8086 Microprocessor

- The 8086 is a first 16-bit microprocessor developed by Intel (HIgh Performance bulK CMOS)
- It is designed using the HMOS technology and contain approximately 29,000 transistors.

Features -

1. It operates on single +5V power supply.
2. The 8086 has 20 bit address lines, hence it can address 1M byte memory locations.
3. It operates with 5MHz clock frequency.
4. It has 16 multiplexed address/data bus which reduces the number of pins.
5. The 8086 performs arithmetic, & logical operations on bit, byte, word including multiply and divide operations.
6. The 8086 can generate 16 bit I/O address hence it can access 2^{16} = 65535 I/O ports.
7. The 8086 work with register, direct, immediate, register indirect, based and indexed addressing modes.
8. It consist of four segments CS (Code Segment), DS (Data Segment), ES (Extra Segment) and SS (Stack Segment)
9. The 8086 supports multiprogramming.

Architecture of 8086

The 8086 is internally divided into two separate functional units. These are Bus Interface Unit (BIU) & Execution Unit(EU)

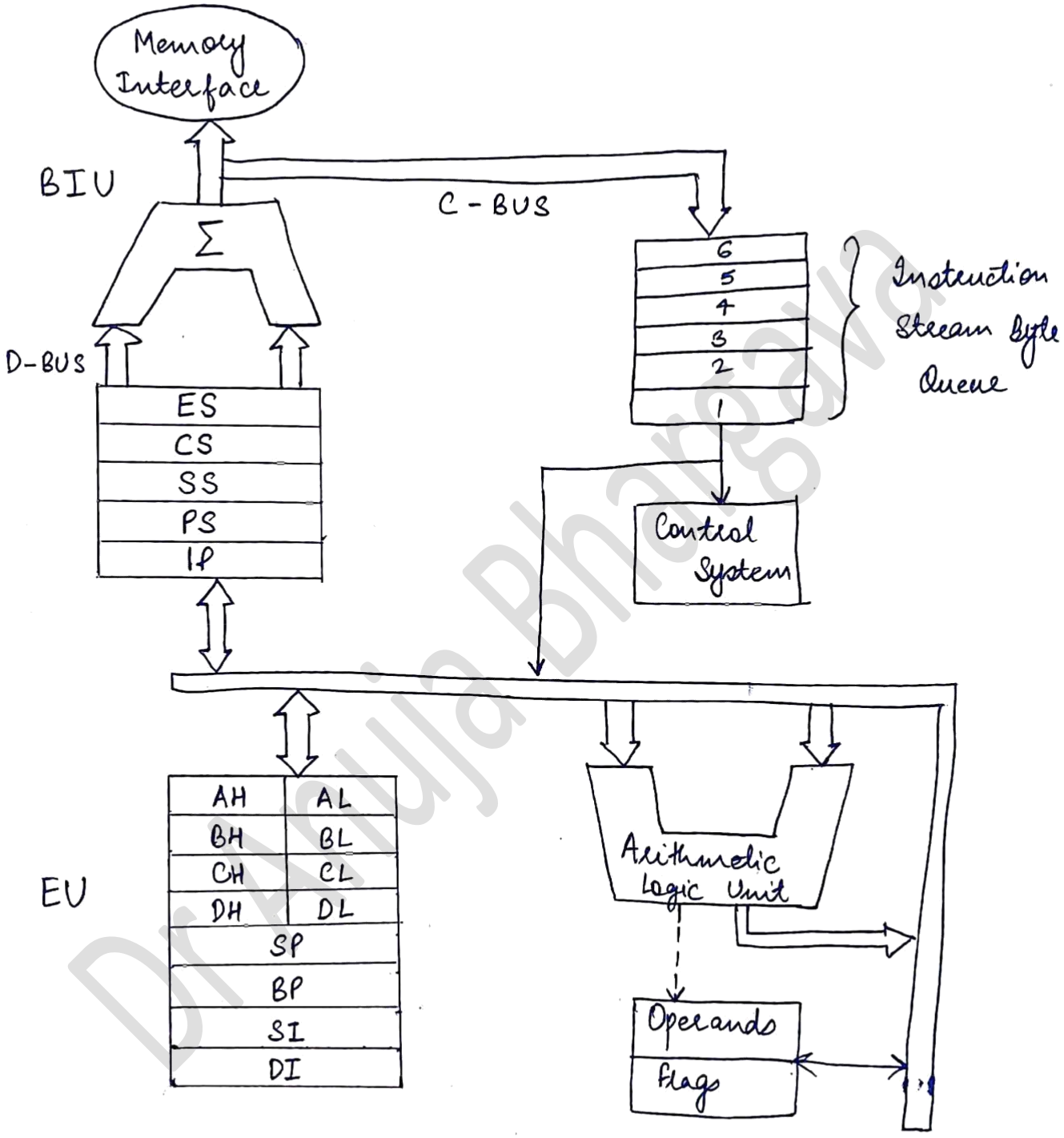


Fig : Internal Architecture of 8086 Microprocessor

1. Bus Interface Unit (BIU):

- The BIU fetches instruction, reads data from memory and ports, and writes data to memory and I/O ports.
- The EU executes instructions that have already been fetched by the BIU.
- The BIU and EU work independently.
- The BIU handles all transfer of data and addresses on the buses for the execution unit.
- The BIU consist of instruction pointer, segment register, instruction queue and address generation / bus control circuitary to provide functions such as fetching and queuing of instruction and bus control.

(i) The Queue

→ The queue is a first in first out group of register in which upto 6 bytes of instruction code are prefetched from memory ahead of time.

→ The BIU can be fetching instruction bytes while the EU is decoding an instruction which does not required use of buses.

→ When EU is ready for its next instruction, it simply reads the instruction from the queue in BIU.

→ If an instruction such as jump or CALL encountered, the BIU will reset the queue and began refilling after passing the new instruction to the EU.

→ The process of fetching next instruction while current instruction is executing is known as pipelining. This scheme greatly speeds up processing.

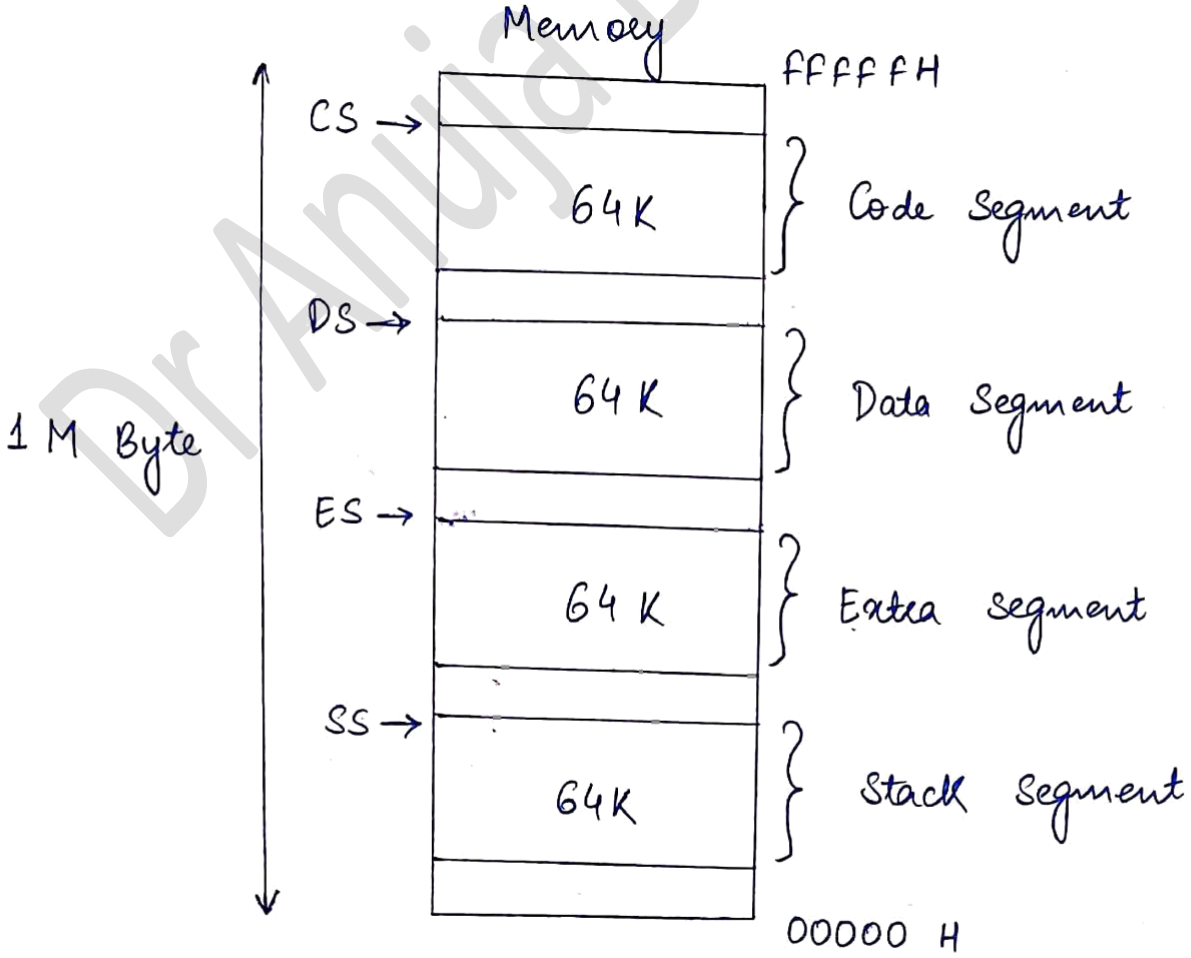
(ii) Segment Register :

→ The BIU has four 16 bit registers. These are Code Segment (CS), Data Segment (DS), Stack Segment (SS) and Extra Segment (ES) registers.

→ The 8086 uses memory segmentation

★ 1 M bytes of memory is divided into segments with maximum size of segment 64 K byte.

★ The 8086 can directly address four segments at a particular time.



* Thus a location with in a segment can be addressed using 16 bits.
* These segment register used to hold the upper 16 bits of the starting address of four memory segments.
* The part of segment starting address stored in segment register is called as segment base.

