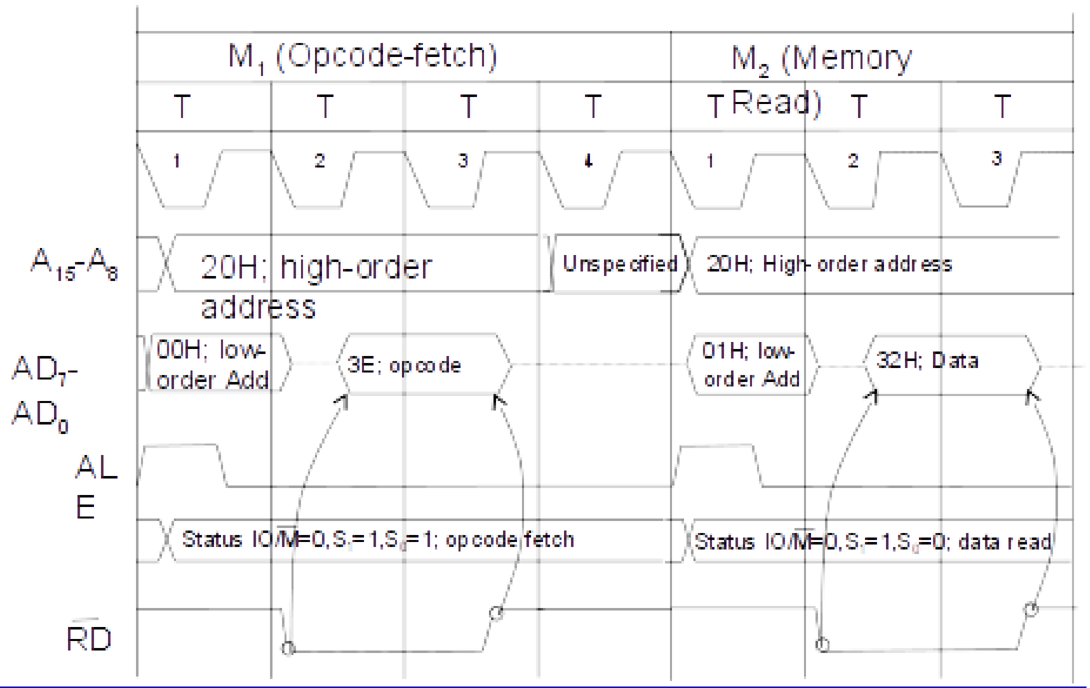
8085 Timing Diagram detailed explanation using example.

What exactly timing diagram tell us?? It tells us the internal behavior of microprocessor while instruction gets processed. Microprocessor generates certain signals from its various pins which could be plotted on oscilloscope and informs user about internal behavior of microprocessor, in response to the instruction getting currently processed.

Lets me talk about 8085 timing diagram using simple example

```
2000H   3EH   ;MVI A, 32H
2001H   32H
```



This timing diagram is for one typical instruction MVI A,32H, which is stored at memory location 2000H.What does it mean?

Every microprocessor instruction is made of opcode and operand.
Opcode is the equivalent Hex code for instruction mnemonic (ex: for opcode MVI A,8 bit, 3E is an equivalent Hex value). This Hex value is predefined and available in data sheet of microprocessor. Operand is the 8-bit (data) or 16-bit number (data/address) mentioned in instruction.

Now, when we write program for microprocessor, these instructions (opcode+operand) get stored in program memory. In our example, opcode 3E get stored at address 2000H and operand 32H get stored at address 2001H.

Every instruction is made of 1 or few machine cycles (predefined in data-sheet), which is made of T-states. T-states are smallest unit of execution in microprocessor, which is equivalent to 1 clock cycle.

For this instruction (ex: MVI A,32H), we need 2 machine cycles (M1 and M2). Machine cycle M1 is make of 4 T-states, where 3 T-states are used for opcode fetch from program memory and 1 T-state is used for its decoding. Similarly, Machine cycle M2 is made of 3 T-states which is needed for operand fetch from program memory. We need only 3 T-states in M2 since decoding is not needed for operand whereas opcode need extra 1 T-state for decoding process and converting mnemonic into equivalent Hex code. As I mentioned earlier, 1 T-state = 1clock cycle since T-state is smallest unit of operation in microprocessor.

Looking more deeper in working of instruction, MVI A,32H, very first requirement is to fetch instruction opcode from program memory so appropriate address need to be loaded in processor address buses. Before program memory address is loaded in processor Address Bus, ALE pin should go from LOW-->HIGH. This would actually multiplex the lower order address/data bus (AD7--AD0) so that it would act as Address Bus (A7--A0). Now we can load program memory address of instruction (00H address on A7--A0 and 20H address on A15--A0) on processor address bus.
NOTE: Lower order address bus AD7--AD0 can be used as address bus A7--A0 (when ALE goes LOW-->HIGH) or as data bus (when ALE goes from HIGH -->LOW)

Working of Timing diagram at T-state level in content to given diagram:

1) In 1st T-state, load lower order address (00H) is loaded in AD7--AD0 and higher order address (20H) is loaded in A15--A8. ALE is maintained HIGH at this instant. Read pin is also maintained HIGH since, no reading is yet to take place (address is just getting loaded right now).

2) During 1st T-state, after address is loaded in address bus, ALE goes from HIGH-->LOW, telling processor to change the state of AD7--AD0 bus to D7--D0 Data Bus after the loaded address (2000H) is identified in entire 64K memory space. This is done so that after memory location 2000H is identified in memory, data can be retrieved in same bus D7-D0.

3) Pins IO/M=0, S1=1 & S2=1 which informs user that "opcode fetch" operation (ex: MVI A,8 bit) is currently getting executed inside microprocessor

4) During 2nd T-state, searching of address 2000H takes place and when the address is found, multiplexing takes place which make AD7--AD0 act as Data Bus D7--D0. Read pin goes from HIGH-->LOW so that reading from required memory location (2000H) could take place in Data Bus. Reading process completes in 3rd T-state, thus, RD signal goes HIGH again.

5) In 4th T-state, both address buses remain in High Impedance state since decoding of opcode take place in Instruction Decoder.

6) By the end of 4th T-state, opcode 3E is fetched from memory location 2000H and decoded. Now fetching of operand 32H need to take place.

