M & A (Microcontroller & Application)

↳ 8085 → Microprocessor.
↳ 8051 → Microcontroller.

1) 8 7 F B . 39
+ 9 A B C . 45
1 9 9 B 7 . 7 7

11
+12
23 → 23
−16
07 ⇒ $(17)_{16}$
↑ Time Sub.

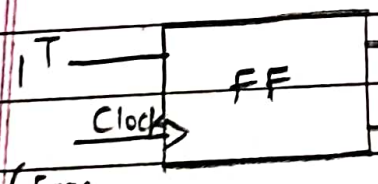
2) 1 0 1 1 0 . 1 0 1
+ 0 1 1 1 1 . 1 1 1
1 0 0 1 1 0 . 1 0 0

(OR) 16 | 23 | 1
16
7

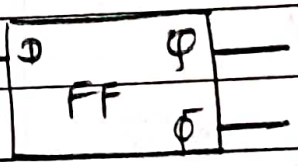
* Memory :-

↳ Primary memory
↳ Secondary Memory

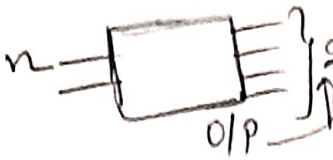
↳ RAM (Random access Memo
↳ ROM (Read only Memory



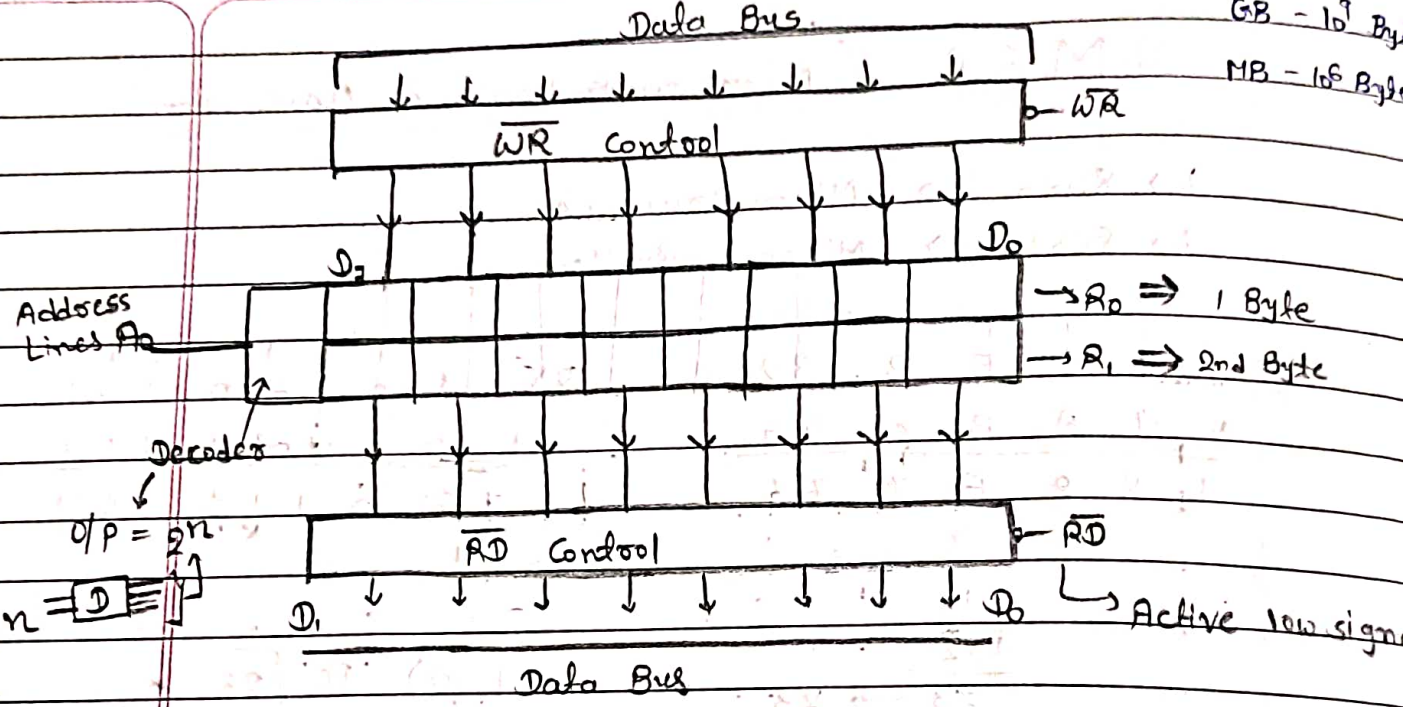
(Edge Triggered) 1-Bit Memory (RAM)



1 byte = 8 bits.

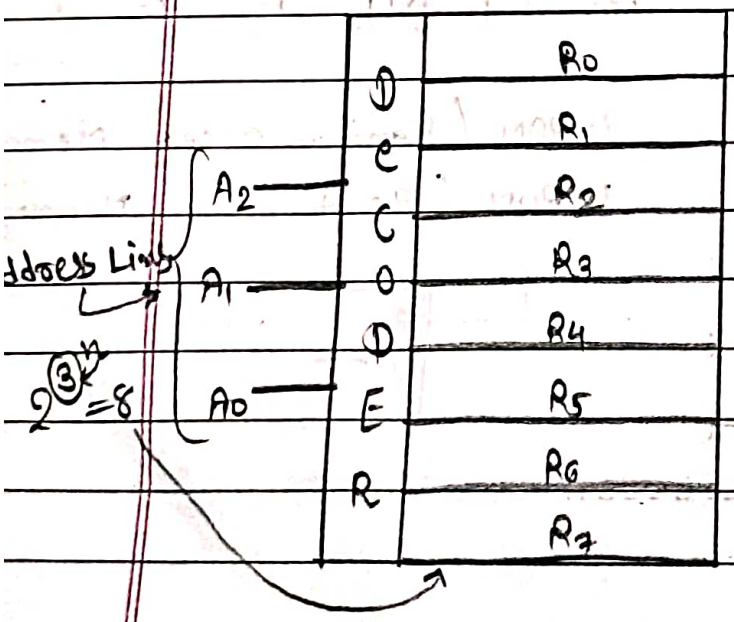


GB - 10^9 By
MB - 10^6 Byt

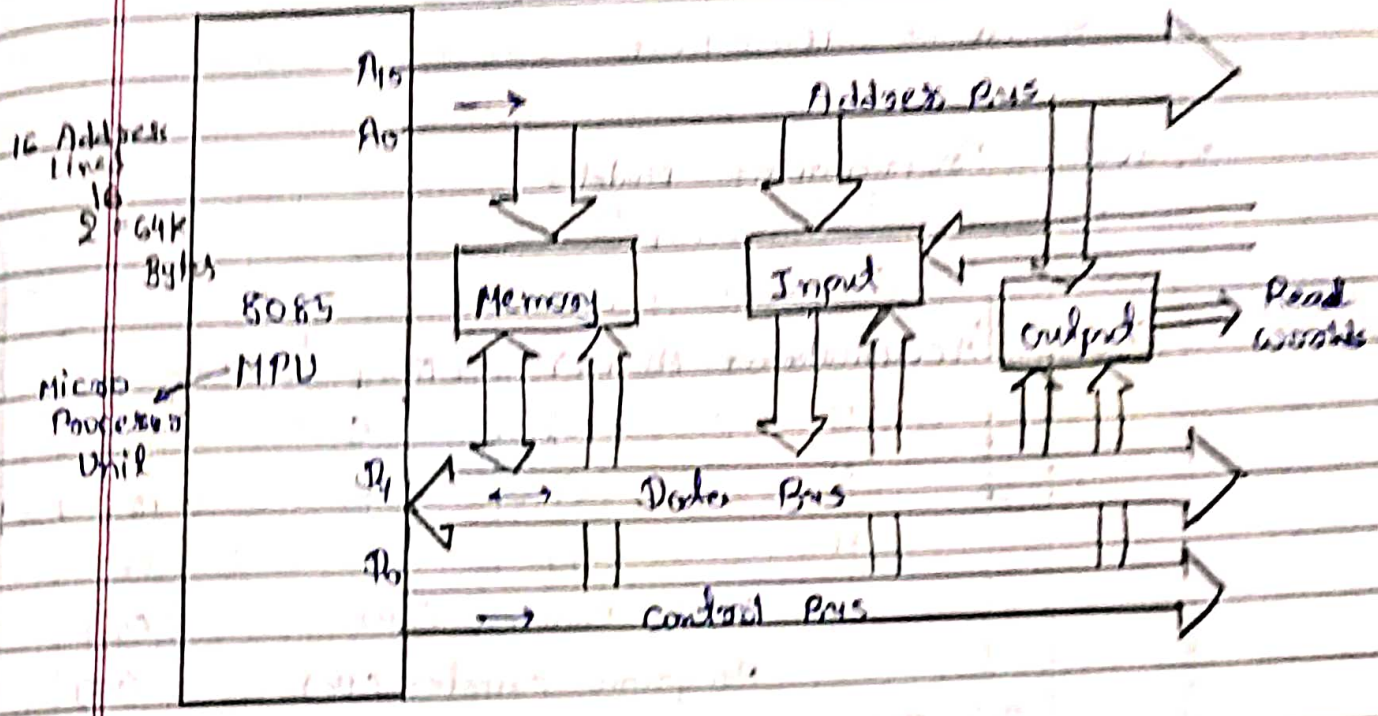


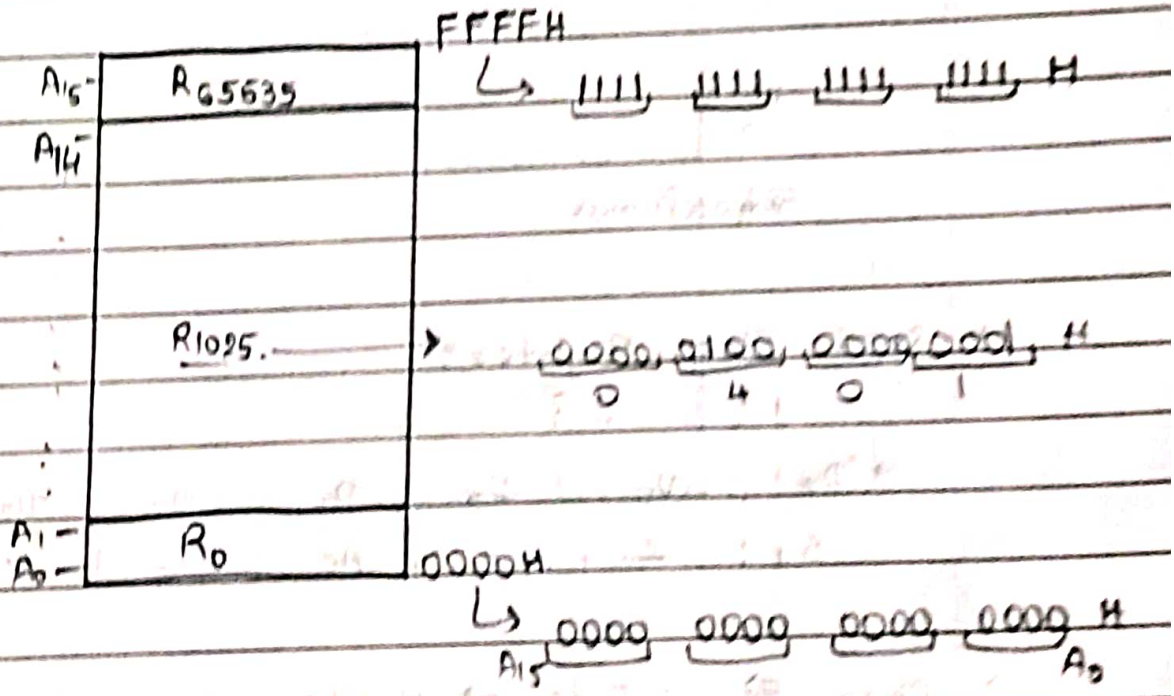
Address Line (A_0)	Reg. selected
0	R_0
1	R_1

2^{16} = 64KB



* 8085 Bus Organization :



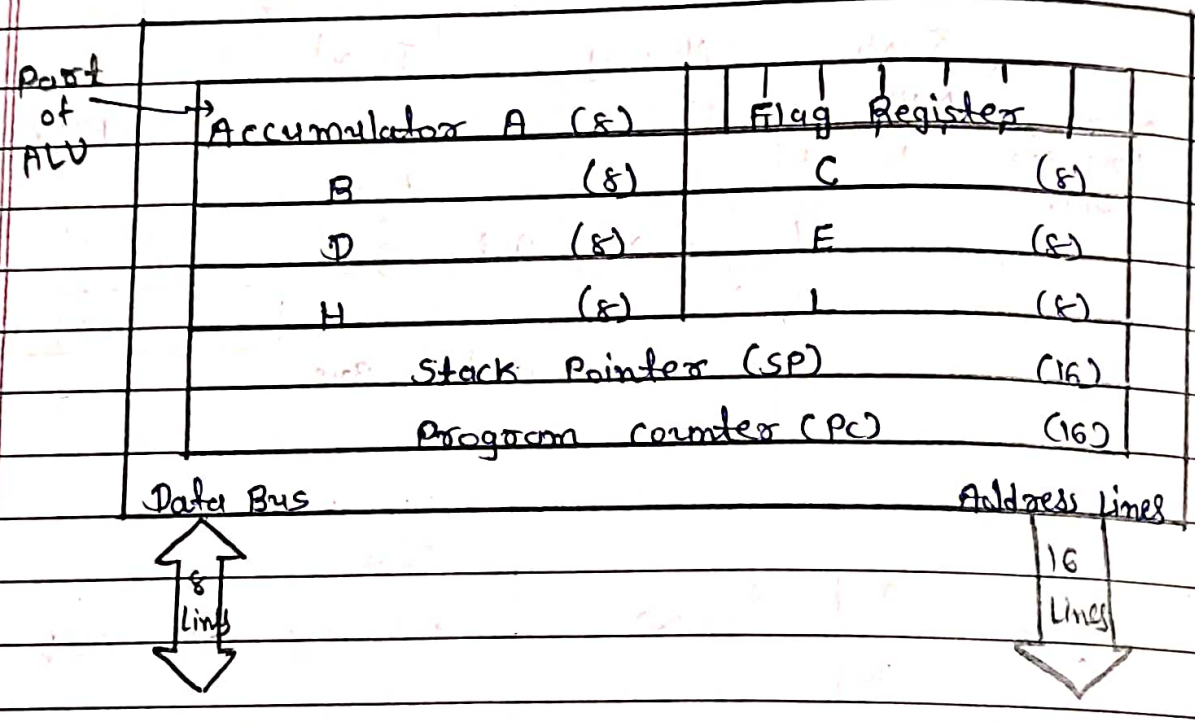


* Size of Memory :-

2^n = Memory

n = no. of address lines

* 8085 Programming Model :-



* 8085 Flag Register :-

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z	-	AC	-	P	-	CY

→ Char ⇒ 1 Byte → 8-bits.

↳ $2^8 = 256$

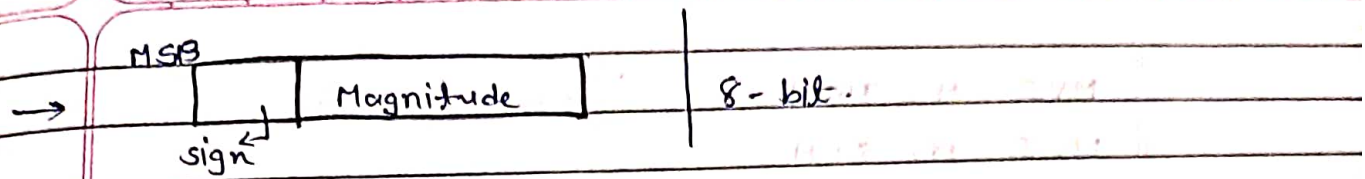
unsigned char → 0 – 255

signed char → $\frac{256}{2} = 128$

-128 → -1

128 → -Ve | 128 → +Ve

0 → +127



If MSB is 0 → +ve num

1 → -ve num.

Ex:- 0 111 1001 ⇒ +Ve.

MSB is 0

Ex:- 1 000 1001 ⇒ -Ve.

MSB is 1.

→ in Hexa

89 H

94H 85H ABH → 1 MSB (Negative value).

05H 19H 25H → 0 MSB (Positive value).

+ve

0000 0000 → 00H

:

0111 1111 → 7FH

-Ve

1000 0000 → 80H

:

1111 1111 → FFH

MVI A, 95H
MVI B, 89H
ADD B
HLT

95H → 1001 0101
89H + 1000 1001
1 1 0001 1110
MSB = 1 (sign)
E → 1EH

→ If MSB is = 0 so, S = 0. → (sign will be 0).

→ Z (Zero) = 0 If all result 0 → Z = 1.
If result non zero → Z = 0.

→ AC (Auxillary carry)
If $D_3 \rightarrow D_4$ is carry so, AC = 1
If no carry = 0.
AC = 0.

→ CY = (last final carry)
If last final carry is Availabe so CY = 1
If not so, CY = 0.

→ P = (Even and odd). even parity = 1. } so, Parity = 1
(parity)
If 1111 → so Parity is 1
If all are 0 so parity is also 1.
~~If 0000 → so Parity~~

0000 0000 NOR → 1

0000 0001 → 0

NOR logic is apply to design zero flag

00 1
01 0
10 0
11 0.

