☆ I/O Interfacing:

① Memory mapped I/O ⇒ I/O devices are treated as Memory.

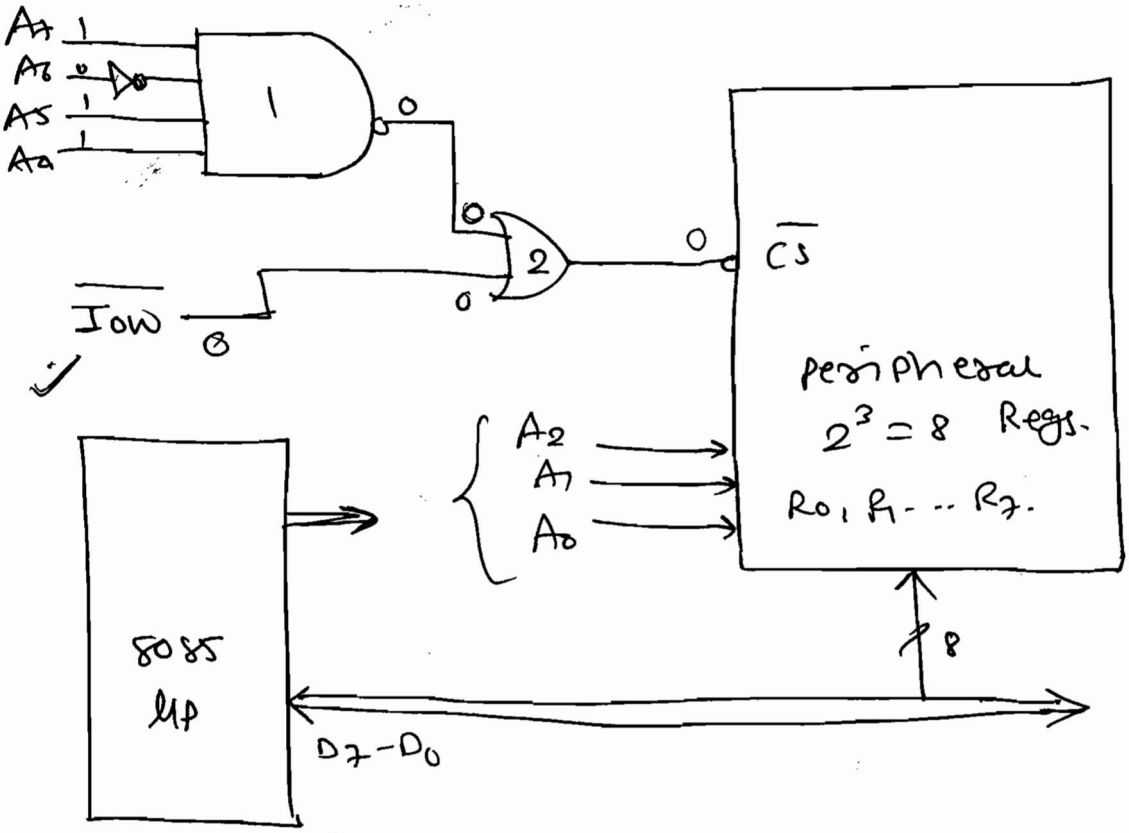
② I/O mapped I/O ⇒ mem, I/O.

	mem mapped I/O		I/O mapped I/O	
	mem	I/O	mem	I/O
① No. of Addr lines	16	16	16	8 (A_7-A_0)
② Control signals	$\overline{MEMR}$, $\overline{MEMW}$,	$\overline{IOR}$, $\overline{IOW}$,	$\overline{MEMR}$ $\overline{MEMW}$	$\overline{IOR}$, $\overline{IOW}$
③ No. of peripherals	* (i) — (ii) 60 KB	64 KB 4 KB	64KB + 2^8 = 256 I/O devices	

*

	mem mapped I/O	I/O mapped I/O.
Advantages	1) IO/$\overline{M}$ pin is not required. 2) No separate instn for Io devices. 3) Arithmatic and Logic operations are directly performed on I/O device data.	1) Max capacity of µP is utilized. 2) Less Hardware Complexity for I/O interfacing.
Disadvantages:	1) more Hardware Complexity for I/O device interfacing	(1) Separate Instn for I/O device are required. (2) Can't Perform Arith Logical operations

Ex-1 A peripheral is interfaced to the 8085 µP as shown in the following figure. Determine (i) The mode of interfacing (ii) No. of internal Registers in the peripheral and their addresses.



Ans: 1) I/O mapped I/O Mode

(∵ only 8 addr lines used and Control signal is $\overline{IOW}$).

(2) A_5 A_2, A_1, A_0 are Connected directly to the peripheral, no of Internal Registers

$= 2^3 = \boxed{8}$

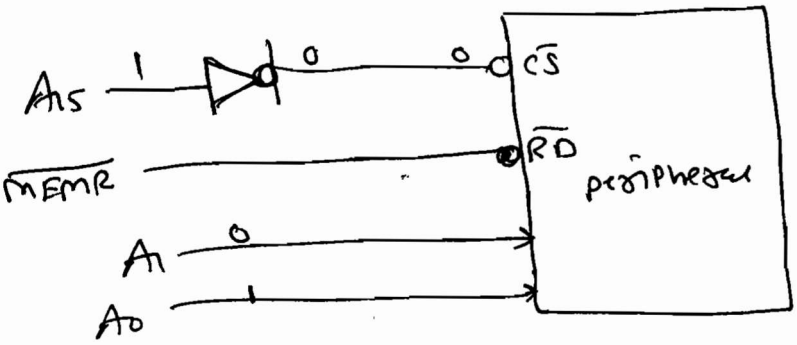
~~A7 A6 A5 A4 A3 A2 A1 A0~~

	A_7	A_6	A_5	A_4	A_3 (Unused)	A_2	A_1	A_0	X=0	X=1
R_0	0	0	1	1	X	0	0	0	$B0_H$	$B8_H$
R_1	1	0	1	1	X	0	0	1	$B1_H$	$B9_H$
⋮	⋮					⋮			⋮	⋮
R_7	1	0	1	1	X	1	1	1	$B7_H$	BF_H

→ Address range: $B0_H$ – BF_H.

⇒ In IO mapped IO mode the value of higher order address mode A_{15} - A_8 is same as the value of Lower order add. Bus. (A_7 - A_0).

Ex: Determine the max. addr. of Register B in the following peripheral:



Given ⇒

A_1	A_0	Register
0	0	A
0	1	B
1	0	C
1	1	D

Ans: Memory mapped I10.

→ For Register B.

A_{15}	A_{14} ---- ..	A_2 A_1 A_0
1	x xx xxxx xxxx	x x 0 1.

→ Max Add. for Reg. 'B' is when all x's = 1.

1111 1111 1111 1101.

$= (FFFD)_H$.

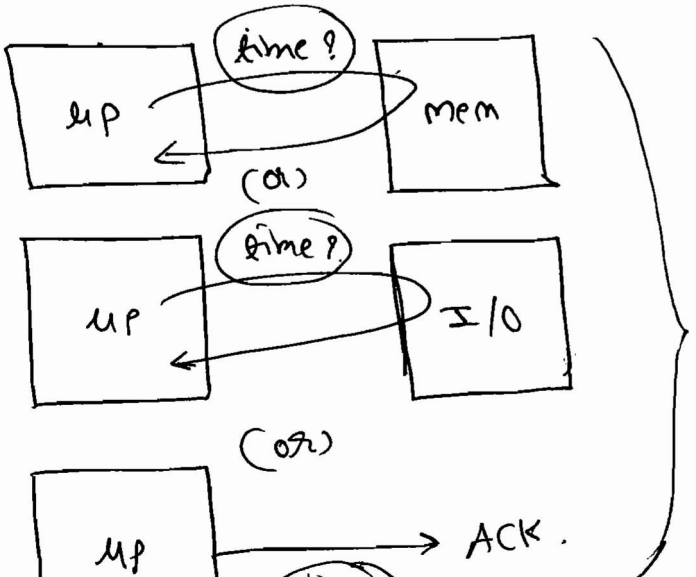
(1) Instruction cycle:

→ It is the time required to execute an instruction

→ Range: 1 MIC to 5 MIC.

(2) Machine cycle:

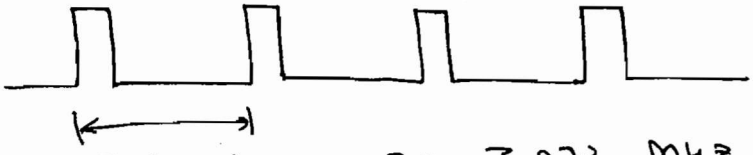
→ It is the time required to Complete one operation of excessing memory, excessing Ilo device or sending an acknoledgement.



3-T to 6-T state.

(3) T State:

→ It is the task performed in one clock period.



* Types of Machine cycle:

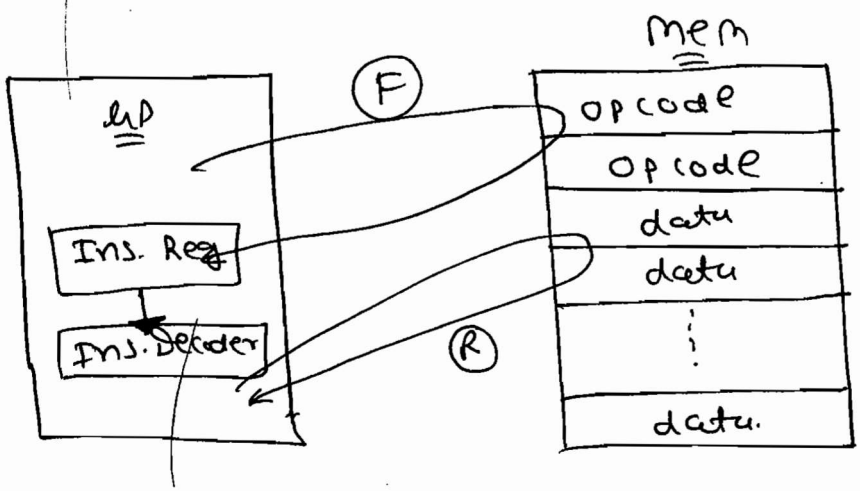
1) Opcode fetch MIC (F) ⇒ (4T) = 3T + (1T)

2) mem. Read MIC (R).
3) mem. Write MIC (W).
4) I/O Read MIC (I)
5) I/O - Write MIC (O).

(3T)

(1T) ⇒ Time to decode the opcode.

6) Int. Ack MIC.
7) Hold Ack MIC.



⇒ Special MIC:

(1) S = opcode Fetch MIC (6T).

*

ALP (Low Level)
(Assembly Language Program)

8085 µp
Hand coding

8085 µp
Assembler

MIC Language

HLL (C, C++, Java).

Compiler
Interpreter

MIC language

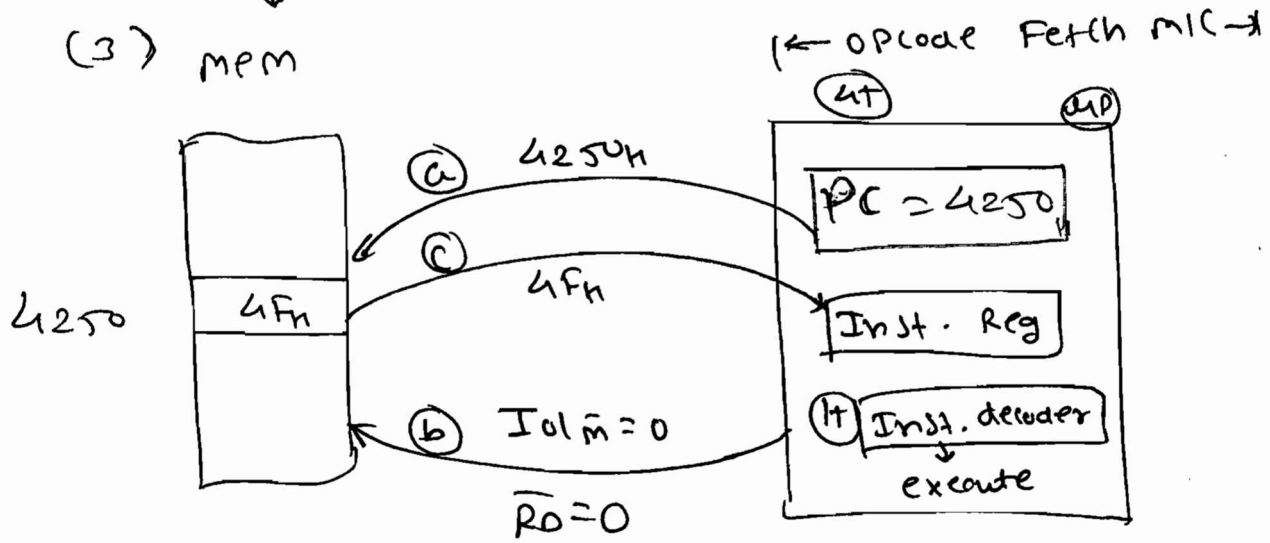
Eg:

(1) MOV C, A ⇒ Ass. lang instruction. (mnemonic).

