74

(74)

9

-: HAND WRITTEN NOTES:-

OF

ELECTRONICS & COMMUNICATION

ENGINEERING (1)

-: SUBJECT:-

MICROPROCESSOR & COMPUTER ORGANIZATION

9

(2)

Microprocessor → IES ≡ 20Q.
→ PSU
→ Gate — 1 programming + 1 Memory interface
Question
= 4 marks.

Definition:- it is electronic device that fetch instruction from memory execute them & provided result.

it is integrated chip which has computing & decision making capability.
↳ Analysis of data.

µp	bit of µp	Tech
4004	4-bit	PMOS
8008	8-bit	NMOS
8080	8-bit	NMOS
** → 8085	8-bit	NMOS
→ 8086	16-bit	HMOS
improved by (IBM) 8088	8/16 bit	HMOS ↳ H = high density channel

Note - Externally 8-bit internally 16 bit.

80186	16 bit
80286	16 bit
80386	32 bit
80486	32 bit
80586 ≡ pentium	≡ 64 bit

i7 processor - BiCMOS technology

pentium
pentium pro.
P-II
P-III
P-IV

Main Memory:- RAM

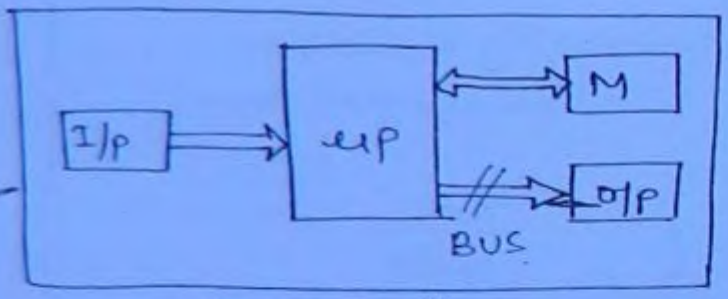
→ instructions or progr. feed in main memory.

ROM - S/w of processor instgallin it & it come in picture only at the time of power switch ON.

- in that all system related information stored.

µ-computer ≡ µP + I/P Device + O/P Device + Memory.

if µP is work as CPU in a computer then this computer is known as µ-computer.



µ-controller

µP = Hardware + Software
µP

Note ① if µ computer on single plateform then it is known as µ-controller.

② A processor cannot perform any task on it own.

ASIC ≡ Application specific integrated chip.
└→ A programmed µ-controller

BUS :- it is group (parallel combination) of metal wire that is used for interfacing b/w different devices.

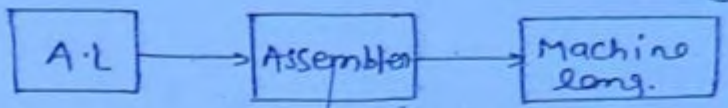
Machine language :-

if command (or instruction) written in binary pattern then it is called machine language.

→ it is plateform dependent language

Assembly language :- if binary pattern replace by english like word that is called "Mnemonics"

→ Assembly language is also platform dependent. (5)



@→ if this task is perform manually then it is known as hand Assembly

→ Assembler is nothing but a S/w that convert Assembly language to machine language.

Low Level language :-

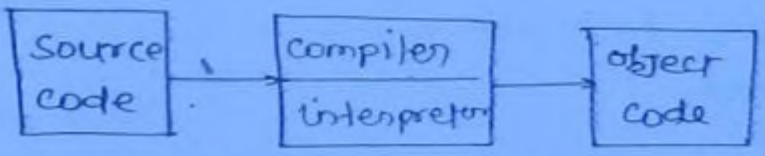
that language which are platform specific or platform dependent

Ex- M·L & A·Lang

High level language :-

All platform independent language is called H·L·L

Ex- C, C++, Java.



User understandable language

machine understandable language

→ Compiler read whole program at once produce it object code which is executed by µ-processor

→ Compiler is nothing but S/w. ex C, C++

- interpretor read one instruction at a time from source code produce it object code which is executed by µp b4 reading next instruction and interpretor is a S/w.

8085 µP :- Ex BASIC (Beginner's purpose symbolic instruction code)

↳ 8085A is a commonly known 8085 µp.

↳ it is based on NMOS technology.

↳ 8085 µP is 8-bit µp.

→ instruction set of 8080 µp is exactly common in instruction set of 8085.

upward compatibility

Note: ~~upward~~ of 8080 + RIM + SIM = inst. set of 8085

two new inst. (RIM, SIM) are added in the inst. set of 8080.

RIM – Read interrupt mask

SIM – Set interrupt mask.

(6)

bit of processor:-

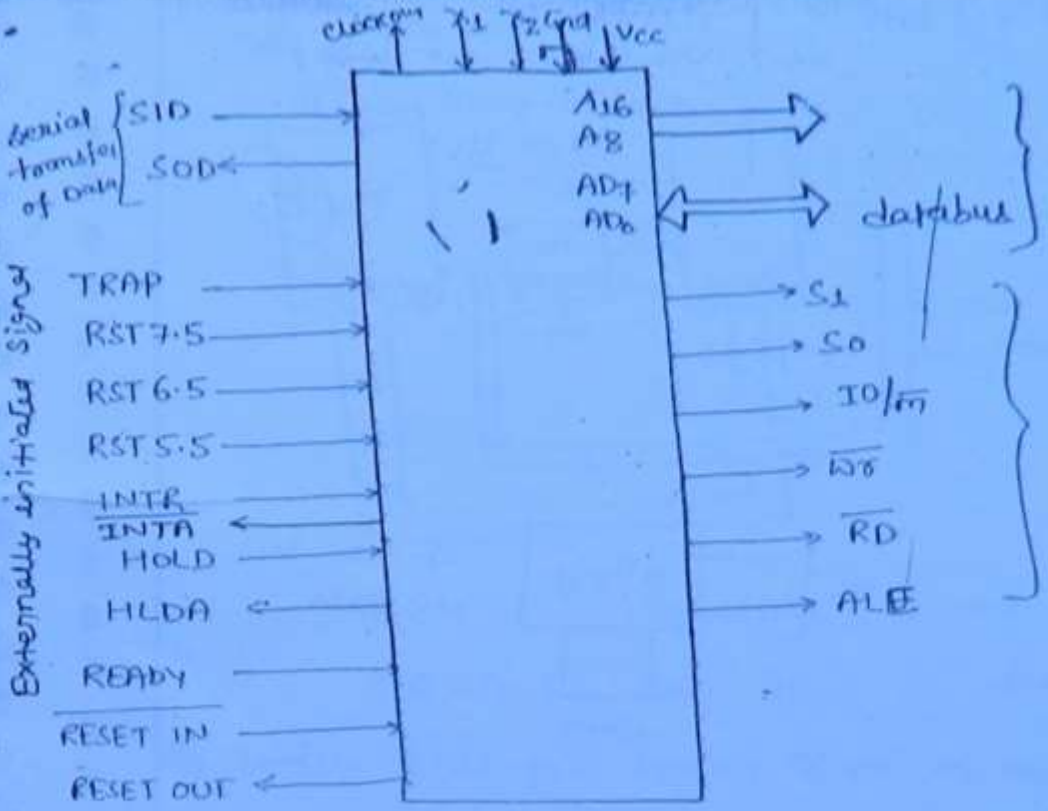
- Size of ALU is defined as bit of µprocessor.
- in one machine cycle no. of bit of data execute is known as bit of µp.
- f = 3MHz (in 1971 after that upto 5MHz).
  - (i) Speed ∝ f
  - (ii) Speed ∝ Size of ALU i.e → Data BUS.

tech of 8085:- External Arch. (IES-1φ)
internal Arch.

External Arch:-

- it is 40 pin µp.
- there is no dizable enable pin 8085 µp chip
- one Ground pin that is pin no 20.
- all 40 pins divide in 6 group as per there function.
  - (i) Addr Bus/pin
  - (ii) Data pin/bus
  - (iii) Control & status pin
  - (iv) Serial i/p & o/p pin.
  - (v) externally initiated signal pin.
  - (vi) power supply & freq pin.

{ Bus is nothing but parallel combination of metal wire.



(7)

in 8085

** total no of pins direction outward = 27

——— inward to up = 21

(1) Address pins/Bus :-

→ Add bus of 16 bit

→ Unidirectional/Outward

2^{10} = Kilo

2^{20} = Mega

** → Total no of Memory location that can interfaced with 8085

$= 2^{16}$ location

$= 2^6$ K

= 64K location

→ no of bit stored in Memory location that is known as word size

Data pins –

no of data pins = 8

Bidirectional

Maximum memory that can interfaced with 8085 up

$= 2^{16} \times 8$ bits

$= 2^{16}$ byte

= 64 K byte $= 2^6 \times 1..K = 64$ K byte

Note:-

Memory = $(2^N \times d)$ bits

N – Add.
d – data

If d = 8

2^N byte

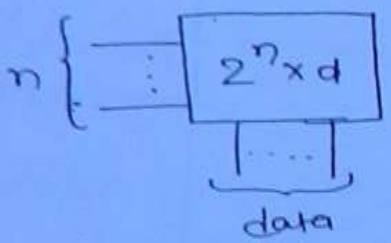
Lower 8 pins of add. bus are used as data pin

data pin + add. pin = 16
(8) (8)

8

$2^{16} \times 8$ Memory. max. no. of pins.

Add. = 16
data = 8
$\overline{RD}$ = 1
$\overline{WR}$ = 1
Vcc = 1
Gnd = 1
28
Chip enable + 1
pin 29.



n – add. bus
d – data.

Control & status pin :-

S_1, S_0 } By measuring status of these two pin we can find out which machine cycle going on inside the µp.

$\overline{RD}/\overline{M}$:-

0 transfer of data from memory
1 transfer of data from I/O devices

$\overline{WR}$ = 0 = Write operation perform.

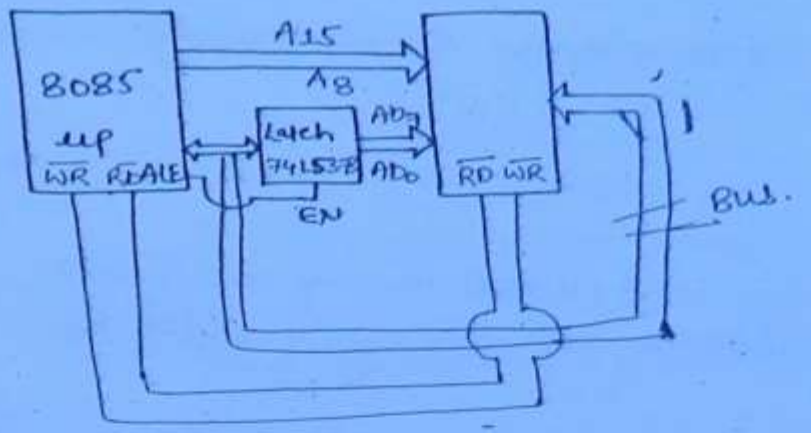
$\overline{RD}$ = 0 = Read operation will perform.

$\overline{RD} = 0$, $\overline{WR} = 0$ } ×
→ Simplex Mode

$\overline{RD} = 1$, $\overline{WR} = 1$ } ✓

Active low (8085) → Only five pins are active low

ALE → add latch enable.



(9)

74LS373
↳ D-flip-flop

ALE = 1 $AD_7 - AD_0$ ≡ add. bus.

ALE = 0 $AD_7 - AD_0$ ≡ data bus.

→ By using this pin lower 8-pin of address bus can be ~~demultiplex~~ used as data ~~pin~~ also. & multiplexing/ ≡ TDM
Demultiplexing

Power supply & freq. pins :-

→ V_{CC} = +5V

→ pin no no 20 = GND

→ f = 3MHz (5MHz 8085 also available but not so popular)

Note - there is 6MHz Crystal oscillator connected b/w X_1 X_2 and a internally devided by two CKT is used to get 3MHz clock freq. (becoz 3MHz Crystal oscillator stability is less).

