Microprocessor 8085

Timing Diagram

Instruction Cycle: the time required to complete the execution of an instruction

The 8085 instruction cycle consists of 1-6 machine cycles, or one to six operations.

Consists of 4 basic steps

(I) Fetch
(II) Decode
(III) Execute
(IV) Store

✓ An instruction requires several machine cycles

✓ Every instruction will require an opcode fetch (it is the first machine cycle of every instruction)

✓ Then after that, some instructions require mem. read / mem. write or two mem. reads or two memory writes

If you know the timing diagram of 5 basic machine cycles (Opcode Fetch, MR, MW, IOR, IOW), thereafter you need to draw them continuously, in a sequence to get the timing diagram of the instruction (together)

✓ So, the whole concept of timing diagram is to understand how to draw the 5 fundamental machine cycles.

Machine Cycle The time required to complete one operation of accessing memory, I/O, or acknowledging an external request.

transferring one value of data anywhere in the computer.

A machine cycle consists of 3-6 T-states.

T-state (T stands for transition) The machine cycle takes multiple clock periods. A portion of operation (one subdivision of the operation) performed in one system clock period is called as T-state.

✓ When we say how much time is required by this machine cycle means how many trigger are required or, on every clock pulse there is a transition from one state to another

Each T-state is precisely equal to one clock period. It is the basic unit to calculate the execution time of the instructions.

✓ The T-state starts at the falling edge of a clock.
✓ Time here is given in terms of T-states (clock pulses)

Timing Diagram — Every clock pulse is a trigger to the MPU to do something now.

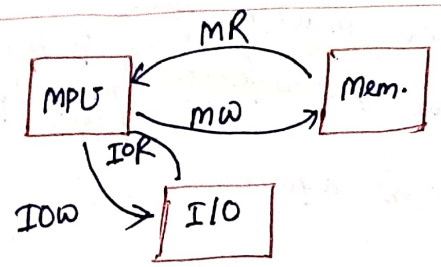
It is a graphical representation of the execution time taken by each instruction.

The execution time is represented in T-States.

In 8085 $1\ T\text{-state} = \frac{1}{f} = \frac{1}{3\ MHz} = 0.333 \times 10^{-6}$ sec

$= 0.333\ \mu sec$

$= 333\ n sec$



Basic machine cycles

- Opcode Fetch
- Memory Read
- " Write
- I/O Read
- I/O Write

In 8085, data bus is of 8-bits, hence in one machine cycle only 1 byte of data is transferred.

S_1 S_0 for the sake of simplicity

Machine Cycles	IO/$\overline{M}$	$\overline{RD}$	$\overline{WR}$	S_1	S_0	T-States
Opcode Fetch	0	0	1	1		4T/6T
Mem. Read	0	0	1	1	0	3T
Mem. Write	0	1	0	0	1	3T
I/O Read	1	0	1	1	0	3T
I/O Write	1	1	0	0	1	3T

✓ 3 T-States mean 3 triggers, i.e, there are three small activities taking place in the m/c cycle.

⇒ Why 3 activities?

✓ Suppose you think about Mem. Read Operation.
(i.e., MPU is reading data from Memory)

that means first MPU will give some address to identify the mem. location from where the data is stored.

✓ When previous machine cycle is over, when new machine cycle (M/C) is going to start, MPU will give the address of mem. location

✓ Previous M/C cycle, is over, MPU is sleeping,
Something has to trigger MPU,

✓ We, the human beings the living things understand time is passing
✓ But MPU is a dead thing/object.
For MPU time means trigger.
Something triggers MPU to do a new activity.

When the previous M/C cycle, is over,
the first trigger will tell MPU to produce ~~a new~~ an address.
That means in the first T-state, MPU will release an address
which will be placed on the address bus,
go to the memory
and select the location.

(I)

it will take some time,
in nano seconds
though a negligible time is required to select a mem. location
but in the Timing Diagram it will be looking like a big time
because we are drawing everything in nanoseconds.

