24/8/22

Micro processor

1) 8085 & 8086 } Microprocessors → Assembly level
   - Has its own programming language

8085 → primitive processor
- easy to learn
- No one use this now-a-days but easy to learn for startup microprocessor.

2) 80851 - Micro controller — embedded c.

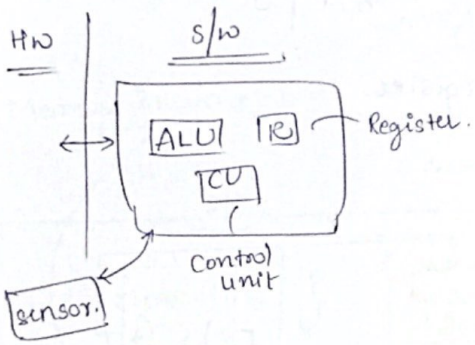
3) ARM controller (Ardino)

- Hardware part of microprocessor is called Architecture of the processor.
- Software is life, it is the programming part

Machine level language
↓
Assembly level language
↓
High level language

How this processor interacts with outer part of world / surrounging or external part of world?

- Sensor



either fetch or store the data

25/08/22

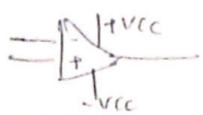
Micro processor

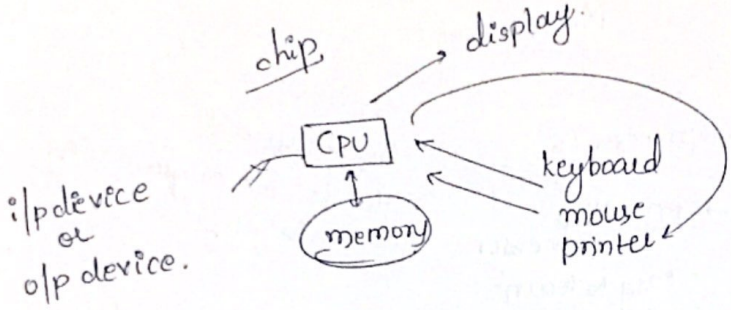
Small (10^{-6}) (NANO fabrication)

Something that manipulates data

optimal in both speed and time

→ OPAMP (operational Amplifier)





- Digital logic devices are of two types.

Combinational Sequential.

- Computer is called "Digital State Machine".

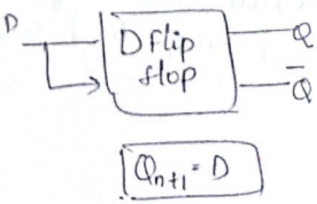
- ALU → one of the basic logic unit

Other components - registers

A bunch of memory locations.

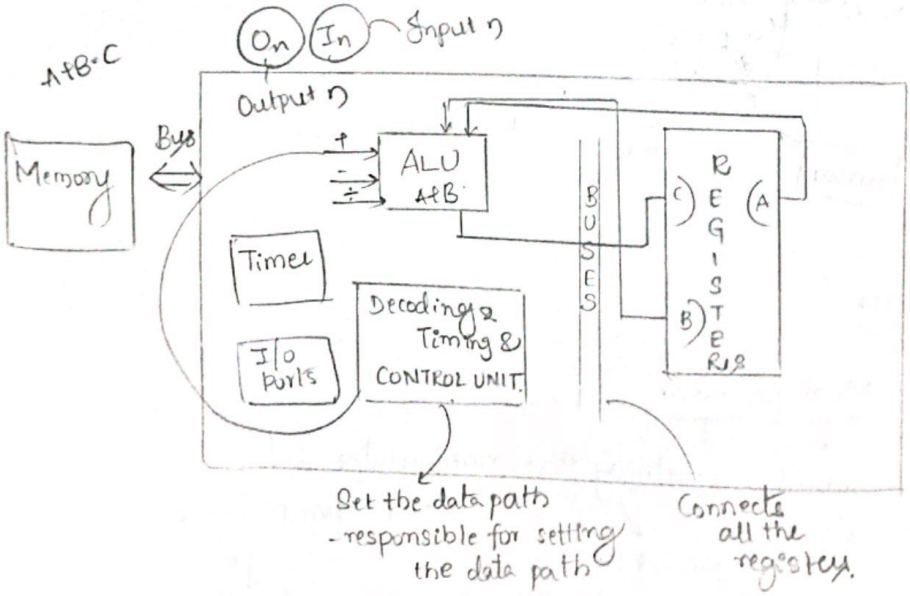
- 1 bit of memory is D F/F.

or register ✓



D	Q_n	Q_{n+1}
1	0	1
1	1	1
0	0	0
0	1	0

- Combination of F/F is a register.



Three types of Buses

- A Bus (Address Bus)
- D Bus (Data Bus)
- C Bus (Control Bus)

- Control unit is generating signals such that data bus is generated.
- All the components of CPU, if fabricated on a single chip, it is called Microprocessor
- CPU on a single chip is called "Microprocessor."

- Registers are not required to store the program
- We have to connect external memory to the register, to store the data.

External Memory interacts with processor using external BUS

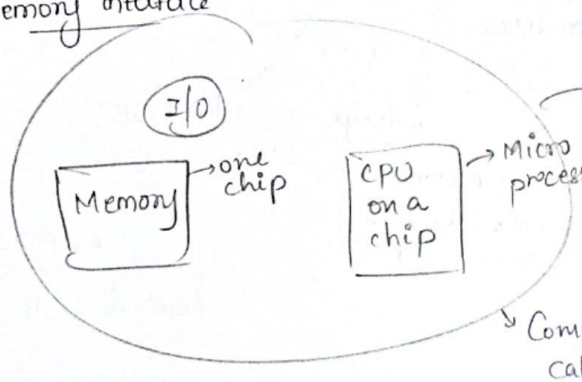
Memory: huge volume of registers and cache.

Registers have an address.

I/O device also interacts with processor via bus. Considered as extra memory.

both address & data will be present in the bus & as well as control unit part like whether to read/write.

Memory interface



Computer on a chip is called "Micro Controller"

26/8/22

8085 Microprocessor

↓

8-bit Microprocessor

Optimized to perform 8-bit operations.

↙ ↘ ↙ ↘

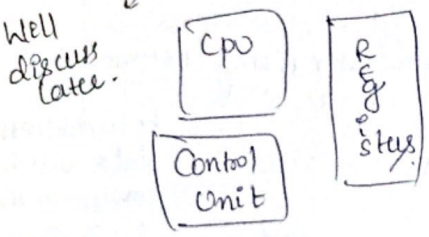
Time resources. minimum for 8-bit operations

Arithmetic logic.

R

Hardware Model

Well discuss later.



Programming Model

We can access.

(Special purpose register)								
ACCUMULATOR (8-bit)		S	P	-	AC	-	Z	- CY
B (8)		C (8)						
D (8)		E (8)						
H (8)		L (8)						
PCH	PROGRAM COUNTER (PC) (16-bit)							PCL
	STACK POINTER (16-bit)							
W (8)		Z (8)						

like MSB

least LSB.

Registers

- General purpose register
- Special purpose register
- Registers that are not accessible to user.

B C D E H L

pair up.

BC DE HL (we can use combined versions like this to store 16-bit)

MSB LSB.

(Most significant bit)

least significant bit.

→ S → sign flag

P → parity flag (even no. of ones ⇒ set, odd no. of ones ⇒ reset)

AC → Auxiliary carry flag. (set while BCD addition)

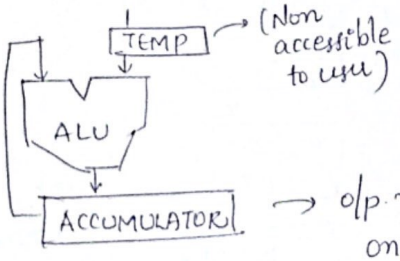
Z → zero flag (set when arithmetic addition results zero)
CY → carry flag

→ PROGRAM COUNTER → special
- (points to the next ins)
stores the address of the next instruction to be executed
- We can't access PCH & PCL independently. We have to only store 16-bit.

special purpose.