Clockout :- by using this pin we send clock freq. to other interfaced peripherals for better synchronization.

Serial i/p o/p pins :-

these pins are used for serial transfer of data by using RIM & SIM inst.

RIM ≡ Serial i/p data | SID = 1 ≡ Serial i/p data

~~SIM~~ ≡ Serial o/p data | SOD = 0 Serial o/p data

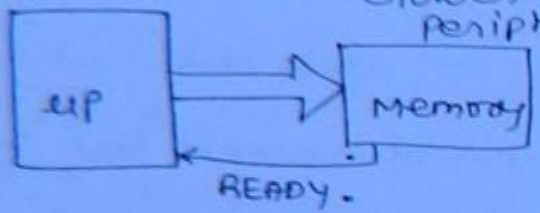
Externally initiated signals:-

- TRAP
- RST7.5
- RST6.5
- RST5.5
- INTR

} hardware interrupts (5pins) (in 5 option are not given) then 6 pins.

HOLD :- By using this pin DMA (direct memory access) like peripherals make request to up for relinquish (Vacant) the data pin

READY :-



By using this pin slower peripherals make interfaced with up.

RESET IN →

By using this pin reset command gives to up for reset.

Note → as soon as up receive gives reset pin command it generate reset out command for reset other interfaced peripherals

Internal Arch. :-

ALU - Arithmatic logic unit

- Accumulator
- Registers
- Register array
- Temporary Reg.
- Latch

→ All arithmatic & Logic operation will occure in ALU

→ control timing ckt

→ interrupt control ckt

Note - Size of ALU = 8 bit (accumulator)

Registers

- User accessable Register (General purpose register)
  - → B
  - → C
  - → D
  - → E
  - → H
  - → L
  - } 8 bit
- Special purpose Register
  - User Access
    - 8-bit { → Accumulator (A)
      - → Status regi (flag reg.)
    - → Stack pointer (SP)
    - → program counter
    - } 16 bit
  - User not Accept
    - 8-bit { → temporary reg.
      - → interrupt Reg
    - → Latches → 16 bit

Note → in 8085, 8, 8bit register and two 16 bit register (user accessed)

Accumulator – it is 8-bit special type of Register.

→ when Arithmatic & Logic operation b/w the two number, one number always taken from Accumulator and result of Arithmatic and logic operation will store in Accumulator

Status Register (flag reg. F)

→ it is 8-bit special type register.

→ in 8085 five flags are define at the diffeunt bits of this register.

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

- S → Sign flag
- Z → Zero flag
- X → don't care
- AC → Auxiliary carry flag
- P → parity Flag
- CY → Carry flag

Note → Status of flag affect according the result of arithmatic and logic operation.

Sign flag (S) →

if on the result of Arith. & Logic operation D7 bit or MSB is zero

- D7 = 0 (+ve) → S = 0 (Reset) (no is +ve)
- D7 = 1 (-ve) → S = 1 (Set) (no is -ve)

Zero flag :-
If in the result of Arith & Logic operation all bits Zero then Z = 1 (Set).

If any bit is one
Z = 0 (Reset)

Parity flag →
If in the result of Arith & Logic operation no. of one's are even then P = 1 (Set)
if no. of ~~1's~~ are odd then P = 0 (Reset)

* 8085 is odd parity based system.

:- Result of Arith & Logic
0 0 0 0 0 0 0 0
S = 0
Z = 1
P = 1

Carry flag (CY) :-
~~If~~ in the arith & Logic operation Carry pass from D_7 or MSB then
CY = 1 (Set)
Otherwise CY = 0 (Reset)

I→ In subtraction operation Carry flag can work also Borrow flag.

→ If Borrow is taken at D_7 bit then
CY = 1
Otherwise CY = 0

Ex:
1 0 1 1 0 0 1 0
0 1 1 1 0 0 1 0
1 0 0 1 0 0 1 0 0
CY = 1 (Set)

Auxillary Carry flag :-
If Carry pass from lower nibble to upper nibble or
$D_3 \rightarrow D_4$ AC = 1 (Set)
Otherwise AC = 0 (Reset)

Ex:
upper ← lower nibble
D_7 D_6 D_5 D_4 | D_3 D_2 D_1 D_0
d_1 d_6 d_5 d_4 | d_3 d_2 d_1 d_0

→ ~~For programmer~~ status of four flag available. CY, Z, S, P.
~~For~~ Flag is available.

Imp → AC flag used internally for the adjustment of content of Accumulator in BCD formats by assuming earlier operation was BCD addition operation. (13)

When DAA (Decimal Adjustment of ACB) inst. will execute.

Stack pointer:- (SP)

→ it is 16 bit special type Reg

→ SP hold the add. of top of stack that define inside the Memory.

Programm Counter:- (PC)

→ it is 16 bit special type of Register.

→ Programm counter hold the address of next instruction to be fetch.

$$PC \leftarrow PC+1$$

NT→ For the storage of 16 bit data then two register can be combined in specific way-

B C = B

D E = D

H L = H

→ higher byte of 16 bit data always store in higher order register = B, D, H

→ lower order byte of data always store in C, E, L

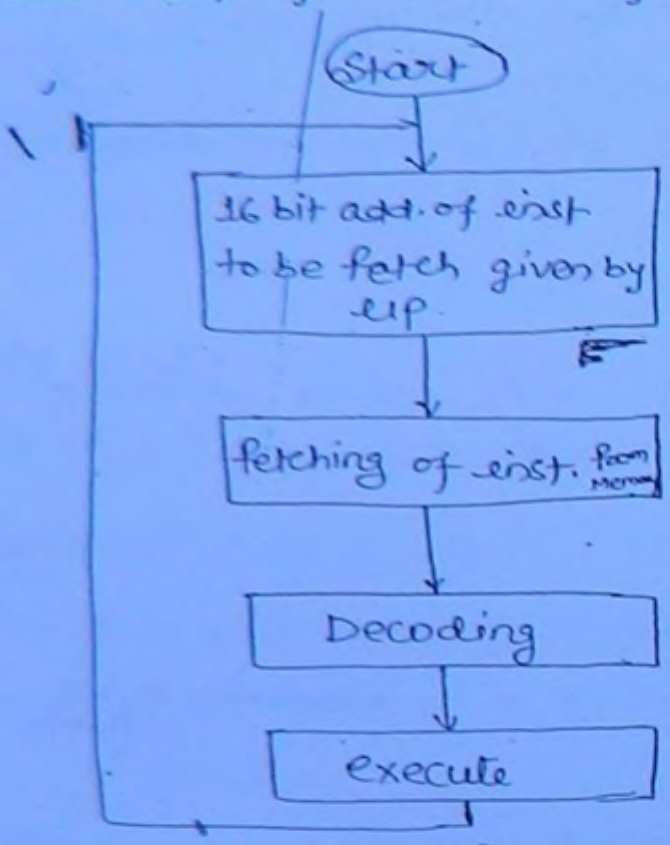
NT **PSW (Programm status word)**

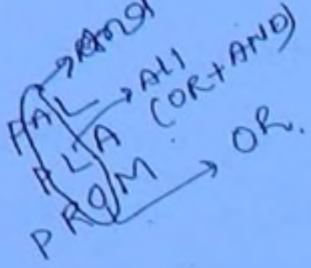
it is user defined 16 bit register in which higher 8 bits are accumulator & lower 8 bit status register

PSW = AF

A- Accumulator

(i) flow chart of execution of program





instruction :-

→ It is command that given to up to perform a task

→ every instruction has two field.

opcode → Operational code → Which type of task to be perform

operand → on which data

→ Lable : opcode operand ; discription

Lable → add of inst.

opcode operand → instn

discription → operation (optional)

→ in 8085, 74 opcodes are define and by using them 256 inst are possible but 246 instructions are available.

MOV B, C } both are different inst.
MOV A, B }

→ Ex.

```
MOV  A C   ≡ 1 byte
MVI B ,28H ≡ 2 byte
LXI D , 2589H ≡ 3-byte
```

(27H, FOH, 90H)

```
1000 : 27
1001 : F0
1002 : 28
1003 : 90
1004 : 89
1005 : 25
```

(MOV) program always store in continuum memory location because we use PC ← PC+1

→ no. of memory byte register to feed any inst. that is known as IWS (inst. word size).

⇒ on the basis of IWS, instructions can be classified in three groups.

(1) 1 byte (IWS = 1 byte)
(2) 2 byte (IWS = 2 byte)
(3) 3 byte (IWS = 3 byte).

(i) if in the inst R, Rp (or) no operand then IWS = 1 byte.

R = Register
Rp = Register pair

Ex

```
MOV A,B
NOP      } IWS = 1 byte.
LDAX B
```

(ii) if in the inst. 8 bit no. (add. or data) is present IWS = 2 byte

Ex MVI C, 25H ≡ IWS = 2 byte

IN 23H ≡ 2 byte

(iii) if in the inst. 16 bit no. (add. or data) is present then

IWS = 3 byte

Ex

```
LXI SP , 2718H  = 3byte = IWS
LDA    5586     = IWS = 3byte.
```

NI →

if a 16 bit no store in memory then higher order byte always at higher order add. (memory add.) lower order byte always at lower order add. (memory add.).

** → in 8085 i/p and o/p port add. of 8 bit

So no. of i/o device = 2^8 = more no. of i/p o/p

(at a time) data devices = 256+256 = 512.

instruction
- NAME
- FORMATE
- IWS
- operation
- Add. Mode.
- flag - status
- machine cycle
- T-state.

Group I

8-bit data transfer instruction :-

MOVE instruction.

(i) MOV Rd, Rs

Rd = destination Register
Rs = source register

Rd } A, B, C, D, E, H, L & M
Rs }

ex MOV B, C

IWS = 1 byte

operation - Content of Rs will pass in Rd (copy of)

[Rd] ← [Rs]

[[ ]] ← data
add

Ex

B = 26
C = 59H
MOV B, C
B = 59H
C = 59H

(ii) MVI R, 8 bit data

(move immediately)

IWS = 2 byte

operation - [R] ← 8 bit data { 8 bit data will store in R

R: A B C D E H L & M

Ex MVI B, 29H

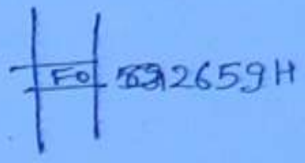
B = 29H

{ if last character of opcode I then data otherwise address

M — it is 8-bit user define register - in memory and address of M is the content of HL pair.

Ex

[H] = 26H
[L] = 59H
MOV A, M
A = F0



(iii) IN 8 bit port add.

IWS ≡ 2 byte

operation - when this inst. will execute data available at the 8-bit port address will store in Accumulator.

[A] ← [8-bit port Add.]

Ex IN AFH

[A] ≡ BEH

10111110

10101111 port add.
A F
Keyboard

'P' - 10111110
B E

(iv) OUT 8 bit port add.

IWS = 2 byte

operation :- when inst. will execute content of Accumulator will available at 8-bit port add. given in the inst.

Ex:
```
MVI B, 92H
MOV A, B
OUT 12H
```

Note :- All above four inst. are data transfer inst. So status of flag will not affect

Group II:-

Machine control instruction :-

1) NOP no operand

NOP

IWS = 1 byte

Operation :- when this inst. will execute processor will not perform any task.

Note :- this inst. is used for creation of delay

2) HLT No operand

IWS = 1 byte

Operation :-

when this inst. will execute further increament of program counter will stop.

Note :- it is last inst. of every program.

Ex

```
1000H MOV BC
1001  MVI A  29H   (1002)
1003  IN   20H    (1004)
1005  MOV DE
1006  HLT
```

PC = 1007

Note :- all above two inst. are machine control inst. and that will not affect status of flag.

Addressing Mode :-

→ Form of add. of data given in the inst is known as add. Mode

① Register add. Mode :-

if add. of data given in the form of Register then Reg. add.

Ex. MOV B, C

(19)

② Direct add. Mode :-

if add of data given directly in the inst.

Ex: IN 25H

OUT 32H

③ immediate add. Mode :-

if data itself given in the inst.