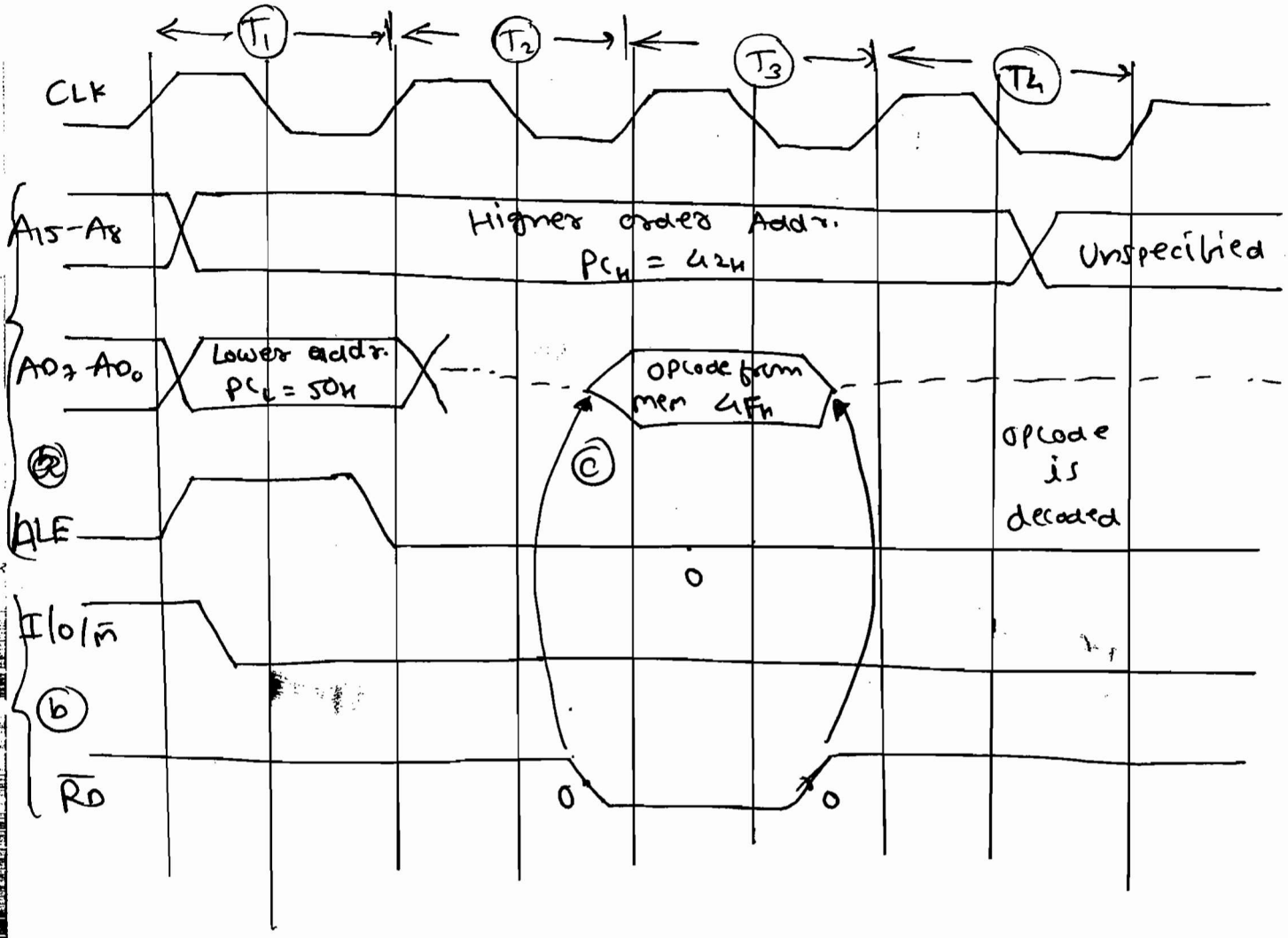
Hand coding

(2) Opcode = 0100 1111₂.
= 4FH.

(3) MEM



* Timing Diagram of Opcode Fetch M1C:



→ $PC_{16} = PC_H + PC_L = 4250H$

→ In T_2 State of any machine cycle the PC is incremented

NOTE: In the opcode fetch machine cycle if T_4 State is removed then it converts into the timing diagram of Memory read M1C cycle.

* Types of Instructions (Based on size):-

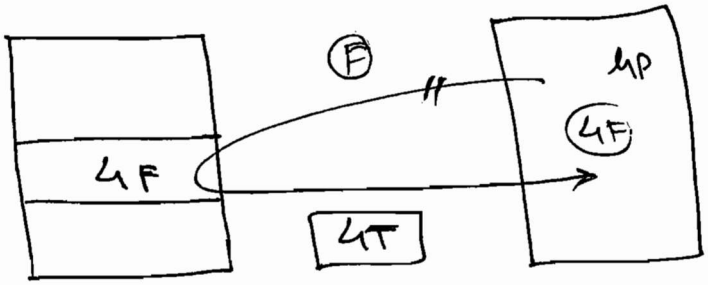
① 1- Byte Instructions.

② 2- Byte Instructions.

③ 3- Byte Instructions.

⇒ ① 1- Byte Instruction:

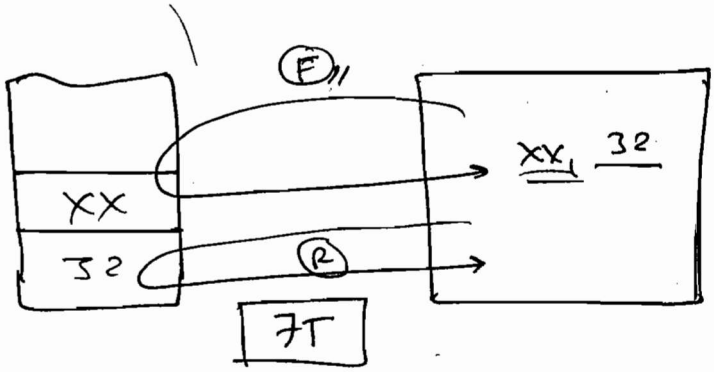
e.g MOV C, A



⇒ ② 2- Byte Instruction:

⇒ MVI C, 32 ⇒

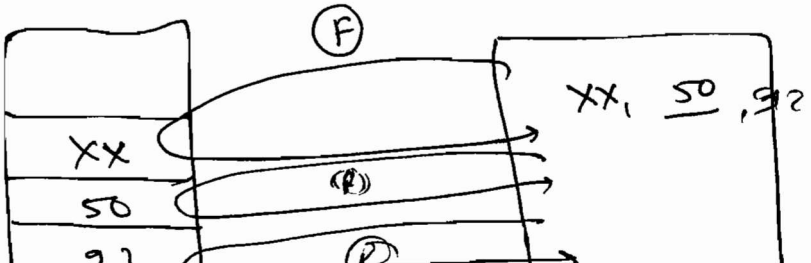
Opcode = XX



③ 3- Byte Instruction:

e.g. LDA 9250

Opcode: XX



→ In Any instruction cycle the first machine cycle is opcode fetch machine cycle.

Eg.

① XRI 46_H ⇒ 2B

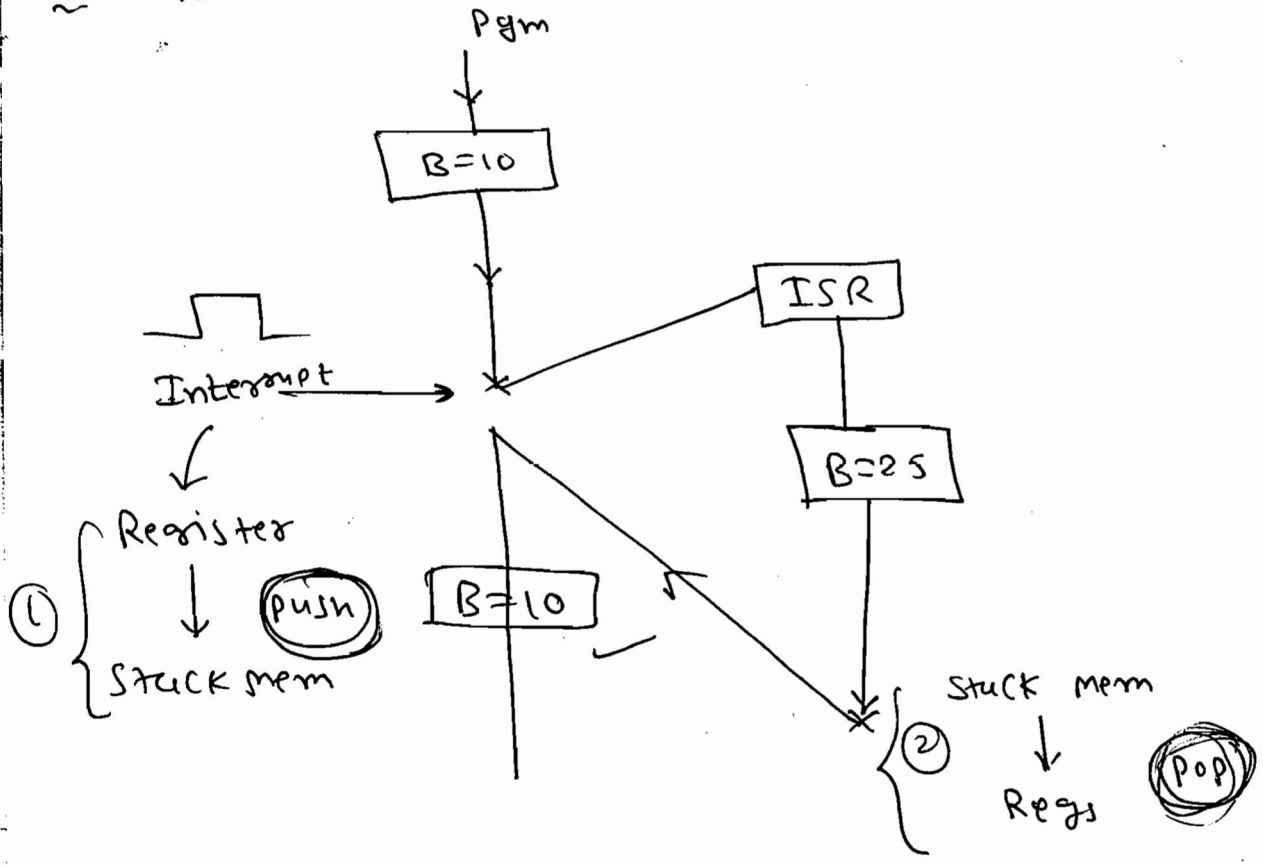
② Push PSW ⇒ 1B.

③ LDAX B ⇒ 1B.

④ LHLD 8300_H ⇒ 3B.

⑤ DAD SP ⇒ 1B.

* Stack:



⇒ SP_{16} = Stack Pointer$_{16}$:

∴ LIFO = Last in First out.

① Push RP

- Push B
- Push D
- Push H
- Push PSW

Predecrement

⇒ decrement SP + push RP_H

⇒ decrement SP + push RP_L

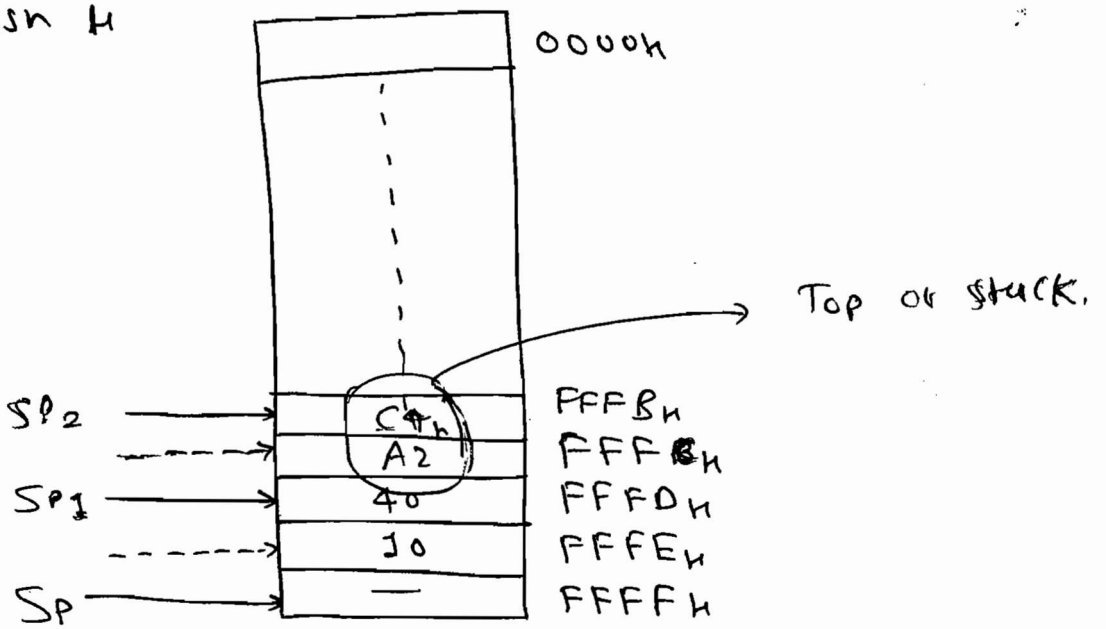
$RP_{16} = RP_H + RP_L$

Eg. Given $SP = FFFF_H$; $BC = 1040_H$; $HL = A2C4_H$.