(II) The next thing MPU has to do is
make $\overline{RD}$ to LOW value
asking for data stored in location selected

Now, should the MPU give the address and make the $\overline{RD}$ Low simultaneously?

⇒ Not possible,

how can MPU ask for data (making $\overline{RD}$ LOW) before the location is selected?

That means after giving Address MPU has to wait for sometime (though a very small amount of time but here everything will be shown in nano seconds) for the location to be selected.

That means MPU will be sleeping while it is waiting,

That means we need another trigger to tell MPU now make $\overline{RD}$ signal Low

That means first T-states tells the MPU to realse the address

Second T-state makes MPU to make the $\overline{RD}$ signal LOW

$\overline{RD}$ goes LOW that means that '0' value travels to Memory and Memory Releases Data and that Data travels to MPU

This travelling takes some time which is called propagation delay.

Now, MPU will see data is coming through the data bus,

Now MPU has to capture the data and store it into the appropriate register.

To do that also MPU will take some time.

How much time?

=) infinite time?

No, MPU does not understand time.

✓ Someone has to trigger MPU that is the Third T-state.

✓ Hence, 3rd T-state tells MPU now Capture the data and store it in appropriate reg.

That is why Mem. Read operation will take 3 T-states.
The same goes on Mem. Write
I/O Read and
I/O Write operations.

Mem. Write

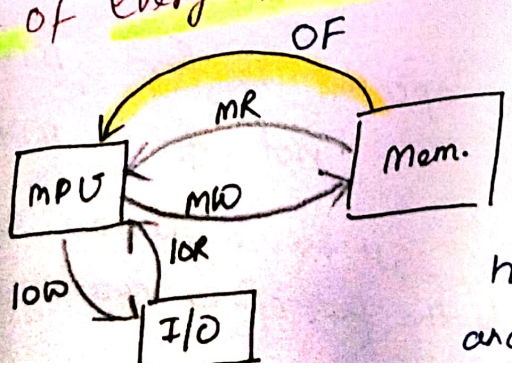
First T-state – MPU will give the address
2nd T-state – MPU will give the $\overline{WR}$ control signal (make it LOW)
3rd T-state – Memory will capture the data.

✓ Other than the four m/c cycles (MR, MW, IOR, IOW) there is one major m/c cycle (opcode fetch) which is present in every instruction because every instruction has an opcode.
Hence, this m/c cycle is compulsory.

To execute an instruction, MPU first fetch the instruction, to fetch an instruction, MPU first of all fetch its opcode.
✓ So, the first m/c cycle of every instruction is opcode fetch.
✓ Then depending on the complexity / simplicity of the instruction there may or may not be other m/c cycles.

✓ Every timing diagram of every instruction will begin with ~~the~~ opcode fetch.

✓ Opcode Fetch is a Mem. Operation
Hence $IO/\overline{M}$ is 0



Opcode Fetch is a READ operation, hence $\overline{RD} = 0$ and $\overline{WR} = 1$

✓Someone has to trigger MPU that is the Third T-state.

✓Hence, 3rd T-state tells MPU now capture the data and store it in appropriate reg.

That is why Mem. Read operation will take 3 T-states.
The same goes on Mem. Write
I/O Read and
I/O Write operation.

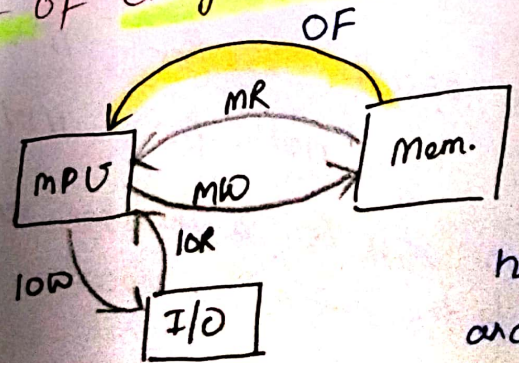
Mem. Write

First T-state – MPU will give the address
2nd T-state – MPU will give the $\overline{WR}$ control signal (make it LOW)
3rd T-state – Memory will capture the data.