Ex:- MVI A, FFH
MVI B, 01H
ADD B
HLT

AC=1

```
  1 1 1 1 1 1 1 1
+ 0 0 0 0 0 0 0 1
-----------------
1 1 0 0 0 0 0 0 0
```

Carry=1, MSB=0, S=0, Z=1

S=0
Z=1
Ac=1
CY=1
P=1
↳ Parity zero as a even num

Ex:- MVI A, F9 H
MVI B, 55 H
ADD B
MVI C, 67H
ADD C
HLT

```
  1 1 1   1
  1 1 1 1 1 0 0 1
+ 0 1 0 1 0 1 0 1
-----------------
1 1 0 1 0 0 1 1 1 0
```

S=0
Z=0
Ac=0
CY=1
P=1

```
  0 1 0 0 1 1 1 0
+ 0 1 1 0 0 1 1 1
-----------------
  1 0 1 1 0 1 0 1
```

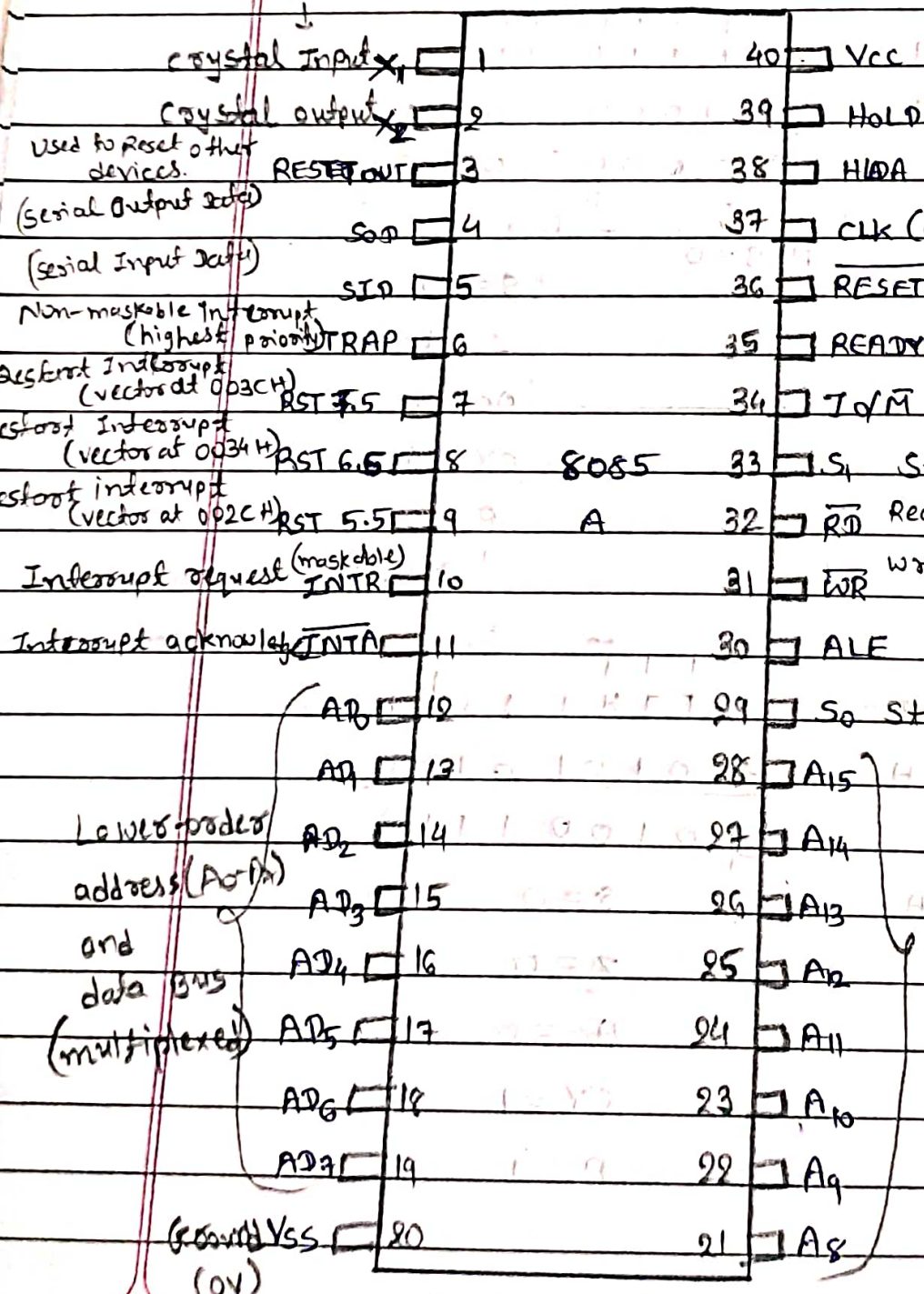
S = 1

Z = 0

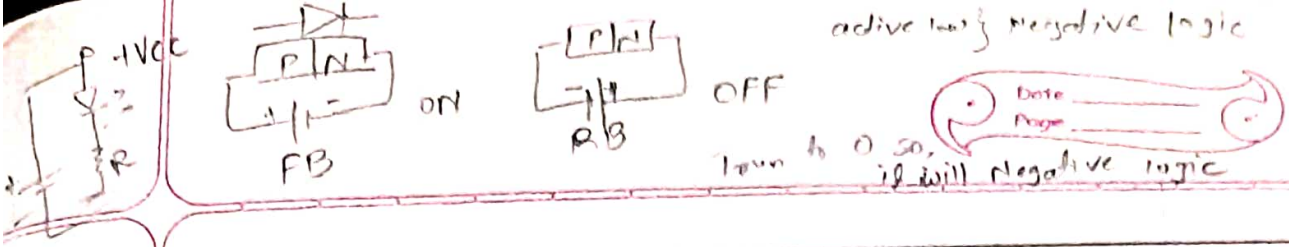
AC = 1

CY = 0

P = 0



8085 Pinout



* Data Transfer Instructions:-

$M \Rightarrow 2^{16} \Rightarrow 64\,KB$

16 → No. of Address Lines

1.) MOV Rd, Rs

2.) MVI

Mnemonics

OPCODE — MOV,

OPERANDS — A, B

MOV M, R ; M → Memory

MOV R, M

* Copy data on memory location 2000H from Register A.

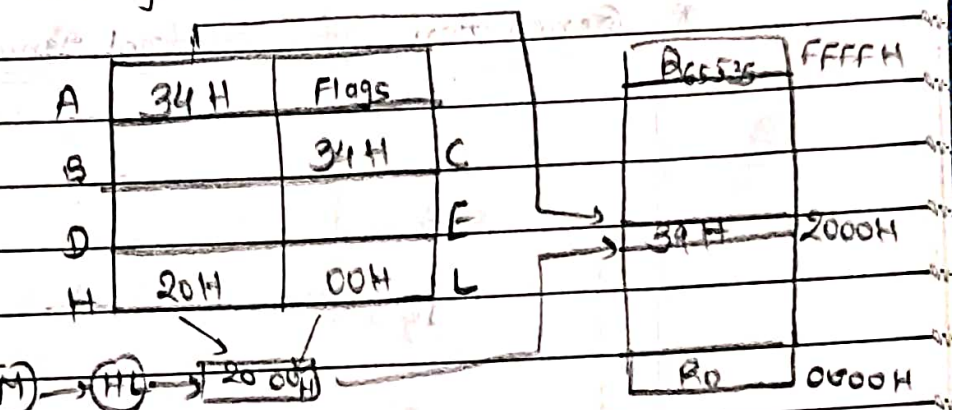
MVI A, 34H

MVI H, 20H

MVI L, 00H

MOV M, A

MOV C, M



* Load to data Bytes In Register, RA and RC Add them and then store the result on to memory location 1050 H. also Indicate the status of flags After Addition.

MVI A, 32 H — 0011 0010

MVI C, 13 H — 0001 0011

ADD C — 0100 0111

ADD C: A + C → A

MOV A, C

LXI H, 10 H

MVI L, 50 H

MOV M, A

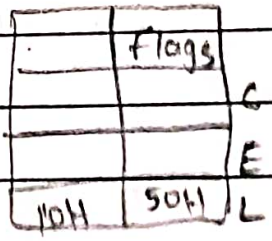
S = 0

Z = 0

AC = 0

CY = 0

P = 1



3) LXI

[Load 16 bit (x) Immediate data]

LXI Rp, 16 bit Imm. Data

↳ Register Pair ⇒ B → BC

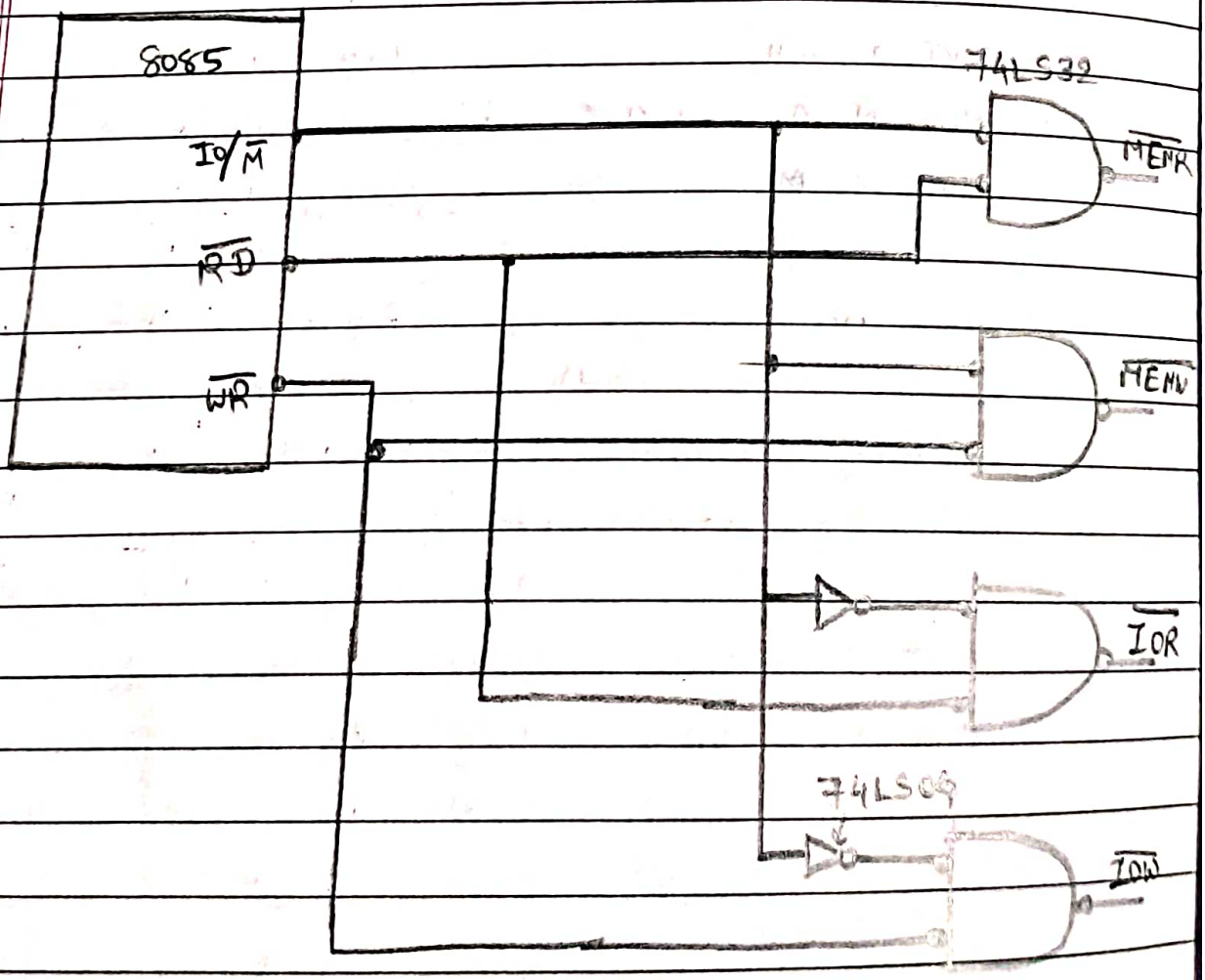
D → DE

H → HL

SP → Stack Pointer.

LXI H, 1050H

* Generation of control signals ($\overline{MEMR}$, $\overline{MEMW}$, $\overline{IOR}$, $\overline{IOW}$)

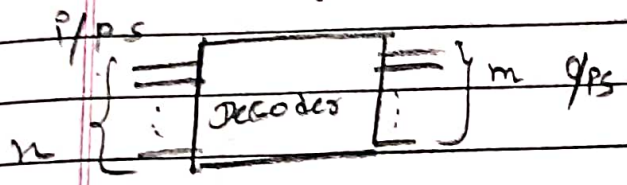


⇒ Bubbled NAND Gate

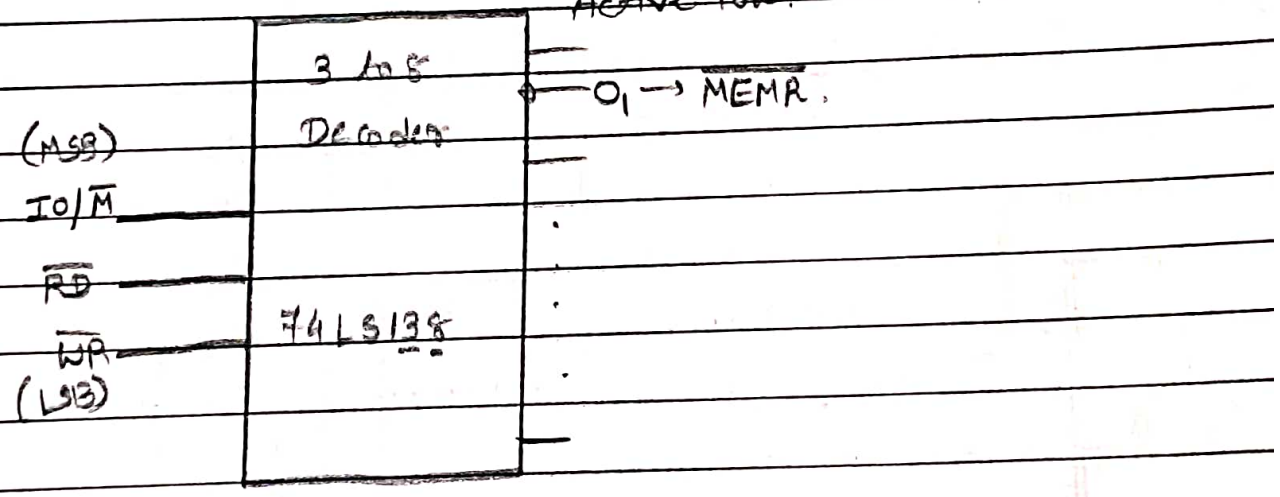
$(\overline{A \cdot B}) = \bar{A} + \bar{B}$

$= A + B$

* Generation of control signals using Decoders:-

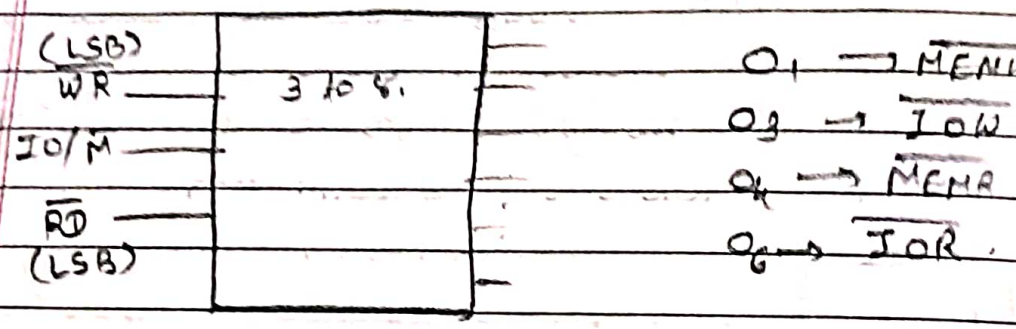


$m = 2^n$

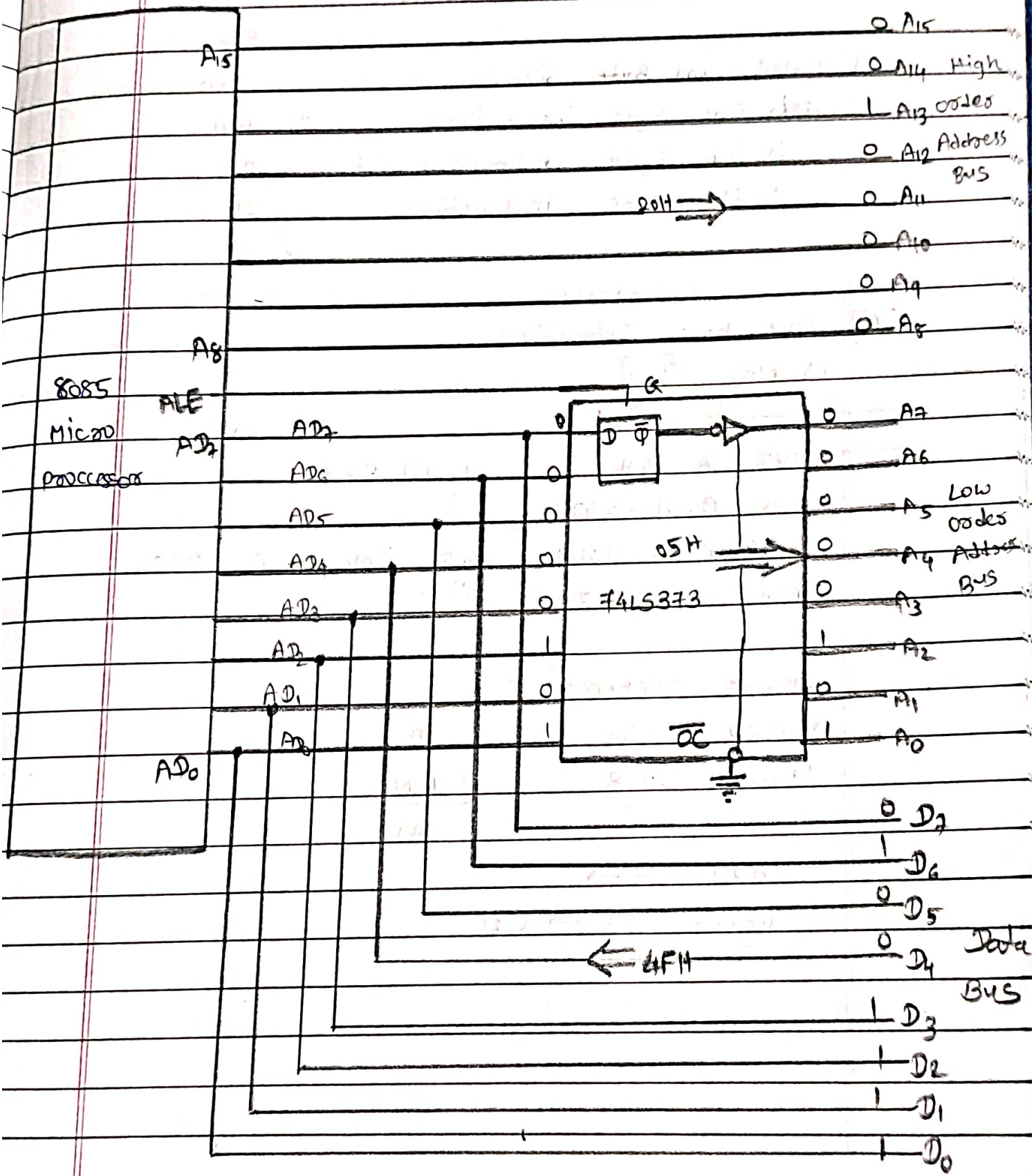


	IO/M	RD	WR	
	0	0	0	Unused
O_1	0	0	1	$\overline{MEMR}$
O_2	0	1	0	$\overline{MEMW}$
	0	1	1	Unused
	1	0	0	Unused
O_5	1	0	1	$\overline{IOR}$
O_6	1	1	0	$\overline{IOW}$
	1	1	1	Unused.

* Generate the control signals using the appropriate decoder. consider the ~~write~~ Write signal is at MSB and ~~write~~ Read signal is at LSB.



* Demultiplexing Address and Data Bus :-



~~* Cat~~

* Classification of instructions :-

[1] Based on Byte size :-
- ↳ One byte instructions
- ↳ two byte instructions
- ↳ three byte instructions.

Reg. codes

Code	Reg.
000	B
001	C
010	D
011	E
100	H
101	L
110	M (Memory)
111	A

[A] One byte Intructions

1.) MOV Rd, Rs

MVI A, 32H — 01 Rd Rs R

MOV B, A → 47H

MOV C, B → 48H — 2nds MOV C, A 4FH (H)

MOV D, C → 51H

OPCODE OPERAND

→ ADD R/M — A + R/M (A)

10000 R R R

ADD B →

10000 000 → 80H

[B] Two byte Instructions:-

1.) MVI Rd, 8-bit data
(Mov Immediate 8-bit data).

MVI M, 8-bit data,
one byte

0 0 Rd Rd Rd 1 1 0 → 1st Byte

8-bit data → 2nd Byte

MVI A, 35H

00 111 110
3 EH

3EH
35H

MVI 00 --- 110
ADD 10000 ---
MOV 01 -------

Ex:- MVI A, 39H 00 111 110
MVI B, 52H 00 000 110
ADD B 10000 000
MVI C, A2H 00 001 110
ADD C 10000 001
MOV M, A 01 110 111
MOV H, A 01 100 111
MVI E, F9H 00 011 110
ADD E 10000 011
MOV D, A 01 010 111

[C] Three byte Instruction

LXI

00 --- 0001

00-B-BC
01-D-DE
10-H-HL
11 S-SP

Memory Location	Mnemonics.	Byte size	Hex code
1000H	MVI A, 39H	2	3E
1001H			39H
1002H	MVI B, 52H	2	06H
1003H			52H
1004H	ADD B	1	80H
1005H	MVI C, A2H	2	0EH
1006H			A2H
1007H	ADD C	1	81H
1008H	MOV M, A	1	77H
1009H	MOV H, A	1	67H
100AH	MVI E, F9H	2	1EH
100BH			F9H
100CH	ADD E	1	83H
100DH	MOV D, A	1	57H

* Memory Interfacing with 8085 :-

Ques :- Interface a 1 KByte ram with 8085 show all design steps and draw the final Circuit diagram. use NAND Gate for Chip select logic.