Ex MVI D, 29H — if opcode last character I then immediate add. Mode. Reverse not correct

④ indirect add. Mode :- indirect Reg. add. Mode.

if Content add. of data given in the form of Content of Reg. then indirect add. Mode

Ex: MOV B M.

⑤ implicit add. Mode :-

If add of data not require it is define in opcode only

Ex: NOP, CMA.

Group (III)

8-bit Arithmetic instruction

(i) ADD R

TWS = 1 byte

R → A, B, C, E, D, H, L & M

Operation → Content of R will add in [A] & first result will store in [A]

[A] ← [A] + [R]

Add. Mode R ≠ M → Regis add.

R = M → indirect add.

Ex: [A] - 25H
[B] = 3CH
ADD B
[A] = 61H

25H
01
1 1

S	Z	AC	P	Cy
1	0	1	0	0

[A] = 11000001 = C1H

(20)

1) ADI 8bit data

IWS → 2byte

Operation:- 8 bit data will get added in accumulator & result will store in Accumulator.

[A] ⟵ [A] + [8bit data]

Add. Mode — immediate add. mode.

2) SUB R

IWS = 1 byte

R → A, B, C, D, E, H, L & M.

Operation:- When inst. will execute content of R will get subtracted from content of A & result will store in A.

[A] ⟵ [A] − [R]

Add. Mode

Register Add. mode R ≠ M

Indirect R = M

Ex:-

[A] = 29H

[B] = 35H

SUB B

[A] = 0010 1001

[B] = 0011 0101

[A] = 1111 0100

S	Z	AC	P	Cy
1	0	?	0	1

3) SUI 8bit data IWS = 2byte

8 bit data get subtracted from [A] & result will store in [A]

[A] ← [A] − 8bit data Add. mode

Q: write down one line inst. that make content of ACC 00H regardless of previous value.

(i) SUB A

(ii) MVI A 00H.

(21)

(iv) INR R

IWS = 1 byte

R = A, B, C, D, E, H, L, & M

Operation — when this inst. will execute content of R will increase by 1 & result will store in R

$[R] \longleftarrow [R] + 1_{LSB}$

Ex: B = 21H

INR B

$[B] = 00100001 + 1_{LSB}$

00100010

[B] = 22H.

Add. mode — R ≠ M Reg. add.

R = M Indirect Reg. add. Mode.

(vi) DCR R

IWS = 1 byte

R = A, B — —

Operation :— When this inst will execute content of R will decrease by one & Result will store in R.

$[R] \longleftarrow [R] - 1_{LSB}$.

Note :→ ADD, ADI, SUI, SUB will affect status of all flag

→ INR & DCR will affect only four flag (S, Z, AC, P)

→ INR & DCR will not affect carry flag

B = FFH

INR B

[B] = 1111 1111 + 1

0000 0000

S	Z	AC	P	CY
0	1			previous

Group IV - 8-bit Logical operation :-

1) AND operation :-

(i) ANA R.

IWS = 1-byte

R → A, B, C, D, E, H, L & M.

Operation :- When this inst. will execute content of R will get AND operation with [A] bit by bit result will store in A.

Ex:

[A] = 0000 1100 (0CH)

[B] = 5EH

[A] = 10101100 - ACH

[B] = 01011110 - 5EH

[A] = 00001100

(ii) ANI 8-bit data :-

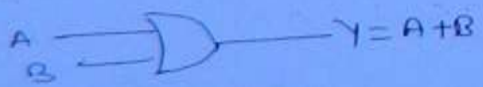
IWS = 2 byte

Operation :- 8 bit data will get AND operation with content of Accumulator and Result will store in [A]

Add. Mode = Immidiate add.

Ex ANI 00H

(2) OR operation :-



i) ORA R

IWS = 1 byte

R = ABCDEHL&M

Operation :- Content of R will get OR operation with content of A bit by bit & result will store in A.

Ex: MVI A 29H

MVI C 1EH

[A] = 29H = 00101001

[B] = 1EH = 00011110

[A] → 00111111

if R≠M Register add. Mode

R=M indirect add. Mode.

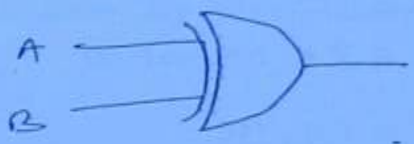
(iv) ORI 8 bit data

IWS – 2 byte

Operation:- 8 bit data will get OR operation with content of ACC. bit by bit & result will store in [A].

Add. Mode = immidiate

(v) Ex-OR operation →



$Y = A \oplus B = A\bar{B} + \bar{A}B$

(i) XRA R

IWS = 1 byte ; R → A B C D E H L & M.

operation:- when this inst will execute content of [R] will get EX-OR operation with [A] bit by bit & result will store in [A].

Ex:

[A] = 0101 0011 = 53H

[D] = 1011 0010 = B2H

XRAD → 1110 0001 = [A] = E1H

(ii) XRI 8 bit data :-

IWS = 2 byte

Operation:- 8 bit data will get EX-OR operation with content of [A] bit by bit & result will store in [A]

Add. Mode → immidiate Add. Mode.

CMA no operand Mode :- IWS = 1 byte

Operation:- when this inst will execute CMA [A] will get Complimented bit by bit & result

$[A] \leftarrow \overline{[A]}$

Ex: $[A] = 29H = 0010\ 1001$

CMA

$[A] = 29H = 00101001$

$[A] = D6H = 11011110$

Ex:- flag can be classified in two catgories.

i) Affect according the final result of A & L operation
= S, Z, P.

ii) Affect according the process of A & L operation
= Cy, AC

	S	Z	AC	P	Cy
ANA	✓	✓	1	✓	0
ANI	✓	✓	1	✓	0
ORA	✓	✓	0	✓	0
ORI	✓	✓	0	✓	0
XRA	✓	✓	0	✓	0
XRI	✓	✓	0	✓	0
CMA	X	X	X	X	X

X → not affect

✓ → According the result

0 → Reset

1 → set

→ CMA will not affect any status of any flag.

→ ~~AND, OR, EXOR with~~ always reset the ~~OF~~ Reset Carry flag.

→ AND operation always set the AC flag.

→ OR & EX-OR Operation will always Reset the AC flag.

Ques

MVI A, 2AH

ADD A

ORI AFH

INR A

CMA

HLT

after the execution of HLT instruction, status of flag.

Soln

[A] = 2AH

[A] = 0 0 1 0 1 0 1 0
      0 0 1 0 1 0 1 0

[A] = 0 1 0 1 0 1 0 0

AFH = 1 0 1 0 1 1 1 1

      1 1 1 1 1 1 1 1

                    1   INR A

      0 0 0 0 0 0 0 0

S	Z	AC	P	Cy
0	0	1	0	0
1	0	0	1	0
0	1	1	1	0

↳ status of flag

Ques

FF00 MVI A 23H

FF02 MVI B 32H

FF04 XRA B

FF05 ADI 88H

FF07 HLT

after execution of above program value of program counter (PC)

[B] = ? and PSW (program status word)

[A] = 23H = 0010 0011

[B] = 32H = 0011 0010

[A] = 0001 0001

88H = 1000 1000

      1001 1001

[A] = 99H , [PC = FF08]

S	Z	x	AC	x	P	x	Cy
0	0		0		1		0
1	0	0	0	0	1	0	0

8 4

PSW = A F

= 99 84 H

Ans:- After the Arithmetic operation status of flag Register BBH Then content of [A] may be

(26)

@ DBH ⓑ 75H ⓒ 65H ✓ ⓓ B6H

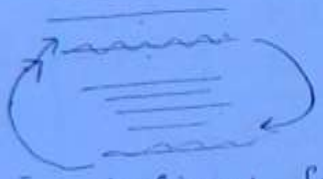
	S	Z	x	AC	x	P	x	CY
	1	0	1	1	1	0	1	1
BB								
DB	1	1	0	1	1	0	1	1
75	0	1	1	1	0	1	0	1
65	0	1	1	0	0	1	0	1
✓ B6	1	0	1	1	0	1	1	0

P=0 odd parity.
Z=0 S=1

ⓐ Auxillary (AC) and carry (CY) are check when process is given only then these parity will find out.

Branch operation :-

- Loop is the special case of branch operation



- In 8085 3 instruction is defined for branch operation.

1. JUMP
2. CALL
3. RESTART

Jump instruction :-

JMP 16 bit address → Vector Location (unconditional JMP)

IWS = 3 byte

Operation :- when this instruction will execute control will jump at given vector location (address)

Ex

Add. Mode → Immediate Add. Mode

Eg:

```
1000  MVI B 27H
1002  MOV D, E
1003  JMP 2918H  ----> 2918
1006  ------
```

1004 → 18
1005 → 29

★ Jump instruction is data transfer instruction so status of flag will not affect.

Instruction cycle, Machine cycle and T-state :—

Instruction cycle :-

total time required by execution to execute the instruction is known as instruction cycle.

★ Instruction cycle is the combination of one or more than one machine cycle.

or

Instruction cycle out of 6 machine cycle one or more than one machine macycle will exist.

Machine cycle :—

in 8085 Six type of Machine cycle exist

[1] Memory Read M. cy (R)
[2] Opcode fetch M. cy (F/c)
↳ (Machine code fetch M. cy)
[3] Memory write M. cy (W)
[4] Input Read M. cy (I)
[5] output write M. cy (O)
[6] Bus idle M. cy (B)

Definition :- total time required by execution to execute different type of task in the execution of instruction.

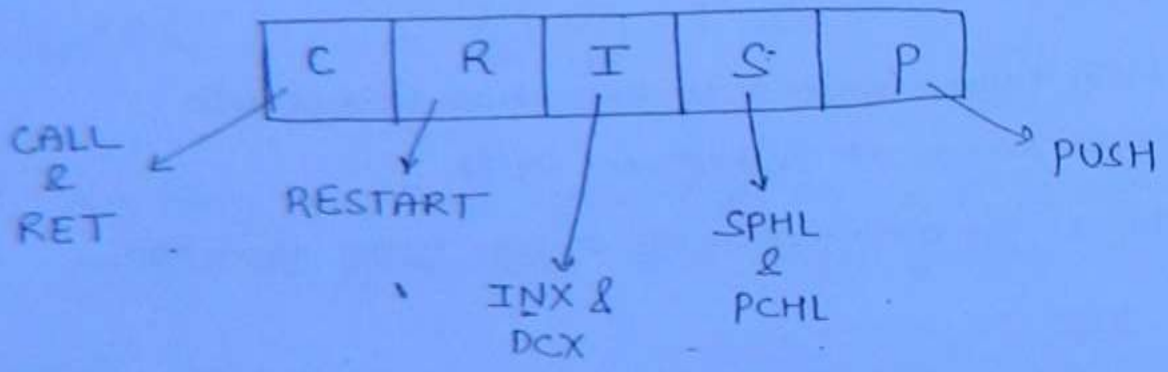
[1]- Fetch Machine cycle (F/S)—

- It is first or only first machine cycle of every inst.
- total time required by execution to fetch opcode (M. cy) from memory
- opcode fetch M. cy is the special case of Memory Read operation.

F = 4T → (238 code)

S = 6T → (8 code)

opcode/machine code required 6T state M. cy



[2]- Memory Read Machine cycle (R):—

- Time required to read 8-bit data from Memory.

$$1R = 3T$$

$$T \text{ state} = 1T = \frac{1}{f}$$

[3]- Memory Write Machine cycle (W):-

- time required by execution to store 8 bit data in Memory

$$1W = 3T$$

[4]- Input Read Machine cycle (I):-

- Time required by execution to read 8-bit information from I/P port

$$1I = 3T$$

[5] - o/p write M.cyle (O) :-

Time required by execution to make available 8 bit data at o/p.