✓ Other than the four m/c cycles (MR, MW, IOR, IOW) there is one major m/c cycle (opcode fetch) which is present in every instruction because every instruction has an opcode.
Hence, this m/c cycle is compulsory.

To execute an instruction, MPU first fetch the instruction, to fetch an instruction, MPU first of all fetch its opcode.
✓So, the first m/c cycle of every instruction is 'opcode fetch.'
✓Then depending on the complexity / simplicity of the instruction there may or may not be other m/c cycles.

✓ Every timing diagram of every instruction will begin with opcode fetch.



✓Opcode Fetch is a Mem. Operation
Hence IO/$\overline{M}$ is 0

Opcode Fetch is a READ operation, hence $\overline{RD}=0$ and $\overline{WR}=1$

Opcode Fetch is an Output Operation / Input operation?

⇒ Input operation (for the MPU)

Hence $S_1 = 01$ $S_0 = 0$

But that's a problem because if $S_1 S_0 = 10$ then the rows for two m/c cycles will becoming the same.

	IO/M	$\overline{RD}$	$\overline{WR}$	S_1	S_0
Opcode Fetch	0	0	1	1	1
Mem. Read	0	0	1	1	0

But opcode fetch and mem. read are not the same operation.

In both the cases, we getting / transferring 1 byte of data/ information from Mem. to MPU.

✓ In opcode fetch, we fetch opcode from mem. to MPU

✓ In Mem. Read, we get the data from " to MPU

Ex.

MVI B, 25H ; B ← 25 H

↓

The opcode is for MVI B

not for MVI B, 25 H

as 25H is a number and number can not be the part of an opcode.

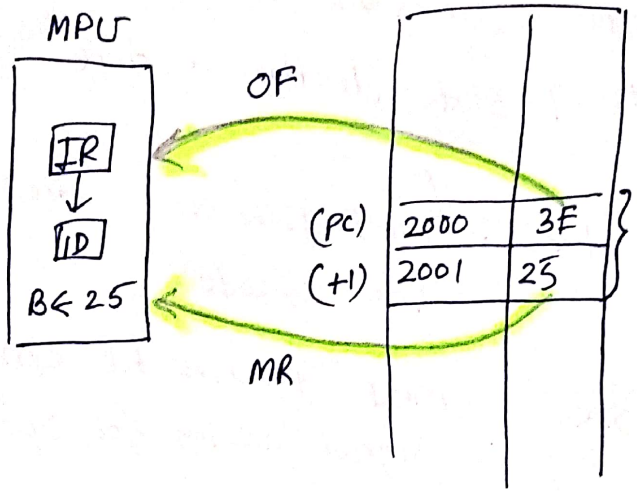
MVI B, 25H

Opcode Operand

1 byte for Opcode
1 " " Operand

Hex Code
3E
25

It is a 2-byte of instruction, hence it requires two MR cycles.



In both the cases, we transfer 1 byte of information.

When MPU fetches 3E, it comes to Instruction Reg. (IR) Then it will go to Instruction Decoder (ID) which will decode the opcode. and understands 3E stands for MVI B

Then MPU will know that B is requiring the value and understand one more byte is required to complete the instruction.

Difference between Opcode Fetch and Memory Read

Both will transfer 1 byte of information, they are similar but not the same.

But in opcode fetch we get opcode which has to be decoded. In Memory Read we get the data which need not to be decoded, data can be directly put in the B register.

Opcode Fetch = Mem. Read + Decode

In Opcode Fetch, you do everything as you do in Mem. Read plus we have to decode the opcode.

Hence Opcode Fetch and Mem. Read can't have same set of signals. Hence to distinguish an Opcode Fetch from MR

We set the status signals as

S_1	S_0
1	1

; Opcode Fetch

- Opcode Fetch requires 4 T-states.
- First 3 T-states are same as Memory Read.
- In the 4th T-state, decoding is done.

1st T-state : MPU gives the address of the opcode

2nd T-state : MPU gives the $\overline{RD}$ control signal (asking for opcode)

3rd T-state : MPU captures the opcode and store it in IR.

