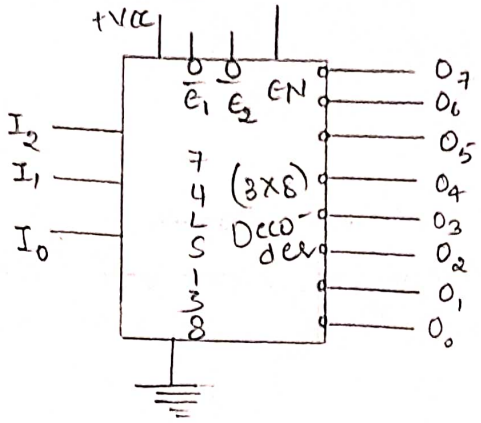
22/09/22

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

You can use 74LS138 Decoder.

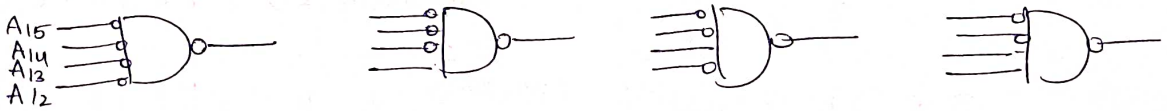


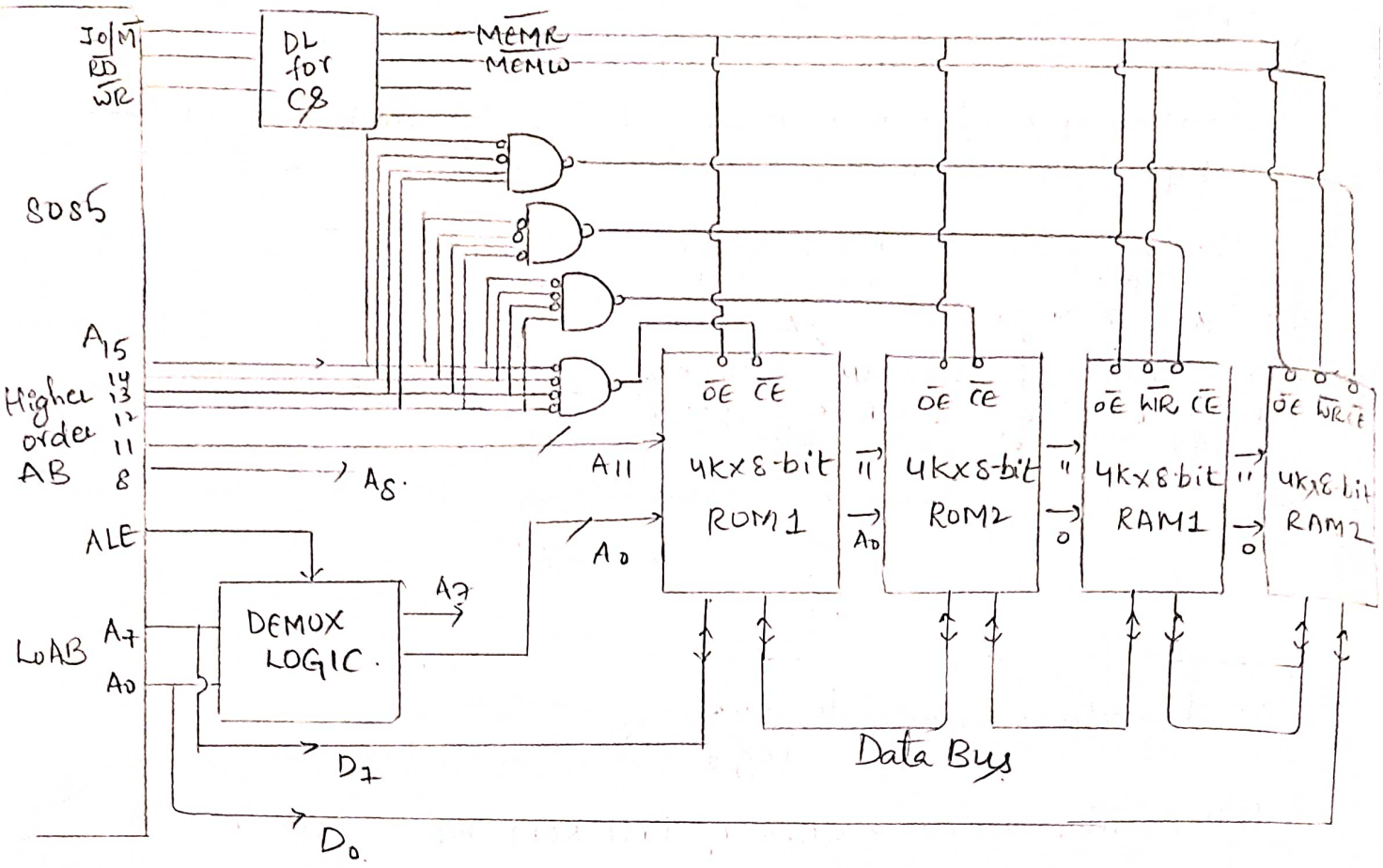
No. of address lines = $\frac{\log(4k)}{\log 2} = 12$

