ Control Register.

So, Counter 2 = 82H ; Control Register : 83H

(11) Here, counter - 2 (1 0) in Mode 0 (0 0 0)

Count $(50000)_{10}$ = $(C3:50)_H$ ⇒ Count both MSB LSB

So (1 1)

MSB LSB

Control Word,

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

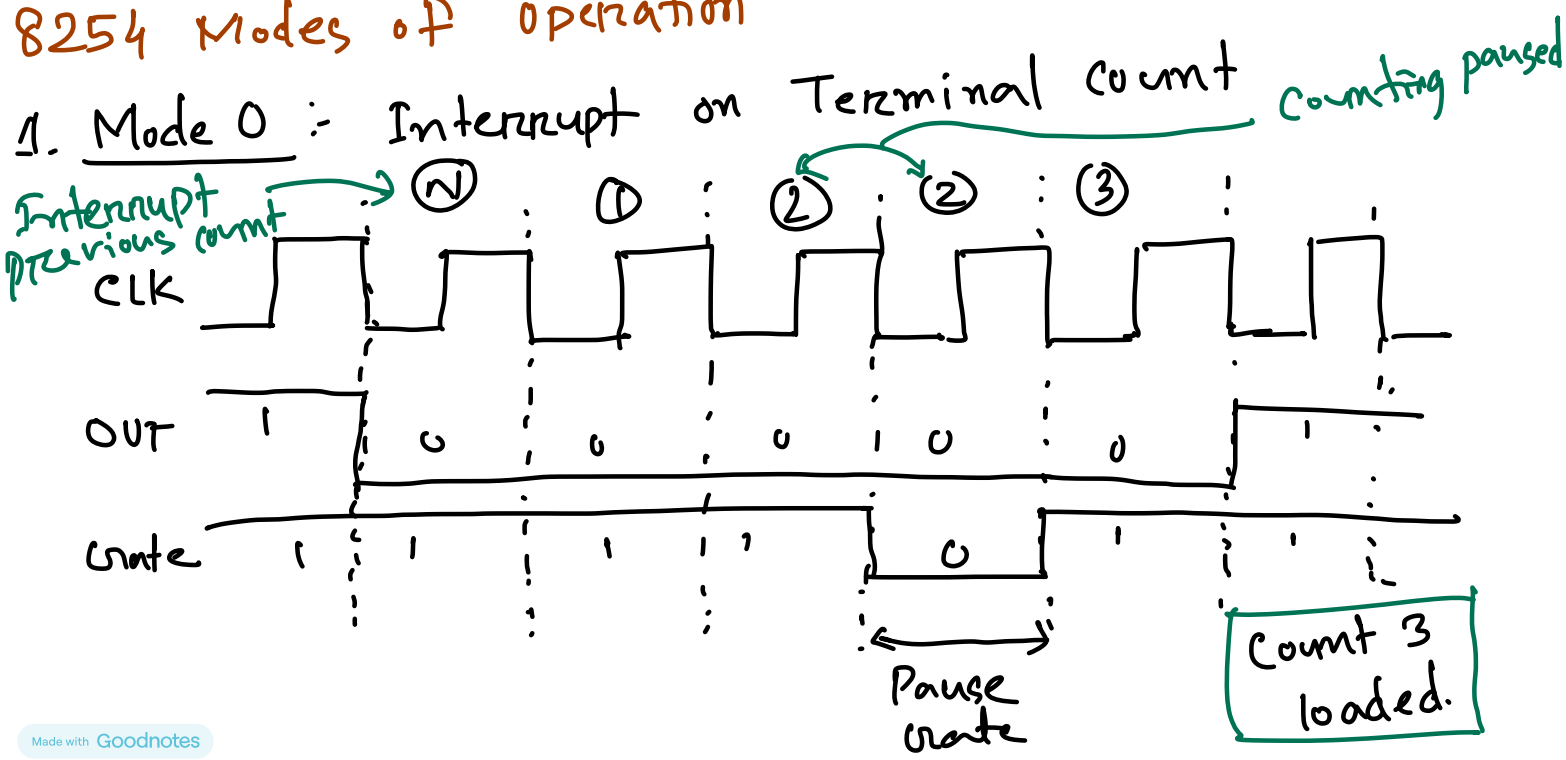
[BCD not] = B0H

Program:

```
MOV AL, B0H   } CW → CR
MOV DX, 83H   }
OUT DX, AL

MOV AL, 50H   } LSB OUT
MOV DX, 82H   }
OUT DX, AL

MOV AL, C3H   } MSB OUT
MOV DX, 82H   }
OUT DX, AL
```

8254 Modes of operation

1. Mode 0 : Interrupt on Terminal Count



→ Output remain low until counting is finished.

