

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.
 When we say how much time is required by this machine cycle means how many triggers are required, A portion of operation (one subdivision of the operation) performed in one system clock period is called

Each T-state is precisely equal to one clock period.

On every clock pulse there is a transition from one state to another. It is the basic unit to calculate the execution time of the instructions.

(2)

✓ The T-state starts at the falling edge of a clock.

✓ Time here is given in terms of T-states (clock pulses). Every clock pulse is a trigger to the MPU to do something now.

Timing Diagram

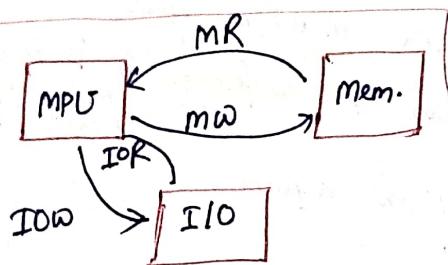
It is a graphical representation of the execution time taken by each instruction.

The execution time is represented in T-states.

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

$$= 0.333 \text{ microsec}$$

$$= 333 \text{ nsec}$$



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

Basic machine cycles

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

for the sake of simplicity

Machine Cycles	$\overline{I/O/M}$	\overline{RD}	\overline{IOR}	S_1	S_0	T-States
Opcode Fetch	0	0	1	1	0	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.

(3)

⇒ 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 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 is over,

the first trigger will tell MPU to produce a ~~new~~ 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.

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 RD to LOW value

asking for data stored in location selected



Scanned with OKEN Scanner

Now, should the MPU ~~give~~ 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 realize the address.

Second T-state ~~tells~~ 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.

(III) ✓ 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 WR control signal (make it Low)

3rd T-state - Memory will capture the data.

✓ Other than the four m/c cycles there is one major m/c cycle (opcode fetch)

(MR, MW, IOR, IOW)
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.

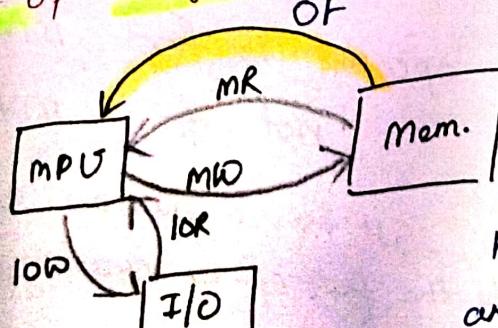
✓ So, the first m/c cycle of every instruction is opcode fetch.

✓ 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

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

✓ Opcode Fetch is a Mem. Operation
Hence $\overline{I0}/\overline{m}$ is 0



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

(5)

✓ 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 there is one major m/c cycle (opcode fetch) (MR, MW, IOR, IOW)

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.

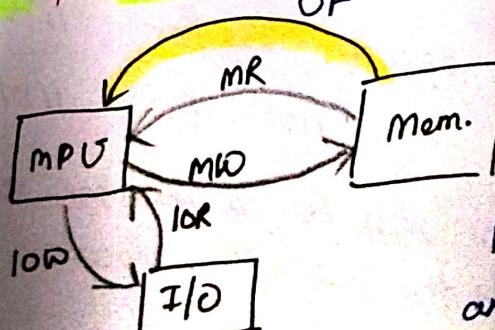
✓ So, the first m/c cycle of every instruction is opcode fetch.

✓ 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

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

✓ Opcode Fetch is
a Mem. Operation
Hence $IOW \bar{m}$ 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 become the same.

	10/M	RD	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 ~~data~~^{opcode} from mem. to MPU
- ✓ In Mem. Read, we get the data from " to MPU