7) Again ALE signal goes from LOW-->HIGH, which multiplex lower order address/data bus AD7--AD0 and they start acting as Address Bus A7--A0. Now, address of operand 32H, which is stored at address 2001H, is loaded in Higher order (A15--A8) and Lower order Address Bus (A7--A0) and searching of address (2001H) begins in entire 64K memory range.

8) In 5th T-state, ALE again goes back from HIGH-->LOW, telling processor to use AD7--AD0 as Data Bus after memory location 2001H is identified.

9) Pins IO/M=0, S1=1 and S2=0, thus, informing user that currently, "operand read" process is taking place within microprocessor.

10) When 2001H memory location is identified in program memory, AD7--AD0 start acting as D7--D0 since ALE already goes from HIGH --> LOW. RD signal goes from HIGH-->LOW so that read operation can take place in 6th T-state . Once reading is completed, RD signal again goes back from LOW--> HIGH.

NOTE: Since this instruction is based on simple Data transfer instruction, no actual work is done by instruction i.e. no loading or storing of opcode/data takes place from memory/IO. If any instruction where memory access or I/O access is done, few additional machine cycles would be needed to complete that work.

What is a timing diagram?

A timing diagram in the field of embedded systems refers to a graphical representation of processes occurring with respect to time.

In other words, the representation of the changes and variations in the status of signals with respect to time is referred to as a timing diagram.

Take the below illustration as an example. It is a timing diagram for the instruction ****. You don't need to go into details now. Everything is explained in this post in the upcoming sections. For now, just understand what it represents.

*Image (Timing diagram of any instruction)

On the vertical axis, various signals are represented with possible values of high and low. Also, the values present on the address and data bus are also represented. Time is represented on the horizontal axis.

Timing diagrams give us a perspective and help us understand the process of execution of a particular instruction in detail. We get to see what's going on inside the microprocessor in every clock cycle. It is an extremely minute level of examining the working of the microprocessor. And as you'll learn in your embedded systems career, it is quite handy too. They help us visualize the whole process with respect to time.

You must have realized the importance of timing diagrams if you have read our post on 'Buses in 8085' and went through the section on demultiplexing address from the address/data bus.

Here are some more reasons why you must study timing diagrams.

Some more reasons to study timing diagrams

The 8085 is an elementary processor to understand the basics for a beginner. But as you proceed ahead in the field of embedded systems and study more about microprocessor designs and architecture, many new concepts are introduced. Some of them are listed below.

Instruction pipelining

SIMD instructions in ARM

Peripheral design protocols

Real-Time Operating Systems RTOS

You don't need to worry about these concepts at present, but the point of listing them here is that the study of timing diagrams now will be helpful for you in understanding these concepts in the future.

Common terminologies

T state
Processors work in synchronization with a periodic signal called ‘clock signal’. The part of any operation carried out during one time period of this clock signal is known as a T state. Sometimes, ‘T state’ is also used to refer to a time of one clock period. For studying the rest of the concepts, it becomes a basic unit of measuring time.

Instruction cycle
For the execution of any instructions, basically, two steps are followed – fetch and then execute. The time (or the number of ‘T states’) required to fetch and execute an instruction is called an instruction cycle.

Fetch cycle
When you write an assembly language program, every instruction of that program is converted to corresponding hex codes, which is then converted to binary and is stored somewhere in the memory. During the process of running an uploaded program, the processor first has to read from the memory the instruction that is to be executed. This process of reading the code for the instruction to be executed is called the fetch cycle. We can say that during this cycle, instruction to be executed is fetched from memory.

Execution cycle
After fetching the code for the instruction that is supposed to be executed, the next step is to execute that instruction. This process is referred to as an execution cycle.

Machine cycle
The time required for the microprocessor to access memory or an IO device either for a read operation or a write operation is called a machine cycle.

An instruction cycle consists of the fetch cycle and the execute cycle.

Control signals in 8085 timing diagram

There are some output signals in 8085 that tell us about the processes going on inside the microprocessor. These signals tell us about the type of machine cycle going on at any particular time. If we understand the function of these signals, then we can put them together, and their combined values will give us a complete picture of what stage the microprocessor is in. It’s like detective work. These signals are listed in the below table, along with their meanings.

Signals	Significance
IO/M	Tells us whether a particular operation (or a machine cycle) is in concern with the memory or an IO device. IO/M = 1 (Microprocessor is talking to an IO device) IO/M = 0 (Microprocessor is talking to the memory))

S1 and S0	Tells us if a read/write operation is going on or not. S1 and S0 = 00 (Halt) S1 and S0 = 01 (A write operation) S1 and S0 = 10 (A read operation) S1 and S0 = 11 (Either opcode fetch or interrupt acknowledge) Don’t panic if you dont understand some of the things completely. We will be discussing different machine cycles in the subsequent sections, and everything will be clarified.
ALE	ALE stands for Address Latch Enable. This output signal given by the microprocessor tells us whether the AD0-AD7 is carrying the address or is available for data transfer. ALE = 0 (AD0-AD7 is available for data transfer) ALE = 1 (AD0-AD7 is carrying the lower eight bits of the address)
RD	This is an active low signal and simply tells us whether it is a read operation when it is low.
WR	This is an active low signal and simply tells us whether it is a write operation when it is low.