① push B ② push H.

Ans: ① push B.

② push H

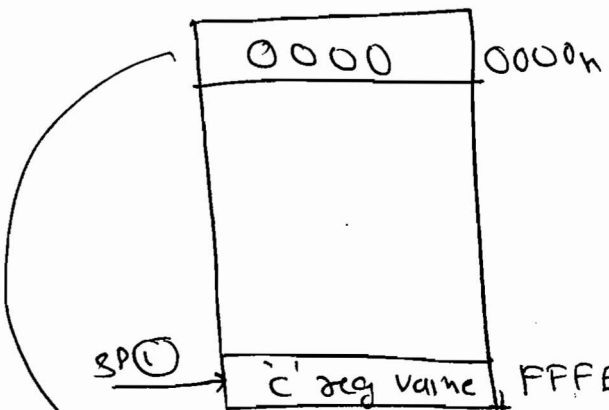


P1) SP = 0000H.

① Push B.

BC, regs ⟶ mem Locations = ?

→ 0000
− 1
FFFF
1



② POP RP

→ Post increment

POP B
POP D ⇒ get 1 Byte from Stack into RP_L + Increment SP.
POP H

POP PSW ⇒ get 1 Byte from Stack into RP_H + Increment SP.

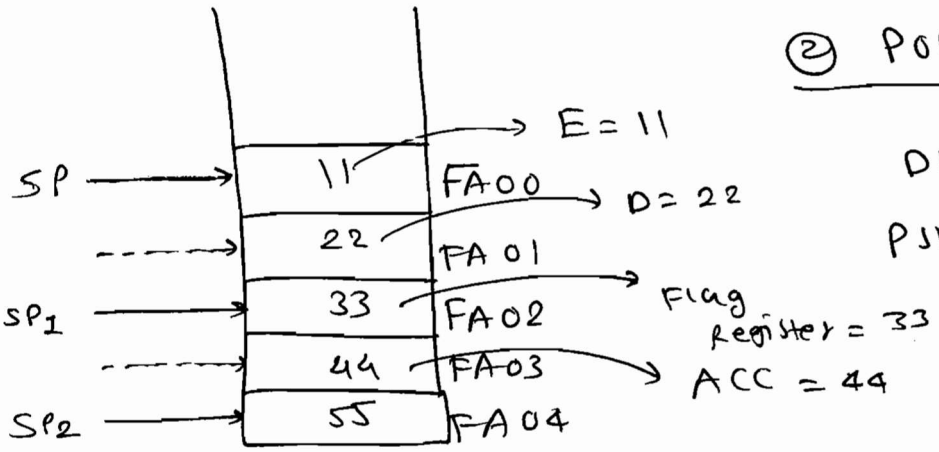
Eg. Given

① POP D

② POP PSW

DE = ?.

PSW = ?.



$DE = 2211_H$

$PSW = 4433_H$.

P2) Let SP = FFFF ; BC = 5065_H ; HL = $A1F4_H$

① PUSH B → LIFO

② PUSH H

③ POP D

④ POP PSW

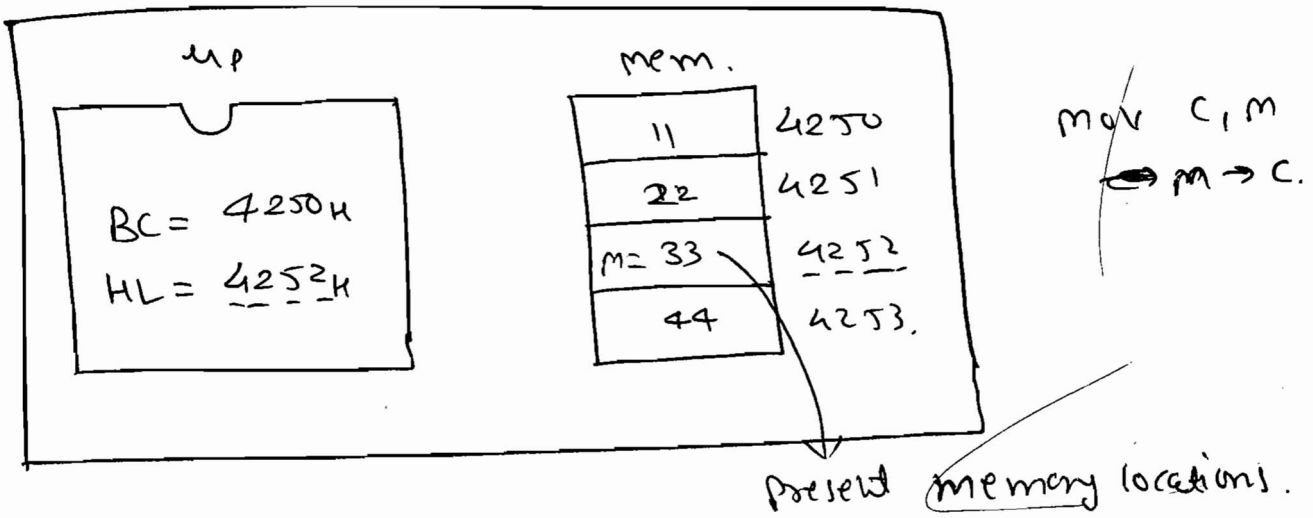
HL → DE ⇒ DE = $A1F4_H$.

BC → PSW ⇒ PSW = 5065_H.

* Types of Instructions:

1) Arithmatic
2) Logical
3) Data Transfer
4) Branching
5) Machine related, I/O.
6) Additional.

* Reference:



(4250) = 11H.

(4253) = 44H

((BC))

⇓

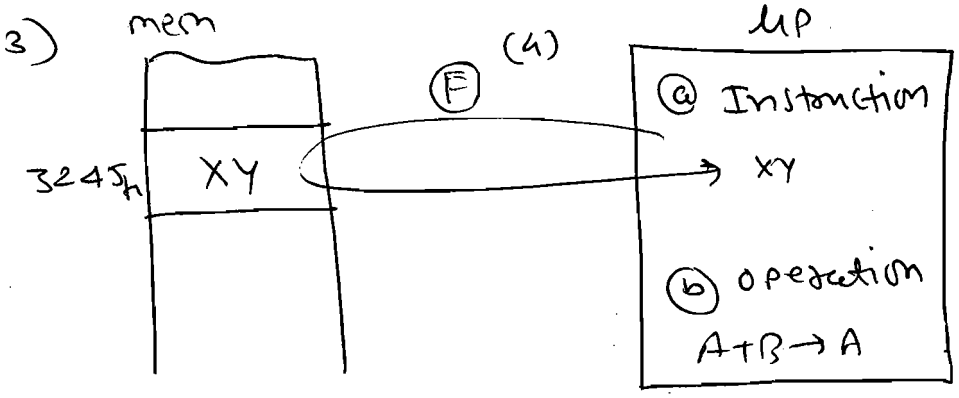
(4250) = 11H.

M = ((HL)) = 33H.

① Arithmatic Instruction:

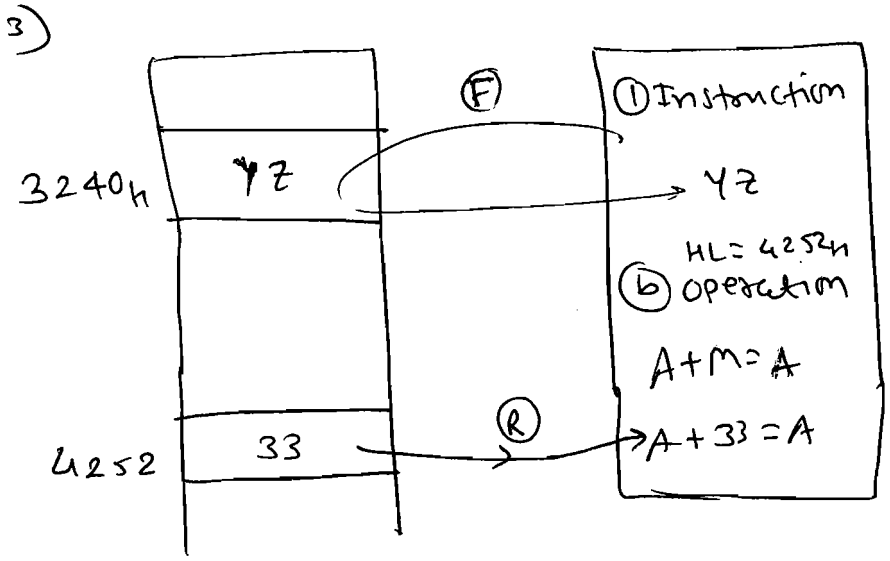
Sr. NO.	Instruction	operation	B|M|T	Flags affected
1)	ADD R	A+R → A	1|1|4 (F)	All

E.g. 1) ADD B ⇒ (2) opcode = XY



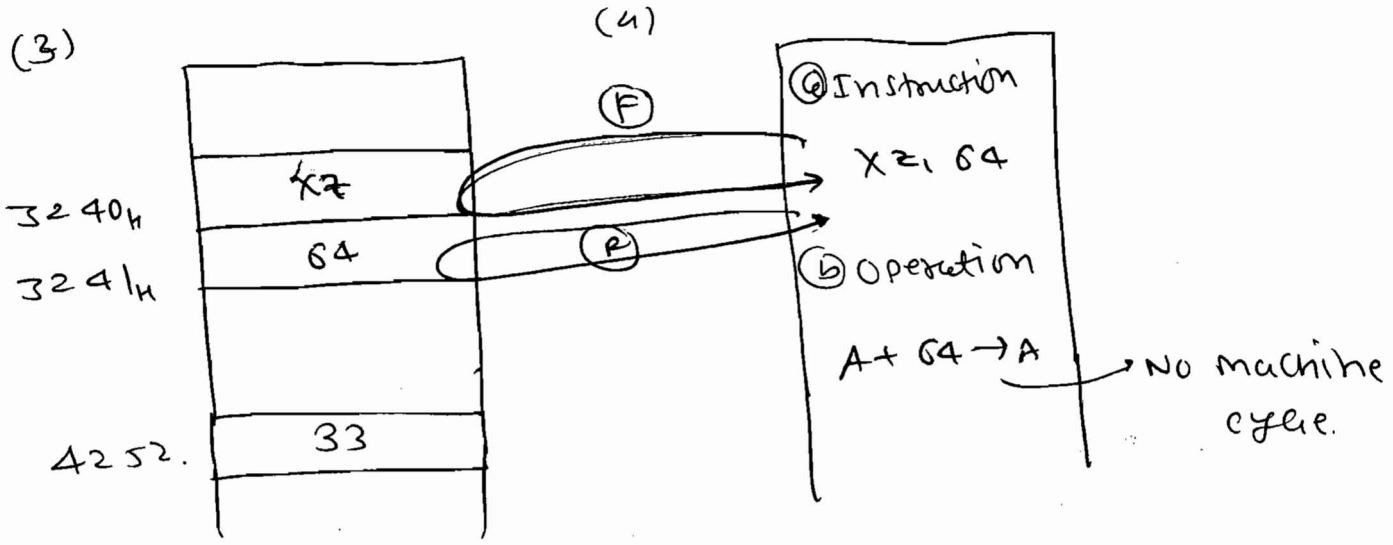
2)	ADD M	A+M → A	1|2|7 (F.R)	All.

E.g 1) ADD M ⇒ (2) opcode = XY.



3) ADI 8 bit data A + 8 bit data → A 2|2|7 (F.R) All

Eg. (1) ADI 64_H ⇒ (2) Opcode = XZ, 64_H.