→ **Step 1 :-** Identify No. of Address lines & chip Select Lines.

Chip size = 1 KB (1024 Bytes) = 2^n → n = 10 Address lines [$A_0 \rightarrow A_9$]

chip select lines = 16 − 10 = 6 [$A_{10} \rightarrow A_{15}$]

Step 2:- Decide the chip select logic & Address Lines 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

← chip select lines → | ← Address Lines →

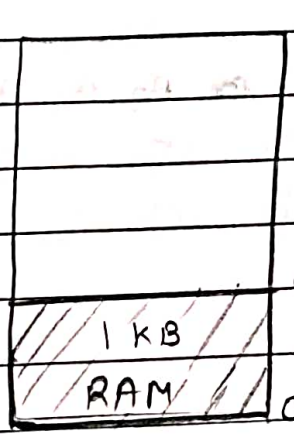
Starting Add.

0 0 0 0, 0 0 | 0 0, 0 0 0 0, 0 0 0 0 → 0000H

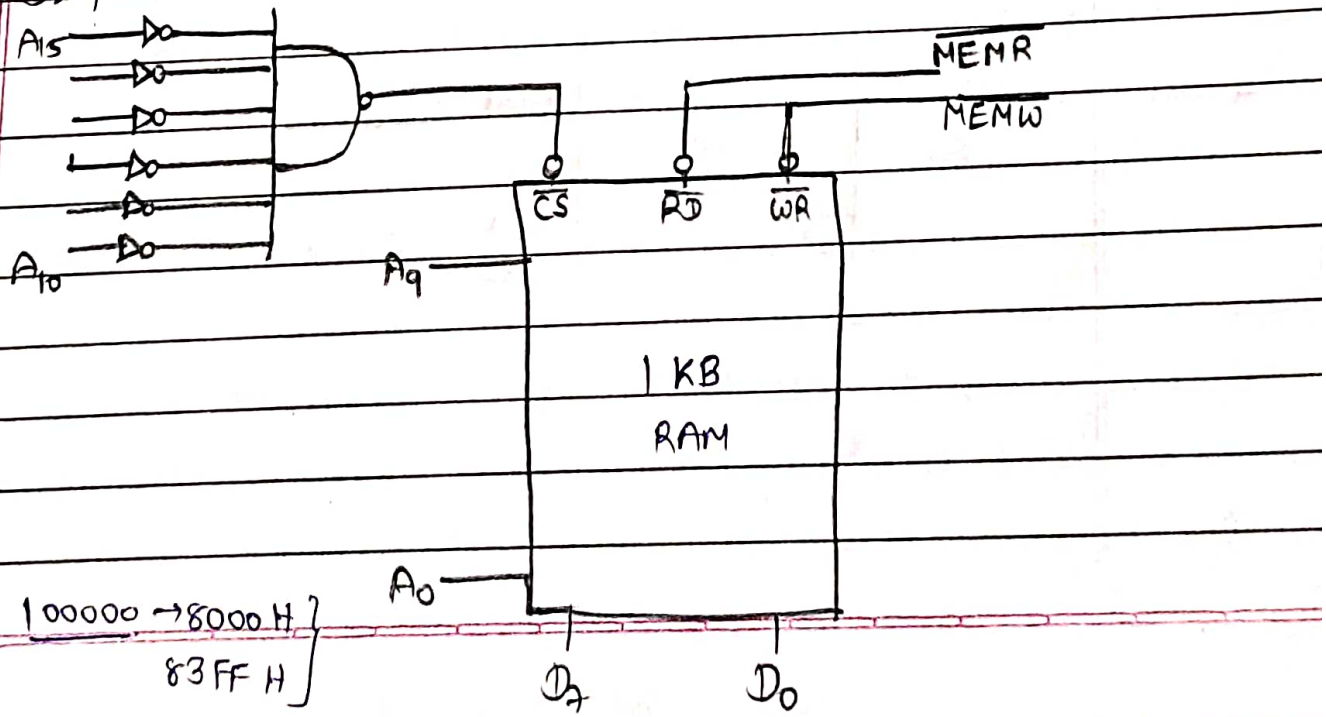
0 0 0 0, 0 0 | 1 1, 1 1 1 1, 1 1 1 1 → 03FFH

Ending Add.

Step 3:- Memory Map.



Step 4:- Final (circuit (Hardware)) Diagram.



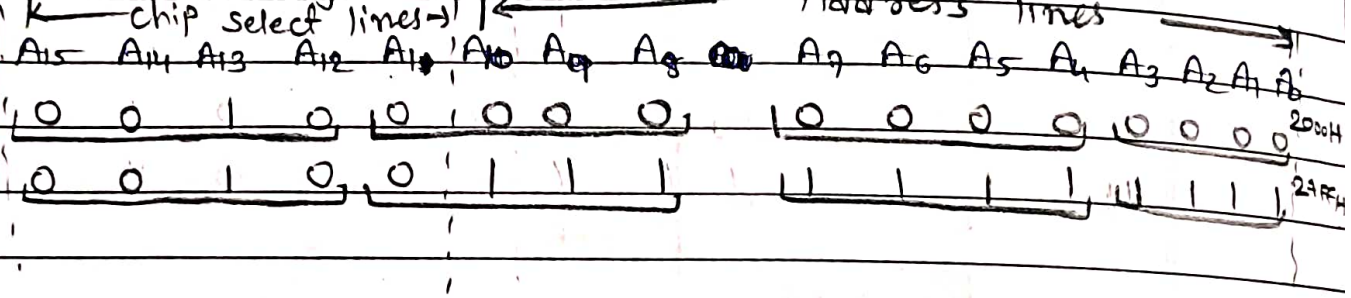
Ques:- Interface a 2 KB ROM with 8085 the chip select logic should be non zero show a design steps and draw circuit diagram

→ Step 1 :- Indentify No. of Address lines & chip select lines

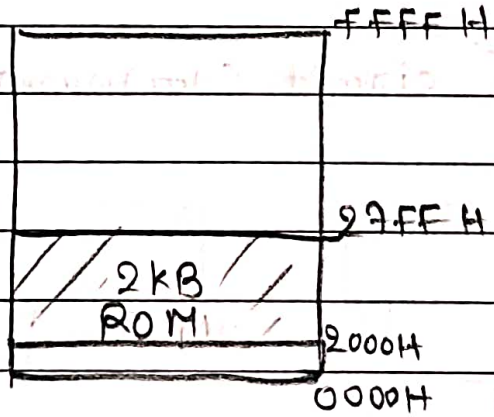
chip size = 2 KB = 2^n = n = 11 Address lines
(1024) 2×2^{10} [$A_0 \rightarrow A_{10}$]

Chip select lines = 16 − 11 = 5 [$A_{11} \rightarrow A_{15}$]

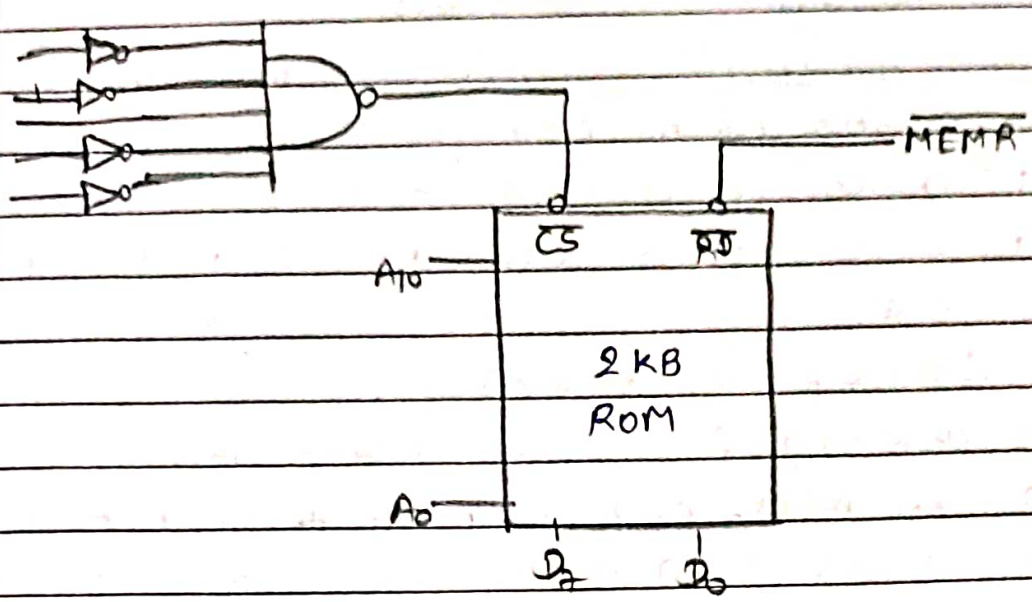
Step 2 :- Decide the chip select logic & Address range



Step 3 :- Memory Map



Step 4:- Final circuit (Hardware) Diagram



15_{10} = OFH
16_{10} = 10H
255_{10} = FFH
256_{10} = 100H

Ques:- A 256 Bytes RAM has Ending Address BOFFH what should be the Starting Address?

→ chip size = 2^8 = 8 Address lines.

B000H

Starting Address = Ending Address - (Size of RAM - 1)

(BOFFH - (100H - 1))

BOFFH - OFFH

Ques:- Interface 512 Bytes ROM and 4KB RAM with 8085 use NAND Gate for Chip select logic and show all design steps and draw a final circuit diagram

→ step 1 :- ~~Finde~~ Identify No of Address lines & chip selecte lines

→ for chip 1

→ chip size = 512 Bytes ROM

2^8

↳ n = 8 Address lines (A0 → A7)

→ Chip select lines = 16 - 8 = 8

→ for chip 2

→ chip size = 4KB RAM

$4 \times 1024 \rightarrow 2^2 \times 2^{10}$

n = 12

-Chip select lines = 16 - 12 = 4

Step :- 2

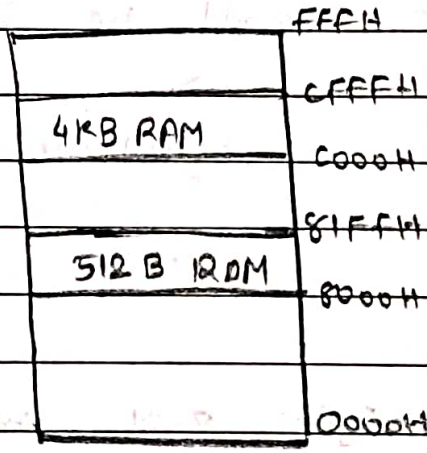
→ for chip 1 = (512 Bytes).

← chip select →	← Address lines →															
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	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8000H
1	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	81FFH

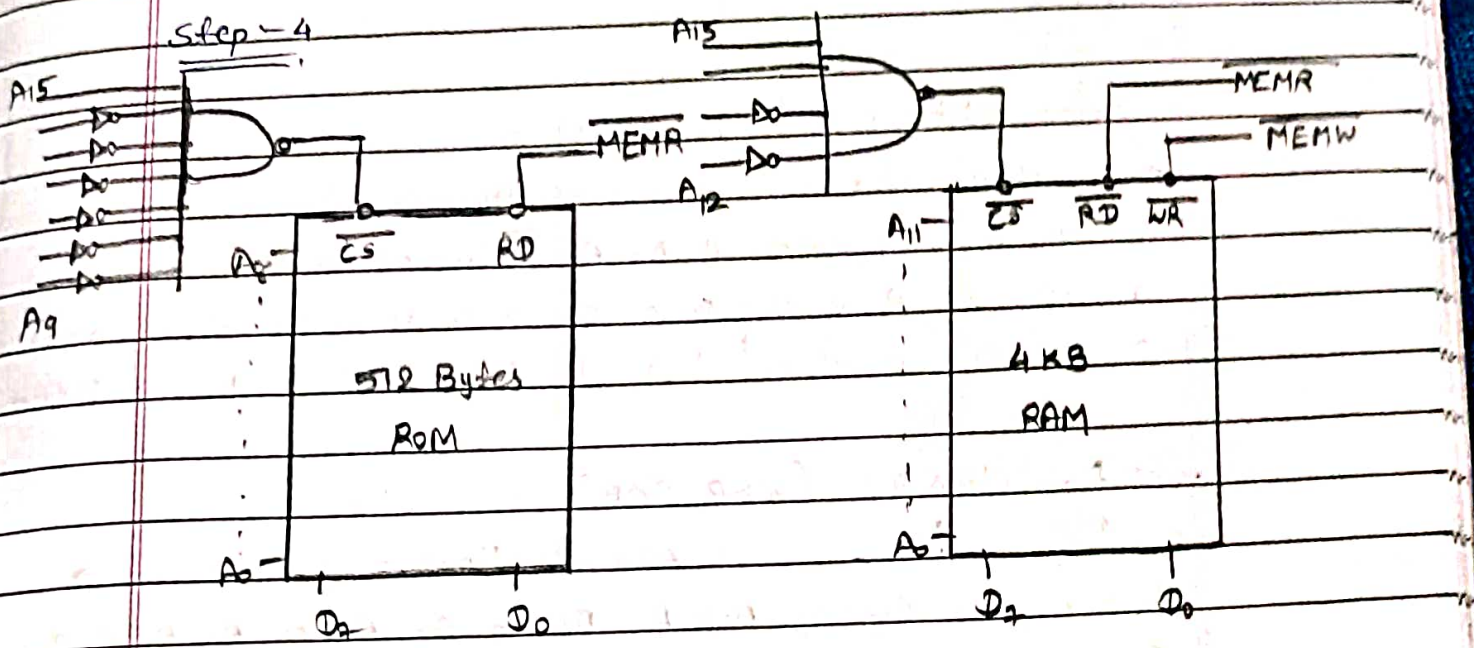
→ for chip 2 = (4KB RAM)

← chip select →	← Address lines →															
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	0	0	0	0	0	0	0	0	0	0	0	0	0	0	C000H
1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	CFFFH

Step 3 :-



Memory Map.



Ques:- Interface a 1 KB ROM and 8 KB RAM with 8085 And Show that the ROM is connected at the starting Address of the memory map show all the design steps and draw the final circuit diagram.

→ Step 1 :- Identify No. of Address lines & chip select line

Chip 1 :- chip size = 1 KB = 2^{10} → n = 10 Address line
chip select lines = 16 - 10 = 6.

Chip 2 :- chip size = 8 KB = 2^{13} → n = 13 Address lin
chip select lines = 16 - 13 = 3.

Step 2:-

→ for chip 1 :- (1 KB ROM)

← chip select → | ← Address lines →

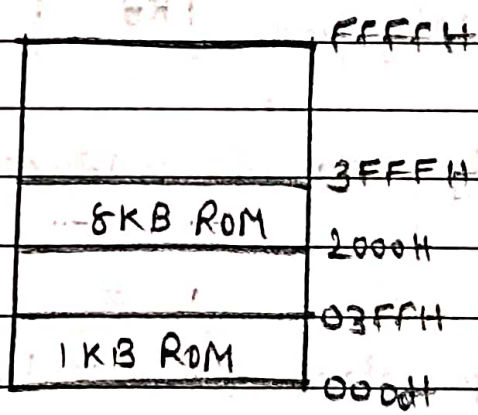
A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000H
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	03FFH

→ for chip 2 :- (8 KB RAM)

← chip select → | ← Address lines →

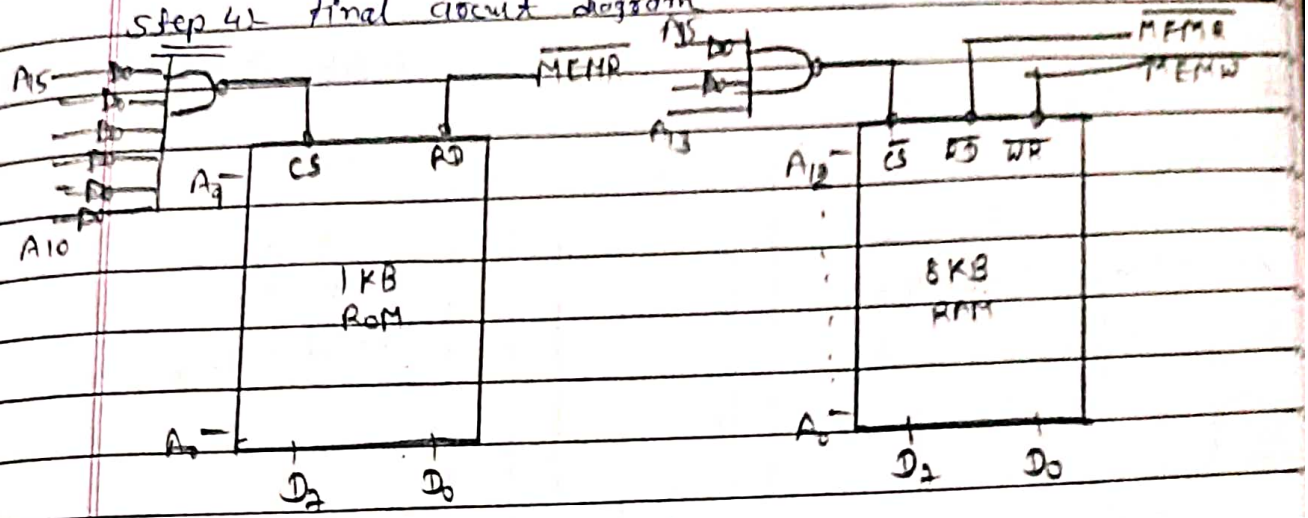
A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2000H
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3FFFH

