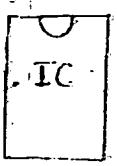
up → Mech. + Electrical + Electronic

↓

Vaccum tubes

↓

Transistor (1947)
William shockly
John Bardeen
Walter Bratlain



SSI – < 10 Components (small scale integration)

MSI – (10-100)

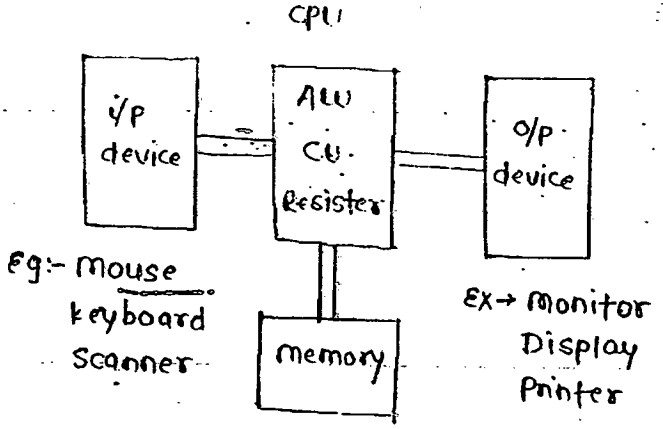
LSI – (100-1k) (Large Scale integration)

VLSI – >1k (very large scale integration)

SLSI –

ULSI –

Computer →



Eg:- Mouse
keyboard
Scanner

Ex → Monitor
Display
Printer

Processer → It is a s/c component designed by using VLSI tech. & it includes ALU, CU, Resistor of a CPU in a single package.

Intel – DRAM

Busicom (Japan) – Calculator.

* Intel is a company which 1st designed processer.
* For a processer mem. is connected externaly.
* Resister is not a mem. because it contains temporary data in it.

1971 → Intel 4004 → 4bit up

Bit → Binary digit (0/1)

Nibble → Group of 4 bit

Byte → 8 bits

Word length → Depands on type of up

1972 → Intel 8008 → 8bit

1974 → Intel 8080 → 8bit

1977 → Intel 8085 → 8bit

1978 → Intel 8086 → 16bit

Intel 8088, ... 80286, 80386 (32bit), ...,

Pentium, dual core, ..., i3, i5, i7 (64bit)

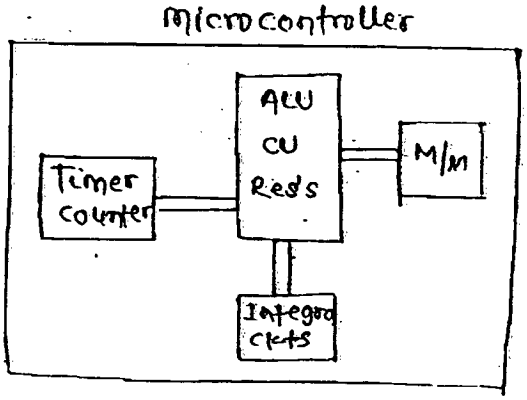
* Wordlength → No. of bits a processer can process at a time.

Eg:- 8bit up — 1B

16bit — 2B

32bit — 4B

64bit — 8B



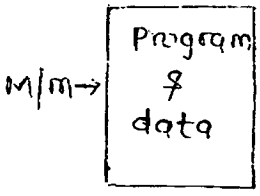
Microcontroller	Microprocesser
(1.) It contains ALU, CU, Res's	(1.) It contains ALU, CU & resistor
(2.) Contains internal m/m.	(2.) No internal m/m.
(3.) It has integrating ckt's, timer/counter.	(3.) No interfacing ckt's, timer/counter.
(4.) Used for specific purpose application	(4.) Used for general purpose appli?
(5.) Eg.→ Intel 8051, 8031 (8 bit) PIC - 8 bit/16 bit, TMS 1000, Intel 80196.	(5.) Eg.→ Intel 8085, 8086, Z80, M6800 i5, i7.

(4) Depanding on how prog & data are stored in the mem. there are 2 types of architecture:-

(1.) Von-Neumann archi (or) Princeton archi.

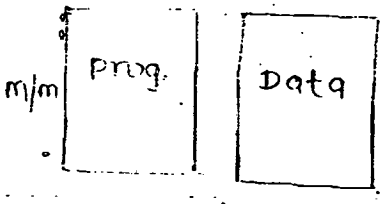
(2.) Harvard architecture.

Von-Neumann

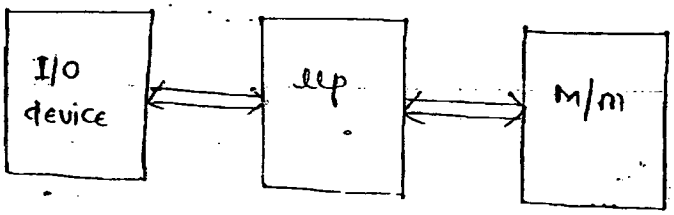


Eg.→ Intel 8085, 8086

Harvard archi.



Eg.→ Intel 8051 (uc)



Opcodes (Operati code

↑

Instruction

↑

Data & prog.

Basic Operation of up:-

(1.) Opcode fetch
(2.) M/m Read
(3.) m/m write
(4.) I/o Read
(5.) I/o write

} m/c cycle

(1.) Opcode fetch → Reading or excessing instruction i.e. opcode from the memory into processer.

(2.) Memory Read → Reading (or) excessing data from mem.

(3.) Memory write → Sending data (or) xferring data to mem.

(4.) I/O Read → Reading (or) excessing data from i/p port (or) device

(5.) I/O write → Transferring data to o/p port (or) device.

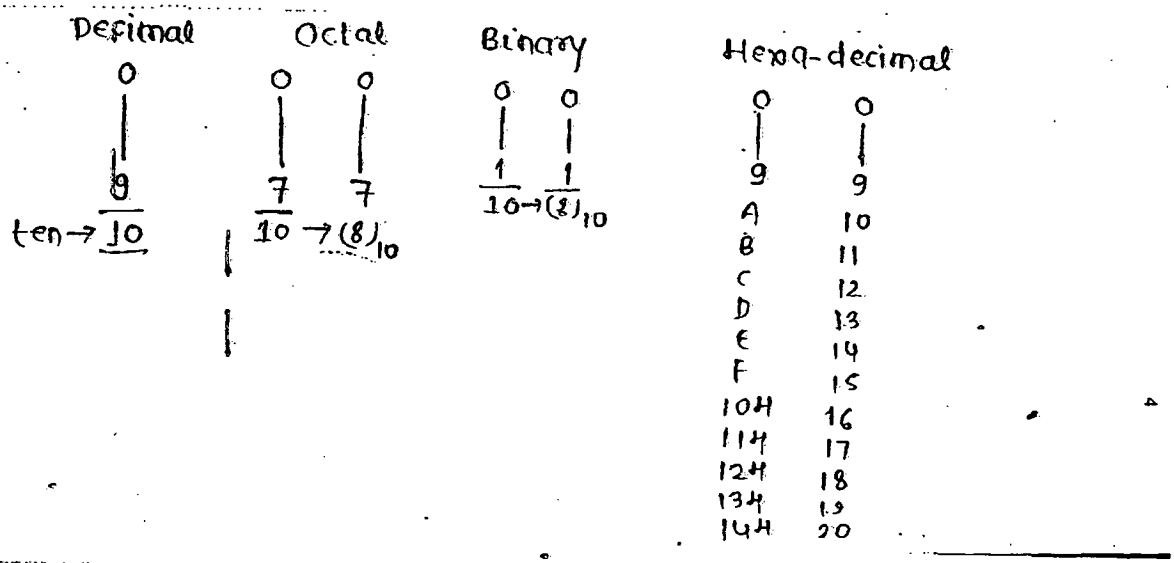
8085 syllabus (8bit)

(1.) System bus
(2.) Internal architecture
(3.) Pin layout / Description
(4.) Functional description
(5.) Programming model
(6.) Instruction Format
(7.) Addressing mode
(8.) Instruction set classification
(9.) Simple programs
(10.) Interfacing

(1.) Architecture – 1 to 4 (50% – 60%)
(2.) programming – 5 to 9 } (40% – 50%)
(3.) Interfacing – 10 }

* All the data is represented only in HEXA-DECIMAL.

Number system →



Addition →

Decimal

$$\begin{array}{r} 9 \\ +7 \\ \hline 16 \end{array} \qquad \begin{array}{r} 9 \\ +5 \\ \hline 14 \end{array}$$

Hexa-decimal

$$\begin{array}{r} F \\ +7 \\ \hline 16H \end{array} \qquad \begin{array}{r} F \rightarrow (9) \\ +5 \\ \hline 14H \end{array}$$

$$\begin{array}{r} {\scriptstyle 1}\,FF \\ +84 \\ \hline 183H \end{array} \qquad \begin{array}{r} {\scriptstyle 1\ 1\ 1\ 1}\,FFFF \\ +9765 \\ \hline 19764H \end{array} \qquad \begin{array}{r} {\scriptstyle 1\ 1\ 1\ 1}\,DED\,EH \\ +7359H \\ \hline 15237H \end{array}$$

Subtraction →

Manual Processer

A-B = A+(-B)

$$\begin{array}{r} 96H \\ -35H \\ \hline 61H \end{array} \qquad \begin{array}{r} 96H \\ +CBH \\ \hline (1\,X)\,61H \end{array}$$

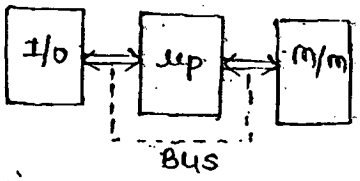
$$\begin{array}{r} 35H \\ 0011\ 0101 \\ 1100\ 1010 \longrightarrow \text{1's comp.} \\ +\quad\ 1 \\ \hline 1100\ 1011 \longrightarrow \text{2's comp.} \\ \hline C\ B\ H \end{array}$$

(15)	(16)
F	10
-3	5
C	B

(1.) $\begin{array}{r} 86H \\ -22H \\ \hline 64H \end{array}$ $\qquad \begin{array}{r} F\ 10H \\ -2\ 2 \\ \hline D\ EH \end{array}$ $\qquad \begin{array}{r} 86 \\ DE \\ \hline (1\,X)\,64H \end{array}$

(2.) $\begin{array}{r} 73H \\ -18H \end{array}$ $\qquad \begin{array}{r} 10H = 16 \\ 673H \\ -18H \\ \hline 5BH \end{array}$ $\qquad$ (OR) $\begin{array}{r} F\ 10H \\ -1\ 8H \\ \hline E\ 8H \end{array}$ $\qquad \begin{array}{r} 78H \\ +E8H \\ \hline 5BH \end{array}$

* System Bus →



* Group of cond^{r} (or) wires used for communication b/n processer, mem. & I/o devices.
* There are 3 types of bus:-
  (i) Address Bus
  (ii) Data bus
  (iii) Control bus.

(i) Address bus → * It is used to Xfer the address of either mem. or I/o device from the processer.

* It defines the max^{m} mem. that can be connected to a processer. given by the relation $\boxed{2^n = N}$

  where n = no. of add. lines

  N = no of add./mem. loc^{n}

* It is of 16 bits in length for 8085.
* Address bus is unidirectional.

Memory → * It is a collection of resistors.

* Resistor is a collection of F/F.s.

* A F/F is a mem cell which can store a bit i.e. 0 (or) 1.
* Most of the mem. are designed to store 8 bits per each mem. loc^{n}. Therefore mem. is given or represented in terms of byte.

Relation b/n add. line & mem. →

2^{10} ⟶ 1 Kilo Byte ⟶ 1KB

2^{20} ⟶ 1 Mega byte ⟶ 1MB

2^{30} ⟶ 1 Gega byte ⟶ 1GB

2^{40} ⟶ 1 Tera byte ⟶ 1TB

Que→ A processer has 24 add. lines. Calculate the max^m mem that can be connected.

Soln→ $2^n = N$

$2^{24} = 2^4 \times 2^{20}$

$= 16\,(1MB)$

$= 16MB$

Que→ A mem of 512 TB can be connected to a processer. Find the add. lines required.

Soln→ $2^n = N$

$= 512TB$

$= 2^9 (2^{40})$

$2^n = 2^{49}$

$n = 49$

Que→ A mem of 120GB is to be connected to a processer. Find the min^m add. line req.

Soln→ $2^n = N$

$= 120GB$

$2^{36} = 64GB$

$= 120GB$

$2^{37} = 128GB$

$n = 37$

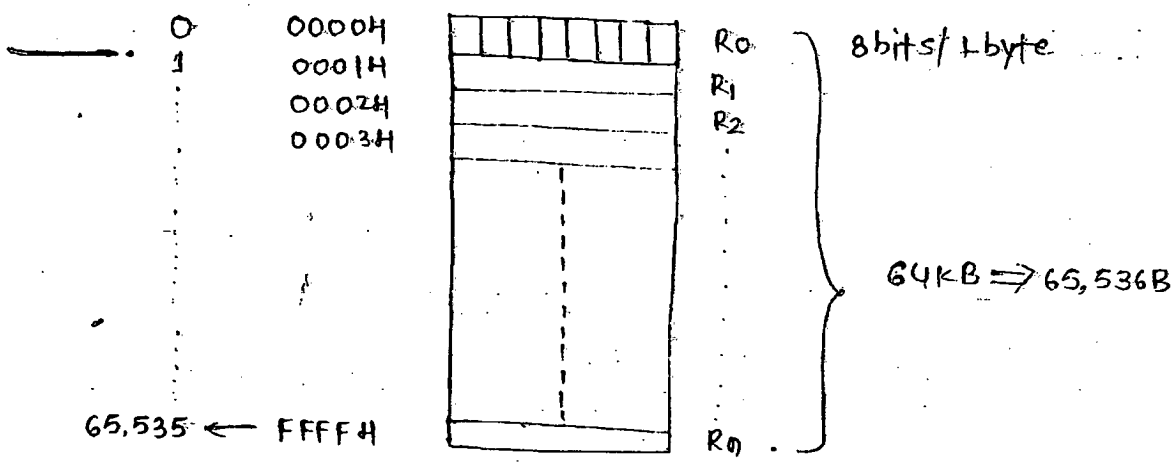
Que→ The max^m mem that can be connected to 8085 is __ ?

Ans→ $2^n = N$

$2^{16} = 64KB$

$= 64 \times 1024B$

$= 65,536B$



(2) Data bus → * It is 8 bit bus in length. For 8085.

* It is used to Xfer data b/n processer, m/m & I/O device.

* Data bus is bidirectional.

* There is no seprate data bus in 8085. The lower 8 add. line can be used either as add. or data bus. with the help of a control signal ALE (add. latch enable)

Add. latch enable — ALE

* 1; all 16 lines → add. bus

* 0; A_{15}-A_8 → add. bus

AD_7-AD_0 → data bus

	A_{15}-A_8	AD_7-AD_0 → multiplexed add/ data bus
ALE=1;	Higher order add. bus	lower order add. bus

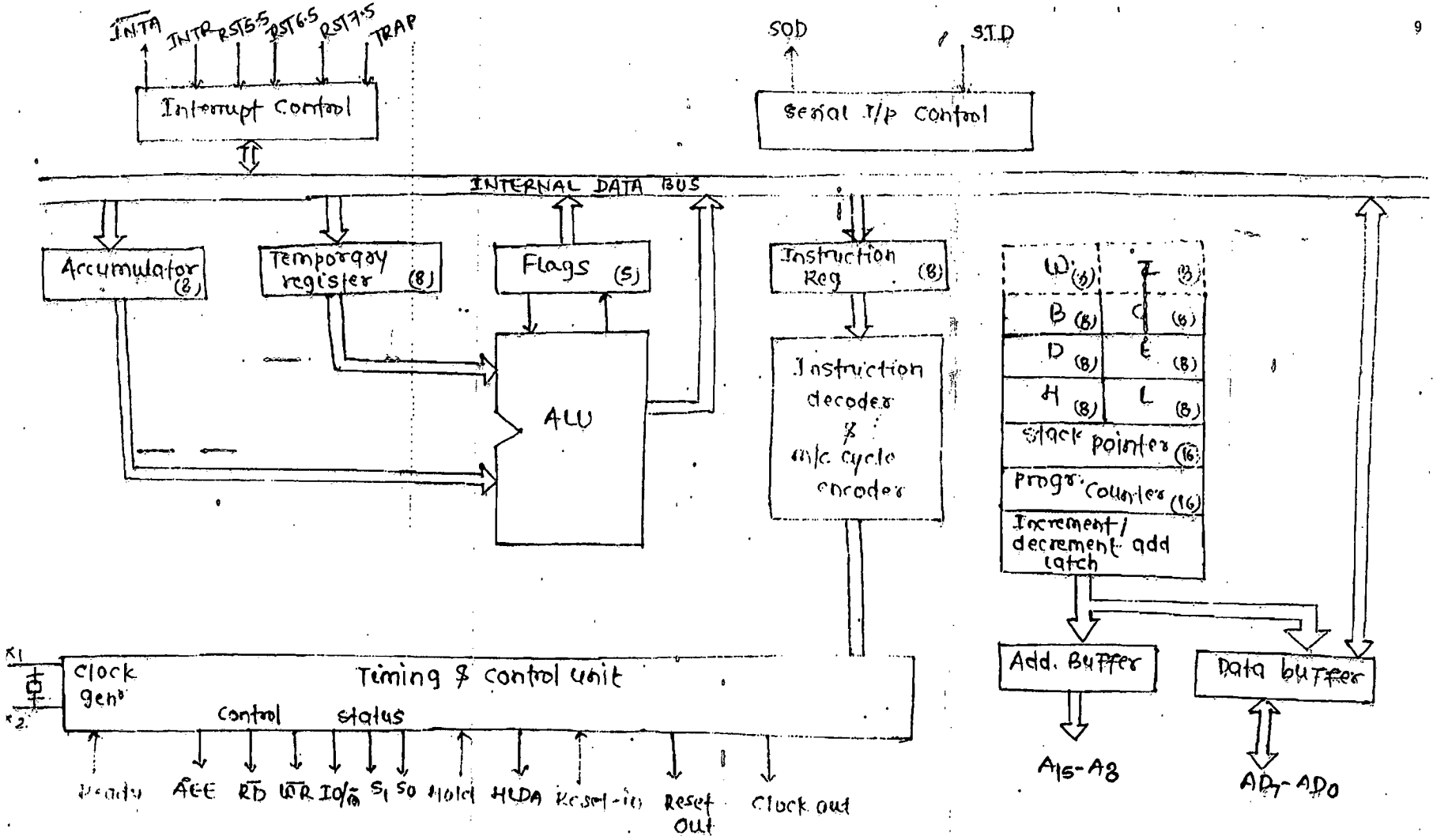
(3) Control bus → * It is a group of diff control signals req; For various operations of the processer.

* It is partialy unidirectional & partialy bidirectional.

Control bus is :-

(a) Uni-di (b) bidi (c) both (d) None

(b) Uni-di (b) bi-di (c) partialy bi-di & uni-di (d) None.


INTA
INTR
RST5.5
RST6.5
RST7.5
TRAP
Interrupt Control
SOD
SID
Serial I/P Control
INTERNAL DATA BUS
Accumulator (8)
Temporary register (8)
Flags (5)
ALU
Instruction Reg (8)
Instruction decoder & m/c cycle encoder
W (8)
Z (8)
B (8)
C (8)
D (8)
E (8)
H (8)
L (8)
Stack pointer (16)
Progr. Counter (16)
Increment/ decrement- add latch
Add. Buffer
Data buffer
A15-A8
AD7-AD0
X1
X2
Clock gen
Timing & control unit
Control
status
Ready
ALE
RD
WR
IO/M
S1
S0
Hold
HLDA
Reset-in
Reset out
Clock out

* Internal archi. is devide into 5 functional unit :-

(1.) Register unit

(2.) ALU

(3.) Timining & control unit

(4.) Interrupt unit

(5.) Serial I/o control unit

(1.) Register Unit → * There are 2 types of register:-

(a.) General purpose reg.

(b.) Special purpose reg.

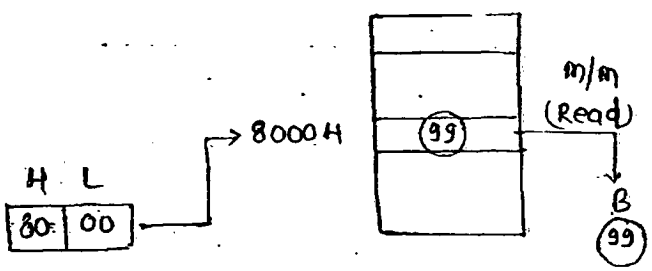
(a.) General purpose reg. → * There are 6, 8 bit general purpose reg. namely B, C, D, E, H, L.

* There are 3 reg pairs of 16 bits in length i.e. BC, DE & HL.

* Any of the reg. pairs can be used to point to the mem. but HL pair is known as default mem or data pointer.

Eg.→ MOV B, M.

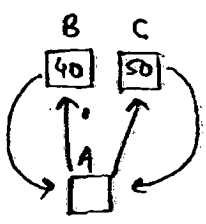
M indicates mem. content (or) mem. reg whose add. is present in HL pair only



Accumulator → * It is an 8bit multipurpose reg. by which almost all ALU Operation are performed.

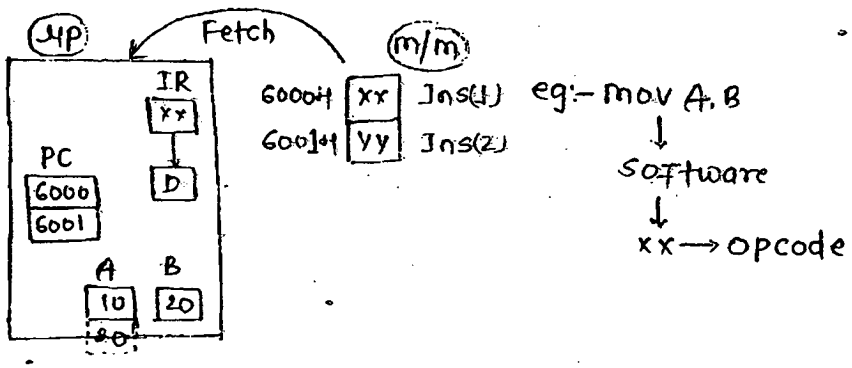
* One of the bytes must be present in acc.