2) (1) SUB R A − R → A 1|2|7 (F.R.) All.

(2) SUB M A − M → A

(3) SUI 8 bit data A − 8 bit data → A

3) (1) ADC R A + R + CY → A 1|1|4 (F) All.

ADC M A + M + CY → A 1|2|7 (F.R.)

ACI 8 bit data A + 8 bit data + CY → A 2|2|7 (F.R.)

4) SBB R A − R − CY → A 1|2|7 (F.R.) All.

SBB M A − M − CY → A

SBI 8 bit data A − 8 bit data − CY → A

5) INR R R + 1 → R 1|1|4 (F)

INR M M + 1 → M 1|3|10T (CF, RW).

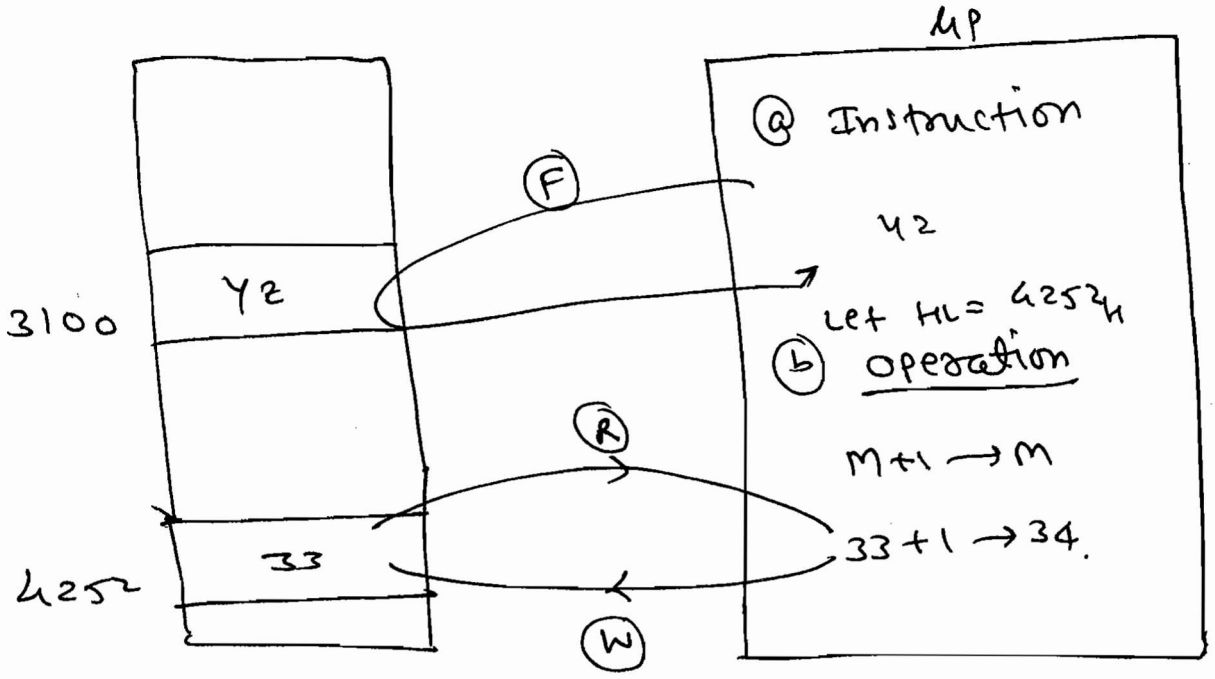
DCR R R − 1 → R 1|1|4.

S, Z, P, AC affected. but CY not

1) INR M 2) opcode = YZ.

3) Memory.

4)



6) INX R_p $R_p + 1 \rightarrow R_p$ 1 | 1 | 6 (5)

DCX R_p $R_p - 1 \rightarrow R_p$ 1 | 1 | 6 (5)

No flags are affected [∵ operation doesn't take place in ALU].

INX B
INX D
INX H
INX SP

7) DAD RP HL + RP → HL 1 | 3 | 10 (FBB)

* only carry flag if carry occurs out of 16 bits.

DAD B
DAD D
DAD H
DAD SP

ADC (3T)

H + B + CY

L ← ADD (3T)

C → ACC

H L

(R) Logical operations:

Sr. No.	Instruction	operation	B|M|T	Flags affected.
1) a)	ORA R	A ∨ R → A	1|1|4	* CY=0 S, Z, P are affected.
	ORA M (1)	A ∨ M → A	1|2|7 (FR)	
	ORI 8bit data (1)F (2)R	A ∨ 8bit data → A	2|2|7 (F.R.)	
b)	XRA R	A ⊕ R → A	1|1|4	
	XRA M	A ⊕ M → A	1|2|7 (FR)	
	XRI 8bit data	A ⊕ 8bit data → A	2|2|7 (F.R.)	
c)	ANA R	A ∧ R → A	1|1|4	
	ANA M	A ∧ M → A	1|2|7 (FR)	
	ANI 8bit data	A ∧ 8bit data → A	2|2|7 (FR)	

2) ~~CMR~~

CMP R

A - R 1|1|4

A < R ⇒ CY=1, Z=0

A = R ⇒ CY=0, Z=1

A > R ⇒ CY=0, Z=0.

* Register values are unchanged. only flags are affected.

S, P, AC are affected.

CMP M A - M 1|2|7 A<M ⇒ CY=1, Z=0

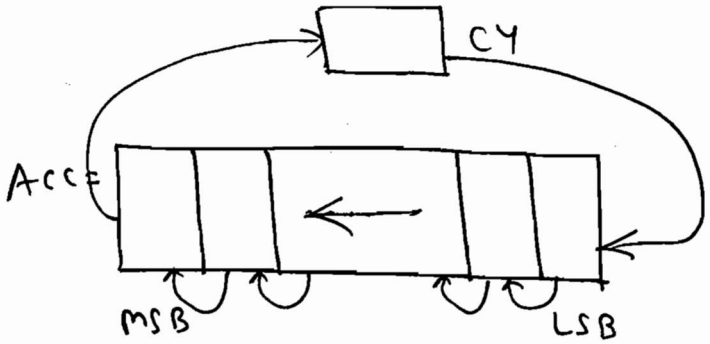
CPI 8 bit data — A - 8bit data — 2 | 2 | 7

A < 8-bit data ⇒ CY=1, Z=0

A = 8-bit data ⇒ CY=0, Z=1

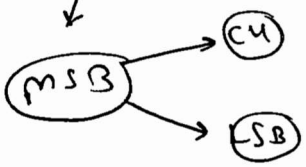
A > 8 bit data ⇒ CY=0, Z=0

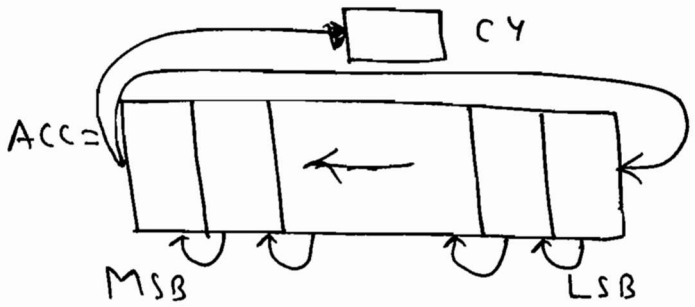
3) RAL (with CY)



only carry flag is affected.

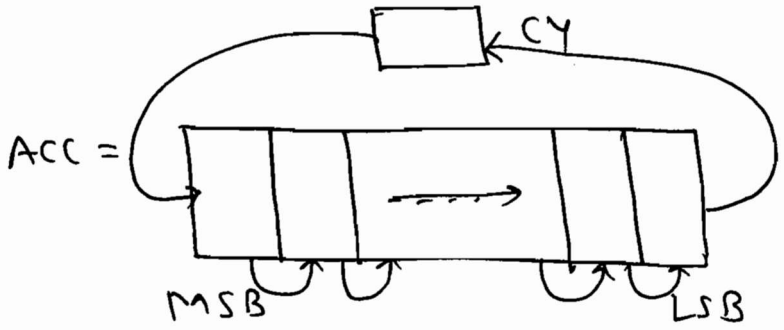
RLC (without carry).





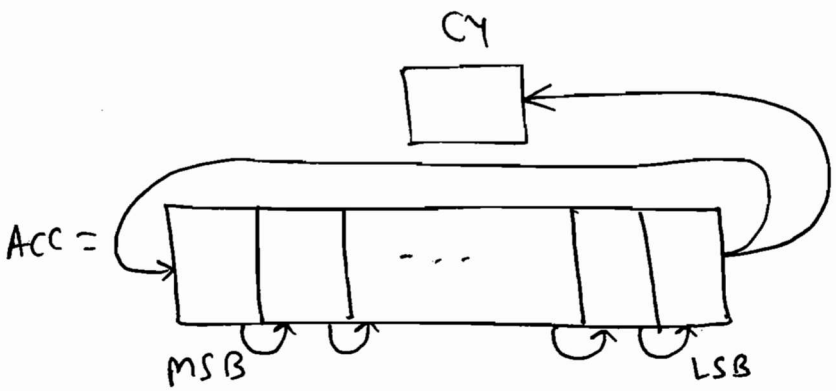
1 | 1 | 4

4) RAR with carry



only 'CY' flag is affected.

RRC without carry.



2 | 1 | 4

4) CMA $\quad \bar{A} \rightarrow A$ $\quad$ 1|1|4 $\quad$ No flags are affected (∵ operation is within Acc.).

✓ CMC $\quad \overline{CY} \rightarrow CY$ $\quad$ 1|1|4 } only 'CY' flag.

✓ STC $\quad CY = 1$

Ex-1

let, $A = F2_H$.

$CY = 1$.

XRA A $\Longrightarrow A \oplus A \rightarrow A$.

$F2_H \oplus F_2 \rightarrow 00_H$.

Ans: $A = 00_H$

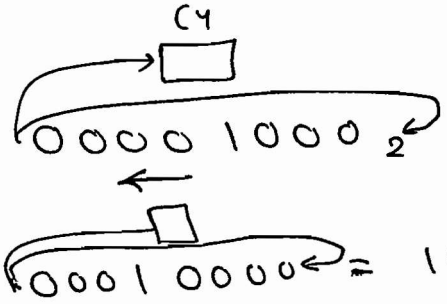
$CY = 0$.

Ex-2 Let $A = 08_H$

1) RLC

3) RLC

Ans: $A =$



$= 00010000 = 10_H$

$A = 10_H = 16_{10}$.

$A = 00100000 = 20_H$.

Ex-3 Let. $A = 3C_H$. ; All flags are cleared.

CPI 7F. $\quad S = ? \quad CY = ?$

Ans: CPI $7F_H$.

$\therefore \rightarrow A - 7F.$

$= 3C_H - 7F_H.$

$= +bd_H.$

$\therefore A = 3C_H.$

$S = 1$

$Z = 0$

$P = 1$

$CY = 1$

$AC = 1$

$$\begin{array}{r} 12 \\ 2\ 1C \\ 3C \\ -\ 7F \\ \hline BD_H \end{array}$$

$= 10111101$

Ex-4

→ In a MP XOR (P,Q) is defined as $P \oplus Q \Rightarrow P$

what is the fn of the following program.

XOR (r_2, r_1). → $r_2 = r_2 \oplus r_1$.

XOR (r_1, r_2) → $r_1 = r_1 \oplus r_2$. $\Rightarrow r_1 = r_1 \oplus r_2 \oplus r_1 = 0 \oplus r_2 = r_2$

$r_1 \leftrightarrow r_2$

XOR (r_2, r_1) → $r_2 = r_2 \oplus r_1$.

$\Downarrow$

$r_2 = r_2 \oplus r_1 \oplus r_2.$

$r_2 = 0 \oplus r_1$

$\boxed{r_2 = r_1}$

Ans: Swap.