→ Count ⇒ start at the falling edge of clk signal

→ Output becomes high when full count is finished.

→ If one counting is progressed and another counting operation is given -

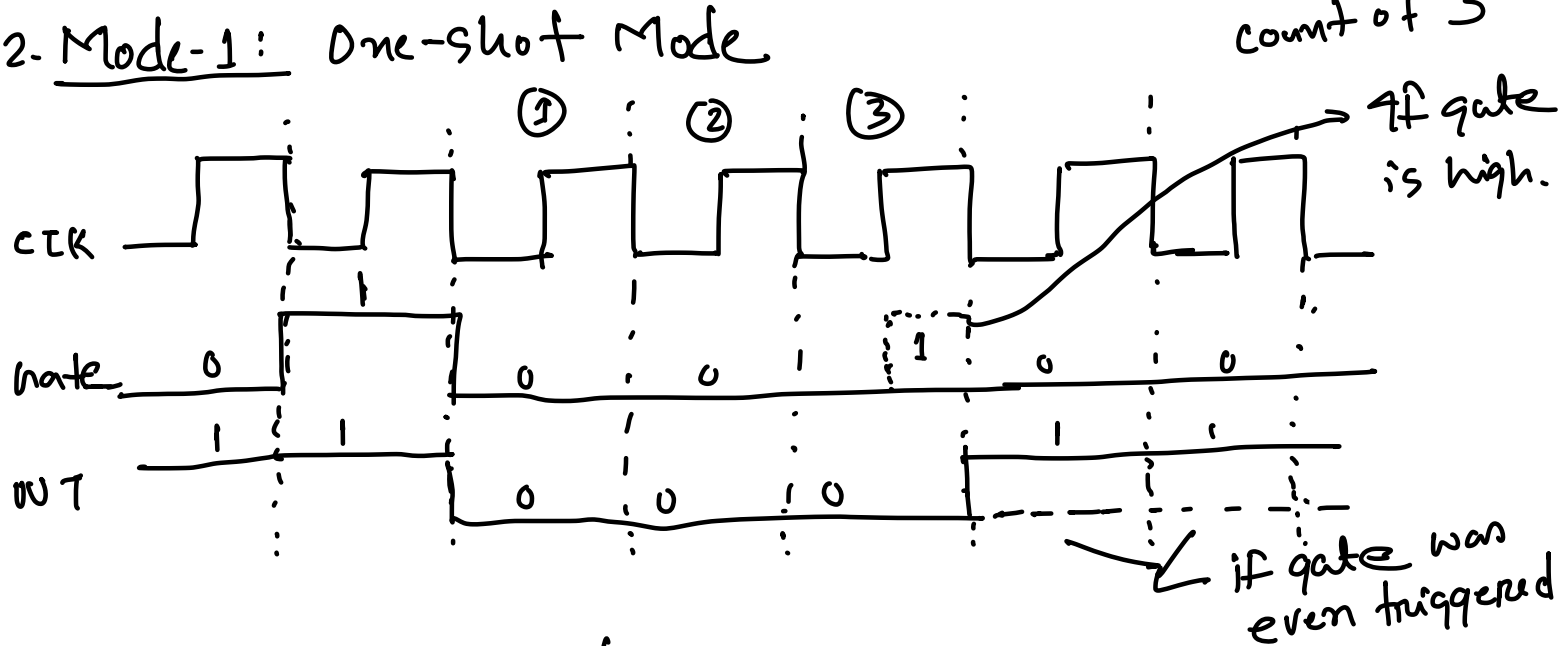
- first byte stops previous counting (N)
- from second byte new count starts.

