#8086 Microprocessor :

• Introduction of 8086 µp :

Intel develop the first 16 bit microprocessor is 8086. It was developed using H mos tech. (high density short channel mosfet). It is 40 pin DIP IC. 8086 have not any internal ckt so it require external clock source with 33% duty cycle.

• Features of 8086 µp :

(1) It is 16 bit µp so ALU, internal registers and instruction are designed to work by with 16 bit binary data.

(2) It has 16 bit data bus.

(3) It can read or write data to m/m and IO port either 16 bit or 8 bit at a time.

(4) It has 20 bit address bus so it can directly access $2^{20} = 1048576$ (1MB) m/m location (each m/m loc has 8 bit data storage capacity) The 16 bit data stores and two consecutive m/m location.

(5) It can generate 16 bit I/O add, so it can access $2^{16} = 65536$ I/O ports.

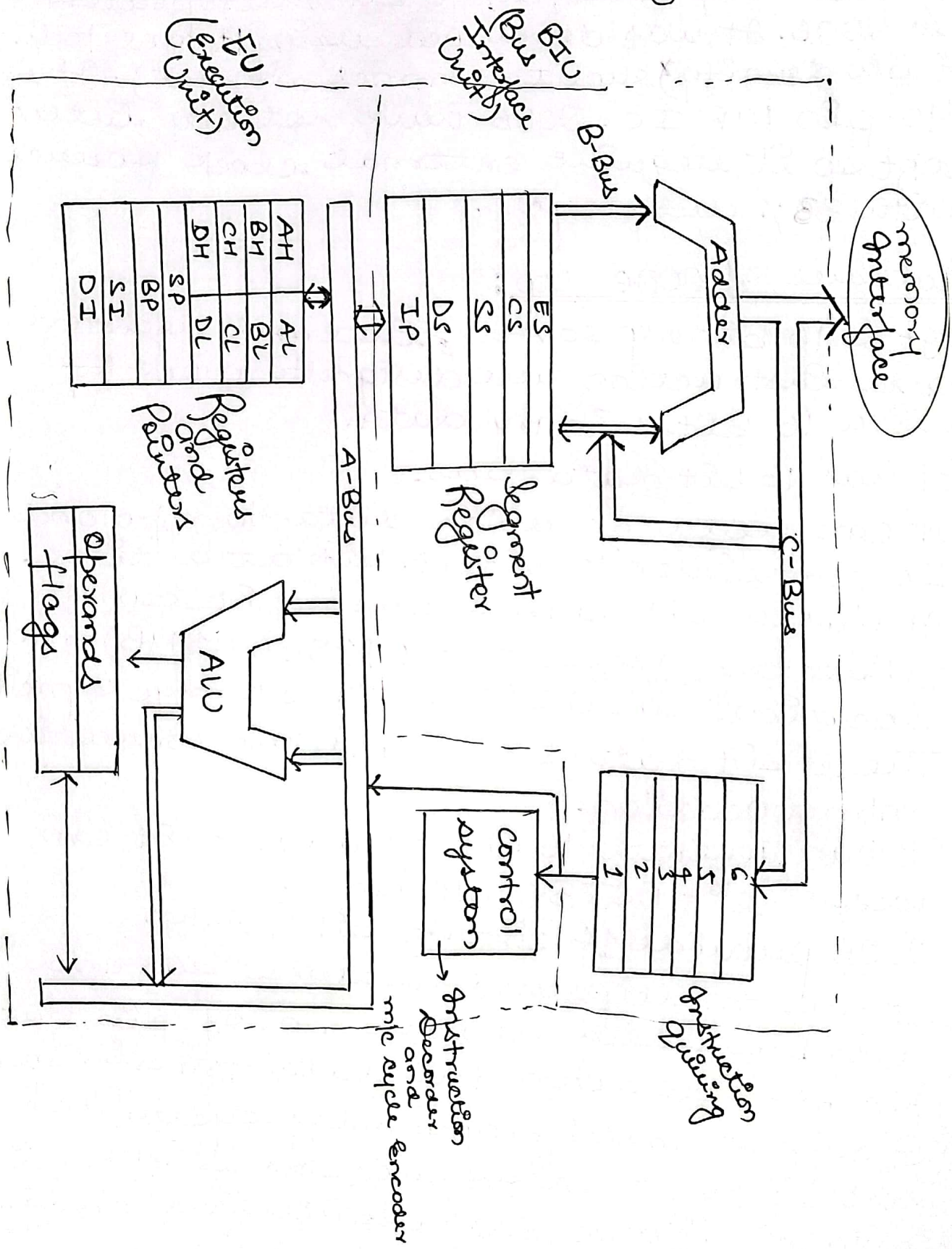
(6) 8086 provide 14 sixteen bit Register.

(7) It has multiplexed address bus and data bus which reduces no. of pin but ~~the no~~ slow down the data transfer speed.

(8) It support multiprogramming system the code of two or more processes is store in m/m at a (some) time and execute time multiplex system.

(9) It can perform A & L op. on bit, byte, word and decimal no. Excluding multiplying and division

Internal Architecture of 8086 Up:



● The architecture of 8086 divided into two units:
(i) BIU
(ii) EU

Both units can work simultaneously to increase system speed and increase the inst. execution per unit time.

● Bus Interface Unit:
BIU provide interfacing b/w up & real world. It provide 16 bit bi-directional data bus and 20 bit address bus. BIU is responsible for performing all external bus op.

● Functions of BIU:
(1) BIU sends address of m/m or I/O.
(2) It fetches inst. from the m/m.
(3) It ~~also~~ read or write data from or into I/O port or m/m.
(4) It supports inst queuing to pre-fetch 6 inst. from m/m.
(5) It provide address re-location facility.