Types of machine cycles

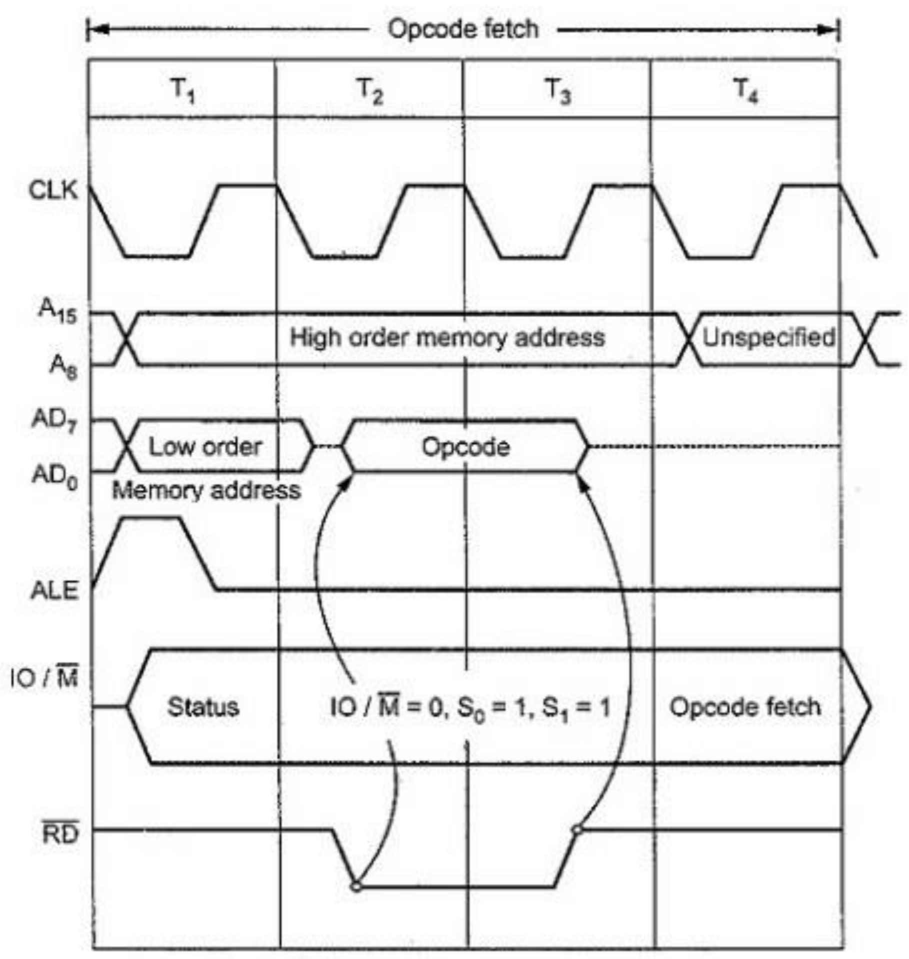
There are seven different types of machine cycles in 8085, which are listed below. We will discuss each one of them one by one.

Sr. No.	Machine cycle	IO/M	S1S0	Other control signals
1	Opcode fetch machine cycle	0	11	RD = 0
2	Memory read machine cycle / Operand Fetch machine cycle	0	10	RD = 0

3	Memory write machine cycle	0	01	WR = 0
4	IO read machine cycle	1	10	RD = 0
5	IO write machine cycle	1	01	WR = 0
6	Interrupt acknowledge machine cycle	1	11	INTA = 0
7	Bus Idle machine cycle	0	00	INTA = RD = WR = 1

Opcode fetch machine cycle

The opcode fetch machine cycle (OFMC) involves the fetching of the opcode of the instruction to be executed and the decoding process of that opcode. Usually, it consists of four T states. The timing diagram of a typical OFMC is explained below.



1st T state

During the first T state, the address of the location where the opcode is stored is loaded on the address bus. In 8085, this address is stored in a 16-bit register called the program counter. Higher eight bits of the address are loaded on A8-A15, and the lower eight bits of the address are loaded into AD0-AD7 for demultiplexing.

Also, the ALE signal becomes active in the first T state to indicate that the data on AD0-AD7 pins are the lower address bits.

IO/M signal becomes low at the beginning of the first T state to indicate that the opcode will be fetched from memory (reading from memory).

At the beginning of the first T state, signals S1 and S0 take the value 1 and 1 respectively to indicate that it is an opcode fetch machine cycle.

2nd T state

By the beginning of the 2nd T state or the end of 1st T state, the ALE signal goes low. By this time, 8085 expects that the lower address bits are latched, and AD0-AD7 is free to be used as a data bus.

At the beginning of the second T state, RD goes low, indicating that the read process has started. Meanwhile, higher address bits are present in A8-A15, and lower address bits are expected to be latched.

As RD goes low, the opcode (eight bits) is loaded into the data bus AD0-AD7.

3rd T state

The opcode loaded on the data bus is present there until the middle of the third T state.

During the third T state, RD goes up, indicating that the read operation is completed and ‘the opcode is fetched’ and placed in the instruction register.

The data on the data bus and the higher address bits on A8-A15 exist until the middle of this T state.

4th T state

During the fourth T state, the fetched opcode is decoded. There is nothing much to observe in the timing diagram during this process.

In case of some simple one-byte instructions like STC (set carry flag), execution is also completed during the fourth T state. One such instruction is MOV A, D.

During the fourth T state, after decoding the opcode, the microprocessor decides if it needs fifth and sixth T states, or should proceed to the next machine cycle.

PC is incremented by 1 here or in the sixth T state if the OFMC is extended upto sixth T state.

5th and 6th T state

In case of one-byte instructions that operate on 16-bit data and some other instructions, OFMC may extend up to six T states. During the fifth and sixth T states, execution of these instructions takes place. Since these instructions are simple, they get executed in the OFMC itself. Examples of such instructions are DCX, INX, PCHL, SPHL, CALL, RSTN and conditional RET.

Memory read machine cycle

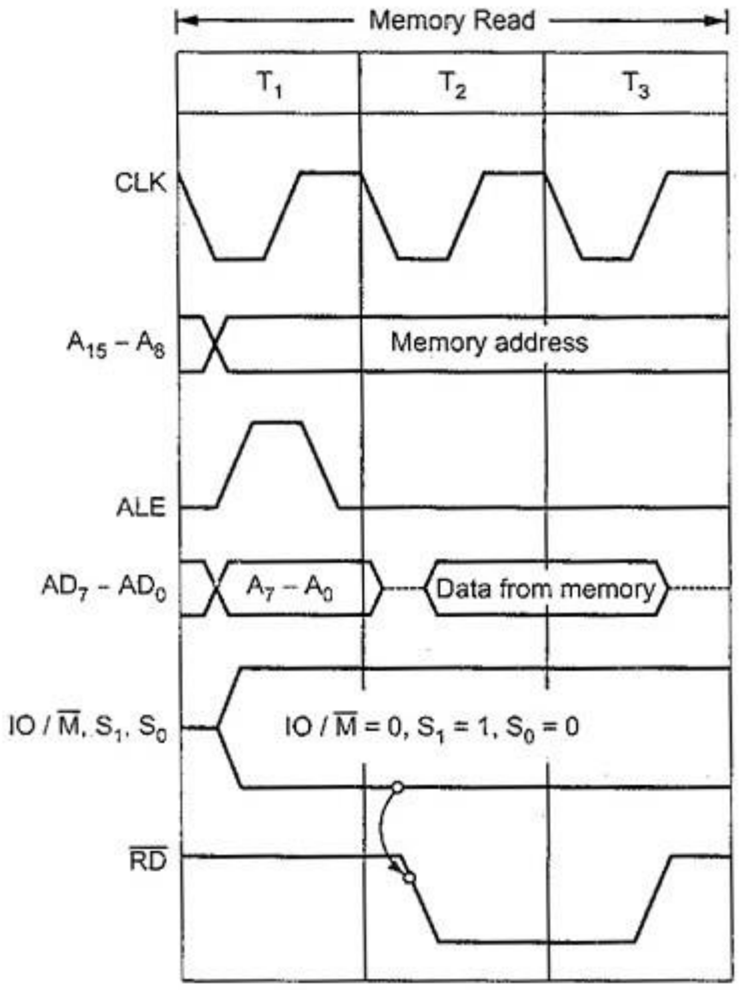
Contents from a memory location are read during the memory read machine cycle (MRMC). This cycle is also known as the operand fetch machine cycle. But there are cases when MRMC is not used for operand fetch but for reading data at given memory location. This machine cycle spans over three T states. Each of these T states is explained here along with a timing diagram. The first three T states are almost the same as the first three T states of Opcode Fetch Machine Cycle.