* Result is stored in accumulator.



(b) special purpose reg →

(i) Program Counter → * It is a 16 bit reg which contains add. of next ins. to be executed. (or) it takes care of prog. flow (or) control.



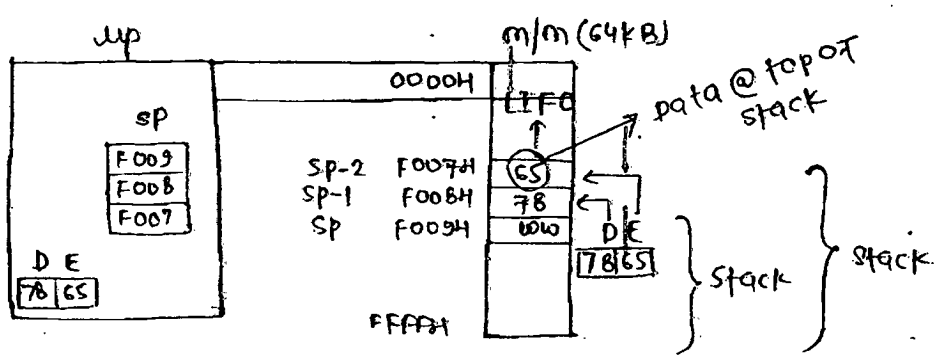
(ii) Instruction reg → * It is an 8 bit reg. which contains opcode of present instruction.

(iii) Instruction decoder & m/c cycle encoder → After the opcode is fetched into ins. reg. it is decoded in this block with the help of μprog.

* A no. of m/c cycles (or) operations are assigned acc. to type of ins.

μprog. → It is a prog. written by ~~skilful~~ cheap designer to make the processor understand what an ins. is (or) it indicates the type of opern to be performed for an ins.

(iv) Stack pointer → * It is a 16 bit reg. which contains add. of the data present at top of the stack (or) it points to top of the stack mem.



Stack
(a) LIFO (b) FLLO (c) FIFO (d) None

Que → What is meant by stack?
Ans → * It is a part of read write mem. which is used to store temp. data & also the content of PC when subroutines are in used.
* A technique involved in stack is Last in 1st out.

Note: * When data is stored in stack mem. Stack pointer is decremented similarly when data is accesed from stack mem. stack pointer is incremented.
* There is no provision to store a single reg. content in the stack mem. A reg. pair is possible in 8085.

(v) Temporary reg. → * W, Z are the 2, 8 bit temp. reg. which are not accesible by the user (or) programmer.
* They are used by the processer in some ins.

(vi) Falg reg. →

D7	D6	D5	D4	D3	D2	D1	D0
S	Z	⊗	AC	⊗	P	⊗	CY

S - Sign AC - Auxiliary Carry CY - Carry
Z - Zero P - Parity

* There are 5 flags in 8085 which give the status after an ALU operations.
As the result for most of the ALU operation is present in accu. flags are affected or modified by the content of accu. except for few ins.

S → 0, 1 } According to D7 bit of accu.

In signed operations → 0 → +ve Result
1 → -ve Result

Z → 1, if result in A → 00H
0; otherwise.

* zero flag is also affected for GPR in some ins.

Eg:- Increment & decrement. [INR & DCR]

* AC → 1; If there is a carry from D_4 bit ← D_3 bit
(or)
Higher nibble ← lower nibble
0; otherwise.

* P → 1, if no of 1's in accu. → even
0; if no of 1's in accu. → odd

* Cy → 1; if there is a carry i.e. out of D_7 bit
0; otherwise.

Eg:-

1 → AC

A → 98H → 1001 1000

B → 6FH → 0110 1111

Cy 1 | 07H

Accu.

0000 0111

S Z X AC X P X Cy

0 0 0 1 | 0 0 0 1

1 1H

Eg.→

A → 7BH

+ C → 84H

FFH

1111 1111

S Z X AC X P X Cy

1 0 0 0 | 0 1 0 0

8 4H

If the result in 'A' is

00H
11H
22H
...
FFH

} P=1

Eg:-

A → 3E

D → C2

Cy 0

① 00

S Z X AC X P X Cy

0 1 0 1 | 0 1 0 1

5 5H

(2.) ALU (Arithmetic & logic Unit) → * It is a combination of accum, temp. reg., flag reg. & arithmetic & logic circuits

* The various arithmetic operation are add^n, sub^n, increment & decrement.
* The logical operation are AND, OR, X-OR, compare, compliment, Rotate.

Note:- * There are no seprate ins for multiplication & devision in 8085.
* Flags are affected (or) modified only for arithmetic & logic operation.

(3.) Timing & Control unit → * It is responsible for generating various control, timing & status signal req. for operations of the processer.

X_1X_2 → * A crystal oscillator is connected b/n X_1 & X_2 to provide clock freq.

* A crystal is used as it provides more stable oscillations compare to RC & LC oscillator which are temp. depandants.

* An internal clock gen^r takes the ref. clock & produces operating freq. which is half of crystal freq.
* It also provides other freq. required for internal operations of the processer.
* The operating freq. of 8085 is 3MHz.
* It can range b/n 3MHz - 6MHz

$$f_{clk} = \frac{f_{crystal}}{2}$$

ALE → * It is used to make AD_0-AD_7 either as add. (or) data bus.

1; → All 16 lines → Add. bus
0; → A_{15}-A_8 → Add. bus
AD_7-AD_0 → data bus

$\overline{RD}$ → Read control signal
0; Active

$\overline{IOR}$ → Write control signal

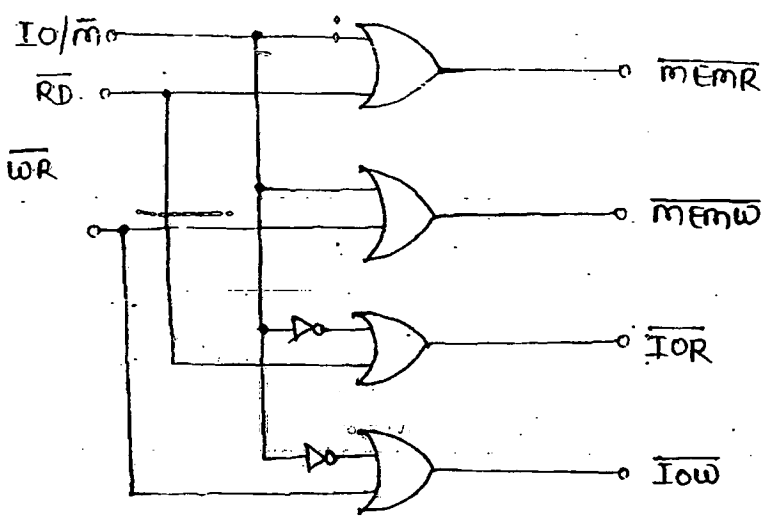
0; Active

IO/$\overline{M}$ → Status signal

0; m/m operation

1; I/O operation

IO/$\overline{M}$	$\overline{RD}$	$\overline{WR}$	Operation	Control signal
0	0	1	Fetch / m/m read	$\overline{MEMR}$
0	1	0	m/m write	$\overline{MEMW}$
1	0	1	I/O Read	$\overline{IOR}$
1	1	0	I/O write	$\overline{IOW}$



Status lines (S_1 S_0) → * These are status lines which give the status of bus cycle for an operation.

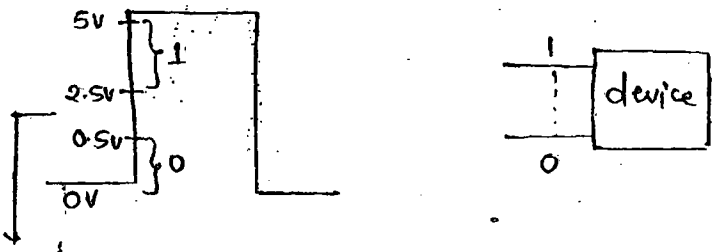
IO/$\overline{M}$	S_1	S_0	Status
Z	0	0	Halt
0	0	1	m/m write
0	1	0	m/m Read
0	1	1	Fetch

High impedance (or) Tristate (or) Floating →

1	0	1	I/O w
1	1	0	I/O Re
1	1	1	$\overline{INTA}$

Interru Acknol

Z	X	X	HOLD
Z	X	X	Reset



0.5V - 2.5V (confussion (or) ambiguity)

↓

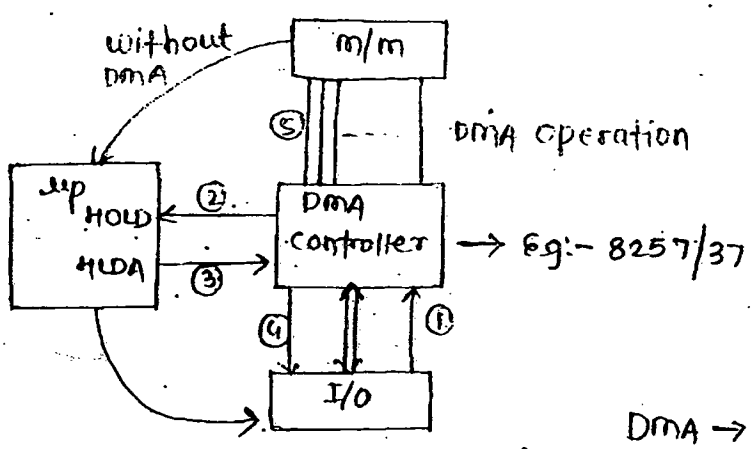
These must be avoided

↓

Buffer

Z → High impedance state → line doesn't draw any current.

HOLD & HLDA →



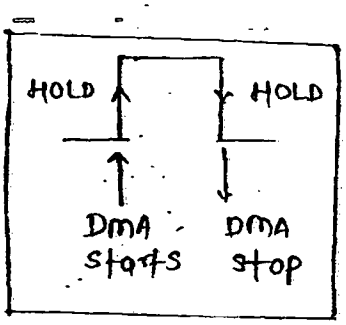
1B → 2 insts

1KB → 1024 x 2

2048 B

→ Eg:- 8257/37

DMA → Direct mem. Access



* when more data is to be Xferd b/n mem. & I/O at a faster rate DMA operation is used with the help of DMA controller

HOLD → * High active I/p signal to the processer from DMA controlle requesting the buses for DMA operation.

* The processer completes current operation (or) m/c cycle after receivi HOLD req & responds with HLDA (Hold ack).
* After this state the control of buses is granted to DMA controller
* After a complition of data Xfer b/n mem & I/o the Hold signal is pulled down by DMA controller.
  Therefore control of buses is accessed by processer
* This process is known as DMA.

HLDA → * o/p signal in responce to HOLD req.

Que → DMA is performed b/n ?
(a) m/m & ip (b) m/p & I/O (c) m/m & I/O (d) None
Ans → m/m & I/o.

Note:- *Without using DMA 2 ins are required to Xfer 1 byte.
* When DMA is performed processer would be in wait / idle state.

* Modes of DMA operation →
(1.) Burst mode
(2.) Cycle stealing tech. (or) short burst mode.

(1.) Burst mode → *A HOLD signal remains high unless untill the data is completely Xfered b/n m/m & I/o.
* The processer is present in wait state for more time.
  Therefore more interference.
  Eg:- Xfering data from Hard disk.

(2.) Cycle stealing tech → The total data to be Xfered b/n m/m & I/o is devid into blocks.
* Blocks are Xfered in a seq. of interval of time.
* The time of processer is efficiently used as it is interrupted only whe there is a data Xfer.
  Therefore less interference.

eg:- Audio fn & File Xfer appn at a time.

Interleaved DMA → *The buses of the processer are used when opcode fetch is being done i.e decoding operation.

* The DMA controller uses the buses to Xfer one byte for each opcode.

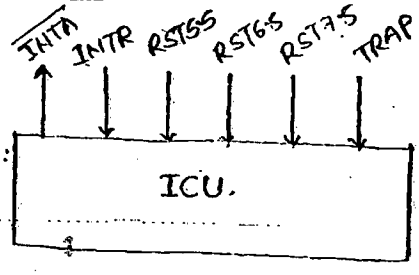
Reset-in → * Low active i/p signal to reset the processer.
* Program counter is initialised to 0000H

Reset-out → * It indicates that processer is reset.
* It can be connected to I/o device to reset it.
* It is an o/p signal.

Clock-out → * O/p pin on which the same operating freq of the processer is available.
* It can be used to connect to the clock I/p of I/o for synchronous operation.

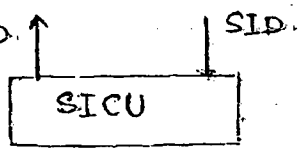
Ready → * I/p signal to the processer for a from a slow speed I/o.
* If ready is high then only processer will either transmitt data to (or) receive data from I/o device.

(4.) Interrupt Contol unit →



* Interrupt is a method by which an I/o device informs the processer that it requires the service of processer.
* There are 5 hardware interrupts in 8085 acc to priority.
(1.) TRAP / NMI / RST4.5 | (NMI → Non-maskable interrupt)
(2.) RST 7.5
(3.) RST 6.5 } Restart
(4.) RST 5.5
(5.) INTR → Interrupt request.

(5.) Serial I/o Control Unit →



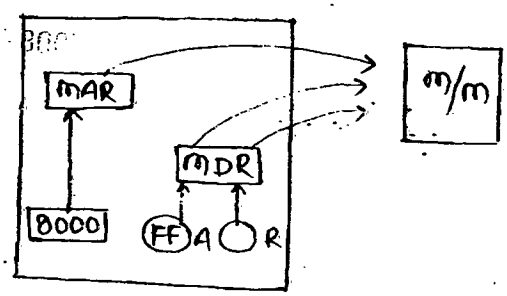
* It is used for serial comm. b/n processer & I/o devices.
  SID → Serial i/p data pin. (Used to received serial data)
  SOD → Serial o/p data pin. (To Xmitt serial data)
* Both SID & SOD are internaly connected to D7 bit of accu.

* MAR (mem. add reg.) → * It is used to Hold the add. before it is placed on add. bus.

* MBR/MDR (mem. buffer/ data reg.) → * It is used to hold the data before it is Xfered to m/m (or) when the data is accessed from m/m.

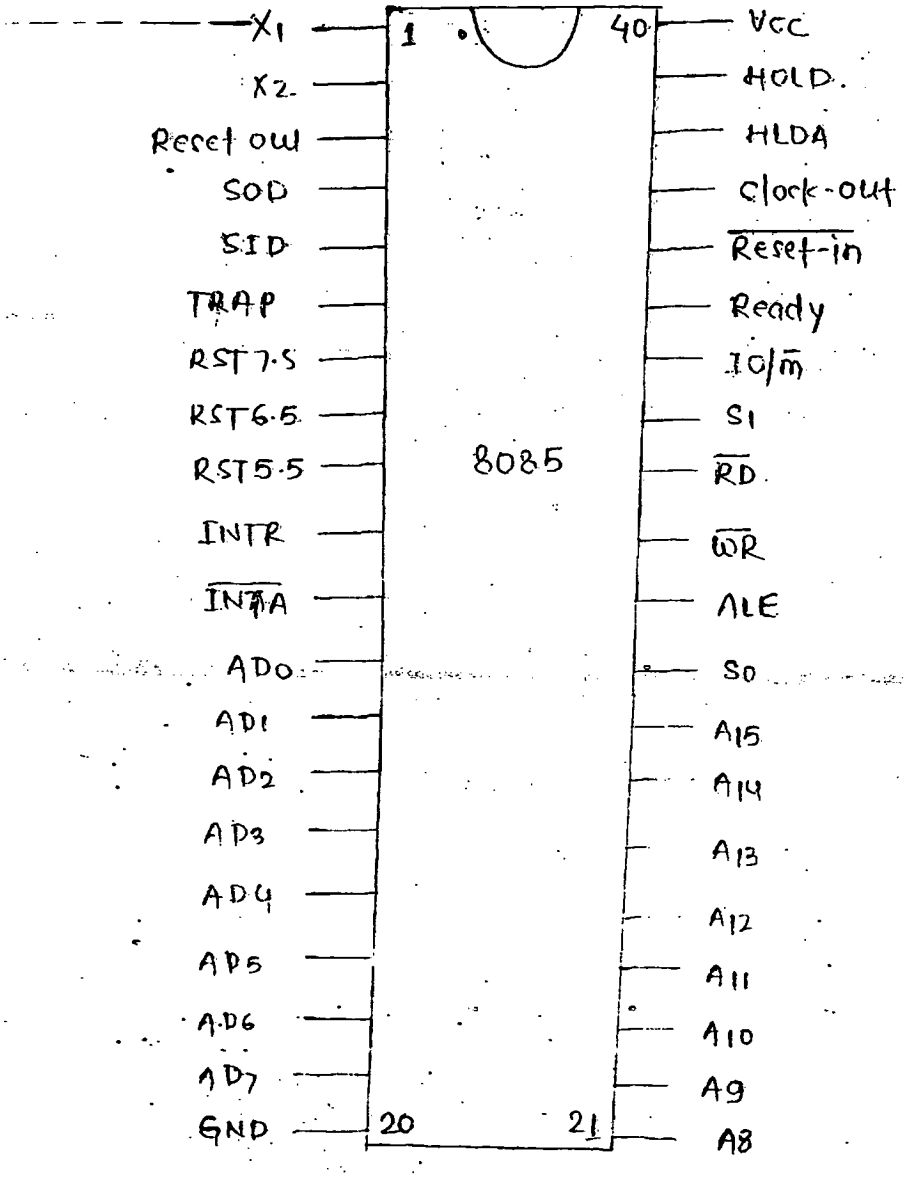
*

* MAR & MDR are internal reg. which are not accesible by the user.



Pin discription → * 8085 is an 8bit processer having 16 add. lines.
* It can addressed a m/m of 64KB.
* It require +5V DC supply.
* Operates with 3MHz.
* It is a pin of 40 pin IC, Present in DIP (dual inline package).

8080	8085
f = 2MHz	f = 3MHz
ϕ_1, ϕ_2	ϕ
Slow in operation	Faster than
±5V, +12V	+5V



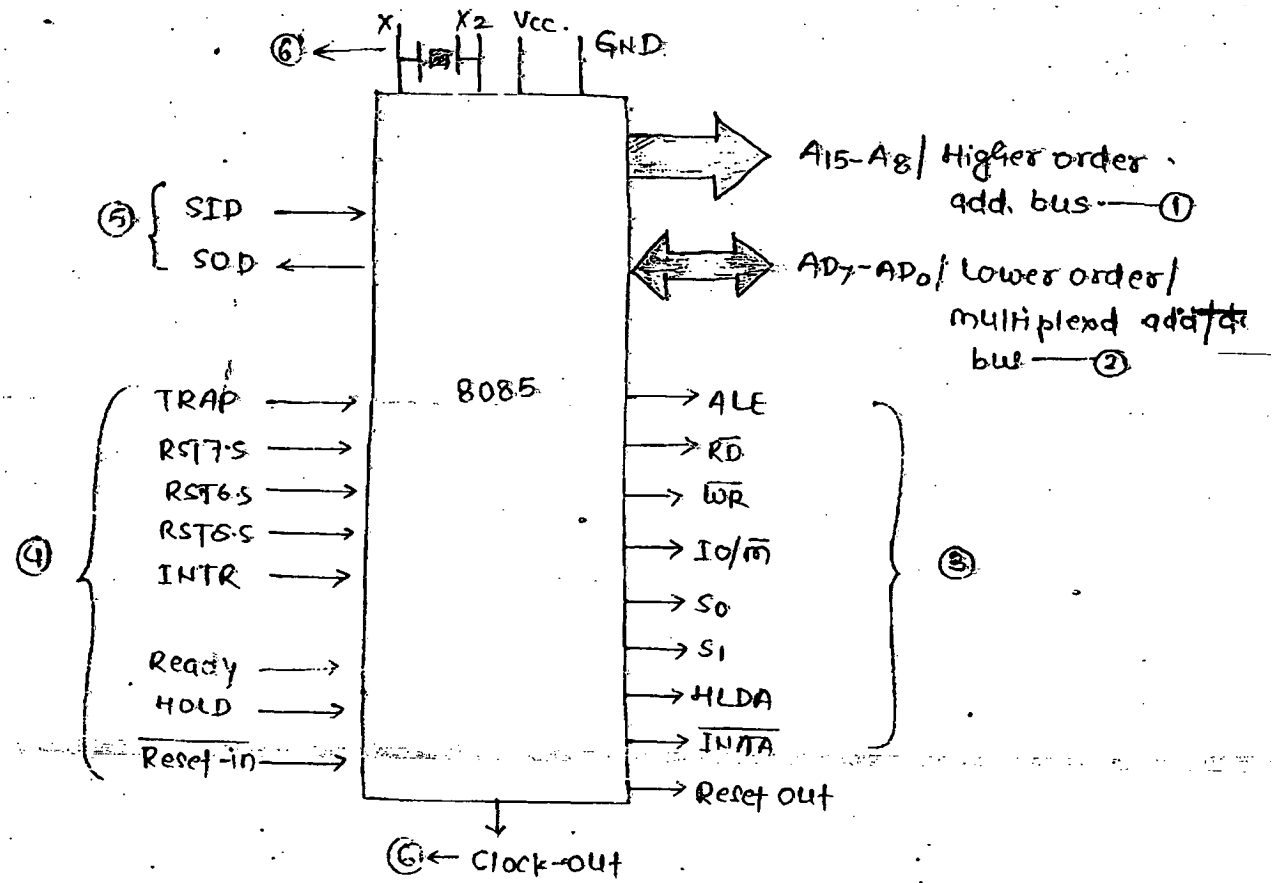
* Functional description →

(1.) Classification of signals

(2.) Interrupts.