● Instruction Queue:
To speed up program execution, BIU fetches 6 instruction from m/m and store sequentially in a group of register called I.Q. With the help of IQ, it is possible to fetch next instruction when the current inst is in execution. The IQ operate on the principle of FIFO. The size of IQ is 6 Byte and tranfer pre-fetches to execution unit.

● Execution Unit:
EU works parallel with BIU and provide info. about the location of next

instruction or data to be fetch.

- **The EU performs following three main operations:**

(1) Execution of all inst.
(2) It providing address to BIU for fetching data of inst.
(3) Manipulating various registers as well as flag register.

- **Execution Unit consist following functional blocks:**

(1) Contual CKt and Instruction Decoder.
(2) ALU
(3) Flag Register
(4) General Purpose Register
(5) Stack Pointer
(6) Index Register and Pointer.

- **Contual CKt of EU:**

Direct all the internal op. of up. first it decode the inst. and generate contual signal or its required part acc to inst.

- **ALU:**

ALU perform 8 bit or 16 bit mathematical or logical operation,

- **Register Organisation:**

8086 has set of registers known as general purpose register and special purpose registers All registers are 16 bit. The General purpose register can be use either 8 bit or 16 bit. register by programmer they can be use for holding data, Variables, Intermediate data, or temporary data.

These register are also used like counter or storing offset address of some particular operations. The special purpose register are used as segment register, pointers, index register offset storage register. The general data registers are – AX, BX, CX, DX where AX is used as Accumulator, it can be used for 8 or 16 bit operation.

Ax	AH	AL
Bx	BH	BL
Cx	CH	CL
Dx	DH	DL

Fig: General Purpose Register

CS	code segment
SS	stack segment
DS	Data segment
ES	Extra segment

Fig: Segment Register.

The physical address of 8086 is 20 bit to access 1Mb mpm but the register and mpm locations can store 16 bit data. so mpm segmentation is used in 8086. In this ~~this~~ whole one mpm divided in segments

8086 allow 4 active segments 16 bit segment register are provided in BIU which store the starting add. of each segment. The 4 segment are; Code, Data, Extra, stack segment.

#Function of Segment Register:

Code hold the segment upper or mpm segment from where BIU is fetches Inst. of Code byte.

Stack register is used for upper 16 bit starting address of a program (all stack related Inst.) Data segment register or extra segment are used to store upper 16 bit starting add. of m/m data segment where the required data is stored.

#Pointer & Index Register:

- **Instruction Pointer:**
  It is used to store upper 16 bit address of next Inst in code segment which is to be fetch by BIU.
- **Stack Pointer:**
  SP store 16 bit offset add. from start of the segment and provide 20 bit physical address of the stack related Inst.
- **Base Pointer:**
  BPR can be used in case of stack pointer. for generation of 20 bit physical address.
- **Source Index:**
  SI register is used to hold the Off set add. of the data word in data segment. It is also used to calculate the physical address for data segment.
- **Destination Index:**
  DI Register is used to store data for extra segment.

*Default Registers for physical Address:

Types of mem	Default Segment	Alternate Segment	Offset / local addr.
Instruction fetch	CS	None	IP
Stack operation	SS	None	SP, BP
General Data	DS	CS, ES, SS	effective addr.
String Source	ES	CS, SS	SI
String Destination	ES	None	DI
BX Use as Base addr. or Pointer	DS	CS, ES, SS	Effective addr.
BP use as Pointer	SS	CS, ES	Effective addr.

*Flag Register:

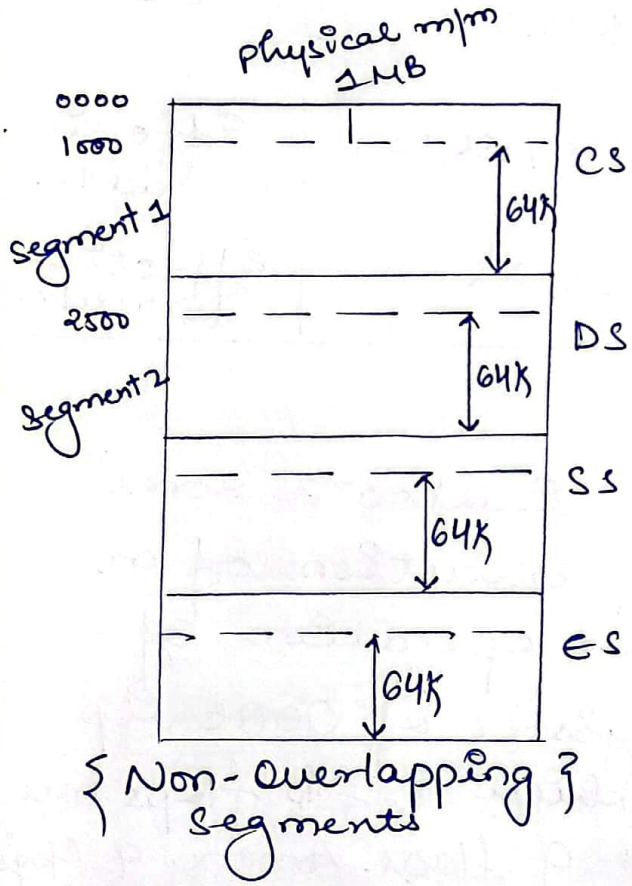
flag is a flip flop which indicate some conditions produced by execution of an inst. or it control certain operation of execution unit. flag Register of 8086 up is a 16 bit register in which 9 flip flops are are active which indicates 9 flags. Among 9 flags the six flags are used to indicate the conditions and 3 flags are used to control certain op. of processor.