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

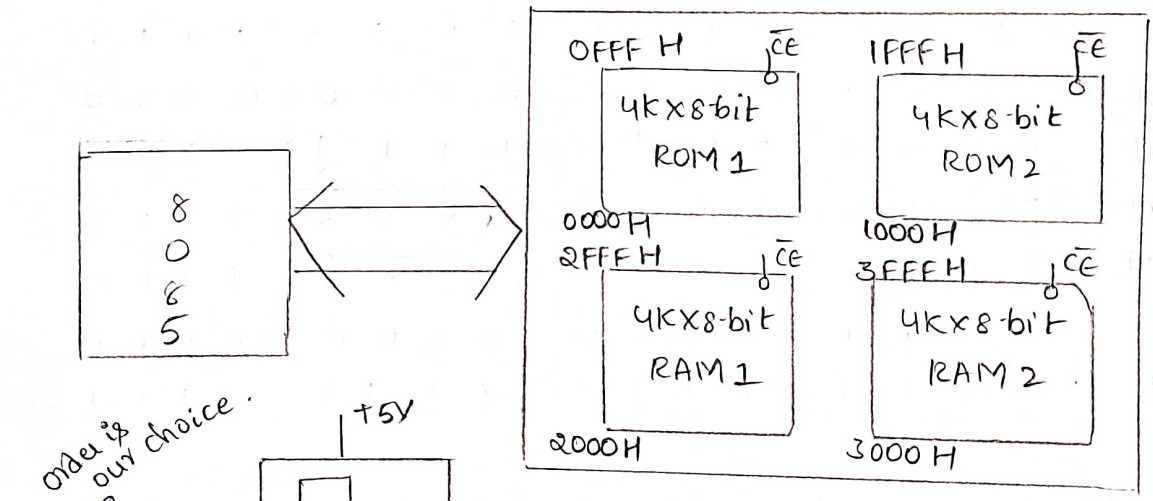
		chip select lines				Address select lines											
		A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
1st ROM	0000 H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0FFFH	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
2nd ROM	1000H	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	1FFFH	0	0	0	1	1	1	1	1	1	1	1	1	1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

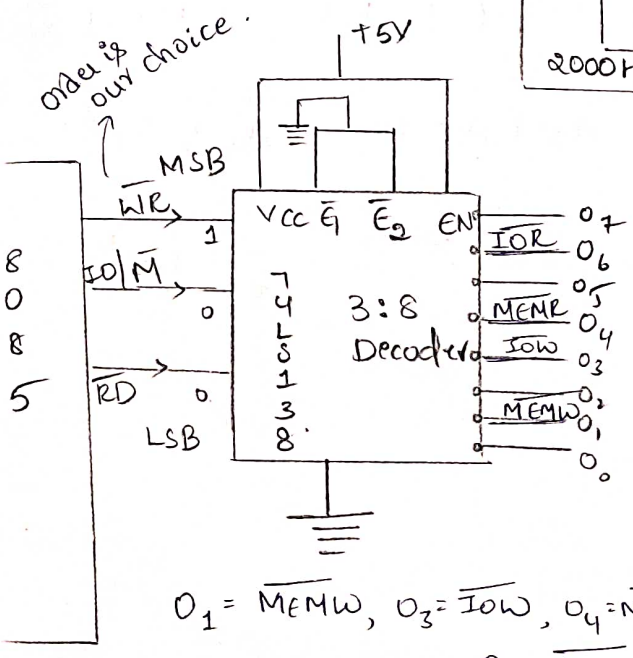
$A_{13}=0, A_{12}=0$ $A_{13}=0, A_{12}=1$ $A_{13}=1, A_{12}=0$ $A_{13}=1, A_{12}=1$





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



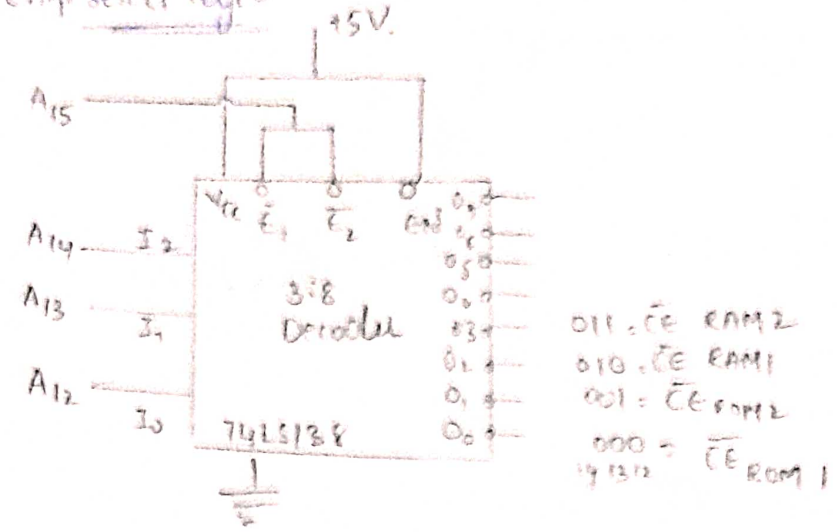


WR	IO/M	RD	0 1 2 3 4 5 6 7 (Hex)
X	0	0	0
(MEMW) 0	0	1	1 0 1 1 1 1 1 1
X	0	1	0
(IOW) 0	1	1	1 1 0 1 1 1 1 1
(MEMR) 1	0	0	1 1 1 1 0 1 1 1
X	1	0	1
(IOR) 1	1	0	1 1 1 1 1 0 1 1
X	1	1	1

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

$O_6 = \overline{IOR}$.

Chip select logic



28/9/22

1) Interface the following memory with 8085.

a) 8k x 8-bit RAM

b) 4k x 8-bit ROM.