(1.) Classification of signals → * Signals in 8085 are grouped into 6 types

(1.) Address bus
(2.) Data bus
(3.) Control & status signals
(4.) Interrupt & externally initiated signals.
(5.) I/O serial ports
(6.) power supply & freq. signal.



Interrupts → * It is an internal (or) external signal which may disturb (or alter the seq of execution of the processer.

* This are classified as:-

(1.) Hardware & Software.

(2.) Maskable & Non-maskable.

(3.) Vectored & Non-vectored.

(1.) Hardware interrupt → * There are 5 hardware interrupt acc. to priority.

(a.) TRAP
(b) RST7.5
(c) RST6.5
(d) RST5.5
(e) INTR

* They are physicaly connected to the I/o device (or) appn.

Software interrupts → * There are 8 software interrupt which can be used as ins and also along with INTR.

*

RST0
RST1
RST2
⋮
RST7

RST n (n 0-7)

(2.) Maskable interrupt → Interrupts which can be ignored (or) avoided when they are triggered (or) maskable.

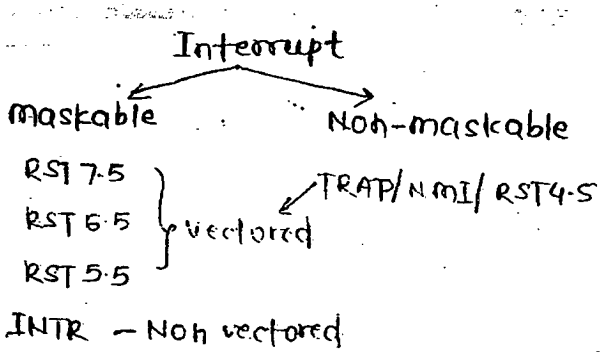
Non maskable → Interrupts which can't be ignored (or) avoided. are non-maskable.

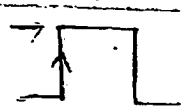
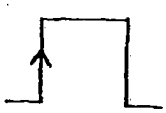
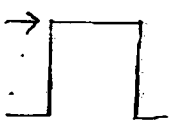
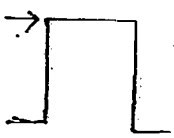
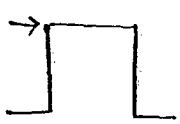
** Eg → TRAP is the only NMI in 8085. It must be connected to highly prioritised event in practical applications.

(3.) Vectored interrupt → Interrupts which have specific add. loch in the m/m.

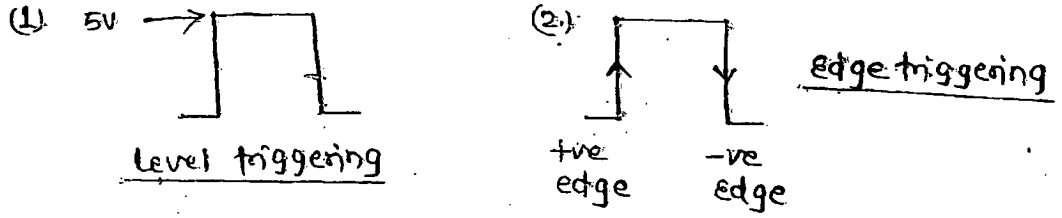
Non-vectored → Interrupts which do not have specific add. loch in the m/m.

** Eg → INTR is the only non-vectored interrupt in 8085.

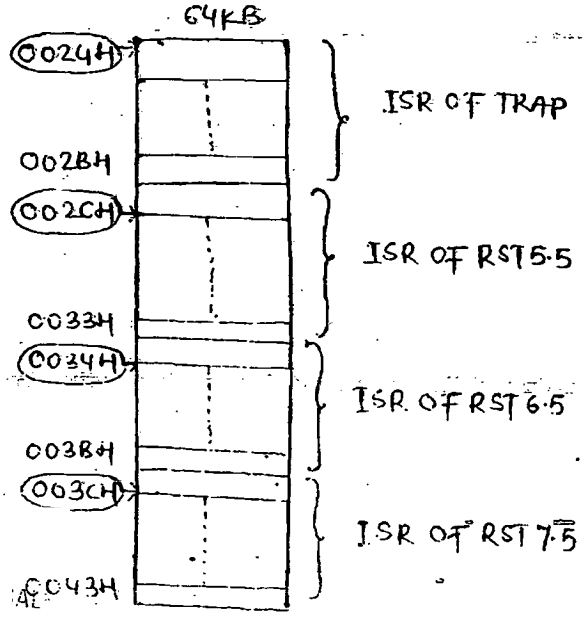


Priority	Interrupt	Type of triggering	Vector add.
1.	TRAP		0024H
2.	RST 7.5		003CH
3.	RST 6.5		0034H
4.	RST 5.5		002CH
5.	INTR		—

★ Types of triggering →



★ Interrupt Service Routine

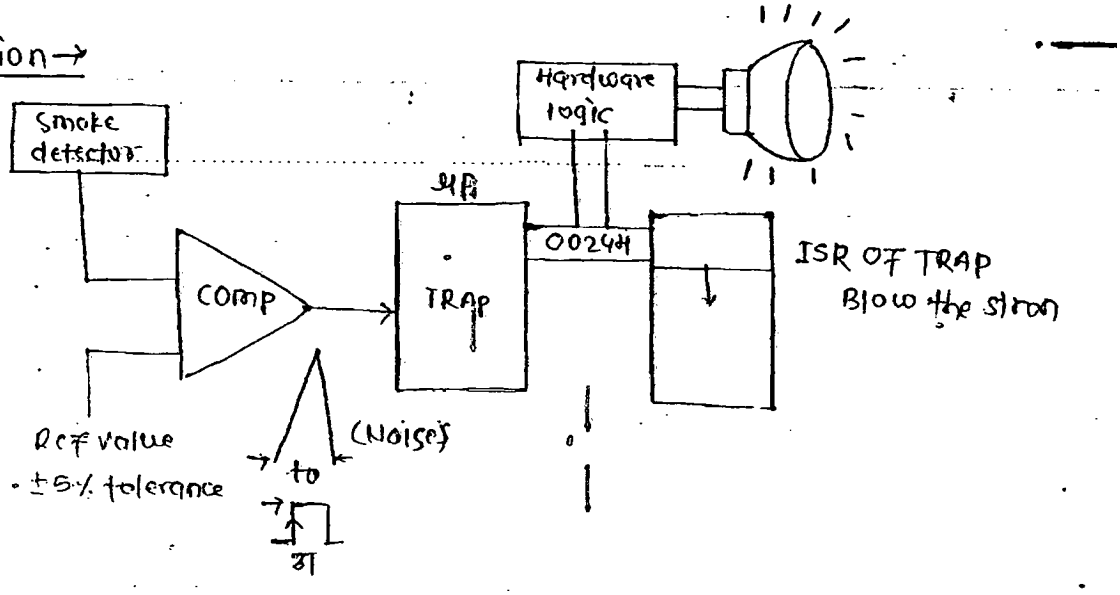


★ An I/O device which is connected to interrupt pin has some appli'n prog. to be executed known as interrupt service routine (ISR)

(OR)

A prog. that must be executed in responce to an interrupt is known as ISR.

Application →

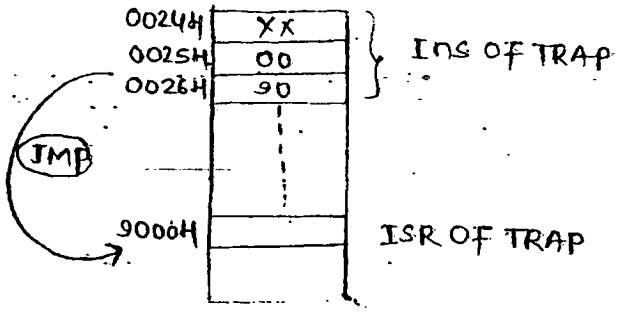


* In the above eg a siren must be blown when smoke (or) fire is detected by the sensor.
* TRAP is both edge & level triggered.
* It is edge triggered so that it may be responded quickly.
* It is level triggered in order to differentiate b/n original signal from practical applⁿ & error signal due to noise.
* The signal on TRAP pin must be high for atleast 3 clock periods in order to avoid error signal due to noise.

$$f = 3MHz, \quad T = \frac{1}{f_{clk}} = \frac{1}{3\times10^6}$$

$$\boxed{T = 0.3\mu s}$$

* If the length of ISR is more than 8 bytes (or) when hardware & software interrupts are to be used an unconditional jump ins can be included such that the prog. can be continued further.



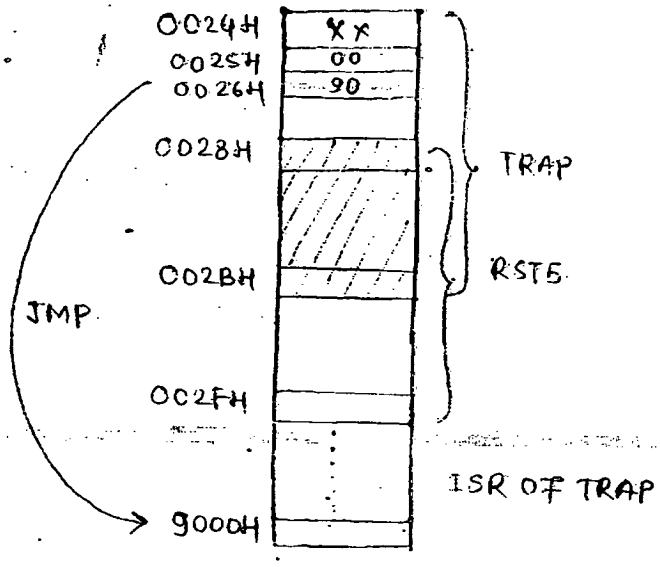
- Assumption
  - RST0 → 0000H - 0007H
  - RST0.5 → 0004H
  - RST1 → 0008H - 000FH
  - RST1.5 → 000CH
  - RST2 → 0010H - 0018H
  - RST2.5 → 0014H
  - RST3 → 0018H - 001FH
  - RST3.5 → 001CH
  - RST4 → 0020H - 0027H
- Hardware Interrupt
  - RST4.5 0024H (TRAP)
  - RST5 → 0028H - 002FH
  - RST5.5 002CH

RST 6 → 0030H - 0037H

Hardware Int. → RST 6.5 → 0034H

RST 7 → 0038H - 003FH

RST 7.5 → 003CH



* Vector Addⁿ for Interrupts →

$$n \times 8 = (x)_{10} = (y)_{16} \qquad n = 0 \cdots 7$$

eg:- (1.) RST 6

$6 \times 8 = (48)_{10} = 0030H$

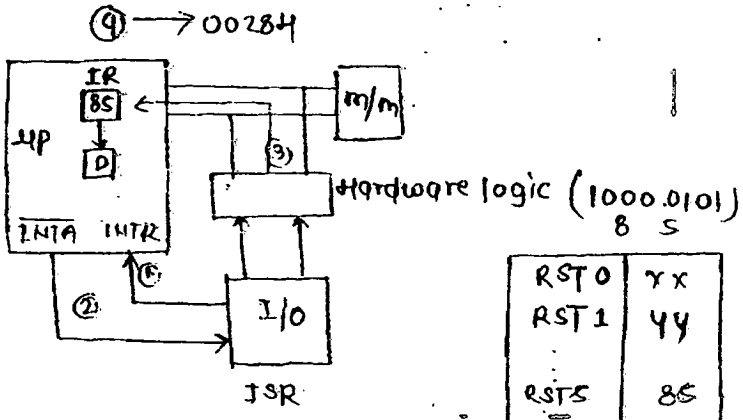
16 | 48
3 0 → $(30H)_{16}$

(2.) RST 7.5

$7.5 \times 8 = (60)_{10} = 003CH$

16 | 60
96 12
→ $(3C)_{16}$

INTR →



RST 0	xx
RST 1	yy
RST 5	88
RST 7	zz

* In order to use INTR a programmer mst select one of the software interrupts vector addresses to store the comsponding ISR of I/o.
* An external hardware logic ckt can be designed to produce (or) generate the opcode of selected RSTN.
* The INTR pin is verified by the processer in the last clock period of an ins cycle.
* If it is present the processer responds with $\overline{INTA}$ & waits for the responce from I/o.
* The external hardware logic provided the opcode which is accessed into ins. reg., decoded & the contol of prog. is transfered to vector add. of RSTN. & execution continews.
* As there is no specific add. loch for INTR it is known as non-vectored interrupt.

Note:- * $\overline{INTA}$ is req. only for INTR not for vectored interrupt.

* Instruction Related to interrupts →

- EI = Enable interrupt
- DI = Disable interrupt
- SIM = Set inte. mask

SIM → * Use to mask the interrupt (or) make them available

- RIM = Read interrupt mask

RIM → * Use is to know the status of pending interrupts

* The above ins. are mention for maskable interrupts only.

Que → Explain the interrupts in 8085.

Programming model →

Program → set of instructions

Instruction → It is a command given to a computer to performs specific task.

m/c level language (it is a binary representation of ins specific to a sys.)

Assembly level language → Ins are written in seprate words mnemonics which are partialy in understood by the prog.

*** Both assembly & m/c level Langbages are together known as low level languages.

MOV A, B ⟶ 78H

ADD C ↑ Assembly

01111000 ↑ m/c code

High level language → They are m/c indepandant

Eg. → C, C++, JAVA

Softwares → compiler, → It converts high level language to m/c level language where the entire prog. is converted at a time.

Eg:- Turbo C, XLC, JavaC

Interpretor → It converts H.L.L to m.L.L line by line.

Eg:- M-basic

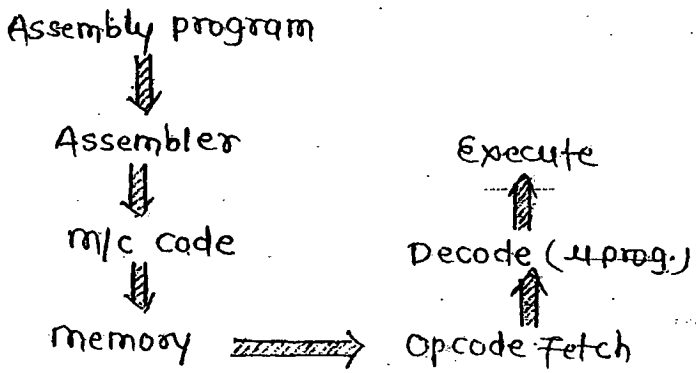
Assembler → It converts assembly prog. to m/c code.

Eg:- MASM – Microsoft macro assembler

Loader :- Hex file → Binary (object file/compile)

↓

Executable code



* Basic steps of execution of an ins. →

(1.) Fetch (2.) Decode (3.) Execute.

Que. → Describe the programming model in 8085?

Instruction format →

Opcode	Operands

Opcode → Operation code

* It indicates the type of operation to be performed for an ins.

Operands → Data

* It is the data on which operation is to be performed.

* Some ins. may have opcode & operand together as a single byte others may have one byte of opcode & one byte of operand (or) 2 byte of operands.

Length of ins. → * No. of byte an ins occupies in the mem.

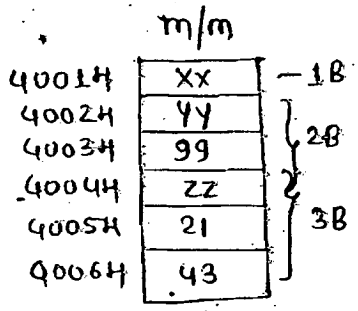
* There are 3 types of ins. classified acc. to length.

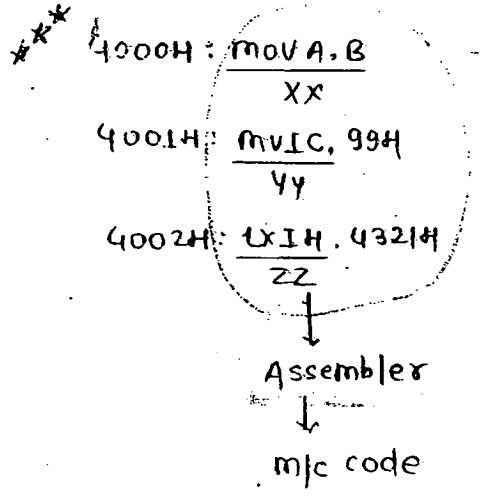
(1.) 1 Byte/1 word ins.

(2.) 2 Byte/2 word ins.

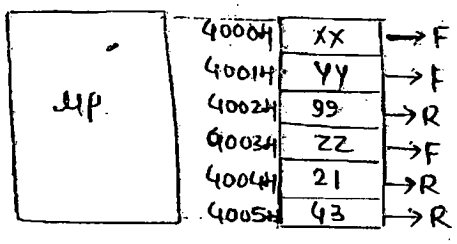
(3.) 3 Byte/3 word ins

1 Byte/1 word	2 Byte/2 word	3 Byte/3 word
MOV A, B	MVI C, 99H	LXI H, 9321H
Opcode — operand	Opcode — operand	Opcode, operand
1B	1B 1B	1B 2B





m/m representation of a prog.



Memory Rule → * In all m/m related operations the lower byte of data (or) lower reg. content is transfer to lower add. locn.

* Similarly higher byte (or) higher reg. content is TF to higher add. locn (or) vice-versa.

* Every reg. in the processer is given unique code.

* There are 74 diff. opcode in 8085 which result in 246 ins.

74 opcodes — 246 ins.

(1) MOV RD, RS
01DDDSSS
MOV A, B
↓
Assembler
↓
01111000
78H

(2) MOV A, A - XX
A, B - YY
⋮
MOV B, A
B, B
⋮
MOV M, M X (not possible)
(63 ins)

B - 000
C - 001
D - 010
E - 011
H - 100
L - 101
A - 111
M - 110 (m/m related)

BC - 00
DE - 01
HL - 10
SP - 11

* Indentifying length of ins. →

2 Byte ins. → If an insⁿ contains one byte of operand (Add or data) i.e. of 2 byte.

3 Byte ins. → If an insⁿ has 16 bit value i.e. add or data length is of 3 bytes.

Remaining all insⁿ are one byte in length.

* Addressing Mode → * These are various formats specifying the operands (or) they indicate how data is accessed for an insⁿ.

* There are 5 addressing mode in 8085:-

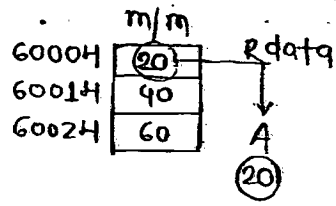
(1.) Immediate Am.
(2.) Direct Am.
(3.) Indirect Am
(4.) Register Am
(5.) Implicit/Implied Am.

(1.) Immediate Am → Data is present in the ins itself.

Eg. → MVI B, 74H
↓
data
(mov immediate)

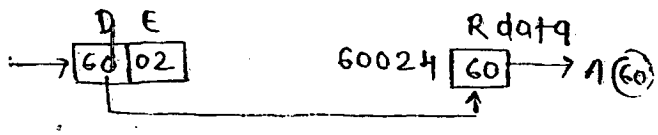
(2.) Direct Am → Add. of the data is present in the insⁿ

Eg. → LDA 6000H



(3.) Indirect Am → Add. of the data is present as content of another reg. pair.

Eg. → LDAX D [load accu. with data present in D]



Note :- X in an insⁿ indicates that operation is performed on a reg. pair.

(4.) Register Am → Data is transfered b/n reg.

Eg.→ MOV C, D.

(5.) Implicit Am → *No operands (or) data is present in the insn but som of operⁿ is performed by the insn

* The operation may be performed in a single reg.

Eg.→ CMA (comp. accu)

A

FF → 00

* Timing diagram → * It is a pictorial representation of execution of an insn with the help of various control, timing & status signals

T-states → * It is one subdivision of an operation performed in one clock period.

* Subdevision are internal states synchronised with sys clock.

$$f_{clk} = 3MHz; \quad T = \frac{1}{f_{clk}} = \frac{1}{3 \times 10^6} = 0.33\mu s$$

T = 0.33 μs

m/c cycles → * It is a time req. to access either m/m or I/o device.

* It is also equilant to the time req. to IF a data byte to m/m. (or) I/o.

1 m/c → 3-6 T states

Insn cycle → * The time req. to complete the execution of an insn.

* One insn cycle may have 1-5 m/c cycle

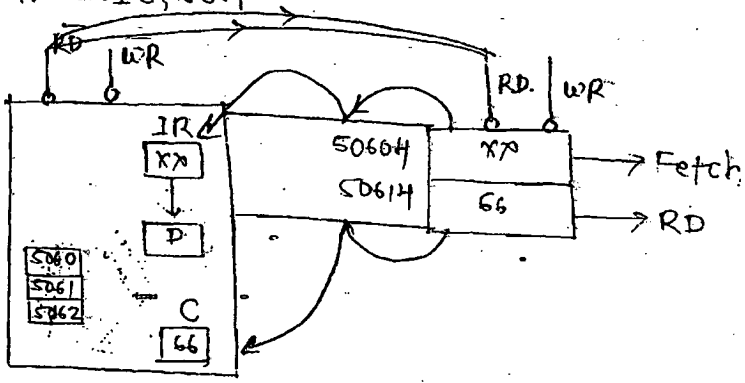
1 Insn cycle – 1-5 m/c cyde.

Note → The maxᵐ no. of T-state possible for execution of an insn in 8085 is 18.

* eg:- CALL 16 bit add. → 3Byte, 5 m/c cycle's, 18 T-state

It is the only insn have long time.