- Once the three T-states are over and the opcode comes into the MPU, it takes one more T-state to decode the opcode. This is why Opcode Fetch requires 4 T-states.

And some instructions which are little more complex in nature has 6 T-states for Opcode Fetch.

(There are 8 instructions in the Instruct set that have 6 T-states opcode fetch)

Demultiplexing the Bus $AD_7 - AD_0$

(separating out address and data from the multiplexed bus)

$A_{15} - A_0$

- $A_{15} - A_8$ (this bus always carry address, we don't bother about this bus)
- $A_7 - A_0$, $D_7 - D_0$ → Common Bus (multiplexing)

$AD_7 - AD_0$ can carry addr/data

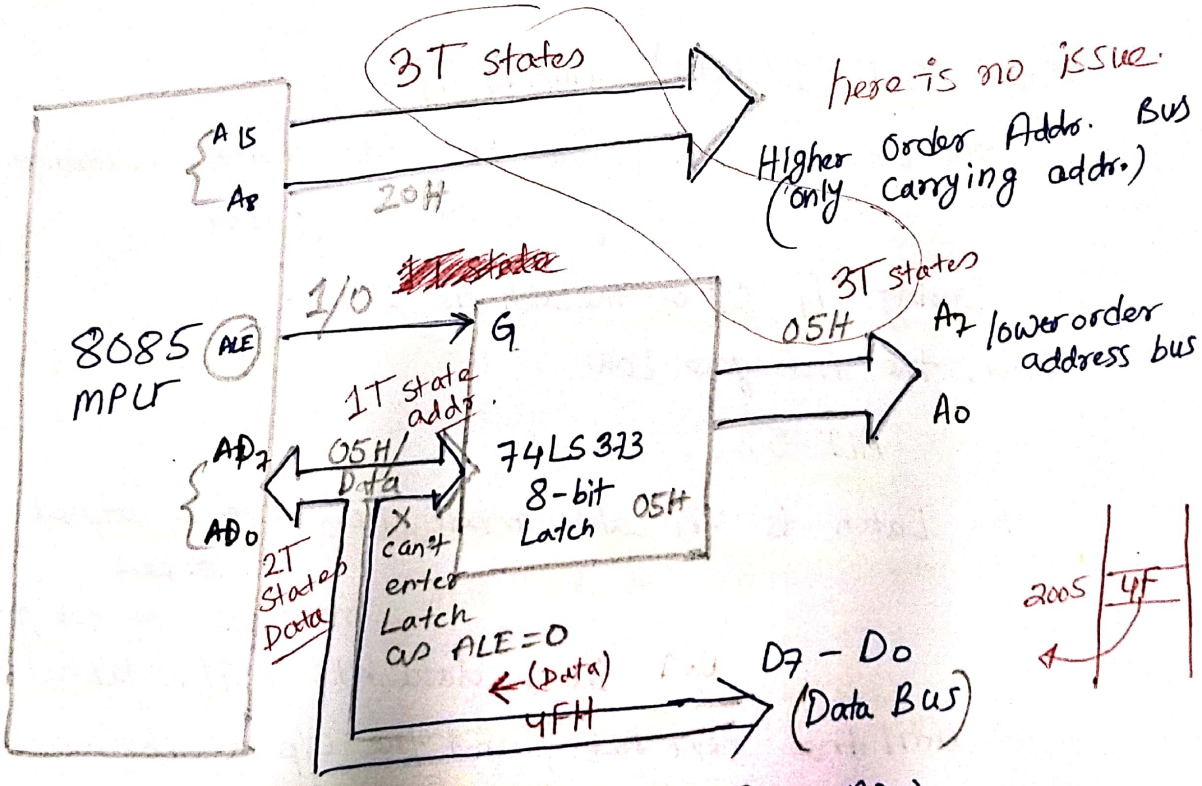
But this bus

There is a signal ALE (Address Latch Enable)

ALE = 1 → the Bus ($AD_7 - AD_0$) is carrying address

ALE = 0 → the Bus ($AD_7 - AD_0$) carrying data

Locth	Hex Code	Instruction
2005H	4FH	MOV C, A



For low order Address / Data Bus ($AD_7 - AD_0$) we need to separate out Addr. and Data (i.e., demultiplexing)

(*) What is the need of multiplexing first and then followed by Demultiplexing.