ROM
4K → 12 Address select lines

8K → 13 Address select lines
RAM

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

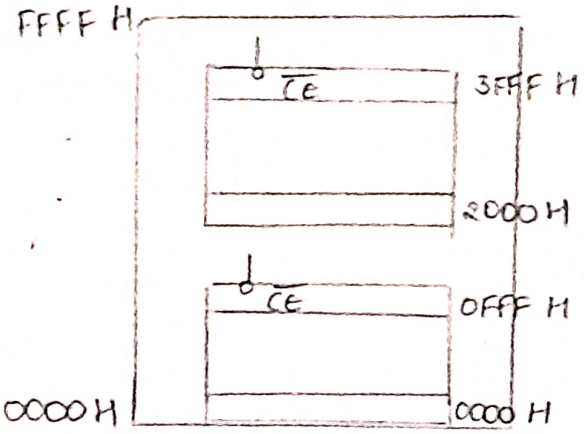
CSL: A_{15} A_{14} A_{13} A_{12} — Address select lines: A_{11} A_{10} A_9 A_8 A_7 A_6 A_5 A_4 A_3 A_2 A_1 A_0

We can use any combination among all 8 possibilities except 000

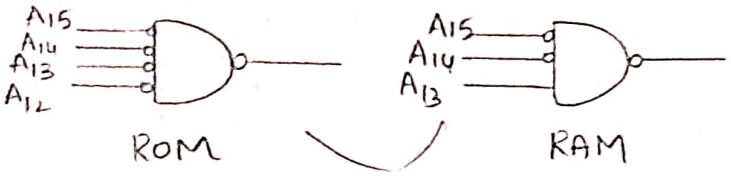
ASL

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

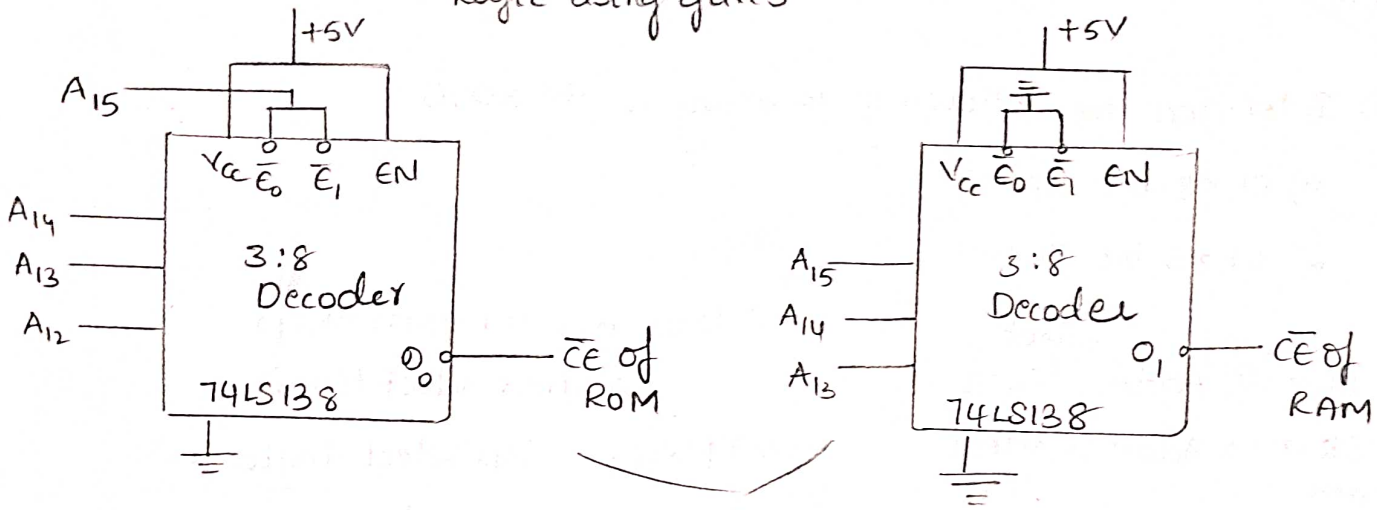
it clashes with ROM



} blank
No memory between 1000 to 1FFF H



Chip Select Logic using gates



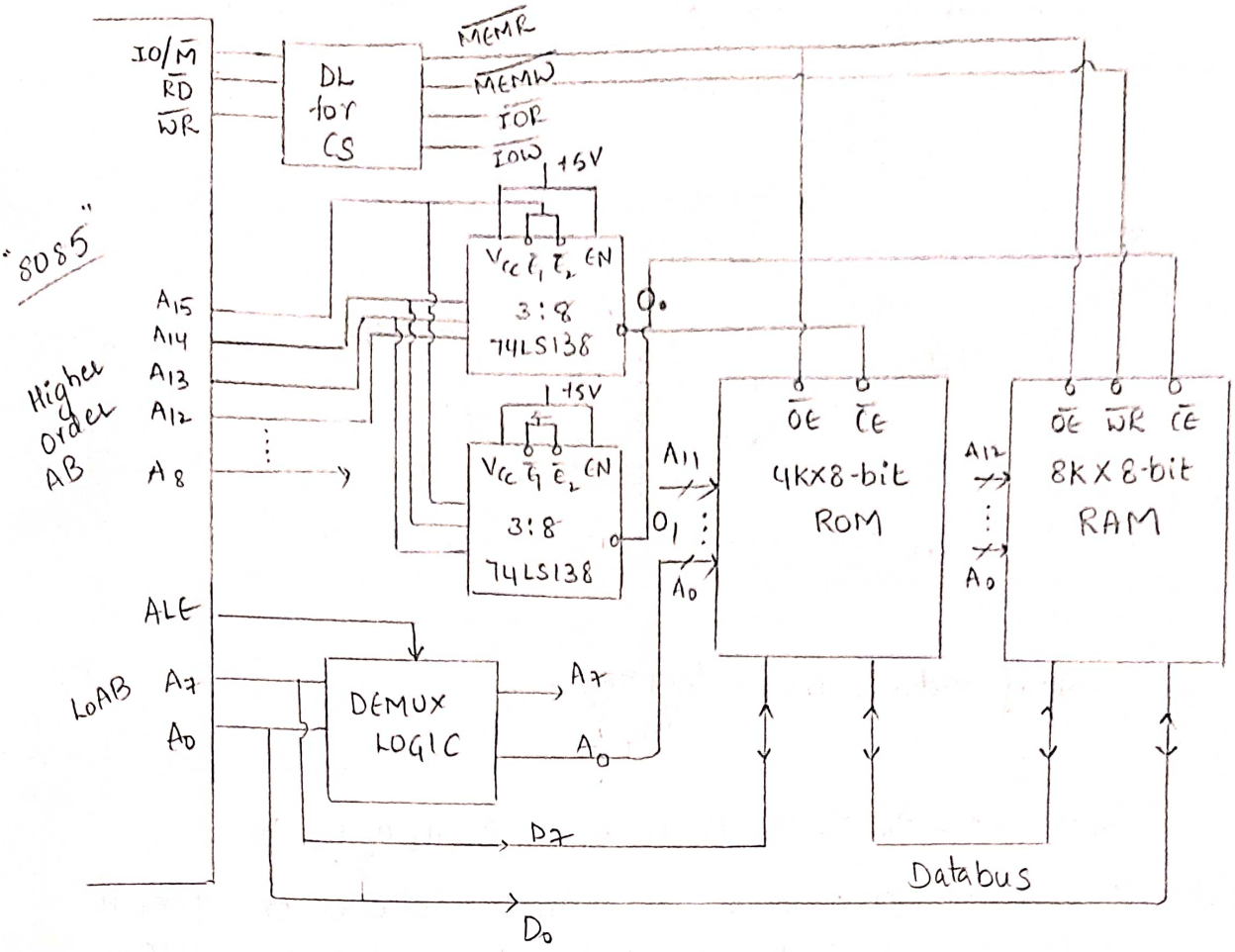
Individual Decoders for ROM & RAM.

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		
ROM 1	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	
ROM 2	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1000 H	O_0 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_1 of second
	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3FFF H	