Eg:- 5060H: mvi C, 66H



Basic steps → (1) Fetch (2) decode (3) execute

ALE → 1; All 16 lines → Add. bus

0; A15–A8 → Add. bus

AD7–AD0 → data bus

S_1	S_0	State
0	0	Halt
0	1	write
1	0	Read
1	1	Fetch

Reason for Unknown

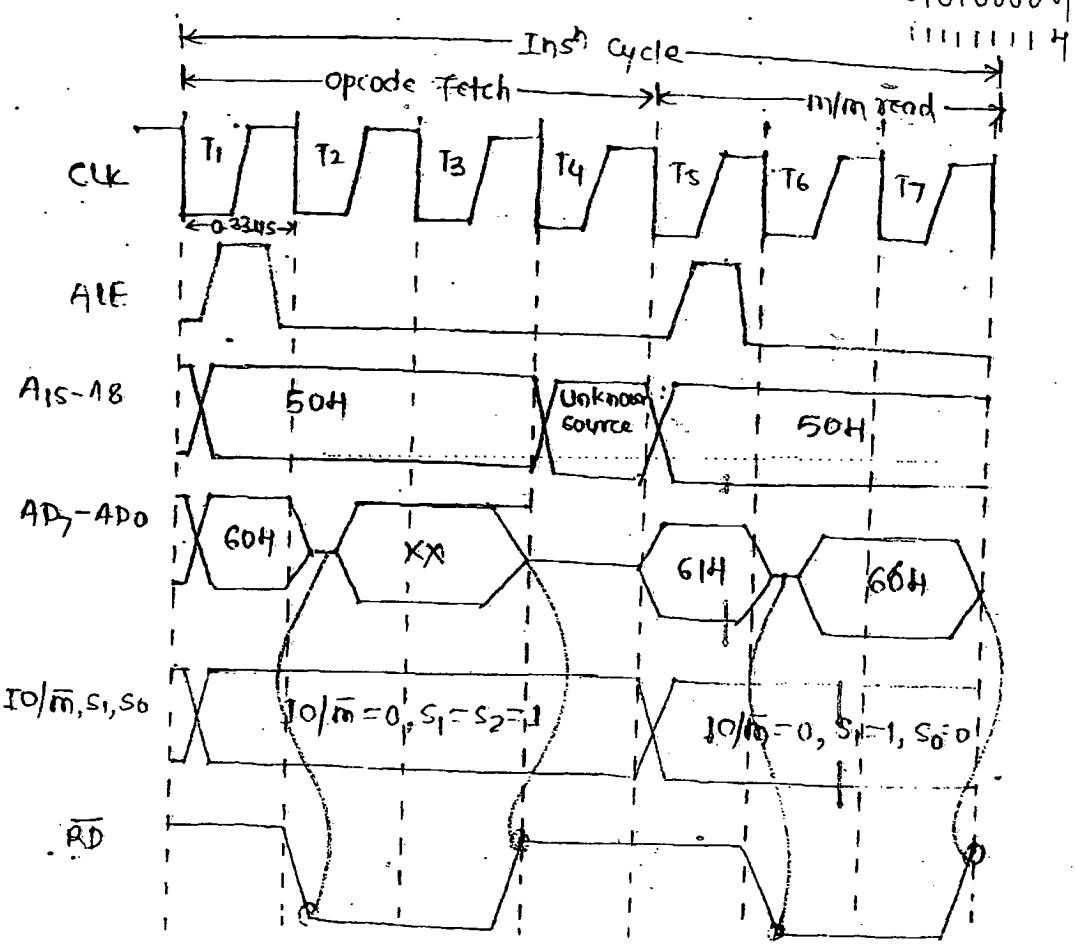
5060H

↓

FF00

01010000 H

11111111 H



* Opcode fetch is the 1st m/c cycle for execution of any ins^n which may be 4 (or) 6-T states.

Opcode fetch →

T_1 → * ALE is high indicating all lines act as add.
* A_8-A_{15} has higher byte of pc.
* AD_0-AD_7 has lower byte of pc.
* $IO/\overline{M} = 0$, indicating mem. operation.
* $S_1 = S_0 = 1$ for Fetch.
* $\overline{RD} = 1$ (inactive) as there is no data bus available.

T_2 → * As ALE becomes low AD_0-AD_7 act as data bus.
* When $\overline{RD}$ is activated opcode from m/m loc^n is accessed on to data bu

T_3 → * Opcode from the data bus is accessed into IR.

Note → It requires 2 T-states to access either an opcode (or) an data byte from m/m loch into the processer after the add. is latched.

T_4 → * Opcode is decoded & execution may also be completed for some ins^n like mov A, C, but for the ins^n considered a m/m Read operation is required.

Memory Read →

T_5 → * ALE is high to point next add.

T_6 & T_7 are similar to T_2 & T_3 except that $S_1 = 1$, & $S_0 = 0$ for RD.
In this ins^n 66H is accessed into reg. C in T_6 & T_7.

Note:- * ALE is high in the 1st T-state of a m/c-cycle.
* Status of higher byte of add. bus in 4th T-state is unknown as the processer may point to the new add. in a very next T-state in some ins^n after decoding.
* The value of pc is changed from 5060H to 5061H in the 1st T-state after the add. is latched.

Conclusion →

(1.) length of ins^n – 2byte
(2.) No. of m/c's – 2 (F:R)
(3.) Total T-states – 4 + 3
(5.) No. of ins^n cycle = 1

(4.) Total execution time
= clock period x No. of T-state x count value
= $\frac{1}{f_{clk}} \times T\,state \times CV$
= $\frac{1}{3 \times 10^6} \times 7 \times 1 = 2.334\,\mu s$

Count value → No. of times an ins^n (or) set of ins^n executed.

T-states

	min	max
Fetch	4	6
m/c cycle	3	6
instruction	4	18

MVI C, 66H ⟶ 2B, 2 mc/c, 7 T-states

Note:- * The no. of m/c cycles req. for execution of ins^n may (or) may not be equal to length of ins^n.

* For the above eg. no. of time $\overline{RD}$, $\overline{WR}$ & ALE control signals activated are 2, 0 & 2 respectively.

Instruction set Classification

Instruction

Type of operation
(1.) Data Xfer/copy inst.
(2.) Arithmatic inst.
(3.) Logical int
(4.) Branching inst.
(5.) M/c Control inst.

Length of inst.
(1.) 1 Byte/word instⁿ
(2.) 2 Byte/word instⁿ
(3.) 3 Byte/word instⁿ

(1.) Data Xfer/copy instⁿ → * In this group the data is xfered from source to destination.

* It may be in between :-
(i.) Reg-Reg (ii) Reg ↔ m/m (3.) Reg ↔ I/o (4.) Reg ↔ data.

* Terminology used is data Xfer operation →

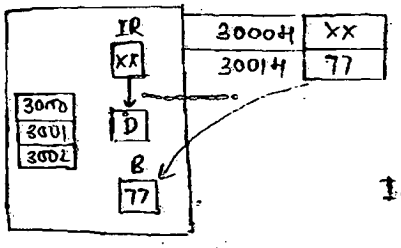
M - move
L - Load - Access from m/m
S - (store) - Send to m/m
PUSH Rp - Rp ⟶ stack m/m
POP Rp - Rp ⟵ stack m/m
IN - Accumulator ⟵ data from i/p port/devices.
OUT - Accumulator ⟶ Send to o/p port/devices.

Note → In almost all data Xfer operations the content of source is unchanged after the operations.

	length of instⁿ	No. of m/c	T-states
MVI R, 8 bit data -	2B	2	7 (4+3)

↓
move immediate 8-bit data to reg.

Ex:- 3000H: MVI B, 77H

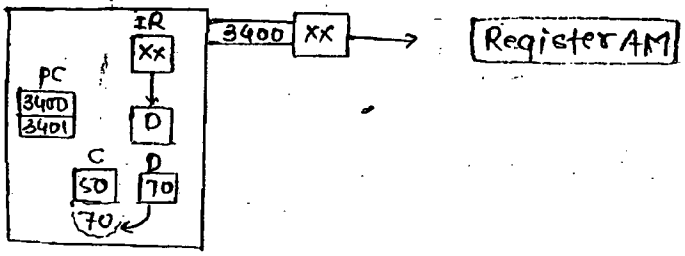


Immediate addressing mode.

* MOV Rd, R_{BS} → 1B, 1 m/c cycle, 4T

mov/copy/xfer the content of source reg. to destination reg.

Eg:- 3400H: MOV C,D



* LXI Rp, 16 bit value - 3B, 3 m/c cycle, 10T

Load immediate 16 bit value into reg. pair.

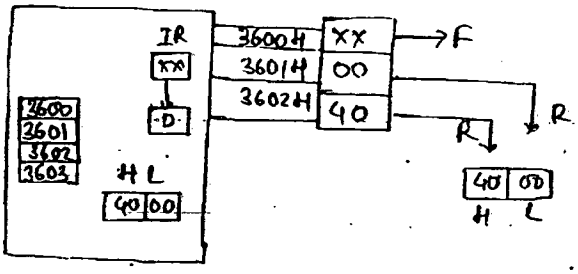
X → pair; LXI B, _____

LXI D, _____

LXI H, _____

LXI SP, _____

Eg:- 3600H; LXI H, 4000H



Immediate AM

RD	WR	ALE
3	0	3

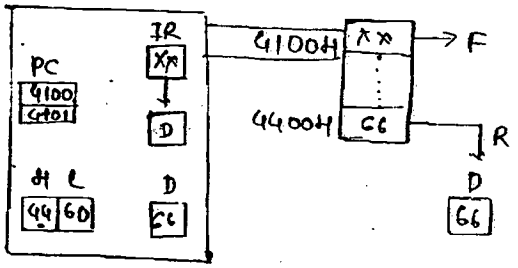
(F+R+R)

F R R
4 3 3 = 10T

* MOV R,M - 1B, 2, 7

MOV the content of m/m to reg. (Ref of m/m H,L)

Eg:- 4100H: MOV D,M



Indirect AM

F+R = 7

LXI H,4500

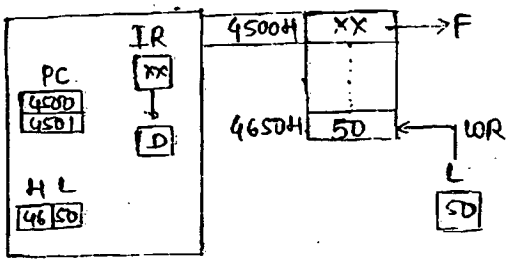
MOV D,M - add. (HL)

MOV L,M - data

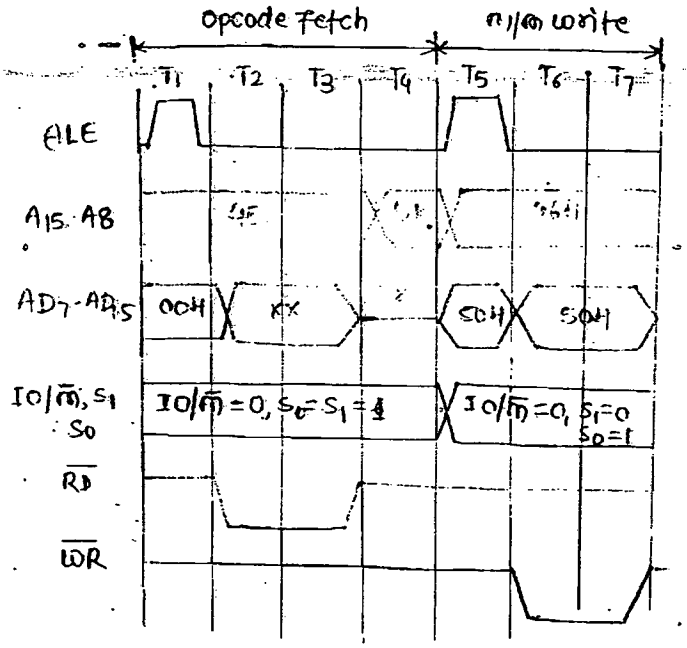
* Mov M,R → 1B, 2, 7

Mov the content of reg. to m/m,

Eg:- ~~4500H : MOV M,L~~ 4500H; MOV M,L



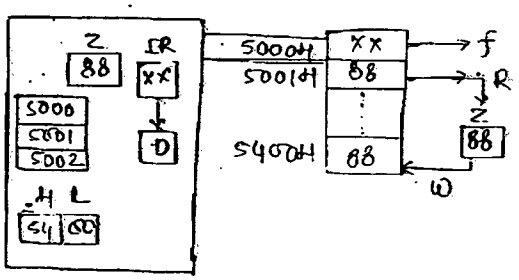
F+WR=7
(4) (3)
Indirect Am



* MVI M, 8 bit data - 2B, 3 m/c, 10T

Mov immediate 8 bit data to m/m

Eg:- 5000H, MVI M, 88H



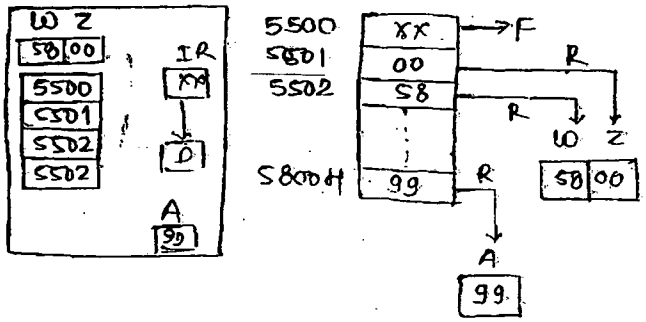
F R W
4 3 3 → 10T
Indirect/Immediate
Am

RD | WR | ALE
2 1 3

* LDA 16 bit add. – 3B, 4 m/c, 13T

Load accumulator with data present at 16 bit add.

Eg:- 5500H: LDA 5800H

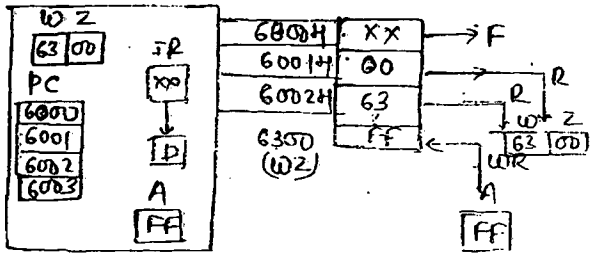


F+R+R+R = 13
Direct Am

* STA 16 bit add. – 3B, 4m/c, 13

Store the content of accumulator at 16 bit add.

Eg:- 6000H: STA 6300H; if [A] → FFH

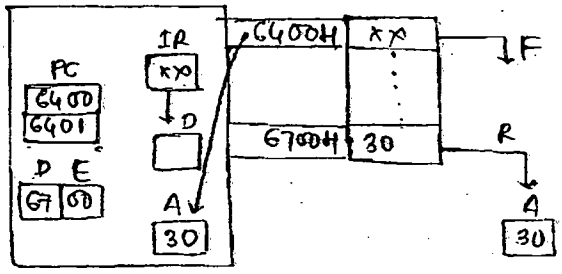


F+R+R+W = 13T
Direct Am

* LDAX Rp – 1B, 2 m/c, 7T

. Load accumulator with data present at add. of reg pair.

Eg:- 6400H: LDAX D



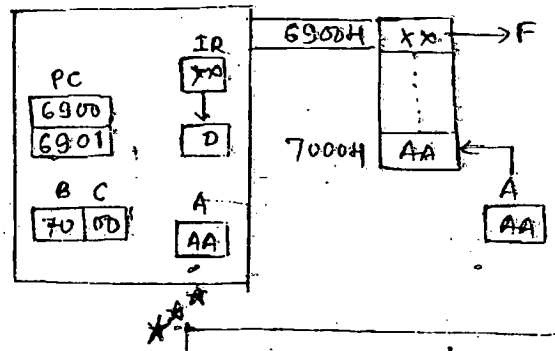
✓LDAX B
✓LDAX D
X LDAX H = ~~MOV A,M~~

* STAX Rp – 1B, 2m/c, 7T

Store ~~accum~~ the content of accumulator at the add. present in reg pair.

✓STAX B
✓STAX D
STAX H = ✓MOV M A

Eg:- 6900H: STAX B



F+W=7T
Direct AM

BC	LDAX B	STAX B
DE	LDAX D	STAX D

HL → MOV R, M
MOV M, R

So HL is known as default reg.

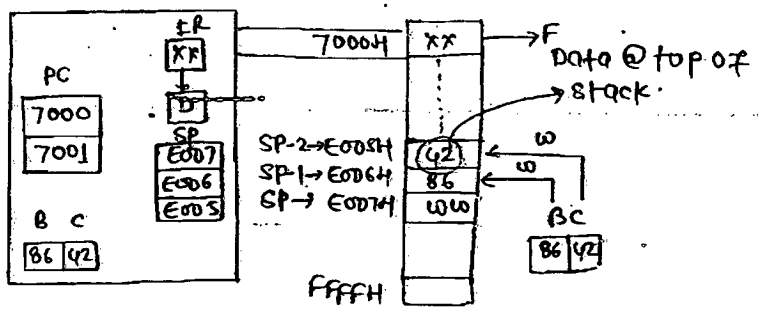
* PUSH Rp - 1B, 3 m/c 12T
Store/push the content of reg. pair into stack m/m.

SP = SP-2 stack grows down

PUSH B
PUSH D
PUSH H
PUSH PSW → To access flags

F+W+W
6+3+3=12T

Eg:- 7000H : PUSH B



Que → LXI SP, D409H
HL → 1234H
After PUSH H :→ SP?
Top of stack →?

Ans → SP-2 → D407 | 34 |
SP-1 → D408 | 12 |
SP → D409 | ww |

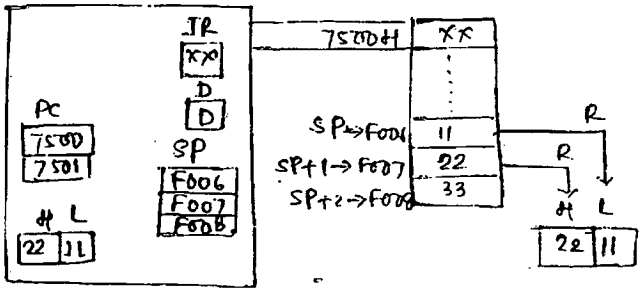
H: 12 | 34

* POP Rp- 1B, 3 m/c, 10

Read (or) access data present at top of stack (2 bit) pointed by SP into reg. pair.

POP B
POP D
POP H
POP psw ⟶ flags are modified.

Eg:- 7500H; POP H



* IN 8bit port add. ⟶ 2B, 3, 10

Read (or) access data from 8 bit port address into accumulator.

Port indicates connection of an I/O device.

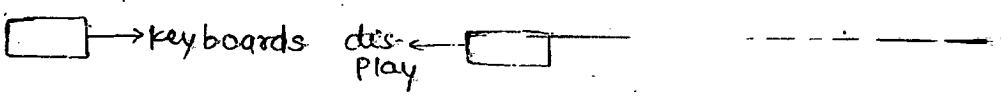
$2^8 \longrightarrow 256$

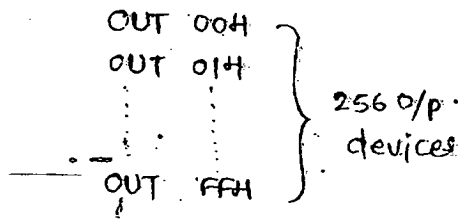
0000 0000

0 0H

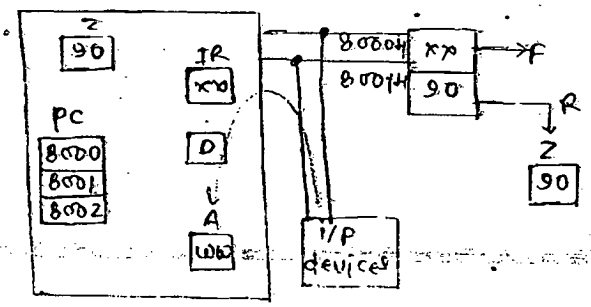
FFH

* As I/O is given 8 bit port add. (256 i/p devices & 256 o/p devices are possible)
* This method of identifying an I/O device with 8 bit port add. is known as I/O mapped I/O
* If an I/O has 16 bit add. it is known as m/m mapped I/O.

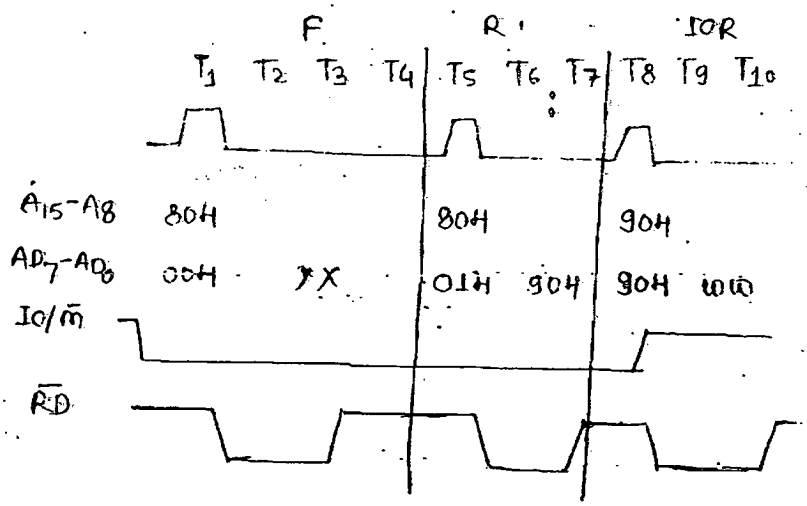




Eg:→ 8000H: IN 90H



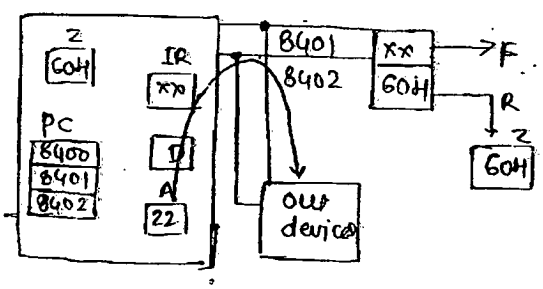
F R IoR
4 3 3 → 10
(or)
F R I
4 3 3 → 10



* OUT 8bit port add. – 2B, 3, 10

Transfer the content of accu. to 8 bit port add.

Eg:- 8400H: OUT 60H ; If [A] → 22H



F R IOW
4 3 3 → 10
(OR)
F R O
4 3 3 → 10

Que → Write the diff between I/O mapped I/O & m/m mapped I/O?