Ex:-

MVI B, 25H ; $B \leftarrow 25H$



The opcode is for MVI B
not for MVI B, 25H

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

MVI B, 25H

Opcode

25H

Operand

1 byte for Opcode

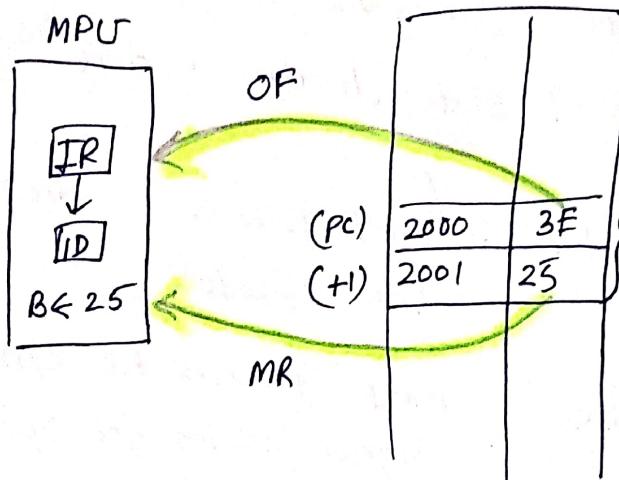
1 " " Operand

Hex Code

3E

25

It is a 2 byte of instruction, hence it requires two ^{mc} 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 : MPUR gives the address of the opcode

2nd T-state : MPUR gives the RD control Signal (asking for opcode)

3rd T-state : ^{MPUR} Captures the opcode and store it in IR.

Once the three T-states are over and the opcode comes into the MPUR, 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.

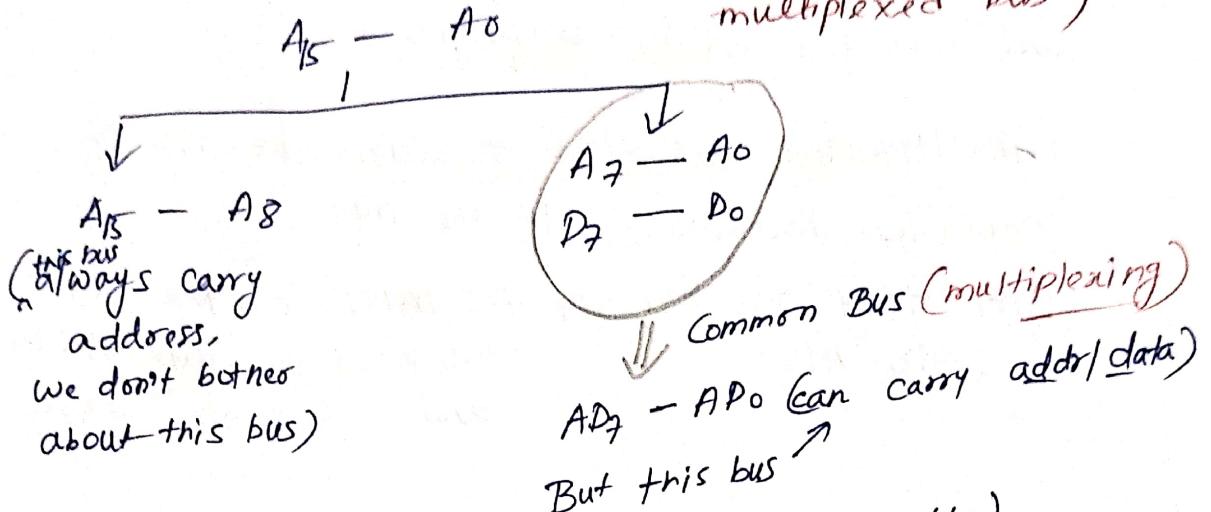
(There are 8 instructions

in the instruct set that have 6 T-states opcode fetch)

9

Demultiplexing the Bus $AD_7 - AD_0$

(Separating out address and data from the multiplexed bus)