Step 3:-

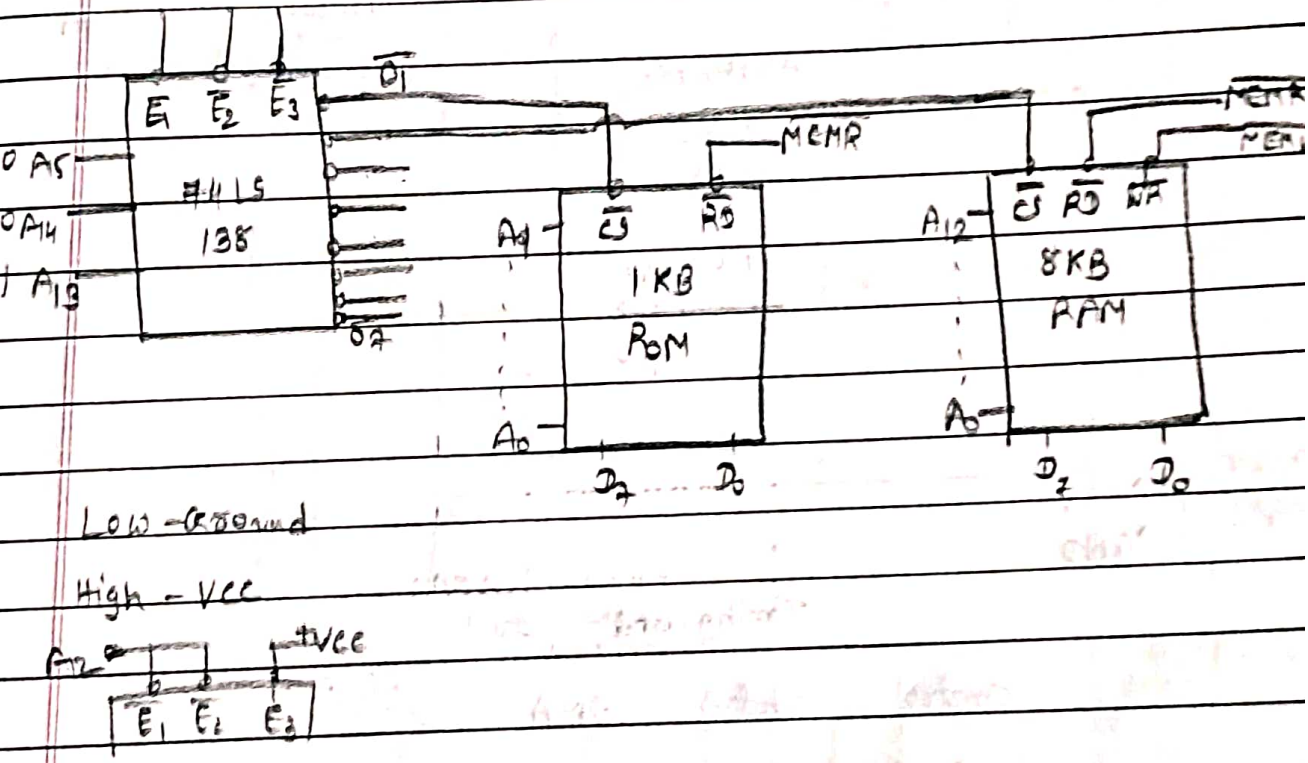


Memory Map

Step 4: Final circuit diagram

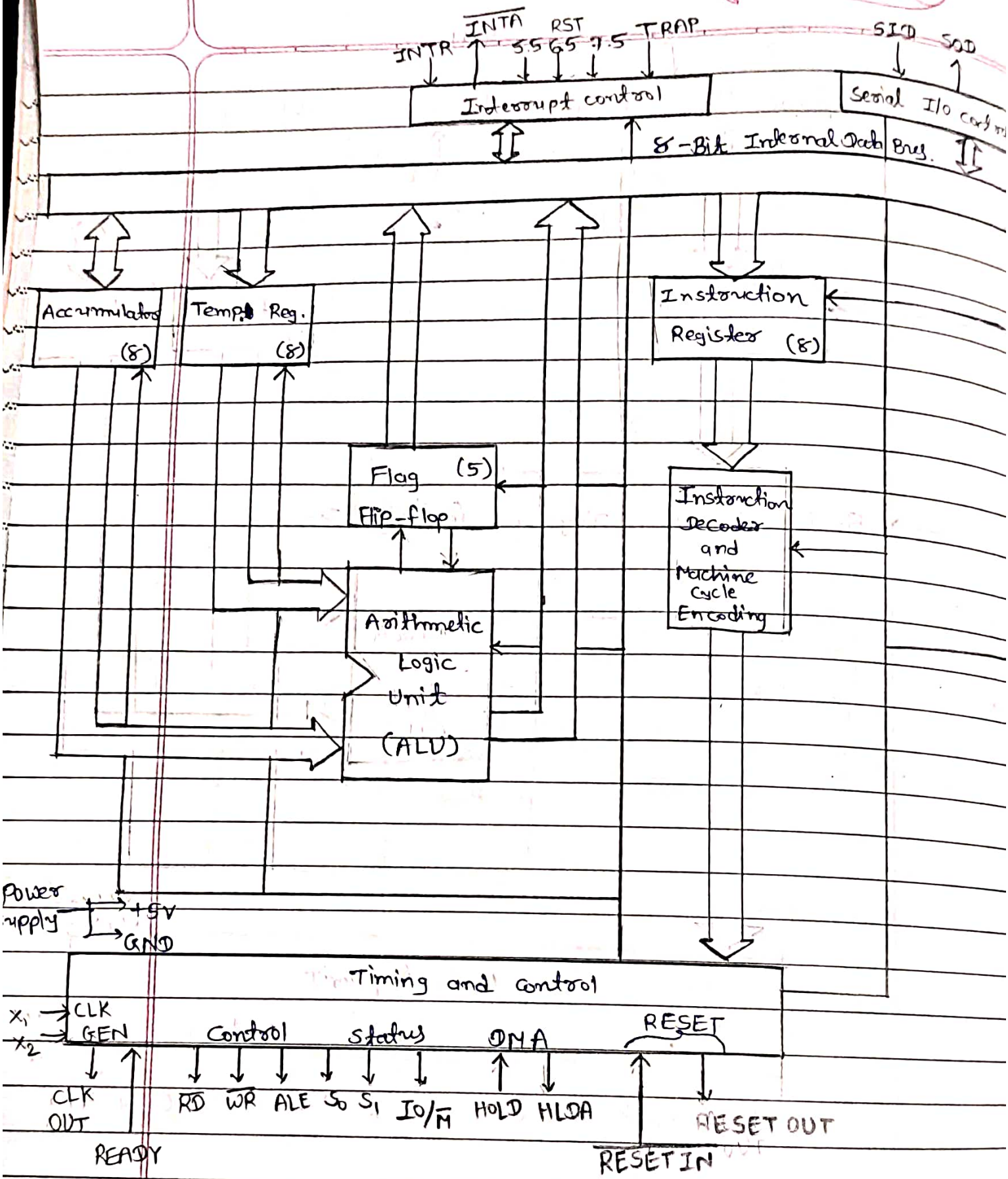


→ Final circuit Diagram (using Decoder):-



Low - Ground

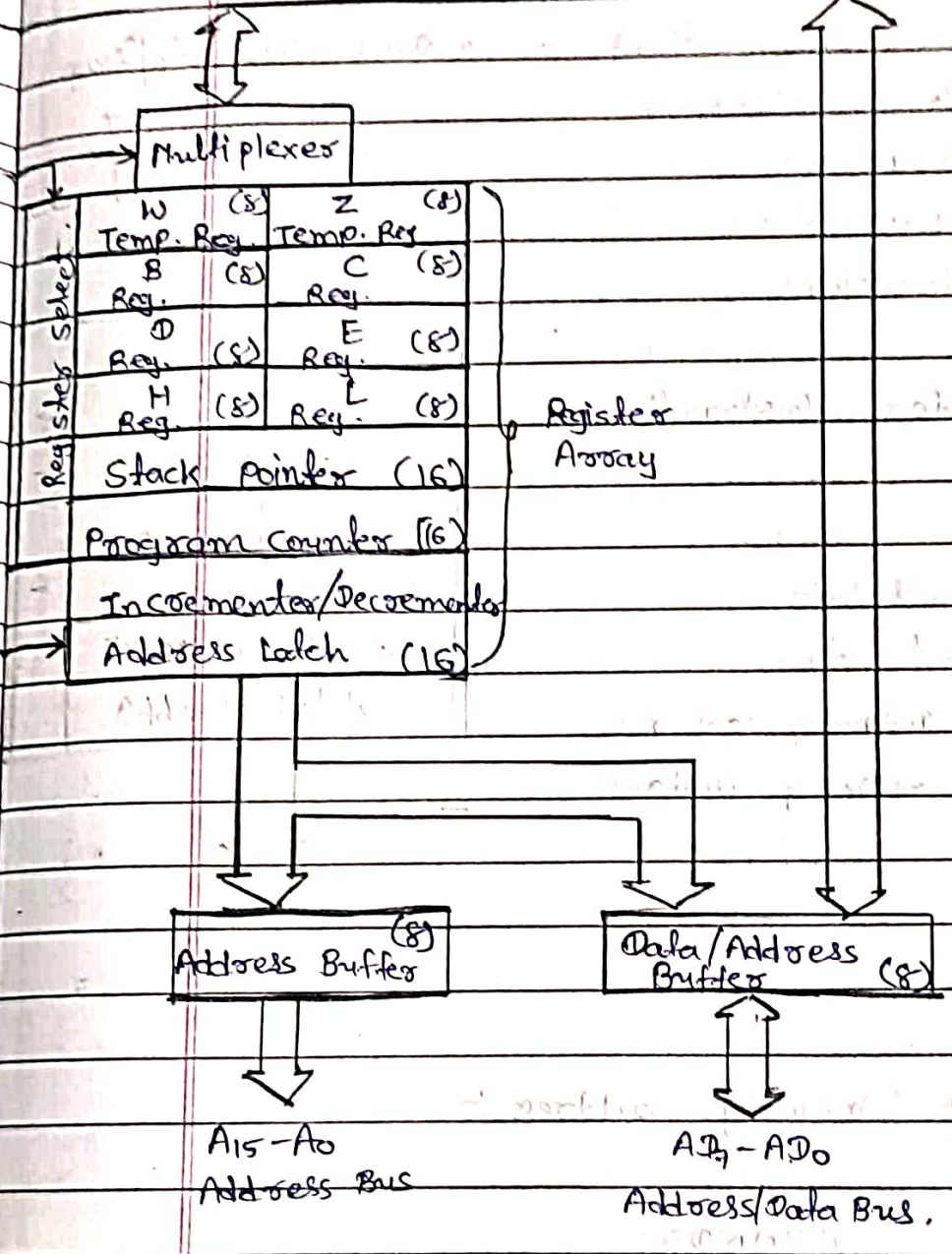
High - Vcc


INTR
INTA
RST 5.5 6.5 7.5
TRAP
SID
SOD
Interrupt control
Serial I/O control
8-Bit Internal Data Bus
Accumulator (8)
Temp. Reg. (8)
Instruction Register (8)
Flag (5) Flip-flop
Instruction Decoder and Machine Cycle Encoding
Arithmetic Logic Unit (ALU)
Power supply
+5V
GND
Timing and control
X1
X2
CLK GEN
Control
Status
DMA
RESET
CLK OUT
RD
WR
ALE
S0
S1
IO/M
HOLD
HLDA
RESET OUT
READY
RESET IN

SID SOD

Serial I/O control

Data Reg.



* Classification of Instruction :-

↳ Based on operations :-

1. Data Transfer Instructions ——→ Don't Affect any flags.
2. Arithmetic Instructions
3. Logical Instructions
4. Branching Instructions
5. Machine control Instructions.

* Data Transfer Instructions :-

1. MOV Rd, Rs
2. MVI Rd, 8-bit data
3. LXI RP, 16-bit data
4. LDA 16-bit memory address
5. STA 16-bit memory address
6. LDAX B/D
7. STAX B/D

4) LDA 16-bit memory address :-

OPCODE OPRANDS :-

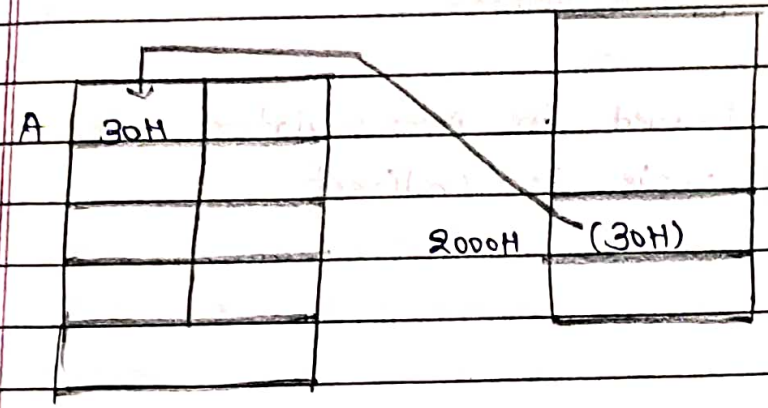
LDA 16-bit Memory address ; LDA 2000H

- Load Accumulator directly from 16-bit memory address (Location) provided in the instruction
- No flags are affected.
- Works with Accumulator only.

LxI → 00__0001.

- Addressing mode is Direct Addressing mode.
- MOV M, A → Indirect Addressing mode.

LDA 2000H



Stored the data from Memory to Accumulator.

LXI RP, 16-bit Imm. Data

5) STA 16-bit memory address:-

3 byte Instruction.

OPCODE	OPRANDS:
STA	16-bit memory address.

- Store Accumulator's data into memory location provided in the instruction.

6) LDAX: B/D:-

1 Byte instruction.

OPCODE	OPRANDS
LDAX	B/D ; B → BC Pair D → DE pair.

- Load Accumulator from the 16-bit memory location provided by the register pair (BC or DE).
- No flags are affected.

LXI Rp, 16-bit Imm. data
↳ B → BC pair
D → DE Pair
H → HL pair
SP → Stack pointer

- Data is loaded in Accumulator only.
- Addressing mode is Indirect

7) STAX B/D

1 Byte Instruction

OPCODE	OPRANDS	
STAX	B/D	; B → BC pair
		D → DE pair

- store Accumulator's data into memory location provided by the register pair (BC or DE).

* Branching Instructions:-

↳ Conditional
↳ Unconditional → JMP

1) Unconditional Jump:-

fixed with 8 bit so it 2 bytes

OPCODE	OPRANDS	
JMP	16-bit Memory Location	; JMP NEXT (Label)

- Jumps unconditionally to the given memory

location of the Program
- No flags are affected.
- 3 byte Instruction.

Ex:- START :- LDA 2000H
STA 3000H
JMP START

Conditional Jump:-
1.) JZ 16-bit memory Location Jump on Zero.
2) JNZ 16-bit memory Location
3) JC 16-bit memory Location
4) JNC 16-bit memory Location.

1.) JZ :- Jumps when zero flags is set. (Z=1),
Jump on zero.
- No flags are affected.

2) JNZ :- Jump when non zero flags is set (Z=0).
Jump non zero.
- No flags are affected.

3) JC :- Jump when carry flags is set (CY=1)
Jump on carry.
- No flags are affected.

4) JNC :- Jump when non carry flags is set (CY=0)
Jump non carry
- No flags are affected.

* Arithmetic Instructions :-

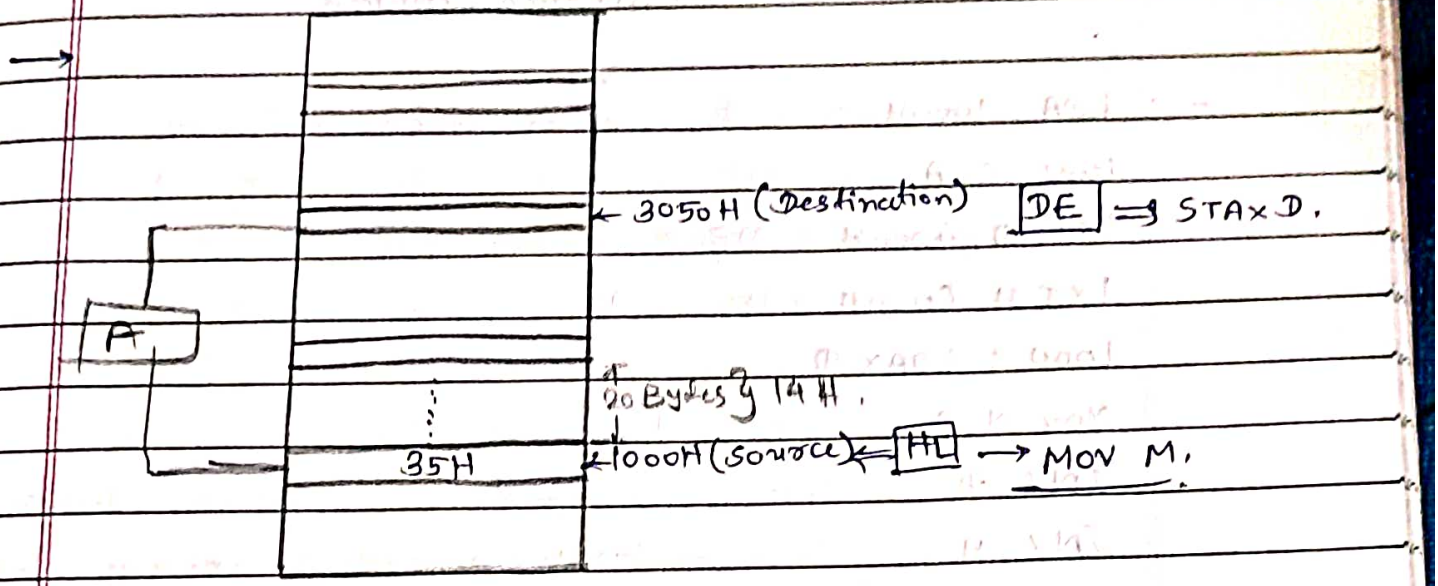
1) OPCODE OPRANDS
ADD R/M

- ADD the 8-bit data (content) of Register or memory location (provided by HL pair) with Accumulator.
- Result is stored in Accumulator.
- It is a one-byte instruction.
- All flags are affected
- Accumulator is by default one of the operands

* INR R/M :- Increment the content of Register or memory location (provided by HL pair) by 1.
- All flags except Carry flags are affected
- One byte Instruction
- If the data is FF ⇒ the register data will be 00H after increment.

* DCR R/M :- Decrement the content of Register or memory location (Provided by HL pair) by 1.
- All flags except carry flags are affected
- One byte Instruction
- If the data is FF → so it will FEH after decrement.

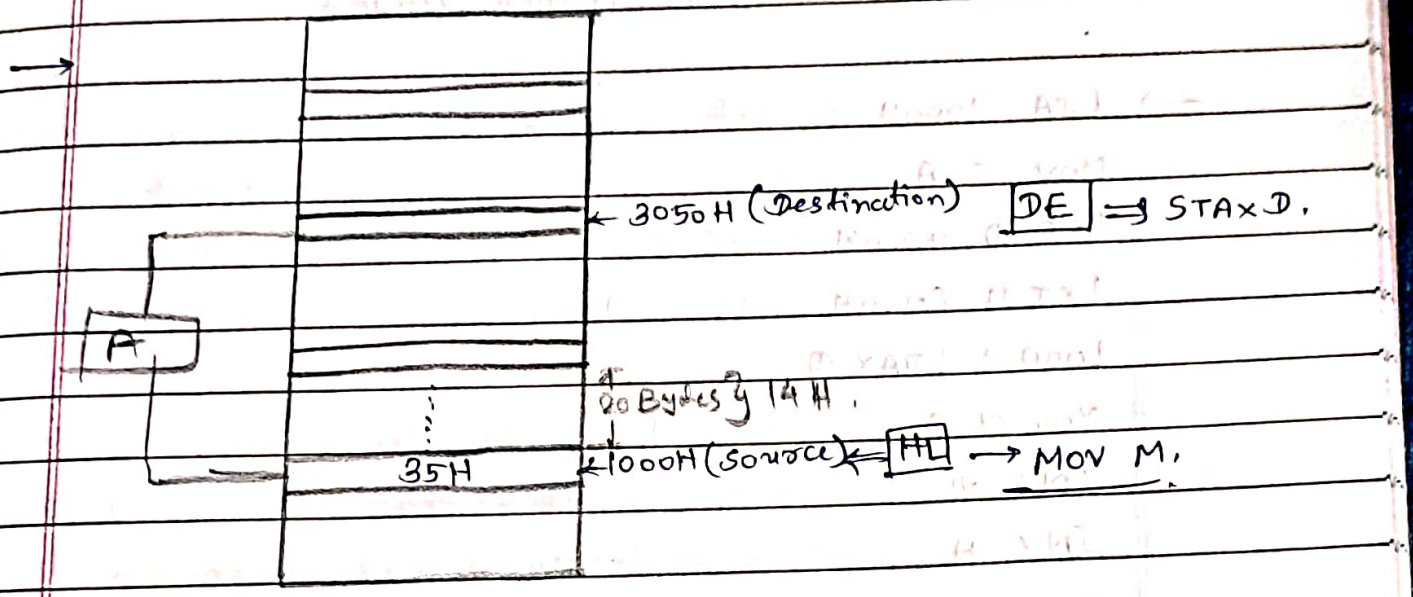
Prog: Write a program to transfer a block of data from memory location 1000H to memory location 3050H. Consider the size of block is 20 bytes. 50



LXI H, 1000H ; HL = source starting address (1000H).
LXI D, 3050H ; DE = destination starting address (3050H).
MVI C, 14H load counter = 20 (14H = 20 decimal).
LOOP: MOV A, M; copy data from memory (source) to A.
STAX D ; Store A into memory at DE (destination).
INX H ; Increment register pair by 1.
INX D ; Move to next destination address.
DCR C ; Reduce counter
JNZ LOOP ; Jump back if counter not zero.
HLT ; when counter = 0.