Gate signal Effects:

→ Counting increase each cycle if Gate = 1

→ Counting is paused and current count is saved (latched) till Gate is high again.

2. Mode-1: One-shot Mode



→ Gate work as a trigger

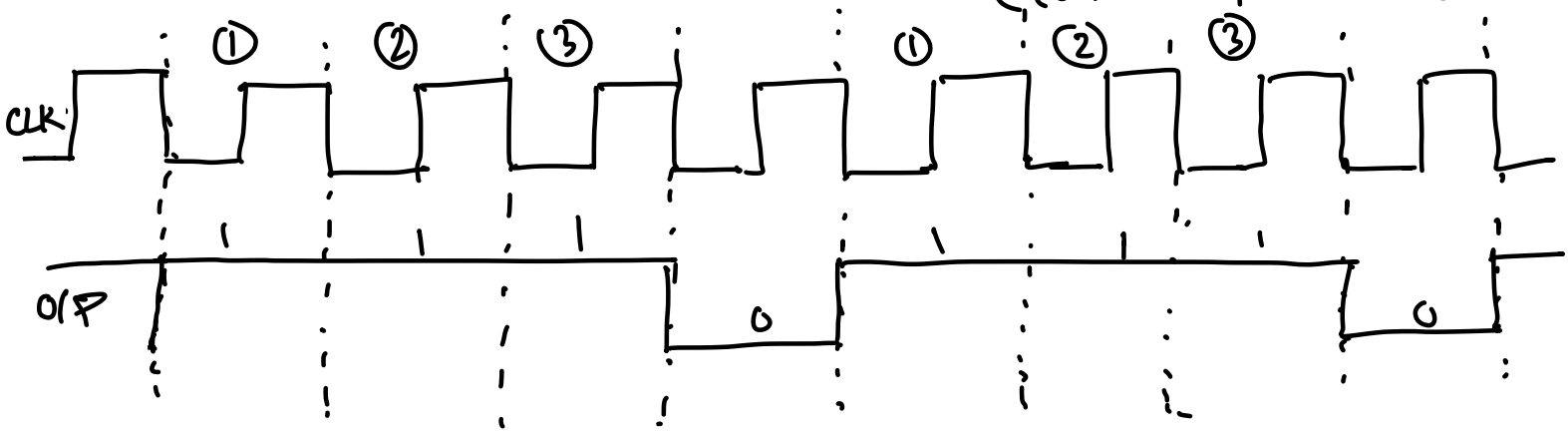
→ Gate high → counting start.

→ Another counting doesn't start or interrupt the current counting until Gate becomes high.

→ While counting output remains low

3. Mode – 3 :- Rate Generator / Divide by N counter.

(count repeatitively)



→ Let's assume 3 Pulses are to be count

→ For 1st 3clk cycle, counting continues
During this time output is high.