⇒ Multiplexing was done to reduce the no. of pins (less hardware) in the MPU.

Once they are out of the MPU, on the ckt ($AD_7 - AD_0$) we need separate Addr. Bus and separate Data "

(*) Latch is used to store/hold the value even if we stop sending the value (giving i/p to latch) the latch will continue producing the value. (We have used D-FF to Hold the value)

✓ There are 8-Flip flops and can store an 8-bit information

When ALE = 1 (High) during T_1

the Latch is transparent i.e., o/p changes according to the i/p data.

During T_1 o/p of the latch is 05H.

When the ALE goes Low

ALE = 0,

the Latch is in Latch mode, i.e., o/p is latched or locked irrespective of the i/ps.

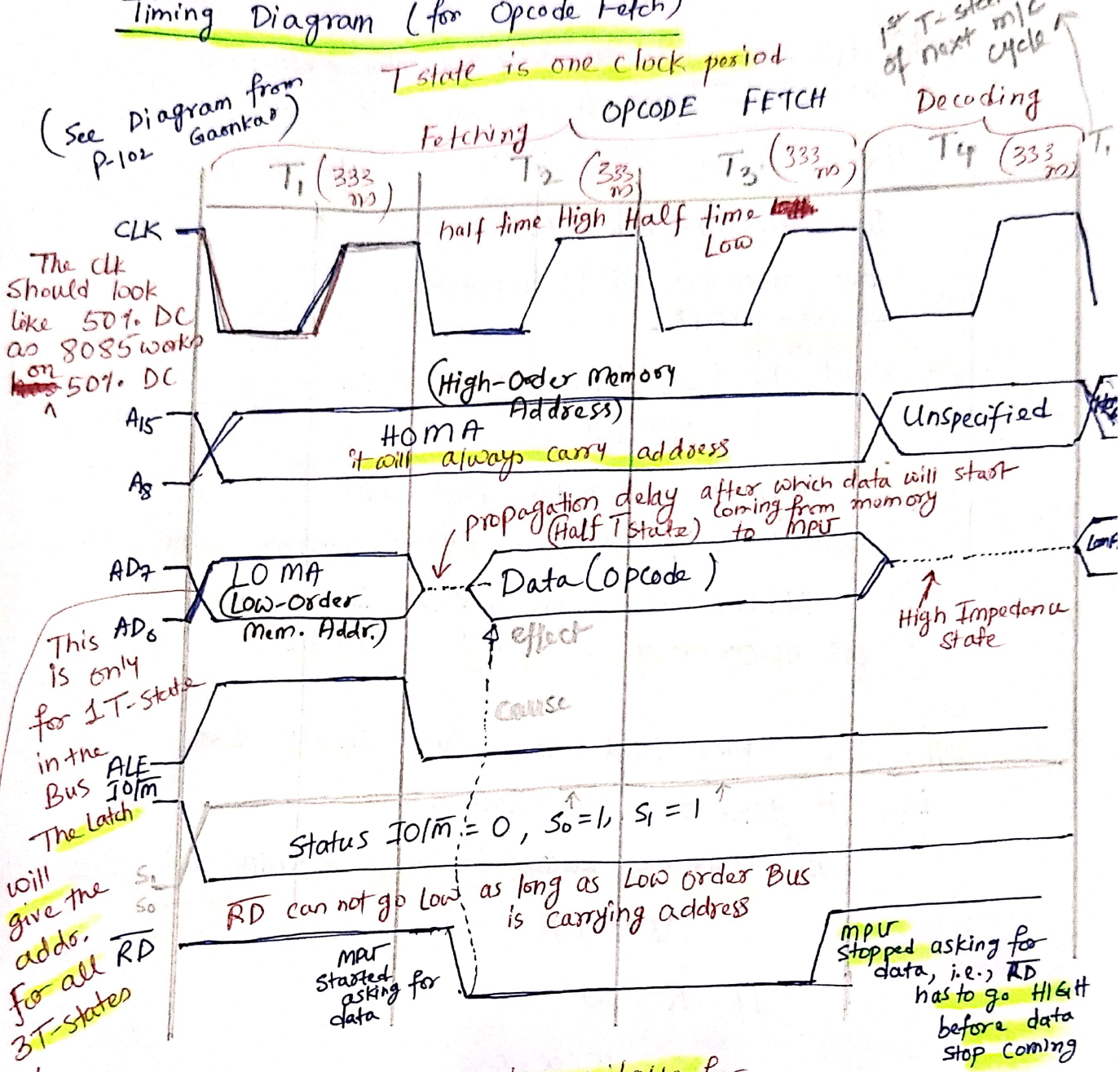
i.e., data byte 05H is latched

until the next ALE, and the o/p of the Latch represents the low-order address bus A_7-A_0

Timing Diagram (for Opcode Fetch)

(See Diagram from Gaonkar p-102)

T state is one clock period



i.e.,
Our Full Address (A15 – A0) is available for all 3 T-states

AD0–AD7 will carry Addr. for sometime and data for sometime.

→ initially AD7 – AD0 ~~LOMA~~ will carry address,
When it will carry address the Latch should capture the address,

⇒ Who tells the Latch that bus (AD7 – AD0) is carrying address?

⇒ (ALE)
That is when the bus is carrying address, ALE = 1 (for First T-state), HOMA is ~~High~~ for 3T-states and LOMA is for 1T states
The Latch will give the address for all 3T states.

- The data coming from the selected mem. location is captured by MPU by the end of 3T-states.
- That is, Fetching of opcode is over after 3T states.
- Now decoding will happen
- But decoding will be happenning inside MPU.

⇒ What are the buses doing at that time?

NOTHING.

The High-Order Addr. Bus will carry an unspecified value (Garbage Value)

Nobody cares for the value is carries. as no Operation is going on the Bus

The Low-order Addr. Bus just shuts down i.e., it goes to High Impedance.

its resistance becomes infinite.