Ans →

I/O mapped I/O	m/m mapped I/O
(1.) I/O has 8 bit port add.	(1.) I/O has 16 bit add.
(2.) $2^8 \to 256$ I/P & 256 O/P devices are possible.	(2.) $2^{16} \to 65,536$ add. addresses are shared b/n m/m's
(3.) Ins't used → IN 8bit port add.	(3.) Inst'n used :- LDA 16 bit add. (All ALU oper'n related to m/m Ex:- ADDM, SUBM.)
(4.) Control signal → $\overline{MEMR}$, $\overline{IOR}$, $\overline{IOW}$	(4) Control signal → $\overline{MEMR}$, $\overline{MEMW}$
(5.) Less hardware Required.	(5) More hardware ckt is req.

(2.) Arithmatic Operations →

(1.) Addition → * It is possible with & w/o carry.

* One byte is present in accu. & result is stored in accu.

* All flags are affected.

Eg:- ADD R → ADD B (1B, 1, 4T)

A B
F9 6D

```
    A     F9
  + B     6D
  cy (1)  66
```

0110 0110
S Z X AC X P X CY
0001 0101
1 5 H

```
     1
     AA
  cy 55
  (1) 65
```

0110 0101
S Z X AC X P X CY
0001 0101
1 6 H

ADDM → 1B, 2, 7.

eg:- 9000H : ADDM

9000H [XX]
:
9300H [55]

```
 7E
 3B
 B9
```

1010

```
 AA
 55
 FF
```

1111 1111
1000 0100
8 4 H

PSW = FF84H

ADI 8bit data - 2B, 2, 7

eg - ADI 7EH

A
[3B]

```
 3B
 7E
 B9
```

ADC R - 1B, 1, 4

Eg:- ADC C

A · C · CY

(9F) (23) (1)

9F
23
1

C3

1100 0011

SZXACXPXCY

1001 0100

9 4H

ADC M - 1B, 2, 7T

Eg:- 9300H : ADDM

9300 [XX]

9500H [D8]

HL — A — CY

[95 00] [28] [0]

1
D8
28
0

166

0110 0110

SZXAC X PXCY

0101 0101

5 5H

ACI 8 bit data - 2B, 2, 7

Eg:- ACI 03H

A — CY

[B0] [1]

B0
03
1

B4

1011 0100

SZXAC X PXCY

1000 0100

8 4H

(2.) Subtraction → * It is possible with & without borrow.

* CY is treated as borrow.

* Internaly 1's comp. method is used for subtraction.

* Carry is complimented after the operation. (only 1µp method)

* In unsigned operation consider (S - ignore) CY → 0 → +ve result
1 → -ve result

* Signed operation → S → 0 → +ve result (CY ignore)
1 → -ve result

SUB R 1B, 1, 4T

Eg:- SUB D

A — D

(98) (47)

manual	processed
A - D	98 4
98	CY 1 B9 (-0)
47	↓ 51
51	0

SZXACXPXCY

0001 0000

F | 104
-4 7

B 9

-47H → B9H

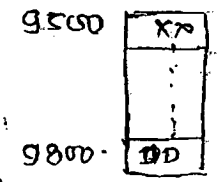
If lower nibble of result < lower nibble of 'A'

AC = 1

else = 0

SUB m - 1B, 2, 7

Eg:- 9500H SUBM



HL: 95 00 | A: DD

DD
− DD (CY = 1)
00

S Z X AC X P X CY
0 1 0 0 0 1 0 0
5 4H

SUI 8bits-data - 2B, 2, 7

Ex:- SUI CE H

A: 41

10H
3 4 1 10H 17
−CE 13
73

0111 0011
SZXACX PXCY
0000 0001
0 1H

Subtraction with borrow →

SBB R → 1B, 1, 4T

A + R + CY
A − (R + CY)
II I

Eg:- SBB H

A: 61 | B1: AB | CY: 1

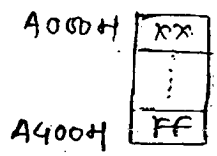
1st → B H → AB
CY + 1
AC

B1 → A
AC → 1st

D5
SZXACXPXCY
1000 0001
8 1H

SBB m - 1B, 2, 7T

Eg:- A000H: SBB m



HL: 44 00 | A: 94 | CY: 1

1
FF
+ 1
100

leavelt 94
−00
94

SZXACXPXCY
1000 0000
8 0H

SUI 8bit-data - 2B, 2, 7T

Eg:- SBI 24H

A: 30 | CY: 0

~~A+CY=80~~
24H

30
−24
0C

SZXACX XPXCY
0000 0100
0 4H

Increment → * Increment reg.

R/m / Rp+1. cy is not affected.

* Remaining flags (S,Z,AC,P → GRR's)
* Result → Operand used.
* Internaly addition is performed.

INR R - 1B, 1, 4T

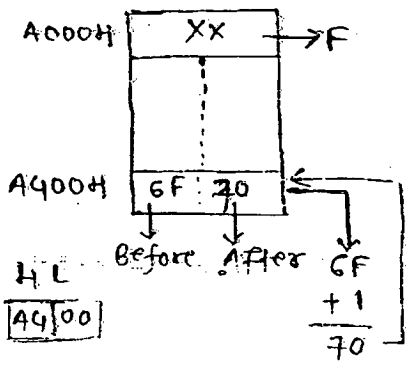
Eg:- INR C

```
   FF
Cy  +1
(X) 00
```

SZXACXPXCY
0 1 0 1 0 1 0 X
(X → previous status)

INR M - 1B, 3, 10T

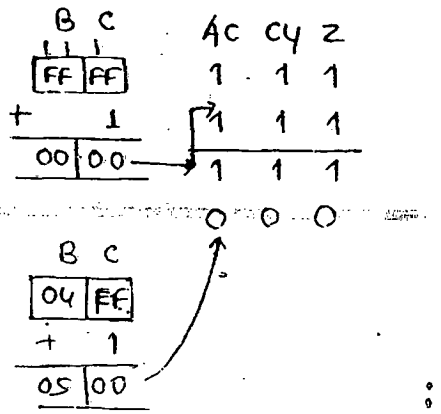
Eg:- A000H : INR M



INX Rp - 1B, 1, 6T

Eg:- * No flags are affect

INX B



Decrement →

R/m / Rp - 1

* CY → not affected.
* S,Z,AC,P → GRR's
* Result → Operand used.
* Internal subtraction.

DCR R - 1B, 1, 4T

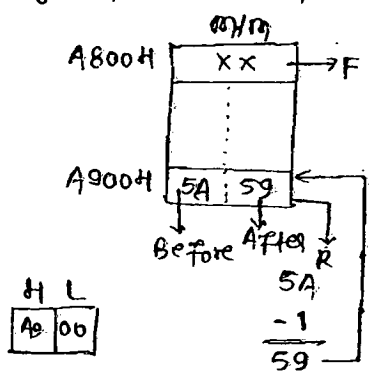
Eg:- DCR D

```
 D      D
00     00
      - 1
      FF
```

SZXACXPXCY
1 0 0 0 0 1 0 X
(X → previous state)

DCR M - 1B, 3, 10T

Eg:- A800: DCR M



Ex: SZXACXPXCY
0 0 0 1 0 1 0 X

DCX Rp - 1B, 1, 6T

Eg:- No. flags are affect

```
DCX. H
 H  L
40 00
   -1
3FFF
```

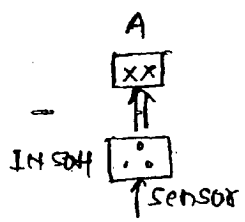
(3.) Logical Instructions →

(1) AND →

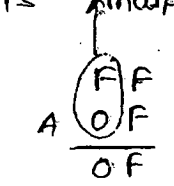
* Used to mask the bit.
* AC=1, CY=0

AND Gate

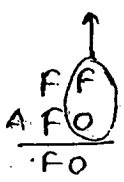
A	B	O/P
0	0	0
0	1	0
1	0	0
1	1	1



Higher Nibble is masked

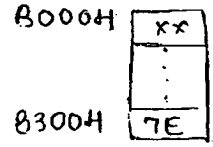


Lower nibble is masked.



ANA R - 1B, 1, 4T

Eg:- B000H: ANA M




```
  9A
  7E
------
1001 1010
0111 1110
------
0001 1010
  1   AH
```

S Z X AC X P X CY
0 0 0 1 0 0 0 0
1 0H

ANA R - 1B, 2, 7T

ANA D

A D
66 99

```
  A   66  0110
A D   99  A1001
---   --  -----
      00  0000
```

S Z X AC X P X CY
0 1 0 1 0 1 0 0
5 4H

ANI - 8 bit data 2B, 2, 7

Eg:- ANI BDH

B
48

```
  48
A BD
------
0100 1000
1011 1101
------
0000 1000
  0   8
```

S Z X AC X P X CY
0 0 0 1 0 0 0 0
10H

(2) OR →

* Used to set the bits
* AC=CY=0

OR Gate

A	B	O/P
0	0	0
0	1	1
1	0	1
1	1	1

ORA R-1B, 1, 4T

Eg:- ORA L

A (33) L (CC)

A	33
0 L	0 CC
0011	0011
1100	1100
1111	1111
F	F

SZXACXPXCY
1000 0100
8 4H.

ORA M-1B, 2, 7T

Eg:- B500H: ORA M

B500H [XX]
B800H [4B]

H L
B8 00

A
13
4B

0001 0011
0100 1011
0101 1011
5 BH

SZXACXPXCY
0000 0100
0 4H

ORI 8 bits data-2B, 2,

Eg:- ORI FFH

86
FF
FF

SZXACXPXCY
1000 0100
8 4H

(3.) EX-OR →

* Used to set/reset the bits
* AC = CY = 0

EX-OR Gate

A	B	O/P
0	0	0
0	1	1
1	0	1
1	1	0

XRA R-1B, 1, 4T

Eg:- XRA A

A	00H
XX	1
XX	
00	FF H

SZXACXPXCY
0100 0100

30
-30 AC=0
00

55
55 AC=1
00

XRA M-1B, 2, 7T

Eg:- C000H: XRA M

C000H [XX]
C400H [E7]

H L
C4 00

A
88

88 1000 1000
⊕ E7 1110 0111
0110 1111
6 FH

SZXACXPXCY
0000 0100
04H

XRI 8 bit data-2B, 2,

Eg:- XRI F0H

A (32)

F0
⊕ 32

1111 0000
0011 0010
1100 0010
C 2H

SZXACXPXCY
1000 0000
8 0H

	F	Result
AND	0	0
	x	x
OR	0	F
	x	F
EX-OR	x	1's comp of 'x'

Que → XRI of H

(a.) Reset's lower Nibble
(b.) Reset's highest Nibble
(c.) Compliments higher nibble
(d.) Compliments lower nibble

Soln →

$$\begin{array}{rr} 28 & 1000 \\ \oplus\ 0F & \oplus\ 1111 \\ \hline 27 & 0111 \end{array}$$

(4.) Compare →

Compare the content of

A compare → R / m / 8 bit data

* Accumulator → Unchanged.
* Internaly instruction.
* Result → status of cy & z
* S, AC, P → Result of subtraction.

If A < R/m/8bit data then cy=1; z=0

A = ——11—— : z=1, cy=0

A > ——11—— ; cy=z=0;

CMP R-1B, 1, 4T

Ex:- cmp B

A: 96 B: 87

A<B: cy=1; z=0

$$\begin{array}{rr} & 36H \\ & -87H \\ \hline & AF \end{array}$$

cy = 1 → Because Borrow

CMP m-1, B, 2, 7T

Eg:- C700H : cmp m

C700H: XX
C900H: 44

H L: C9 00

A: 44

A=m; z=1, cy=0

CPI 8bit data-2B, 2, 7

Eg:- CPI 17H

A: 8D

A > data; cy=z=0

$$\begin{array}{rr} & 8DH \\ & -17H \\ \hline & 76 \end{array}$$

S Z X AC X P X CY
1000 0101
85H

44
-44
00

S Z X AC X P X CY
0101 0100
54H

(5.) Compliment →

* CMA – 1B, 1, 4T

Compliment content of accumulator.

* No. of flags are affected.

* Internally 1's compliment.

Eg:-

A
11

CMA
0001 0001
1110 1110
E E

CMC – 1B, 1, 4T

Compliment carry flag if CY

0 → 1
1 → 0

STC – 1B, 1, 4T

Set carry flag. if CY

0 → 1
1 → 1

(6.) Rotate →

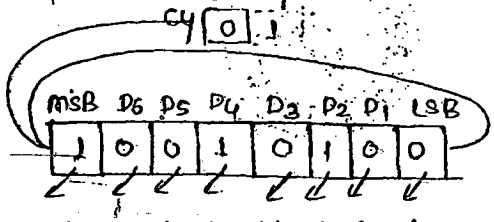
* Only possible with Accumulator content.
* Only CY → affected
* S, Z, AC, P → not affected.

RLC
RAL
RRC
RAR
} 1B, 1, 4T

RLC → Rotate the content of accu. 1 bit left w/o carry.
RAL → 1 bit left with carry
RRC → 1 bit right w/o carry
RAR → 1 bit right with carry

(1.) RLC →

If A = 94H ; CY = 0

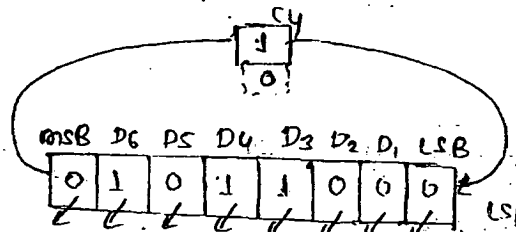


0 0 1 0 1 0 0 1 LSB → MSB

2 9

(2.) RAL →

If A = 58H , CY = 1



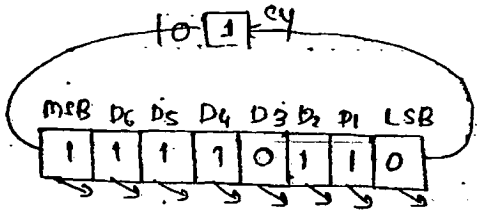
LSB ← CY

1 0 1 1 0 0 0 1

B 1 H

(3.) RRC →

If A = F6H ; CY = 1



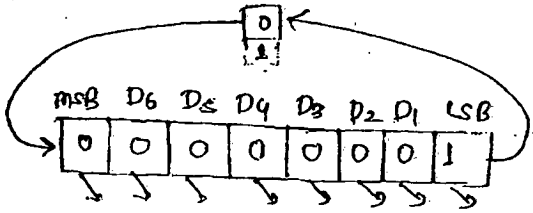
LSB → MSB

0 1 1 1 1 0 1 1

7 B

(4.) RAR →

If A = 01H ; CY = 0



0 0 0 0 0 0 0 0

0 0

Que:→ After execution → _ _ _ _ _ _

	CY	Z
①	1	1
②	1	0
③	0	1
④	None	

Ans → None

(4.) Branching instruction → *In this group the control of prog. is Xfered from one loc^n to another. conditionaly (or) unconditionaly.

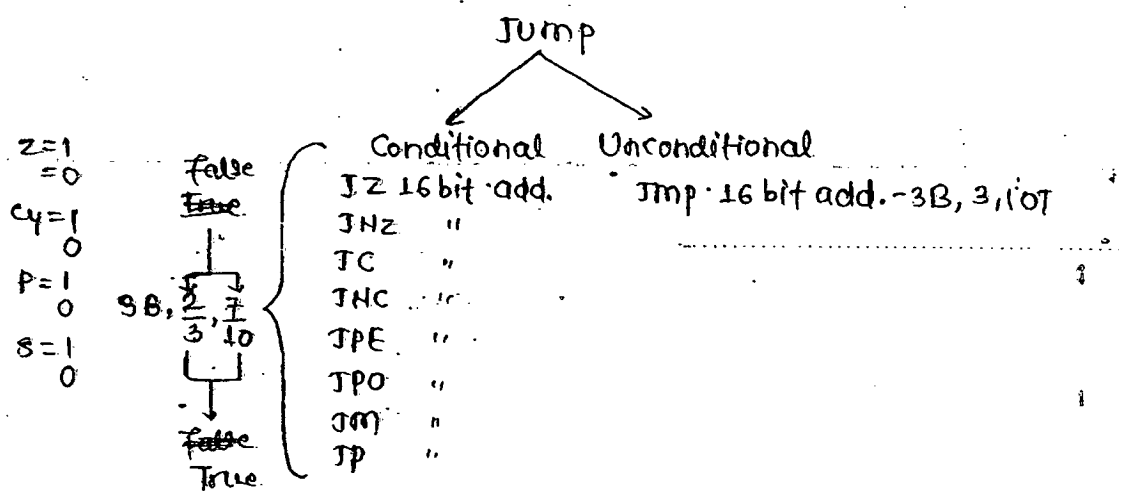
① Conditional Instruction → They depand on status of flag affected for previous ALU operation except ALE operation.

Condition true → control of progr is transfer to 16 bit add. & execution continues.

Condition false → Control of prog. is transfer to very next ins^n

② Unconditional Instructions → Control of prog is transfer to 16 bit add. Unconditionaly.

Eg:- JUMP, CALL, RETURN.



Purpose of Jump insⁿ →

This are used for transfer the control of prog. from 1 loch to another conditionally (or) unconditionally.

Conditional insⁿ →

Note → One m/c cycle is wasted when the condⁿ is false.

JNZ 16 bit add - 3B, 2/3, 7/10 →

* Jump if no zero 16 bit add.

Eg:-

```
        4000H : MVI C , 02H
                 XX
Loop1 : 4002H : DCR C
                 YY
        4003H : JNZ : loop1 ; Z=0
                 ZZ
        4006H : HLT
                 **
```

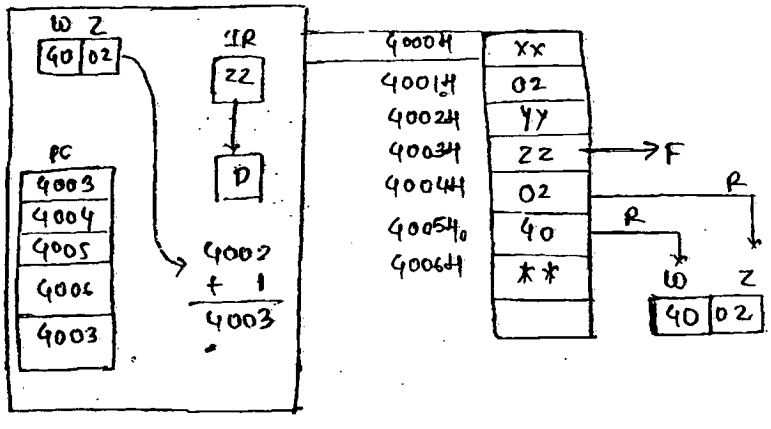
C
02

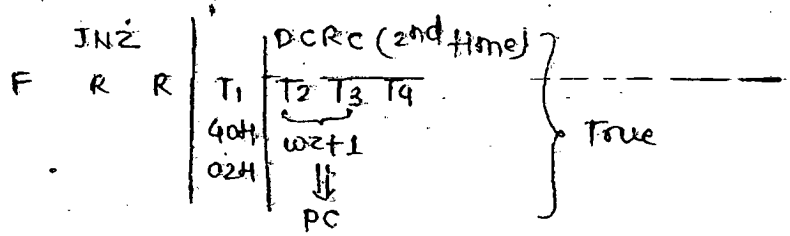
(1.) 01 : Z=0 → True
(2) 00 : Z=1 → False.

* In the above eg. the insⁿ between lebals are known as loop insⁿ,
* The controlling reg. for the loop is reg. C.
* If C has n, the loop executes for n no. of times. where condition is true for n-1 times & false only once.

Que → when condⁿ is true control of the prog. is xfer to (or) the content of pc is 4002H

Que → when condⁿ is false the control of prog. is transfer to (or) the content of pc is 4006H





* The operation for remaining conditional JUMP insⁿ is similar to JNZ except that flags are different.
* All the JUMP insⁿ are immediate Am.

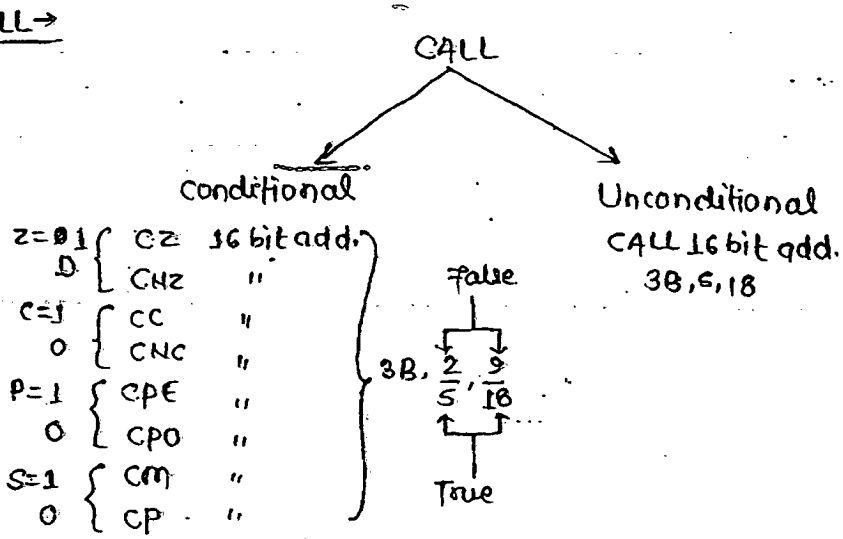
Unconditional JMP 16 bit add.→ 3B, 3 m/c cycle, 10T

Control of the prog is transfer to 16 bit add. unconditionaly.

F+R+R
4+3+3 = 10T

Eg:-
```
LXI B, 9009H
MOV A,C
JMP Quit
XORI FFH
ANI FFH
QUIT: HLT
```

CALL→



* This insⁿ are used to CALL a subroutine within a main prog.