Code Segment (CS) Register

→ The code segment holds the program code.
→ The upper 16 bits of the starting address of coded segment is loaded in CS Register.
→ CS register is of 16 bit wide.

eg If CS = 3456 H, that means code segment starts from Memory location 34560 H (3456 * 10), which is of 20 bit address.

Data Segment (DS) Register

→ During program execution, the microprocessor access a memory location for read or write operation.
→ In 8086, there will be a separate data segment where the data will reside.
→ The upper 16 bits of the starting address of data segment is loaded in DS register.
→ The data segment holds the data constraint needed by the program.

Stack Segment (SS) Register

→ The upper 16-bits of the starting address of Stack Segment is loaded in SS Register.

(iii) Instruction Pointer (IP)

→ It points to the instruction to be fetched and executed next.

→ The IP register holds the 16-bit address of the next code byte with in the code segment.

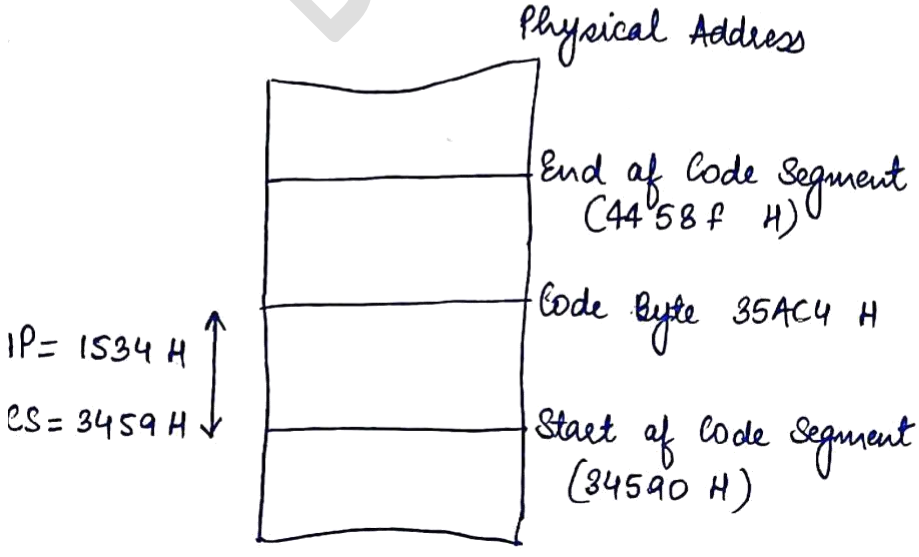
→ The value contained in IP is referred as offset, because the value must be offset from (added to) the segment base address in CS to produce the required 20-bit physical address.

Generation of 20-bit address ★

How 20 bit address is generated by BIU using the 16 bit IP.

Assume that IP content is 1534 H. This is called offset value or effective address.

Let the content of CS be 3459 H. This is called segment base value.



$$\begin{array}{r} CS \quad 34590 \\ IP + \quad 1534 \\ \hline 35AC4 \end{array}$$

- The CS Register points to the base or start of current code segment. The IP contains the offset from base address to the next instruction byte to be fetched.
- Here, the content of CS register are shifted left four bit position before the content of IP are added to it.

Eg Content of CS = 3459 H

Shifted by four position = 34590 H

∴ Code Segment starts at 34590 H

Now, fetch an instruction from memory location which is 1534 H bytes away from the beginning of code segment.

∴ 34590 H + 1534 H (using adder in BIU)

sends out as 35AC4 H as physical address on 20-bit address pins.

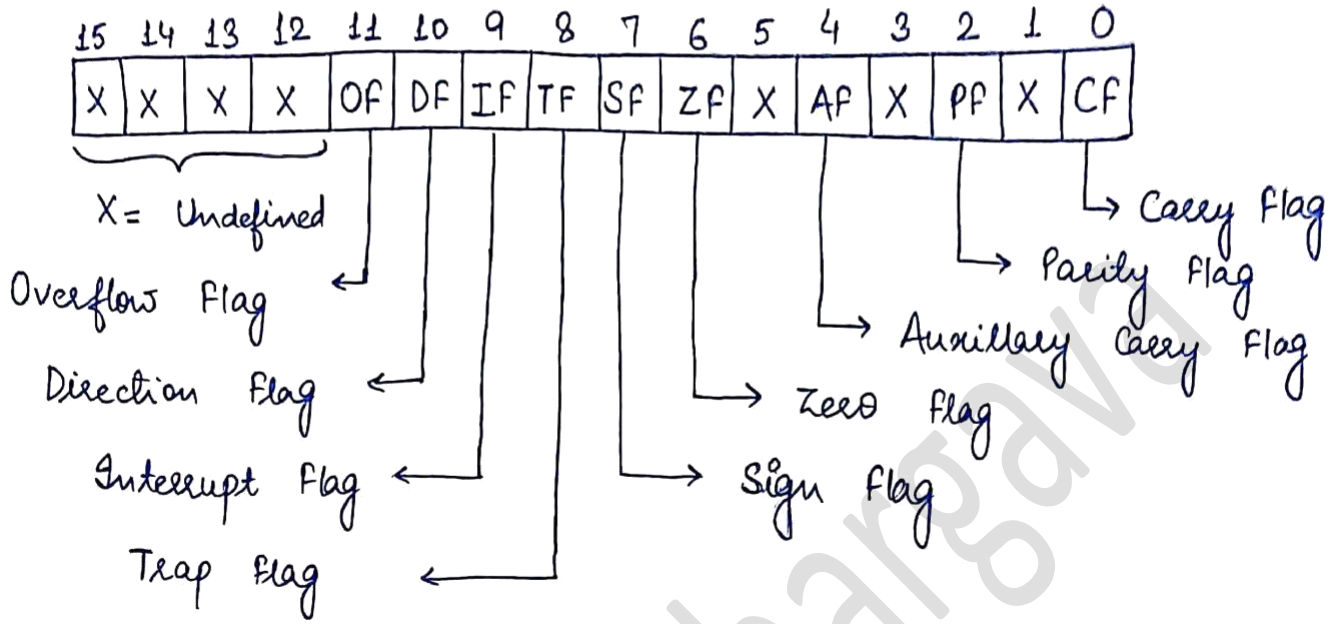
- To access data in memory the 8086 must produce a 20 bit physical address. It does this by adding a 16-bit value called the effective address to one of the four segment base.

(2) Execution Time :- (EU)

- It is responsible for executing the instruction fetched by BIU.
- It consist of Instruction Decoder (ID), Control Unit (CU), Arithmetic Logic Unit (ALU), general purpose register, flag register and other registers.

Flag Register :-

- The 8086 PSW contains 16 bits, but 7 of them are not in used. Each bit in PSW is called a flag.
- A flag is a flip-flop which will be in 1 or 0 state.



- The 8086 flags can be divided into two types
  (1) Conditional flag (2) Control flag

(1) Conditional Flag

→ It stores the data condition according to the result of the previous operation in ALU.

→ There are six conditional flags

(i) CF (Carry Flag)

It is set to 1, if there is a carry out of MSB during addition operation and borrow in overall subtraction operation

(ii) ZF (Zero Flag)

It is set to 1, whenever result stored in register or memory location is zero.

(iii) AF (Auxilliary Carry Flag)

It is set to 1, if there is a carry out of bit 3 during addition or borrow during subtraction.

(iv) SF (Sign Flag)

It is set to 1, if the result is negative.

It is set to 0, if the result is positive.

Note :- This flag is meaningful only if we are working with signed number.

(v) PF (Parity Flag)

After arithmetic operation, the 8086 count the numbers of 1's in the result.

It is set to 1, if there are even number of 1's.

It is set to 0, if there are odd numbers of 1's.

(vi) OF (Overflow Flag)

It is set to 1, if overflow occurs i.e result is out of range. The processor gives a wrong answer sometimes, when it is working with signed numbers. If we add two positive numbers, the result is expected to be positive. But in some cases result is negative, which indicates wrong answer. In such cases OF is set to 1.

(2) Control Flag

→ It is used to control certain operations of the processor.

→ There are three control flags

(i) IF (Interrupt Enable Flag)

→ If it is set to 1, a maskable interrupt can be recognized by the CPU, otherwise these interrupts are ignored.

→ The INTR is maskable interrupt of 8086.

→ If the IF is reset to 0, the 8086 does not respond to the request on INTR.

→ If the IF is set to 1, the 8086 responds to the request on INTR pin.

→ The IF bit has no effect on NMI, which is a non-maskable interrupt.

(ii) TF (Trap Flag)

→ When this flag is set to 1, the 8086 enters in a 'single stepping' mode.

→ To detect the error in the programs, it is necessary to run the programs one instruction at a time. This process is called 'single stepping' through a program.

→ But there is no instruction in 8086 to directly set the TF flag to 1, or reset the TF flag to 0.

→ The instructions given below are used to set TF to 1.

```
PUSH F              : Push flag on stack
MOV BP, SP          : Copy SP to BP
OR WORD PTR, 0100H: Set TF bit
POP F               : Restore flag register
```

→ The instruction given below are used to reset TF to 0.

PUSH F	:	Push flags on stack
MOV BP, SP	:	Copy SP to BP
AND WORD PTR, 0FEFF H	:	Reset TF bit
POP F	:	Restore flag register

(iii) DF (Direction Flag)

→ It is used by string manipulation instruction.

→ The content of SI (Source Indexed Register) and DI (Destination Indexed Register) is automatically incremented or decremented by setting the DF.

→ Automatic increment of SI and DI is done if DF is 0.
Automatic decrement of SI and DI is done if DF is 1.

Numerical.

eg1 Calculate flag status.