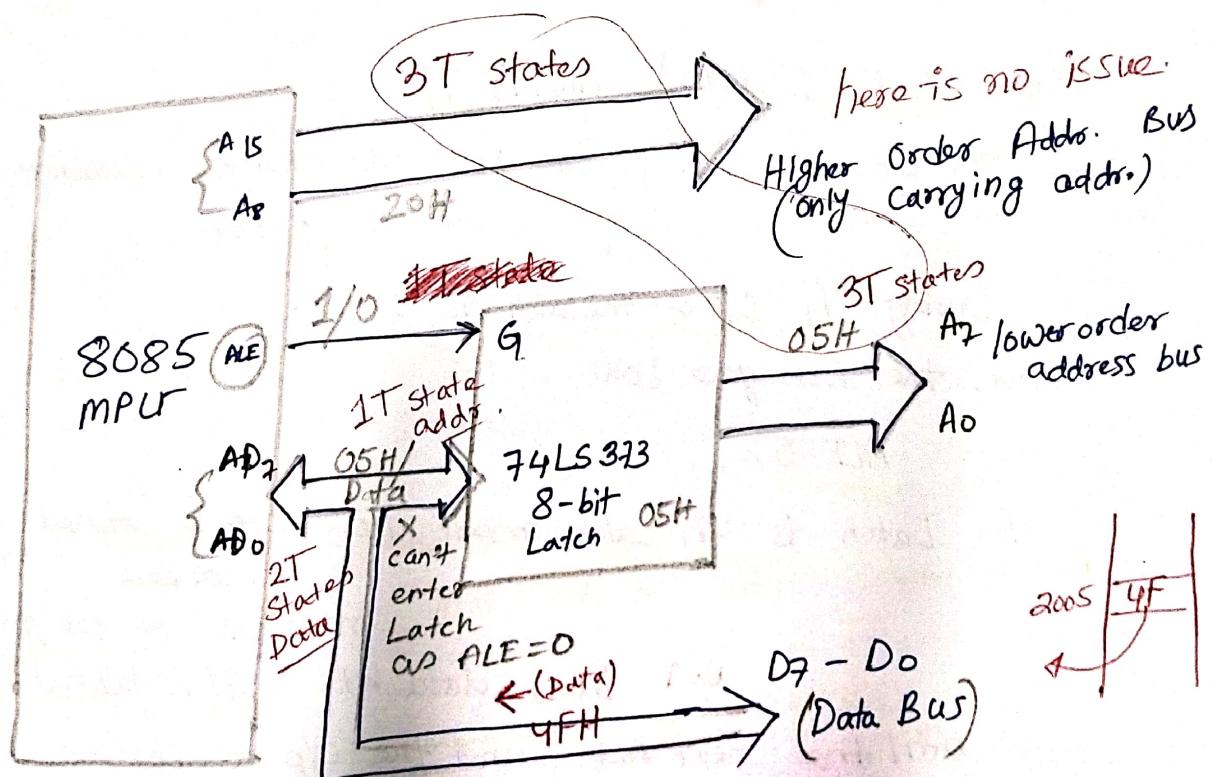
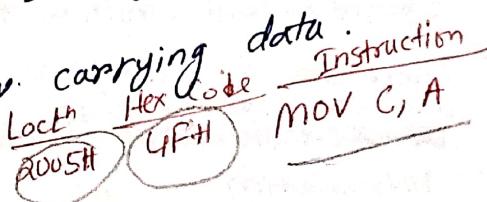
There is a signal ALE (Address Latch Enable)

$ALE = 1$

the bus is carrying address
($AD_7 - AD_0$)

$ALE = 0$

the bus carrying data
($AD_7 - AD_0$)



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



Scanned with OKEN Scanner

* 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 (AD₇-AD₀) 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 to 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 T₁