$$V = IR$$

If $R = \infty$

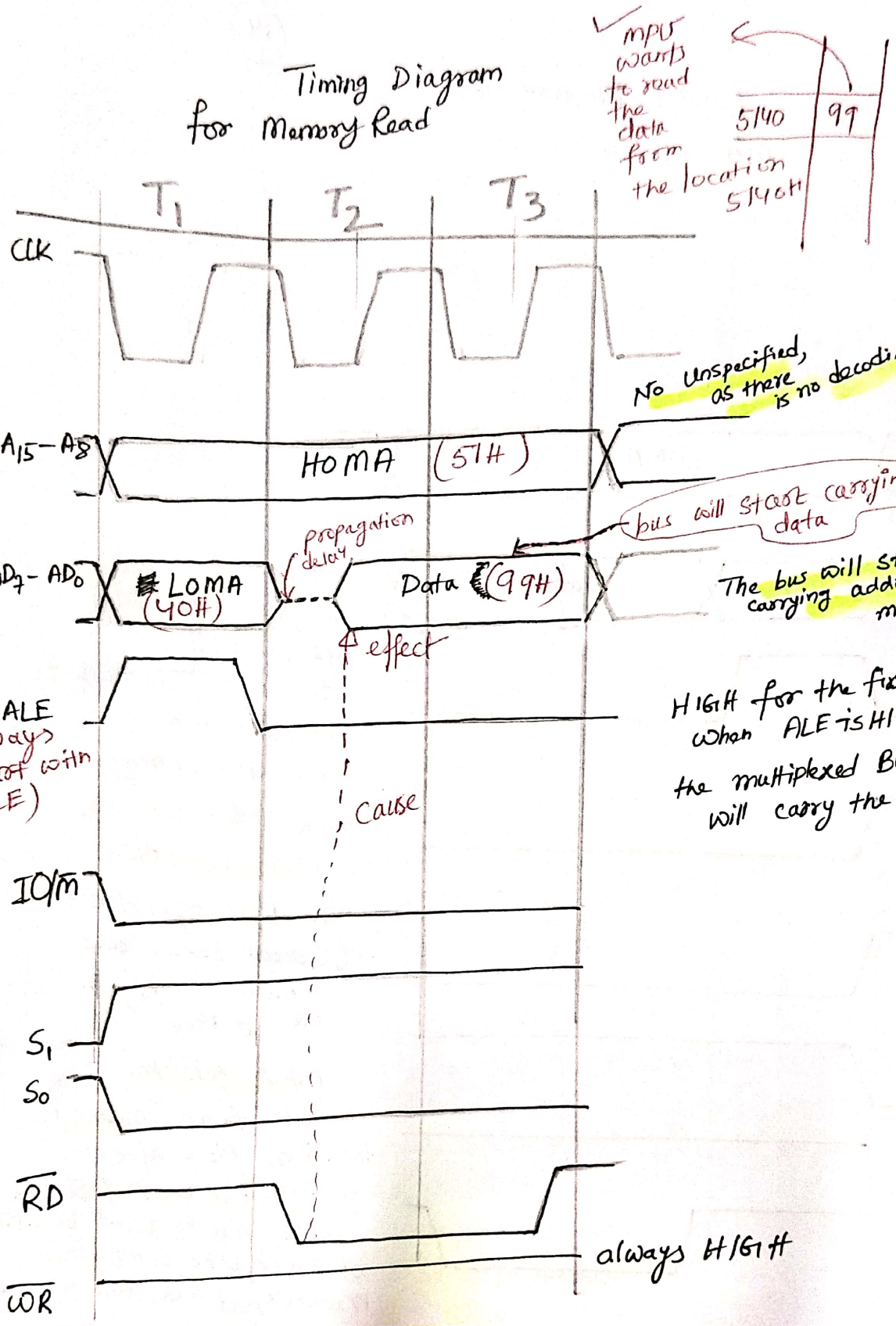
$$V = I \times \infty$$

$$I = \frac{V}{R}$$

$$= \frac{V}{\infty} = 0$$

Whatever voltage you apply, you are not able to pass current(I) as R is ∞

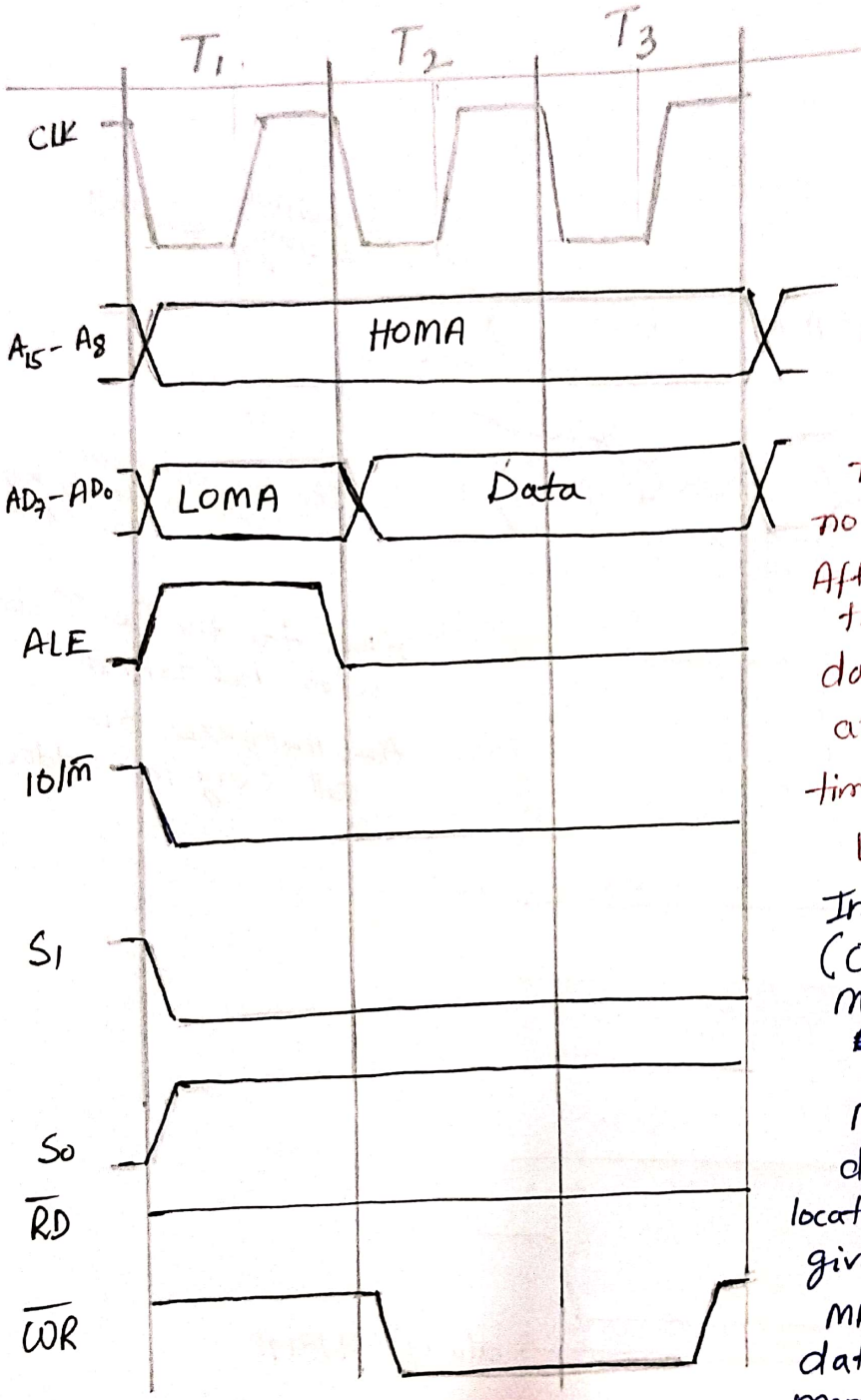
Timing Diagram for Memory Read



RD will not go low in the 1st T-state as in 1st T-state Bus is carrying address.

As soon as address is out of the Bus RD goes Low and it will be remaining Low till the end of 3rd T-state.