1) ADD AX, BX

Before Execution

AX = 2345 H 0010 0011 0100 0101

BX = 3219 H + 0011 0010 0001 1001

0101 0101 0101 1110

After Execution

AX = 555E H

BX = 3219 H

SF = 0 , ZF = 0 , PF = 0 , CF = 0 , AF = 0 , OF = 0

2) ADD BX, CX

Before Execution

BX = 1234 H

CX = 7F2E H

```
  0001  0010  0011  0100
+ 0111  1111  0010  1110
-----------------------
  1001  0001  0110  0010
```

After Execution

BX = 9162 H

CX = 7F2E H

SF = 1, ZF = 0, PF = 0, CF = 0, AF = 1, OF = 1

3) ADD BL, CH

Before Execution

BL = 04 H

CH = FC H

```
    0000  0100
+   1111  1100
-------------
1   0000  0000
```

After Execution

BL = 00 H

CH = FC H

SF = 0 , ZF = 1 , PF = 1, CF = 1 , AF = 1, OF = 0

General Purpose Registers

- The EU has four general purpose registers labelled as AX, BX, CX and DX. These registers are used to store 16 bit data.
- These registers are divided into two 8 bit portions.
  - AX is divided as AH and AL.
  - BX is divided as BH and BL.
  - CX is divided as CH and CL.
  - DX is divided as DH and DL.

Other Pointers

SP - Stack Pointer

It points to the top of the stack and is used for stack operation.

BP - Base Pointer

It is used as base register for accessing the stack memory

Advantage of Segment Register

The advantage of using segment register for segmenting 1 M byte memory space are as follows

(1) It facilitates the use of separate areas of a program code, its data and stack. So that program will be fetched from CS, while program data will be stored in ES and DS.

(2) Since SS area of memory is different from CS and DS, there is no possibility of stack getting overlapped.

(3) It has Multiprogramming environment.

(4) It allows memory capacity of 1M byte even though the address associated with individual instructions are only 16 bit.

(5) By structuring memory into separate area for separate operations, the programs are shorter, faster and more structured.

8086 Addressing Modes :-

- The way in which an operand is specified in the instruction or the various methods of accessing data in the execution of an instruction is called addressing modes.
- The 8086 addressing modes can be classified into five major categories.

1. Immediate Addressing Mode.

→ 8 or 16 bit data can be specified as part of instruction. The immediate operand can only be the source operand. Operand is specified with in the instruction itself.

eg (i) MOV CL, 03H : Move the 8-bit data 03H in CL.

(ii) MOV DX, 1234H : Move the 16-bit data 1234H in DX.

(iii) ADD AL, 03H : The 8-bit data 03H is added with AL and the result is stored in AL.

(iv) ADD AX, 1234H : The 16-bit data 1234H is added with AX and result is stored in AX.

2. Register Addressing Mode

→ The source and destination operand are specified in registers.

→ The operand can be 8-bit or 16-bit wide.

b) Indirect addressing mode.

→ In this, effective address is calculated from the content of registers.

→ IAM, subdivided into five categories

(i) Register Indirect Addressing Mode.

In this mode, EA is provided in an index register.

eg: MOV [DI], 1234 H : destination operand is a memory location specified using register indirect add. mode.

MOV AX, [BX] : source operand is a memory location specified using register indirect addressing mode.

(ii) Based Addressing with Displacement Mode.

In this mode, EA is sum of 8 or 16 bit displacement and the content of base register.

$$EA = \begin{Bmatrix} (BX) \\ (BP) \end{Bmatrix} + \begin{Bmatrix} \text{8 bit displacement} \\ \text{16 bit displacement} \end{Bmatrix}$$

eg: MOV [BX-2], 1234 H :

MOV AX, [BX + 200]

MOV [BX-5], AX

(iii) Indexed Addressing with Displacement Mode

In this case, effective address is sum of 8 or 16 bit displacement plus the content of index register (SI or DI)

eg MOV [DI + 2345], 1234

MOV AX, [SI +45]

(iv) Based Indexed Addressing Mode

In this, effective address is sum of base register (BX or BP) and indexed register (SI or DI) both of which are specified in instruction.

eg MOV [BX + DI], 1234

MOV AX, [SI +BX]

(v) Based Indexed Addressing with displacement mode

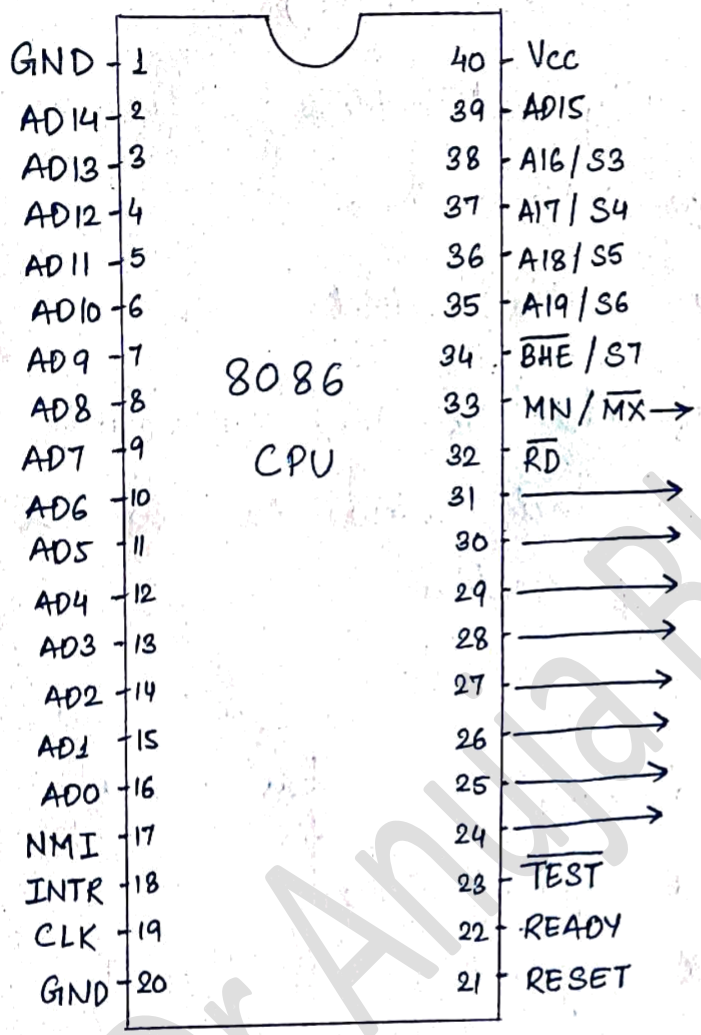
In this case, effective address is sum of 8 bit or 16 bit displacement plus base register (BX or BP) plus index register (SI or DI).

eg MOV [DI + BX + 37 H], AX

MOV AL, [BX + SI + 278H]

8086 Pin Configuration

- The 8086 have a 20-bit address bus, 16-bit data bus, three power supply pins and 17 pins devoted to control and timing functions.



Minimum Mode	Maximum Mode
HOLD	RQ/GT0
HLDA	RQ/GT1
WR	LOCK
M/IO	S_2
DT/R	S_1
DEN	S_0
ALE	QS0
INTA	QS1

- 8086 has ability to operate in two modes

(i) Minimum Mode (Uniprocessor System)

(ii) Maximum Mode (Multiprocessor System)

Minimum mode is used for small system in which 8086 generates all necessary signals. Maximum mode is used for medium to large system which includes two or more processor

1) Address / Data Bus ($AD_0 - AD_{15}$)

- $AD_0 - AD_{15}$ lines are 16-bit multiplexed address.
- During T_1 state, $A_0 - A_{15}$ contain address.

2) Address / Status lines

(A_{16}/S_3, A_{17}/S_4, A_{18}/S_5, A_{19}/S_6)

- $A_{16}/S_3 - A_{19}/S_6$ lines are multiplexed address / status bus.
- During T_1 state, they serve as address lines.

During $T_2 - T_4$ state, these are status output lines.

Note: The 20 pins ($AD_0 - AD_{15}$ and $A_{16} - A_{19}$) corresponds to the CPU's 20 bit address bus and allow the processor to access 1,048,576 unique memory location.

- Characteristics of status lines

S_4	S_3	Characteristic
0	0	Extra Segment Access
0	1	Stack Segment Access
1	0	Code Segment Access
1	1	Data Segment Access

S_5 - Indicates the status of interrupt enable flag.

S_6 - It is always zero, and no significance is attached to it.

3) Bus High Enable / Status ($\overline{BHE}$ / S_7)

- The 8086 outputs a low on this pin during read / write and interrupt acknowledge cycles in which data are to be transfered from an odd address in higher order byte (AD_{15} - AD_8) of the data bus or word transfer take place from even address on AD_{15} - AD_0.
- $\overline{BHE}$ can be used with AD_0 to select even / odd memory or I/O ports.

Operation	$\overline{BHE}$	A_0
1. Write / Read a word at even address	0	0
2. Write / Read a byte at even address	1	0
3. Write / Read a byte at odd address	0	1
4. Write / Read a word at odd address	0	1

4) NMI : Non Maskable Interrupt

- It is positive edge triggered interrupt.
- The NMI has higher priority than INTR.
- It is a vectored interrupt, that means 8086 knows where to branch, to service NMI interrupt.

5) INTR : Interrupt:

- It is a maskable interrupt input.
- It can be enable by STI (Set interrupt flag) and disabled by CLI (Clear interrupt flag) instruction.

6) CLK : Clock :

- The 8086 comes in different clock frequency in the range between 2 to 8 MHz,

Version	Frequency
8086-1	10 MHz
8086-2	8 MHz
8086	5 MHz

7) Reset :

- It terminates the processor activity.
- It can be done by reset button or power on button.

8) Ready :

- This pin is used for interfacing peripherals with processor.
- It is acknowledgement signal from memory or I/O interface that the CPU can complete the current bus cycle.

9) $\overline{TEST}$:

- It is used to synchronize the 8086 with external hardware such as coprocessor.
- It is examined by the CPU during the 'wait' instruction in multiprocessing environment.

10) $\overline{RD}$: (Read)

- It is active low signal when 8086 is reading data from memory or I/O location.

11) V_{CC} :

Supply Voltage $\pm 5V \pm 10\%$

12) GND (Ground)

2 Pins are used for ground.

13) MN / $\overline{MX}$ (Minimum / Maximum Mode)

It is high for minimum mode and low for maximum (Uniprocessor)

(Multiprocessor) mode operation.

Pins for Minimum Mode :-

1) $\overline{INTA}$ (Interrupt Acknowledge)

- It indicates recognition of an interrupt request.

2) ALE (Address latch Enable)

- It is used to demultiplex $AD_0 - AD_{15}$ into $A_0 - A_{15}$ and $D_0 - D_{15}$. It indicate address is available on the pin.