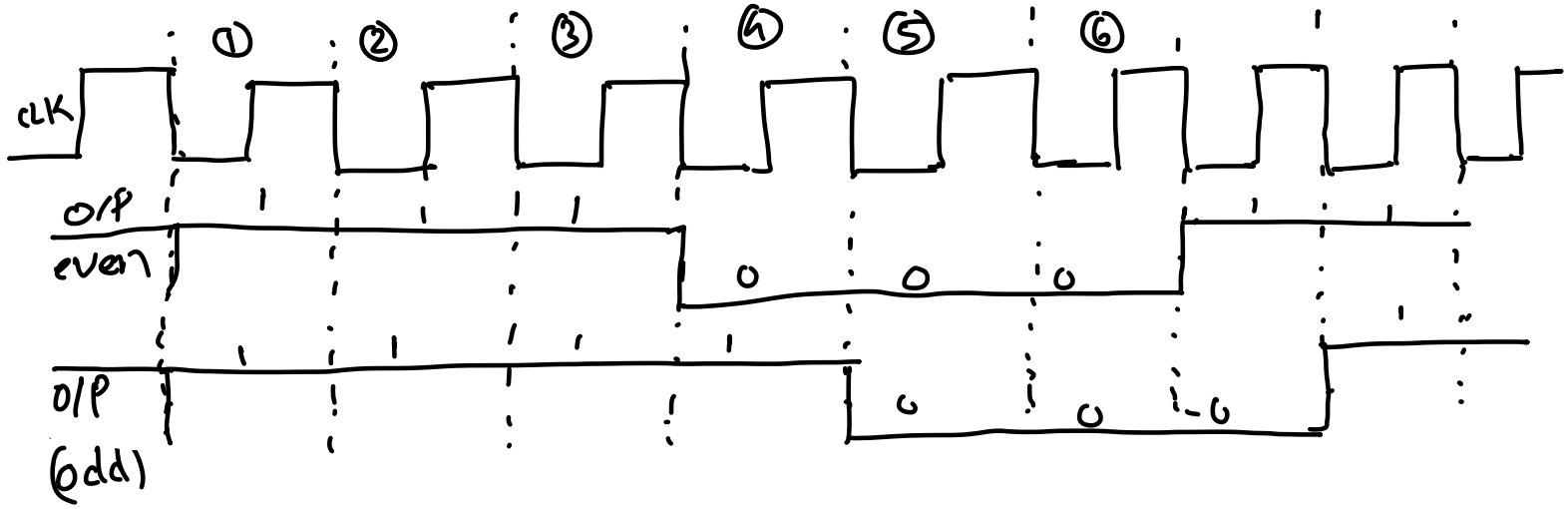
→ After that for 1 clk cycle, O/P is low.

→ After that for 3 clk cycle output is high
and counting start again

This happen repeatitively

→ Gate = 1 → Normal counting
Gate = 0 → No counting.

4. Mode 3: Square wave Generator



→ For Even (6) Counting: (assuming)

- → for 1st 3 clk signal (N/2) o/p high = 1
- → for Next 3 clk " (N/2) o/p low = 0
- → Then next 3 clk " (N/2) o/p high = 1

} This repeates

→ For odd (7) Counting:

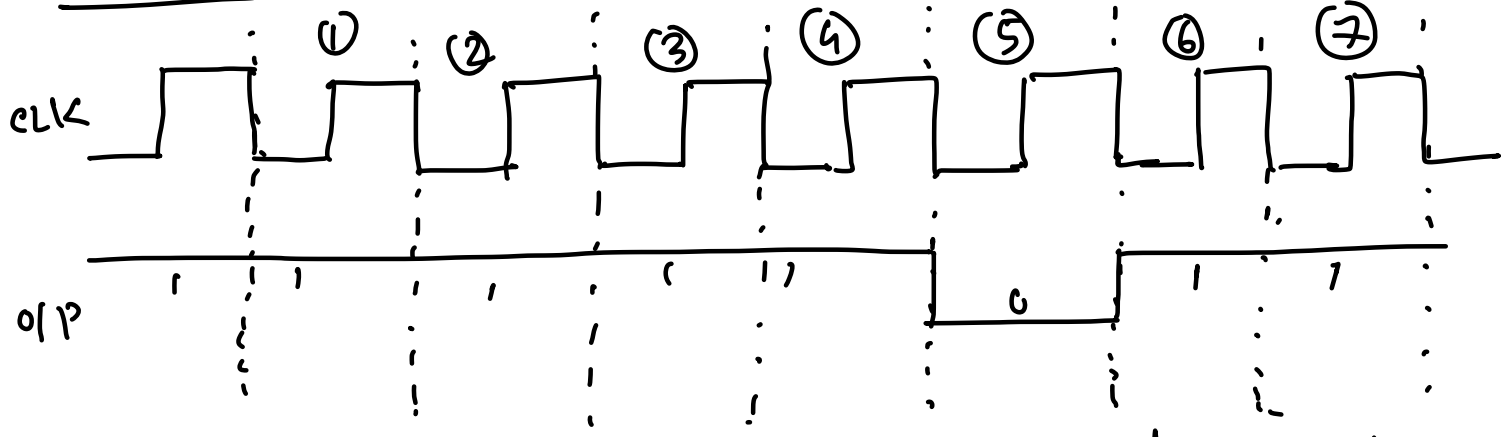
- → for 1st 4 clk $(\frac{N+1}{2})$ o/p high = 1
- → for Next 3 clk $(\frac{N-1}{2})$ o/p low = 1

} repeates

→ Gate always high

→ for even, duty cycle 50%. ; for odd, duty cycle > 50%

5. Mode-4: Software Triggered Strobe (5 count)



→ o/p remain set as soon as the mode is set.