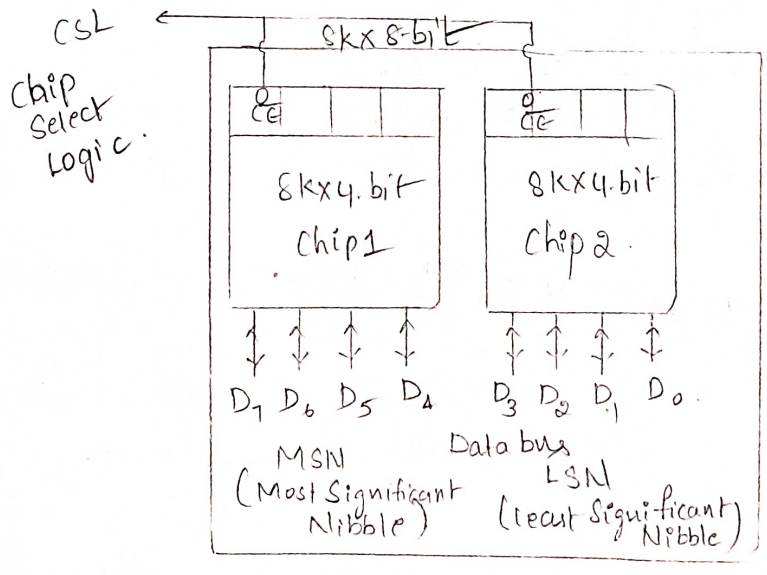
6/10/22

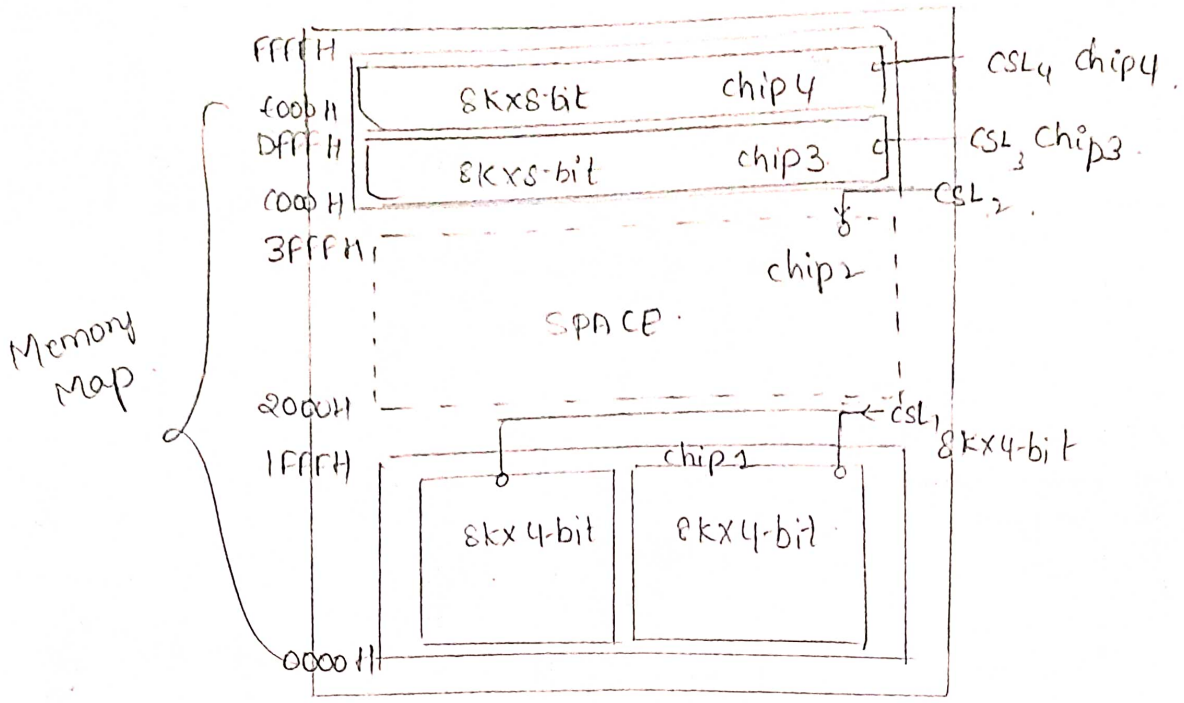
Question : Design an interface for.

a) 8K RAM using 8Kx4-bit memory chips 0000H.

b) Provide a space to interface additional 8k memory

c) 16k memory using $\frac{8kx8bit}{M \times N}$ chips and ending address is FFFF H.