3) $\overline{DEN}$ (Data Enable)

- It is low during memory or I/O read / write.
- It becomes low, little after ALE is made 0, so that the data on the data pins are settled.

4) DT/$\overline{R}$ (Data Transmit / Receive)

- It is an output signal controls the direction of data flow through transceivers.
- If it is 1, the buffer is used to transmit the data.
  If it is 0, the buffer is used to receive the data.

5) M / $\overline{IO}$: (Memory / I/O Operation)

- For memory operation M / $\overline{IO}$ = 1
  I/O operation M / $\overline{IO}$ = 0

6) $\overline{WR}$ (Write):

- It is used to indicate that processor is performing write Memory or write I/O operation.

7) HOLD & HLDA (Hold Acknowledge)

- The 8086 processor is in HOLD situation, whenever DMA is required to access directly.
- The microprocessor indicates to the DMA controller that it has entered the hold state, by sending HLDA signal.

Pins for Maximum Mode

1) QS_1, QS_0 (Queue Status)

- It reflect the status of the instruction queue.

QS_1	QS_0	Interpretation
0	0	No operation
0	1	First byte of instruction is executed.
1	0	Queue is empty.
1	1	Next byte of instruction is executed.

2) $\bar{S}_0$, $\bar{S}_1$, $\bar{S}_2$ (Status Signal)

- These are used for timing and control signals.

$\bar{S}_2$	$\bar{S}_1$	$\bar{S}_0$	Function
0	0	0	Interrupt Acknowledge
0	0	1	Read I/O Port
0	1	0	Write I/O Port
0	1	1	Halt
1	0	0	Instruction fetch
1	0	1	Read Memory
1	1	0	Write Memory
1	1	1	Inactive

3) $\overline{LOCK}$:

- It is active low signal used to prevent other processor from getting access to the system bus.

4) $\overline{RQ}$ / $\overline{GT_0}$ & $\overline{RQ}$ / $\overline{GT_1}$: (Request & Grant Lines)

- It is used to force the processor to release the local bus at the end of processor cycle.
- $\overline{RQ}$ / $\overline{GT_0}$ having higher priority than $\overline{RQ}$ / $\overline{GT_1}$
- The request / grant function of 8086 are as follows

(i) A pulse from another local bus master ($\overline{RQ}$ / $\overline{GT_0}$ or $\overline{RQ}$ / $\overline{GT_1}$ pins) indicates a local bus request to the 8086.

(ii) At the end of 8086 current bus cycle, a pulse from 8086 to the requesting master indicates that the 8086 has the system bus. and tristated the outputs.

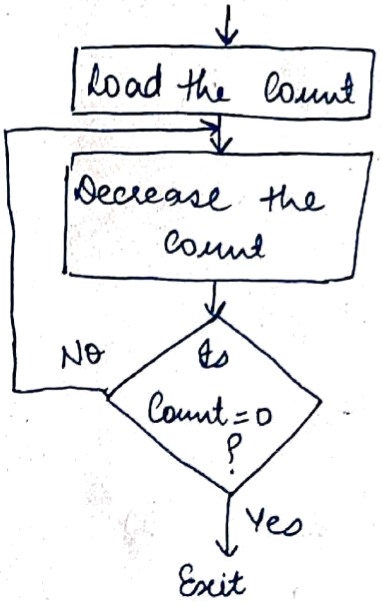
Then the new bus master sends a low signal on $\overline{RQ}$ / $\overline{GT_0}$ or $\overline{RQ}$ / $\overline{GT_1}$ pins.

The 8086 then regains bus control.

Counters and Time Delay

- Counters are used primarily to keep track of events.
- Time delay is used to set timming between two events.

Using One Register




```
        MVI C, FF H
loop    DCR C
        JNZ loop
```

	How many times the instruction is executed	States
MVI C, FF H	1	7 X 1
DCR C	255	4 X 255
JNZ loop	255	10 X 255

How 255 ?

$$(FF)_H = 15 \times (16)^1 + 15 \times (16)^0$$
$$= 255$$

Total States = (7 X 1) + (4 X 255) + (10 X 255)

= 3577 states

Let 8085 microprocess f = 2 MHz

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

$$\therefore \text{Delay time} = 3577 \times 0.5 \times 10^{-6}$$
$$= 1788.5\ \mu sec.$$
$$= 1.78\ m\ sec$$

Using two registers (Loop with in loop)

		How many times the instr is executed	States
	MVI B, 10H	1	7 X 1
loop1	MVI C, 78H	16	7 X 16
loop2	DCR C	120 X 16	4 X 120 X 16
	JNZ loop 2	120 X 16	10 X 120 X 16
	DCR B	16	4 X 16
	JNZ loop 1	16	10 X 16

How 16? $(10)_{16} = 1 \times (16)^1 + 0 \times (16)^0$

$= (16)_{10}$

How 120? ~~$(120)_{16} = 1 \times (16)^2 + 2 \times (16)^1 + 0 \times (16)^0$~~

$(78)_{16} = 7 \times (16)^1 + 8 \times (16)^0$

$= (120)_{10}$

Total States = $(7X1) + (7X16) + (4X120X16) + (10X120X16) + (4X16) + (10X16)$

$= 27223$ states

$\therefore$ Delay time $= 27223 \times 0.5$

$= 13611.5$ μ sec.

Note: To give maximum delay, both registers must be loaded with maximum value i.e FF H.

Using Register Pair

	T-State	How many times loop executed
LXI D, FFFF H	10	1
loop: DCX D	6	65535
MOV A, D	4	65535
ORA E	4	65535
JNZ loop	10	65535

Total State = $(10 \times 1) + (6 + 4 + 4 + 10) \times 65535$

$= 1572850$

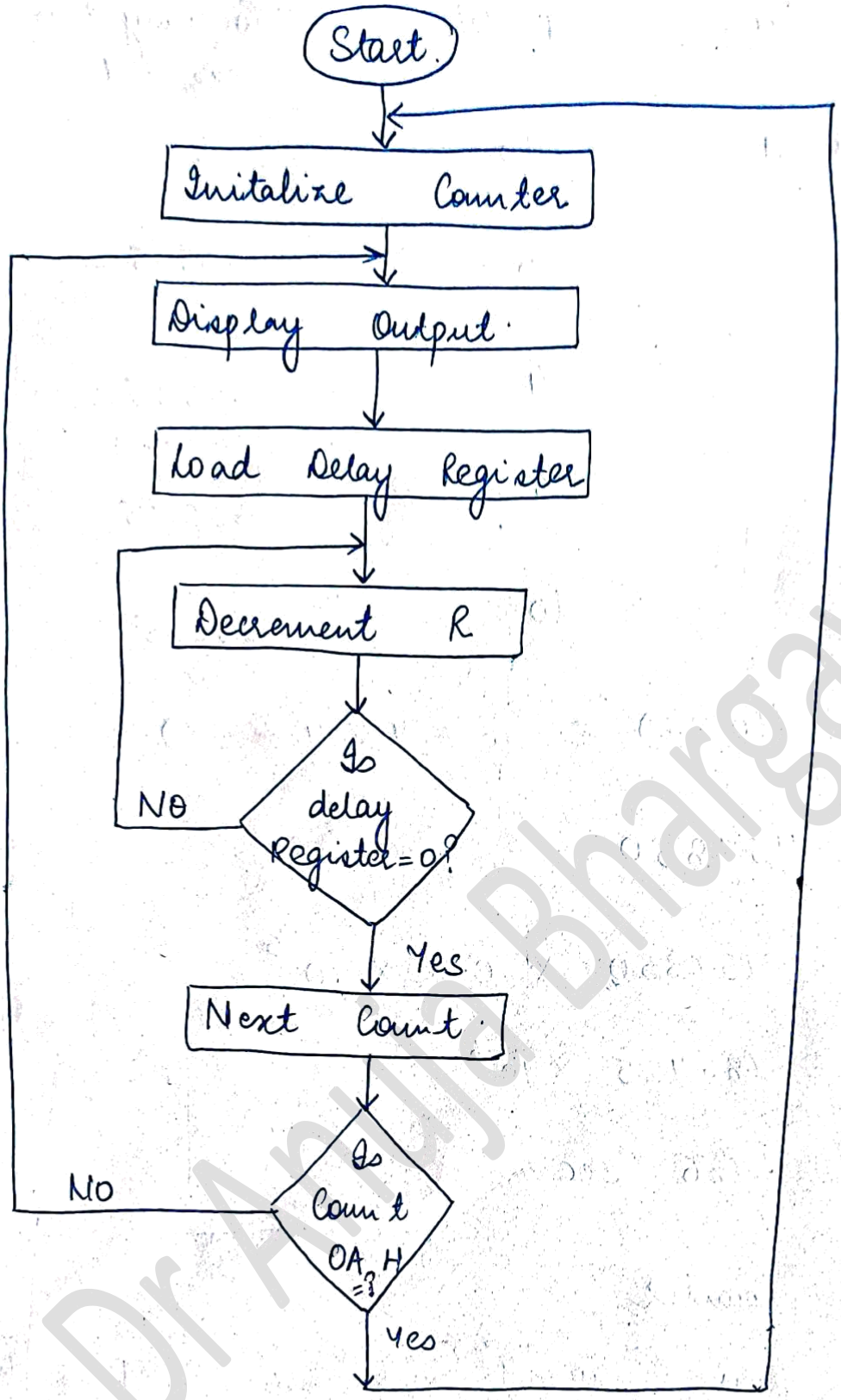
Delay time $= 1572850 \times 0.5 \times 10^{-6}$

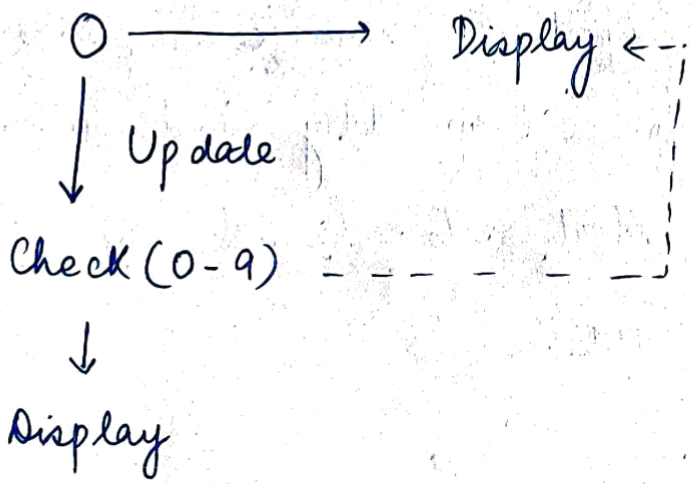
$= 786425 \times 10^{-6}$

$= .786$ sec

Zero to Nine Counter

WAP to count from "0" to "9" with one second of delay between each count. At the count of 9, the counter should reset itself to "0" and repeat the counter sequence continously. Use register-pair H-L to set up delay and display each count at one of the output ports. Assume the clock freq. of the microprocessor is 1 MHz.