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

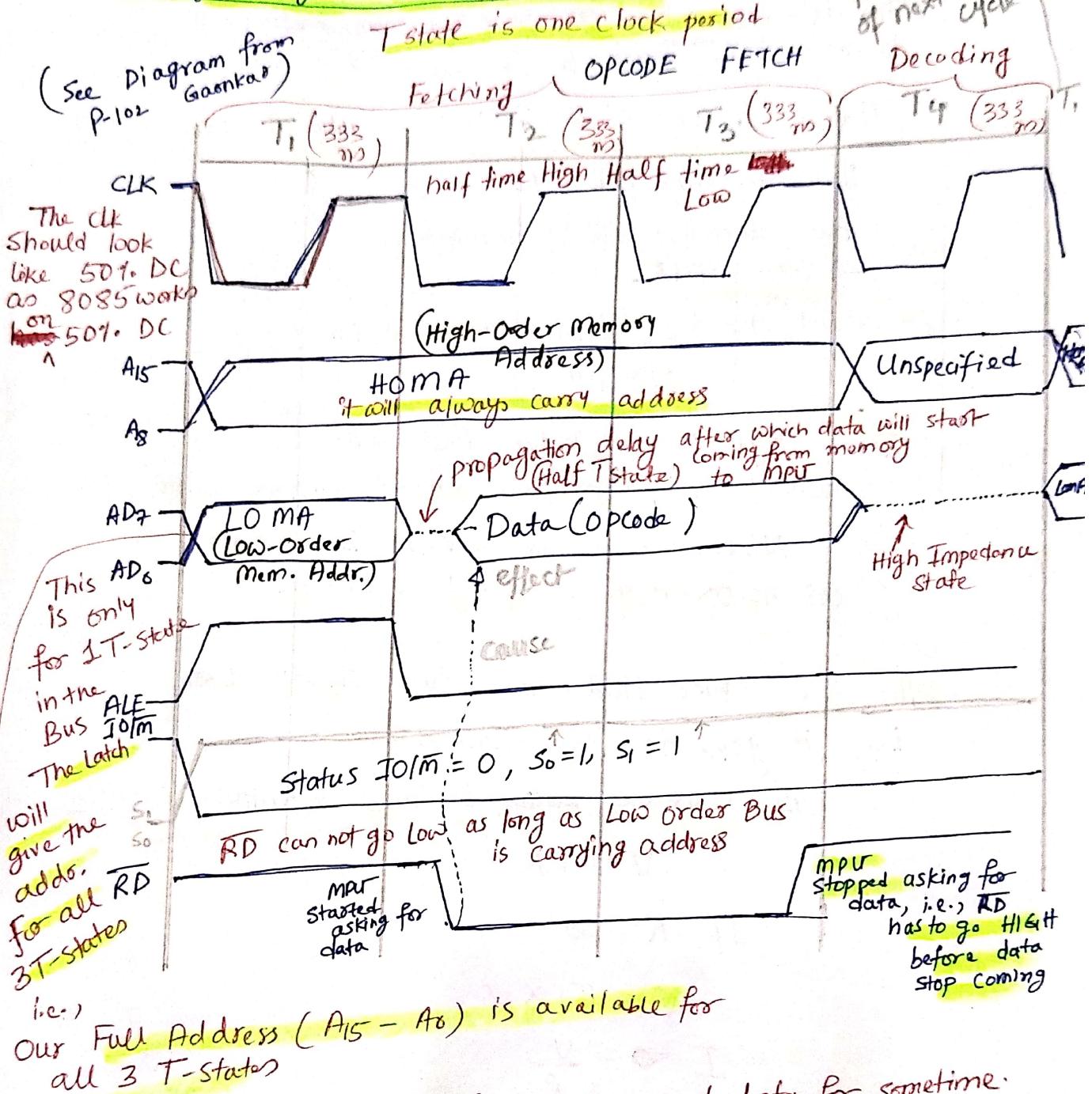
During T₁ O/p of the latch is 05H.

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/p's. i.e., data byte 05H is latched until the next ALE, and the O/p of the Latch represents the low-order address bus A₇-A₀

Timing Diagram (for Opcode Fetch)



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

→ initially AD₇-AD₀ ~~will~~ 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 ~~for~~ for 3T-states and LOMA is for IT states. The Latch will give the address for all 3T states.

- ↙ The data coming from the selected mem. location is captured by MPLT by the end of 3T-states.
- ↙ That is, Fetching of opcode is over after 3T states.
- ↙ Now decoding will happen
- ↙ But decoding will be happening inside MPLT.
- ⇒ 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 as 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~~ becomes infinite