There are certain machine cycles where Program Counter is incremented and some, where it is not incremented. The simplest way to differentiate between the two is that

If the address is loaded into the address bus from the program counter, then PC in incremented at the end of that machine cycle otherwise PC is not incremented.

For reading the operands, memory read machine cycles are executed. In such machine cycles, PC is incremented as the address is loaded from the PC.

But during the execution of instruction MOV A, M the last machine cycle is the memory read machine cycle in which the address is loaded on the address bus from the HL register. Hence, the PC is not incremented in this case



1st T state

Higher address bits loaded into A8-A15.

Lower address bits loaded into AD0-AD7.

ALE signal goes high in the beginning to indicate that AD0-AD7 contains lower address bits.

IO/M goes low since it is a memory operation.

S1 and S0 become 1 and 0 respectively, indicating Memory Read Machine Cycle.

ALE goes low by the end of the first T state. Lower address bits are expected to be latched by this time.

2nd and 3rd T states

RD goes low, indicating the initiation of the read operation.

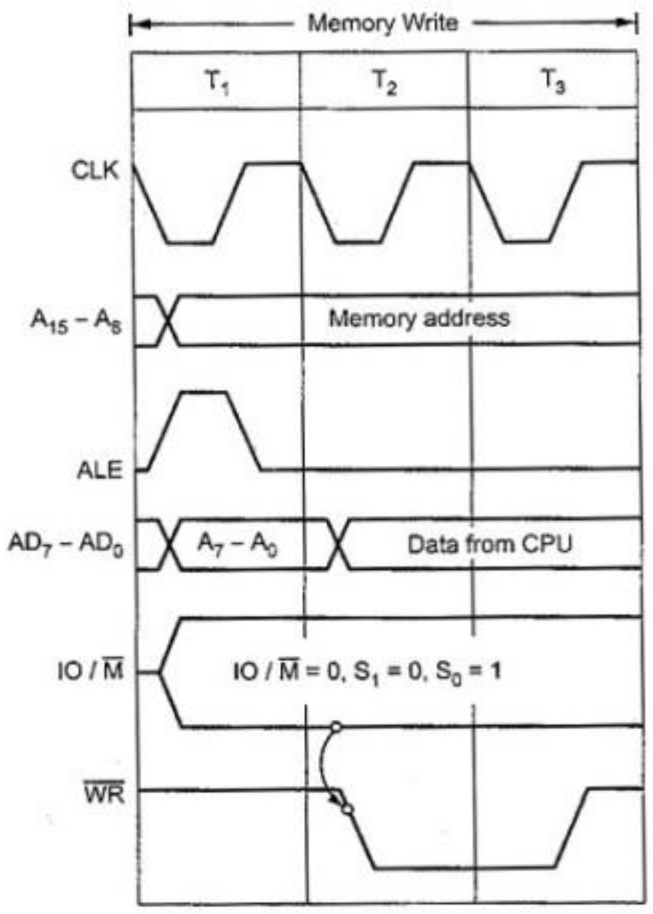
Data is read from the memory location and is loaded into the data bus AD0-AD7. The data is loaded into the data bus at the beginning of the 2nd T state and exists until the end of the third T state.

By the end of the third T state, RD goes high, indicating the end of the read operation.

PC is incremented by 1 (only in cases described in the note in the red box above).

Memory write machine cycle

Contents are written to a memory location/stack during a memory write machine cycle (MWMC). This machine cycle spans over three T states. Each of these T states are explained here along with the timing diagram. PC is not incremented in this machine cycle. This is very similar to MRMC, except a few differences.



1st T state

Higher address bits loaded into A8-A15.

Lower address bits loaded into AD0-AD7.

ALE signal goes high in the beginning to indicate that AD0-AD7 contains lower address bits.

IO/M goes low since it is a memory operation.

S1 and S0 become 0 and 1 respectively, indicating MWMC.

ALE goes low by the end of the first T state. Lower address bits are expected to be latched by this time.

2nd and 3rd T states

WR goes low, indicating the initiation of the write operation.