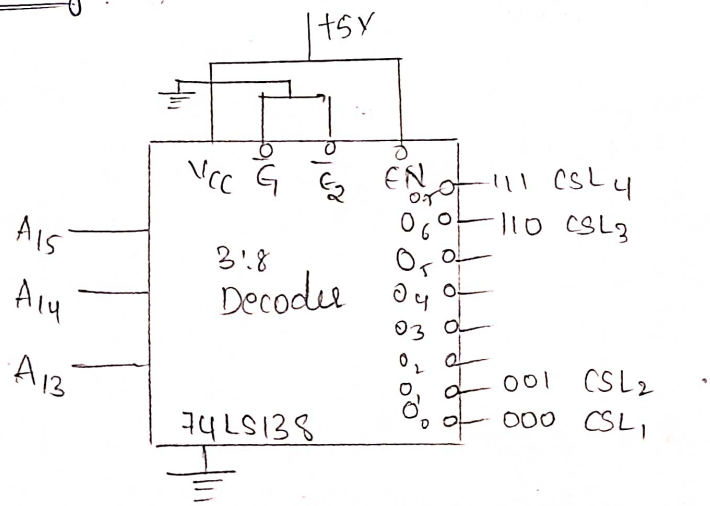


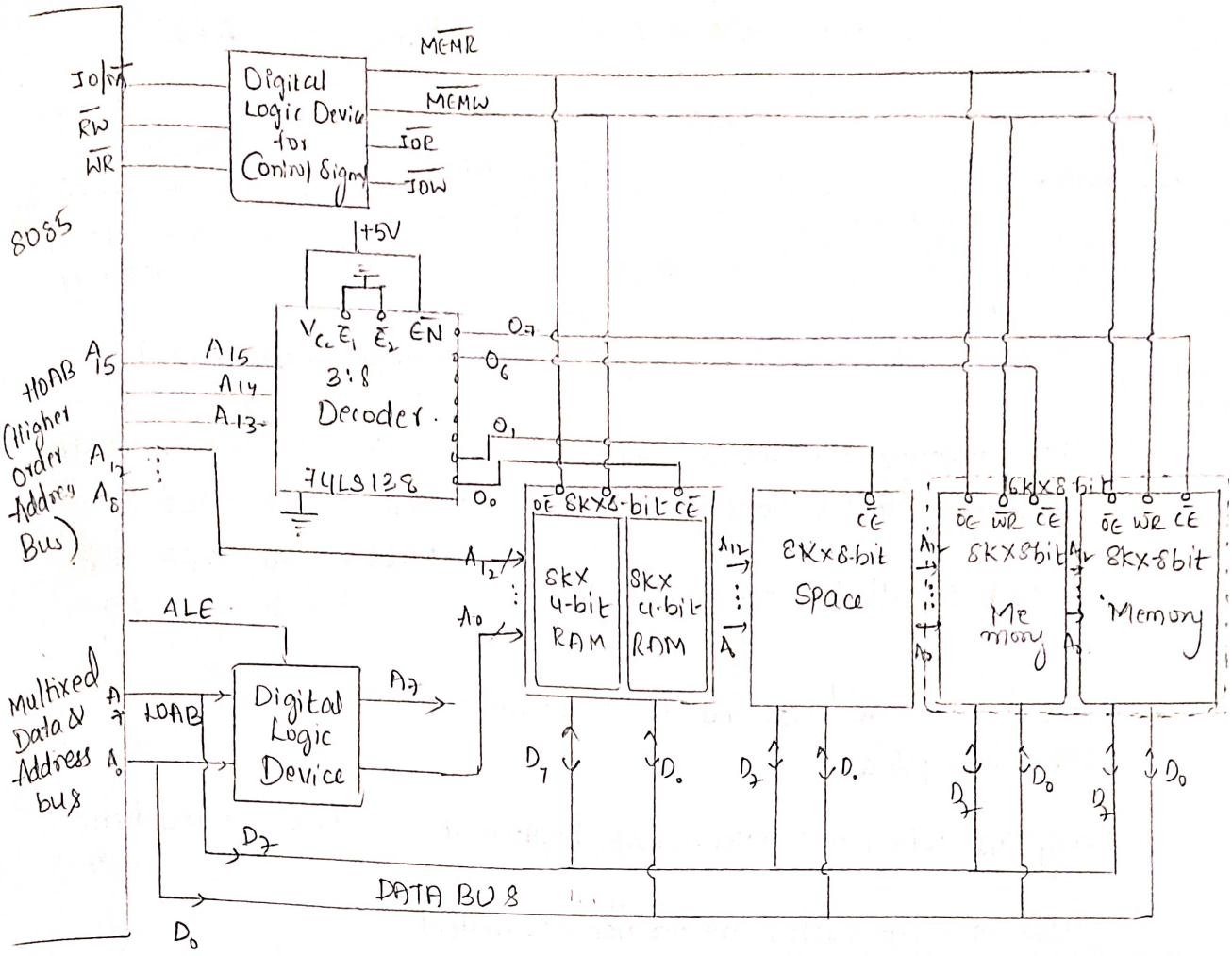


No. of address lines = $\frac{\log(8K)}{\log 2} = 13$

	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

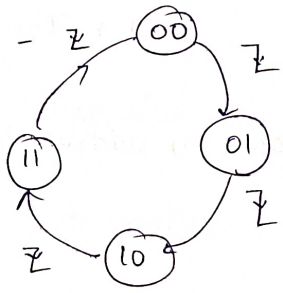




10/10/22

Digital state Machine

- 2-bit counter -

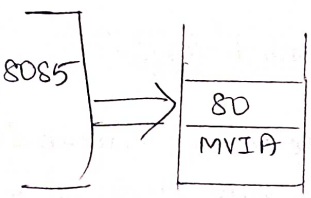


Upon the application of clock pulse, it is moving from one state to other state mc's

In synchronous with clock pulse.

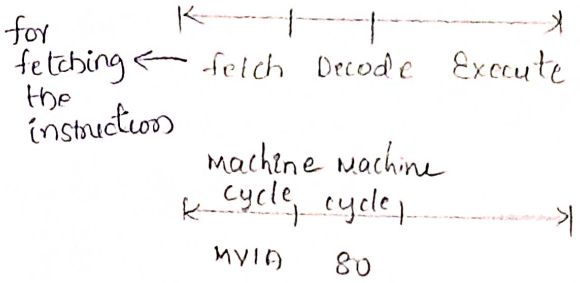
- Microprocessor is called as state machine
- Body also (breathe in and breathe out)

Consider the instruction MVI A,#80H.



Time taken by the processor to execute the a instruction completely is called instruction cycle.

INSTRUCTION CYCLE - Complete instruction



fetch - even it is also ROM but how it differents from ROM we will see later

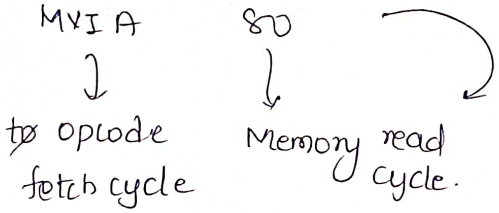
- While completing the instruction bus level activities happen.
  ↓
  which is called as machine cycle.

- Decoder decodes and timing & control unit generates signals so that data path is created.
  What is data path & how it is created?

- There can be one/more bus level activities, i.e one/more machine cycles.

Every time when bus level activity happen, it is called one machine cycle.

- These machine cycles are further decimated