$$V = IR$$

If $R = \infty$

$$V = I \times \infty$$

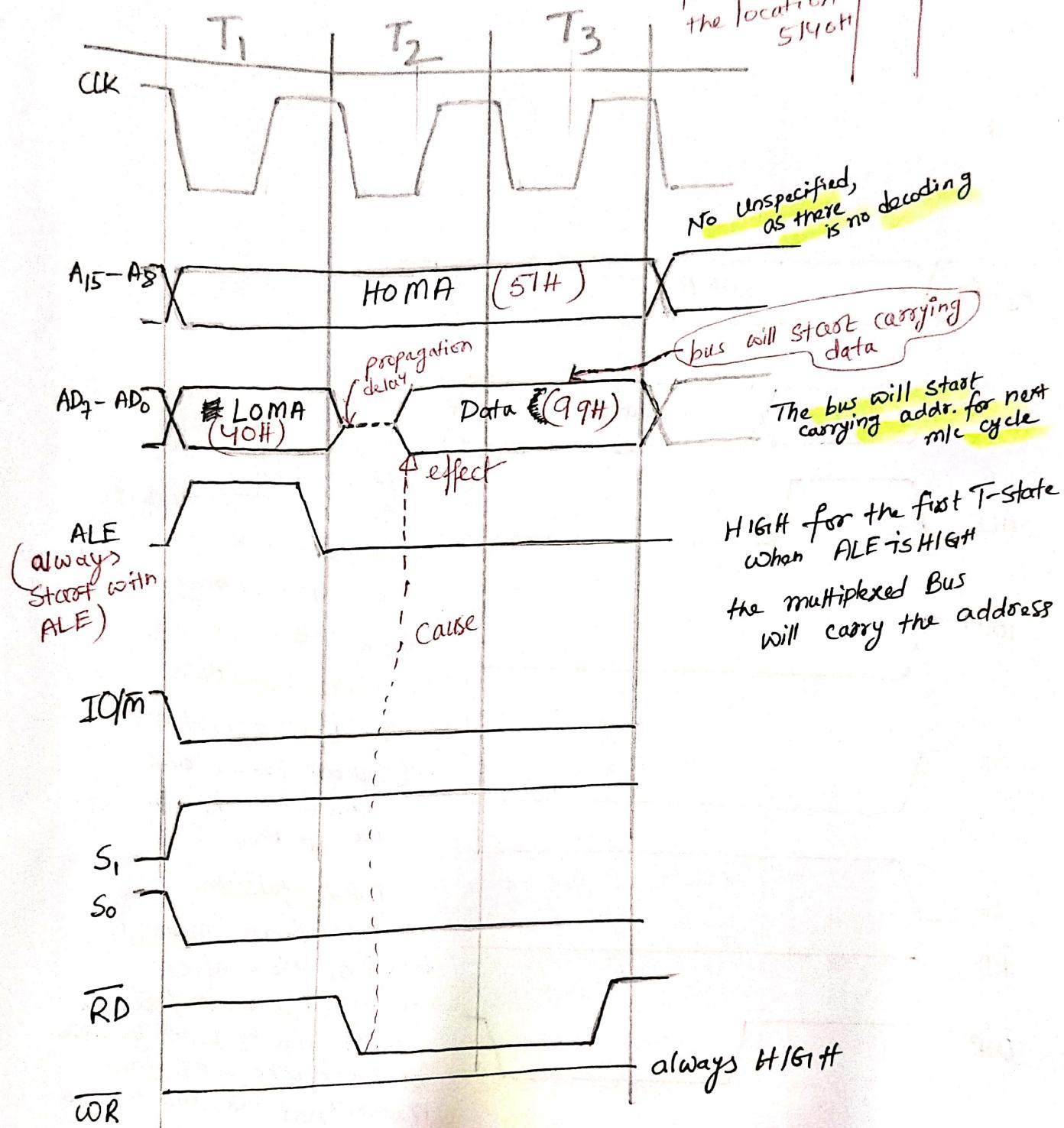
$$\begin{aligned} I &= \frac{V}{R} \\ &= \frac{V}{\infty} = 0 \end{aligned}$$