- STACK POINTER - always points to the top of the stack.
- W, Z special role to perform.
  - temporary registers
  - Not accessible to user.

- ACCUMULATOR - o/p register of ALU.



- There are 2^n representations for n bit.
- for $n=16$, $2^{16} = 65,536 = 64k$
  $= 64 \times 1024$ → memory locations

2^{16} no. of memory locations.

EFH → Hexa decimal representation
255

1 nibble = 4 bits.

Most significant Nibble ← MSN LSN

$2^7\ 2^6\ 2^5\ 2^4\ \ 2^3\ 2^2\ 2^1\ 2^0$

MSb (Most Significant bit) LSb

$15\ 14\ 13\ 12\ 11\ 10\ 9\ 8 \quad 7\ 6\ 5\ 4\ 3\ 2\ 1\ 0$
$2^{15}\ 2^{14}\ 2^{13}\ 2^{12}\ 2^{11}\ 2^{10}\ 2^9\ 2^8 \quad 2^7\ 2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$ → 16-bit representation

MSB Most significant byte LSB.

- Accumulator & flag together called program status word.

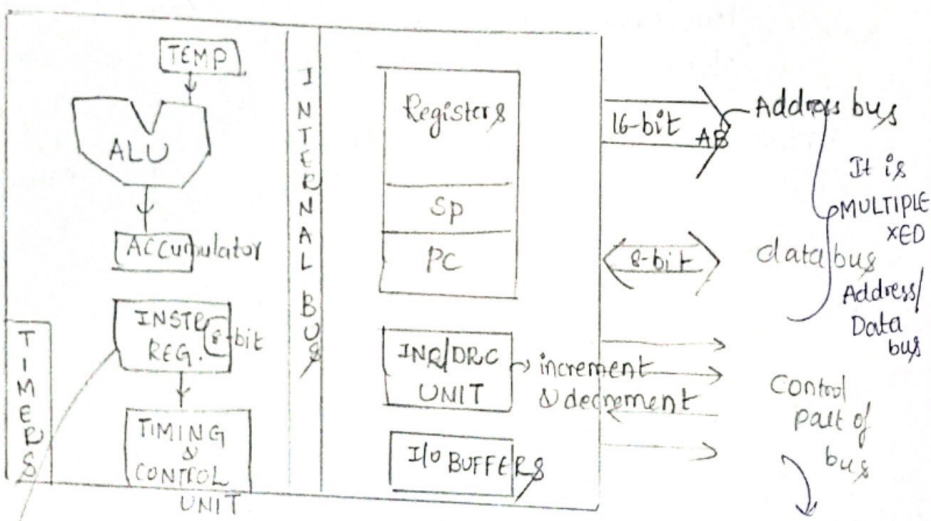
Nxt class we will discuss about programming Model of 8085.

7/9/22

8085 (μP) → Microprocessor

8-bit processor

Hardware Model



16-bit Address Bus → This processor has a 16-bit address bus (outward)

8-bit → and also has a 8-bit data bus (both sided)

Another tmp register ⇒ Instruction Register (forgot in last class) - it is a special purpose register.

Role: Every time when processor fetches the instruction, it gets copied to instruction register.

MOV B,A ⇒ move content of A to B

- There are no dedicated pins to establish, 16-bit in 8085.

Instruction has 2 parts :-

Pneumonic & data

Code part

- It varies 2bit, 4-bit, 3-bit

Instruction set of 8085 :- (Assembly language)

Types of instructions :-

register → inside the processor

Memory → outside the processor.

① Data Transfer instructions

(i) Register to Register ⇄ in both ways

(ii) Register to Memory (writing the data)

(iii) Memory to Register (Reading the data)

(in terms of time the data bus is multiplexed)

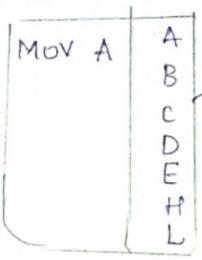
X Memory to memory.

(i) Register to register :-

MOV B,A B ← A

↓

destination



All these possibilities for each register

(ii) Register to Memory

MVI A, #8BH

more i ↓ Move immediate to A

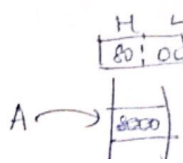
(iii) Memory to Register

LXIH

LXI H, #8000H

load indexed immediate.

immediate next two memory locations is copied to HL pair



MOV M, A [M] ← A

8000H → address of memory.

MOV C,M C←[M]

↓

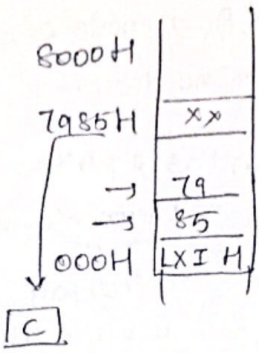
LXI H, #7985H

MOV C,M

↓

Content of memory is copied to C

H	L
79	85



② Arithmetic & logic instruction

- Addition → Add Reg
- SUB
- AND
- OR
- XOR

ADD B A←A+B
content of B added to A and stored in A

ADI, #85H.

↓

Add immediate

A←A+85

ADD M A←A+[M] (HL)

HL pair

* HL register is the only pointer to Memory

LXI H

LXI B #16 bit

LXI D #16 bit

LXI SP #16bit

} We can also store 16bit data in these registers but these are not memory pointers

AND

AND A (A←A AND A)

AND B (A←A AND B)

ANI, #00H (A←A AND 00H)

Q: Write a program to perform the following operation

1) Move the content of B register to L register

MVI B, #00H

MVI L, #05H

MOV L,B

2) Move data 85H to C register

MVI C, #85H

3) Move content of C register to memory location 800H

```
LDXH, #8000H
MOV M,C.
```

4) Move content of memory 9855 H to D register

```
LDX H, #9855H
MOV D,M.
```

HLT → halt the execution ⟶ Machine control instruction

At the end of program this instruction has to write

5) Add B register with memory [M] 8000H and

```
MOV A,B          | LDXH H, #8000H
ADD M            | MOV A,M
                 | ADD B.
```

9/9/22

→ opcode operand

MOV A,B.

what operation has to be to executed

Program:

3 byte - LXI H 8000H
2 byte - MVI A, #65H
1 byte - ADD M
1 byte - MOV M, A
1 byte - HLT

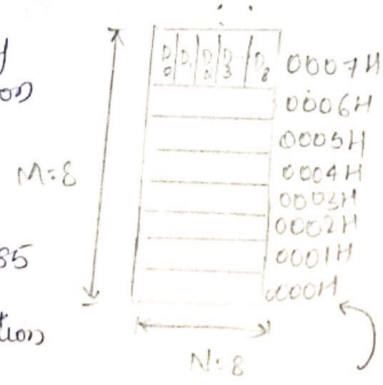
} 5 instructions

size of

(M X N) → width of each memory location

span of Memory

- no. of memory location



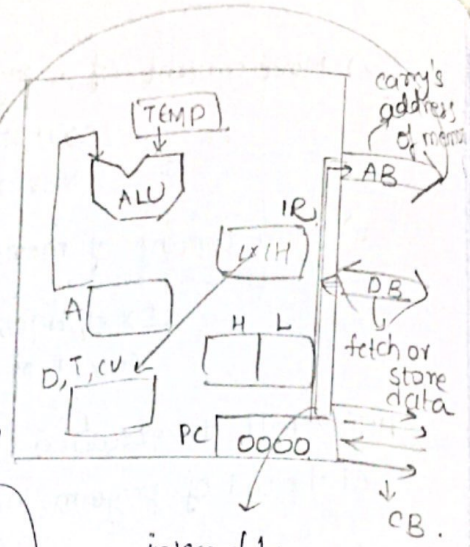
(8 x 8 bit) Memory (R/w memory)

- Every opcode takes - 1 byte in 8085

LXI H, 8000 H. → 3 byte instruction

1 byte — hexadecimal it requires 2 memory location

HLT	0007H
MOV M,A	0006H
ADD M	0005H
65H	0004H
MVI A	0003H
50	0002H
00	0001H
LXIH	0000H



→ address is sent through AB.