A			
			→ 51H
D	30H	50H	E (305
H	10H	00H	L (100
		→ 01H	

Ques: Write a program to transfer a block of data from memory location 1000H to memory location 3050H. Consider the size of block is 20 bytes. 50



LXI H, 1000H ; HL = source starting address (1000H).
LXI D, 3050H ; DE = destination starting address (3050H).
MVI C, 14H ; load counter = 20 (14H = 20 decimal).
LOOP: MOV A, M ; copy data from memory (source) to A.
STAX D ; Store A into memory at DE (destination).
INX H ; Increment register pair by 1.
INX D ; Move to next destination address.
DCR C ; Reduce counter
JNZ LOOP ; Jump back if counter not zero.
HLT ; when counter = 0.

A			
			→ 51H
D	30H	50H	E (3050H)
H	10H	00H	L (1000H)
		→ 01H	

7. counter data instruction
complete the table
find memory location

Ques:- Transfer a block of data from source memory location 4000H to destination memory location C000H. Consider ~~of~~ the size of the block is stored at the memory location 1000H. use HL pair as Destination pointer.

→
```
LDA 1000H ; Get size of block → Accumulator = N
MOV C, A  ; put counter into c register
LXI D, 4000H ; DE = Source pointer (4000H)
LXI H, C000H ; HL = Destination pointer (C000H)
Loop : LDAX D ; Get Data from source (DE)
MOV M, A  ; Store data into destination (HL)
INX D     ; Increment DE → next source address
INX H     ; Increment HL → next destination address.
DCR C     ; Decrease counter
JNZ Loop  ; Repeat until C=0
HLT       ; stop
```

* Data Transfer with I/O :-

1. IN
2. OUT

1. IN :-

OPCODE	OPERANDS
IN	8-bit port Address

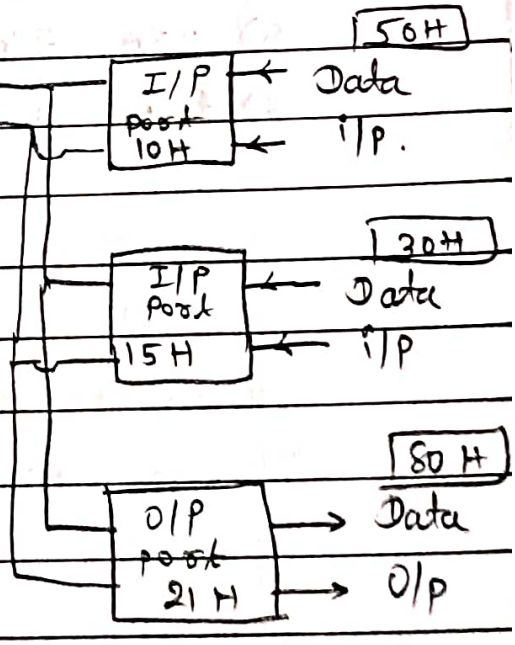
2. OUT :- OUT 8-bit Port Address.

- It is a two byte Instructions.
- It loads data from i/p port into Accumulator only.
- No flags are affected

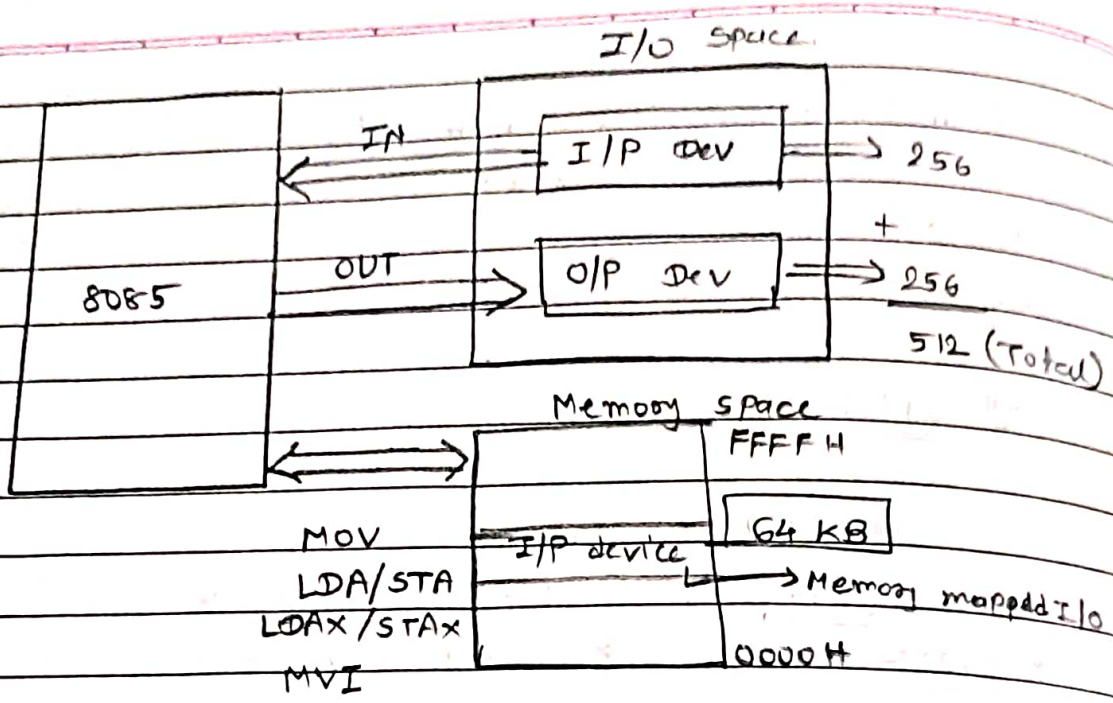
Ex: Load 2 bytes from the Input ports 10H and 15H into 8085 add this bytes and send the result on output port 21H.

→ IN 10H
MOV B, A
IN 15H
MOV C, A
ADD B
OUT 21H
HLT

A	50H		
B	50H	30H	C
D			E
H			L



Diff. between Memory Mapped I/O and I/O Mapped I/O.



Logical Instructions:-

AND OR XOR Carry.

1) ANA R/M
2) ANI 8-bit Imm. data
3) ORA R/M
4) ORI 8-bit Imm. data.
5) XRA R/M
6) XRI 8-bit Imm. data

Q- Mask all even bits of a data byte recived from port 01 H

Q- UnMask bit D4 to D7 in this above data byte

Q- Complement all odd bits of the above data byte finally send the updated byte to output port 11 H.

IN 01H
ANI 0AAH
OUT 02H
HLT

IN 01H
ANI 0FFH
OUT
HLT

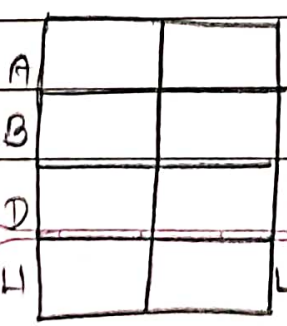
* Arithmetic Instructions :-

1) ADD
2) ADI — 8 bit Imm. data.
3) ADC
4) ACI

3) OPCODE OPERANDS
ADC R/M

- Add the contents of register or memory location (provided by HL pair) with register A, with carry.
- Result is stored in register A.
- All flags are affected
- It's a one byte instruction

Ex: Add 2 16 bit data loaded in BE and HL pair store the result in BC pair as well on memory locations 1001 H (lower byte) and 1002 H (upper byte).



5.) INR R/M
6.) DCR R/M
} These instructions affects all flags except carry flag.

Ex:- Increment a content of memory location 2000H by 1

```
LXI H, 2000H
INR M
INR L
INR M
```

H	20	01	H	L

7.) INX Rp
8.) DCX Rp.
- Increment / Decrement the register pair by 1.
- No flags are affected.
- Rp can be BC, DE, HL pair & Stack pointer.
- It is one byte Instruction.

```
LXI B, 10FFH
INX B ⟶ 11, 00, H
```

B C

```
  0001 0000 1111 1111
+                    1
  ---------------------
  0001 0001 0000 0000
```

1 1 0 0

9.) SUB R/M
10.) SUI 8-bit Imm. data
11.) SBB R/M
12.) SBI 8-bit Imm. data

SUB, SUI } - Subtracts the operands from reg A.
- result is stored in reg A.
- All flags are affected

SBB, SBI } - Subtracts the operands & carry (borrow) flag from reg. A.
- result is stored in reg. A.
- All flags are affected
- Subtraction is performed with 2's complement method internally.

SUB R/M ⇒ A
− R/M
A ← Result.

SBB R/M ⇒ A
R/M
Borrow (Carry)/A

(05) − (03) = +(02)

Step:-1 0000 0011
1111 1100 ← 1's (or -1)'s Comp.
+ 1
1111 1101 ← 2's comp. of (03 H)

Step:-2 Add it with (05) H:-
0000 0101
+ 1111 1101
1) 0000 0010

Step:3 Complement the carry in step: 2

After complementing :-
- If carry = 0 ; Result is positive & in correct magnitude form.
- If carry = 1 Result is negative and in r's complement form.

Ex:- (03) − (05) = −(02).

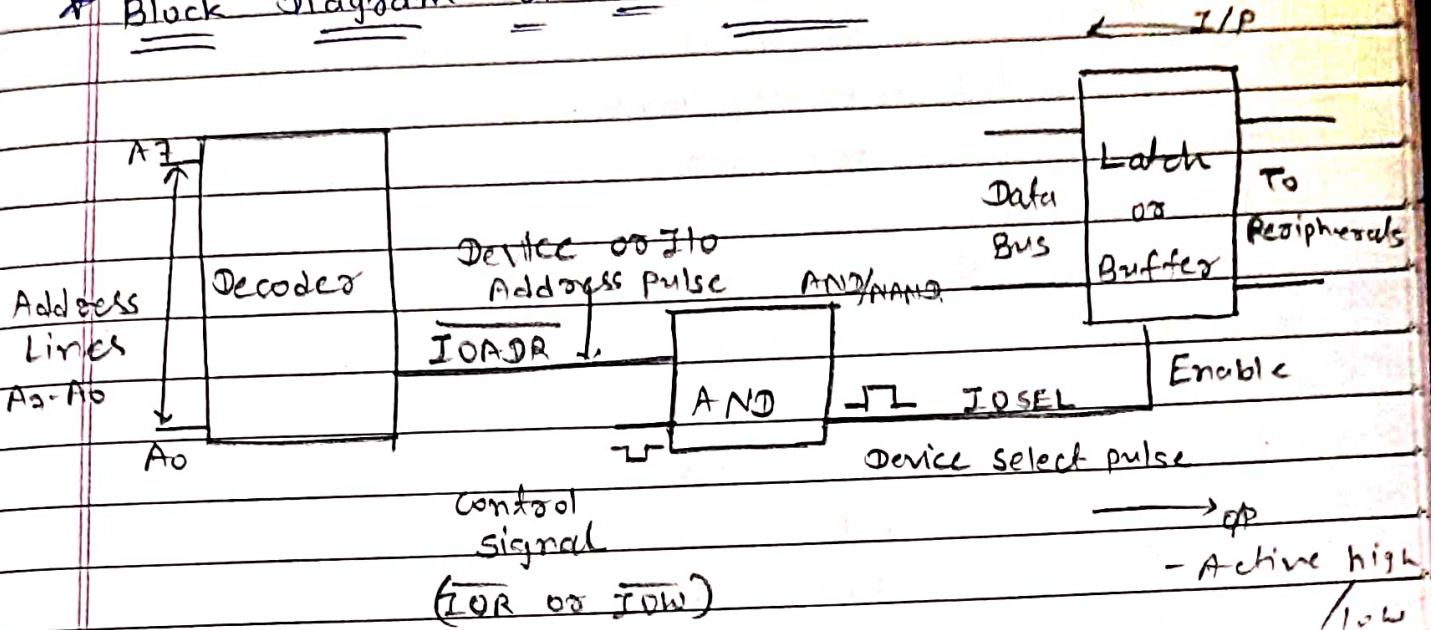
→ Step:1
```
  0000 0101
  1111 1010  ← r's (r-1)'s comp
+        1
-----------
  1111 1011  ← r's comp.
```

Step:2 Add it with (03) H.
```
  0000 0011
+ 1111 1011
-----------
 0 1111 1110
```

Step:3 After complement
Carry = 1.
→ Take r's comp- to find actual magnit

```
  1111 1110
+ 0000 0001
-----------
  0000 0010 ⟶ −(02)
```

Block Diagram of I/O Interface :-



Decode Logic for LED output port. (Hardware for O/P Port ~~Instruct~~ Interface :-

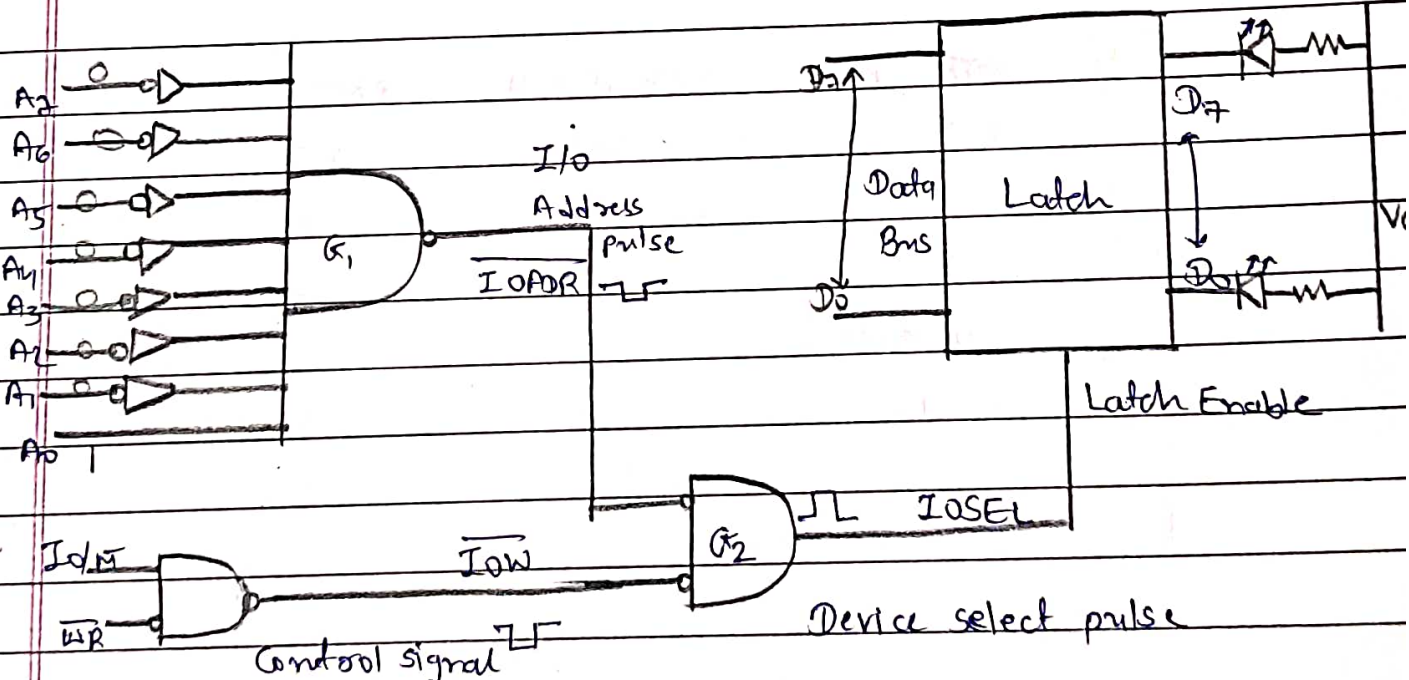


Figure 5.6

Control

If 0 so LED is ON

Table : 5 : 1, Page. No 162.