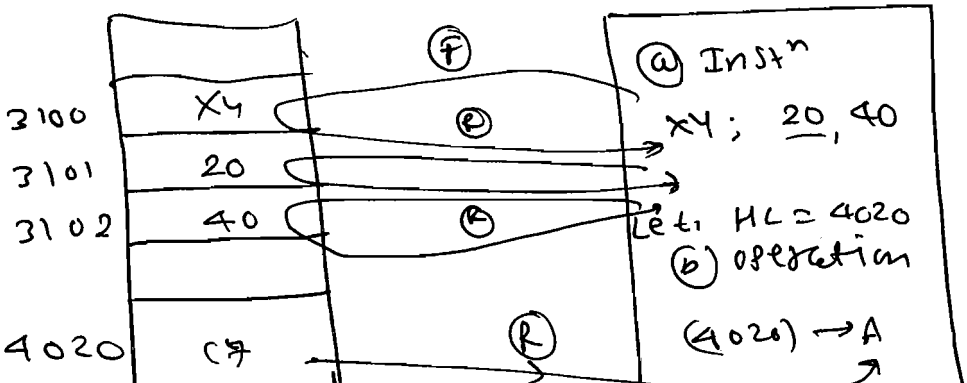
$r_2 \longleftrightarrow r_1$

(3) Data transfer Instruction:

Sr. No.	Instruction	operation	B/M/T.
1)	MOV Rd, Rs	Rs ⟶ Rd (copied)	1/1/4.
	MOV R, M	M ⟶ R	1/2/7 (FR)
	MOV M, R	R ⟶ M	1/2/7 (FR).
	MVI R, 8bit data	8 bit data ⟶ R	2/2/7 (FR)
	MVI M, 8 bit data	8 bit data ⟶ M	2/3/10 (FRW).
	LXI R_P, 16 bit data	16 bit data ⟶ R_P	3/3/10 (FRR).
2)	LDA 16 bit address	(16 bit addr) ⟶ A	3/4/13. (F.R.RR).

1) LDA 4020 2) opcode = X4, 20, 40.

3) Memory 4)



	Address
M_1	PC = 3100H
M_2	PC = 3101H.
M_3	PC = 3102H
M_4	PC = 4020H

2) STA 16 bit address A → (16 bit address)
↳ ④ write

3 | 4 | 13.
(F.R.R.W)

3) LDAX Rp ((Rp)) → A.

1 | 2 | 7T
(F.R.)

e.g. LDAX B

((BC)) → A.

(4250) → A.

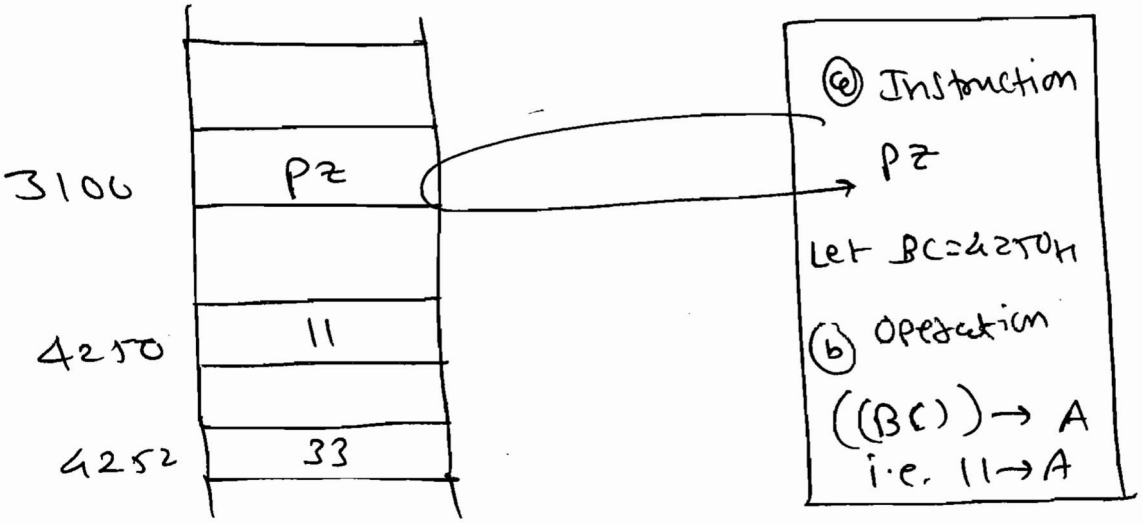
11 → A

1) LDAX B

2) Opcode = Pz.

3) Memory

4) MP



→ LDAX B.

LDAX D.

~~LDAX H~~ ⇒ ((HL)) → A.

⇒ i.e. M → A = MOV A, M.

⇒ STAX Rp A → ((Rp)) 1 | 2 | 7T (FW).

④ LHLD 16 bit address

(16 bit addr) → L ④

(16 bit addr +1) → H

⑤ 3|5|16 (FRR RR)

e.g.

LHLD 4251

(4251) → L i.e. L = 22.

(4251+1 = 4252) → H i.e. H = 33.

SHLD 16 bit address

L → (16 bit address)

H → (16 bit address +1)

3|5|16 (FRR WW).

⑤ XCHG HL → DE

1|1|4.

⑥ a) PCHL HL → PC

b) SPHL HL → SP

Eg: 2

* FE0 is not possible.

NOTE:→

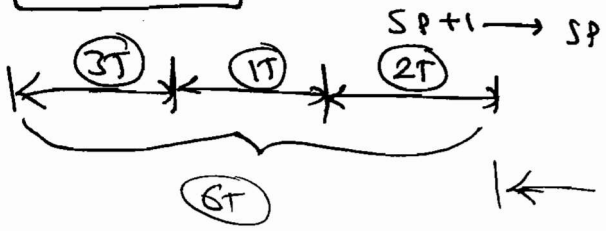
→ In 8085 µP opcode fetch machine cycle is extended by two T-states if the PC (or) SP value is changing. This results in opcode fetch machine cycle as shown in the following example.

Eg: 1 ADD B ⇒ (4T)

A + B → B X

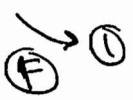
* INX R_p

INX Sp

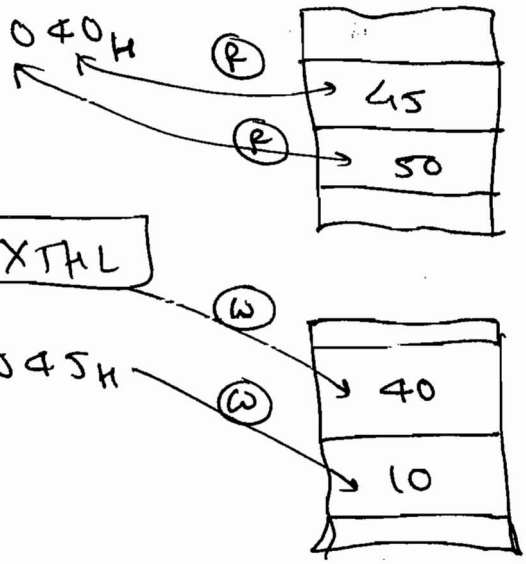


7) XTHL — Top of the Stack ↔ HL

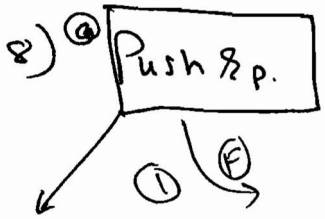
1 | 5 | 16



E.g. :→ HL = 1040_H



8) a. Push R_p.



Pre-decrement

R_{P_H} → ((Sp-1)). (2, W)

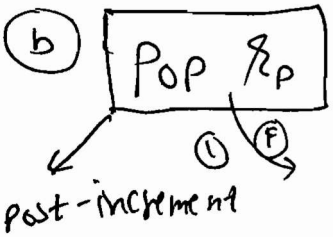
R_{P_L} → ((Sp-2)). (3, W)

and Sp = Sp - 2.

1 | 3 | 12

(S W W / 6 3 3)

b. Pop R_p



post-increment

((Sp)) (1) → R_{P_L}.

((Sp+1)) (R) → R_{P_H}.

Sp = Sp + 2.

1 | 3 | 10

(F R R / 4 3 3)

Eg: Let, $(F250)_H = 4B_H$.

$(F251)_H = 50_H$

$(F252)_H = F2_H$

$(FF53)_H = 04_H$.

① LXI, H, F250. ⟶ HL = F250 i.e M = $4B_H$.

INX H. ⟶ HL = F251 i.e. M = 50_H.

DCR M ⟶ M = $4F_H$.

MOV C, M. ⟶ C → $4F_H$. ⇒ M (copied) → C

C = ?

∴ **C = $4F_H$.**

② LHLD F251. ⟶ L = 50_H, H = $F2_H$. (F251) → L i.e. L = 50_H, (F252) → H i.e. H = $F2_H$ ∴ HL = F250h i.e. M = $4B_H$.

INX H ⟶ H = $F2_H$, L = 51_H. ⟶ i.e. M = 50_h.

INR M ⟶ M+1 = 51_H.

MOV C, M. ⟶ C = 51_H.

C = ?

C = 51_H.

③ LXI D, F252. ⟶ DE = $F252_H$

LXI H, F253. ⟶ **HL = $F253_H$.** ⇒ M = 04_H.

LDAX D. ((DE)) → A i.e. (F252) → A. A → F2.

ADD M A+M = F2 + 04_H. = A.

A = ?

A = 06_H with CY = 1.

Ex-4 Find the Value of Accumulator after executing the following programm.

(3) Address	(1) Mnemonics	(2) opcode
(4) ↓ 3000, 01 (execute)	MVI A, 36	3E, 36 ⇒ A = 36
3002, 03	ADI 40_H	82, 40. ⇒ A + 40 ⇒ A = 76
3004, 05106	STA 3007	32, 07, 30.
3007	~~XRA A~~ HLT	~~AF~~ 76
3008	HLT	76.

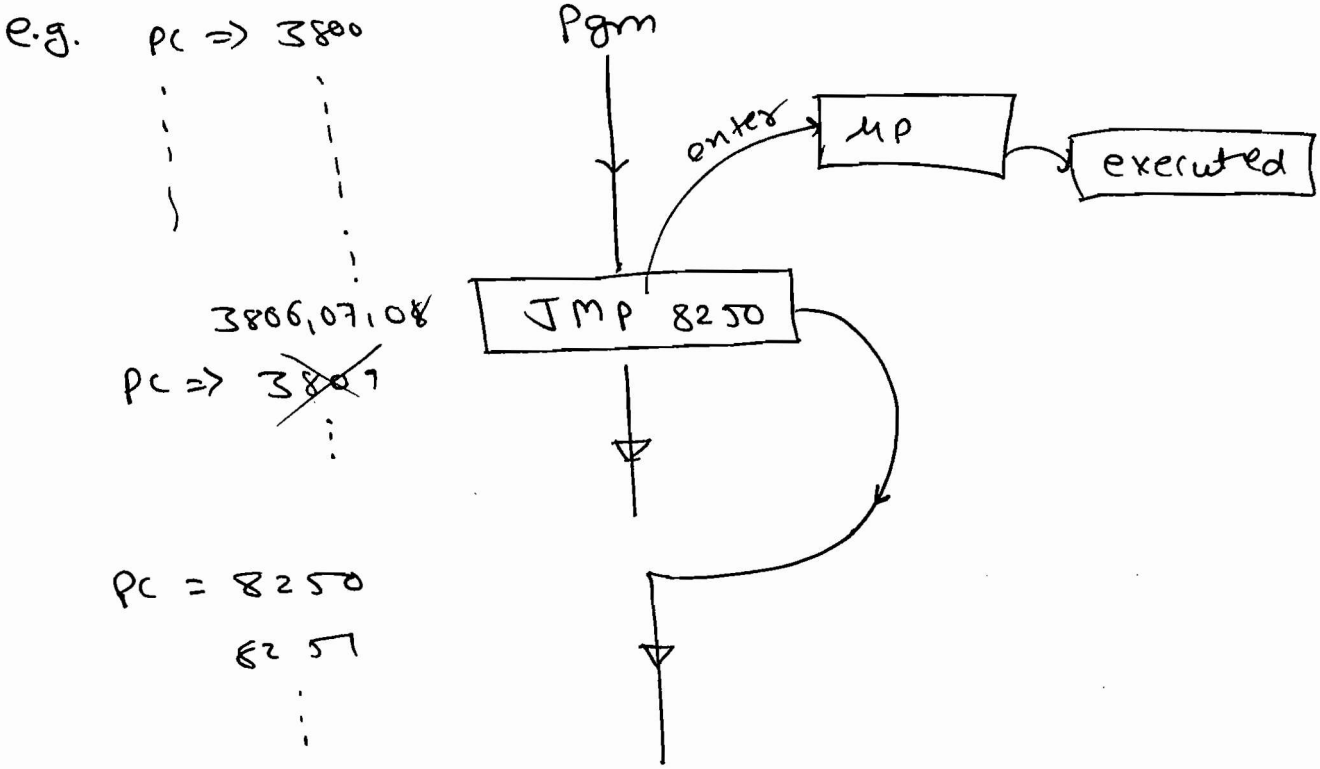
$A = 76_H$.

* 'XRA A' instⁿ is not executed because its opcode 'AF' is Replace by '76' which is opcode of HLT Instonction.

(4) Branching Instructions:

* Unconditional Jump.

Sr. No.	Instruction	operation	B | M | T	flag
1) (a)	JMP 16 bit address	16 bit addr → PC	3 | 3 | 10.	(FRR).