CPI data

[A] - data

A > data, CY = 0

A < data, CY = 1

```
Next    MVI   B, 00 H
        MOV   A, B          (T-States)
Display OUT   PORT 1        10
        LXI   H, Count      10
loop    DCX   H             6
        MOV   A, H          4
        ORA   L             4
        JNZ   loop          10
        INR   B             4
        MOV   A, B          4
        CPI   0A H          7
        JNZ   Display       10
        JMP   Next
```

Delay in loop

Delay = Delay in loop + Delay outside the loop

$= [24T \times (Count)_{10}] + 45T$

Given, Delay = 1 sec

& $f = 1\ MHz \Rightarrow T = 10^{-6}$ sec.

$\Rightarrow 1 = [24 \times 10^{-6} \times (Count)_{10}] + 45 \times 10^{-6}$

$\Rightarrow (Count)_{10} = 41664.79 \simeq 41665$

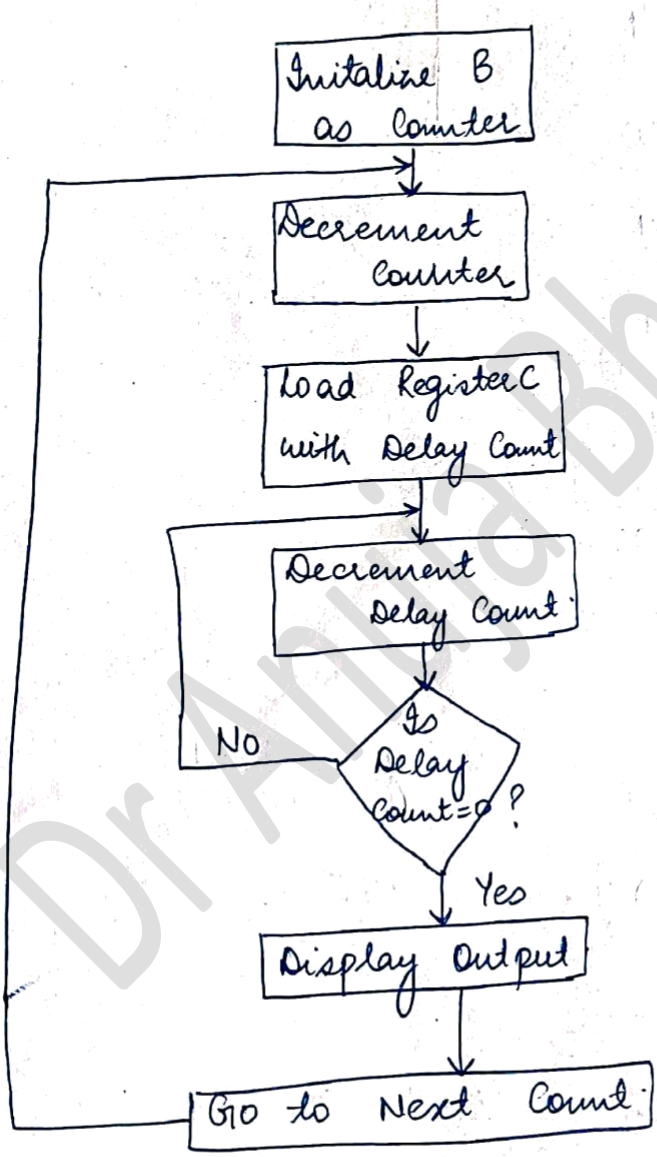
$(Count)_H = A2C1\ H$

16	41665	
16	2604	1
16	162	C
	10	2

Hexadecimal Counter

WAP to count continously in hexadecimal from FF H to 00 H in a system with 0.5 μs. clock period. Use register C to set up 1 ms delay between each count and display the number at one of the output ports.

→ The problem has two parts, the first to set up a continous down counter and the second is to design a



```
        MVI   B, 00H
Next    DCR   B          4T
        MVI   C, Count   7T
Delay   DCR   C          4T
        JNZ   Delay      10T
        MOV   A, B       4T
        OUT   PORT 1     10T
        JMP   Next       10T
```

Delay = Delay Inside Loop + Delay Outside Loop

$= [14T \times (Count)_{10}] + 35T$

Given, Delay = 1 ms & T = 0.5 μs

$\Rightarrow 1\ ms = (Count)_{10} (14 \times 0.5 \times 10^{-6}) + (35 \times 0.5 \times 10^{-6})$

$\Rightarrow (Count)_{10} = (140)_{10} = 8C\ H$

8255 (Programmable Peripheral Interface)

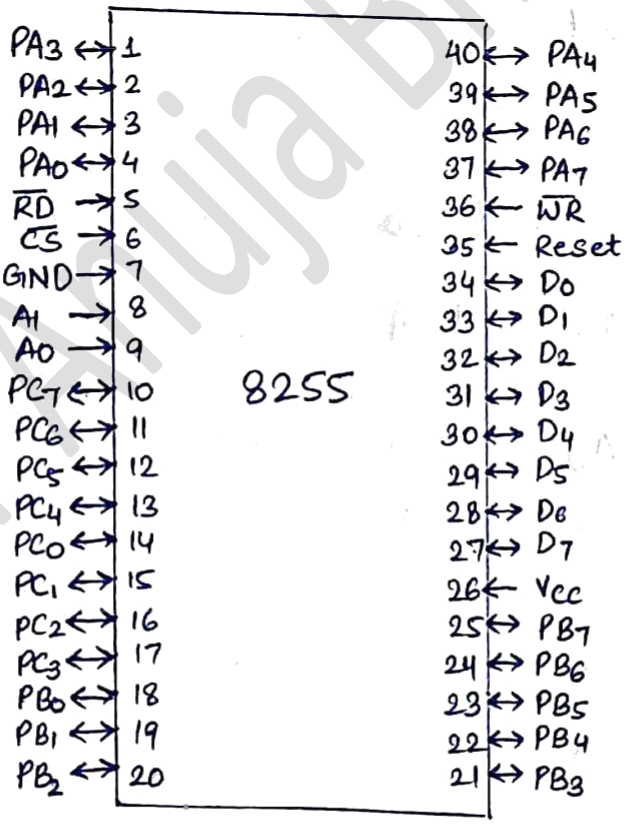
→ The 8255 is a multiport device, useful for interfacing peripheral devices.

→ It has three 8-bit ports, namely Port A, Port B, & Port C.

→ Port C further divided into two 4-bits ports : Port C Upper & Port C Lower.

→ Therefore total 4 ports are available : two 8-bit ports and two 4-bit ports.

→ Each port can be programmed either as an input port or output port.



→ It is a 40 pin I.C, operates on 5V supply.

→ $PA_0 - PA_7$: 8 pins of port A.

$PB_0 - PB_7$: 8 pins of port B

$PC_0 - PC_3$: 4 pins of Port C lower

$PC_4 - PC_7$: 4 pins of Port C Upper

→ $\overline{CS}$ (Chip Select)

The low status of signal enables communication between the CPU and 8255.

→ $\overline{RD}$ (Read)

When $\overline{RD}$ goes low, it allows CPU to read data from the input port of 8255.

→ $\overline{WR}$ (Write)

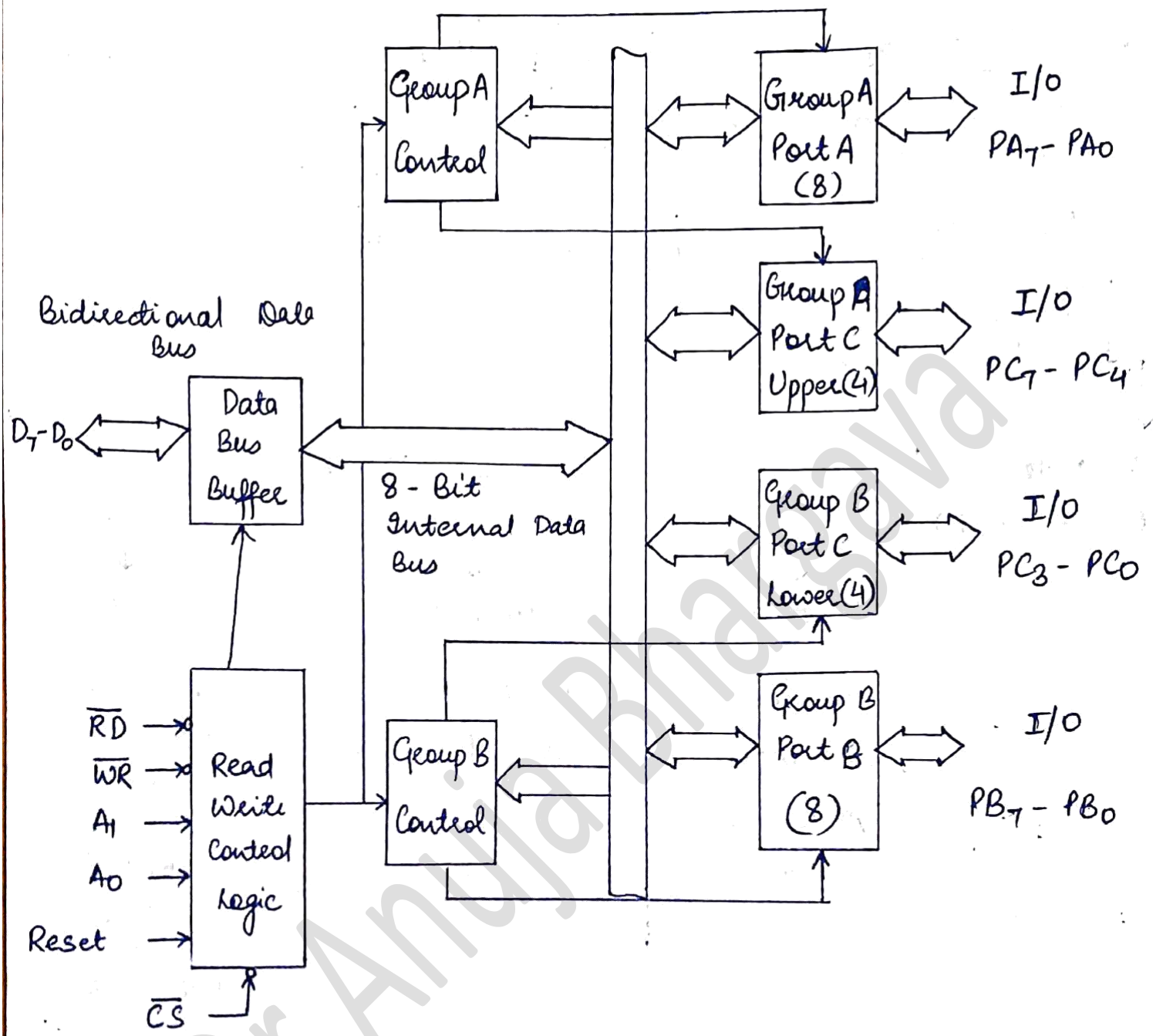
When $\overline{WR}$ goes low, it allows CPU to write data to the output port of 8255.