First thing is
- Memory read operation or operand fetch.
- Read control signal is sent by the processor
- LXIH will be put on to Data bus. (data will be to put)
- Content of DB is copied to instruction register

if data then stored in registers.

Memory read operation.

internal bus

Content of pc is loaded onto Address bus through internal bus.

memory location identification and it points to that.

* Once the address is put on Address bus pc gets incremented.

- After ADD M, content of HL register is put on the address bus.

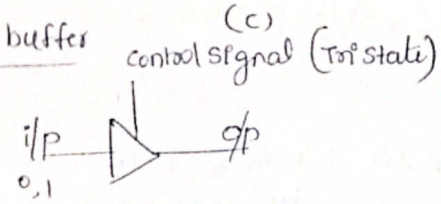
Content of Memory is stored in TEMP.

[TEMP → not accessible to user.]

65
76
DB

- "DB" is stored in Accumulator

Tristate buffer



C	i/p	o/p
1	0	0
1	1	1
0	0	z → high impedance state.
0	1	z

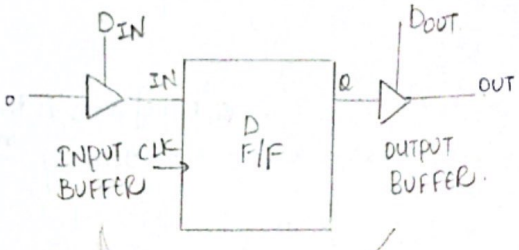
if C = 1 ——— closed switch

if C = 0 ——— open switch

- high impedance state
- data can't move

- Proper maintaince of control unit is required as there is only one internal bus in which all the data will move. It should be in such a way that one after other data should move.

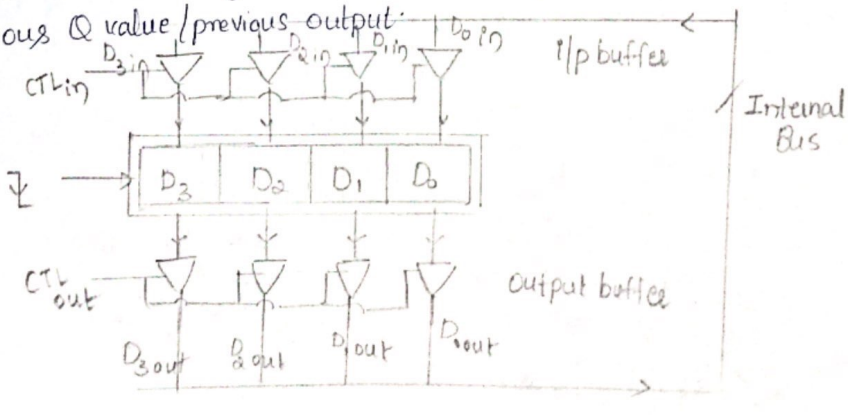
14/9/22



helps to control the dataflow

CLK	I/p	Q
↓	0	0
↓	1	1

if D_{IN} = 0 the I/p is high impedance, the output is same as previous Q value / previous output



How to store the data? / Write operation

1) 4-bit data on internal bus

2) $CNT_{in} = 1$ & $CNT_{out} = 0$ (if $CNT_{out} = 1$ the at a time i/p data and previous o/p data will be on the bus)

3) Apply clock pulse

Read Operation

1) Apply control$_{out}$ = 1 & Control$_{in}$ = 0

↓

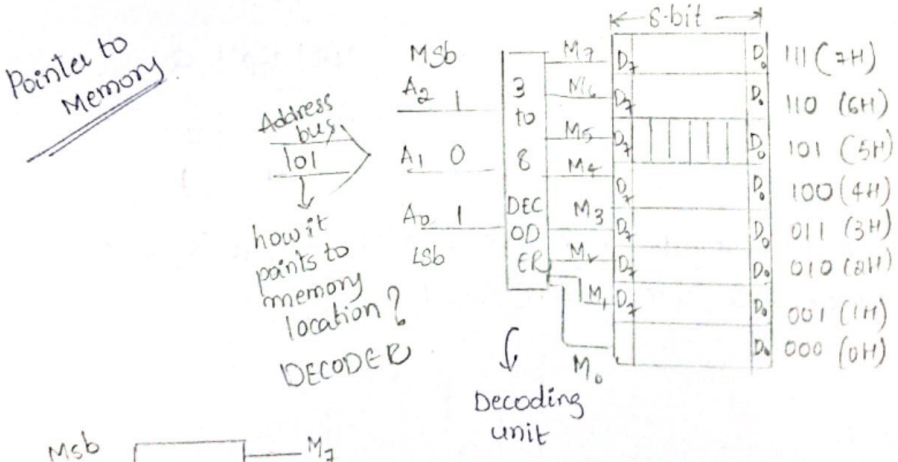
Data will be available in internal bus.

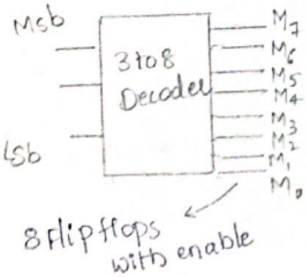
- Every register has a set of input and output buffers.
- CTL_{in} can be taken as WRITE

  CTL_{out} = READ.
- A memory is not only has registers, it has set of input buffers as well as output buffers. which controls the data flow.

Analyze MOV A,B:-

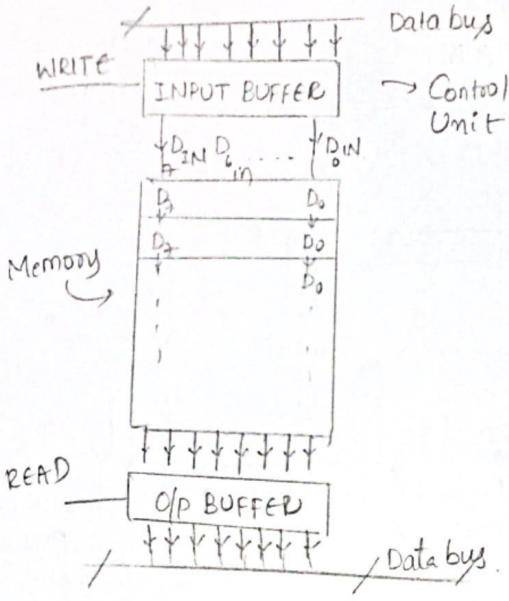
(before going to that let us design a 8x8 bit memory. 64 bits of data is there in memory.





Every memory will have a decoding unit as well as control unit.

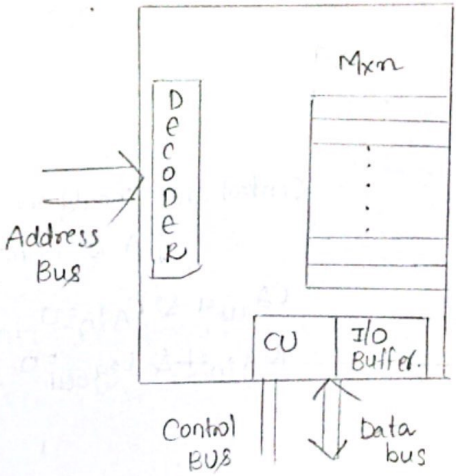
Write on memory location



- Address is available on Address bus.
- Decode the address.