* TF → Trap flag.
* IF → Interrupt flag
* DF → Direction flag
* Overflow flag → OF

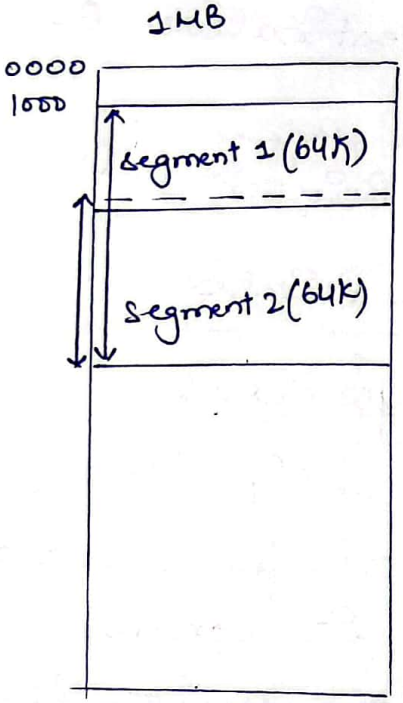
X	X	X	X	OF	DF	IF	TF	SF	ZF	X	AF	X	PF	X	CF
B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0

Memory Segmentation:

8086 µp has 20 bit add. Bus. So, it can access 2^{10} MB m/m location. to organise large m/m space memory segmentation is require and in this complete physical m/m divided into no. of logic segment and each segment is 64KB in size and address by 1 of the segment register. The segmentation are perform in two manner non overlapping segments or overlapping segment.



{Non-overlapping segments}

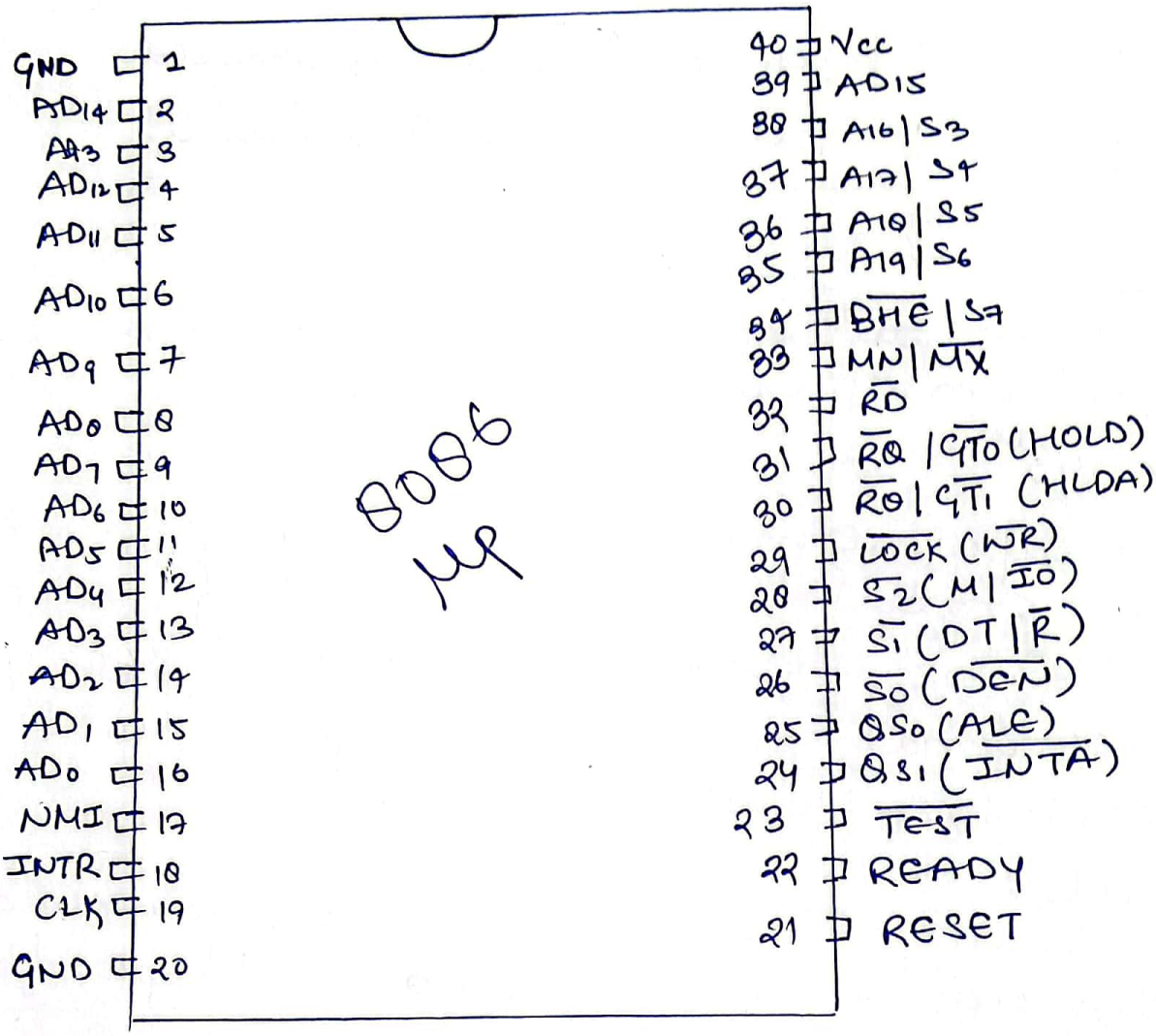


{overlapping segment}

Advantages of m/m segmentation:

(1) It improves m/m addressibility.
(2) It allow inst. in code/data/stack and portion of program more than 64KB by using other segments along with 1.
(3) It facilitate use of separate m/m for data & stack.