Timing Diagram for Memory Write



This bus is having no propagation delay. After address there is immediately data because we are drawing the timing diagram of Write Operation

In Read operation (Opcode Fetch ant Memory Read,), ~~the~~ to Read

MPU gets the data from memory location, Now after giving $\overline{RD}$ control signal MPU has to wait because data has to come from memory and take time to reach MPU (data has to travel) that is nothing but the propagation Delay. Now, when we do the Write Operation, Address is given by MPU as well Data " " " MPU as MPU will give address, immediate MPU will give the data after that.

(*) Ex.

MVI A, 32H

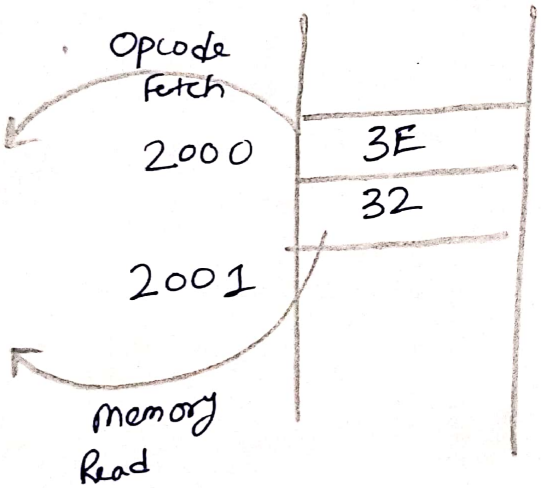
Mem. location
2000H
2001H

Machine Code

0011 1110	→ 3EH
0011 0010	→ 32H

Instruction

MVI A, 32 ; Load
; Byte 32H
; in the Acc.



So, we need to draw the Timing Diagram of Opcode Fetch (4 T-states) and the Timing Diagram of Memory Read (3 T-states)