Subroutine→ Set (or) group of insⁿ which perform specific operation can be written as a seprate prog. away from main prog is known as Subroutine..

* It may be repeatatelly used within a main prog.

Unconditional CALL → CALL is 16 bit add: 3B, 5, 1B

Immediate (or) Indirect Am

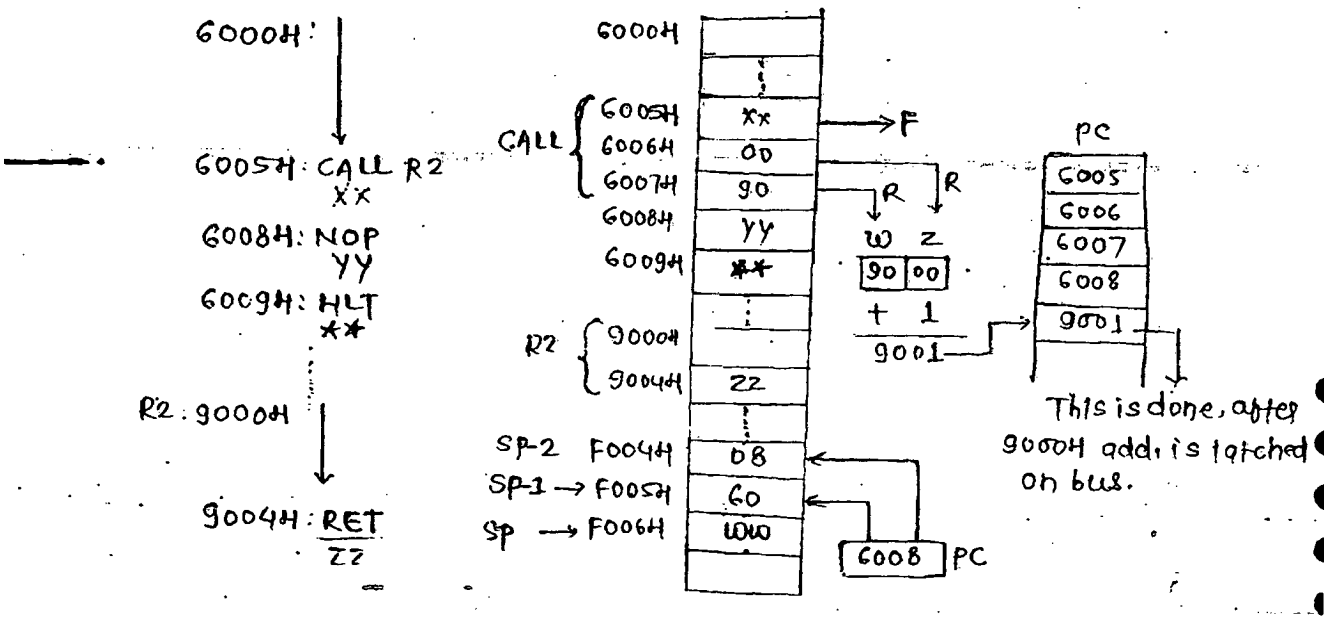
* When CALL is executed:-

(1) The content of pc (or) add. of the insn next to CALL is pushed onto stack m/m.

Therefore SP → SP-2.

(2) The control of prog. is Xfer to subroutine add. & execution continues.

main prog:-



F R W W W = 18T

*In the above eg. when CALL is executed :-

(1) The control of prog. is transfered to (or) content of pc 9000H.

(2.) The value of SP's is SP-2 (FOO4H).

(3) Data present at top of the stack is lower byte of add. of the insn next to CALL.

Eg:- LXI SP, D390H

5008: CALL R3

R3 : 8000H

(1.) The control of prog. is goes to 8000H after CALL execution.

(2.) The value of sp-2 is D390H → SP
D38FH → SP-1
D38EH → SP-2

(3.) Data present at the top of add. be 5008H
5009H
500AH
500BH → OB

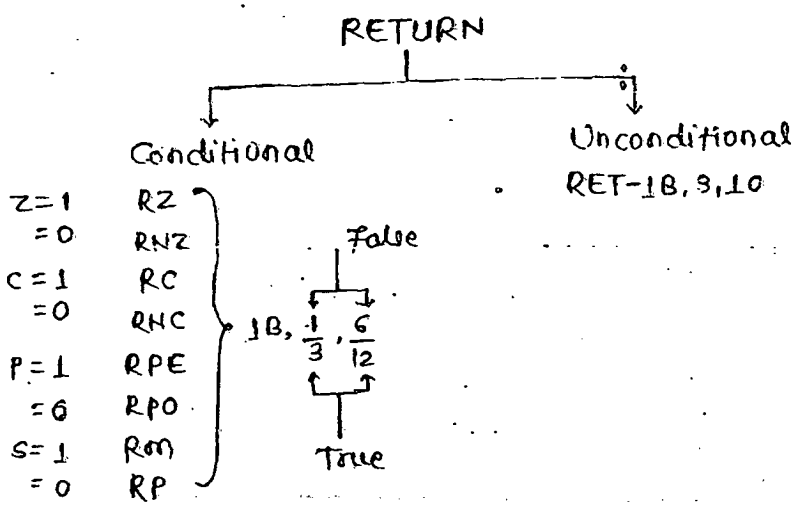
Conditional CALL →

(a) True condⁿ → The operation is similar to unconditional CALL i.e. 5m/c (or) 18T are required.

(b) False condⁿ → Two m/c i.e. 9T are required.

Note → SP is unchanged when the condⁿ is false.

RETURN →



* This are used as last insⁿ of a subroutine (or) ISR such that the control of prog. may RETURN to main prog.

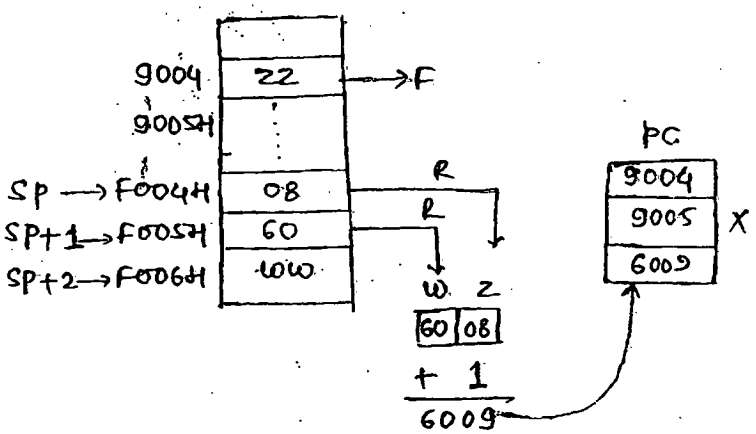
Unconditional RETURN → 1B, 3, 10

* When RET is executed :-

(i) The data present at top of the stack (2 bytes) pointed by sp is loaded into pc.

Therefore SP → SP+2

(2.) Control of the prog. is transfer to 16 bit add. & execution continues



Conditional RETURN →

Condition true → Operation is similar to unconditional return, except that fetch is of 6T.

Condition false → Only one m/c cycle i.e. 6T states are req.
* Fetch is 6T states either for true (or) false.

RSTn → 1B, 3, 12T

n → 0-7

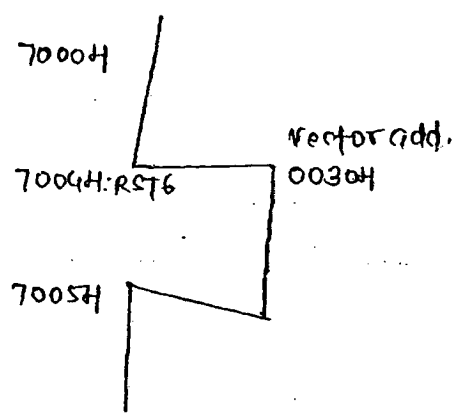
n × 8 = (x)10 = (y)16

RST 0 - 0000H

RST 1 - 0008H

RST 6 - 0030H

RST 7 - 0038H



* These are 8 software interrupts in 8085 which can be used either as instn (or) along with INTR.

** It is also known as one-byte unconditional CALL.

F + W + W
6 + 3 + 3 = 12T

When RST-n is executed →

(1.) The content of PC (or) add. of ins^n next to RSTn is pushed on to stack.

∴ SP → SP-2

(2.) Control of prog. is xfer to vector add. of RST-n calculated from.

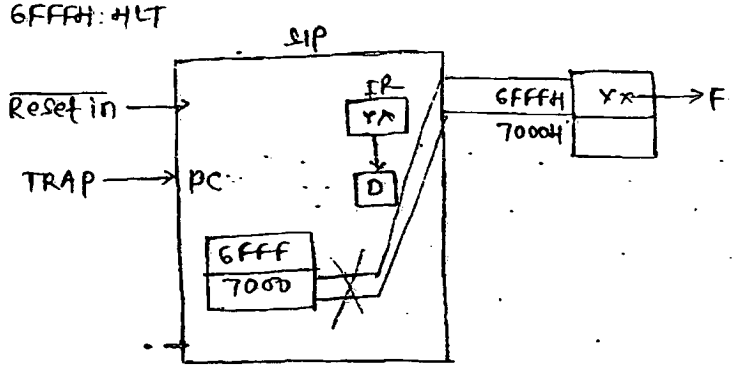
$$n \times 8 = (X)_{10} = (Y)_{16}$$

* When RST-n is used along with INTR, $\overline{INTA}$ is activated instead of $\overline{RD}$.
* RST-n is used as break-point interrupt for software ins^n.

(5.) Machine control ins^n →

* **HLT → 1B, 2, 5T**
* It is used to stop the execution of prog.
* The processes enters into HLT acknowledgement cycle more than one wait state included for every clock period.
* Internaly prog. counter is disconnected from the add. bus as there is no provision for next Opcode fetch the prog stops.
* A reset (or) hardware interrupt is req. to come out of HLT state.

Eg:- 6FFFH: HLT



F + Bus idle
4 + 1 = 5

* **NOP - 1B, 1, 4T** → No operation, is performed by the ins^n but a delay of 4T state is included in execution time used for writing delay prog. & also when the processes communicates with slow speed I/o devices.

Eg:-
```
IN 80H
NOP
OUT 40H
```

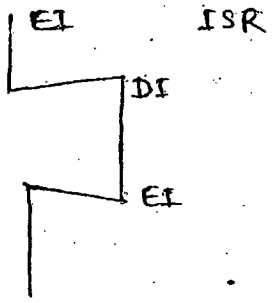
* **Disable Interrupts** → 1B, 1, 4T

Used to disable maskable interrupts (INTR, RST5.5, RST6.5, RST7.5)

* **Enable interrupts** → 1B, 1, 4T

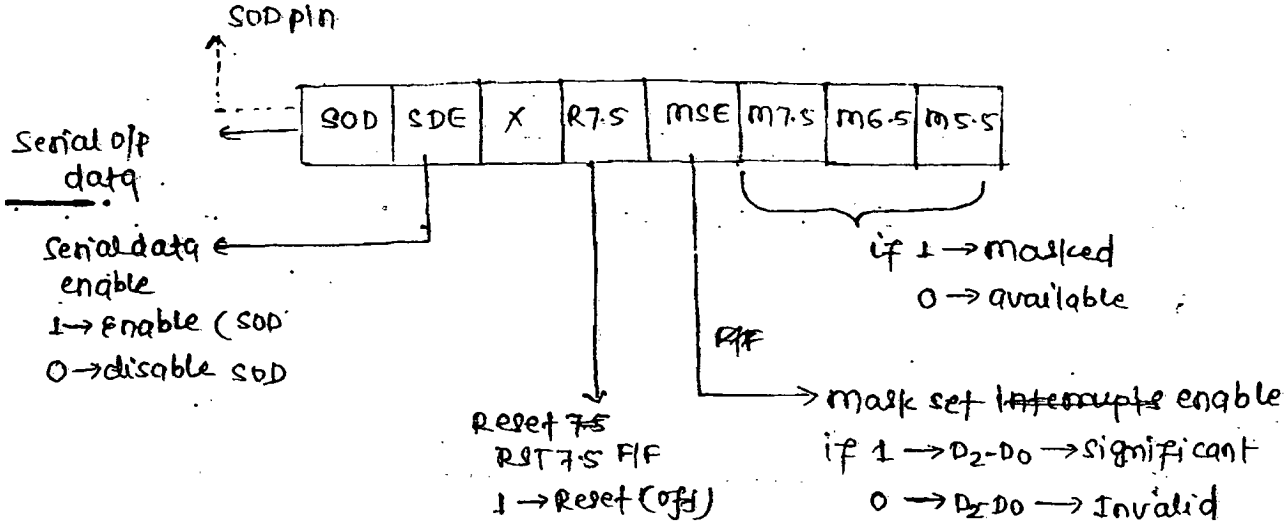
Used to enable maskable interrupts & internal interrupt enable F/F is set (hi..

* It is used at initial insⁿ of a main prog. & also at the last insth of an ISR. such that the processer may be ready to serve another interrupt.



* SIM – set interrupt mask →

* It is a multipurpose insⁿ used to masked the interrupts (or) make them available.
* It is also used to Xfer serial data through SOD pin. valid only for RST 7.5, RST 6.5, RST 5.5. used along with content of accumulator.



* D_3 → It is the control bit over D_0–D_2 –
  – To make them significant (or) invalid.
* D_0–D_2 indicate interrupt masked (or) available.
* D_4 is an extra provision for RST 7.5 to reset it.
* D_6 is the control over D_7.
* D_7 is the serial data to be Xmitted through SOD pin.

Que.→ After the execution of following insⁿ:-

(a) EI
MVIA, C8H
SIM

(b) EI
MVIA, 4EH
SIM

find (i) Interrupt masked.
(ii) Interrupts available.
(iii) serial data transmitted.

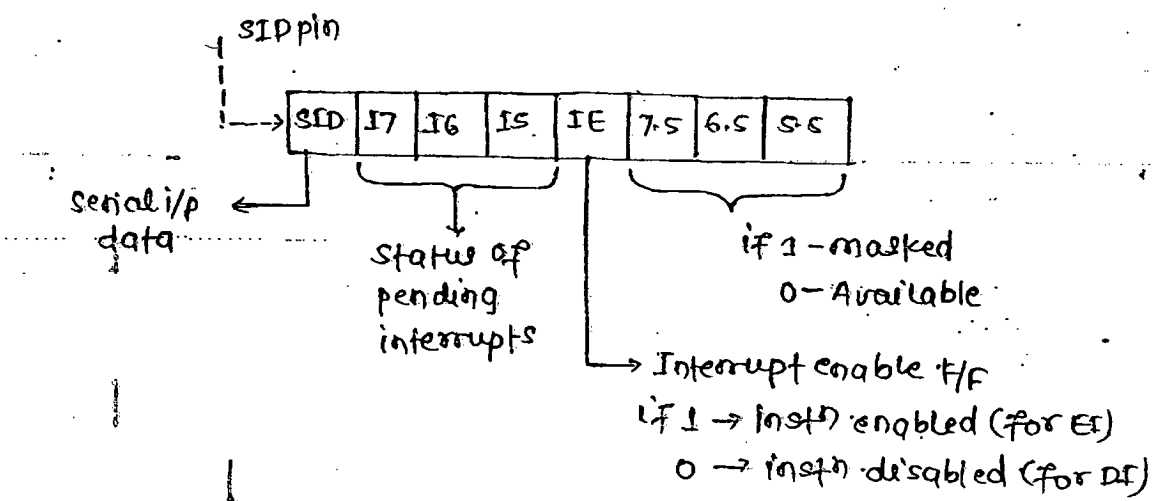
Solⁿ (a) (i) Interrupt masked - RST 5.5
(ii) Interrupt available - RST 7.5 & RST 6.5
(ii) serial data transmitted - 1

(b) 4E H → 1st see this
0100 1110

(i) M7.5, M6.5
(ii) RST 5.5
(iii) 0

*RIM → (Read interrupt mask) → 1B, 1, 4T

* Multipurpose insⁿ used to know the status of pending interrupts & also to receive serial data through SID pin. valid only for RST 7.5, RST 6.5 & RST 5.5.
* The status of interrupts & serial data is loaded into accu. after execution of RIM.



D_3 indicates interrupt enabled (or) disabled.

D_0-D_2 indicate interrupts masked ~~are~~ (or) available.

D_4-D_6 indicate pending status of interrupts.

D_7 is the serial data received through SID pin.

Que.→ The content of accumulator after execution of RIM is 9CH

Find ① Interrupt masked
② —"— available
③ —"— pending
④ serial data received.

Sol^n→

9C- 1001 1100

① Masked RST7.5
② available 6.5 & 5.5
③ I5 is pending. (5.5)
④ 1.

38→ 0011 1000

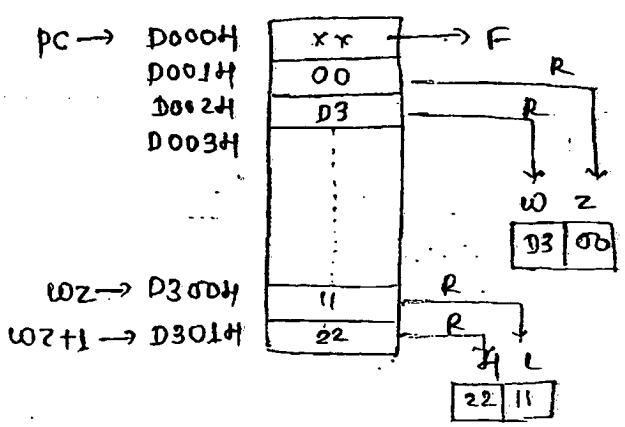
① ~~7.5 6.5 5.5~~ None
② All
③ 6.5 & 5.5
④ 0

Special Instruction →

(1.) LHLD 16 bit add. - 3B, 5, 16T

Load HL pair direct with the data present at 16 bit add. (2 bytes)

Eg:- D000H: LHLD D300H



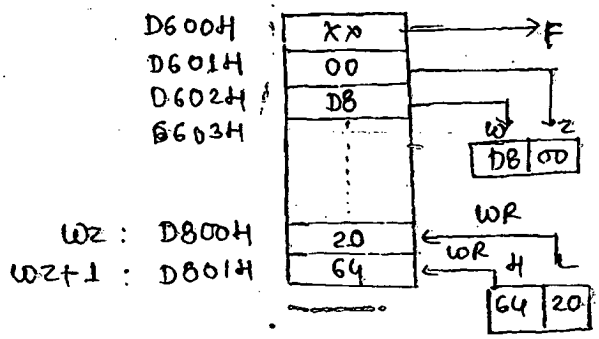
RD | WR | ALE :
5 0 5

FRRR
9333 → 16

(2.) SHLD 16 bit add. → 3B, 5, 16T

Store the content of HL pair at 16 bit add. i.e. 2 loc^{n}.

Eg:- D600H : SHLD D800H : if [HL] → 6420H



RD | WR | ALE

3 2 5

direct Am

(3.) PCHL → 1B, 1, 6

Load PC with HL content.

* It is also known as one byte unconditional jump.
* It is ~~like~~ branching insn & reg. Am [HL → pc]

Eg:- 2000H : LXI H, 2006H

2003H : PCHL

2004H : MVI C, 80H

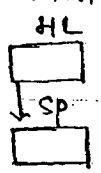
2006H : HLT

C

[?] → undefined

(4.) SPHL → 1B, 1, 6T

~~Store pc with H~~ Copy the content of HL pair to stack pointer.



Reg. Am

Group - data Xfer

Eg:- LXI SP, 8900H

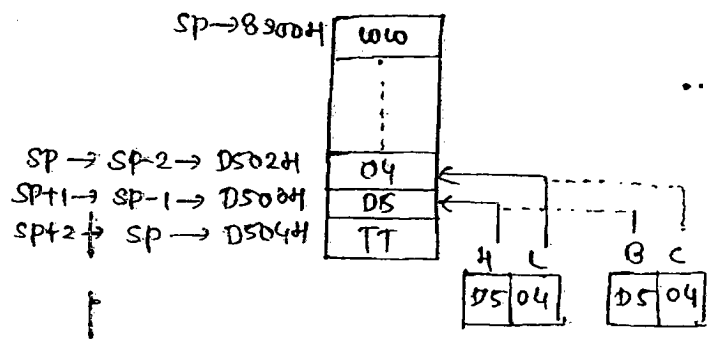
LXI H, D504H

SPHL

PUSH H

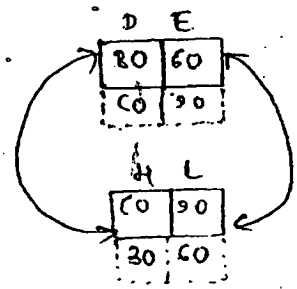
POP B

HLT



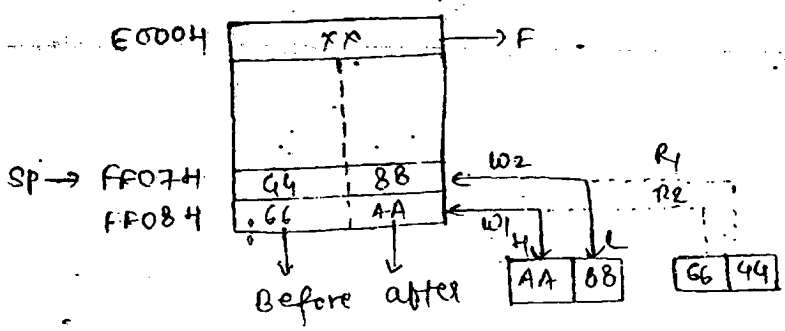
⑤ XCHG → 1B, 1, 4T

* Exchange the content of DE & HL reg. pairs.



⑥ XTHL → 1B, 5, 16T

* Exchange the data present at top of the stack with HL contents. (2 bytes)



⑦ DAD Rp → 1B, 3, 10T

* Add the content of reg. pair to HL pair.

DAD B
DAD D
DAD H → multiply x 2
DAD SP

* Accumulator → Unchanged.
* Result (HL).
* Only cy is affected. if there is a carry out of 16 bits
* S, Z, AC, P → not changed.