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

Timing Diagram for Memory Read

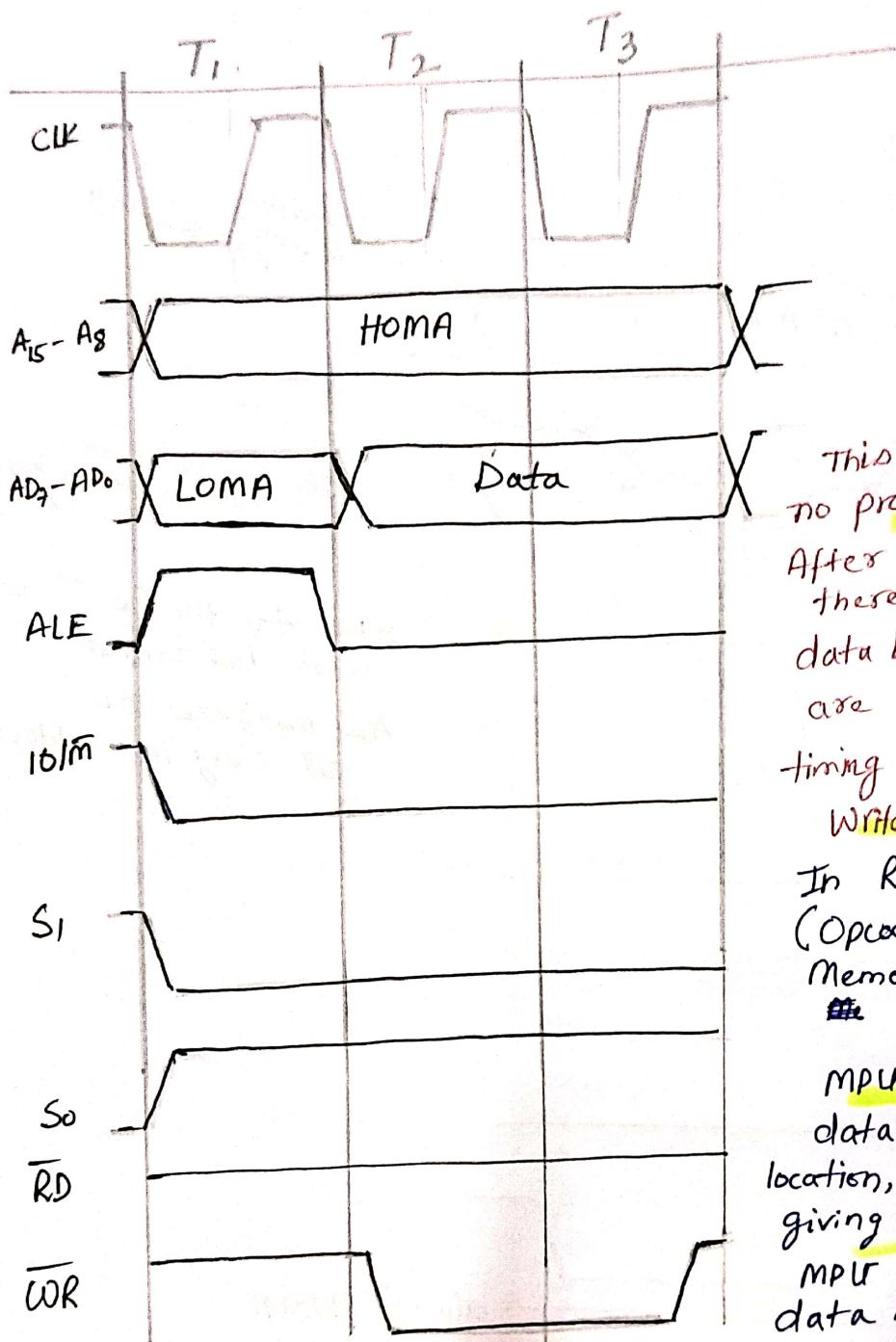
✓ Mpu wants to read the data from the location 5140H

(13)



✓ RD will not go low in the 1st T-state as in 2nd 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 and
Memory Read,)
~~#~~ To Read

MPLR gets the data from memory location, Now after giving RD control Signal MPLR 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
I_e the Write Operation,

propagation Delay. Now, when we do the write operation,

Address is given by MPUR
as well Data " " " " MPUR
as

MPU will give address, immediate MPU will give the data after that.

(*) Ex.

MVI A, 32H

Mem. location

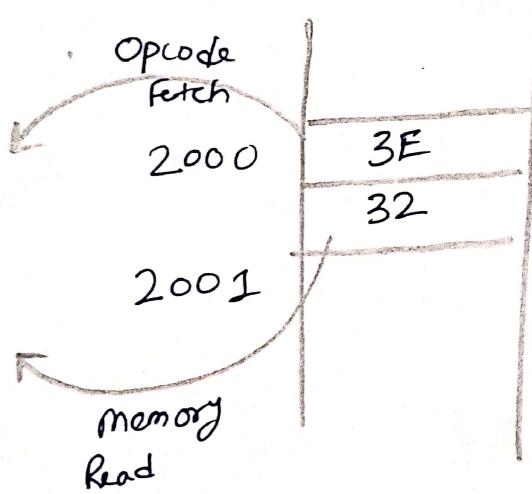
2000H

2001H

Machine Code	Instruction
0011 1110	MVI A, 32 ; Load
0011 0010	; Byk 32H ; in the Acc.

→ 3EH

→ 32H



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