→ When counted is loaded (here 5),

for first (N-1) (here, 5-1 = 4 clk cycle) the o/p will be high.

→ Then for next 1 cycle, o/p is low.

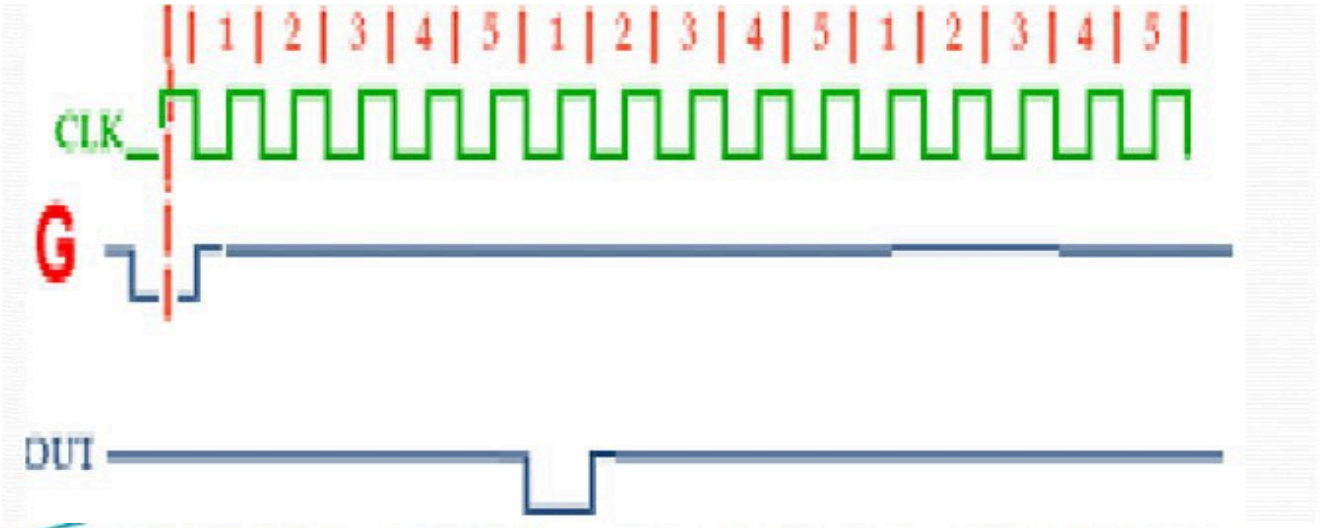
→ After this o/p will be high again.

⇒ The low pass we got (for 1 cycle) can be used as a strobe (a 'ready' signal type) for interfacing 8086 with peripherals

→ Gate high ⇒ Normal count

→ Gate low ⇒ Counting is kept hold (latched).