→ A_0 & A_1

The selection of input port and control word register is done using A_0 and A_1.

	A_1	A_0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$
Port A → Data Bus	0	0	0	1	0
Port B → Data Bus	0	1	0	1	0
Port C → Data Bus	1	0	0	1	0
Data Bus → Port A	0	0	1	0	0
Data Bus → Port B	0	1	1	0	0
Data Bus → Port C	1	0	1	0	0
Data Bus → CWR	1	1	1	0	0

Block Diagram of 8255 :- (Architecture)



→ The block diagram shows two 8-bit ports (A & B), two 4-bit ports (C_U & C_L), data bus buffer and control logic.

→ The control section has six lines

$\overline{RD}$, $\overline{WR}$, $\overline{CS}$, A_0, A_1 (explained in PIN Diagram)

Reset. It is active high signal, it clears the control register and sets all ports in input mode.

Architecture of 8255 consist of three sections.

(1) Data Bus Buffer

- 3-state bi-directional, 8-bit buffer.
- It is used to interface the 8255 to the system bus.
- Based on input, data is transmitted or recieved.
- Here, control word or status information is also transfered.

(2) Read / Write Control Logic

- The function of this block is to manage all of the internal & external transfer of both data & control & status word.

(3) Group A & B Control

- The signals from CPU and send the command to individual control block.
- Group A send the control signal to Port A & Port C Upper
- Group B send the control signal to Port B & Port C Lower.

Port A → It can be programmed by Mode 0, Mode 1, Mode 2

Port B → It can be programmed by Mode 0 & Mode 1.

Port C → It is splitted into two parts : C_U & C_L.

→ It can be programmed by Bit Set / Reset Operation

Control Word Register:

Control Word

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

Bit No. 0 : Used for Port C lower
For Input port the bit is set to 1.
For Output port the bit is set to 0.

Bit No. 1 : Used for Port B.
For Input Port the bit is set to 1.
For Output Port the bit is set to 0.

Bit No. 2 : Used for selection of Mode for Port B.
For Mode 0, the bit is set to 0.
For Mode 1, the bit is set to 1.

Bit No. 3 : Used for Port C Upper.
For Input Port, the bit is set to 1.
For Output Port, the bit is set to 0.

Bit No. 4 : Used for Port A.
For Input Port, the bit is set to 1.
For Output Port, the bit is set to 0.

Bit No. 5 & 6: These bits are to define operating Mode of the Port A.

Mode of Port A	Bit 6	Bit 5
Mode 0	0	0
Mode 1	0	1
Mode 2	1	0 or 1

Bit No. 7: It is set to 1, for simple Input / Output Mode.

It is set to 0, for BSR Mode.

Simple

Modes of 8255 :-

1. Simple Input Output Mode.
2. Bit Set / Reset Mode.

Simple I/O Mode

- In this mode port A & B are used as two simple 8-bit I/O ports and port C as two 4-bit ports.
- The Control Word Format for Simple Input Output Mode is as shown above.

eg 1 Make a control word when ports of 8255 are defined as follows

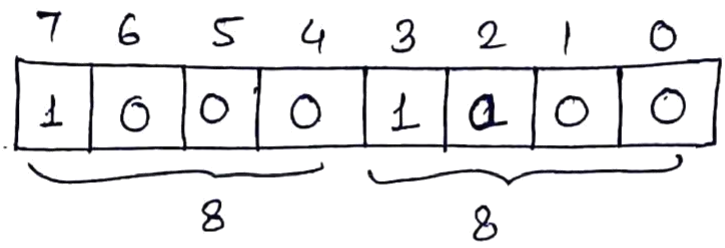
- Port A as Input ~~Output~~ Port
- Mode of the port A - Mode 0
- Port B as Output Port
- Mode of the port B - Mode 0
- Port C upper - Input Port
- Port C Lower - Output Port

Ans :

7	6	5	4	3	2	1	0	← Bit No
1	0	0	1	1	0	0	0	← Control Word Bits
9				8				

Control Word = 98 H

Q2 Form Control Word for 8255 for mode 0 operation Port A - output, Port B → Output, Port C lower → output, Port C Upper - input.



Control Word = 88 H

Bit Set / Reset Mode :-

→ The BSR mode is concerned only with the eight bits of port C, which can be set or reset by writing an appropriate control word in the control register.

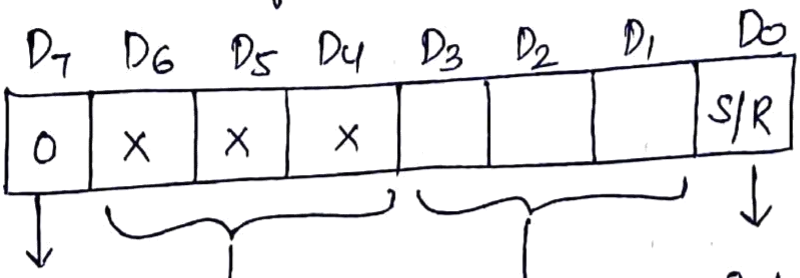
→ A control word with bit $D_7 = 0$ is recognized as BSR control word.

→ It does not alter any previously transmitted control word with bit $D_7 = 1$.

→ Therefore, the input / output operations of port A & B are not affected by BSR control word.

→ In BSR mode, individual bits of port C can be used for applications such as ON / OFF switch.

Control Word for BSR Mode



BSR Mode

Not Used

Generally,
Set = 0

Bit Select

Set = 1

Reset = 0

000	Bit 0
001	Bit 1
010	Bit 2
011	Bit 3
100	Bit 4
101	Bit 5
110	Bit 6
111	Bit 7

Eg

	D7	D6	D5	D4	D3	D2	D1	D0	
To set bit PC_7	0	0	0	0	1	1	1	1	= 0FH
To reset bit PC_7	0	0	0	0	1	1	1	0	= 0EH
To set bit PC_3	0	0	0	0	0	1	1	1	= 07H
To reset bit PC_3	0	0	0	0	0	1	1	0	= 06H

Write a BSR control word subroutine to set bits PC_7 & PC_3 and reset them after 10 ms. Assume a delay subroutine is available.