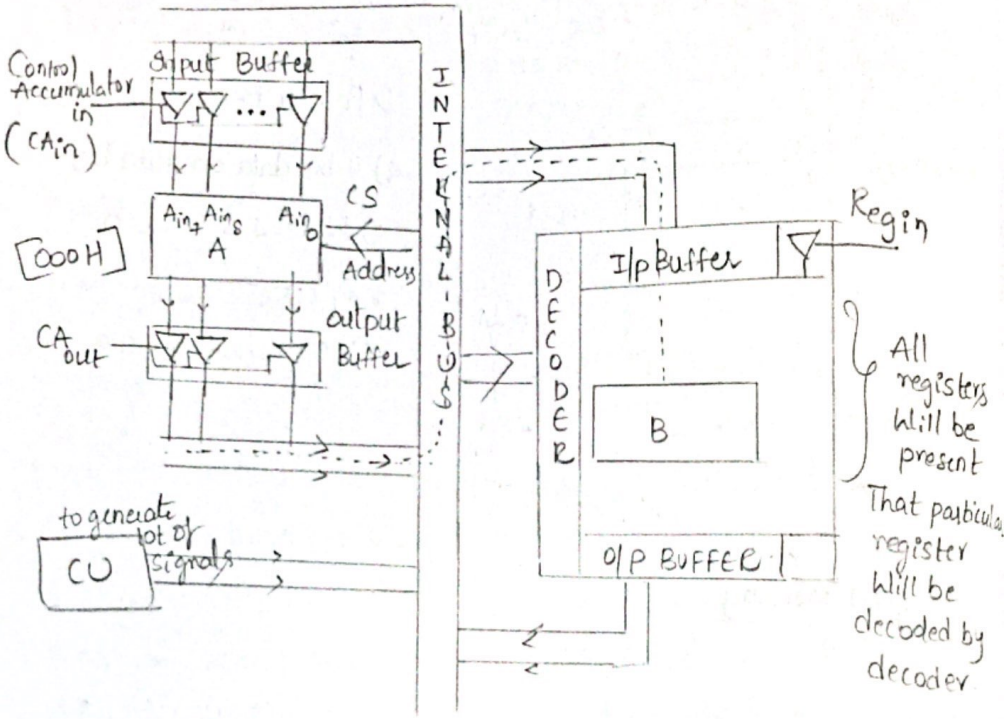
1) Pointer to Memory location
2) 4-bit data on data bus
3) Write = 1 & Read = 0
3 4) CLK
5) STORE/WRITE Data

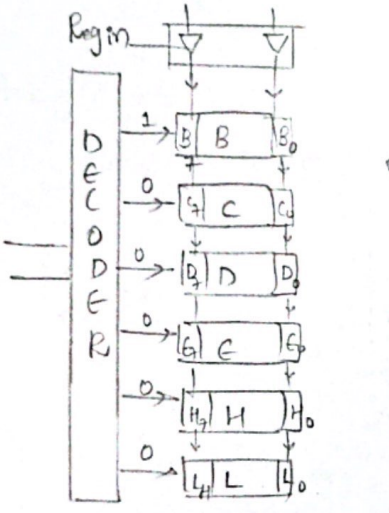
MxN Memory



15/9/22

Execution of instruction MOV B, A





Control unit should give address of A & address of B

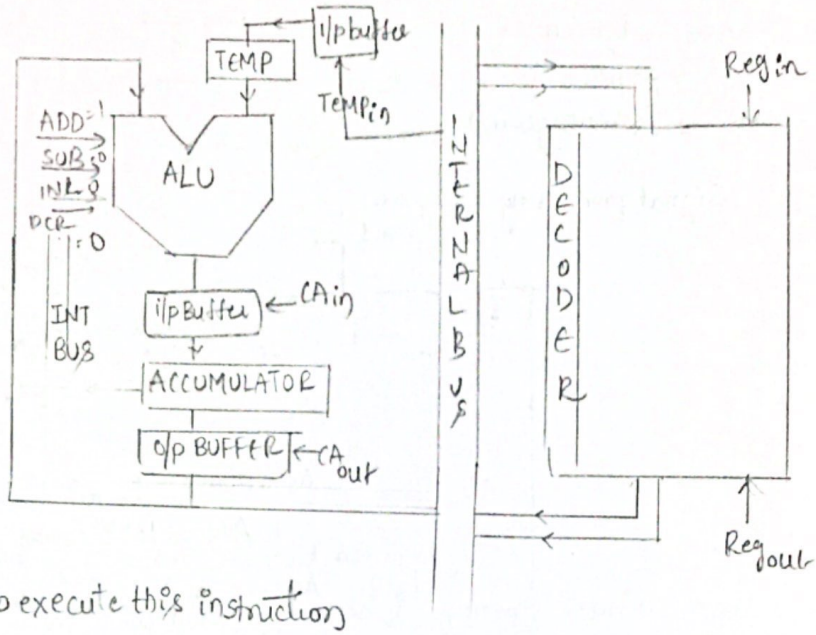
$CA_{out} = 1$ & $CA_{in} = 0$

$Reg_{in} = 1$ & $Reg_{out} = 0$.

Control Unit :- (Typical structure) (need not be 8085's)

SRC ADDR	DEST ADDR	CA_{in}	CA_o	R_{in}	R_{out}	+	-
000	010	0	1	1	0	0	0

ADD C



10 μs → To execute this instruction

3 μs → $CA_{in}=0$, $CA_{out}=1$

9 μs → $CA_{in}=1$, $CA_{out}=0$

} TAKES TIME FOR ADDITION

— One of the way of generating control signals

Not only for 8085

Address of C.

$Reg_{out}=1$, $Reg_{in}=0$

$Temp_{in}=1$, $Temp_{out}=0$.

$CA_{in}=0$, $CA_{out}=1$

Enable add & wait for some time

$CA_{in}=1$, $CA_{out}=0$

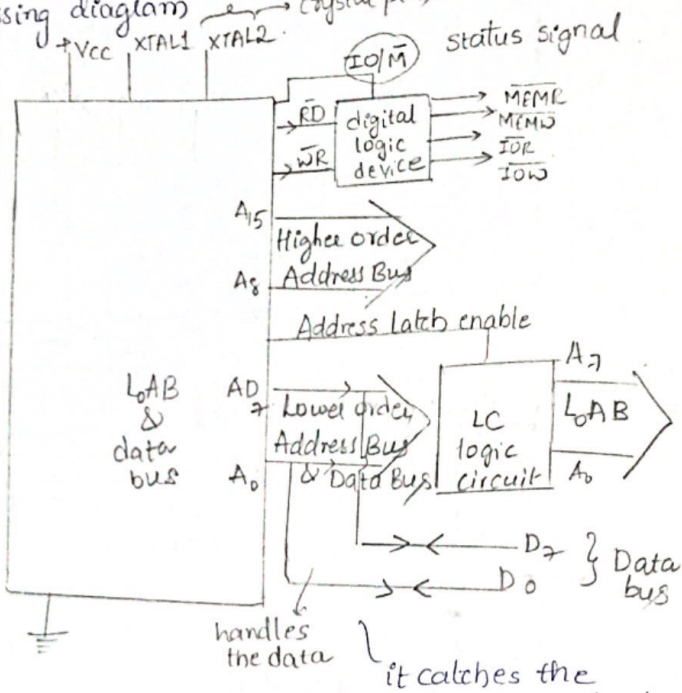
16/9/22

8085 - 8 bit up - There is no dedicated lower order Address bus for 8085 & Data bus

→ 40 pins (pin diagram)

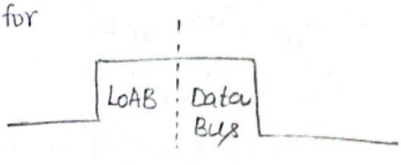
Multiplexed data bus

Signal processing diagram → crystal pins

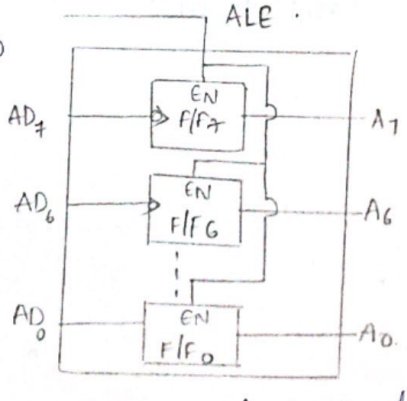


- So it is our duty to demultiplex it.

- This multiplexed bus carry LoAB for only short duration of time.



But as we know memory is twice slower than processor so we have to store the address for longer time on the bus as in this time memory will get processed.



→ Only for certain amount of time it behaves as LoAB

- So For how much time LoAB will be placed on Address Bus ?