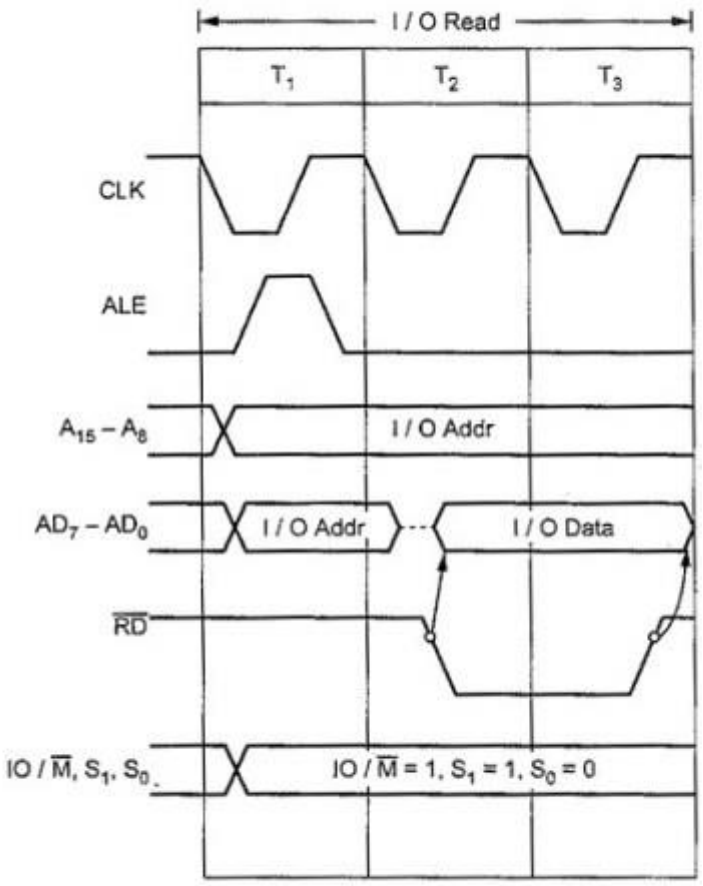
Data to be written is loaded on the data bus at the beginning of the second T state and exists until the end of the third T state when the data is transferred from the data bus to the memory location.

By the end of the third T state, WR goes high, indicating the end of the write operation. Thus, MWMC comes to an end.

IO read machine cycle

Contents from an IO device are read during IO read machine cycle (IORMC). This machine cycle spans three T states and is similar to MRMC except for the IO/M signal. The destination of this read operation is the accumulator. The Program Counter is not incremented here. IO/M goes high instead of going low, indicating that the microprocessor is talking to an IO device. Each of these T states are explained here along with a timing diagram.

There is an important point to be noted here. In 8085, IO devices have an 8-bit address. But we have AD0-AD7 and A8-A15 for addresses. So, whenever the microprocessor is exchanging data with and IO device (during IORMC and IOWMC), the same 8 bits are loaded into both the upper address bus and the lower address bus. During interfacing of an IO device, we can use any one of the upper and lower address buses according to our convenience.



1st T state

8-bit address is loaded into A8-A15.

The same 8-bit address is loaded into AD0-AD7.

ALE signal goes high in the beginning to indicate that AD0-AD7 contains address bits and not data bits.

IO/M goes high since the microprocessor is dealing with an IO device.

S1 and S0 become 1 and 0 respectively, indicating a ‘read’ machine cycle.

ALE goes low by the end of the first T state. Address bits in AD0-AD7 are expected to be latched by this time.

2nd and 3rd T states

RD goes low, indicating the initiation of the read operation.

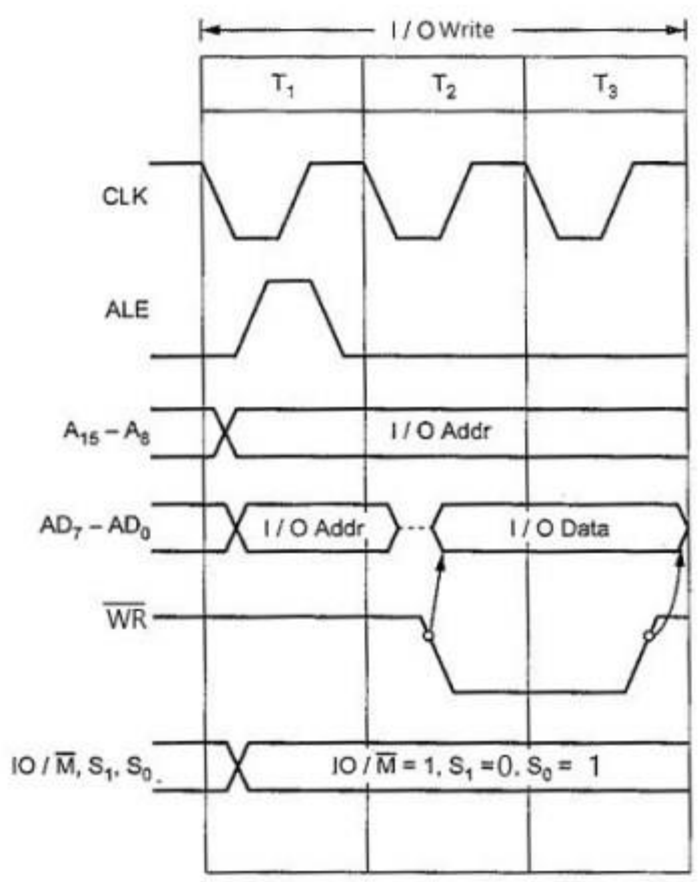
Data is read and is loaded on the data bus (AD0-D7) at the beginning of the second T state and exists until the end of the third T state.

In the third T state, the data is transferred from the data bus to the accumulator.

By the end of the third T state, RD goes high, indicating the end of the read operation.

IO write machine cycle

Contents are written to an IO device during IO write machine cycle (IOWMC). This machine cycle spans three T states and is similar to MWMC except for the IO/M signal. IO/M goes high instead of going low, indicating that the microprocessor is talking to an IO device. The contents of the accumulator are transferred to the data bus and written to an output device in this cycle. The T states are explained here along with a timing diagram for your reference.



1st T state

8-bit address is loaded into A8-A15.

The same 8-bit address is loaded into AD0-AD7.

ALE signal goes high in the beginning to indicate that AD0-AD7 contains address bits.

IO/M goes high since the microprocessor is dealing with an IO device.

S1 and S0 become 0 and 1 respectively, indicating a write machine cycle.

ALE goes low by the end of the first T state. Address bits in AD0-AD7 are expected to be latched by this time.

2nd and 3rd T states

WR goes low, indicating the initiation of the write operation.

Data to be written is loaded on the data bus at the beginning of the second T state and exists until the end of the third T state.

In the third T state, the data is transferred from the data bus to the IO device.

By the end of the third T state, WR goes high, indicating the end of the write operation.

Now, the next two machine cycles are a little different from the above machine cycles.

Interrupt acknowledge machine cycle

Before proceeding on to talking about the interrupt mechanism in 8085. An external IO device issues an interrupt signal (INTR), to tell the microprocessor that it wants a certain task done. When a microprocessor receives such a signal, it responds by making the INTA signal (interrupt acknowledgement signal) low for some time to tell the device that it has received the request and will take the necessary actions.