further decimated & decimated to minute things.

Let's focus on Opcode fetch cycle

opcode fetch cycle
fetch + Decode

1. Content of program counter is placed on address bus
   AB ← PC
   PC ← PC+1 — T_1
2. $O \leftarrow \overline{RD}$, $O \leftarrow IO/\overline{M}$ — T_2 $O \leftarrow \overline{MEMR}$
3. DB ← [M] — T_3
4. IR ← DB
5. Decoding & Data path is created — T_4
   (Decoding & generate timing and control signals)

- control signals create/generate appropriate data path.

RTL
(Register Transfer level Activities. ⇒ These minute things happen in synchronous with clock pulse.

Thus everything happens in T states

- Each T state is in synchronous with clock pulse

∴ As counter moves from one state to other state, processor also moves from one state to other state in synchronous with clock pulse.

Thus, processor is also called Digital state Machine.

(- from start to end process move towards to execution)

Memory read

T1 . AB ← PC
PC ← PC+1

T2. D ← $\overline{MEMR}$

T3 : DB ← [M]
A ← DB — by the end of T3

Thus,

- Opcode fetch is 4 T states
- Memory level is of 3 T states
  Read

if memory read/write is 2T states.

- then it implies memory is twice slower than processor
- if frequency same for clockpulse & memory then it takes only 1 state (T)
- 2 T states ⇒ twice slower
- 3 & 4 T states ⇒ 4 time slower.