↓

This data should provide by processor by a special signal called Address Latch Enable.

↑

Control/status signal

- We are going to store LoA on LC (logic circuit) which consists of registers as long as memory gets processed.
- For how much time LC stores the data depends on ALE signal
- Once As long as ALE is enabled it stores, once it disabled the data will be vanished.

ALE, LC → Catches the data and latches it

Read & write signals

IO device is also an extended memory,

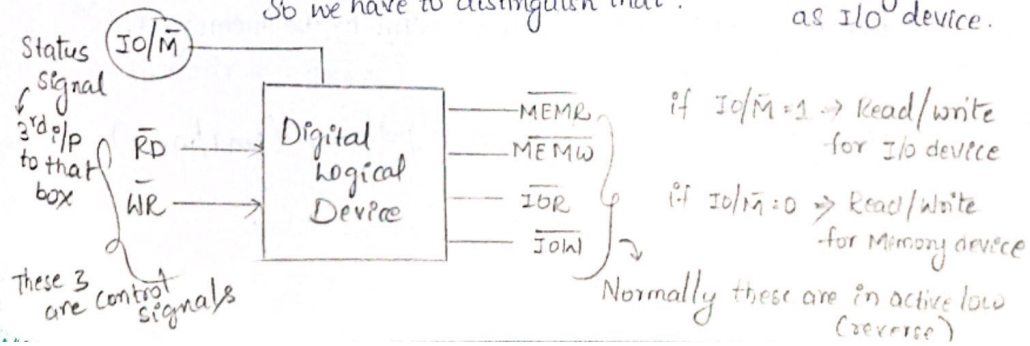
Though, I/O device we have to generate seperate read & write signals for both memory and I/O device.

Memory	I/O
- Used to store huge data for long time	- Assigned for a particular/specific task - Always sense the data

- In case of speed there is a huge diff. I/O is very slow with respect to Memory
- I/O won't need to sense the temperature but processor senses for every micro second. - check!!
- Memory is as fast as processor
- I/O devices are meant to interact with external world They can't sink with processor in speed.

8085 only generates $\overline{RD}$, $\overline{WR}$ signals, but we need Read & write for memory as well as I/O device.

So we have to distinguish that.



if $\overline{MEMW} = 0$ ← Memory write

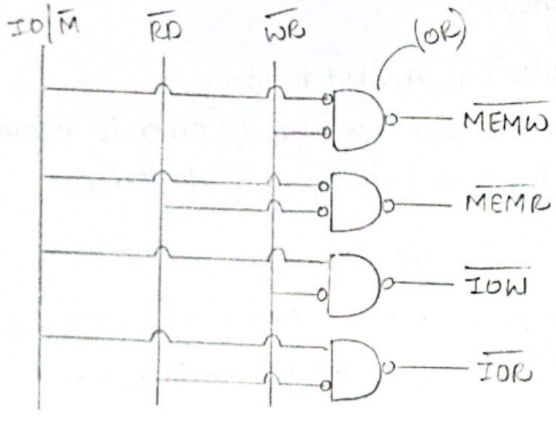
↓

processor intention is to write the data to memory

Truth Table

IO/$\overline{M}$	$\overline{RD}$	$\overline{WR}$	$\overline{MEMR}$	$\overline{MEMW}$	$\overline{IOR}$	$\overline{IOW}$
0	0	1	0	1	1	1
0	1	0	1	0	1	1
1	0	1	1	1	0	1
1	1	0	1	1	1	0

Circuit diagram



17/9/22

- We have seen two logic circuits in the pin diagram till now.
- These need to be designed seperately and integrated to the processor.

Control signals are the processor initiated signals, whereas Externally evaluated signals are called interrupts.

Now, we will learn, How processor will interface with memory.

- First we have to assign the/ fetch m) point to the memory by assigning the address.
- Then, we have to tell the operation to perform. (Read/write).

No.of memory locations = $2^{(N)}$ → no of address lines in your address bus.

$2^3 = 8$

↳ 3 address lines are required to point 8 memory locations (Decoder)

$$N = \frac{\log(\text{Mem size}/\text{locations})}{\log(2)}$$

64kb

$$N = \frac{\log(64 \times 1024)}{\log 2} = 16 \qquad (64 \times 1024 = 2^{10})$$

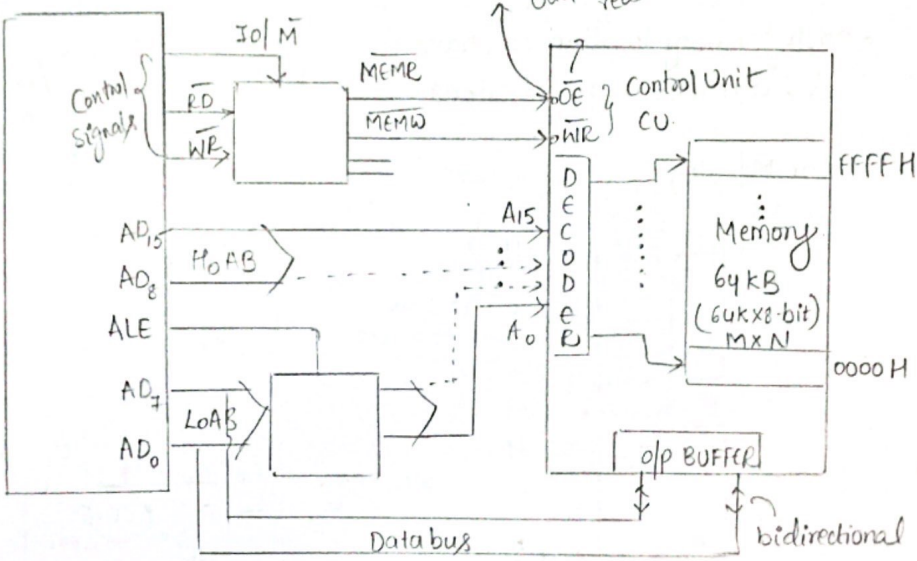
if 8kb, $N = \frac{\log(8 \times 1024)}{\log 2} = 13$

Interface of 8085 with memory :- (Memory Interface With 8085)

64k x8-bit R/W memory

Bubble → indicates Active low → Not an inverter (inside)

Output enable or read bar



MxN

no.of address lines = $\frac{\log(M)}{\log 2}$

$\overline{OE}$, $\overline{WR}$ are control lines of input & output buffers.

32K×8-bit R/W Memory

No. of address lines: $\frac{\log(32K)}{\log 2} = 15$

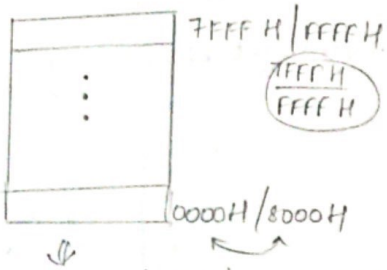
A_{15}	A_{14} A_{13} A_{12}	A_{11} A_{10} A_9 A_8	A_7 A_6 A_5 A_4	A_3 A_2 A_1 A_0	
0/1	0000	0000	0000	0000	8000H/0000H
0/1	111	11 11	1111	1111	FFFFH/7FFFH

Assuming A_{15} as zero,

starting & ending addresses ⇒ 0000 H to 7FFF H.

but simply we can't assume as 0, it is don't care. It can be 1 also.

if $A_{15}=1$ 8000 H to FFFF H

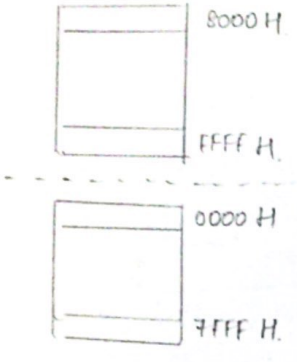


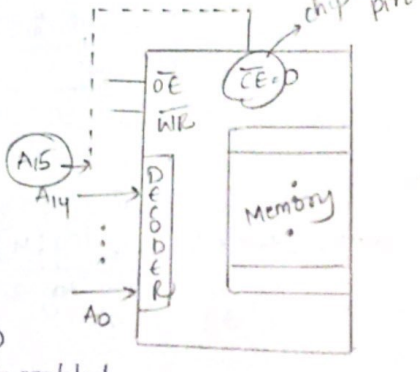
- each memory location can have two different memory locations.