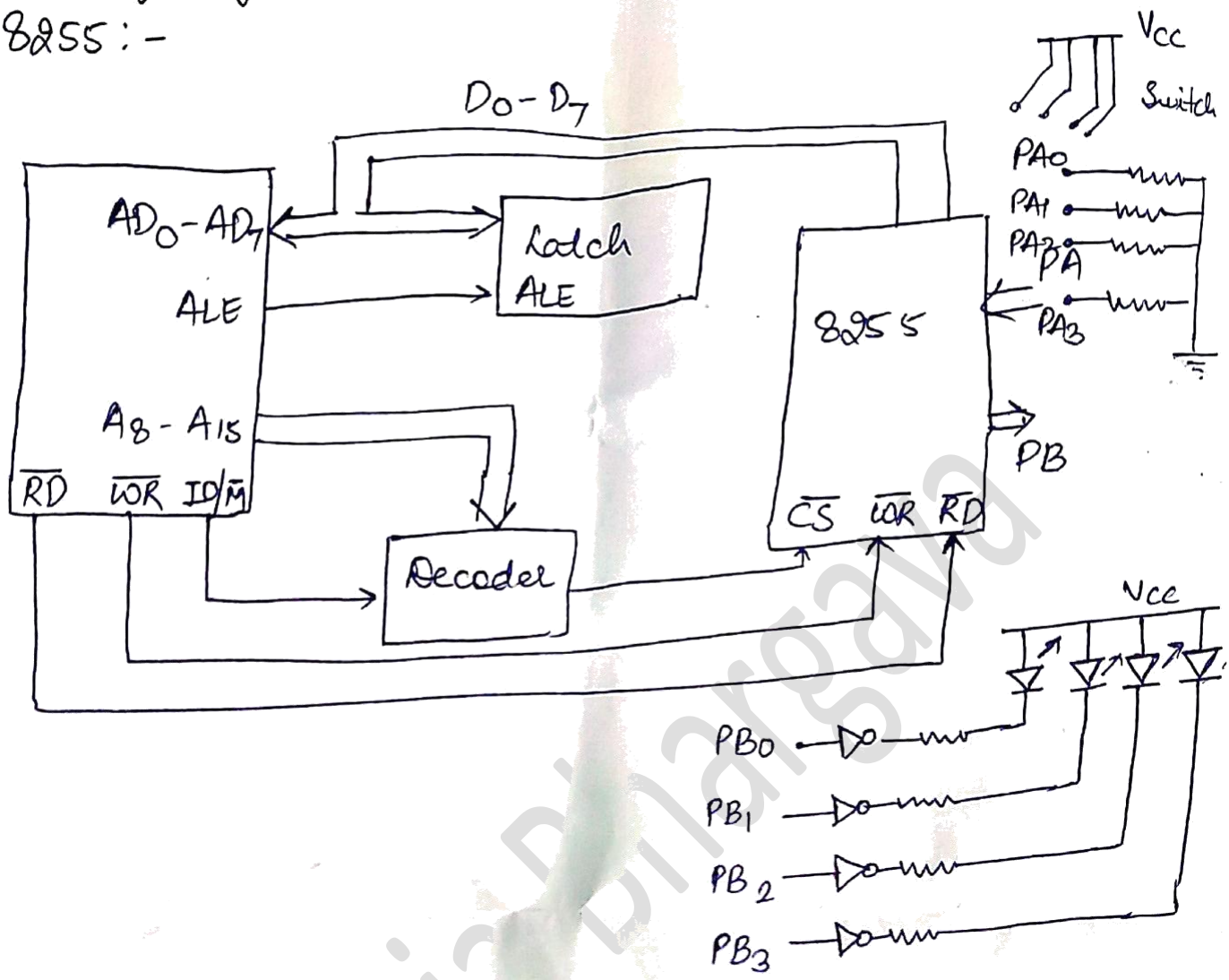
We already know that,

A_1	A_0	Address	Port
0	0	80 H	A
0	1	81 H	B
1	0	82 H	C
1	1	83 H	Control Register

Subroutine,

```
BSR   MVI  A, 0F H  : Load byte to set PC7
      OUT  83 H
      MVI  A, 07 H  : Load byte to set PC3
      OUT  83 H
      CALL Delay    : This is a 10 m-s delay.
      MVI  A, 06 H  : Load byte to reset PC3.
      OUT  83 H
      MVI  A, 0E H  : Load byte to reset PC7.
      OUT  83 H
      RET
```

Interfacing Switches 4 LED's with 8085 through 8255 :-



Control Word : All ports are initalized to mode 0. (90H)

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

Program for initalizing 8255 and performing opr

```
MVI  A, 90 H
OUT  43 H
IN   40 H
OUT  41 H
```

Difference between Memory Mapped IO and IO Mapped

- The microprocessor cannot do anything by itself therefore, it needs to be linked with memory, extra peripherals or IO devices. This linking is called interfacing.

	Memory Mapped IO	IO Mapped IO
1).	IO devices are accessed like any other memory location.	They cannot be accessed like any other memory location.
2)	They are assigned with 16-bit address values.	They are assigned with 8-bit address values.
3)	The instruction used are LDA, STA e.t.c.	The instruction used are IN and OUT.
4)	Cycles involved during operation are Memory Read & Memory Write.	Cycles involved during operation are IO read and IO write.
5)	Any register can communicate with IO device.	Only accumulator can communicate with IO devices.
6)	2^{16} I/O ports are possible to be used for interfacing.	2^8 I/O ports are possible to be used for interfacing.
7)	During read or write cycle IO / $\overline{M}$ = 0	During read or write cycle IO/$\overline{M}$ = 1.

8) No separate control signal required since we have lot of memory space.	Special control signals are used.
9) Arithmetic and logic operation are performed directly on the data.	Arithmetic and logic operation are not performed directly on the data.

Note :- The interfacing of the I/O devices in 8085 can be done in two ways.

→ Memory Mapped I/O Interfacing

In this kind of interfacing, we assign a memory address that can be used in same manner as we used a normal memory location.

→ I/O Mapped I/O Interfacing

A kind of interfacing in which we assign a 8-bit address value to the input / output devices which can be accessed using IN and OUT instructions.

8259 (Programmable Interrupt Controller) :-

- The 8259A is a programmable interrupt controller designed to work with Intel microprocessors 8085, 8086 and 8088.

- The 8259 A interrupt controller can

(1) Manage eight interrupts according to the instructions written into its control registers.

This is equivallent to providing eight interrupts request pins on the processor in place of one INTR (8085) pin.

(2) vector an interrupt request anywhere in the memory map. Since, all eight interrupts are spaced at the interval of either four or eight locations.

This eliminates the major drawback of the 8085 interrupt in which all interrupts are vectored to memory location on page 00 H.

(3) Resolve eight level of interrupts priorities in a variety of modes, such as fully Nested Mode, automatic rotation mode, and specific rotation mode.

(4) Mask each interrupt request individually.

(5) Read the status of pending interrupts, in-service

interrupt and masked interrupts.

(6) Be set up to accept either the level triggered or the edge triggered interrupt request.

(7) Be expanded to 64 priority levels by cascading additional 8259 As

(8) Be set up to work with either the 8085 Microprocessor mode or the 8086/8088 Microprocessor mode.

Block Diagram of 8259A :-

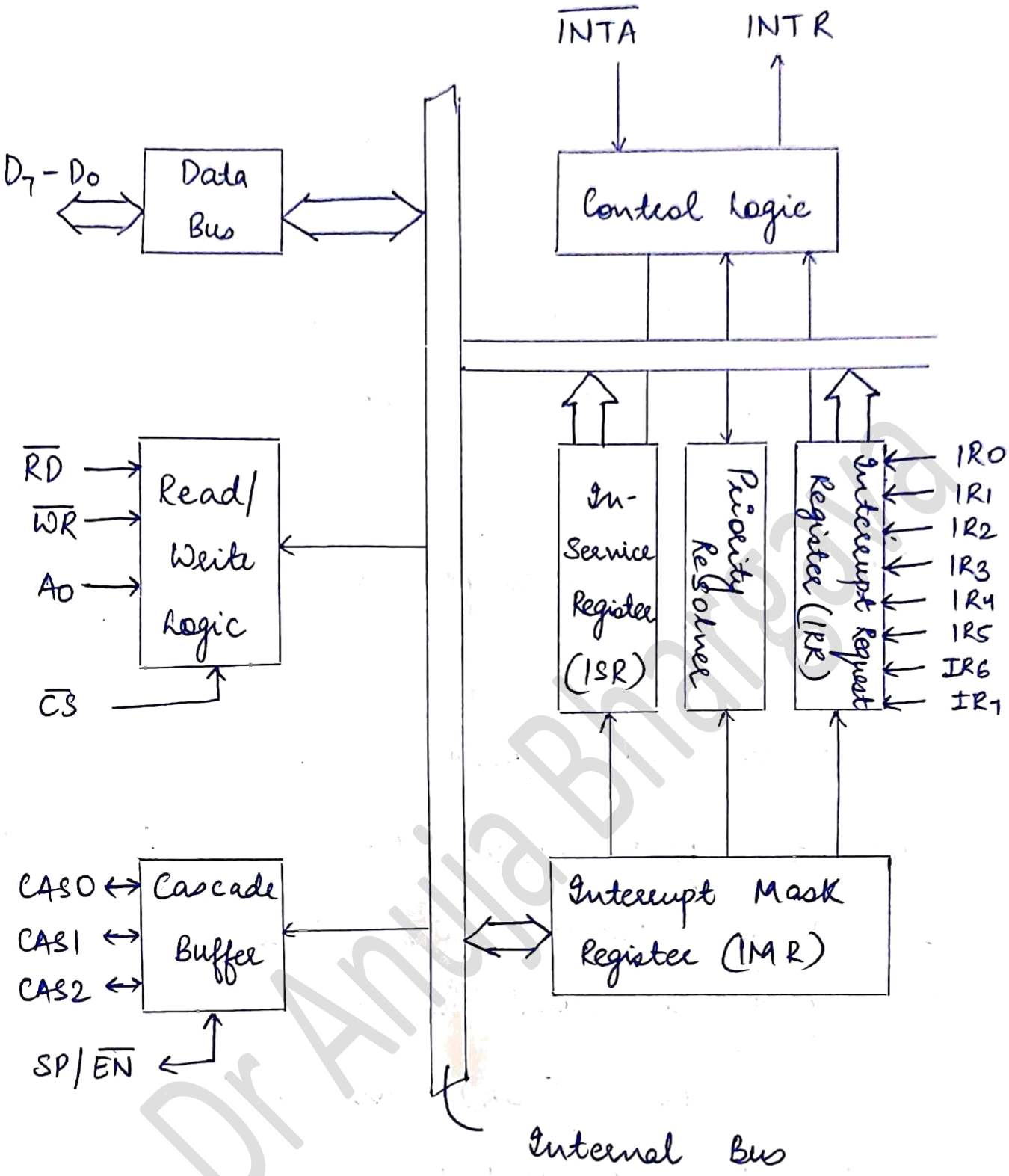
- The internal block diagram of 8259 include eight blocks : control logic, read/write logic, data bus buffer, three registers (IRR, ISR and IMR), priority resolver and cascade buffer.

- Read/Write Logic

→ This is typical read/write control logic.

→ When the address line A_0 is at logic 0, the controller is selected to write a command or read a status.

→ The chip select logic and A_0 determine the port address of the controller.



- Cascade Buffer

→ This block is used to expand the number of interrupt levels by cascading two or more 8259 A.

- **Control Logic**

→ This block has two pins : INT (Interrupt) as an output and $\overline{INTA}$ (Interrupt Acknowledge) as an input.

→ The INT is connected to the interrupt pin of the MPU. Whenever a valid interrupt is asserted, this signal goes high.

→ The $\overline{INTA}$ is the interrupt acknowledge signal from the MPU.