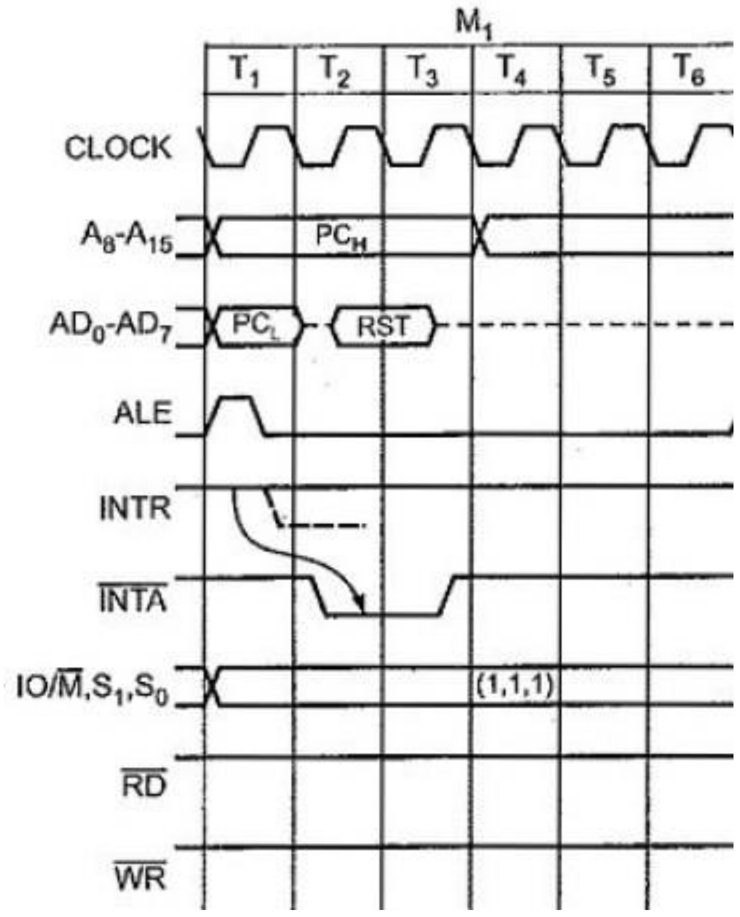
Responding to an external device with INTA signal after receiving an interrupt request – this whole process is carried out in a machine cycle called “Interrupt acknowledge machine cycle.”

To interrupt the 8085 microprocessor, we usually execute one of the two instructions – RST or CALL.

The RST instruction has only one interrupt acknowledge cycle of 6 T-states. Whereas the CALL instruction has three interrupt acknowledge cycles (First -> 6 T-states; Second and Third -> 3 T-states.) Why? Because RST is a one-byte instruction and CALL is a three-byte instruction.

Note that the addresses for the Interrupt Subroutine (ISR) will be provided by the instructions themselves. Hence, the PC won’t be incremented.

The timing diagram of the Interrupt Acknowledge Machine Cycle is given below.



Now, let’s discuss it step by step.

1st T state

The first T state of all the machine cycles involving data transfer is for the demultiplexing of AD0-AD7. The same is the case here. A8-A15 contains higher address bits. ALE signal goes high. AD0-AD7 contains address for that interval of time.

The difference here is that the INTA signal is high. RD and WR both are low. IO/M = 1 and S1S0 = 1

2nd and 3rd T state

These are also similar to the 2nd and 3rd T state of OFMC.

ALE goes low. AD0-AD7 is ready for data transfer and now contains the data until the middle of the third T state.

Remember, we discussed that some OFMC are of 6 T states? Similar to that, an IAMC is also of 6 T states.

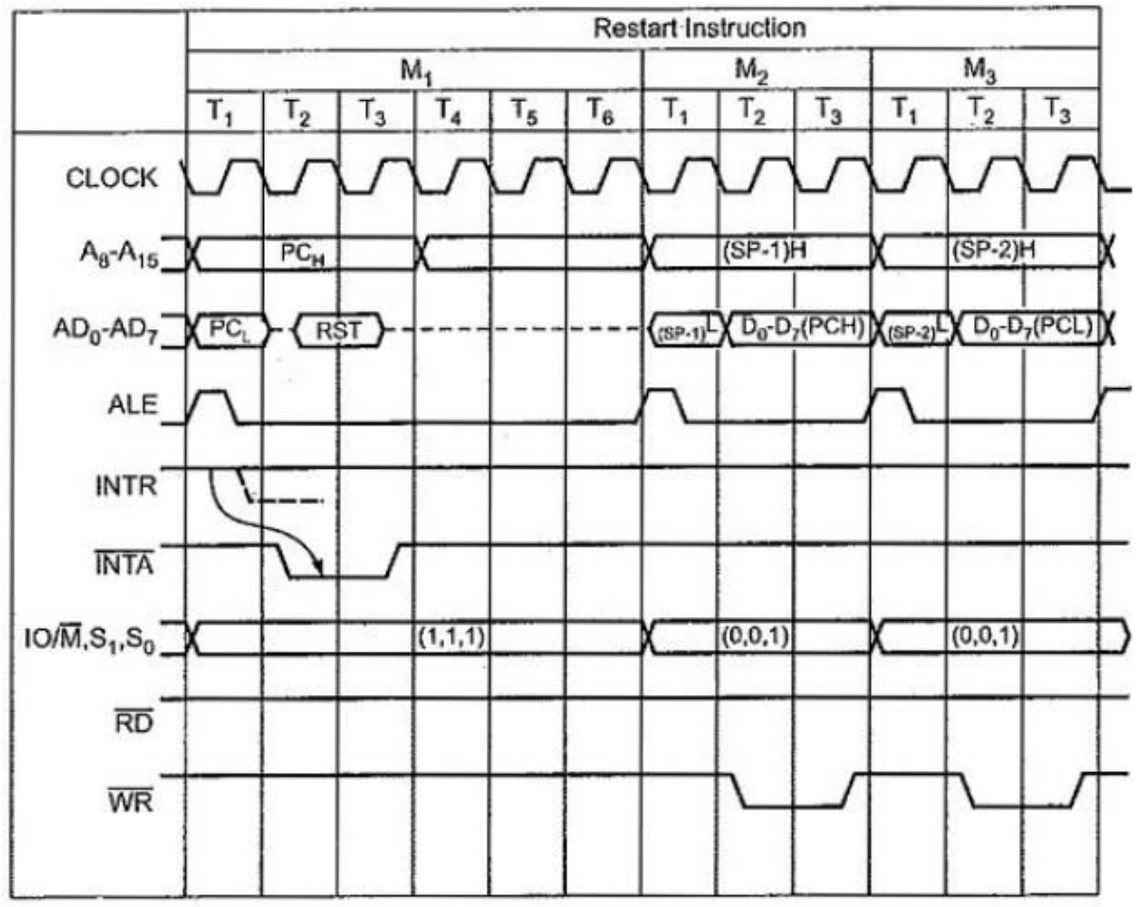
4th, 5th, and 6th T state

During the last three T states, instruction is decoded, and according to the decoded instructions, further machine cycles are executed to complete that instruction.