Pin Diagram of 8086:



- **AD0 to AD15:** In a multiplexed address data Bus. These signal works as address Bus when ALE signal is logic 1 and ALE logic 0 works as Data Bus. ~~Data Bus~~ **D0 – D15 Data Bus:** When HOLD and acknowledgement occur these pins goes in high impedance logic.
- **A19/S6 – A16/S3:** These are multiplexed address and status signal bus. In this status Bit, S6 always remain at logic 0. S5 bit indicate the condition of Interrupt flag. S4 and S3 indicate which segment is accessed during current bus cycle.

S_4	S_3	Function
0	0	Extra segment is accessed
0	1	Stack segment is accessed
1	0	Code / No segment is accessed.
1	1	Data segment is accessed.

- **$\overline{RD}$:**
Whenever this signal is logic 0 data Bus receive data from m/m and I/O device.

- **READY:**
This input signal control the Mp op. and insert wait state in the timing of Mp. when signal on this pin is logic 1 Mp perform idle op. or normal op. when the logic 0 Mp is in wait state / partial interrupt.

- **INTR:**
The interrupt request is input signal and it is a hardware interrupt when interrupt flag is 1 then 8086 processor generate acknowledge signal & & Mp address interrupt mode.

- **$\overline{TEST}$:**
It is a input signal. It is tested by wait inst. If signal on this pin is 0, Mp performs no op. and when signal on this pin is 1, Mp returns to perform op.

- **NMI (Non-Maskable Interrupt):**
It is similar to INTR but it does not check interrupt flag. when Mp receives logic 1 at this pin Mp transfer the program control to a Vector address.

- **RESET:**
The signal on this pin reset the Mp, terminate all the interrupts & transfer the Program control

to m/m FFFF0H.

- **CLK Signal:**

At this pin the external CLK signal provided by a signal generator CKT of 33% duty cycle to synchronise the operation of all internal parts of Mp.

- **Vcc & GND:**

- **MN/$\overline{MX}$:**

The minimum & maximum mode select pin to select the operation of Mp when signal of this pin 1 the single Mp operate and signal of this pin is 0 the more than Mp can operate. Simultaneous. in a system. $\overline{MX}$/MN → GND (maximum mode) → Vcc (minimum mode)

- **$\overline{BHE}$/S7:**

It is Bus high enable. It help Mp to enable MSB most significant byte of databus (D15-D0) during read or write op. and status signal S7 is always logic 1.

- **Minimum Mode pins:**

M/$\overline{IO}$. MN/$\overline{MX}$ pin connected to +5V DC supply the following pins functions in minimum mode op.

- **M/$\overline{IO}$:**

This is memory/I/O pin which indicate the address carries either m/m address or I/O port number.

- **$\overline{WR}$:**

When the signal of this pin is 0, Mp send out data to m/m or I/O device.

- **$\overline{INTA}$:**

In response of interrupt request Mp send acknowledge signal at this pin by connect it to ground.

- **ALE:**
ALE indicate the multiplexed address Data Bus carries address ~~bus~~.

- **DT/$\overline{R}$:** (Data Transmit/$\overline{Receive}$)
When signal on this pin is 1 that control the data is send from microprocessor to peripheral device. When signal on this pin is 0 when up receives data from peripheral device by enable external data bus buffer.

- **DEN:**
In normal op. DEN signal (Data enable) is used to activate data bus buffer for transferring the data b/w up and peripheral device.

- **HOLD:**
At this pin DMA send a request it is same as the HOLD signal of 8085 up.

- **Maximum Mode Pins:**

- **$\overline{S_2}$, $\overline{S_1}$ and $\overline{S_0}$:**
These are status signals which indicate the func of current bus cycle. 8086 Bus controller decode such signals that identify bus relate op. as shown following table-

$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	function
0	0	0	Interrupt Acknowledge
0	0	1	Input Device Read
0	1	0	Output Device Write
0	1	1	Halt
1	0	0	Opcode fetch
1	0	1	Memory Read
1	1	0	Memory Write
1	1	1	High Impedence/Tri-state

- **$\overline{RQ}/GT_1$ & $\overline{RQ}/\overline{GT_0}$:**

At this pin, DMA send request and at the other pin Mp grant permission.

- **$\overline{LOCK}$:**

This pin is used to lock peripheral device of system This pin is activated by using clk signal.

- **QS_1 & QS_0 :**

These are Queue status pin which decode by Mp to identify the status of inst queue register