- **Interrupt Register and Priority Resolver**

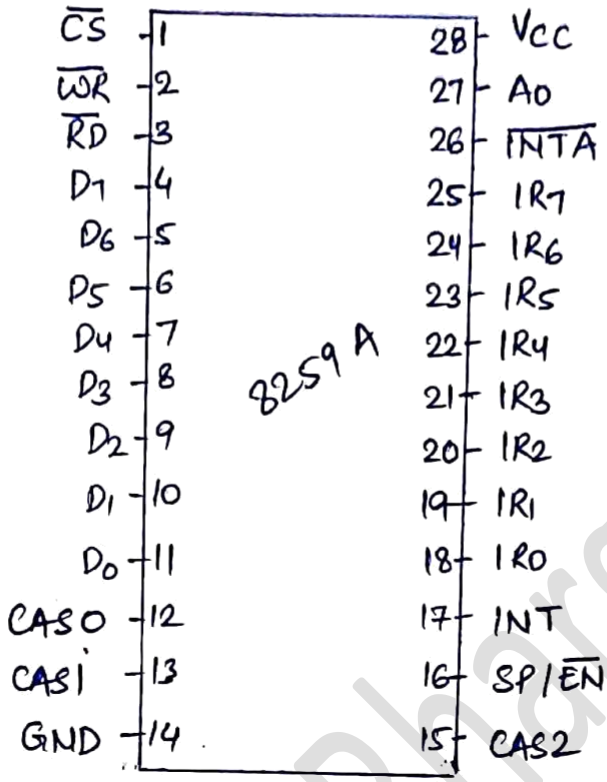
→ The Interrupt request register (IRR) has eight input lines (IR_0 - IR_7) for interrupts. When these lines are high, the request are stored in the register.

→ The in-service register (ISR) stores all the levels that are currently being serviced

→ The Interrupt Mask Register (IMR) stores the masking bits of the interrupt lines to be masked.

→ The Priority Resolver (PR) examines these three registers and determines whether INT should be sent to the MPU.

Pin Diagram of 8259A



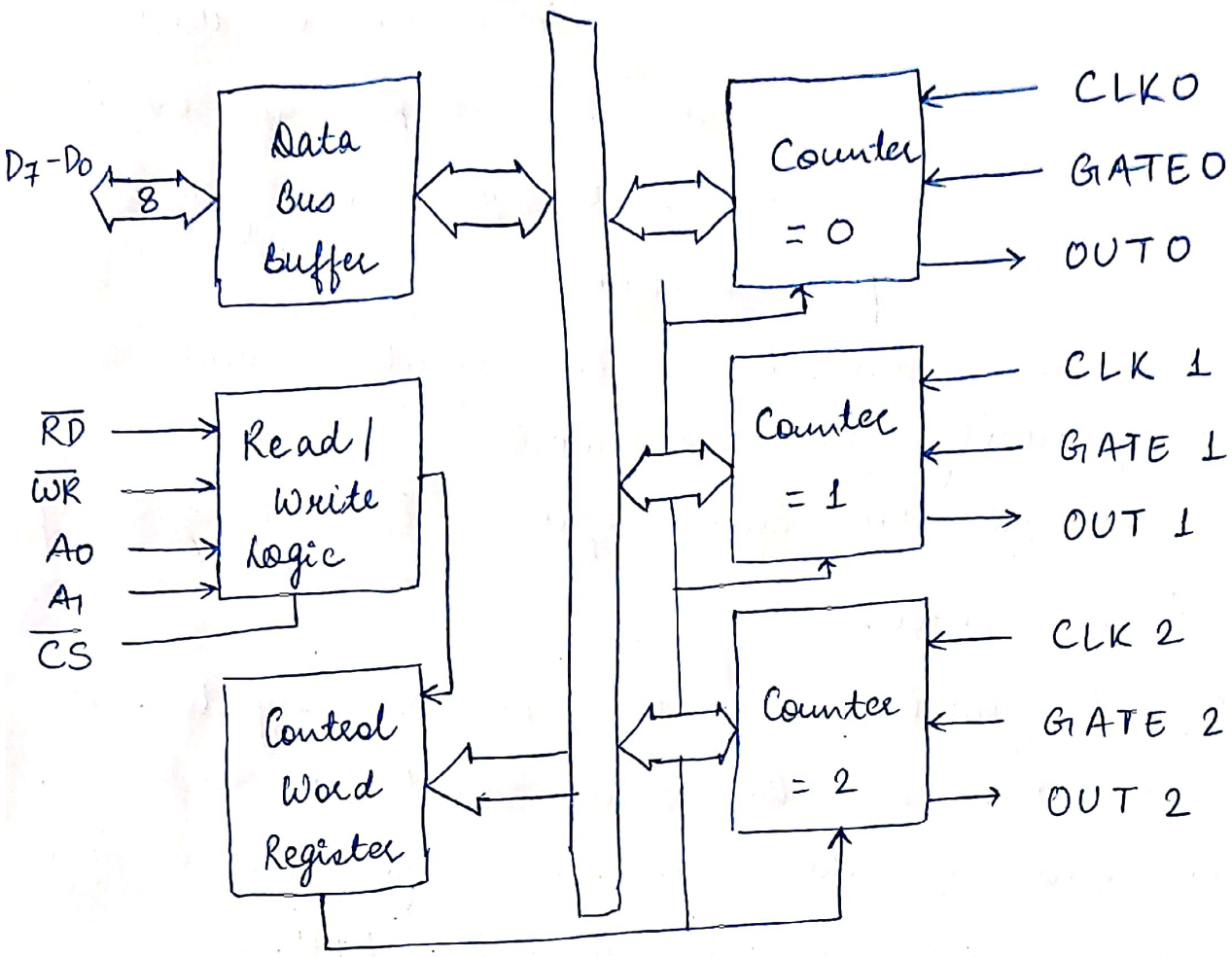
8254 (8253) Programmable Interval Timer: 62

- It generates accurate time delays and can be used for applications such as real-time clock, an event counter, a digital one shot, a square-one generator and a complex waveform generator.
- It includes three identical 16 bit counters that can operate independently in six modes.
- It is a 24 pin DIP and requires +5V power supply.
- To operate a counter,
  - → a 16-bit count is loaded in its register, it begins to decrement the count until it reaches 0.
  - → At the end of the count, it generates a pulse that can be used to interrupt the MPU.
  - → The counter can count either in binary or BCD.
  - → A count can be read by the MPU while the counter is decrementing.

Note:- 8254 is upgraded version of 8253 with difference -

1. 8254 operate at DC-8 MHz (82542 at 10MHz)
   8253 operate at DC to 2 MHz
2. 8254 includes a Status Read-Back command that can latch the count and the status of the counters.

Block- Diagram



- Block- diagram of 8254 include three counters (0,1,2) and data - bus buffer, read / write control logic, and control register.
- Each counter has two input CLK (Clock) and GATE and one output signal (OUT)

- **Data Bus Buffer**

→ This tri - state, 8 - bit, bidirectional buffer is connected to the data bus of the MPU.

- **Control Logic**

→ The control section consist of five signals - $\overline{RD}$ (Read), $\overline{WR}$ (Write), $\overline{CS}$ (Chip Select), Address lines A0 & A1

→ In the peripheral mode, $\overline{RD}$ & $\overline{WR}$ signals are connected to $\overline{IOR}$ and $\overline{IOW}$.

→ In the Memory - mapped I/O, these are connected to $\overline{MEMR}$ (Memory Read) and $\overline{MEMW}$ (Memory Write)

→ Address lines A_0 and A_1 of MPU are connected to lines A_0 and A_1 of the 8254.

→ The control word register and counters are selected according to the signals on lines A_0 and A_1

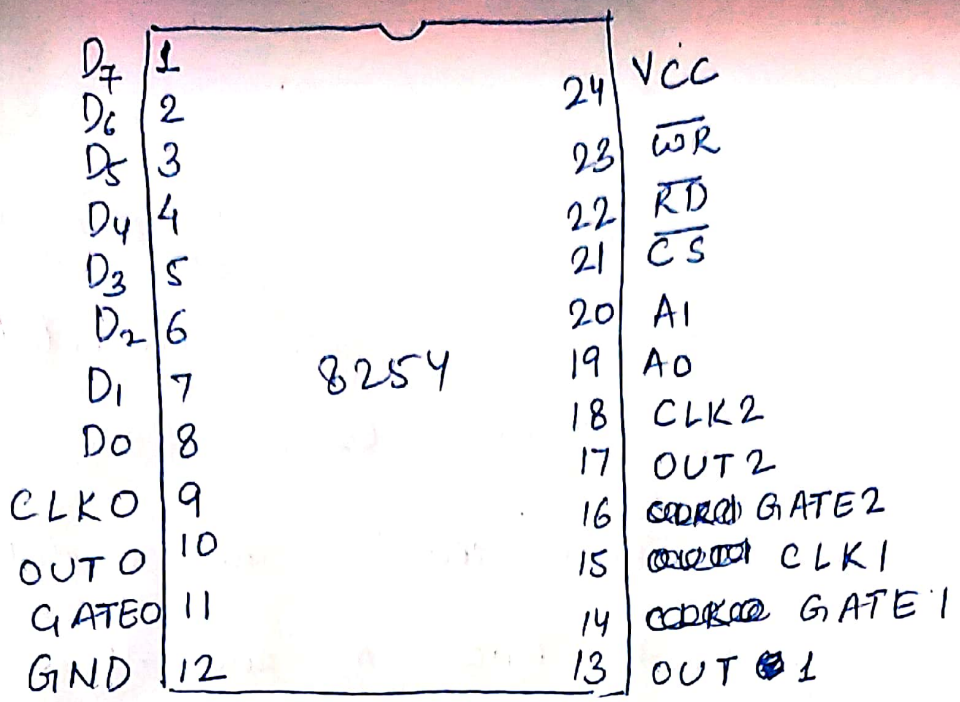
A_1	A_0	Selection
0	0	Counter 0
0	1	Counter 1
1	0	Counter 2
1	1	Control Register

Control Word Register

→ The register is accessed when lines A_0 and A_1 are at logic 1.

→ It is used to write a command word which specifies the counter to be used, its mode and either a Read or Write operation

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SC1	SC0	RW1	RW0	M_2	M_1	M_0	BCD



8254 operates in six modes

1. Mode 0 : Interrupt on Terminal Count
2. Mode 1 : Hardware Retriggerable One-Shot
3. Mode 2 : Rate Generator
4. Mode 3 : Square Wave Generator
5. Mode 4 : Software Triggered Strobe
6. Mode 5 : Hardware Triggered Strobe

GND	Ground
Vcc	+5 V
$D_7 - D_0$	Data Bus 8-bit
CLK N	Counter clock Input
GATE N	Counter Gate Input
OUT N	Counter Output
$\overline{RD}$	Read Counter
$\overline{WR}$	Write
$\overline{CS}$	Chip Select
$A_0 - A_1$	Counter Select