switch

Question Paper of Mid-1

Answer

Q-4 1) (i) LXI H, 1050H
MOV A, M
LXI H, 4010H
MOV M, A

(ii) LDA 1050H
STA 4010H

(iii) LXI B, 1050H
LDAX B
LXI D, 4010H
STAX D

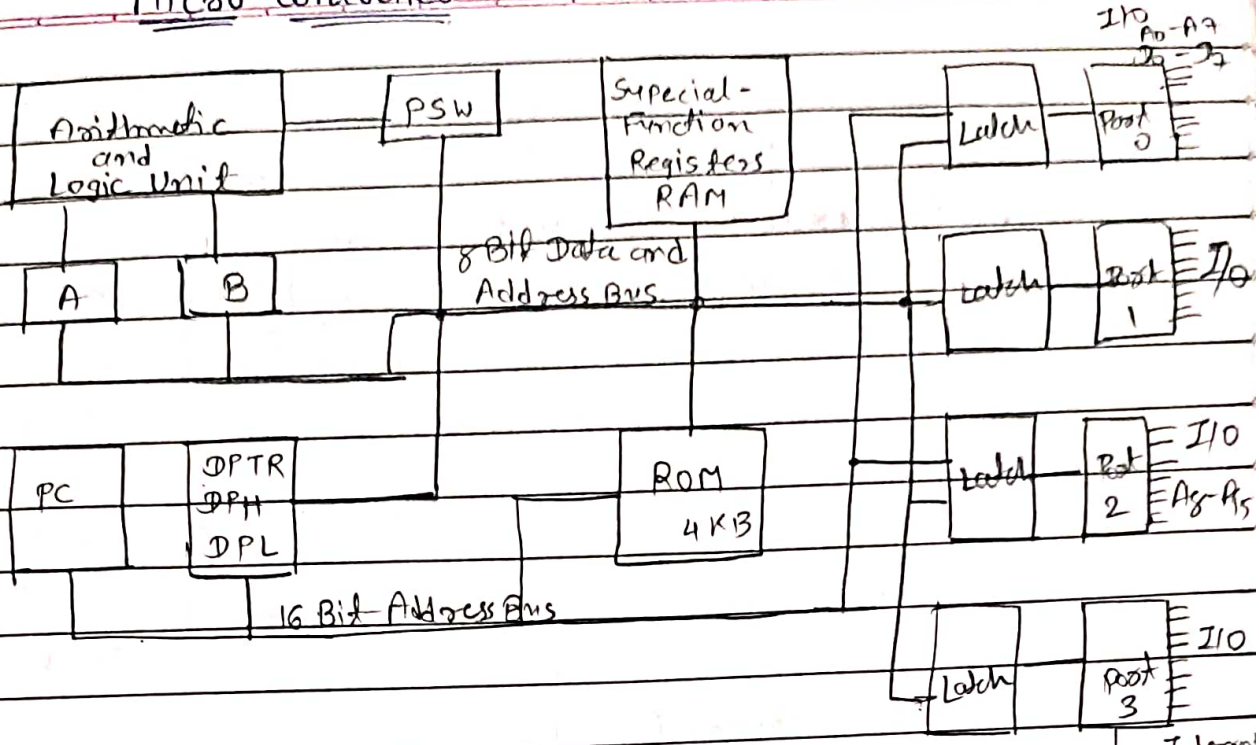
2) 55H → Masking
96H → UnMasking

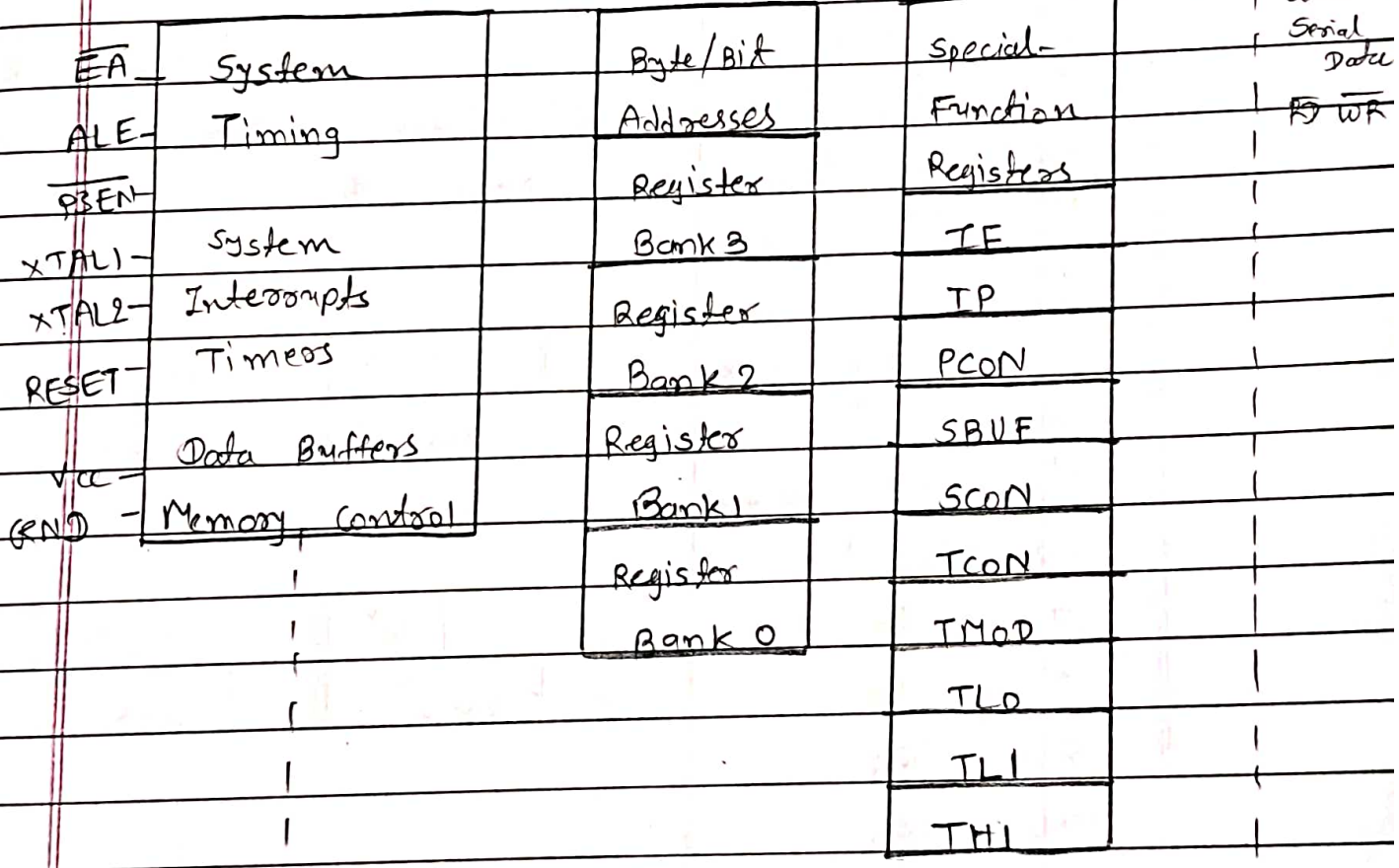
Q-4 1) XRA A , MVI A, 00H , ANI 00H , SUB A
1 Byte 2 Byte 2 Byte 1 Byte

2) MOV C, M
ADD L

8051

Micro controller





Internal RAM structure

Block Diagram of 8051

8051 Programming Model

8	E0*
A Register	

8	F0*
B Register	

Math Registers

8	B8*
IP Register	

8	A8*
IE Register	

Interrupt Register

8	8C
TH0 counter	

8	8A
TL0 counter	

Time/Counter Registers

8	98*
SCON Register	

8	99
SBUF Register	

Serial Data Registers

8	81
Stack Pointer	

8	83	8	82
Data Pointer DPH			DPL

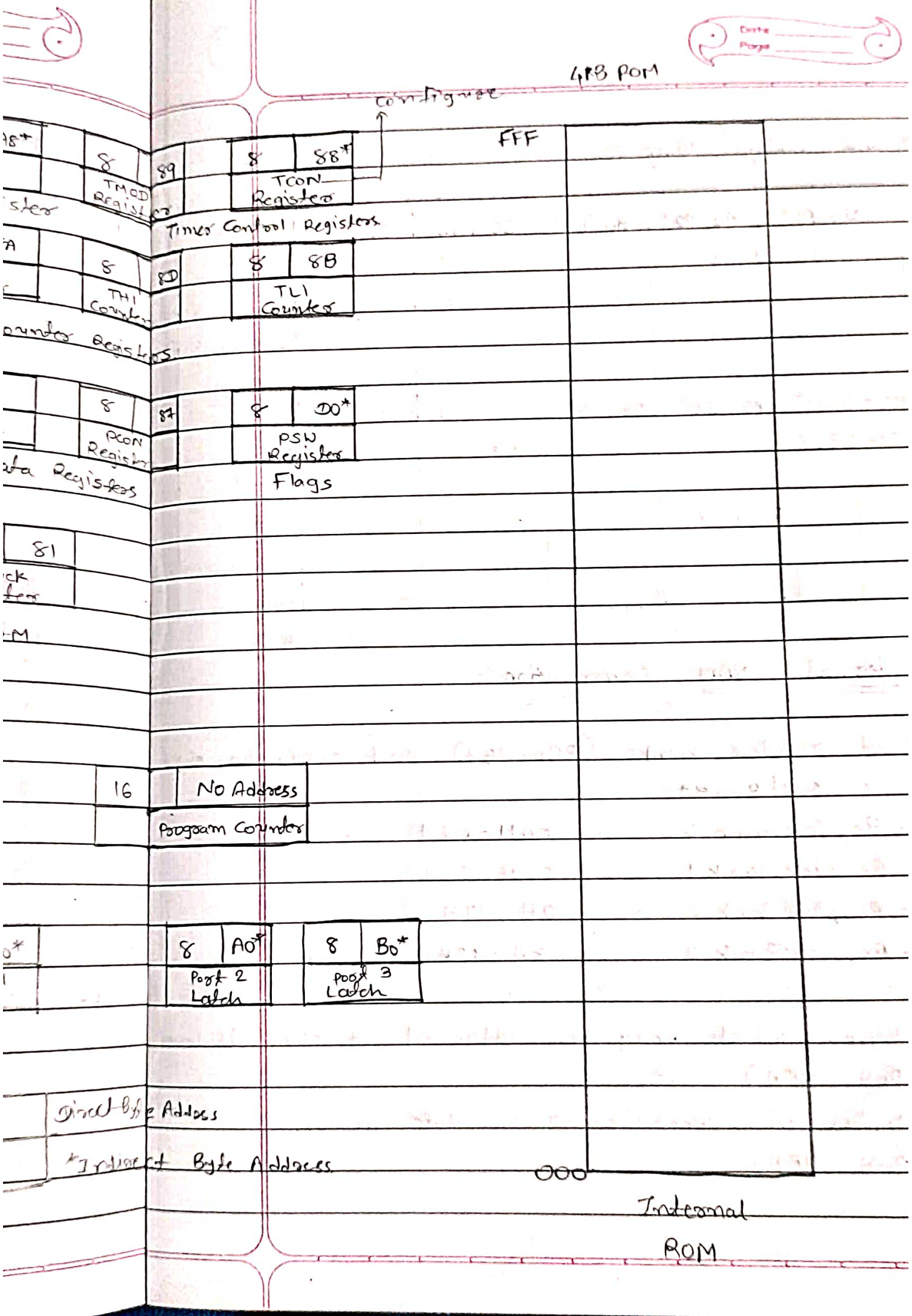
8	80*
Port 0 Latch	

8	90*
Port 1 Latch	

Byte Address	Internal RAM	
7F		
	General-Purpose Area	
30		
2F	Bit Address Area	7F
20		00
1F	Register Bank 3	
18		
17	Register Bank 2	
10		
0F	Register Bank 1	
08		
07	R7	
06	R6	
05	R5	
04	R4	
03	R3	Register Bank 0
02	R2	
01	R1	
00	R0	

Bit Address for this RAM Area only.

Number of Bits		
		Direct Bit
		*Indirect


4KB ROM
configure
FFF
8 | 88*
TCON Register
8 | 89
TMOD Register
Timer Control Registers
8 | 8B
TL1 Counter
8 | 8D
TH1 Counter
Counter Registers
8 | D0*
PSW Register
Flags
8 | 87
PCON Register
Data Registers
81
16 | No Address
Program Counter
8 | A0*
Port 2 Latch
8 | B0*
Port 3 Latch
Direct Byte Address
*Indirect Byte Address
000
Internal ROM

* SFR Address logic :-

80 - P0*	90 - P1*	A0 - P2*	B0 - P3*	D0 - PSW*	E0 - A*	F0 - B*
81 - SP						
82 - DPL						
83 - DPH						
87 - PCON						
88 - TCON*	98 - SCON*	A8 - IE*	B8 - IP*			
89 - TMOD	99 - SBUF					
8A - TL0						
8B - TL1						
8C - TH0						
8D - TH1						

* Internal RAM Organization :-

→ Four register banks (RB0-RB3), each containing eight registers R0-R7

- Register Bank 0 — 00H - 07H
- Register Bank 1 — 08H - 0FH
- Register Bank 2 — 10H - 17H
- Register Bank 3 — 18H - 1FH

→ 16 bytes, which may be addressed at the bit level (20H - 2FH)

→ 80 bytes of general-purpose data memory (30H - 7FH)

* Internal RAM organization:-

	1F	R7
	1E	R6
Bank 3	1D	R5
	1C	R4
	1B	R3
	1A	R2
	19	R1
	18	R0
	17	R7
Bank 2	16	R6
	15	R5
	14	R4
	13	R3
	12	R2
	11	R1
	10	R0
	0F	R7
	0E	R6
Bank 1	0D	R5
	0C	R4
	0B	R3
	0A	R2
	09	R1
	08	R0
Bank 0	07	R7
(Default Bank working Registers)	06	R6
	05	R5
	04	R4
	03	R3

7F

Bit address 2FH

16 × 8 = 128 Bits

Registers		
2F	7F	78
2E	77	70
2D	6F	68
2C	67	60
2B	5F	58
2A	57	50
29	4F	48
28	47	40
27	3F	38
26	37	30
25	2F	28
24	27	20
23	1F	18
22	17	10
21	0F	08
20	07	00

Bit Addressable 30

02 01 00
R2 → R2 R1 R0

General Purpo

RS - Register Bank select Bit

↓ (Q P0H)

↓ 00H

* The 8051 Flags and PSW

PSW.7	PSW.6	PSW.5	PSW.4	PSW.3	PSW.2	PSW.1	PSW.0
CY	AC	F0	RS1	RS0	OV	—	P

$C_7 \oplus C_6$ — overflow

Set / Reset Based on content of Reg A

8051 P: odd = 0. P = Even

RS1	RS0	Mode of operation	RAM address
0	0	Select Register Bank 0	00H - 07H
0	1	Select Register Bank 1	08H - 0FH
1	0	Select Register Bank 2	10H - 17H
1	1	Select Register Bank 3	18H - 1FH

8085	8051
CY	CY
AC	AC
P	P
Z	F0
S	OV

P → complement

→ MOV A, # 53 H

→ MOV PSW, # 08 H → Select Reg. Bank and clear all Flags.

```
MOV  A, # 35 H     // Load Immi. data 35H in Accumulator
MOV  B, A          // Copy Reg. A contents into Reg. B
MOV  R0, # 60 H    // Load Imm. data 60H in reg. R0 of Bank 0
MOV  PSW, # 08 H   // Load 08H in PSW to select reg. Bank 1
MOV  R0, # 70 H    // Load 70H Immi. data in R0 of Bank 1
MOV  PSW, # 18H    // Load 18H in PSW to select reg. Bank 3
MOV  R1, # 80 H
```

Ch:-3 Moving Data

2→ Addressing Modes :-

MOV dest, source

1.) Immediate Addressing Mode
2) Register Addressing Mode
3) Direct Addressing Mode
4.) Indirect Addressing Mode

1.) Immediate Addressing Mode :-

OPCODE	OPERANDS	COMMENTS
MOV	Rx, #n ;	Load an 8-bit Imm. data (n) into reg. Rx (R0-R7 of current active reg. bank)
MOV	A, #n ;	
MOV	DPTR; #nn;	Load 16-bit Imm. data (nn) into DPTR

21 → SFR
11 → Bit Addressable
10 → Byte Addressable

MOV DPTR, #3495H ;
↓ ↓
DPH DPL

MOV SFR, #n ;

MOV P1, #0FH

2) Register Addressing Mode :-

OPCODE	OPERANDS	COMMENTS
MOV	A, Rx	; copy contents of Rx (R_0-R_7 of current active reg Bank) into Accumulator
MOV	Rx, A	

Caution : MOV Rx, Rx ✗ Invalid

Ques:- Swap the contents of Register R2 and R4 of reg. Bank1 using Register Mode only.

→
```
MOV PSW, #08H
MOV A, R4
MOV R3, A
MOV A, R2
MOV R4, A
MOV A, R3
MOV R2, A
END
```

Ques:- Swap the contents of Register R_2 of Bank1 and R4 of Bank2 using Register Addressing Mode Load the nessesary data into register Before swapping.

```
MOV  PSW , #08H
MOV  R2 , #20H
MOV  A , R2

MOV  PSW, #10H
MOV  R4 , #21H
MOV  R3, A
MOV  A, R4

MOV  PSW , #08H
MOV  R2 , A
MOV  PSW , #10H
MOV  A , R3
MOV  R4 , A
END
```

3.) Direct Addressing Mode:-

OPCODE	OPERANDS	COMMENTS
MOV	add, #n	; Load 8-bit imm. data (n) into the addressed reg.
MOV	A, add	
MOV	add, A	
MOV	Rx, add	
MOV	add, Rx	
MOV	add1, add2	

MOV 81H, #30 ; Initialize Stackpointer to 30H.

$R_0 \rightarrow 02H$ MOV 02H, 30H

$R_4 \rightarrow 04H$ MOV 04H, 50H

MOV A, 02H MOV R_3, 02H MOV 05H, 02H

MOV 02H, 04H

MOV 04H, A

$R_0 \rightarrow$ Bank1 MOV 02H, 30H

$R_4 \rightarrow$ Bank2 ;

4] Indirect Addressing Mode :-

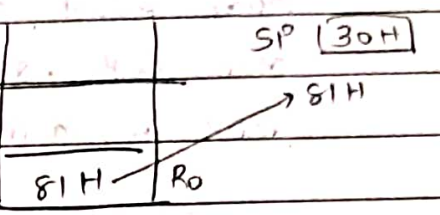
→

OPCODE	OPERANDS	COMMENTS
MOV	@Rp, #n	; Load an imm. 8-bit data(n into the reg. Pointer by Rp (R0 or R1 of current active reg. Bank)
MOV	@Rp, A	; copy the contents of Reg. A into the reg. Pointed by Rp (R_0 or R_1...)
MOV	@Rp, add	
MOV	add, @Rp	
MOV	A, @Rp	

MOV R0, #81H ; Initialize pointer R0 to address of SP

MOV @R0, #30H ; Load 30H data into SP using indirect add mode (R0 is pointing to 81H).

↳ At the address pointed by R0.



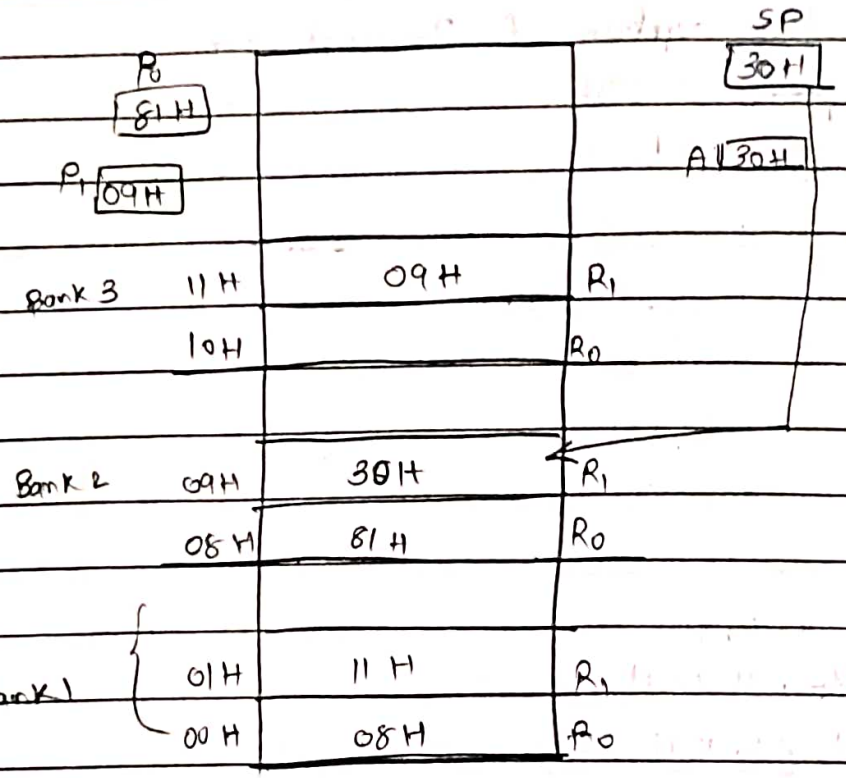
* Swap the value of R_2 and R_4

```
MOV   R0, #02H
MOV   R1, #04H
MOV   R2, #20H
MOV   R4, #30H
MOV   A, @R0
MOV   02H, @R1
MOV   @R1, A
```

address

```
MOV   R0, #08H   Load imm. data 08H into R0
MOV   R1, #11H   Load imm. data 11H into R1
MOV   A, #09H    Load data into Accumulator (A)
MOV   B, #81H    Load data into reg. B
MOV   @R0, 0F0H
MOV   @R1, 0E0H
MOV   80H, 08H
```

```
MOV  90H, 11H
MOV  PSW, #08H
MOV  @R0, #30H
MOV  0D0H, #10H
MOV  @R1, 81H
MOV  A, @R1
```



✱ External Data Moves :-

OPCODE	OPERANDS	COMMENTS
MOV X	A, @Rp	; Copy the data from the address pointed by Rp (R0 or R1 of current Active Reg. bank) of external RAM into the reg A.

@Rp → 8-bit pointer

@DPTR → 16-bit pointer.

00 – 7F } Indirect addressing

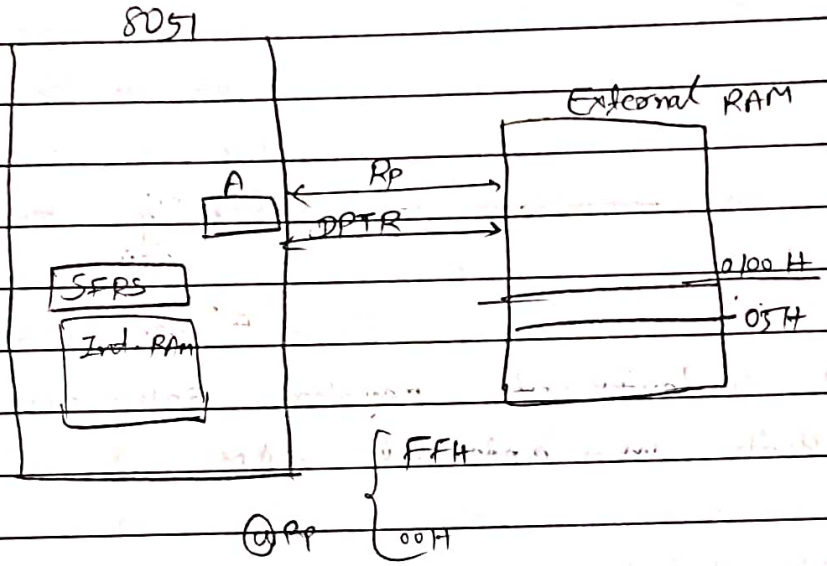
Data Page

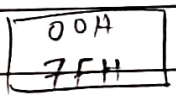
80 – FF, Bit addressing area.