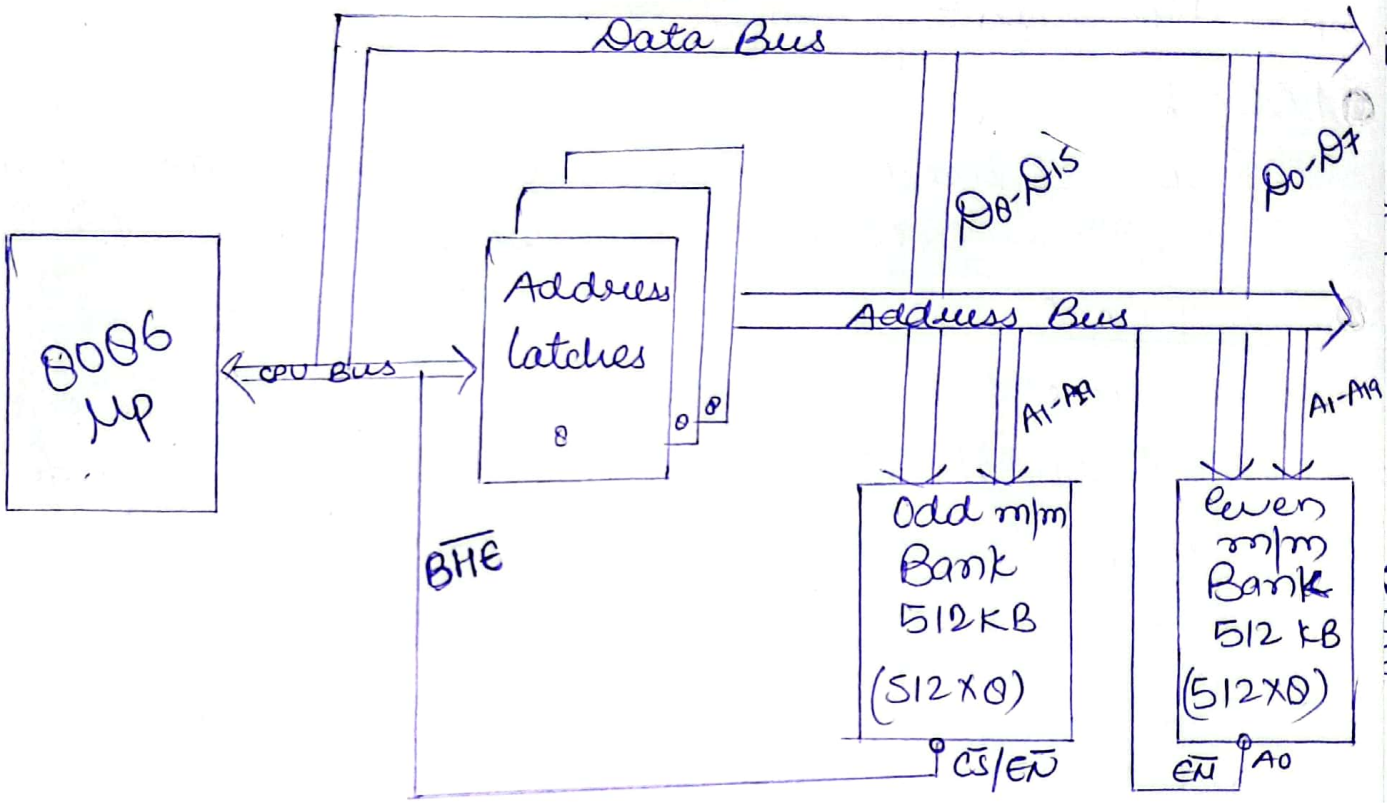
QS_1	QS_0	function.
0	0	No operation
0	1	First Byte of opcode
1	0	Queue is empty
1	1	Subsequent bytes of opcode

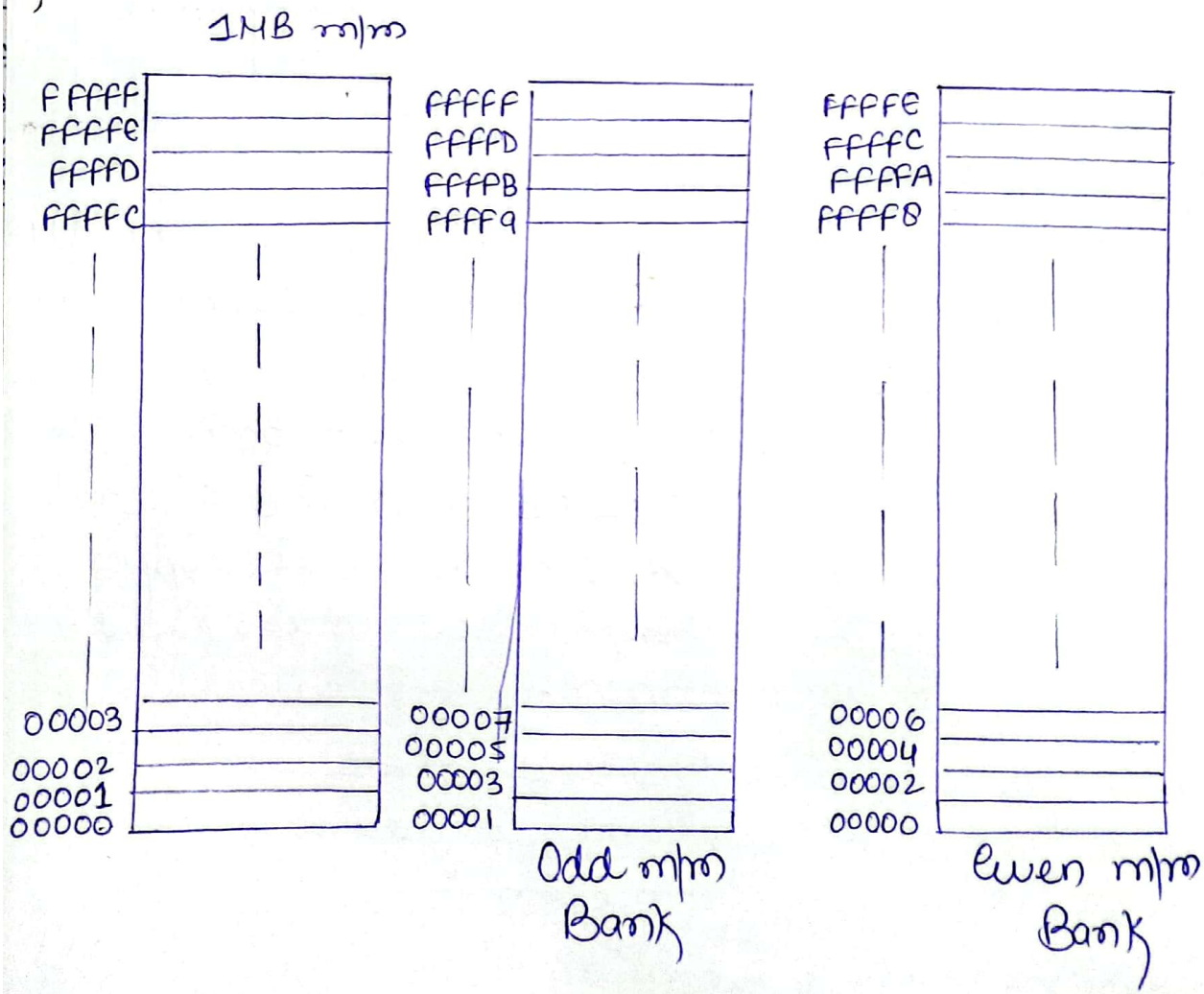
Memory addressing of 8086 Mp :