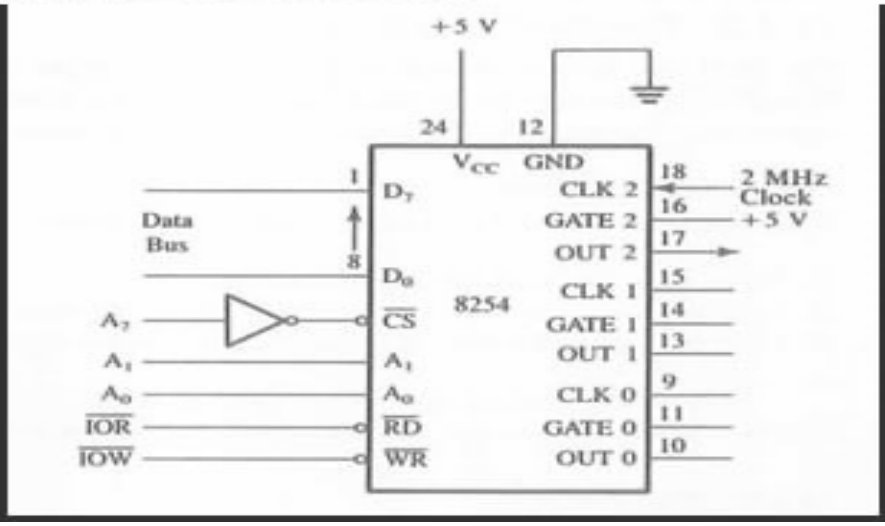
6. Mode-5: Hardware triggered strobe



- This mode generates a strobe in response to the rising edge at the trigger
- Mode is used to generate a delayed strobe in response to an externally generated signal
- Once mode is programmed and counter loaded, OUT goes HIGH
- Counter starts counting after the rising edge of the trigger (GATE)
- The OUTPUT goes LOW for one clock period, when the terminal count is reached
- Output will not go LOW until the counter content becomes zero after the rising edge of any trigger
- GATE is used as trigger input

Write instructions to generate a pulse every 50 µs from Counter 0 (refer to Figure 15.26). **Example 15.3**

To generate a pulse every 50 µs, from Counter 0, it should be initialized in Mode 2 (refer to Figure 15.27 for modes), and Gate 0 should be high. **Solution**



⇒ Here, referring to mode-2 and Counter-0

Now, clock frequency, $f = 2\,MHz = 2 \times 10^6$ Hz.

$$\therefore T = \frac{1}{f} = \frac{1}{2 \times 10^6} = 0.5\,\mu s.$$

So, we have to count, $= \frac{50}{0.5} = (100)_{10} = 64H$

As, 8 bit So, it is LSB. So, Read/Write operation for LSB.

∴ Control word :

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	0	0	1	X	1	0	0	= 14 H

Here, Port Address, Counter 0 = 80 H

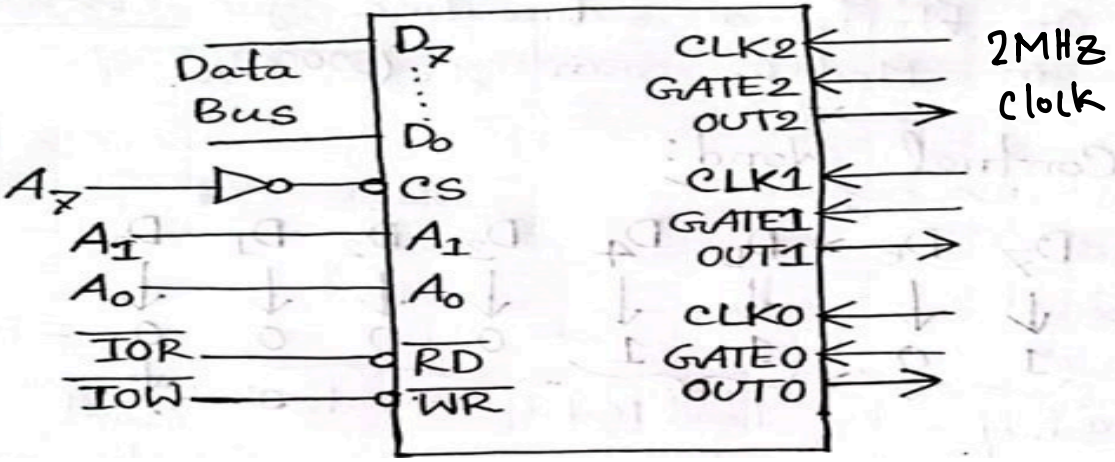
Control Register = 83 H [Same as Previous math]

Now, Program,

```
MOV AL, 14H
MOV DX, 83H
OUT DX, AL
MOV AL, 64H
MOV DX, 80H
OUT DX, AL
```

Example 15.4 Write instructions to generate a 1 kHz square wave from Counter 1 (refer to Figure 15.26). Assume the gate of Counter 1 is tied to +5 V through a 10 k resistor. Explain the significance of connecting the gate to +5 V.

Question : The 8254 as a Counter.



⇒ Here, Square Wave of 1KHz in Counter 1 to be generated,

So, Mode-3 and counter - 1 used.