1 O = 3T

[6] Bus idle Machine cycle :- (B)

during the ~~exprer~~ execution of some special inst. time for which Bus of ~~i~~p will be in idle condition and

1 B = 3T

** this Maching cycle will exist in the execution of DAD inst.

Inst.	M.Cy.	T-state
MOV B C	F	4T
MVI D, 29H	FR	7T
MOV B M	FR	7T
IN 25H	FRI	10T
OUT 19H	FRO	10T
ADD B	F	4T
ADD M	FR	7T
NOP	F	4T
HLT	F	4T/5T
SUB M	FR	7T
INR D	F	4T
INR M	FRW	10T
ANA M	FR	7T
CMA	F	4T
JUMP 16bit add	FRR	10T

	IO/$\overline{M}$	$\overline{RD}$	$\overline{WR}$	S_1	S_0
Fetch M·cy	0	0	1	1	1
Memory Read M·cy	0	0	1	1	0
Memory Write M·cy	0	1	0	0	1
i/p Read M·cy	1	1	0	1	0
o/p Write M·cy	1	1	0	0	1

S_1	S_0	
1	1	→ fetch M·cy.
1	0	→ Read operation
0	1	→ Write operation.

Conditional JUMP inst. :-

JC 16 bit add. if Cy = 1

JNC " Cy = 0

JZ " Z = 1

JNZ " Z = 0

JP " S = 0

JM " S = 1

JPE " P = 1

JPO " P = 0

IWS = 3 byte

operation:- If condition satisfy then jump instruction will execute otherwise skip it.

→ Add Mode → immidiate Add Mode same as unconditional Jump.

- Conditional JUMP
  - satisfy = FRR = 1
  - Not satisfy = FR = 7T

→ if is also data transfer inst. so state of flag is not affect

16 bit data Tx. instruction:-

(i) LXI Rp 16 bit data :-

Rp = BC, DE, HL, SP.
B D H

IWS → 3 byte.

- 16 bit data given in the instruction will store in Register pair.

[Rp] ← [16 bit data]

Add. Mode → immidiate Add Mode.

M. Cycle FRR

LXI B 2011 H

Note

LXI SP 16 bit data

[S] = 16 bit data.

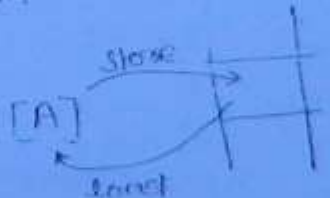
this inst is use for initilisation state in machine Memory.

LXI SP 11FFH

[SP] = 11FF

(ii) LDA 16 bit address :-

IWS = 3byte



- data available at the Memory add that is given in inst.

[A] ← [[16 bit add.]]

Add Mode → Direct add. Mode

M.Cy FRRR

LDA 20FC

iii) STA 16 bit add :-

IWS = 3 byte.

Operation: content of [A] will store at the memory location, that is given in instruction.

[A] ⟶ [[16 bit add]].

Ex: STA 29DE H

2000 - STA
2001 - DE
2002 - 29

Machine cycle → FRRW

Add. Mode → direct addressing mode.

iv) LDAX Rp :-

IWS = 1 byte

Operation :- data available at the 16 bit memory location of Rp will load in [A].

Ex: LXI B 2919H
LDAX B
[A] = FC

add. Mode →

LDAX Rp
- → indirect Reg. add. mode (source data).
- → Register add. mode (destination data).

M cycle → FR

v) STAX Rp :-

IWS = 1 byte

Content of [A] will store at Memory location that is content of Register pair.

[A] ⟶ [[Rp]]

- STAX Rp → indirect add. (destination data)
  → Register add. (source data)

Note →

[1]- for LDAX and STAX possible value of Rp.

R_p = B/C
D/E

[2]- All above inst. is data transfer instruction so status of carry flag will not affect.

Ques

```
    MVI   A   F0H
    ORA   A
Loop: INR   A
    JNC   Loop,
    HLT
```

How many time loop will execute.

[A] = F0 = 1111 0000

	Cy
1111 0000	
	0
[A] → 1111 0000	0
1	
1111 0001	
1	
1111 0010	
1	
.	
.	
.	
1111 1111	
1	
0000 0000	

Q. INR will not affect Carry flag.

Loop will execute ∞ time.

Q:- MVI A FOH FR = 7T
ORA A F = 4T
LOOP: INR A F = 4T
JNZ LOOP FRR/FR = 10T/7T
HLT F = 4T.

If f = 3MHz then find total time of execution of program.

total T-state = 7T+4T+15T (14T) + 1 (4T+7T) + 4T
= 236T
= 236 x $\frac{1}{3}$ μsec
= $\frac{236}{3}$ μsec

16 bit Arithmatic instruction :-

1) INX Rp
IWS = 1byte.
Rp → BC, DE, HL.

Content of Reg. pair will increase by 1 and Result will store in Reg. pair.

[Rp] ← [Rp] + 1_{LSB}

Add. Mode → Reg Add mode
M. Cy → S (C.T).

2) DCX Rp
IWS = 1 byte
Rp → BC, DE, HL.
Content of Rp will decrease by 1 and result will store in Rp.

[Rp] ← [Rp] − 1_{LSB}.

Note :- INX & DCX will not affect status of flag.

(iii) DAD Rp

IWS = 1 byte.

Rp → BC, DE, HL

Content of Rp will get added with [HL] and Result will store in [HL] pair.

→ Reg. Add. mode.

*** → M.Cy = FBB ≡ 10T

Note → DAD inst. is affect only one flag that is carry flag

Ques =

```
1000H : MVI A A1H
1002H   LXI H 1007H
1005H   SUB M
1006H   OUT 05H
1008H   HLT
```

[A] = A1H

[H] = 10H ; [L] = 07H

[A] = 1010 0001

[M] = 0000 0101

[A] = 1001 1100

9 C

after the execution of this program display the o/p port

9CH Ans

(iv) HL pair is add. of M and Content of HL pair is the add that add data present is M.

Ques:

```
LXI H 2000H
LDA 2002H
XRA M
MOV E,A
MVI D,20H
LDAX D
OUT 01H
```

Memory Location	Data
2000H	00H
2001H	01H
2002H	02H
2003H	03H

after the execution of this program

[H] = 20H , [L] = 00H

[A] = [02]H

[A] = 0000 0010

[M] = 0000 0000

[A] = 0000 0010 = 02H

[E] = 02H [D] = 20H

8-bit logical Rotational instruction:-

Execution of these instruction based upon content of [A].

after the execution of these instruction content of accumulator will rotate by one bit either left or Right as per instruction.

RLC No operand → content of [A] rotate by one bit left without carry.

RAL No operand → Content of [A] Rotate by 1 bit left with carry

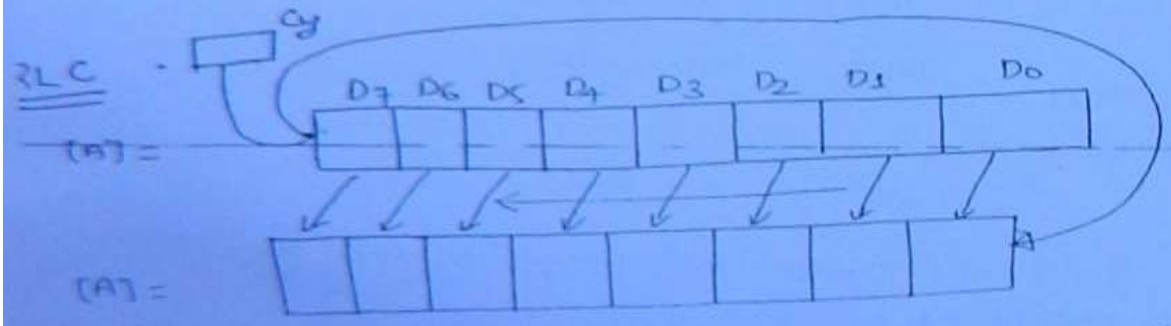
RRC No operand → content of [A] Rotate by 1 bit Right without carry.

RAR No operand → Content of [A] will Rotate by one bit right with carry.

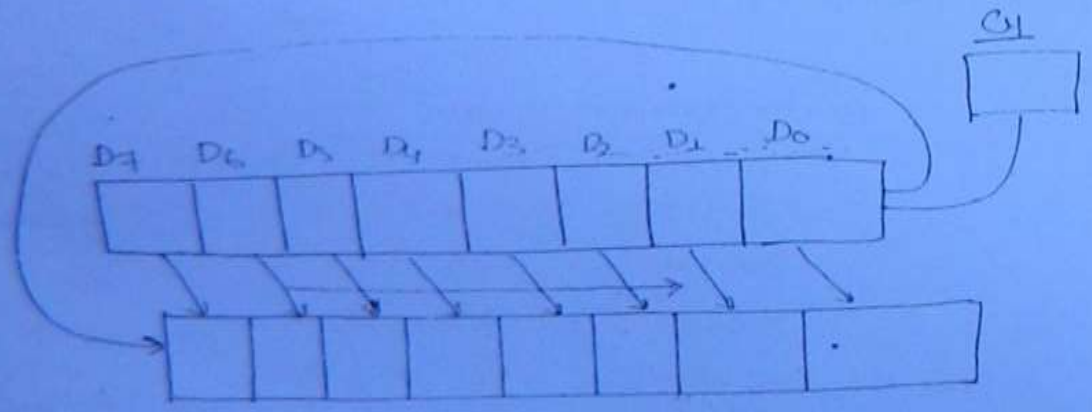
IWS = 1 byte

Operation:- As per inst. Content of [A] will Rotate by one bit.

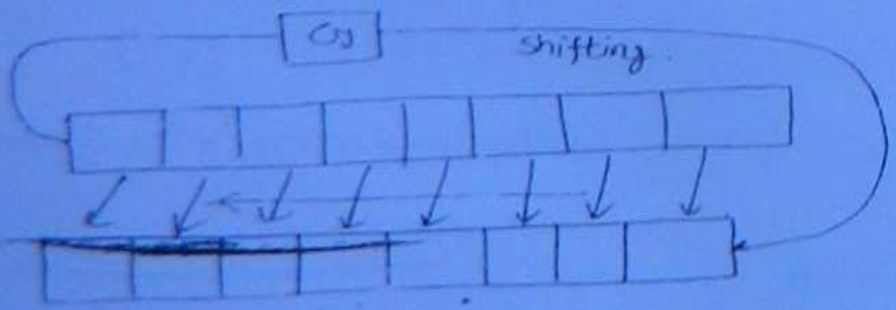
RLC



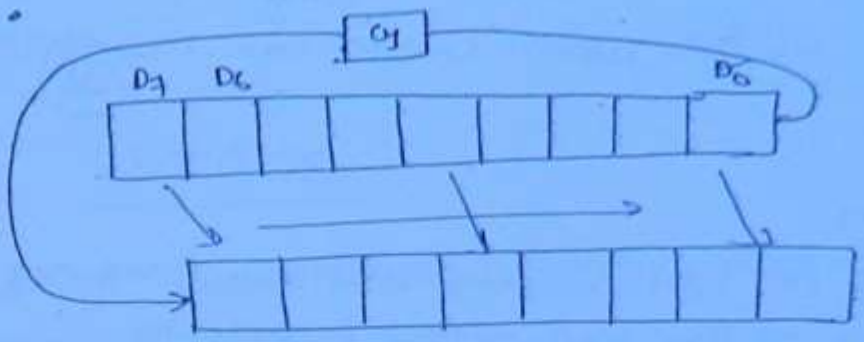
RRC



RAL



RAR



Note :- Logical rotation will affect only one flag that is carry flag

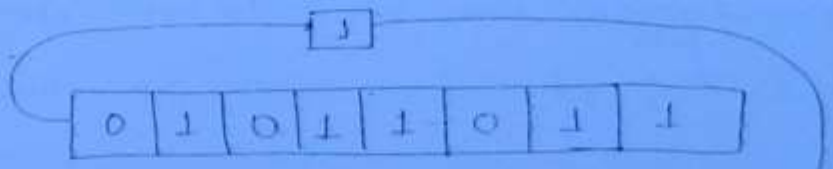
→ Implicit Add Mode.