The decoded instruction is either CALL instruction or RST instruction. The execution of decoded instruction is completed in the subsequent machine cycles.

Let us have a brief look at the timing diagram of RST and CALL instructions.

Timing diagram of RST instruction



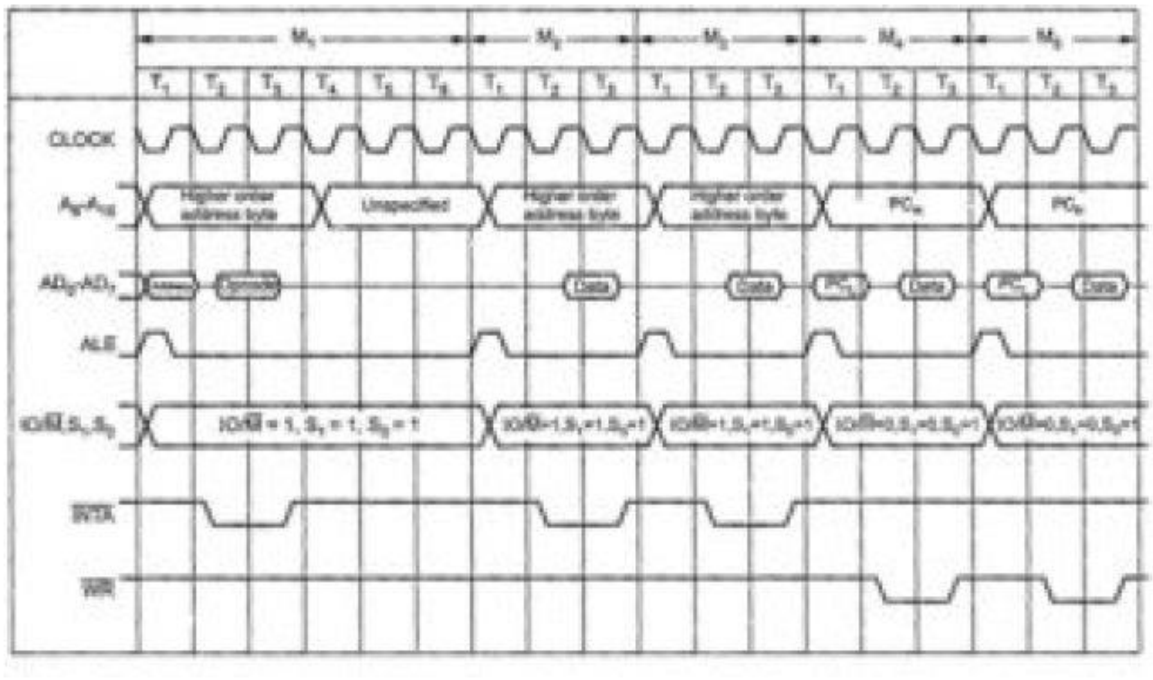
Notice that the instruction cycle of RST instruction consists of three machine cycles.

The first one is the Interrupt Acknowledge Machine Cycle explained above. At the end of the IAMC, the instruction is decoded.

Since it is RST instruction, two more machine cycles, both of them being MWMC, are executed.

In the second and third machine cycles, the contents of the program counter are written on the stack and after the execution of RST instruction, program execution jumps to the interrupt service routine.

Timing diagram of CALL instruction



The instruction cycle of CALL instruction consists of five machine cycles.

The first three are the Interrupt Acknowledge Machine Cycle explained above. At the end of the IAMC, the instruction is decoded.

In the second and third machine cycles, the device which caused the interruption gives the address of the location where the program location is supposed to jump after getting the interrupt signal.

The fourth and fifth machine cycles are MWMC. During these machine cycles, the microprocessor saves the contents of the program counter into stack since it will be executing the interrupt service routine and will have to return to that location again.