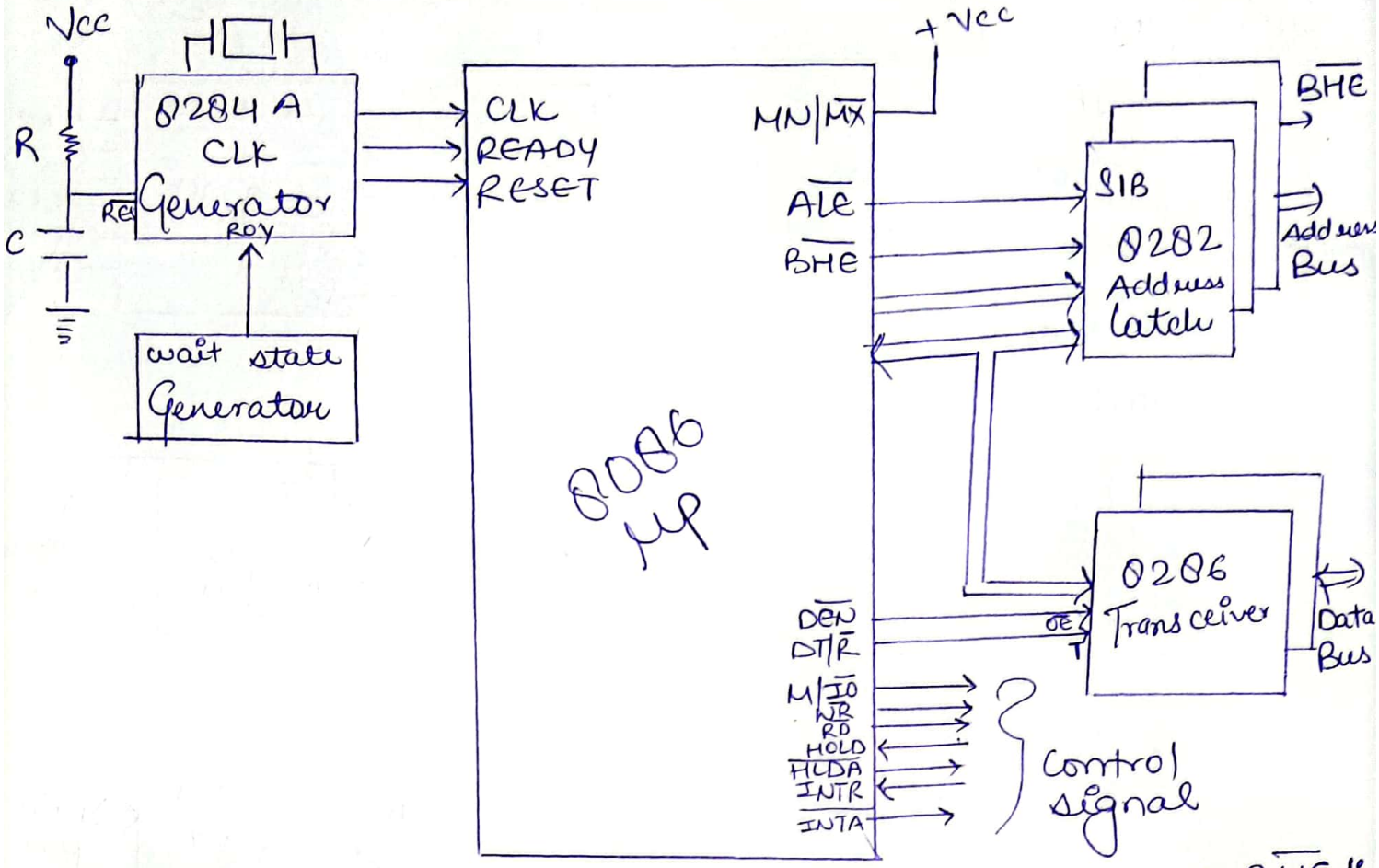
8086 has 20 bit add. Bus by this it can operate 1MB of 8 bit data storage. 16 bit data carried out must be stored in two consecutive m/m location so 1MB physical m/m must be divided into two m/m bank of 512 KB m/m size of each is c/d even bank (lower bank) and second one is upper bank. Two possible m/m banks can access (add) by following four manners.

2. 8 bit data from even bank / address Bank
2. 8 bit data from odd (upper Bank) or address
3. 16 bit data starting from even Bank
4. 16 bit data starting from Odd address Bank.

Organization of even and odd m/m in 8086 Based System:

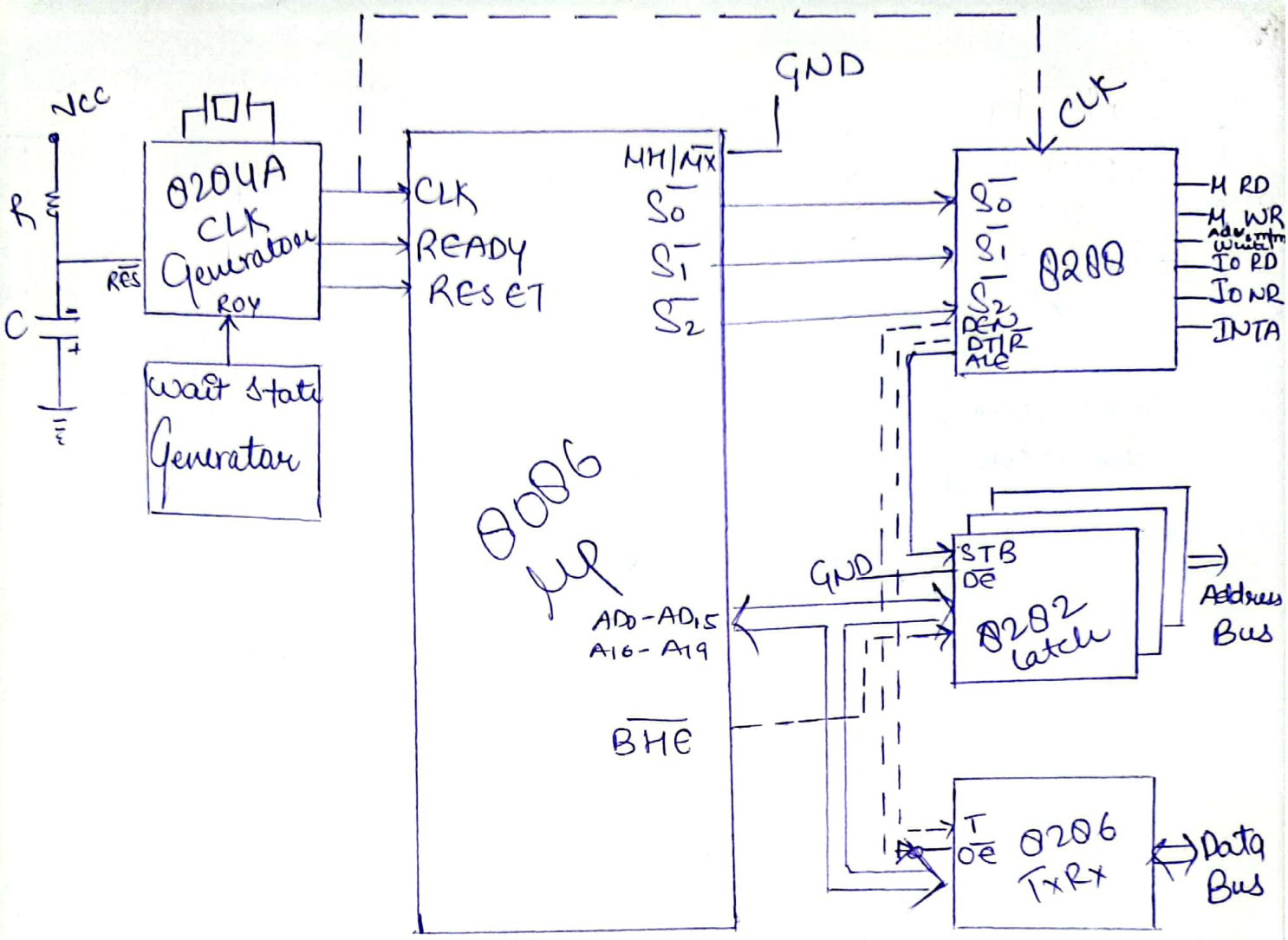






As shown in fig. The AD0-AD15, A16/S3 - A19/S6 and BHE/S7 are multiplex signal and these signal demultiplex by external latch and a ALE signal generated by μp to enable the latch. The 8286 is a data transmitter and receiver it is used to connect input/output devices with μp. DT/R signal connected to T input of Transceiver which enable it and control the direction of data flow from μp and to the μp. The CLK Generator is an imp. Component of minimum mode op. It performs following function—

(1) Control signal generation.
(2) RESET or synchronize signal.
(3) Provide ready signal to check μp or peripheral devices are ready for op.
(4) Provide CLK signal to peripheral device.



In maximum mode configuration, the additional CKt are required to generate control signals which convert status signal $\overline{S_0}-\overline{S_2}$ in IO/M/m transfer signal and also generate Control signals for data flow by 8282 latch and 8286 Transceiver.

#Addressing Mode of 8086 μp:

8086 μp fetches Inst (Code) from m/m by generating 20 bit physical address with the help of Inst pointer and code segment. There are no. of m/m location and registers where the operand may be specified according these registers & pointers. 8086 μp support 12 AM which are arrange in 3 group that are: Data AM, Program m/m AM and stack m/m AM.

(1) **Register A.M:**
In this mode the source and destination operand are both to be contain general purpose register
Eg → MOV Bx, CX (16 bit)
MOV BL, BL (8 bit)

(2) **Immediate A.M:**
In this A.M, 8 bit or 16 bit data is specified as the part of the inst.
Eg → MOV BL, 26H
MOV Bx, 4057H

(3) **Direct A.M:**
In this direct addressing mode, 16 bit effective address is mention in instruction.
Eg → MOV AL, [3000H]

(4) **Indirect A.M:**
In this mode, the effective address is either in a pointer reg. or index reg.
Eg → MOV [DI], BX
MOV, DL, [BP]

(5) **Base + Index AM:**
The AM is similar to Indirect A M because the m/m add. is indirectly indicate by base address or 1 Index register. The Base reg. hold the starting location of any data array while Index reg. hold last location of element of data array.
Eg → MOV CX, [BX+DI]

(6) **Register Relative AM:**
In this AM, the m/m address are stored in Base or Index register with a direct address. This is similar to Base + Index A M.

Eg → MOV CX, [BX + 0003H]

(7) Base relative Index A M:

In this the Base address, index register and a direct displacement (direct address) are specified in the Inst. to generate the physical address all the 3 contents are added upto Eg: MOV AL, [BX+SI+10H]

(8) String A M:

This AM used index register for indicate the of source or destination. The source index point out the first byte of source operand and destination index is used to point out first byte of word in destination operand. When direction flag is '0' the SI and DI automatically incremented.

DF=0, SI & DI Incremented
DF=1, SI & DI decremented

The data segment is used for source and ES is used for destination.