→ M.Cy = F

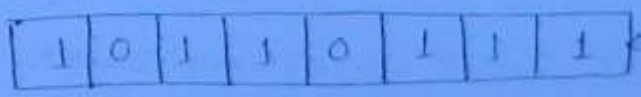
Ques :-
```
MVI A, B9H
ADI A2H
RAL
RLC
HLT
```

What is the content of Acc & status of flag.

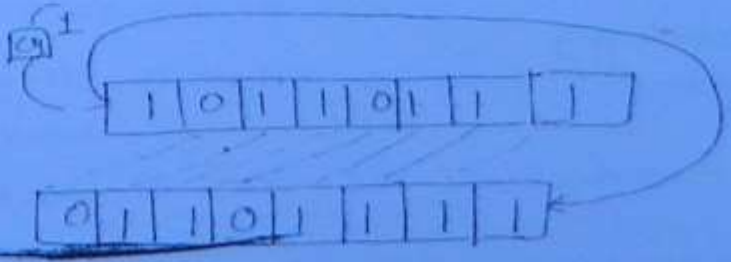
[A] = B9 = 10111001
10100010
[A] = 01011011 Cy = 1



RAL



RLC



7 Logical compair instruction:-

i) CMP R

IWS = 1 byte

R = ABCDEHL & M

operation :- Content of [R] will compare with content of [A] and status of flag will affect accordingly.

Note :- compare inst. is just equivalent to sub. inst. of (A-R) and status of flag will affect according result of (A-R) but result (A-R) will discard it means content of [A] and content of [R] will not change.

	Cy	Z
[A] > [R]	0	0
[A] < [R]	1	0
[A] = [R]	0	1

⇒ ~~Rest of flag affect according the result of (A-R).~~

R ≠ M Register add. mode

R = M Indirect add. mode

M. Cy

R ≠ M = F

R = M = FR

ii) CPI 8 bit data

IWS = 2 byte

8 bit data will compare with content of [A] and status of flag will affect accordingly.

⇒ immediate add. mode

Note - CMP & CPI will affect status of all flag

STACK

it is group of continuous memory location in main memory. that is used for temporary storage of information during the execution of main program.

- Stack is define by instruction LXI SP 16 bit data.
- add of top of stack always store in stack pointer.
- at a time two byte data can store or retrive from stack
- if two byte data stored at the top of stack then it grows upwards in numerically decrease order of its address
- in 8085 two inst are define for store or Retrieve of two byte data at/from top of stack that is

PUSH & POP.

PUSH → store

POP → Retrieve

during the execution of CALL subroutine add. of next instruction will store at the top of stack automatically.

PUSH Rp :-

1 byte Rp ⇒ BC, DE, HL, PSW

Operation :- when this inst will execute content of Rp will store at top of stack.

→ after the execution of PUSH inst. content of stack pointer decrease by 2

→ it is data transfer inst. so status of flag will not affect

→ ~~Reg. content~~ (2) - there is no conditional push inst.

POP Rp

IWS = 1 byte.

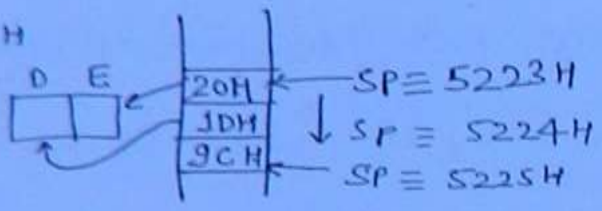
Rp = B, C, DE, HL, PSW
(B, D, H)

(40)

Operation :- when this inst will execute two byte data will retrieve from top of stack & store in Rp.

Eg:

```
LXI SP 5223H
POP D
D=?
E=?
```



:- After the execution of pop inst. content of SP will ↑ by two.

Add. Mode

POP Rp → indirect add. Mode (source data)
→ Register add. mode (destination data)

M. Cy — FRR.

- it is data transfer inst. so status of flag will not affect.

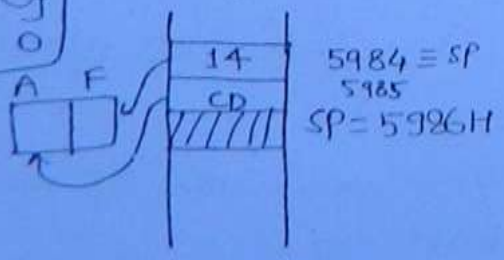
Eg:- but in POP PSW
status of flag may get affected indirectly.

```
MVI A, 29H
ADD A
LXI SP 5986H
LXI B CDH
                14
PUSH B
POP PSW
```

[A] = 29H = 0010 1001
[A] = 0010 1001
[A] = 0101 0010

S	Z	AC	P	Cy
0	0	1	0	0

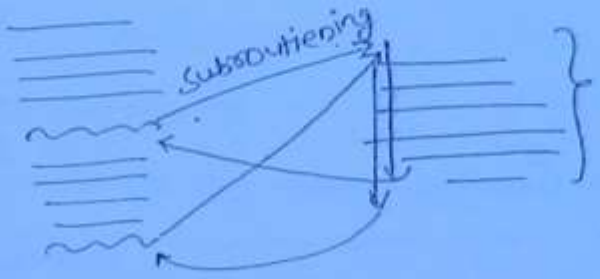
[B] = CDH
[C] = 14H



[A] = CDH
14H = 0001 0100
C x x A P x Cy

Subroutien :-

it is set of inst. that written seperately from main program regarding the task that occur in main program is known as subroutien.



in 8085 two inst. are available for subroutiening.

① CALL inst :-

uncondition CALL inst :-

CALL 16 bit add

IWS = 3byte.

Operation - when this inst will execute control will jump at 16 bit vectors location given in the inst. but before jump add of next inst will store at top of stack.

Ex: 1000 LXI SP 201SH
1003 MOV B C
1004 CALL 5920H → jump 5920H
1007 MVI D 18H

PUSH

CALL = PUSH + JMP

Note → After the execution of call inst content of stack pointer will decrease by two.

*** Note :- immidiate add mode.

MOV SRRWW (1BT)

Conditional CALL inst :-

CC	16 bit add.	CALL inst. will execute if	Cy = 1
CNC	"		Cy = 0
CZ	"		Z = 1
CNZ	"		Z = 0
CP	"		S = 0
CM	"		S = 1
CPE	"		P = 1
CPO	"		P = 0

IWS ≡ 3byte

Note: if condition satisfy then CALL inst. will execute
if condition not satisfy then skip it.

CZ 2011H

CALL 1019H → immediate
→ Register
→ indirect Regi.

Add mode → immediate

Machine code → Condition Satisfy ≡ SRRWW
→ Condition not satisfy ≡ SR

Note ① CALL inst. is data transfer inst so status of flag will not affect

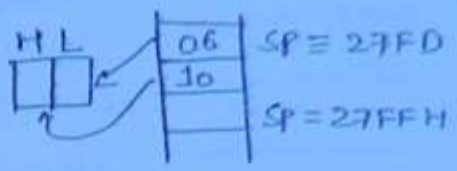
② CALL inst. (conditional in satisfy case (or) unconditional) is the largest of inst. of 8085 & it take 18T

Ques

			SP	HL
1000	LXI SP 27FFH	(a)	27FF	1006
1003	CALL 1006H	(b)	27FD	1006
		(c)	27FF	1003
		(d)	27FD	1003

SP = 27FF

HL = 1006



RET inst :-

- unconditional
- conditional

Note → it is last inst. of every subroutien.

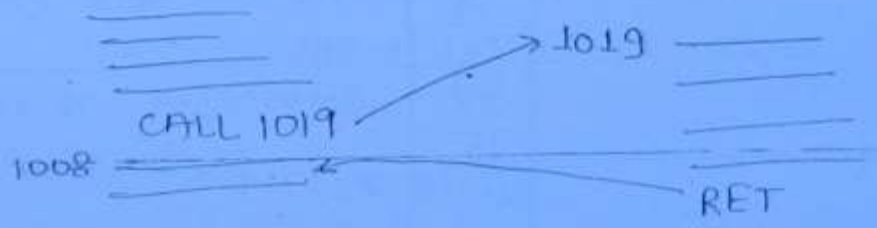
→ RET no operand.

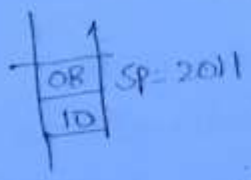
IWS =

operation :- when this inst will execute

(i) two byte data will retrieve from the top of stack & store in program counter.

→ & next inst. fetch from this vector location.





Add. Mode —

- implicit
- indirect

M. cy = 5RR

RET = POP + JMP

Note :- After the execution of RET inst Content of SP will ↑ by two

Q: 1000H: LXI SP 2719H

1003 CALL 3000H

1006 ———

———

———

3000H: LXI H 1919H

PUSH B

PUSH H

PUSH PSW

LXI SP 3C82H

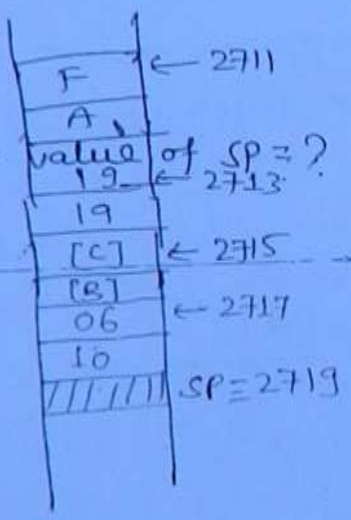
POP PSW

POP H

POP B

RET

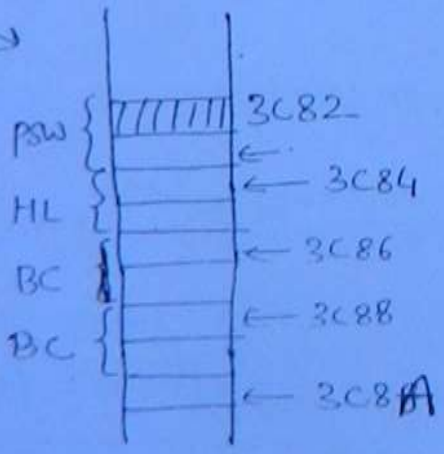
After the execution of program value of SP = ?



SP = 2719

H=19 , L=19

SP = 3C82H



main program ⌐ subroutiens.

```
1002 MOV B C          5985: MVI A, 00H
1003 CALL 5985H       5987: CALL SUB1
1006 MVI D29          (598A) SUB1 : INR A.
                              RET
```

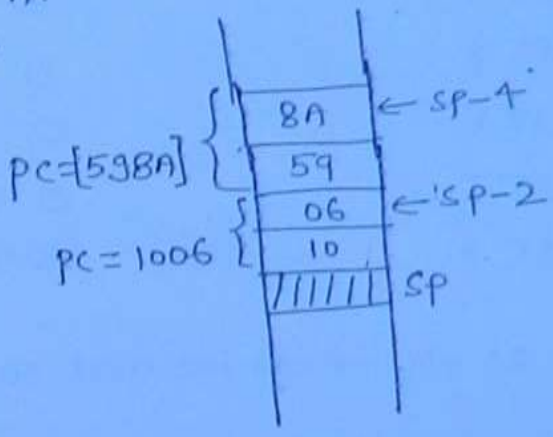
[A]=?

[PSW]=00H

[A] = 01

↓

[A] = 02



RESTART

- it is just like as one byte CALL inst.
- it is used when execution stops in interrupts.

RST N no operand

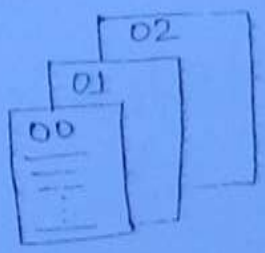
IWS = 1byte

N = 0, 1, 2, 3, 4, 5, 6, 7.

operation:- when this inst will execute control will transfer at specific vector location of memory page no. 00

page no. line no.