Instruction cycle) decimated
Machine cycle) decimated
RTL

Q. if frequency of clock pulse = 3MHz, the calculate time taken to execute MVI A, #80H

1) OF : 4T
2) MR : 3T
7T

1 T state = 1 clock pulse
if frequency = 3MHz

Time = $\frac{1}{3\times10^6}$ = a

7T = $7 \times \frac{1}{3} 10^{-6} = 2.3\times10^{-6}$ s = 2.3 μs.

This all is one way of representing the instruction.
- There is other way - pictorial represented (studied further)

MOV B,A | 1 Byte | 1 M/c | 4T | $4 \times \frac{1}{3} = 1.33\,\mu s$

MVI B, 80 | 2 Byte | 2 M/c | 7T | $7 \times 1/3 = 2.33\,\mu s$

LXI H, 4050 | 3 Byte | 3 M/c | 10 T | $7 \times 1/3 = 2.33$
7+3+3

MOV M,A | 1 Byte | 2 M/c | 7 T | 2.33 μs.
7+3

1) M/c OF cycle

T1: AB ← PC
PC ← PC+1

T2: D ← $\overline{MEMR}$

T3: DB ← [M]$_{AB}$
IR ← DB

T4: Decode & execute

2) M/c MW cycle

T5: AB ← HL

T6: D ← $\overline{MEMW}$

T7: DB ← A
[M]$_{HL}$ ← DB.

11/10/22

Pictorial Representation

Timing Diagram

Other than IO/$\overline{M}$, $\overline{RD}$, $\overline{WR}$ there are another two status signals S_1 & S_0.

	$\overline{MEMW}$	$\overline{MEMRD}$	S_1	S_0
Opcode fetch	1	0	1	1
MEMW	0	1	0	1
MEMR	1	0	1	0
HLT	1	0	0	0

↳ it is like opcode fetch only

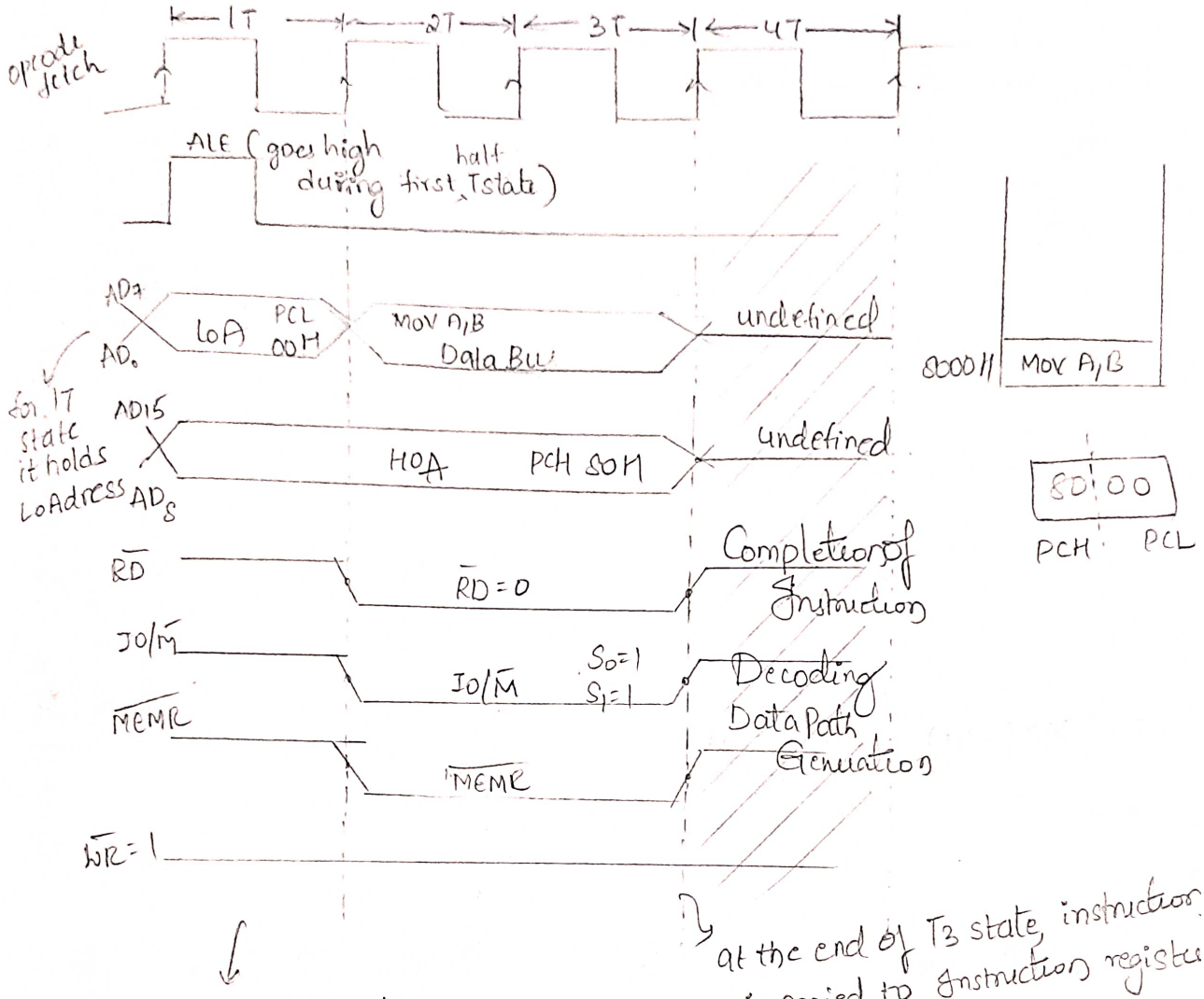
① MOV A,B A ← B

1 Byte | 1 M/c | 4T

Timing diagram of this instruction is the

↓

- Pictorial representation of all instruction cycle, machine cycles and Register Transfer level activities (T states).