MOVX A, @DPTR ;

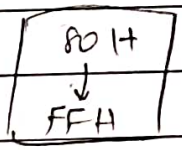
MOVX @Rp, A

MOVX @DPTR, A





128 Reg.



128 address.

→ 21 SFRs

↳ 21 address used.

107 Address are unused at Byte Level.

* Swap the contents of Register 10H in external RAM and Register 1FH in internal RAM

→
```
MOVX A, @DPTR
MOV R0, A
```

```
MOV R0, #10H
MOV R1, #1FH
MOVX A, @R0
MOV 02H, A
MOV A, @R1
MOV @R1, 02H
MOVX @R0, A
```

* Swap the contents of Register 50H and Register 1000H both in external RAM

→
```
MOV DPTR, #50H
MOVX A, @DPTR
MOV R0, A
MOV DPTR, #1000H
MOVX A, @DPTR
MOV R1, A
MOV A, R0
MOV DPTR, #1000H
MOVX @DPTR, A
MOV A, R1
MOV DPTR, #50H
MOV @DPTR, A
END
```

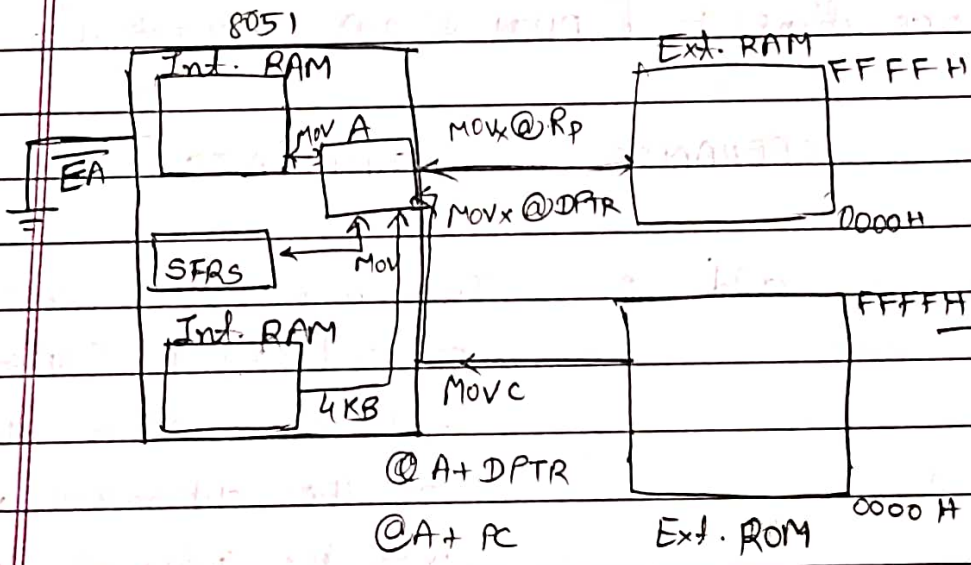
Reading Code Memory :-

OPCODE	OPERANDS	COMMENTS
MOV C (↳ Code Memory)	A, @A+DPTR; (→ indexed pointing)	Copy the contents from the address (in ROM) pointed by A+DPTR into reg. A

→ Only Accumulator is allowed

MOV C A, @A+PC;

NOTE: PC gets incremented to point to next instruction before added with A.



→ If EA is grounded, then Ext. ROM will be accessed.

→ If EA is not grounded, then int. ROM will be accessed (up to 4KB) & beyond that Ext. ROM will be accessed

Ques:- Read the code memory from location 2000H and 2001H and send that data to port no. 1 & port 2 respectively.

```
→ MOV DPTR, #2000H
  MOV A, #00H
  MOVC A, @A+DPTR
  MOV P1, A
  MOV A, #01H
  MOVC A, @A+DPTR
  MOV P2, A
  END
```

A → DPTR

offset		
		2000 + A ← 5 = 2005H
		2005H
	:	
	04H	2002H
	01H	2001H
	00H	2000H

Look up Table Method

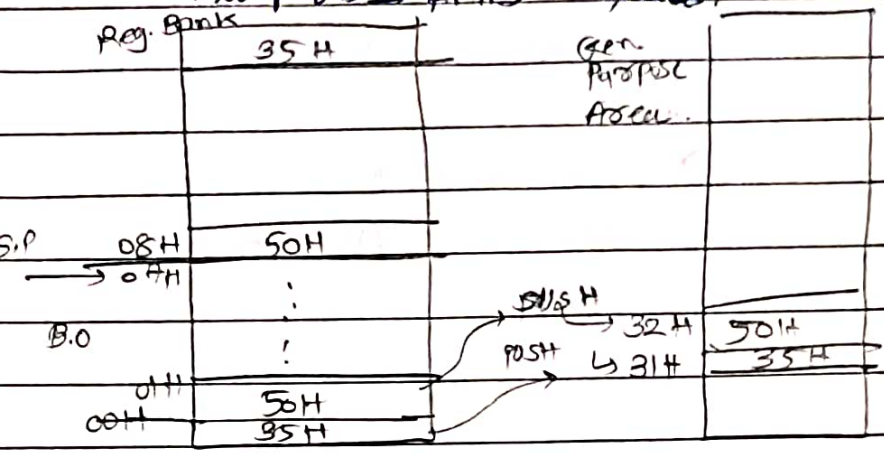
↳ Base of the table

* Stack operations :- [PUSH & POP operations].

OPCODE	OPERANDS	COMMENTS
PUSH	add ;	Increment the stack pointer content by 1 (SP ← SP+1) and then store the contents of the addressed register into the internal RAM address pointed by SP.
POP	add ;	Fetch the byte from the address of internal RAM pointed by SP into addressed register & then decrement SP by 1.

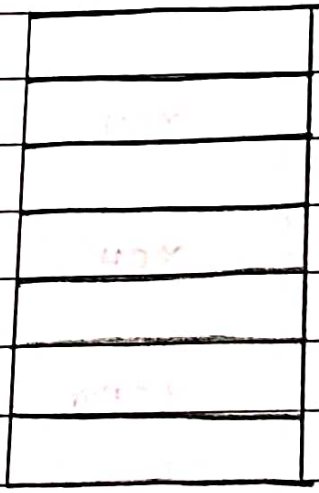
(MOV SP, #30H) { SP ⇒ 07H, All ports ⇒ FFH } Default Reset Status.

```
MOV 81H, #30H
MOV R0, #35H
PUSH 00H
MOV R1, #50H
PUSH 01H
POP 08H
POP 1FH
```



Ques:- Swap the content of R2 of Bank1 & R6 of Bank 3 using stack operation

Ques:- Rotate the contents of Registers R0 → R1 ; R1 → R2 ; R2 → R3 & so on finally R7 → R0 using stack operation only. Use Register bank 0.



→
```
MOV SP, #30H
PUSH 00H ; Send contents of R0
           of Bank 0 on stack
PUSH 01H ;
PUSH 02H ;
PUSH 03H ;
  :
PUSH 07H ;
POP 00H ;
POP 07H ;
POP 06H ;
POP 05H ;
  :
POP 01H ;
```

Ques:- Consider that the controller is by default reset status. If the 8 POP operation are perform. what will be the content in stack pointer.

→ by default → 07H

FFH (07H - 08H).

SP ⇒ 07H

POP ⇒ 06H

POP ⇒ 05H

⋮

POP ⇒ 00H

SP ⇒ FFH.

* Exchange Operations :-

OPCODE	OPERANDS	COMMENTS
XCH	A, Rx	; Exchange swap the contents of reg A & reg. Rx.
XCH	A, add	;
XCH	A, @Rp	;
XCHD	A, @Rp;	Exchange the lower nibble of reg. A & the reg. pointed by Rp.

Ques:- Swap the contents of register R2 and register R4 of register bank 0 using exchange operation.

→
```
MOV R2, #30H
MOV R4, #20H
XCH A, R2
XCH A, R4
XCH A, R2.
```

```
MOV 02H, #15H
MOV 04H, #30H
XCH A, 02H
XCH A, 04H
XCH A, 02H
```

```
MOV R0, #02H
MOV R1, #04H
MOV R2, #30H
MOV R4, #20H
XCH A, @R0
XCH A, @R1
XCH A, @R0
```

In use XCHD use as a ↓

```
MOV A, #59H
XCHD A, @R0
```

A	R2
59H	20H
59H	29H
A	R2.

Ch:- 4

Logical operations

↳ Byte Level
↳ Bit Level

NOTE:- No flags will be effected by logical instructions unless the destination address is directly PSW.

[A] Byte Level operations :-

[A] Logical AND operations :-

OPCODE	OPERANDS	COMMENTS
ANL	A, #n ;	Logicall AND the Accumulator with imm. data (n) & store result in reg. A
ANL	A, Rx ;	
ANL	A, add ;	
ANL	A, @Rp ;	
ANL	add, #n ;	
ANL	add, A ;	

→ Mask the even bits of port 1 and Mask the odd bits of DPL

```
ANL 90H, #0AAH
    or
MOV A, #0AAH
ANL 90H, A.
```

[2] Logical OR operations :-

OPCODE	OPERANDS	COMMENTS
ORL	A, #n ;	Logical OR the Accumulator with imm. data (n) & store result in reg. A.
ORL	A, Rx ;	
ORL	A, add ;	
ORL	A, @Rp ;	
ORL	add, #n ;	
ORL	add, A ;	

[3] Logical XOR operations :-

OPCODE	OPERANDS	COMMENTS
XRL	A, #n ;	
XRL	A, Rx ;	
XRL	A, add ;	
XRL	A, @Rp ;	
XRL	add, #n ;	
XRL	add, A ;	

0 1 0 0 1 1 0 0

4CH

Ques Unmask the bits P_2 to P_5 and P_6 of register B_0 20H using single instruction.

→ ORL 20H, #4CH

Ques Complement the contents of DPH using single Instruction.

→ XRL 83H, #0FFH

⟹ CLR A ; clear Accumulator

⟹ ~~CPL~~ CPL A ; complement Accumulator

⟹ SWAP A ; Swap the lower & Upper nibble of reg. A (internally).

MOV A, #69H

SWAP A ⟹ 96H.

Ques Consider that a packed BCD number is recived on port 1 unpacked it and send the seprated BCD numbers to port 2 and port 3 respectively

→ Packed BCD number

0 ⇒ 0000 like → 58;59H

↓ 05H → P2

9 ⇒ 1001 08H → P3.

MOV P1, #85H

MOV A, P1 ; Assume P1 = 85H

MOV B, A

ANL A, #0FH ; Masking upper nibble → A = 05H

MOV P2, A

MOV A, B
ANL A, #0F0H ; Masking lower nibble → A ⇒ 80H
Swap A ; → A → 08H
MOV P3, A
END.

* Bit Level Logical Instruction :-

		2FH	7FH	7EH	7DH	7CH	7BH	7AH	79H	78H
		2EH	77H	76H	75H	74H	73H	72H	71H	70H
I	B	2DH	6FH							68H
N	I	2CH	67H							60H
T	T	2BH	5FH							58H
E		2AH	57H							50H
R	A	29H	4FH							48H
N	D	28H	47H							40H
A	D	27H	3FH							38H
L	R	26H	37H							30H
	E	25H	2FH							28H
R	S	24H	27H							20H
A	S	23H	1FH							18H
M	A	22H	17H							10H
	B	21H	0FH	0EH	0DH	0CH	0BH	0AH	09H	08H
	L	20H	07H	06H	05H	04H	03H	02H	01H	00H

E

	UNUSED	0FFH							0F8H
	REGB 0F0H	0F7H	0F6H	0F5H	0F4H	0F3H	0F2H	0F1H	0F0H
	UNUSED	0EFH							0E8H
	ACC 0E0H	0E7H	0E6H	0E5H	0E4H	0E3H	0E2H	0E1H	0E0H
	UNUSED	0DFH							0D8H
S	PSW 0D0H	0D7H	0D6H	0D5H	0D4H	0D3H	0D2H	0D1H	0D0H
F	UNUSED	0CFH							0C8H
R	UNUSED	0C7H							0C0H
	IP 0B8H	0BFH	0BEH	0BDH	0BCH	0BBH	0BAH	0B9H	0B8H
	PORT3 0B0H	0B7H	0B6H	0B5H	0B4H	0B3H	0B2H	0B1H	0B0H
	IE 0A8H	0AFH	0AEH	0ADH	0ACH	0ABH	0AAH	0A9H	0A8H
	PORT2 0A0H	0A7H	0A6H	0A5H	0A4H	0A3H	0A2H	0A1H	0A0H
	SCON 98H	9FH	9EH	9DH	9CH	9BH	9AH	99H	98H
	PORT1 90H	97H	96H	95H	94H	93H	92H	91H	90H
	TCON 88H	8FH	8EH	8DH	8CH	8BH	8AH	89H	88H
	PORT0 80H	87H	86H	85H	84H	83H	82H	81H	80H

→ Bit Level Logical Instruction :-

OPCODE	OPERANDS	COMMENTS
MOV	C, b	; copy the addressed bit (b) into carry bit (C)
MOV	b, C	; copy the carry bit (C) into addressed bit (B)
SETB	C	; Set carry bit
SETB	b	; Set addressed bit.

Bit Level
MOV C, 81H ;

Byte Level
MOV A, 81H

C → Bit address level.
↓
Carry.

Ques:- Recive a bit from pin no 2 of port 1 and send it to pin no. 0 of Port 2.

→ MOV C, 92H
MOV 0A0H, C

Ques:- Set 5th bit of register 22H.

→ SETB 22H.5 ⟶ 15H

OPCODE OPERANDS CPL 02H

CLR	C
CLR	b
CPL	C
CPL	b

Ques:- Select reg. bank 2 using bit Level operation consider that the PSW is in default ~~reg.~~ reset status.

SETB D4H
CLR D3H

59H

: 0101 1001

MOV E, B

MVI A, 00H 2 Byte

XRA A 1 Byte — S, Y, AC, X

→ SUB A 1 Byte

ANI 00H 2 Byte

2 KB 2×2^{10} (11) → 0 → 10

11 Address line

8 chip

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

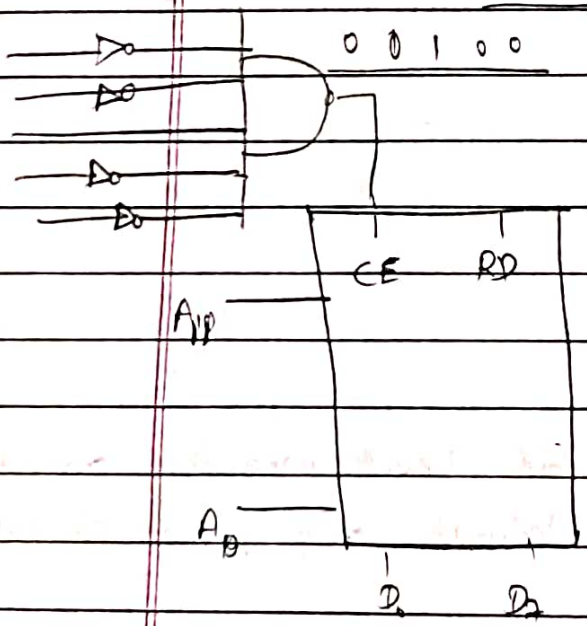
0 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1

2^{12} 4

2^{12} (8)

$2^2 \times 10^3$

4 KB



A_{15} A_{14} A_{13} A_{12}

0 0 0 0

A_{11}

Decoder output.

A15 Combination

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

Bit Level Instruction :-

OPCODE	OPERANDS	COMMENTS
ANL	C, b ;	Logical AND the addressed bit with carry bit & store result in carry bit
ANL	C, /b ;	Logically AND the complement of addressed with the carry bit & store result in carry bit the addressed bit remains unchanged.
ORL	C, b ;	
ORL	C, /b ;	

→ MOV 20H ; #0F4H
SETB C
ANL C, 05H
↓
Bit address.
ANL C, /05H

MOV A, #45H ; Load 45H into Accumulator A.
MOV 20H, A ; Store A into internal RAM address 20H.
MOV 0D0H, #0FFH ; Load 0FFH into PSW (all status bits = 1)
MOV 07H, C ; Store carry flag value into bit Address 07H
MOV P1, 20H ; output RAM [20H] (45H) to port 1.
MOV 08H, PSW ; Copy PSW into RAM address 08H.
MOV 0A0H, 0E0H ; Copy Accumulator (A) into Port 2 (P2 = A)
ORL C, 04H ; OR bit 04H with carry flag and store in C
MOV 92H, C ; Move carry flag to bit 2 of Port 1
MOV 81H, P1. ; Move Port 1 value into stack pointer (SP)

* Rotate Instructions :-

	OPCODE	OPERANDS	COMMENTS
8-bit Rotation	RL	A ;	Rotate Left Accumulator contents by 1-bit (i.e, bit $D_0 \rightarrow D_1$, $D_1 \rightarrow D_2$, ... $D_7 \rightarrow D_0$)
9-bit Rotation	RLC	A ;	Rotate Left Accumulator through (with) carry bit (i.e., $D_0 \rightarrow D_1$, $D_1 \rightarrow D_2$, ... $D_7 \rightarrow$ CY & CY $\rightarrow D_0$)

Rotate Instructions works only Accumulator.

MOV A, #04H → $D_3 D_2 D_0$ = 0000 0100 → 04H

RL A → 0000 1000 → 08H (×2)

RL A → 0001 0000 → 10H (×2)

RLC A → CY = 1, 0001 0000 → 11H

RLC A → CY = 0, 0001 0000 → 10H.

→ RR A ; Rotate Right Accumulator contents by 1 bit (i.e., bit ($D_0 \to D_7$, $D_1 \to D_6$, $D_2 \to D_5$, ... $D_7 \to D_6$))

$D_7 \to D_6 \to D_5 \to D_4 \to D_3 \to D_2 \to D_1 \to D_0 \to$

RRC A

Ques: Swap the lower and upper nibble of register A using all posible methods.

→ MOV A, #75H → 0111 0101

RL A → 1110 1010

RL A → 1101 0101

RL A → 1010 1011

RL A → 0101 0111 → 57H

→ MOV A, #75H 0111 0101
RR A 1011 1010
RR A 0101 1101
RR A 1010 1110
RR A 0101 0111 → 57H.

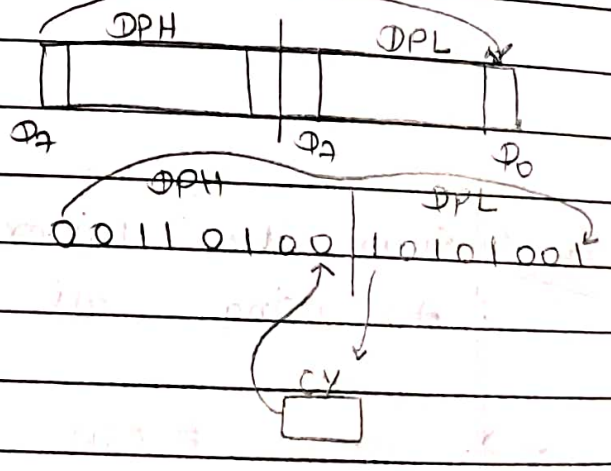
→ SWAP A

Ques:- Send the MSB of R_0 to R_4 to pin 0 of Port 1 one by one.