Now, Clock, $T = \frac{1}{2 \times 10^6} = 0.5\ \mu s.$

Squarewave, $T = \frac{1}{1 \times 10^{-3}} = 1\ ms$

∴ We have to count, $\frac{1 \times 10^{-3}}{0.5 \times 10^{-6}} = (2000)_{10}$

$= (07:D0)\ H$

So, 16 bit, Both LSB and MSB.

Now, Control word,

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	1	1	1	x	1	1	0	= 76 H

From Previous,

Port Address, Counter 1 : 81 H

Control Register : 83H

Program:

```
MOV AL, 76H
MOV DX, 83H
OUT DX, AL

MOV AL, D01A
MOV DX, 811H
OUT DX, AL

MOV AL, 07H
MOV DX, 81H
OUT DX, AL
```

8254 Read Operation

There are three methods :-

(1) Simple Read operation :-

→ Simply read the content of the counter select by A_1 and A_0

→ Clock input of that counter must be turned off either using Gate pulse or external logic. Otherwise the count may be in the process of changing when it is read, giving an undefined result.

(2) Counter Latch Command :

→ SC_0, SC_1 selects one of the three counters

→ D_5, D_4 distinguish this command from a control word. $D_5, D_4 = 0\ 0$.

→ If a counter is latched, then sometimes later it is again latched before the count is read, the second counter latch command is ignored.

→ The count read will be the count at the time the first counter latch command was issued.

A_1, A_0 = 11; CS = 0; RD = 1; WR = 0

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SC1	SC0	0	0	X	X	X	X

SC1,SC0—specify counter to be latched

SC1	SC0	Counter
0	0	0
0	1	1
1	0	2
1	1	Read-Back Command

D5,D4—00 designates Counter Latch Command

X—don't care

NOTE:
Don't care bits (X) should be 0 to insure compatibility with future Intel products.

Problem: Write an instruction to read from counter 0 in Counter Latch Command.

⇒ Counter-0 : 80H. Control register : 83H.

Control word - D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0 = 00H.

0 0 0 0 X X X 0

Program:

```
MOV AL, 00H
MOV DX, 83H
OUT DX, AL
MOV DX, 80H
IN  AL, DX
```

Read command. ←

(3) Read-Back-Control Command:

→ Used to Read several counter at a time.

→ The counter is automatically unlatched during read while the other counter remain latched during they are read.

→ The read-back-control command is needed when it is necessary for the contents of more than one counter to be read at the same time.

→ Count :- logic 0, select one of the counter to be latched.

→ Status :- logic 0, status must be latched to be read status by a counter and it is accessed by a read from the counter.

A0, A1 = 11 $\overline{CS}$ = 0 $\overline{RD}$ = 1 $\overline{WR}$ = 0

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	1	$\overline{COUNT}$	$\overline{STATUS}$	CNT 2	CNT 1	CNT 0	0

D_5: 0 = Latch count of selected counter(s)
D_4: 0 = Latch status of selected counters(s)
D_3: 1 = Select Counter 2
D_2: 1 = Select Counter 1
D_1: 1 = Select Counter 0
D_0: Reserved for future expansion; Must be 0

Status register:

- Shows the state of the output pin
- Check the counter is in null state (0) or not
- How the counter is programmed

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Output	Null Count	RW1	RW0	M2	M1	M0	BCD

D_7 1 = OUT Pin is 1
0 = OUT Pin is 0

D_6 1 = Null Count
0 = Count available for reading

D_5–D_0 Counter programmed mode

Problem: Write a program to count the value of counter 0 and Read the latched status for counter 0

⇒ Here, counter 0 : 80H, (indicating read-back command)

control Register = 83 H

Control Word,

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
1	1	0	0	0	0	1	0	= C2H

D_5 ↓ latch count

D_4 ↓ latch status

↳ for counter 0

Program:

```
MOV AL, C2H   } CW → CR
MOV DX, 83H   }
OUT DX, AL

MOV DX, 80H   } Read
IN  AL, DX    } Latch status.

IN  AL, DX    } Reading LSB of Counter 0
MOV BH, AL    }

IN  AL, DX    } Reading MSB of Counter 1
MOV AH, AL    }
```

[I didn't note 8259]

[There might be writing mistakes. Do pardon that]