Mirror Memory / -fold back Memory

it can be any one.

Mirror Memory





: leave it hanging

- pathal Decoding Technique.

$\overline{CE}$ → chip enable pin, if low then only this particular memory get enabled.

Mirror Memory/Fold back memory going to exists.

- If one or more address lines are left unconnected or uninterfaced with memory as don't care ⟹ Mirror Memory.
  ↓
  Poor Memory.

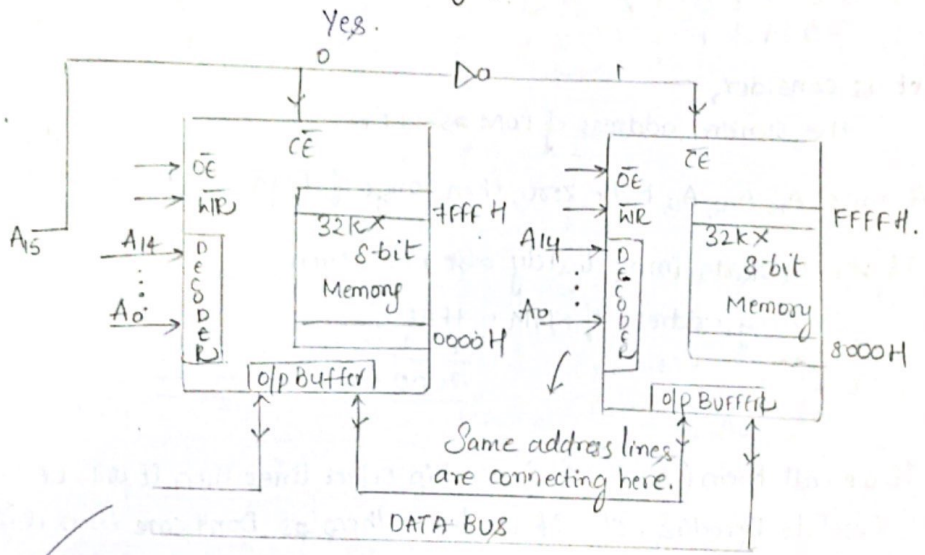
- If we don't need fold back memory, we have to handle those don't care address lines; then I have to connect A_{15} to $\overline{CE}$ then memory will enable only when $A_{15}=0$
  Then no more fold back memory, it becomes Absolute Memory

→ $A_{15}=0$ Address range = 0000 H to 7FFF H

if we put inverter,
then $A_{15}=1$, Address range = 8000H to FFFF H.

Can we connect two 32k memory?

Yes.



Span/size of each memory = 32k
Overall → 64k

Memory Map ⇒ several memory & I/O devices are connected.

21/9/22

Q. Design a circuit to interface following memory chips with 8085.

(i) 8k RAM

(ii) 8k ROM

No. of address lines = $\log(8\times1024)/\log(2)$

= 13.

- Among 16 address lines, 13 are required to point the memory locations.

	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 SELECT LINES			
ROM -	0 0 0	0 0 0 0 0	0 0 0 0	0 0 0 0	0000H
	0 0 0	1 1 1 1 1	1 1 1 1	1 1 1 1	1FFFH
RAM -	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

let us consider,

the starting address of ROM as-0000H. 0000H to

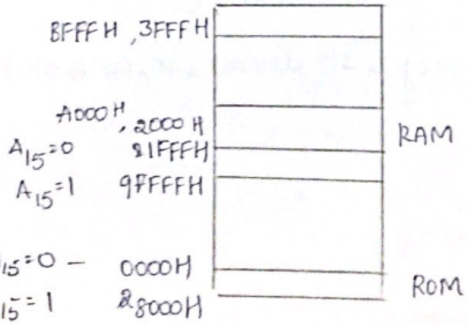
Assume A_{15}, A_{14}, A_{13} to be zero, then range of ROM will be 1FFFH.

- If 8k RAM starts immediately after ROM, then

starting address of RAM - 1FFF
+ 1
2000H

- If we call them (A_{15}, A_{14}, A_{13}) as chip select lines then it will be Absolute Decoding, else if we leave them as Don't care then it is Partial Decoding.

if A_{15} is don't care it can be 0/1.



There will be overlap of memory between RAM & ROM.

- No need to use chip select logic → We can leave all 3 as don't cares which is of low cost

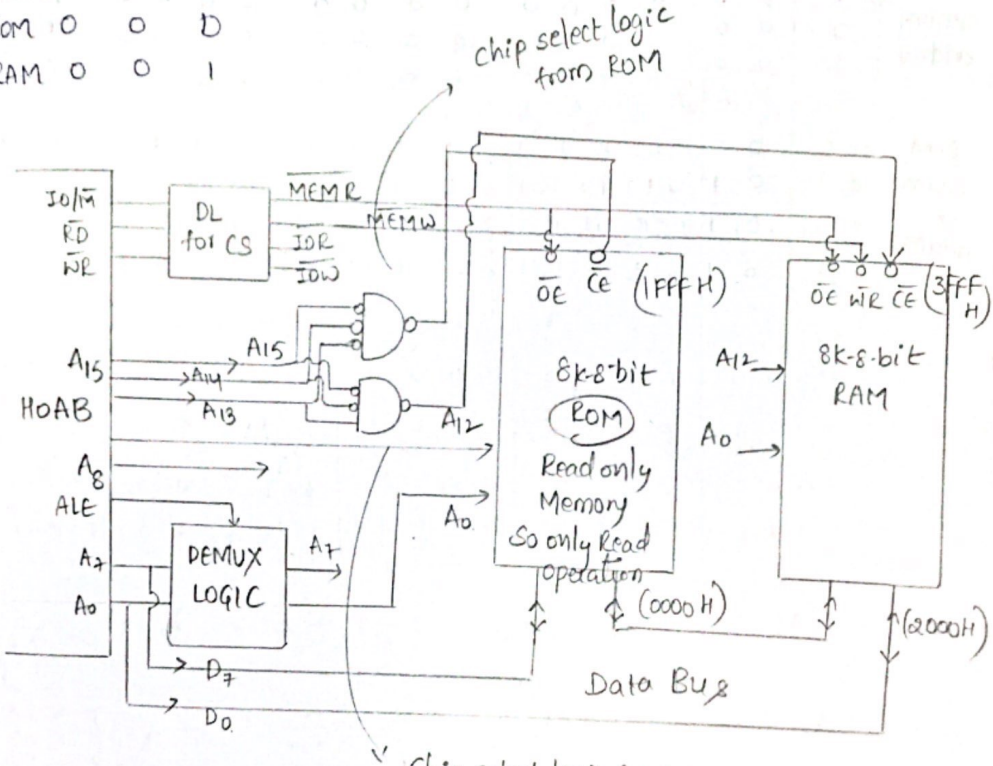
- If we leave one line then instead of 3 i/p and gate, I can take 2 i/p AND gate which reduces cost. (AND)

But there might get an issue in the hardware, so better to go to Absolute decoding → though it is costly.

Memory of one location shouldn't overlap with other location in case of mirror memory → this need to take care.

chip select lines

	A_{15}	A_{14}	A_{13}
ROM	0	0	0
RAM	0	0	1



We can also leave A_{14}, A_{15} as don't care and use A_{13} for chip select logic, "At the cost of mirror memory".

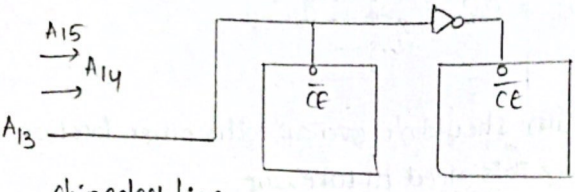
Connect A_{13} to ROM chip enable & $\overline{A_{13}}$ to RAM chip enable.

A_{15}	A_{14}	A_{13}
X	X	0
X	X	1

then each case we can have 4 different address aye for every location.