Eg:- F000H : DAD B

BC
FF 96

H L
34 56
23 EC

cy
1

m/c → F+B+B
4 3 3 → 10T

⑧ DAA → 1B, 1, 4T

63

decimal adjust accumulator (or) decimal adjust accumulator after addⁿ.

Q→ Insⁿ which req. the status of AC for execution?

① ADCB
② SBBH
③ ANAD: AC=1

ⓐ 1 only ⓑ 1 & 2 ⓒ 1,2,3 ⓓ None

Ans → None

* This is the only insⁿ which works with the status of AC in BCD conversions.
* It is used after addⁿ
* It converts 8 bit value present in accu. into 2, 4 bit BCD no.
* It verifies the content of accu. the following condⁿ are used for conversion:-

(1.) If D_3-D_0 of lower nibble of 'A' > 9 / AC=1, adds 6 to it.

(2.) If D_7-D_4 of higher nibble of 'A' > 9 / CY=1, adds 6 to it.

BCD No's →

Binary coded decimal

0 - 0000
1
2
⋮
9 1001

A
↓
F
} Invalid

A → $(96)_{BCD}$
B → $(47)_{BCD}$
CY ①
DD

ADDB
DAA

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	1	0	1	1	1	0	1
0	1	1	0	0	1	1	0
0	1	0	0	0	0	1	1
4				3H			

1 → AC
A → $(89)_{BCD}$
B → $(77)_{BCD}$
CY ①
00

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	0	0	0	0	0
0	1	1	0	0	1	1	0
0	1	1	0	0	1	1	0
6				6			

Q→ Write an assembly language prog. to perform following operation. After execution find:-

(1.) A (b) F (3). PSW (4) length of prog. (5) No. of m/c cycle

(6) Total execution time if f_{clk} = 5MHz

(7) m/m representation of prog. if it starts at 5000H.

[A] ← 79H

[B] ← 25H

[A-B] → [C]

stop.

Sol^n →

5000H	MVI A, 79H	- 2B, 2, 7T FR	79
5002H	MVI B, 25H	- 2B, 2, 7T FR	-25
5004H	SUB B.	- 1B, 1, 4T F	54
5005H	MOV C, A	- 1B, 1, 4T F	S Z X AC X P X CY
5006H	HLT	- 1B, 2, 5T FB	00010000
			10H

0101 0100

(1) 54 (2) 10H (3) PSW 5410 (4) 7B (5) 8

(6) Total execution time:-

Clock period x No. of T-states x count value

$= \frac{1}{f_{clk}} \times$ T-states x CV

$$= \frac{1}{5\times10^6} \times 27 \times 1 = 5.4\,\mu s$$

65

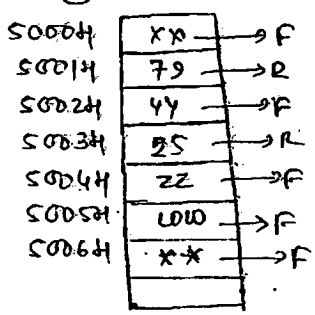
(8) RD | WR | ALE

7 0 7

(9) After execution ~~if A~~ [A] ∧ [5003H]

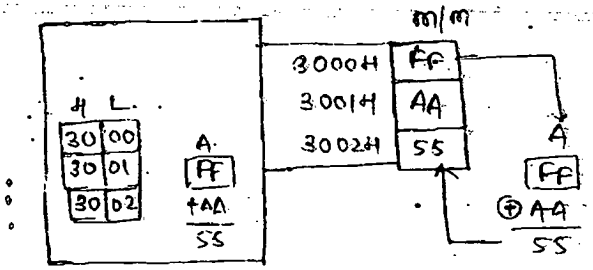
```
  54
∧ 95
----
  04
```

(7) m/m representation



Q⇒ Write an ALP to perform X-OR operation between 1st, 2 m/m loc^n contents & store the result in the next loc^n.

Soln→

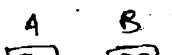


① LXI H, 3000H
MOV A, M
INX H
XRA M
STA 3002H
(or)
INX H
MOV M, A
HLT

Find the result

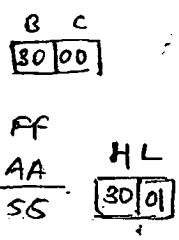
② LDA 3000H
MOV B, A
③ LDA 3001H
XRA B

(II)



③ LXI B, 3000H
LDAX B → A
① LXI H, 3001H
XRA M

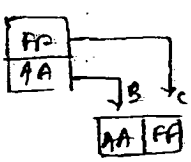
(II)



④ LXI SP, 3000H
① POP B
MOV A, C
XRA B

(II)

SP → 3000H
SP+1 → 3001H
SP+2 → 3002H



Que.→ Write an ALP to access 2-bytes of data i.e. from port 40H & 60H. Add them, compliment result, rotate it right for 5 times; transfer resultant value to port 80H.

Ans.→

```
IN 40H
MOV B, A
IN 60H
ADD B
CMA
RRC
RRC
RRC
RRC
RRC
OUT 80H
HLT
```

```
      MVI C, 05H
L1:   RRC
      DCR C
      JNZ L1 : Z=0
```

C: 05

(1) 04 ; Z=0
(2) 03
(3) 02
(4) 01
(5) 00 ; Z=1

Que.→

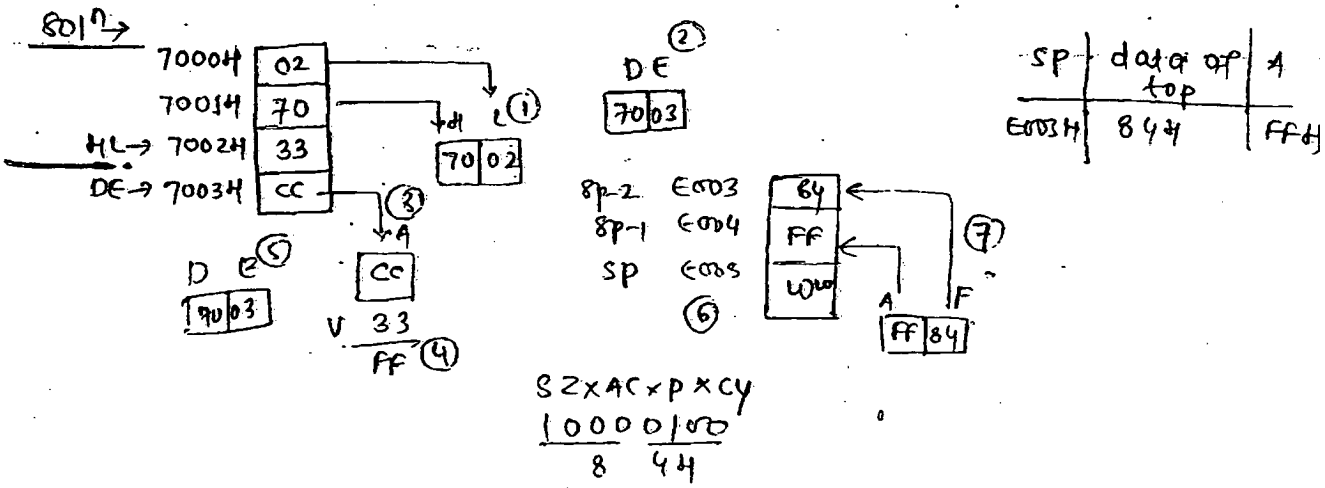
```
LHLD 7000H
LXI D, 7003H
LDAX D
ORA M
DAD D
SPHL
PUSH PSW
HLT
```

7000H	02
7001H	70
7002H	33
7003H	CC

SP → ? data @ top of stack = ?

(A) → ?

Soln →



Que→ Write an ALP to find no. of even & odd no. from 10 bits of data starting at A000H.

Store the count of even no. in register E, odd no. in reg. D.

Soln→

```
        LXI H, A000H
        LXI D, 0000H
        MVI C, 0AH
Rept:   MOV A, M
        RRC/RLC
        JC odd  cy=1
        INR E
        JMP NEXT
odd :   INR D
NEXT:   INX H
        DCR C
        JNZ Rept
        HLT
```

HL	D E	C
A0 00	00 00	09 11
A001	0101	09
		00, Z=1

Timer & Counter problems→

$$\text{Total execution time} = \frac{1}{f_{clk}} \times \text{T-states} \times \text{count value}.$$

* There are no timer/counters in 8085 but programming logic can be implemented to write time delays & count the events.

Q→ Find the total execution time for the following prog. if operating freq=3M

```
        LXI D, 0FF0H   - 10
        MOV A, E       - 4
        ANA D          - 4
        JP skip        - 7/10
        ORI FFH        - 7
        ADI 00H        - 7
skip:   HLT            - 5
```

Soln→

D	E
0F	F0

↓A

F0
∧ 0F
60

S Z X AC X P X CY
0 1 0 1 0 1 0 0

$$T_{ET} = \frac{1}{3\times10^6} \times 33 \times 1$$

$$= 11\mu s$$

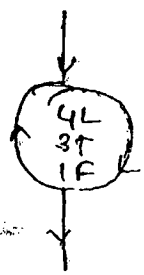
Q.→ TET=?, f_{CLK} = 2MHz

```
    MVI C, 04H   - 7
L1: NOP          - 4
    DCR C        - 4
    JNZ L1       - 7/10
    HLT          - 5
```

Sol→

$TET = T_{OL} + T_{WL}$

$T_{OL} = \frac{1}{2\times10^6} \times 12 \times 1 = 6\mu s$



$T_{WL} = T_{CS} + T_{CNS}$

↓ True ↓ false

$= \frac{1}{2\times10^6}[18\times3 + 15\times1]$

(18×3 → 4)

$= \frac{1}{2\times10^6}[64] = 34.5\mu s$

*Count Related problem →

Q.→

```
    MVI B, XX     7
L2: LDA 7000H     13
    DCR B         4
    JNZ L2        7/10
    STA 8000H     13
    HLT           5
```

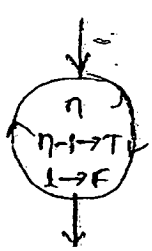
TET = 400μs, f_{CLK} = 3MHz

T_{OL} = ?, $T_{WL} = TET - T_{OL}$

$T_{WL} = T_{CS}$

Sol→

$T_{OL} = \frac{1}{3\times10^6} \times 25 \times 1 = 8.33\mu s$



$T_{WL} = TET - T_{OL} = 400\mu s - 8.33\mu s = 391.67\mu s$

$T_{WL} = T_{CS} + T_{CNS}$

↓ True ↓ false

$391.67 = \frac{1}{3\times10^6}[27(n-1) + 24\times1]$

$n = 43.66 \approx (44)_{10} \rightarrow (2C)_H$

Que.→ MVI B, xx — 7 T1

& L2: MVI C, FFH — 7 x n TET = 100ms ; f_{CLK} = 2MHz

L1: DCR C — 4 x 255 x n

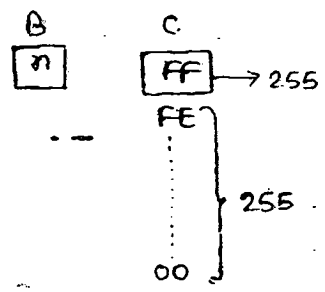
JNZ L1 — 7/10 = [10 x 254 + 7 x 1] n

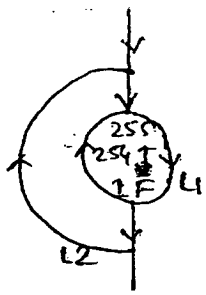
DCR B — 4 x n

JNZ L2 — 7/10 = [10(n-1) + 7 x 1]

HLT — 5 x 1

Sol→





$TET = \frac{1}{f_{CLK}} \times$ NO. of T-states x CV

$100 \times 10^{-3} = \frac{1}{2 \times 10^{6}} [\quad]$

$n = 55.6$

$\approx (56)_{10}$ → convert in to

$= 38H$ ← Hexadecimal value.

Que→ MVI C, 00H

L3: DCR C → ?

JNZ L3

HLT

(a) 16 (b) 255 (c) 256 (d) None

Sol→

C

00 — 1

FF — +1

FE … 00 — 255 ⇓ 256

Q.→ MVI B, FFH

MVI C, FFH

L1: DCR C

JNZ L1

DCR B

JNZ L1

HLT

TET = 100ms

f_{CLK} = 2MHz

Q.→ If Z = 0

LXI B, 0003H

L1: DCR B

JNZ L1

HLT

How many times loop executes

(a) 0 (b) 1 (c) 3 (d) None.

Q:→ XRA A
LXI B, 0003H
L3: DCR B
JNZ L3
HLT
How many times loop executed.
(a) 1 (b) 2 (c) 3 (d) ∞

Ans key →

Exercise (1) →

① (A) ② (D) ③ (D) ④ (D) ⑤ (C) ⑥ (d) ⑦ (D) ⑧ (D) 9(a) 10(d) 11(B) One only
⑫ (b) 13(b) 14(d) 15(a) 16(b) 17(b) 18(a) 19(a) 20(a) 21(b) 22(c)
23(c) 24(b) & (d) 25(b) & base address (operand) 26(c) 27(a) ~~28(~~

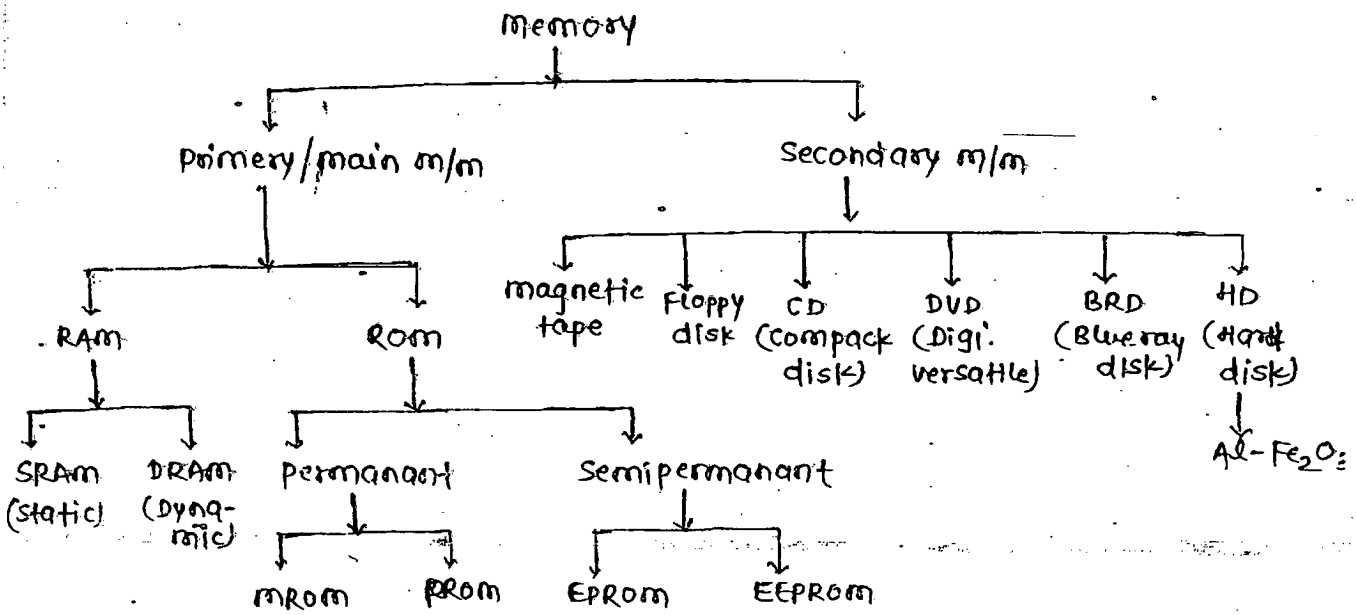
Exercise (2)

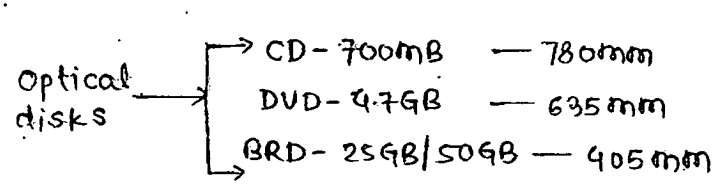
1(c) mod - Remindes 2(a) 3(c) 4(b) 5(c) 6(b) 7(b) 8(a) 9(b)
div - Quetient 10→ 26.5 ~~sec~~ 11(d) 12(d) 13(a) 14(d) 15(d)
x = 3+9+5+4 (a) ↓ ↓
= 21 2050 07

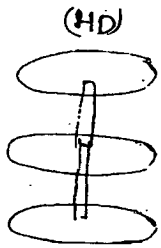
16(d) 17(c) 18(b) 19(d) 20(c) 21(c) 22(d) 23(a).

(10) Interfacing

Clasification of m/m →







* Writing data into CD/DVD is known as burning.
* In optical disk light is used to write/read the data.
* 1 stored as slot, 0 is stored as blank.
* Both RAM & ROM are random is accessing data from m/m i.e. the time taken to access any of the m/m loch is same.

* RAM → (Random access m/m)
  * It is also known as read/write (or) temporry (or) volatile m/m.
  * Data is present only with supply of power.

(1) Static RAM (SRAM) → F/F are used to stored the data.
Bit is stored as voltage.
  * Each m/m cell may have 3-6 transistors.

* Less density (or) storage capacity.
* faster than DRAM. i.e. low access time. & costlier.

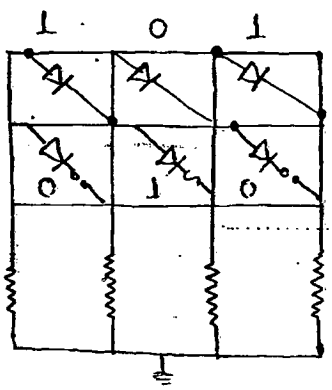
(2) DRAM (dynamic RAM) → * Data is stored as charge b/n gate to seprate substrate of a mos transistor.

* As the charge leaks of frequently DRAM must be refreshed frequently
* Each m/m sell contains one transistor.
* more density (or) storage capacity.
* slow in operation.
* It is cheaper.

* ROM → (Read only m/m)
* It is also known as non-volatile m/m
* The 1st type of m/m is mROM – masked Rom.

(1) mROM → * masking & metalication tech is used is design of m/m.
* m/m arrays contain diodes (or) transistor presence of a indicates one. absence 0.
* Once manufactored can't be modified.

Eg:- multiplication table.



(2) PROM → Programmable ROM → one time programmable ROM.
* m/m arrays contain diodes / transistor ~~at micro~~ nicrome (or) poly-silicon fuses.

fuse → 1

fuse blown off → 0

PROM, PAL, PLA → PLD's (programmable logic devices)

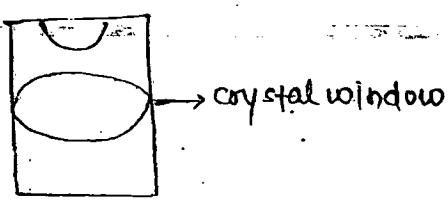
PLA → programmable logic array

PAL → programmable array logic.

Gates	PROM	PAL	PLA
OR	Progra.	fixed	progra.
AND	fixed	progra	progra.

*EPROM → (Erasable PROM)

* m/m chip contains a crystal window on it when exposed to UV light data is erased.



* It requires 20 minutes to erase data (disad.)
* Prefered for mass storage.

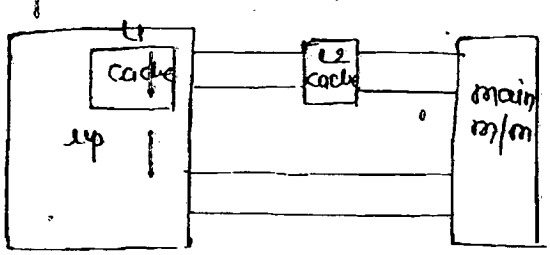
*E^2PROM → (Electrically EPROM)

* votage is used to erase the data.
* Data can be earase but at reg. level & also completely at a time.
* It requires 10ms to write a data bytes.

*Flash m/m → similar to E^2PROM except that data can be erased at block (or) sector level (or) completely at a time.

* Require 10ms write a data.

*Cache m/m → Used to store frequently used data & instructions.



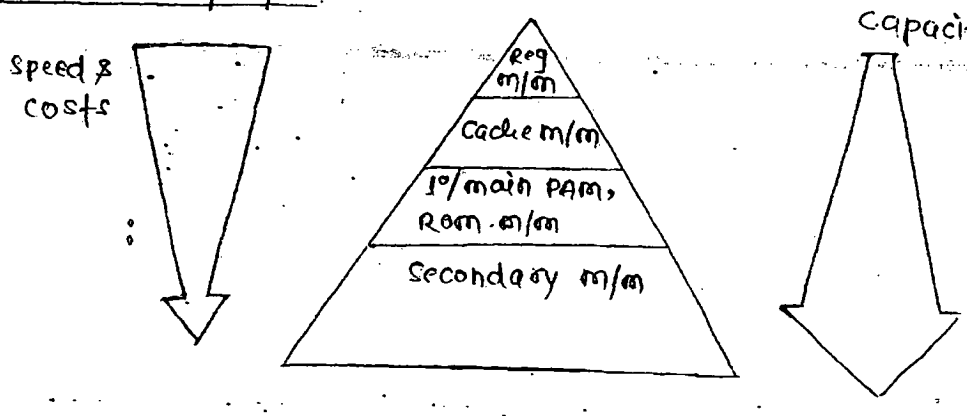
* It is used b/n up & main m/m.

* Register m/m → *It is the m/m due to internal reg. within the processor.
*more the no. of GPR (or) scratch pad reg. more is the speed of up. & ~~caused too~~ cost too.