XY WZ H



Note:- RESTART is software interrupt

inst. RSTN	vector location 00XXH
RST0	0000H
RST1	0008H
RST2	0010H
RST3	0018H
RST4	0020H
RST5	0028H
RST6	0030H
RST7	0038H.

Note :- XX is the hexadecimal conversion of (N×8)

Add. Mode :- Implicit

M. Cy ≡ S

Advance Arithmatic inst :-

ADC R

Ans = 1 byte

R = A, B, C, D, E, H, L & M

Content of R will get added in content of Accumulator with status of carry flag

$$[A] \leftarrow [A] + [R] + [Cy]_{LSB}$$

Ex. [A] = 21H, [B] = 10H } Cy = 1

ADC B

[A] = 0010 0001

[B] = 0001 0000

1

[A] =

Add. Mode - R≠M Register add.
R=M indirect.

M.Cy R≠M F
R=M FR.

(ii) ACI 8 bit data.

IWS = 2 byte.

operation:- $[A] \leftarrow [A] + \text{8 bit data} + [Cy]_{LSB}$.

→ immediate add. mode.

M.Cy ≡ FR.

(iii) SBB R

IWS = 1 byte

R = ABCDEHL & M.

Content of R will get subtracted from content of Accumulator with status of carry flag.

$[A] \leftarrow [A] - [R] - [Cy]_{LSB}$.

R≠M → Register Add. mode

R=M → indirect

Machine Cycle R≠M ≡ F
R=M ≡ FR.

(iv) SBI 8 bit data

IWS = 2 byte.

$[A] \leftarrow [A] - \text{8 bit data} - [Cy]_{LSB}$.

M.Cy = FR

immediate Add. mode.

Inst. Related to HL pair :-

i) LHLD 16 bit add.

IWS = 3 byte.

When this inst. will execute data available at the Memory location that is given in inst. will load in L- and data available at the next memory location will load in H.

[L] ⟵ [16 bit add]

[H] ⟵ [16 bit add. + 1]

LHLD 201C H

[H] = 7BH

[L] = 69H.

69H	201CH
7BH	201DH

M. Cy FRRRR

Add. mode direct

1000	LHLD	
1001	1C	
1002	20	

ii) SHLD 16 bit add

IWS = 3 byte

operation :-

[L] ⟶ [16 bit add]

[H] ⟶ [16 bit add + 1]

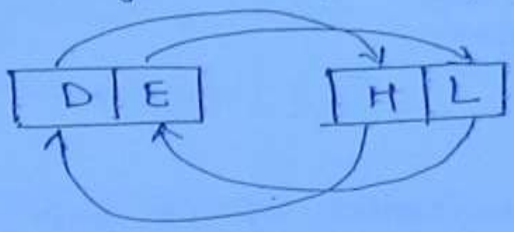
M. Cy (FRRWWW)

1000 SHLD 2019H

(iii) . XCHG no operand.

IWS = 1 byte.

when this inst. will execute content of DE region exchange with the content of HL register pair.



Add. Mode → implicit add.
→ Register add.

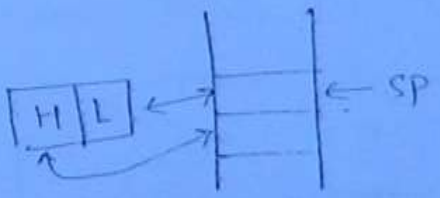
M cy - F

(iv) XTHL no operand

IWS = 1 byte

operation :- when this inst will execute content of HL pair will exchange from top of stack

XTHL = PUSH + POP



M cy F W W R R

Add. Mode → implicit
→ Register
→ indirect Reg.

(v) SPHL no operand

IWS = 1 byte.

when this inst will execute content of HL pair will copy Stack pointer

[SP] ← [HL]

Note → this inst also used for initialising of stack indirectly.

Add. mode → implicit.
→ Register

(50)

M.Cy 5.

(vi) PCHL no operand.

1 byte.

when this inst. will execute content of HL pair, will copy in PC.

$[PC] \longleftarrow [HL]$

Add. Mode → implicit
→ Register.

M.Cy → 5

Note:- All above inst. are data transfer inst. so status of flag with not affe

Que)

```
MVI B 28H    FR   7T
NOP          F    4T
Loop: DCR B  F    4T
JNZ loop     FRR/FR  10T/7T
HLT          F    4T
```

1T = 0.5μs total time elapsed during the execution of program.

total T = 7T + 4T + 33[14T] + 1[11T] + 4T

= 572 T

= 572 × 0.5 μsec

(57)

```
MVI B, 38H        FR    7T
Loop2: MVI C, FFH FR    7T
Loop1: DCR B      F     4T
       JNZ loop1  FRR/FR 10T/7T
       DCR B      F     4T
       JNZ Loop2  FRR/FR 10T/7T
```

$(38)_H = (56)_{10}$

$(FF)_H = (255)_{10}$

Operating freq. = 2 MHz

total time elapsed = ?

total T-state req. to execute inner loop completely once

$= 255(14T) + 11T = 3567T$

total T for program

$= 7T + 55[7T + 3567T + 4T + 10T]$ (satisfy Loop2) $+ 1[7T + 3567T + 4T + 7T]$ (not satisfy)

$= 200932T$

$T = \frac{1}{2MHz} = 0.5\ \mu sec.$

Ques :-

```
LXI H, 2041H    FRR = 10T
MVI A, D6H      FR  = 7T
CMP M           FR  = 7T
MOV B A         F   = 4T
IN 20H          FRI = 10T
SUB B           F   = 4T
OUT 52H         FRO = 10T
INX H           S   = 6T
MOV M, A        FW  = 7T
HLT             F   = 4T
```

(i) How many times cp executes memory read machine cycle of given program. = 6 times

(ii) How many times cp perform memory read operation (or) How many times cp will read data from memory = 17 times

c) How many times up perform read operation or how many times $\overline{RD}$ signal will active low during the execution of program. ≡ 17 times.

d) How many T-state req. of execution of prog ≡ 69T

e) $f = 3$ MHz. So total time req. = ? = 23 μsec.

Some Advance inst :-

i) STC no operand.

IWS = 1 byte.

When this inst. will execute Carry flag will set regardless of previous status. Cy = 1.

→ implicit add. mode.

→ M. Cy – F.

ii) CMC no operand

IWS = 1 byte

When this inst. will execute status of Carry flag will get complemented

$Cy \leftarrow \overline{Cy}$

Add. mode → implicit

M. Cy → F

Note :- above two inst. affect only Carry flag.

iii) DAA

Note :- BCD Number

→ Binary coded no. is not binary no.

→ It is binary coded sys decimal

DAA No operand

Ins - 1 byte

or convert it to option not present in 8085

this inst will execute then content of Accumulator will adjust in BCD formate by Assuming earlier operation was BCD addition

Add. mode → implicit

M. cy → F

Note :-

① if lower nibble of content of Accumulator is greater than 1001 then 0110 get added in it.

② if lower nibble of [A] is less than or equal to 1001 and Auxillary carry flag set the 0110 will get added in it.

③ if upper nibble of [A] is greater ~~then~~ 1001 then 0110 is added in it

④ if upper nibble of [A] is less than or equal to 1001 & Carry flag is set then 0110 will get added in it.

note → this inst will affect all flag.

```
6010H: LXI H 8A79H
6013H MOV A, L
6014H ADD H
6015H DAA
6016  MOV H A
      PCHL
```

[H] = 8AH [L] = 79H

[A] = 79H. [L] = 75H

```
[A] = 01111001
[H] = 10001010
      00000011
[A]   01100110
      01101001 = 69H
```

[H] = 69H

S	Z	AC	P	Cy
0	0	1	1	1
0	0	0	1	0

After the execution of PCHL inst next inst will fetch from

[PC] = 6979H

(a) 6019H
(b) 0379H
(c) 6979H
(d) none of these

Interrupts

In 8085, 5 hardware interrupt pins are available (5)

TRAP → non maskable.

RST 7.5
RST 6.5
RST 5.5
INTR.
} Maskable interrupt.

On the basis of different characteristics interrupts are categories as.

1) Maskable & non maskable interrupt.

interrupt that can make disable is known as = maskable interrupt.

interrupt that can not make disable = non maskable interrupt

in 8085, to control interrupt process there is a flip-flop that is known as interrupt enable flip-flop by set or reset of interrupt enable flip flop - can make enable or disable of interrupt process.

in 8085 ~~two~~ set/Reset two inst are avail to ma Set or Reset of interrupt enable flipflop

1 EI no operand
IWS = 1 byte

when this inst will execute interrupt enable FF will get set and interrupt process will enable

Add. mode → Implicit.

M. Cy → F

1 DI no operand
1 byte

when this inst will execute, interrupt enable FF will get Reset and interrupt process will disable.

Note:- EI & DI is machine control instruction so status of flag will not affected

(ii) Vectored & Non vectored interrupt:- (55)

→ If interrupt is acknowledge & execution transfer at definite vector location of memory page no. 00H then such type of interrupts are known as vector interrupt.

- TRAP
- RST 7.5
- RST 6.5
- RST 5.5

} vector

INTR ⟶ non vector

Note: ① Vectors location for non vectored interrupt provided by externally.

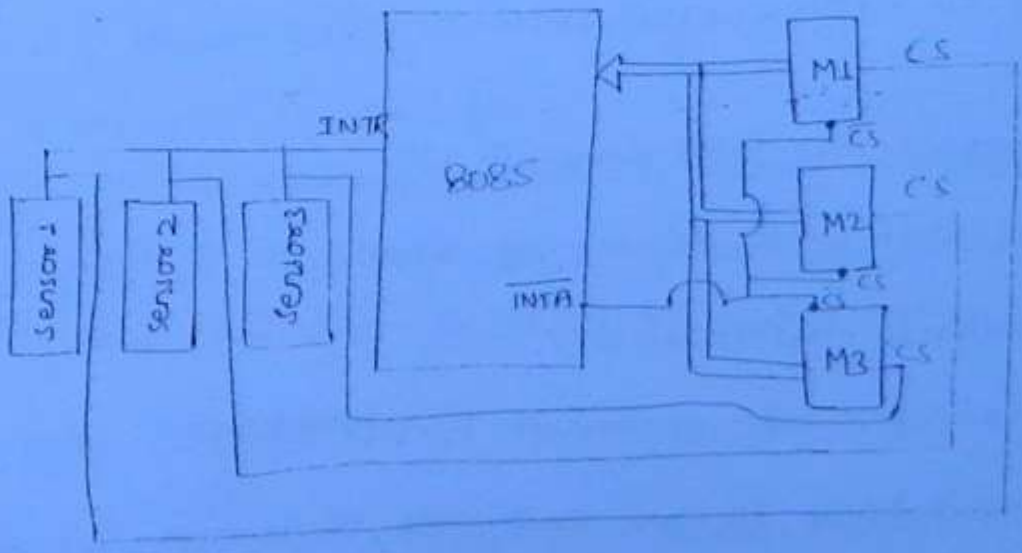
② TRAP is also known as RST 4.5

***

Vectored interrupt	Vector Location.
TRAP	00 2 4
RST 7.5	00 3 C
RST 6.5	00 3 4
RST 5.5	00 2 C

4.5×8 = 36 (360)$_H$ (22)

7.5×8 = (60.0)$_H$



1) Triggering :-

TRAP, RST7.5	edge trigg.
RST6.5, RST5.5, INTR	level trigg.

56

2) TRAP is edge and level both trigg.

3) priority :-

TRAP ↑ highest priority.
RST7.5
RST6.5
RST5.5
INTR ↓ Lowest priority

TRAP has highest priority & INTR has lowest priority.

Ex:- HOLD signal has highest priority in externally initiated signal

- TRAP interrupt independent of EI & DI interrupt.
- maskable & non maskable concept valid only when interrupt enable FF is set.

when a interrupt is acknowledge following step execute automatically.

- execution of current inst will complete 1st.
- Add of next inst will store at top of stack.
- DI inst will execute automatically.
- execution will transfer at interrupt service rotien.

Ex- programmer should have to write last inst of interrupt service routine EI.