→

Ques:- Rotate DPTR Left by 1-bit

→
```
MOV DPTR, #349H
MOV A, DPL
RLC A
MOV 82H, A
MOV A, 83H
RLC A
MOV 90H, 82H
MOV 00H, C
MOV 82H, 90H
```



Ques:- Rotate DPTR Right by 1-bit.

→ MOV DPTR, #1234H 00110100 (DPH) 10101001 (DPL)
MOV A, DPL
RRC A

Answer → MOV A, R0
RLC A
MOV P1.0, C

MOV A, R1
RLC A
MOV P1.0, C

MOV A, R2
RLC A
MOV P1.0, C

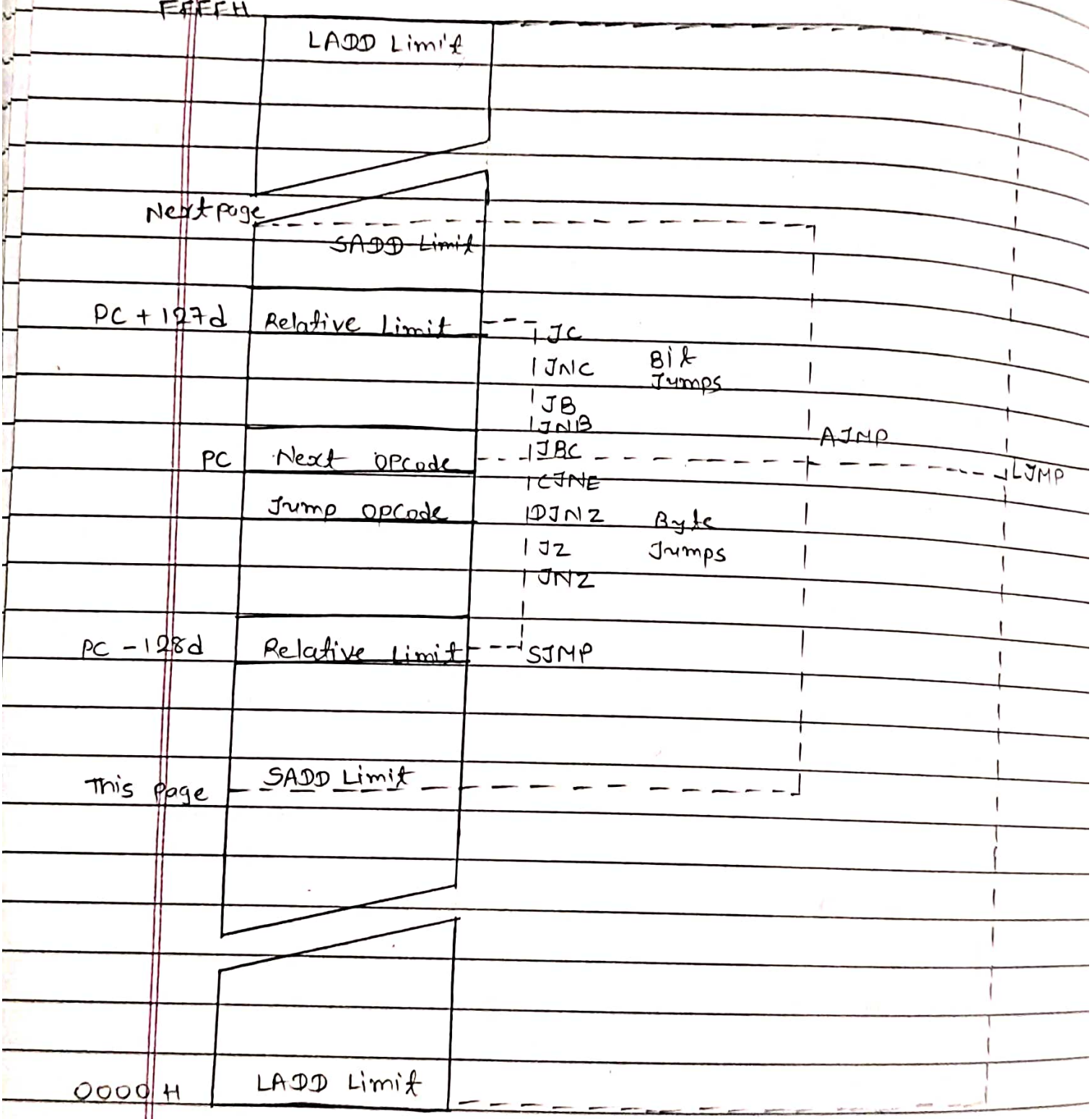
MOV A, R3
RLC A
MOV P1.0, C

MOV A, R4
RLC A
MOV P1.0, C.

END.

Ch:- 8 Jump and Call Instruction.

* Jump Instruction Ranges :-



* Bit Jumps (conditional)

Mnemonic	operation
JC radd	Jump relative if the carry flags is sel (CY=1)
JNC radd	Jump relative if the carry flag is reset (CY=0)
JB b, radd	Jump relative if the addressed bit is set (b=1)
JNB b, radd	Jump relative if the addressed bit is reset (b=0)
JBC b, radd	Jump relative if the addressed bit is set (b=1), and clear the addressed bit along with the Jump.

relative address

PC-128
PC+127

→ No flags are affected unless the bit is Flag bit in PSW in instruction JBC.

* Unconditional Jumps :-

Mnemonic	operation
JMP @A+DPTR	- Jump to the address formed by adding A to the DPTR - Unconditional Jump - Address can be anywhere in program Memory (code memory) - A, DPTR and Flags are not affected

AJMP sadd — Jump to absolute short range address sadd
- Unconditional Jump
- No flags affected

LJMP ladd — Jump to absolute long range address ladd
- Unconditional Jump
- No flags affected

-128 to +127 → SJMP radd — Jump to relative address radd
- Unconditional Jump
- No flags affected.

PC -128 to PC +127 → relative range

```
START: MOV A, P1
       MOV P2, A
       SJMP START
```

Ques: Receive a byte from port 1 and send it serially out on pin 2 of port 2.

→
```
MOV B, #08H
MOV P1, #ABH
MOV A, P1
START: RRC A
       MOV 0A2H, C
       DJNZ 0F0H, START
END
```

Parallal - 8-bits P1
- serially
- P2.2

* Byte Jumps (conditional)

Mnemonic	operation
DJNZ Rx,radd Decrement Jump on Non Zero	Decrement the register Rx by 1 and Jump to the relative address if the result is not zero; No flags are affected.
DJNZ add,radd	Decrement the direct address by 1 and Jump to the relative address if the result is not zero; No flags are affected unless the direct address is PSW.
JZ radd	Jump to the relative address if A is 0; No flags are affected as well reg. A.
JNZ radd	Jump to the relative address if A is not 0; No flags are affected as well reg. A.

CAUTION :- There is no zero flag in 8051, JZ and JNZ checks the contents of Accumulator for zero.

4 Codes 7 Segments

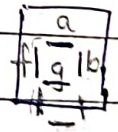
Logic of

Ques:- Receive a byte from Port 2 and count number of zeros in the byte and send this count on port 1.

→
```
MOV R2, #08H
MOV A, P2
MOV R3, #00H
START: RRC A
       JC SKIP
       INC R3
SKIP: DJNZ R2, START
MOV P1, P3
END
```

```
START: MOV A, 0A0H
       MOV R2, #08H
       MOV R3, #00H
NEXTBIT: RRC A
       JNC ZERO
       SJMP CONT
ZERO: INC R3
CONT: DJNZ R2, NEXTBIT
       MOV 090H, R3
       SJMP START
       END.
```

Ques:- Decide the parity in a byte received from port if the parity is odd send the hex code for small o on port 1 if the parity is even send the code for H on port 1.



1100011

→
```
MOV A, P2
JNB
```

Byte Jumps (conditional) :-

Mnemonic	operations
CJNE A, add, radd Compare Jump if Not Equal	Compare the contents of the A register with the contents of the direct address; if they are not equal, then jump to the relative address CY = 1 if A < contents of addressed reg. CY = 0 if A >= contents of addressed reg
CJNE A, #n, radd	Compare the contents of the A register with the immediate number n; if they are not equal, then jump to the relative address. CY = 1 if A < 8-bit immediate number CY = 0 if A >= 8-bit immediate number

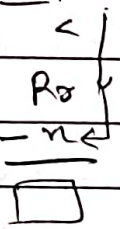
Ques:- Assume that their are 100 byte received from port 1 count how many times the highest value is recive. Send that count on port 2.

→ 1.) → set counter for 100 Bytes to be received from P1.

2) check the received Byte whether it's FFH or not
3) if is FFH, increment counter
4) complete the loop
5) send the count on port 2

* Byte Jumps (conditional)

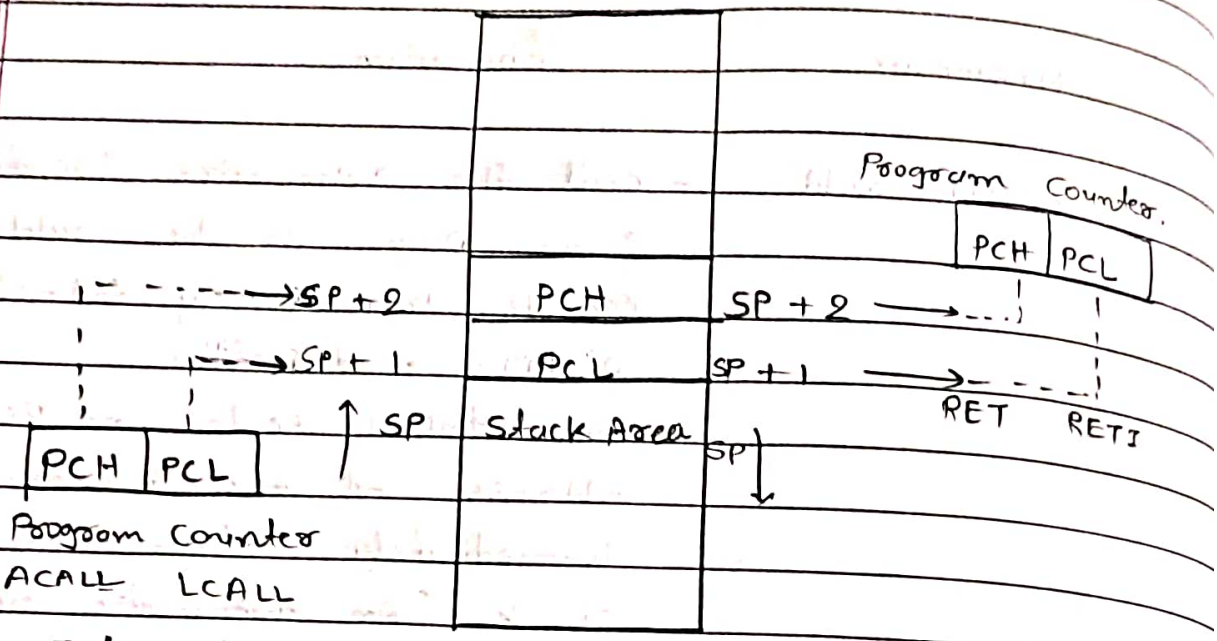
Mnemonic	operation
CJNE Rx, #n, radd	Compare the contents of the Rx Register with the immediate number ; if they are not equal, then jump to the relative address. CY = 1 if Rx < contents of address reg CY = 0 if Rx >= contents of address reg
CJNE @Rp, #n, radd	compare the contents of the address contained in Rp with the immediate number n; if they are not equal then jump to the relative address. CY = 1 if @Rp < 8-bit imm. number CY = 0 if @Rp >= 8-bit imm. number



* Call and Subroutines :-

Mnemonic	operation
ACALL Sadd	- Call the subroutine located on the same page as the address of the opcode immediately following the ACALL instruction - Push the contents of PC i.e address of the instruction immediately after the CALL, on stack (An automatic operation).
LCALL Ladd	- call the subroutine located anywhere in the program memory space - Push the contents of PC i.e address of the instruction immediately after the call, on stack (An automatic operation)
RET	- POP2 bytes from the top of the stack into program counter.

→ Storing and Retrieving the Return Address



* Machine cycle calculation. (Generating Square wave)

1) Generate a square wave and observe on R_0 :-

```
ORG 0000H ;                    MACHINE CYCLES.
START: CPL 097H ;                    2
       ACALL DELAY;                  2
       SJMP START ;                  2
ORG 0300H
DELAY: MOV R3, #COUNT1 ;             1 x 1
BACK : MOV R4, #COUNT2 ;             1 x count1
HERE ; DJNZ R4, HERE ;               2 x count1 x count2
       DJNZ R3, BACK ;               2 x count1
       RET           ;               2 x 1
       END
```

duty cycle = $\frac{T_{ON}}{T_{ON} + T_{OFF}} \times 100\%$

Square = $\frac{0.5T}{T} \times 100 = 50\%$

Tinst = $\frac{C \times 12}{\text{Crystal frequency}}$ decimal

↓

11.0592 MHz

= 12 MHz

Total machine cycles of delay subroutine is :-

$C = (2 \times count2 \times count1) + (3 \times count1) + 3$

Ex:- Generate a square wave of 2 kHz frequency on pin no. 7 of Port 1 in 8051 use register as to generate the delay.

2 kHz P1.7 → 97H first Require delay find.

$f = 2kHz \Rightarrow T = \frac{1}{f} \Rightarrow \frac{1}{2 \times 10^3} = 0.5 \times 10^{-3} = 0.5\,ms$

$= 500\,\mu s.$

No of Machine cycles → C

$TD = \frac{1}{2}T = \frac{1}{2} \times 500 = 250\,\mu s$

$C = \frac{TD \times \text{Crystal frequency}}{12}$

$= \frac{250 \times 10^{-6} \times 12 \times 10^6}{12}$

$= 250$ Machine cycles.

$C = 2 \times C_1 \times C_2 + 3C_1 + 3$

Assuming $C_1 = 10$;

$\therefore 250 = 2 \times 10 \times C_2 + 30 + 3$

$\therefore 250 - 30 - 3 = 20C_2$

$\therefore \frac{217}{20} = C_2$

$\therefore C_2 = 10.85 \approx 11$

4KHz Pl. 7 → 92H

$f = 4KHz \Rightarrow T = \frac{1}{f} = \frac{1}{4\times10^3} = 0.250 \times 10^{-3} = 250\,\mu s$

$TD = \frac{1}{2}T = \frac{1}{2} \times 250 = 125\,\mu s$

$C = \frac{125 \times 10^{-6} \times 12 \times 10^6}{12}$

$= 125$

Machine cycle calculation. (Generating square wave)

1.) Generate a square wave and observe on R_0:

```
ORG 0000H ;                    MACHINE CYCLES.
START: CPL 097H ;                    2
       ACALL DELAY;                  2
       SJMP START ;                  2
ORG 0300H
DELAY: MOV R3, # COUNT 1 ;           1 x 1
BACK : MOV R4, # COUNT2 ;            1 x count1
HERE ; DJNZ R4 , HERE ;              2 x Count1 x count2
       DJNZ R3, BACK ;               2 x Count1
       RET           ;               2 x 1
       END
```

$T = 0.002\ S$

$2 \times 10^{-3}\ S$

$2\ ms$

$2000 \times 10^{-3}\ ms \quad 2000 \times 10^{-6}\ S$

4KHz P1.7 → 9TH 2000 us = 2x ms

= 0.002 S.

$f = 4KHz \Rightarrow T = \frac{1}{f} = \frac{1}{4 \times 10^3} = 0.250 \times 10^{-3} = 250\ \mu s.$

$TD = \frac{1}{2} T = \frac{1}{2} \times 250 = 125\ \mu s$

$C = \frac{125 \times 10^{-6} \times 12 \times 10^6}{12}$

$= 125$

* Generate a Rectangular shape wave of 500Hz freq with 40% duty cycles. Consider the pin 0 of port 2.

→ Duty cycles = 40%.

T_{ON} = 40% of T

T_{OFF} = 60% of T

$T = \frac{1}{f} = \frac{1}{500} = 0.002\ s$

$= 2 \times 10^{-3}\ ms$

$TD = \frac{1}{2} T = \frac{1}{2} \times 2000 = 1000\ s$

$C = \frac{1000 \times 12 \times 10^6}{12}$

$C = 10^9$

$C =$

```
START:
    SETB
    ACALL
    ACALL
    CLR
    ACALL
    ACALL
    ACALL
    SJMP START
```

```
ORG 1000H
START: SETB 0A0H ; Pin 0 of Port 2
       ACALL DELAY
       ACALL DELAY
       CLR 0A0H ;
       ACALL DELAY
       ACALL DELAY
       ACALL DELAY
       SJMP START
ORG 500H
DELAY: MOV R5, #0AH
LOOP : MOV R6, #12H
HERE : DJNZ R6, HERE
       DJNZ R5, LOOP
       RET
       END
```

Ex:- Calculate the maximum delay offered by a delay subroutine having 2 registers as counter and 3 registers as counter seperately calculate this for 12 MHz freq. and 11.0592 MHz freq.

Load counter value 0A0H

ORG 0000H: 0A0H

→

$f = 500\,Hz \rightarrow$ Rectangular wave of 40% duty cycle.

$$T = \frac{1}{f} = \frac{1}{500} = 0.002\ s$$
$$= 2\ ms$$
$$= 2000\ \mu s$$

40%	60%
T_{ON}	T_{OFF}

$\leftarrow T = T_{ON} + T_{OFF} \rightarrow$

$T_{ON} = 0.4 \times T = 0.4 \times 2000 = 800\ \mu s$

$T_{OFF} = 0.6 \times T = 0.6 \times 2000 = 1200\ \mu s$

A common delay to be designed of 20% of T hence

$$T_D = 20\%\ of\ T$$
$$= 0.2 \times T$$
$$= 0.2 \times 2000$$
$$= 400\ \mu s$$

Crystal freq = 12 MHz

$$T_D = \frac{C \times 12}{\text{crystal freq.}}$$

$$\therefore C = \frac{T_D \times f_{req}}{12}$$

$$= \frac{400 \times 10^{-6} \times 12 \times 10^{6}}{12}$$

$$= 400$$

$$C = 3 + 3C_1 + 2C_1C_2$$

$$\therefore 400 = 3 + 3C_1 + 2C_1C_2$$

Assume $C_1 = 10$;

$$\therefore 400 = 3 + 30 + 20C_2$$

$$\therefore C_2 = \frac{400 - 33}{20}$$

$$\therefore C_2 = 18.35 \approx 18;$$

duty cycle = $\frac{T_{ON}}{T_{ON} + T_{OFF}} \times 100\%$

Square = $\frac{0.5T}{T} \times 100 = 50\%$

$T_{insk} = \frac{C \times 12d}{\text{Crystal frequency}}$ ← decimal

↓

11.0592 MHz

= 12 MHz

Total machine cycles of delay subroutine is :-

$C = (2 \times count\,2 \times count\,1) + (3 \times count\,1) + 3$

Ex:- Generate a square wave of 2 kHz frequency on pin no. 7 of Port 1 in 8051 use register as to generate the delay.

2 kHz P1.7 → 97H first Require delay find

$f = 2kHz \Rightarrow T = \frac{1}{f} \Rightarrow \frac{1}{2 \times 10^3} = 0.5 \times 10^{-3} = 0.5\,ms$

$= 500\,\mu s.$

No of Machine cycles → C

$TD = \frac{1}{2}T = \frac{1}{2} \times 500 = 250\,\mu s$

$C = \frac{TD \times \text{Crystal frequency}}{12}$

$= \frac{250 \times 10^{-6} \times 12 \times 10^6}{12}$

$= 250$ Machine cycles.

$C = 2 \times C_1 \times C_2 + 3C_1 + 3$

Assuming $C_1 = 10$;

$\therefore 250 = 2 \times 10 \times C_2 + 30 + 3$

$\therefore 250 - 30 - 3 = 20C_2$

$\therefore \frac{217}{20} = C_2$

$\therefore C_2 = 10.85 \approx 11$