Question: What are all the consequences on the circuit, if A_{15} & A_{14} are considered as don't care lines.

Answer this !!



Chip select line

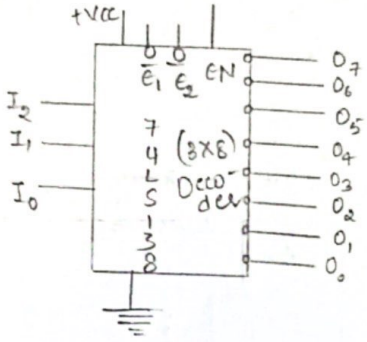
Address select lines

	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_{86}	A_5	A_4	A_3	A_2	A_1	A_0	
ROM starting address	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000H
	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000H
	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8000H
	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	C000H
ROM Ending st address	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1FFF H
	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	5FFF H
	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	9FFF H
	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	DFFF H.

22/09/22

- Interface 4 no.of 4 k memory chips, out of which 2 are ROM chips.

You can use 74LS138 Decoder.

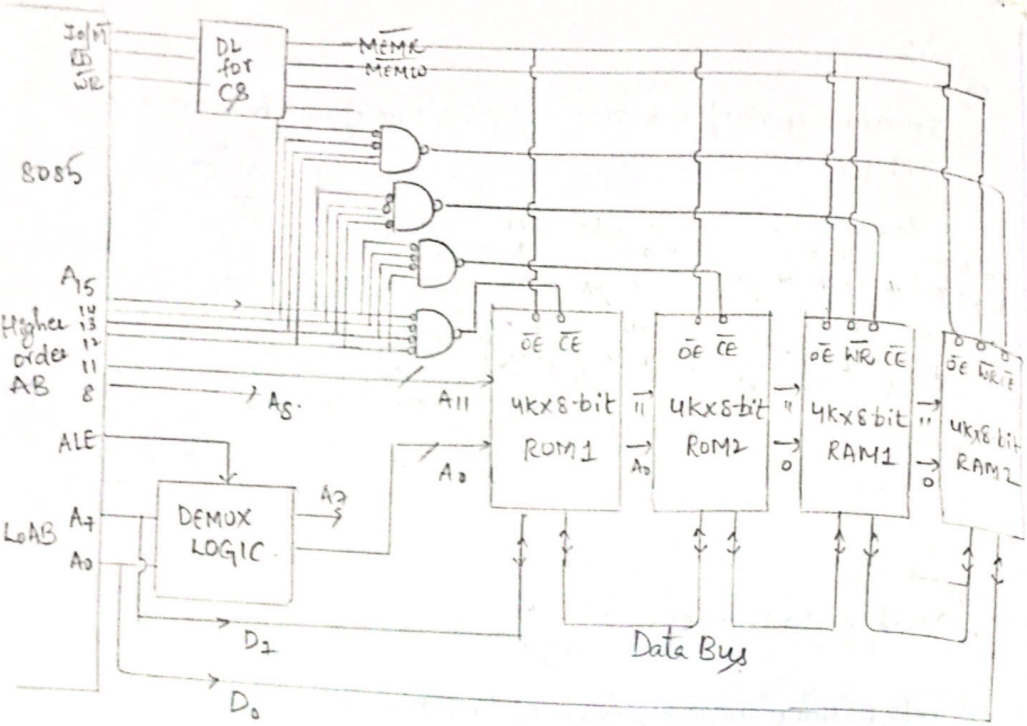


No. of address lines = $\frac{\log(4k)}{\log 2} = 12$

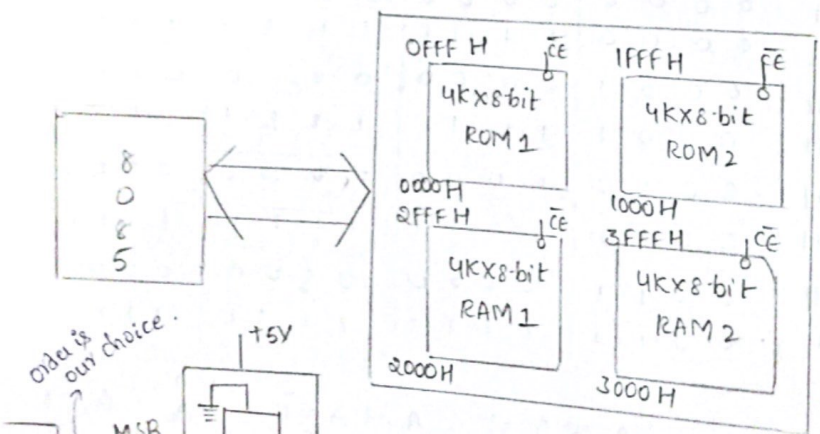
lets assume starting address of first ROM chip is 0000 H.

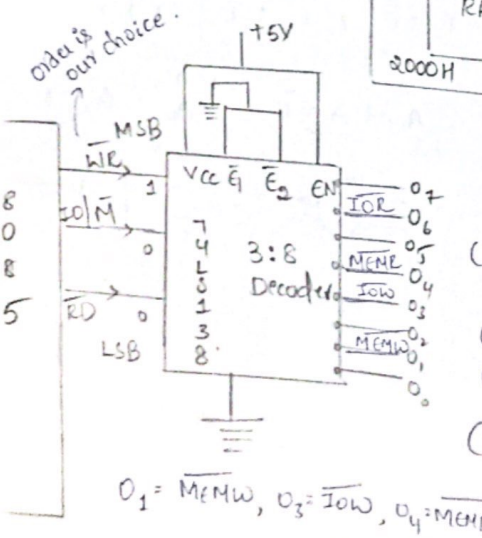
		Chip select lines				Address select lines											
		A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
1st ROM	0000H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0FFFH	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
2nd ROM	1000H	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	1FFFH	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
1st RAM	2000H	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	2FFFH	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1
2nd RAM	3000H	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
	3FFFH	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1

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



Instead of Digital logic for Control signal, We can use 74LS138 (3x8) Decoder



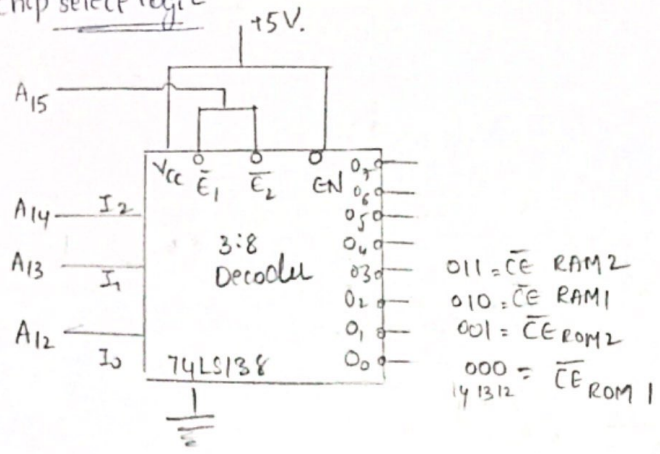


WR	IO/M	RD	0 1 2 3 4 5 6 7
X	0	0	
(MEMW) 0	0	1	1 0 1 1 1 1 1 1
X	0	1	
(IOW) 0	1	1	1 1 0 1 1 1 1 1
(MEMR) 1	0	0	1 1 1 1 0 1 1 1
X	0	1	
(IOR) 1	1	0	1 1 1 1 1 0 1 1
X	1	1	

$O_1 = \overline{MEMW}$, $O_3 = \overline{IOW}$, $O_4 = \overline{MEMR}$

$O_6 = \overline{IOR}$

Chip select logic



28/9/22

1) Interface the following memory with 8085.

a) 8K×8-bit RAM

b) 4K×8-bit ROM.