Eg: MOVS BYTE

(9) Direct Port A M:

Eg: OUT 05H, AL
IN AX, 00H

(10) Indirect Port A.M:

The port no. or address is stored in a register

Eg: IN AL, DX.

(11) Relative AM:

In this AM, the m/m address where the program control is transfer specified within Inst. along with opcode.

Ex: JMP 2000H

(12) Implied AM:

In this AM, inst itself specify the data which is operated by the inst.

Ex → CLC , NOP , STC

- CLC → clear carry flag
- NOP → No Operation
- STC → set carry flag

Instruction format:

In 8086 Mp the inst varies from 1 byte to 6 byte. The 8 bit or 16 bit displacement (direct data or address), opcode and addressing mode is specify by First two bytes of the inst. IF has two byte displacement. It has 2 byte Effective Address. It may have 2 byte displacement and 2 byte segment addres

#Program format:

Assembly language program is written in standard format. It has five fields which arrange in following manner:

Lable	MNEMONICs	operand	operand	Comment
3000H	LXI H	25H	00H	load HL register pair by data 2500H

#looping, Counting & Indexing:

These are programming technique. In looping, the program is instructed to execute certain set of inst. repeatedly to execute a particular task number of times. In counting, program allow to count how many times the inst or set of inst. are to be executed. In indexing, program allow programmer to point the data stored in sequential m/m location one by one.

#The looping technique has four sections:

* Initialisation section
* Processing section
* loop control section
* Result section

- **Initialisation Section:**
The initialisation section establish the starting value of loop for counting how many times loop have to execute.
- **Processing section:**
In processing section, the actual data manipulation access.
- **loop control section:**
The loop control section updates counters, pointers for next inst.
- **Result section:**
The result section analyzes and store the result.

#Counters and time Delays:

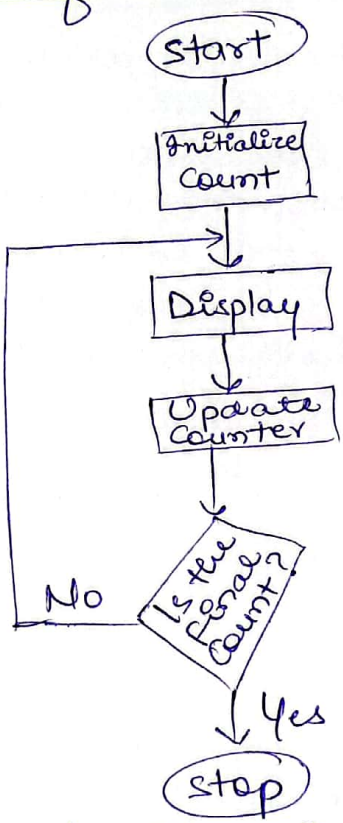
Counters are used to keep track of events and time delay are Imp. In ~~sto~~ second up, a reasonable accurate timing b/w two events. The process of designing counters and timing delay using inst is more flexible and less

time consuming then the design process using hardware.

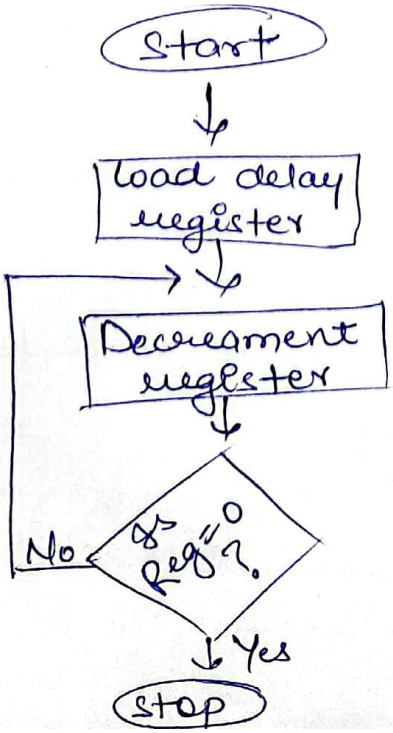
#Counter:

Counter is design by storing a desired number into one register and using increament or decreament inst. A loop is establish to update the count and each count is check to determine upto final count and loop is repeated.

#flow chart of Counter:



#flow chart of Time Delay:



#Time delay using NOP Instruction:

NOP Inst. do nothing but it takes 4 T-state to execute it provide a time delay of 4 time-state

$$1T\ state = \frac{1}{\text{operating freq. of } \mu p}$$

So, no of times to execute NOP Instruction we can get Obtain required time.

#Time delay using Counters:

Counting process can create time delay the exact time of Inst. used in counter subroutine are known as Initial Value of counter. A specific time delay can be determine by as follows:

		No. of T-states
MVI C, count	Load the count	7 T
Back: DCR C	Decreament Register-C	4 T
JNZ Back	If count ≠ 0, Back	10/7 T

In this delay program the loop execute (count-1) times and 1 time. It execute at false condition so total time states in counter program is

$$\text{Total Time state} = \underset{\text{MVI,C}}{7} + \underset{\text{loop}}{(\text{count}-1)\times(4+10)} + \underset{\text{last state (false cond.)}}{(4*7)}$$

*for count = 5

$$\text{No. of T-states} = 7 + 4\times 14 + 11$$
$$= 74 \text{ T-states.}$$

*for freq = 2 MHz

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

*Total time delay for execute the program

$$74 \times 0.5 \mu sec = 37 \mu sec.$$

#Time delay using Nested loop:

If a loop contain another loop in a program then the time delay calculated for individual loops (inner and outer loop) and after Σ of time delay we get total time delay of nested loop program.

```
    MVI B, multiple count
Start: MVI C, Delay count
Back: DCR C
      JNZ Back
      DCR B
      JNZ Start
```

loop2

loop1

The T-state require for execution of inner loop