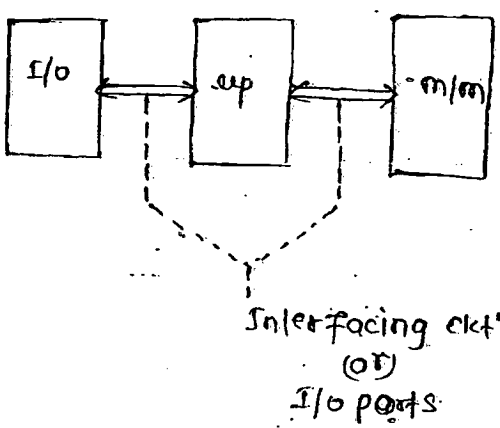
* Bootstrap m/m → *It is used to store prog. related to booting process.
*The process of loading operating sys. software from 2° m/m to 1° m/m is known as booting.

~~* m/m there~~

* m/m Heirarychy →



* Interfacing :-



*designing logic ckt's & writing programms to make the processor communicate either with m/m (or) I/o devices is known as interfacing.
*The logic ckt used are known as interfacing ckt's (or) I/o ports.

Basic Interfacing components:-

(1) Tristate Buffer → * It is used to increase current (or) power handeling capability of a bus when more no. of I/O devices are connected to common bus.

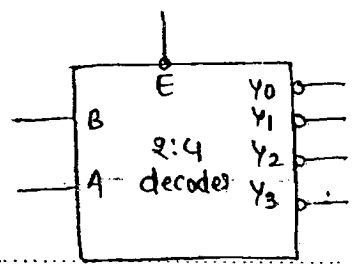
E=0 ; active → logic 0/1
=1 ; Inactive → High impedance (or) tristate.

* In high impedance state the line does not draw any current.
* The device connected to bus behaves as disconnected in high impedance state.
* Buffer can be used to permit one device either to transmit (or) receive among many.

Eg:- 74LS244 → unidirectional buffer
74LS245 → Bidirectional buffer.
LS → Low power shottky

(2) Decoder → * It is logic ckt which identify the combination of i/p signal & selects one of the o/p.

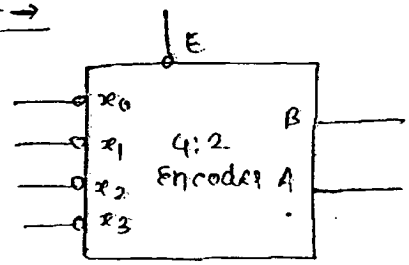
Ex:- 74LS138
↓
3:8 decoder



decoding logic

i/p B A	o/p
00	Y0
01	Y1
10	Y2
11	Y3

(3) Encoder →

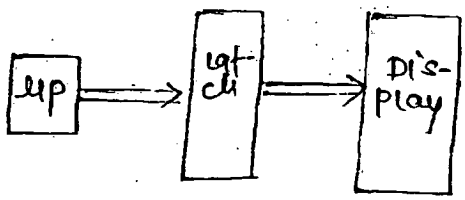


i/p	o/p B A
x0	00
x1	01
x2	10
x3	11

* It is a logic ckt which produces or generates the code of one of the i/p signals at the o/p. used in comm. ckts & also to interface a keyboard to the processor.

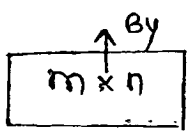
Ex:- 74LS148
↓
8:3 priority encoder

(4.) Latch →



* As the data is present on the bus for few µs, latch is used to hold the data for sometime such that o/p units i.e. display may recognise the data.

Notation (or) representation of m/m chip →

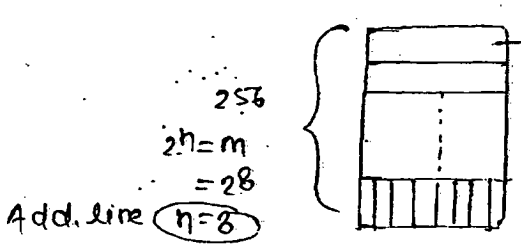


m = no. of add/m/m location

n = no. of bits per location

$2^n \times N = m$

256 x (8) → data bus



capacity – 256 Bytes

Finding the length of add. & data bus →

(i) 1024 × 1024

data lines — 1024

add. lines $2^n = m = 1024 = 2^{10}$

$n = 10$

1024 × 1024

$2^{10} \times 2^{10}$

2^7 $2^3 = 8$ (leave it)

$+ 2^{10}$

$2^{17} \rightarrow$ ~~128kB~~ 128k

capacity → 128k × 8 = 128kB

(ii) 2048 × 256

$m = 2048$

$= 2(1024)$

$= 2^1 \times 2^{10}$

$= 2^{11}$

$n = 11 \rightarrow$ add. lines.

2048 × ~~2048~~ 256

$2^{11} \times 2^8$

$2^{11} \times 2^5$ 2^3 (leave)

$2^{16} \times 8 = 64kB$

64k × 8 = 64kB

Calculating no. of chips req. to design a m/m →

$$\text{Chips Required} = \frac{\text{m/m. to be designed}}{\text{available capacity}}$$

Q. → Find no. of 256 × 8 RAM chips req. to design 4kB.

(a.) 8 (b) 16 (c) 32 (d) 64

Soln →

$1kB \rightarrow 1024\,B$

$= 1024 \times 8$

$= 2^{10} \times 8$

$$\text{Chips} = \frac{4kB}{256 \times 8} = \frac{4(2^{10} \times 8)}{2^8 \times 8}$$

Chips = 16

(OR)

$1kB = 2^{10} = 2^2 \times 2^8$

4 256B

256 × 8

$1kB \rightarrow 2^{10} \rightarrow 2 \times 2^9$

2 512B

512 × 8

$1kB \rightarrow 2^{10} \rightarrow$ 1024 × 8

256×8 ⟶ 256B ⟶ 25 paise

~~256×8~~ 512×8 ⟶ 512B ⟶ 50p

1024×8 ⟶ 1KB ⟶ 1Rs.

Q→ Find no. of 1024×2 m/m chips req. to design 16 KB?

ans→ 64KB

Q→ Find no. of 2048×8 16 for 64kB?

ans→ 16kB 2048×8 - 2kB

$2048 \times 2 \times 8$

$4K \times 8 \longrightarrow 4KB$

$\frac{64KB}{4KB} = 16$

m/m interfacing →

$n - x = y$

n = total add lines of processer

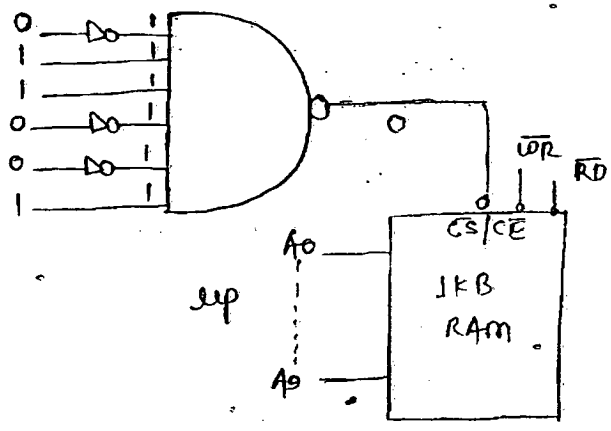
x = no. of add. lines required to for m/m to be interfaced

y = lines for decoding logic.

m/m mapping → Giving names to (or) addressing m/m loch is known as m/m mapping.

*m/m map indicates starting & ending add. range of a m/m chip.

Q→ Find the m/m mapped for following interfacing logic?



$A_{15}\ A_{14}\ A_{13}\ A_{12}\ A_{11}\ A_{10}\ A_9\ A_8\ A_7\ A_6\ A_5\ A_4\ A_3\ A_2\ A_1\ A_0$

0 1 1 0 0 1 | 0 0 0 0 0 0 0 0 0 0

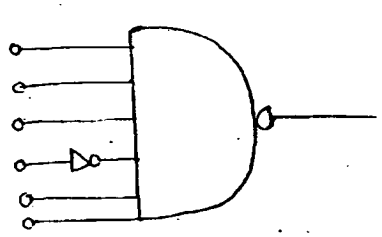
6 4 0 0

0 1 1 0 0 1 1 1 1 1 1 1 1 1 1

6 7 F F

m/m-mapped → 6400kB - 67FFkB

Q→



Sol→

1 1 1 0 1 1 | 0 0 0 0 0 0 0 0

E C 0 0

ECOOH

1 1 1 0 1 1 1 1

E F F F

EFFFH

ECOOH–EFFFH

Q→ Draw the interfacing logic for 1kB RAM is m/m map is ECOOH–EFFFH

Sol→

1kB → $2^{10} = 2^n$

n = 10

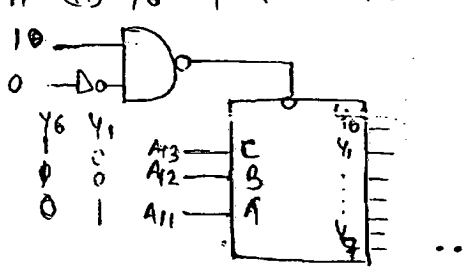
n − x = y

16 − 10 = y

y = 6

Q→ Find the m/m mapped for given interfacing logic if the chip select is connected by

(i) Y_1 (ii) Y_6 of decoder?



8 8 0 0

Y_1 → 1000 1000 0000 0000

Y_6 → 1010 0000

Y_1 → 880OH · 8FFFH

Y_6 → B000H · B7FFH

Q.→ Finding starting & ending addresses.

Method → (1.) Find the no. of add. lines acc. to capacity of m/m chip.

(2.) Put equalent no. of binary ones as that of add. lines. Find the Hexadecimal value that must be added (or) subtracted to get ending add. (or) starting add.

Q.→ Find the ending add. for 4KB ROM if starting add. is 7D3AH

Soln→

$4KB - 2^n$

$n = 12$

1111 1111 1111

F F F

We have to find ending to add:

$$\begin{array}{r} 7D3AH \\ + \quad FFFH \\ \hline 8D39H \end{array}$$

Q.→ Find the starting add. for 8KB RAM is ending add. is 8E4DH

Soln→

$8KB - 2^n$

$n = 13$

0001 1111 1111 1111

1 F F FH

10H 10H

7 D3 10H

$$\begin{array}{r} 8E4DH \\ - \quad 1FFFH \\ \hline 6E4EH \end{array}$$

Digital electronics → Chapter (5)

(1.) access Rate = $\frac{1}{\text{access time}(T_A)}$ $S = (nA)$

(b.) (a)

$T_C = T_A +$ additional time req. before start of next access

↓ cycle time ↓ access time

$(T_C = nS)$ nano secs.

Transfer Rate = $\frac{1}{T_C \text{ (cycle time)}}$ $= (nA)$

(2.)(b) (3)(c) (4)(b) 5(a) 6(d) 7(c) 8(d) 9(c) 10(c).

take A_{12}

A_{15} A_{14} A_{13}	A_{12} A_{11} — A_0
0 1 1	0 0 — 0
	1 — 1
0 1 1	1 0 — 0
	1 — 1

(8.)

A_{15} – A_{12}	A_{11} A_{10} A_9 A_8	A_7 – A_0
represented { X X X	0 0 0 1	
	0 1 0 1	
	1 0 0 0	
⟶ ✓	1 0 0 0	

Hof

* Imp. interfacing IC's →

8251 → USART (universal syn. Asynchronous) →

* Used for serial comm. b/n processer & I/O.

Standards:- Ex:- RS232C, RS422A

RS – Recommended std.

Features of RS232C →

(1.) distance – 50 ft

(2.) speed – 20kbaud

Bit Rate = bits / sec.

Baud = symbols / sec.

(3.) Voltage levels :-

logic 0 :- +3V to +25V

logic 1 :- -3V to -25V

8253/54 → Programmable timer →

* Has 3 timer/counters
* Operates in 6 modes

mode 0 → Interrupt terminal count

mode 1 → ~~Single~~ programmable one shot

mode 2 → Rate gen^r

mode 3 → sq. wave

mode 4 → software triggered mode

mode 5 → Hardware triggered mode

8255 (PPI/PIO) →

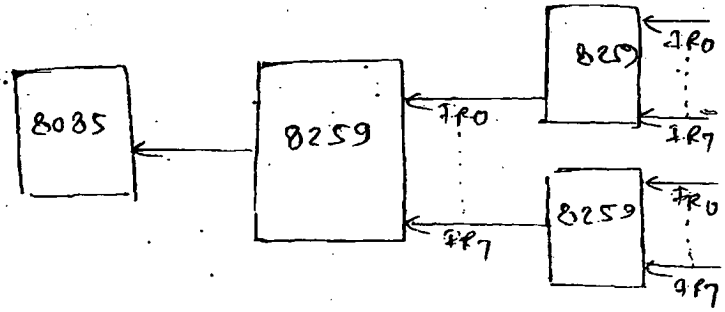
* Programmable peripheral interface
(or)
parallel i/p o/p device

8257/37 (DMA Controller) →

8259 (PIC - Priority interrupt controller) → Used to increase interrupt handling capability of processer.

Each 8259 can support 8 I/o devices in interrupt mode where IR0 has highest priority.

IR0 → Interrupt req; zero.

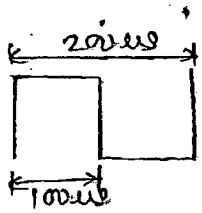


A max^m of 64 I/o devices can be connected in interrupt mode.

8279 - keyboard/display controller

8155 - multipurpose programmable I/o device.

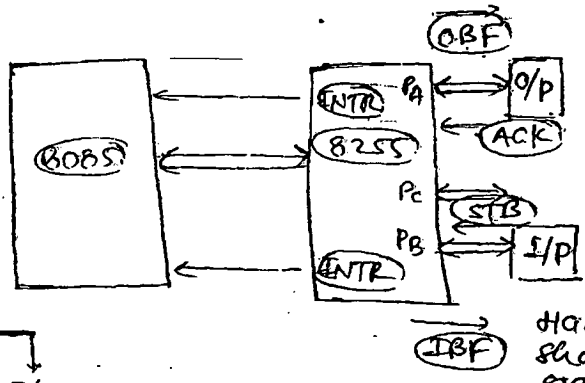
* It has 256B RAM
* 3 ports → PA & PB → 8 bits
PC → 6 bit port.
* Timer/counter
4 modes of operation
mode 0 - single sq.
1 - sq; wave
2 - ~~sq wave~~ pulse on terminal port
3 - pulse on every terminal count.



$$\text{count} = \frac{\text{pulse period}}{\text{Clock period}}$$

$$= \frac{200\,\mu s}{0.33\,\mu s} = (X)_{10} = (Y)_{16}$$

8255 → Handshake signals (or) Control signals.



PA PB PC → 8 bits

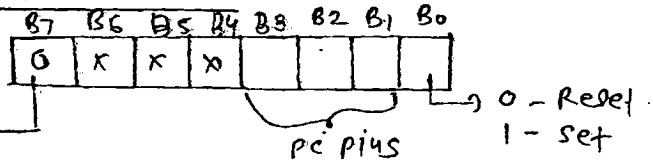
- BSR (Bit set Reset)
- I/O
  - mode 0 → Simple I/O → Any port can be i/p (or) o/p.
  - mode 1 → Strobed I/O mode
    - PA & PB → i/p (or) o/p
    - PC → control / Hand shake signals
  - mode 2 → Bidirectional strobed I/O mode
    - PA ⟺ Bidirectional port
    - PB ⟺ i/p or o/p
    - PC ⟺ control / Hand shake signals.

IBF - I/p buffer full
OBF - O/p —11—
Ack - Acknoledge

A_1	A_0	port
0	0	PA
0	1	PB
1	0	PC
1	1	CWR → control word Reg.

The modes of operation is selected by prog. CWR

CWR format for BSR mode →



~~BSR=1~~
I/O=1

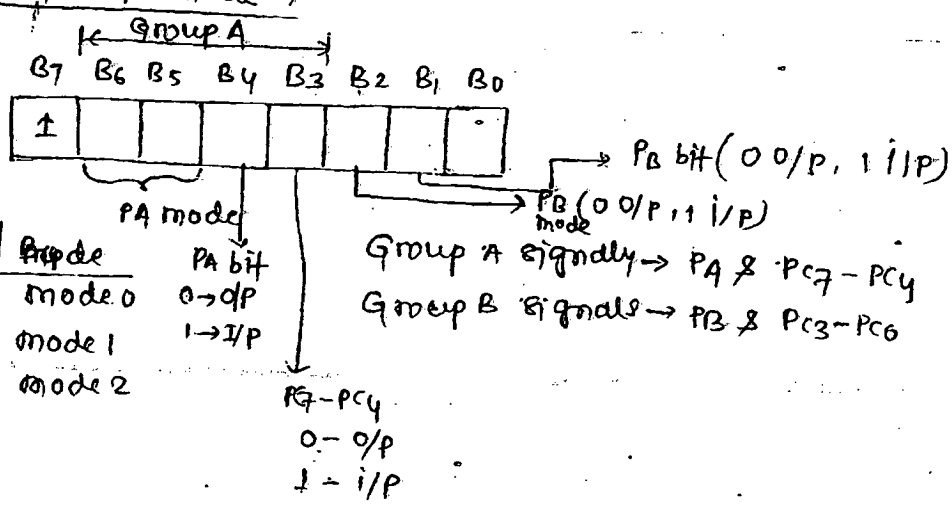
B_3 B_2 B_1	Port C pins
0 0 0	PC_0
0 0 1	PC_1
⋮	⋮
1 1 1	PC_7

Q→ What is CWR data to:

(i) Set PC_4 → 1 — CWR 09H — 0 0 0 0 1 0 0 0

(ii) Reset PC_7 → 0 — 0EH — 0 0 0 0 1 1 1 0

Q CWR format for I/O mode →



B6	B5	Mode	PA bit
0	0	mode 0	0 → O/P
0	1	mode 1	1 → I/P
1	X	mode 2	

Group A signals → PA & $PC_7 - PC_4$

Group B signals → PB & $PC_3 - PC_0$

Find CWR control word for 8255 such that

Q.① P_A → mode 0 → I/P port

P_B → mode 0 → O/P port

P_C → I/P port

1 0 0 1 — 1 0 0 1

9 — 9

② P_A - mode 1 ⇒ O/P port

P_B - mode 1 ⇒ I/P port

1 0 1 0 0 1 · 1 0

A — 6

Imp features of 8086→

* 16 bit µp has 20 add. lines. ; 2^{20} → 1MB
* f_{CLK} = 5MHz ; Range – 5MHz/8 / 10MHz
* +5V dc
* All reg. are 16 bits in length.

Reg ← GDR (general data reg.) → default count reg.

SPR (special purpose reg.)

AX	AH 8	AL 8
BX	BH 8	BL 8
CX	CH 8	CL 8
DX	DH 8	DL 8

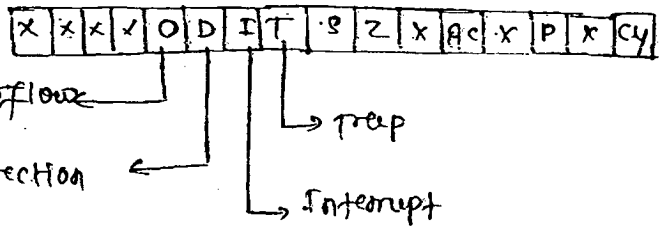
① Pointer reg's → IP → Instⁿ pointer
SP → stack –"–
BP – Base pointer

② Index Reg's → SI → source index reg.
DI → destination ~~reg~~ index reg.

③ Segment Reg⁸
- CS – Code segment
- DS – data segment
- SS – stack segment
- ES – extra seg.

} Used to store Base add.

④ Flag Reg.



TRAP Flag→

TF–1; Single step execution mode. ie. stops after every instn. results can be verified in the reg⁸.
eg:- Debugging.

IF → 1; maskable interrupt (or) recognised
eg:- INTR

DF → Used for string manipulation instn.
- 0; auto-increment mode.
- 1; auto decrement mode.

OF :- It is set if result of a signed operation is not possible to be accomodated in destination reg.

sign	0	1 0 0 1 1 0 1
	+0	1 0 0 0 1 0 1
	1	0 0 1 0 0 1 0

(-ve)

* When processer is Reset control of prog. is xfer to FFFF0H
* It can operate in 2 modes :-
  1. min^m mode
  2. max^m mode

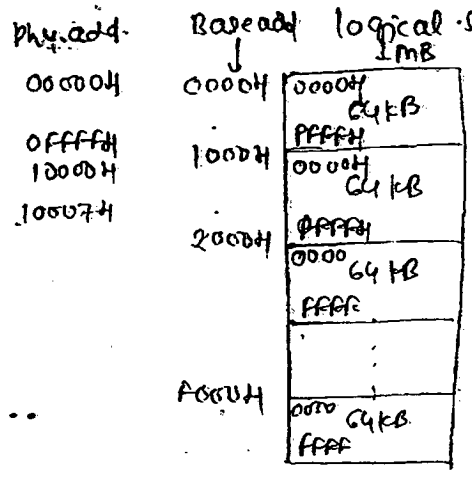
MN/$\overline{MX}$ → 0, maximum of multiprocessor mode → control sig are generated by external clip

1, min^m of single processor mode. (only 8086)

control sig is generated by 8086

* It contains 6 bytes of instn byte queue in which max^m length of instn i.e. 6 bytes can be prefetched.
* At a single interval of time more than one instn can be present at diff stages of execution. known as pipelining tech,
* Speed of operation increases.

Memory segmentation → 1 MB of m/m can be devided into 16 logical segments each of 64kB



Base add. → It indicates the starting loch of segment
It is present in segment regr.

Offset add. → It is a distance of req m/m loch from the base add. in a segment.
It is present in pointer (or) index regr.

physical add. (or) effective add. → It is the posh of req. m/m loch from the begining of mtm.

effective (or) phy. add. calⁿ →

(1) Shift the segment (or) base add. by 4 bit posh left. i.e 1 Hexadecimal digit.

(2) Add the offset add. to shifted segment value.

code segment → prog.
data — " — → data
stack — " — → temp. data
extra — " — → Alternate data segment.

(1) CS: 9000H
IP: OFFFH

9000
0 0 0 0H
+ 0FFF
9FFFH

(2) DS: 8000H
BP: FFFOH

8000
0 0 0 0
+ FFFO
8FFFOH