ROM
4K ⇒ 12 Address select lines

RAM
8K ⇒ 13 Address select lines

Steps:

1) find ASL for each chip (Address select lines)
2) Design chip select logic
3) Write circuit diagram

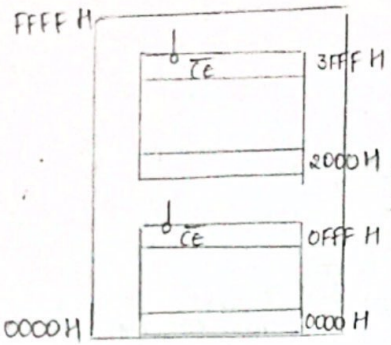
	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
ROM N=12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000H
	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0FFFH
RAM N=13	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2000H
	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3FFFH

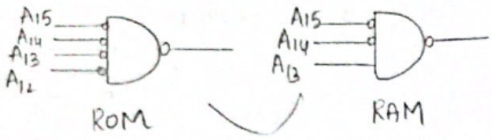
CSL — Address select lines — ASL

We can use any combination among all 8 possibilities except 000

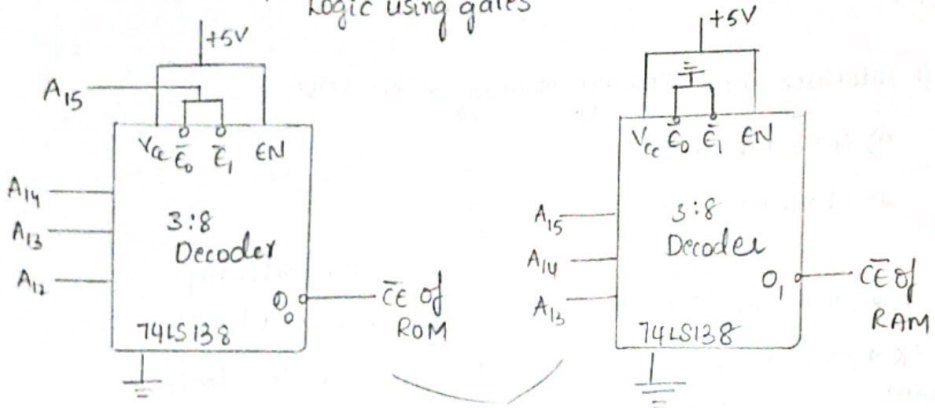
A_{12} → part of CSL in ROM is a part of ASL in RAM

it clashes with ROM





Chip Select Logic using gates



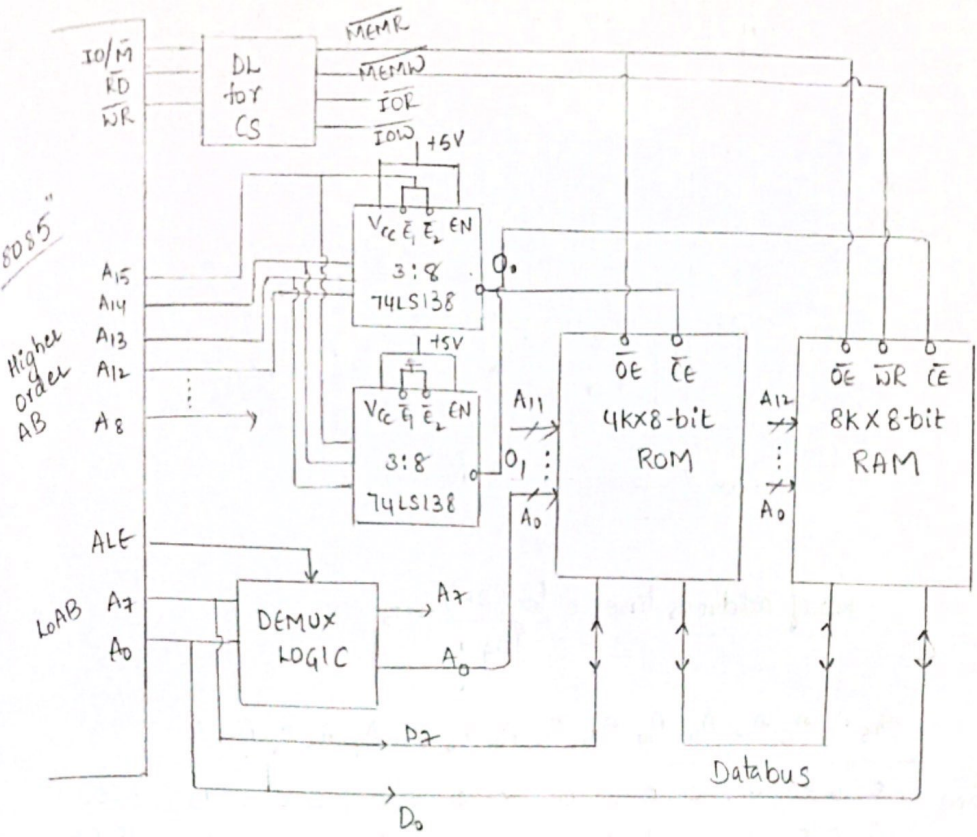
Individual Decoders for ROM & RAM.

chip select logic using Decoder

→ insert another 4K ROM in between them

a) 2,4K ROM
b) 1,8K RAM

	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		
ROM1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000 H	O_0 of first
	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0FFF H	
ROM2	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	01000 H	O_1 of first
	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1FFF H	
RAM	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2000 H	O_2 of second
	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3FFF H	



6/10/22

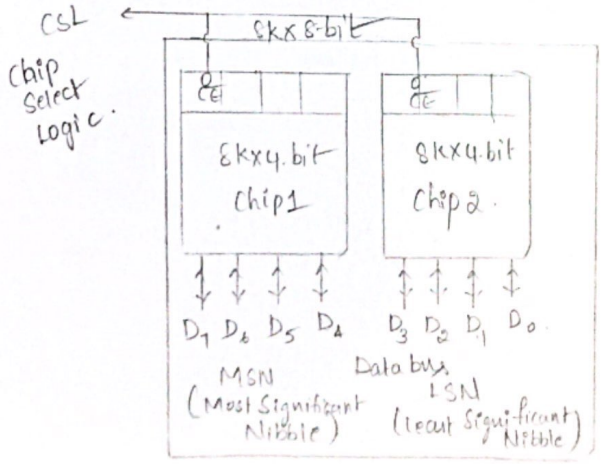
Question : Design an interface for.

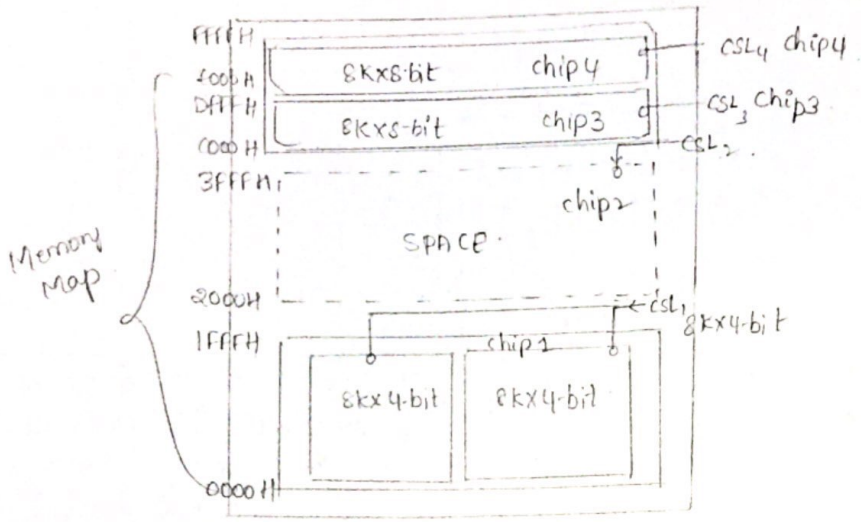
a) 8K RAM using 8kx4-bit memory chips 00000H.

b) Provide a space to interface additional 8k memory

c) 16k memory using 8kx8bit chips and ending address is FFFF H.

M x N





No. of address lines = $\frac{\log(8K)}{\log 2} = 13$

	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	
8K RAM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000H
	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1FFFH
8K space	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
16K Memory	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	C000H.
	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	DFFFH
	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	E000H
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	FFFFH

Chip select logic