* Conditional Jump:

(b) JC 16 bit address ⇒ If CY = 1 (True)

16 bit addr → PC

3 | 2/3 | 7/10 | (False / True)

(1) (2) and CY = ?

→ JNC 16 bit address ⇒ True / If CY = 0

– JZ 16 bit add. ⇒ If Z = 0

→ JM — 16 bit address ⟹ True If S=1

↙ Jump on minus

→ JP — 16 bit address ⟹ If S=0

↙ Jump on plus

→ JPE — 16 bit address ⟹ If P=1

Jump on parity even.

JPO — 16 bit address ⟹ If P=0

Jump on parity odd.

* Unconditional Call:

2) (e) CALL 16 bit address.

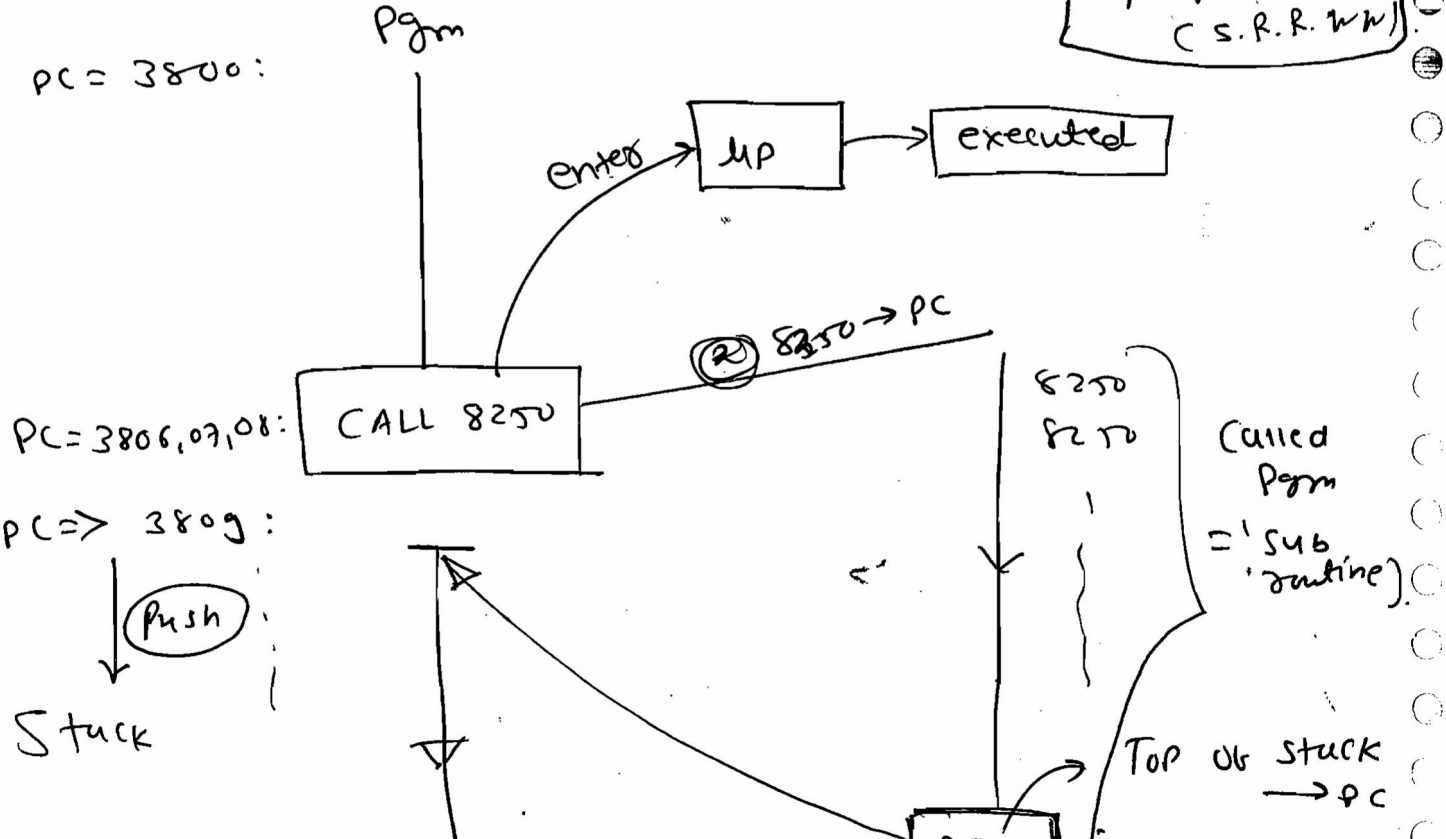
(a) $PC_H \rightarrow ((SP-1))$ (2) (W)

$PC_L \rightarrow ((SP-2))$ (1) (W) and $SP = SP-2$.

(b) 16 bit addr → PC.

3/5/18 (S.R.R. WW)

Eg.



* Conditional CALL:

(b) CC 16 bit add. ⟹ If $CY=1$

S ① ③ ②

ⓐ $PC_H \rightarrow ((SP-1))$ ④

→ $PC_L \rightarrow ((SP-2))$ ⑤

ⓑ 16 bit add. → PC.

(3 | 5 | 18, S.R.R.WW) $SP = SP-2$.

True

→ CNC 16 bit addr. ⓐ if $CY=0$

3 | 2 | 9 ← False

3 | 5 | 18 ← True

→ CZ 16 bit add. if $Z=1$ ~~$Z \neq 0$~~.

→ CNZ 16 bit add. if $Z=0$.

⟹ CM 16 bit add. ⟹ If $S=1$.

CP 16 bit add ⟹ If $S=0$.

CPE 16 bit add ⟹ If $P=1$.

CPO 16 bit add ⟹ If $P=0$.

3) a) RET ⟹ Top of stack → PC

F ① (similar to POP Instⁿ.)

i.e. $((SP))$ → PC_L (R ②)

$((SP+1))$ → PC_H. (R ③)

and $SP = SP+2$.

1 | 3 | 10 (F.R.R.)

b) Conditional:
- RC ✓
- RNC ✓
- RZ ✓
- RNZ ✓
- RP ✓

$1 \mid \frac{1}{3} \mid \frac{4+2 = 6T}{10}$ ← False / ← True

Ex-(1)

8204: CALL 8207 ② → PC = 8207

8207: POP H ⇒ Top of Stack → HL

i.e 8207 → HL

HL = ?

① ↓ Stack

$HL = 8207_H$

Ex-2 In the following program determine the return address of the Subroutine

LXI D, 705F ⇒ DE = 705F

8205 : CALL Sub1 ↓ PC = 8208

8208 : ↓

① push stack

Sub1 : POP H HL = 8208

DAD D HL + DE = HL = FF67

PUSH H ⇒ Top of the stack = FF67

RET. ⇓

PC ⇒ Top of the stack

$PC = FF67_H$

PC ⇒ FF67 :

Ex-3 Determine the acc. value after executing the following program

Main pgm

CALL Sub1

① ↓ XX YY

STACK

Subroutine

Sub1: XRA A ⇒ $A = 00_H$

CALL 4260H.

4260: INR A ⇒ $A = 01_H$ | $A = 02_H$.

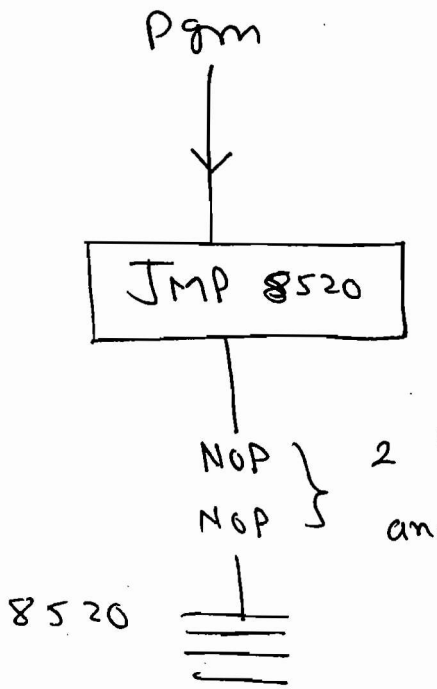
① ↓ RET ⇒ PC = 4260H i.e XXYY

⑤ Machine related, I/O Instruction:

1) a) NOP — No operation — 1|1|4. ⓐ Delay ⓑ To avoid Addr. Relocation.

b) HLT — µp is Halted — 1|2 (OR) more|5 (OR) more.

E.g. NOP



* HLT:

→ After executing the Halt instruction:

(i) it enters into Halt acknoledgement machine cycle.

(ii) the Buses are tristated.

(iii) Registers values are unaffected.

(iv) Either reset (OR) interrupt is required to bring the µp out of the halt state.

⇒ As a Good program we have to use interrupt instead of HLT.

2)

Used in "I/O mapped I/O mode".

(a) IN 8 bit address. → [8 bit addr.] → A, Input device

(1) (2) (3)

2 | 3 | 10 (FRI) (4+3+3)

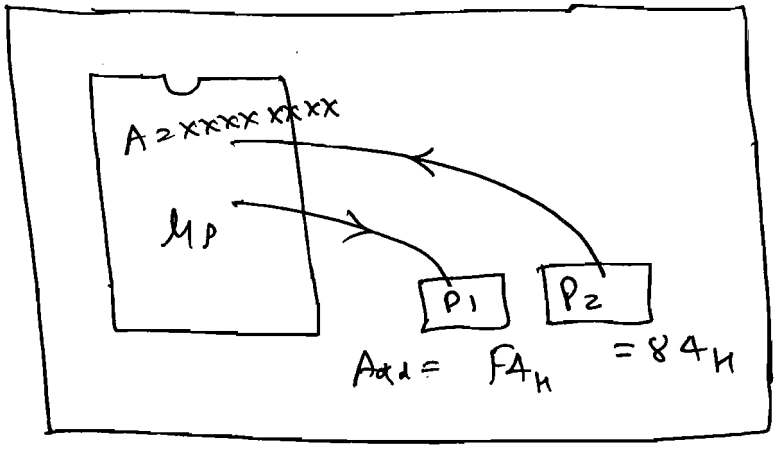
(b) OUT 8 bit add. → A → [8 bit add] → Output device

(F) (1) (P) (2) (W) (3)

2 | 3 | 10 (FRO)

* IN 84_H

OUT $F4_H$.



Ex-1 : After executing the following Program determine which port receives the data

MVI A, 87_H. => A = 87_H

MOV B, A. => B = 87_H

Ans: PORT 2.

Again: JMP Next

XRA B

OUT PORT 1

HLT

Next: XRA B ~~B => 00H~~ A ⊕ B => A = 00_H. | $00_H \oplus 87_H$ => 87_H.

TP Again

0000 0000
+ 1000 0111
1000 0111

$\Rightarrow$ 87_H is sent to PORT2.

Ex-2 The following Programm reads signed no. in 2's comp. form from port-1 when the execution of the program stops?

Again: IN PORT 1 $\Rightarrow$ PORT 1 $\rightarrow$ A = Sxxx xxxx

If CY=0, S=0 +ve no.

RLC $\Rightarrow$ CY=S

JNC Again.

RRC If CY=1, S=1 -ve no.

HLT

$\Rightarrow$ the above program stops the execution. when it reads a -ve no. from port+1.

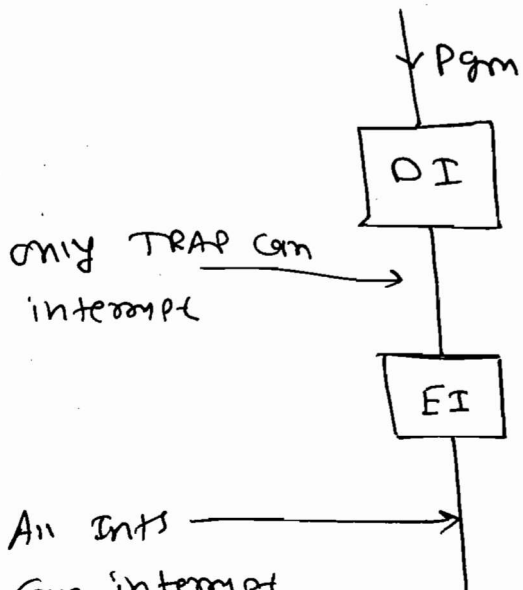
⑥ Additional Instruction:

① @ ~~DI~~ Disable Interrupts 1|1|4

$\Rightarrow$ { RST 7.5, 6.5, 5.5, INTR are disabled.

② EI Enable Interrupts 1|1|4

$\Rightarrow$ All Ints are Enabled.



2) DAA Decimal adjust Acc (BCD additions). 1|1|4 ⇒ All flags are affected.