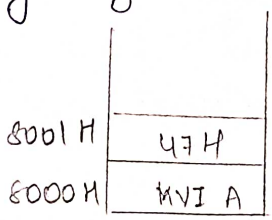
↓

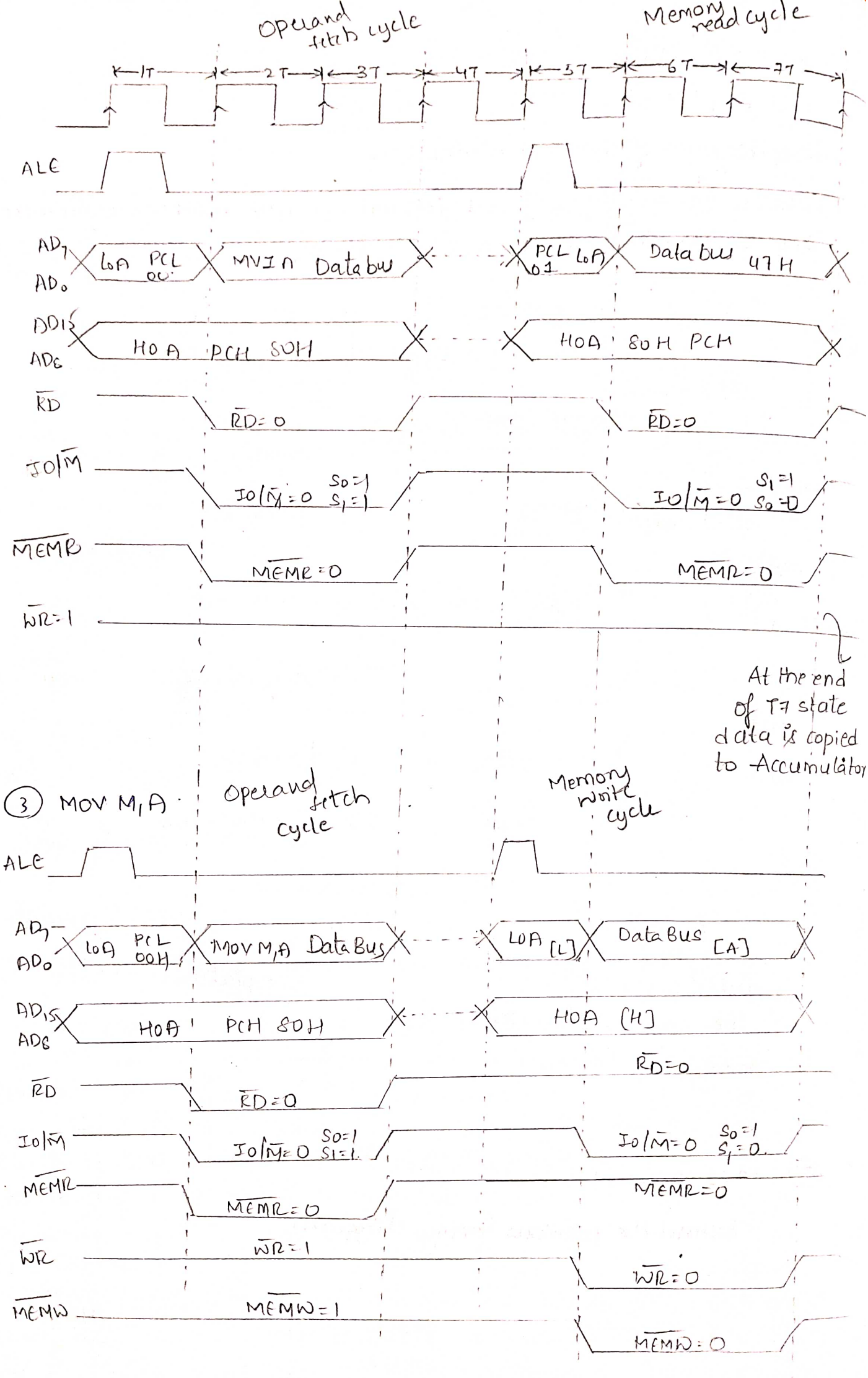
Pictorial Representation of the Register Transfer Level Activities of Processor

↳ at the end of T3 state, instruction is copied to instruction register

② MVI A, 47 H.

Extend the previous timing diagram.




Operand fetch cycle
Memory read cycle
T1
T2
T3
T4
T5
T6
T7
ALE
AD7
AD0
LOA PCL 00
MVI A Data bus
PCL 01
LOA
Data bus 47H
A15
A8
HOA PCH 80H
HOA 80H PCH
RD
RD=0
IO/M
IO/M=0 S0=1 S1=1
IO/M=0 S1=1 S0=0
MEMR
MEMR=0
WR=1
At the end of T7 state data is copied to Accumulator
3 MOV M,A
Operand fetch cycle
Memory write cycle
LOA PCL 00H
MOV M,A Data Bus
LOA [L]
Data Bus [A]
HOA PCH 80H
HOA [H]
RD=0
IO/M=0 S0=1 S1=1
IO/M=0 S0=1 S1=0
MEMR=0
WR=1
WR=0
MEMW
MEMW=1
MEMW=0