- minimum time duration for which a interrupt should

SIM (Set interrupt mask).

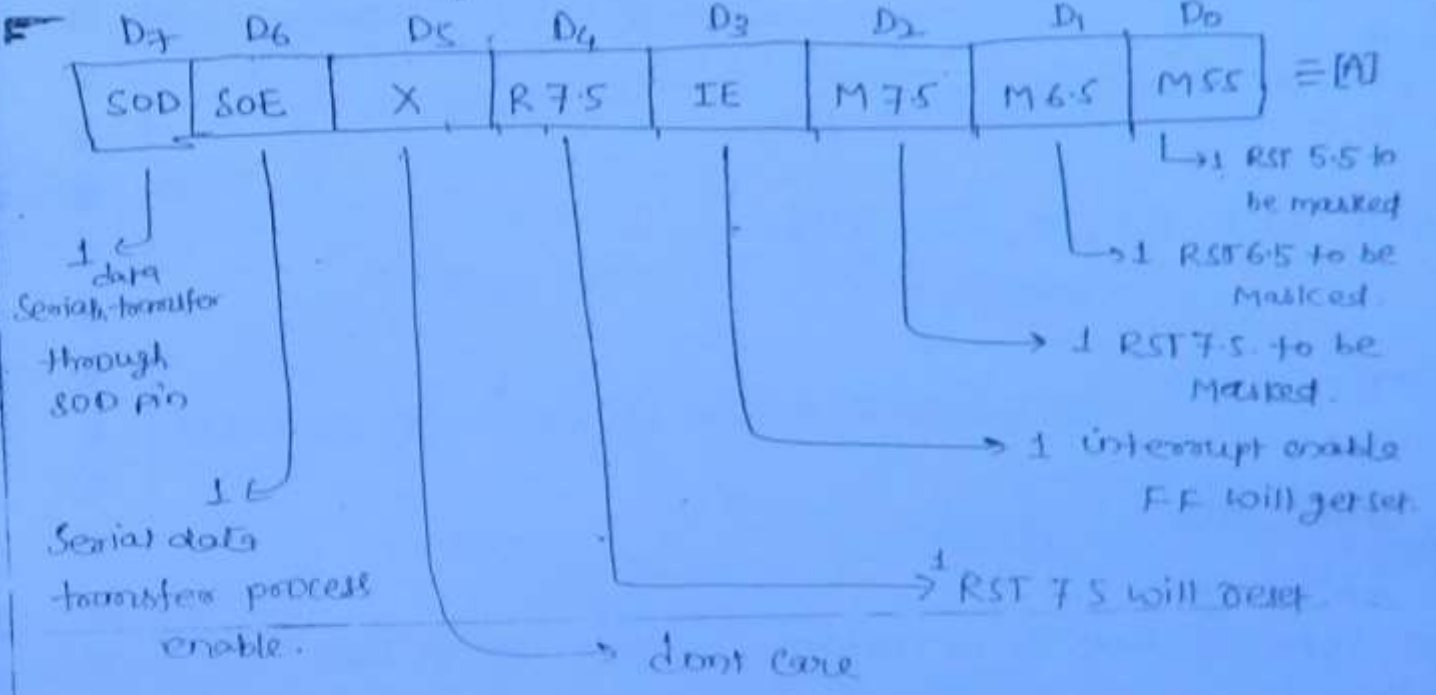
→ by using this inst., interrupt can be masked.

→ Serial transfer of data through SOD pin. (57)

SIM no operand

1 w.s - 1 byte

When this inst. execute on the bases of conten of Accum. and accroding its characteristic task will perform.



implicit & F.

** RIM (Read interrupt mask).

RIM inst is used for

(i) to find out status of pending interrupt.

(ii) to find out status of masked interrupt.

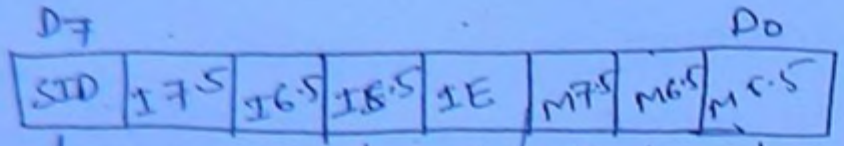
(iii) This inst. is used for Serial transfer of data through SID pin.

RIM no operand.

1 byte

When this inst will execute Accum. loaded as per functions of RIM inst.

A] ≡



1 = RST 5.5 is masked

1 = RST 6.5 — " —

1 = RST 7.5 — " —

1 = interrupt enable FF is set.

1 = RST 5.5 is pending.

1 = RST 6.5 — " —

1 = RST 7.5 — " —

1
Serial transfer of data through SID pin.

- implicit,
- RIM & SIM is machine control inst. so status of flag will not affect.

Note :-

NOP, HLT, EI, DI, RIM, SIM:

3)
```
EI
RIM
ANI 08H
SIM
```

What type of task to be perform by above set of instr.

1) Send bit out on Sop pin.
2) Accept bit in from SID pin
3) Accept RST 7.5 interrupt
4) Reset RST 7.5 interrupt.

Interfacing

Memory interfacing

I/o interfacing → Memory Mapped I/o interfacing
→ I/o Mapped I/o interfacing or peripheral mapped I/o interfacing.

General steps of interfacing :-

Commond signal connect directly b/w the interfaced device

data buses connected directly

data buses { $(MSB)_{\mu p}$ ⟷ $(MSB)_{\text{interfaced device}}$; $(LSB)_{\mu p}$ ⟷ $(LSB)_{\text{interfaced}}$ }

add. bus of interfaced device connected as

$(LSB)_{\text{interfaced device}} \longrightarrow (LSB)_{\mu p}$

$(MSB)_{\text{interfaced device}} \longrightarrow (MSB)_{\mu p}$

Remaining add. bus of μp is used for development of chip chi selection logic

→ chip selection logic develop by two way

(i) by use of logic gate or buffer ckt.

(ii) by use of decoder ckt.

(i) Memory interfacing :-

First write memory in $2^n \times q$ format

3) Chip selection by logic gate :-

(60)

$2 \times 2^{10} \times 8$

$2^{11} \times 8$

Addr = 11

data = 8.

find out add. range of memory interfaced with up.

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

$1\ 1\ 1\ 1\ 1\ A_{10} - - - - - - A_0$

min add $1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 = F800H$

Max add. $1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1 = FFFFH$

Q. $CS = \overline{A_{15}}\ A_{14}\ A_{13}$ is used as chip selection logic of 4K byte RAM in 8085 up sy

(a) $0\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0$

6000H

7FFFH

4K byte $= 2^{12} \times 8$

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

$0\ 1\ 1\ X\ A_{11}\ A_{10}\ A_9\ A_8\ A_7\ A_6\ A_5\ A_4\ A_3\ A_2\ A_1\ A_0$

- min. add.

don't x=0

X=0 → 0110 0000 0000 0000

6000H

X=1 → 0111 0000 0000 0000

7000H

max. add

x=0 → 0110 1111 1111 1111

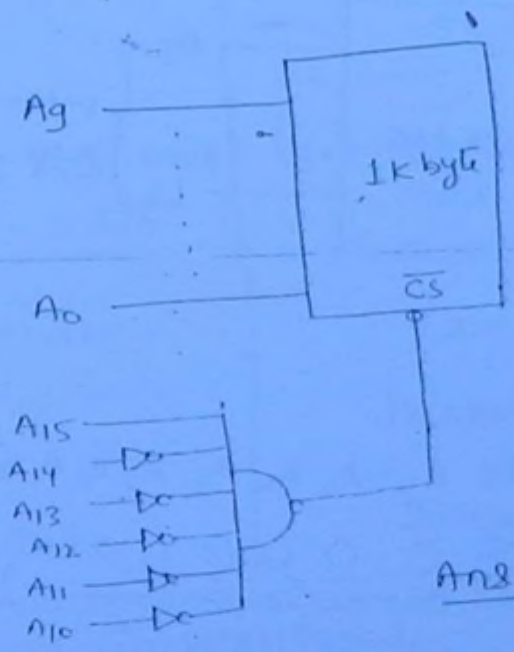
6FFFFH

X=1 → 0111 1111 1111 1111

7FFFFH

6000H – 6FFFFH to 7000H to 7FFFFH.

Q



Consider a memory chip with 1024 byte storage capacity is interfaced with 8085 µp or ~~any~~ processor with 16 add. bus as shown in fig what will be the memory interfaced range

Ans → 8000H to 83FFH.

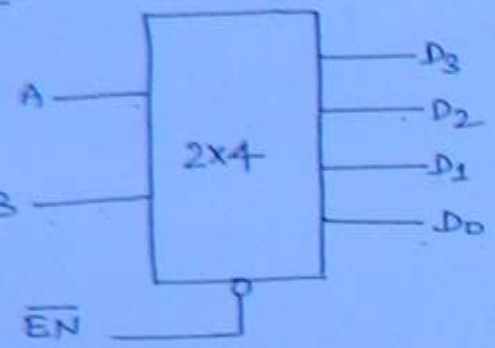
1 0 0 0 0 0 0 a_9 a_8 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0

8000H min. add.

83FFH

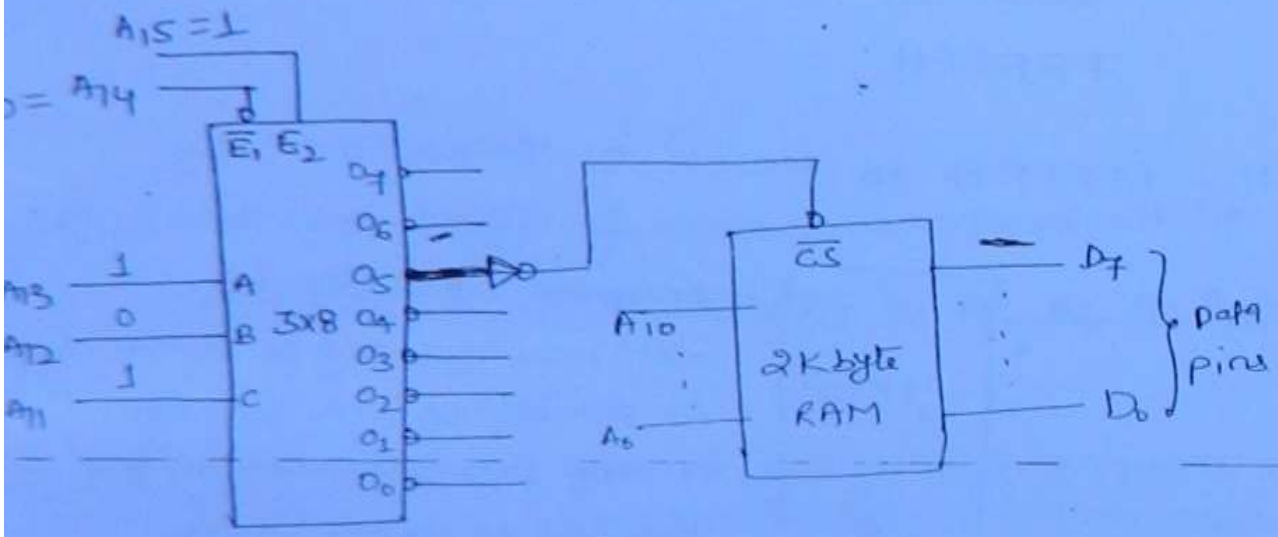
Chip selection logic by decoder ckr :-

decoder :



A	B	D_3	D_2	D_1	D_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

$f(AB) = \bar{A}\bar{B}D_0 + \bar{A}BD_1 + A\bar{B}D_2 + ABD_3$



find out memory range

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

A800H — min. add.

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

AFFFH — max. add.