27_{10}
$+ 36_{10}$
63_{10}

MVI A, 27.
MVI B, $36
ADD B ⇒ A+B → A = 5D$_H$
DAA ⇒ A = 63
HLT

```
      11
27 =  0010 0111
+36 = +0011 0110
      ---------
      0101 1101   x
    + 0000 0110
      ---------
      0110 0011
      ---------
        6    3
    = 63
```

Ex-1 Determine the PC value after executing the following programm.

LXI H, 8A79 ⇒ HL = 8A79.
MOV A, H ⇒ A → 8A
ADD L → A+L → A → 8A+79.
DAA A → 69.
MOV H, A H → 69.
PCHL PC → 6979.

PC = ?

```
  1 1 1
  1000 1010
+ 0111 1001
-----------
1 0000 0011
  0110 0000
-----------
  01 10 0011
  69.
```

NOTE: → DAA Instruction is effective only after

Ex-2 Determine the fⁿ of the following Program in B & C Register Contain BCD numbers

```
MVI  A, 99.   => A = 99.
SUB  C  -> A -> 99 - C =  9's Comp.(C)
INR  A  -> A+1 -> A -> 10's Comp.(C).
ADD  B  -> B + 10's C -> A.
DAA  -> A -> B-C.
HLT  ->
```

~~A -> B-C.~~

Let, B = 64;
C = 42;

99 - C = 99 - 42 = 57 → A

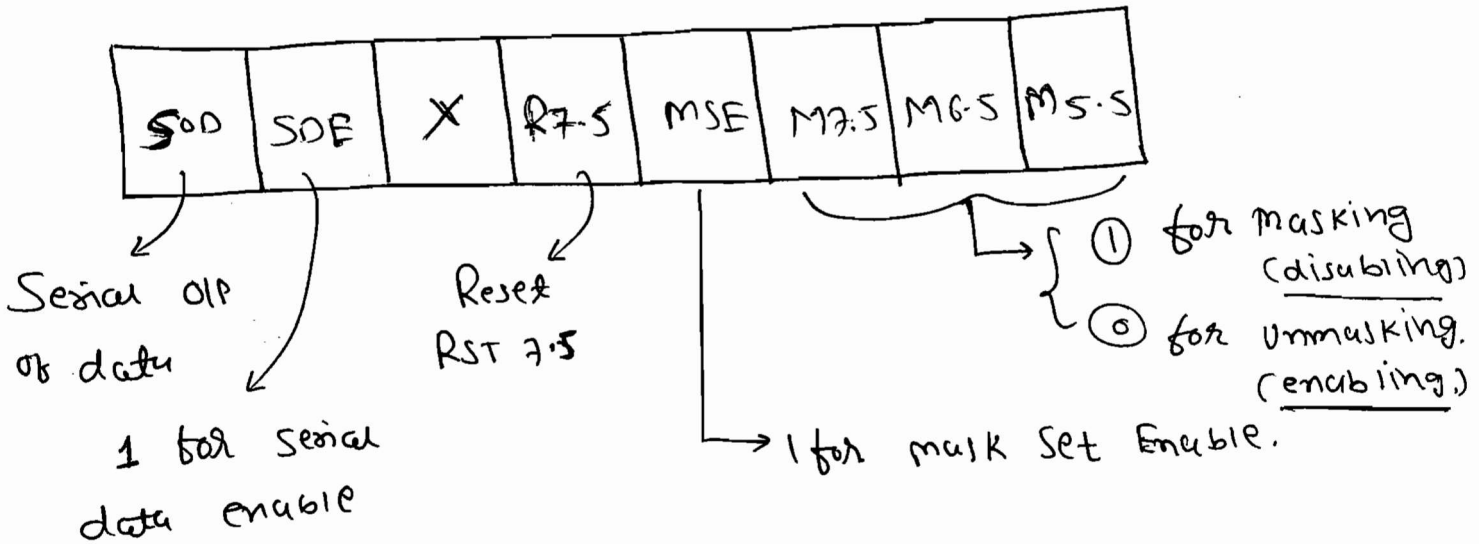
10's of A' = A = 58H.

3) a)

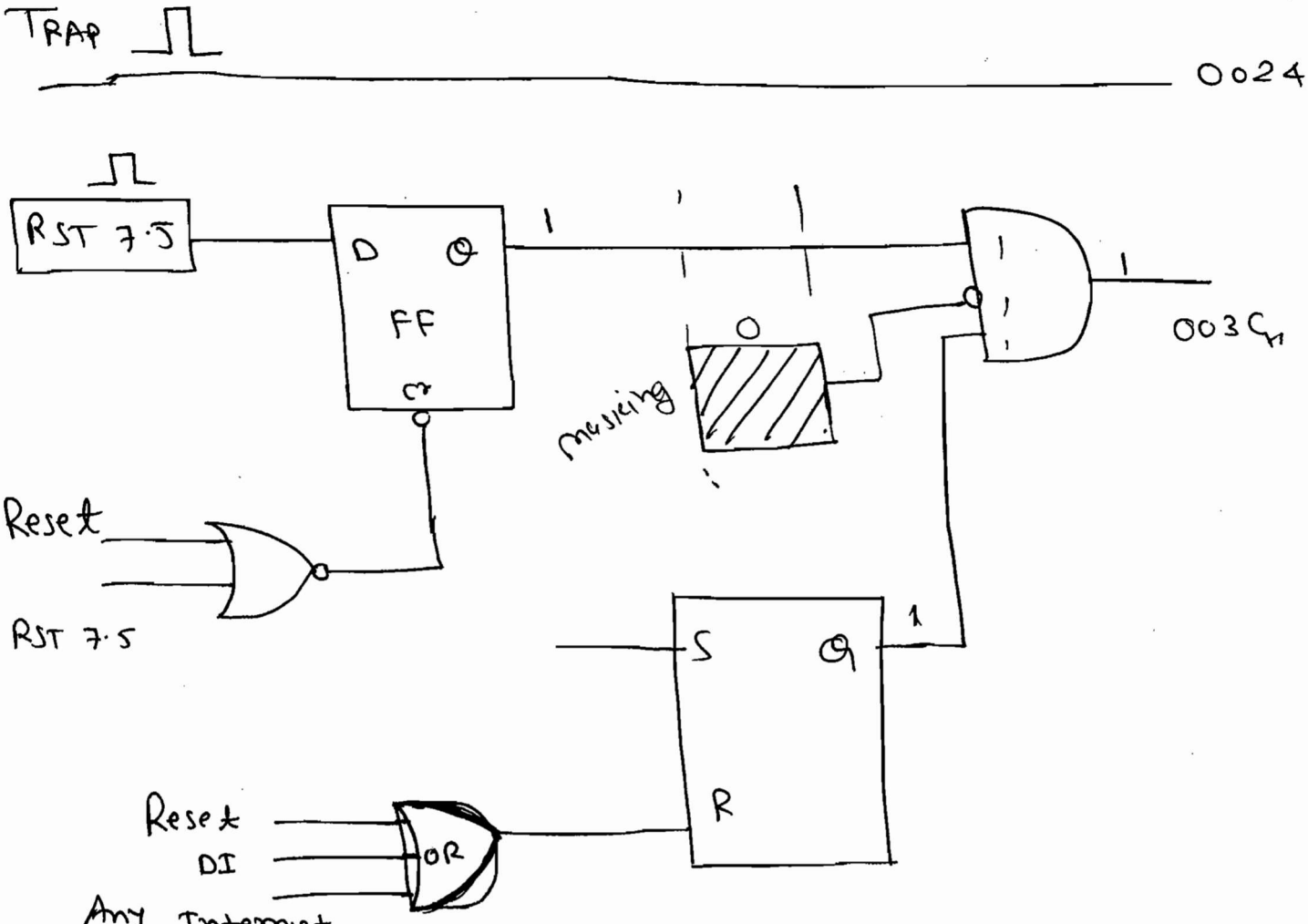
SIM (Set Interrupt Mask):

i) selective disabling Interrupts.

ii) serial output of data.



1	1	4



* For serving RST 7.5 the required values are Q=1, masking = 0, Q = 1.

Ex-1: Write instructions to mask RST 6.5 and RST 5.5.

⇒

```
MVI A, 0BH.
SIM
```

⇒ RST 6.5, 5.5 are masked (disable).

TRAP, RST 7.5 →

INTR Can interrupt

Ex-2 Determine the O/P of the following program. 117

```
Start: MVI A, 40H     => A = 0100 0000
       SIM            } '0' is sent on SOD pin of μP.
       Call 10msDelay
       MVI A, C0H     => A = 1100 0000₂
       SIM            } '1' is sent on SOD pin of μP.
       Call 10msDelay
       JMP start.
```

b) RIM (Read Interrupt Mask)

(a) To know Status of Interrupts.
(b) To read serial input [on SID pin of μP].

1/1/4

Acc =>

SID	I7	I6	I5	IE	I7.5	I6.5	I5.5

- SID: Serial input data
- I7, I6, I5: If 1 => Int is pending
- IE: 1 → Ints enabled. 0 → Ints disabled.
- I7.5, I6.5, I5.5: 1 = Int is masked (disabled). 0 = Int. is Unmasked (enabled).

Ex-1 After executing RIM Instruction the accumulator value is BC$_H$. Determine the status of the interrupt and the serial input.

Ans:

$A = BC_H = 1011\ 1100_2$

- 1 → serial bit on SID pin = '1'
- 1 → Ints are enabled
- 100 → RST 7.5 is masked.

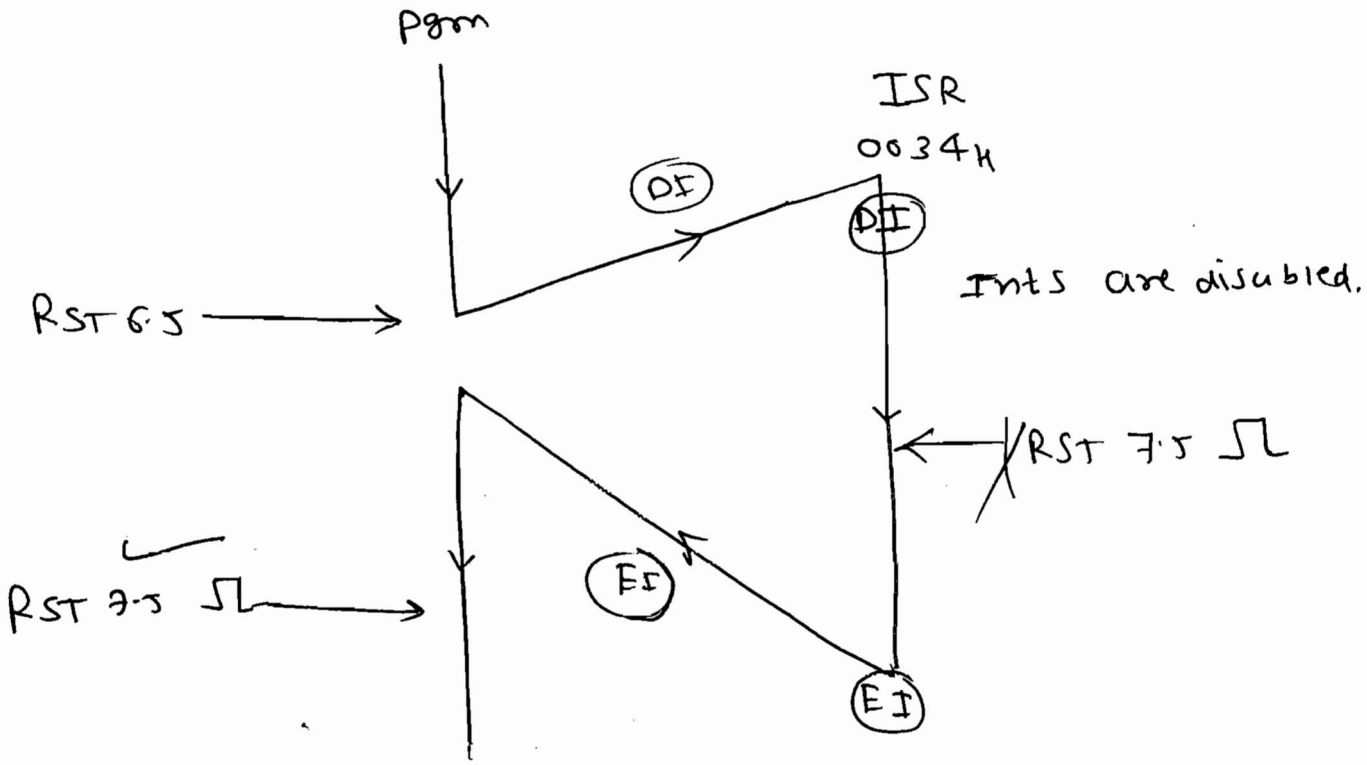
⇒ In 8085 µP the interrupts are disable in the following cases:

(i) When 'DI' instⁿ is executed.

(ii) When µp is Reset.

(iii) When µP Recognizes any one interrupt

⇒

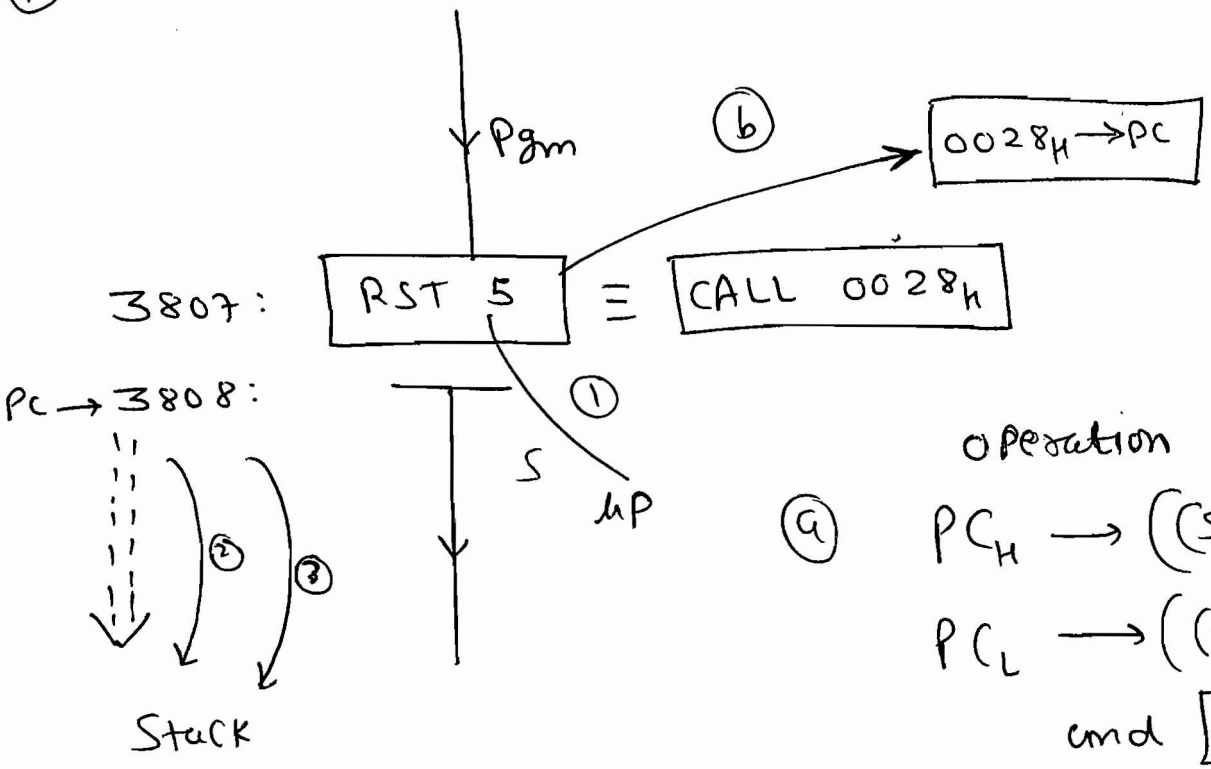


* Restart Interrupts (Software Interrupts)!!

⇒ **RSTn** ; n = 0, 1, ..., 6, 7.

Instruction	opcode	Address.
RST 0	$C7_H$	0000_H
RST 1	CF_H	0008_H
RST 2	$D7_H$	0010_H
RST 3	DF_H	0018_H
RST 4	$E7_H$	0020_H
RST 5	EF_H	0028_H
RST 6	$F7_H$	0030_H
RST 7	FF_H	0038_H

Note.. (*)



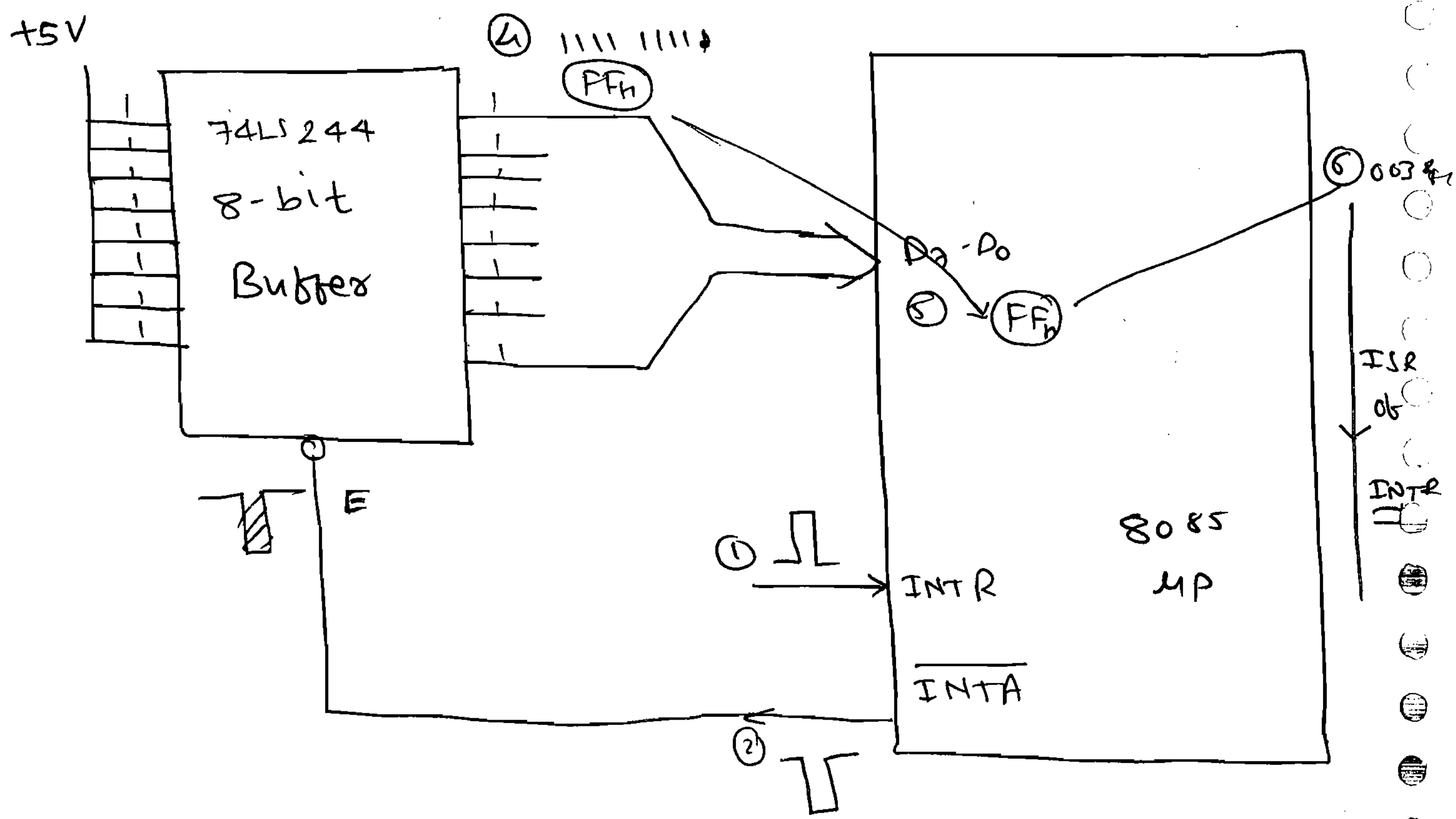
operation

(a) $PC_H \rightarrow ((SP-1))$

$PC_L \rightarrow ((SP-2))$

and SP = SP - 2.

⇒ Providing Addr. Location to NVI . INTR:



Ex-1 How many times the following loops will be executed? Assume initially All the flags are cleared.

(a) LXI B, 0002H Given Z=0.

Loop : DCX B
 JNZ Loop

BC=0001	BC=0000H	BC= FFFFH	...
Z=0 (T)	Z=0	Z=0 (T)	

* Infinite loop [∵ only Arith Instn "DCX B" doesn't affect any flags].

* Exceptions:

① ANA, ORA, XRA ⇒ CY=0.

② INX Rp, DCX Rp ⇒ No flags are affected.

③ INR R/M, DCR R/M ⇒ 'CY' flag is not affected.

b) XRA A given z=0 ⇒ A = 00H. ⇒ Z=1

Imp ** LXI B, 0002 ⇒ B = 0002H

Loop: DCX B ⇒ B = 0001H.

(T) z=0 { JNZ Loop. → Z=1. (F)

↓ (F) z=1

(F) ⇓

* only once it is executed.

[∵ XRA A will set the zero flag.]

Ex-2 In the following Programm, How many times the decrement C instn is executed.

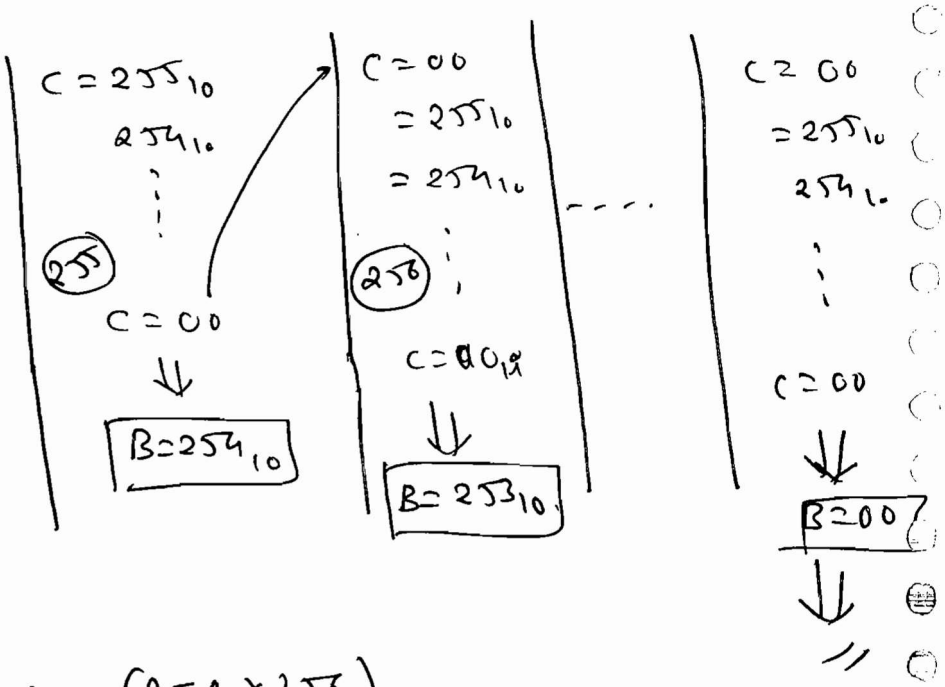
a) MVI C, 00H. ⇒ C=00H

DCR C | C = FFH = 255$_{10}$

* DCR C is executed 256 times.

b) MVI B, 255D.
MVI C, 255D. } B = C = 255D

```
Loop: DCR C
      JNZ Loop.
      DCR B.
      JNZ loop.
```



Ans: (1 X 255) + (254 X 256) times.

DCR is executed

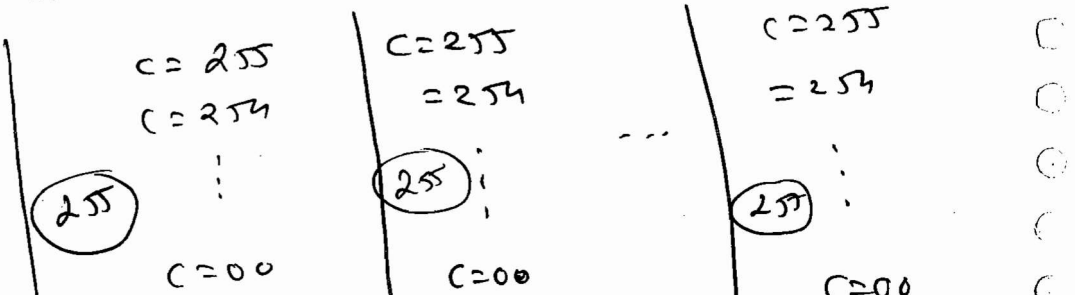
* 1st time => 255 times.

* Remaining 254 times => 256 times

Hence, (1 X 255) + (254 X 256)

c) MVI B, 255D
Loop2: MVI C, 255D. } B = C = 255D

```
Loop1: DCR C.
       JNZ Loop1
       DCR B
```



=> DCR is executed **255 X 255 times**