A800H to AFFFH

Case①:- $f(A\,B\,C)$ = A800 - AFFFH

Case② $f(C\,B\,C\,A)$ = 1 0 1

$A_{12}=1,\ A_{13}=1,\ A_{11}=0$

I/O interfacing →

Memory mapped I/O	I/O mapped I/O interfacing	Characteristic
$\overline{M\cdot RD}$ / $\overline{M\cdot WR}$	I/O $\overline{RD}$ / I/O $\overline{WR}$	Command sig.
64K byte memory shared b/w internal memory & I/O devices.	256 i/p & 256 o/p maximum.	No. of devices interfaced
All memory related inst. Applicable. Ex MVI, LHLD	IN & OUT	instruction
Any memory Reg. and I/O	b/w I/O devices & Acc only	~~b/w I/O dev & Acc only~~ Data trans
Hardware more	Less	Hardwa
Slower	Faster	Executio
Small	large	Applicat

Ex: A & L operation perform Directly with Any Reg but I/O mapped interfacing directly not poss

Some important I/O peripherals :-

8255 = programmable peripheral interface

8279 = " " keyboard & display interfaced

8259 = " " interrupt control

8237 = DMA Controller.

8251 = USART

8155 = programmable I/O port & timer etc.

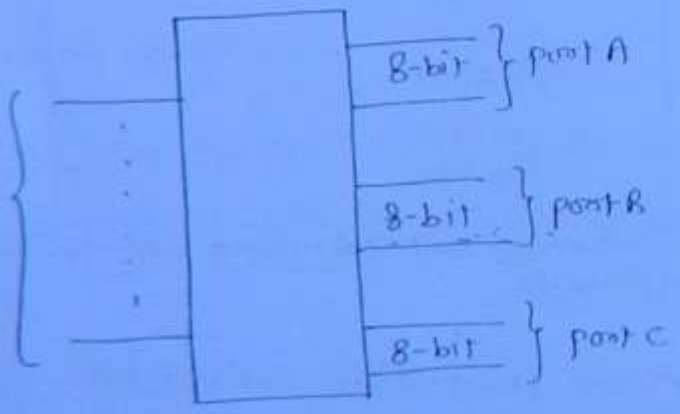
8254/8253 → programmable interval timer.

PSU · Some important digital IC.

- 74182 → look ahead carry generator.
- 74180 → 8 bit parity generator & checker.
- ** 7477 → seven sigment display decoder.
- 7493 → 4 bit binary counter
- 7490 → decade counter.

- 7400 → Quad 2 I/p NAND gate.
- 7402 → Quad 2 I/p NOR.
- 7408 → —"— AND
- 7432 → —"— OR
- 7486 → —"— Ex-OR.

- it is 40 pin IC.
- ~~i/p port add is 8 bit~~



it has three port, port A, B, C.

All three port can be used

- port C can also works as in two port as port C' upper and port C' lower having port add of 4 bit (lower nibble & upper nible)

~~port C upper & port C lower~~ port C lower can also generate control

- it has two mode of operation
  - ↳ bit set Reset mode
  - ↳ I/o mode.

I/o mode

mode '0' General mode
mode '1' Handsacking mode
mode '2' Bidirectional mode.

mode '0' –

In this mode of operation port 'A', Port 'B', port 'C' can work as i/p o/p port individual.

Mode '1'

In this mode of operation port 'A' & port 'B' will work in handsacking mode.
port 'C' upper & port 'c' lower will generate control signal for port A & port B respectively.

mode '2' :-

In this mode of operation port 'A' works in bidirectional mode only & port C Upper generate control signal for it.

Q 22
P-147

A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
1	1	1	1	1	X	0	0	F8-FFH
						1	1	

X = 0
F8 → min add { for 00
FB → for 11

X = 1
FC → { for 00
FF → { for 11

F8 – FB → FC – FF = F8-FFH

Write a ALP of 8085 to transfer 10 byte data that store a memory location starting from 2011H to 3011 as

2011H ⟶ 3011H
⋮
201AH ⟶ 301AH

Soln

```
      LXI B 2011H ;
      LXI D 3011H ;
      MVI H OAD ;
Loop: LDAX B.
      STAX D
      INX B
      INX D
      DCR H
      JNZ loop.
      HLT
```

Q: write a 8085 ALP to add a 16 bit no in locations 5000H (higher byte) and 5001 c (lower byte) with another 16 bit no store in 5002 (higher byte) & 5003 (lower byte) & find result store in BC.

```
LDA 5000H
mov H, A
LDA 5002H
mov L, A
LDA 5002H
mov B, A
LDA 5003H
DAD B
LXI SP 1234H
PUSH H
POP B
```

Introduction of 8086

Comparision b/w 8085 & 8086

8085	8086
it is 40 pin IC	40 pin IC
f = 3 MHZ	f = 5 MHZ
Vcc = +5V	Vcc = +5
Based upon NMOS	Based of HMOS tech.
8-bit up	16-bit up.
Add. line – 16	Add. line 20
data line – 8	data line 16
Max. memory location that can interfaced = 2^{16}	max. memory location that can interfaced = 2^{20}
Size of flag register = 8 bit	Size of flag register = 16 bit
no of flag = 5	no of flag = 9
normal memory concept are used	Memory sigmentation concept is used.

8086-I → 8 MHz
8086-II → 10 MHz

ARCHITECTURE

Internal Arch. of 8086 divide in two parts

- Bus interface unit (BIU)
- Execution unit (EU).

Bus interface unit (BIU) :-

this part manage all type of data/add flow over bus in BIU, a queue of six location is exist & during the execution of one inst in execution unit, next inst are fetch.

Queue based upon first i/p first o/p (FIFO). (68)

All register that available in BIU size of 16 bit

EU (Execution unit)

- All type of Arith. & Logic unit present in this part.
- All ~~type~~ of data execution will perform in EU.
- Some Reg. present in EU can also work as 8 bit & 16 bit both type.

AX ≡ 16 bit reg.
AH = 8 bit
AL = 8 bit
AH – AL = AX

Note → i/p o/p port add in 8086 of 16 bit.

- max. no. of i/p & o/p devices can be connected = 2^{16}.
- max. memory that can interfaced = 2^{20} byte
  = 1 M byte

& 16 bit data will store in two continuous memory location.

- in 8086 up can read two byte data from memory either one m.cy or in two m.cy

if 1st byte at even add then up required only one M.cy to read 16-bit data from memory.

if 1st byte at Odd add then up required two m.cy to read 16 bit data from memory

But in case of 8088 to read 16 bit data from memory always two m.cy required to read data from

but it process 16 bit data at a time so it is also called externally 8 bit & internally 16 bit µp.

Memory Segmentation:

- 1 Mbyte memory divide in 4 segment size of 64 Kbyte in continous memory range.
- Segments can be define in anywhere inside the 1 Mbyte memory.
- every code segment is used for definit type of information storage.
  - Code segment → (feed) inst. set
  - Stack Segment → Temporary storage of information during the execution of main prog.
  - data segment
  - Extra segment } data storage
- Regarding every segment there is 16 bit reg. that will hold top 16 bit bottom add of that particular segment
- Bottom add. of every segment select in such a way that last four bit always zero.

Code Segment	CS	16 bit reg.
Stack Segment	SS	
Data Segment	DS	
Extra Segment	ES	

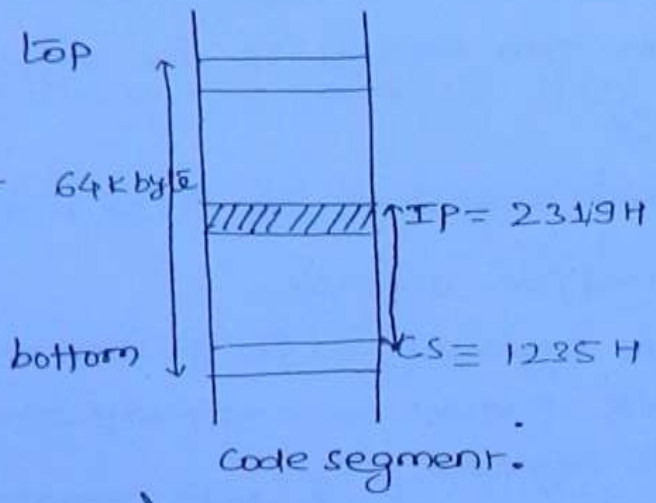
- Regarding each segment another 16 bit reg that will hold 16 bit add. of memory range 0000H to FFFFH (64 Kbyte Range)

& content of this reg. is also known as effective add.

Code segment → IP (instruction pointer)
Stack segment → SP (stack pointer)
Data segment → SI (source index)
Extra segment → DI (Destination index)

Physical Add.

IP of 8086 ep is ~~ruff~~
Analogous to program counter of 8085.



CS : IP
offset

CS ≡ 12350 ← hardwired zero
IP = 2319

physical add. = 14669.

~~physical add~~ also known as, ~~memory add~~ or memory location add.

Flag Register :-

Size - 16 bit

- in 8086, 9 flag are define that can be divide two groups.

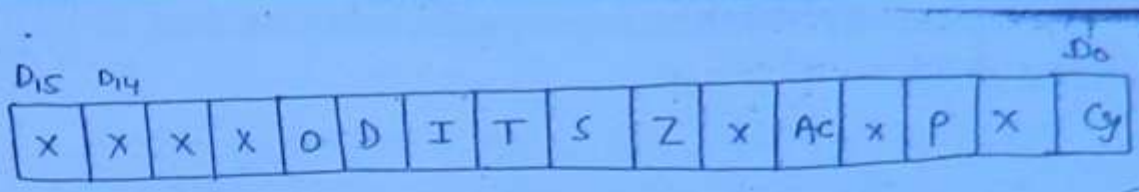
(1) → Conditional flag → Carry flag, Auxillary Carry flag, Sign flag, Parity flag, Over flow flag, Zero flag.

Note - All 5 flag of 8085 is exactly same as 8086

→ process control flag / machine control flag = 3

Trap flag, interrupt process control flag

Direction flag



71

Cy - Carry flag.
P - Parity flag.
Ac - Auxillary Carry flag.
Z - zero flag
S - Sign flag
T - Trap flag.
I - interrupt process control.
D - String Direction.
O - Over flow.

Addressing mode :-
- form of effective add given in the inst. is known as add. mode.

- Reg. addressing mode → if effective add of data given in the form of Register as operand.

Note - Reg add mode is also indirect add. mode

Ex: MOV A [BX]

Immediate add mode :- If data itself given in the inst.

Direct add mode :- If effective add itself given in the inst.

** Register Relative add mode :- effective add of data can be produced by adding 8 bit or 16 bit displacement no.

[BX]
[DI]
[SI]
} + 8 or 16 bit displacement no.

** Base index add:-

$$EA = \begin{Bmatrix} [BX] + [DI] \\ [SI] \end{Bmatrix}$$

Relative Base index add.:-

$$EA = \begin{Bmatrix} [BX] + \begin{matrix}[DI] \\ \text{or} \\ [SI]\end{matrix} \end{Bmatrix} + \text{8 bit or 16 bit displacement no.}$$

GATE

(A) B = 87H
A = 87H
A = 00H S = 0

```
A = 0000 0000
B = 1000 0111
-------------
A = 1000 0111   S = 1
    8    7
```

87H

(B) B = 0AH
C = 0BH

```
    MVI A 00H
loop ADD B
     DCR C
     JNZ loop
     HLT
```

(C) 0100 LXI SP 00FF
0103 LXI H 0107
0106 MVI A 20H
SUB M HL = 0107 = 40H

[A] = 20H = 00100000

ARL Inst

(1) affect all flag → ADD, ADI, SUB, SUI, ADC, ACI, SBB, SBI, DAA, CMP, CPI, ANA, ANI, ORA, ORI, XRA, XRI.

(2) affect only 4 flag → INR & DCR.
Cy will not affect

(3) affect only carry flag → STC, CMC, DAD, Logical Rotation Inst.

(4) not affect any flag → CMA, INA, DCX.

Ex

```
        ORG 7000H
7000H BEGIN  LXI H, 7000H
             MOV A, L
             ADD H
             JP END
             RST 0
END          PCHL
             HLT
```

[A] = 00H
[A] = 70H
[A] = 70H = 01110000

The End