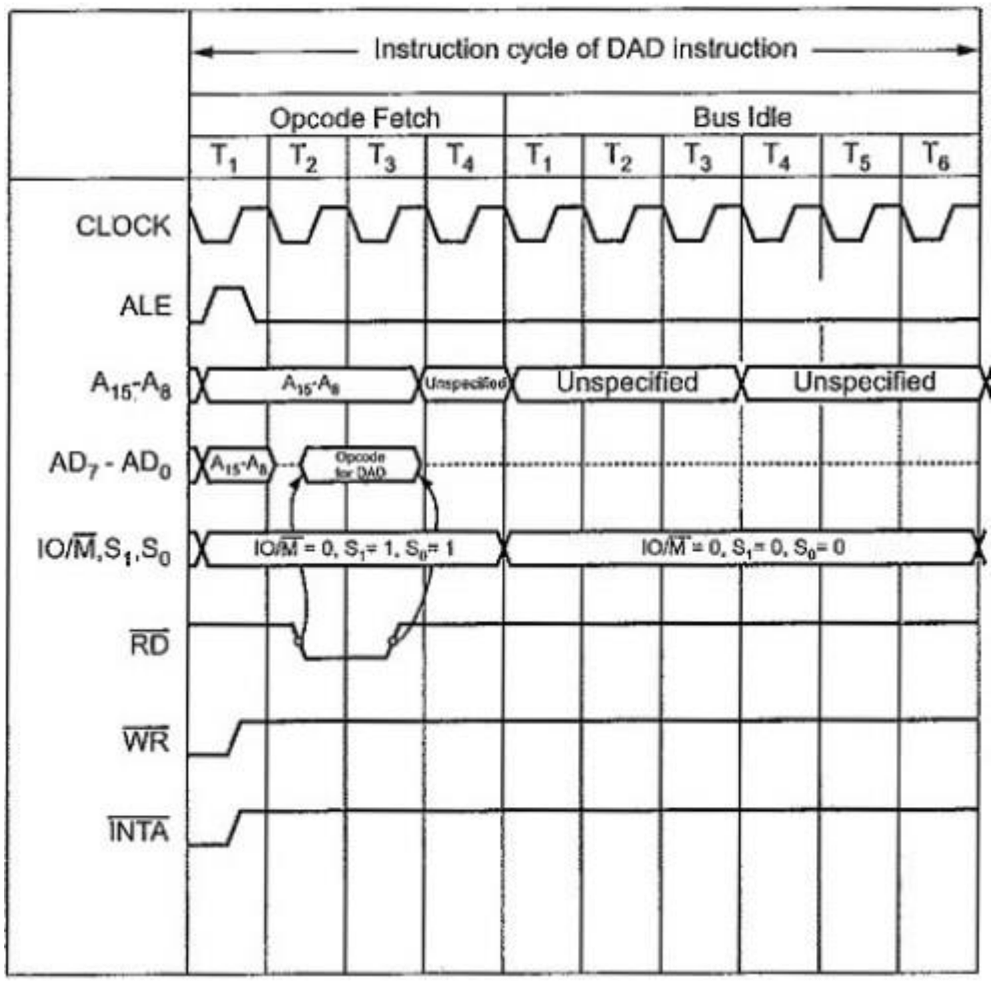
Bus idle machine cycle

There are some situations when no data transfer takes place. But it does not mean that the microprocessor is idle during that time as there might be some operations going on in the CPU. It is the data/address bus that is idle. Such a machine cycle is named a Bus Idle machine cycle (BIMC).

One such example where this situation occurs is the DAD instruction. So, let us have a look at the timing diagram of DAD instruction and understand more about BIMC.

Timing diagram of DAD instruction

The DAD instruction adds the 16-bit contents of a specified register pair with the 16-bit contents of the HL pair and stores the result in the HL pair. So, the first thing is to read the opcode for the DAD instruction, which is achieved in the OFMC. Since the operands (values to be added) are stored in the register pairs and microprocessor does not need to read any values from memory using the buses. It just needs to add the values and store the result. For this task, the microprocessor takes another six T states, which are a part of Bus Idle MC.


Instruction cycle of DAD instruction
Opcode Fetch
Bus Idle
T1 T2 T3 T4 T1 T2 T3 T4 T5 T6
CLOCK
ALE
A15-A8
AD7 - AD0
IO/M̄,S1,S0
R̄D̄
W̄R̄
ĪN̄T̄Ā
A15-A8
Unspecified
Opcode for DAD
IO/M̄ = 0, S1 = 1, S0 = 1
IO/M̄ = 0, S1 = 0, S0 = 0

1st Machine cycle – Opcode Fetch MC

The first machine cycle is the opcode fetch machine cycle about which we discussed previously. It takes 4 T states to get executed.

2nd Machine cycle – Bus Idle MC

The second machine cycle of DAD instruction is BIMC.

Notice that RD, WR, and INTA are inactive during BIMC.

IO/M = S1 = S0 = 0 signifying that it is a Bus Idle Machine Cycle.

Since no data transfer takes place, data present in the address bus and data bus is unspecified.

ALE signal is low for the entire six T states.

The addition operation is carried out in the CPU and is not represented here on the timing diagram.

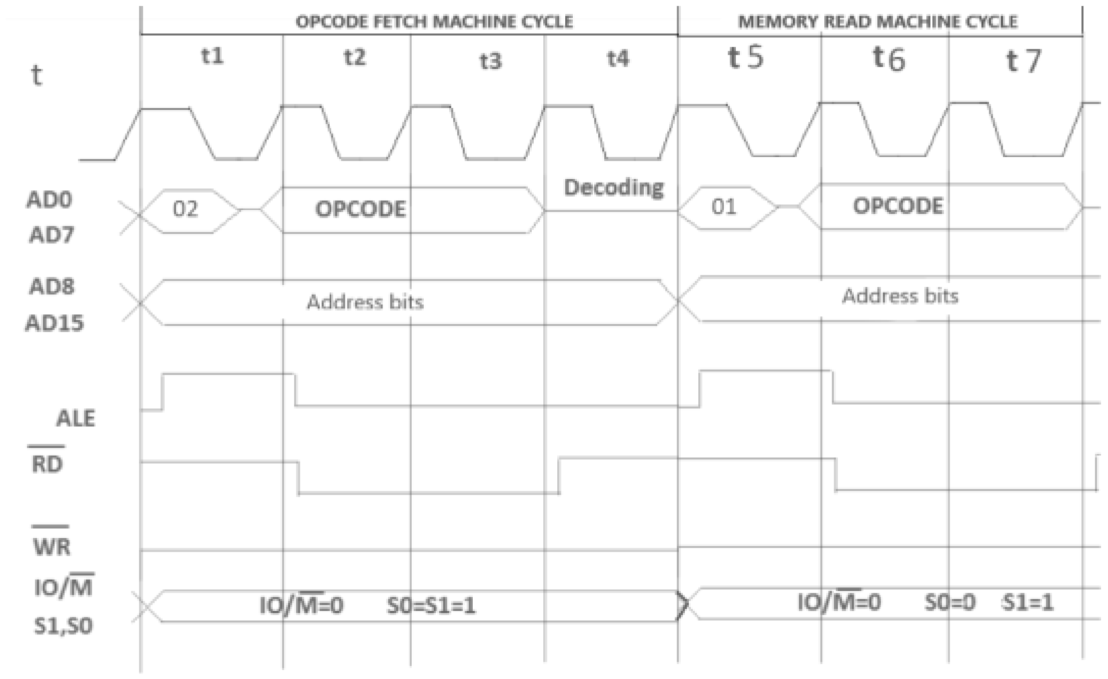
Timing diagrams – Examples

Let us look at the timing diagram of some instructions and revise and improve our understanding of whatever we have learned so far.

MVI Instruction

MVI instruction stores the immediately provided 8-bit data into the specified location, which can be either a register or a memory location. Suppose we have instruction

MVI A, 45H



This is a 2-byte instruction. One byte for the opcode, second byte for the operand.

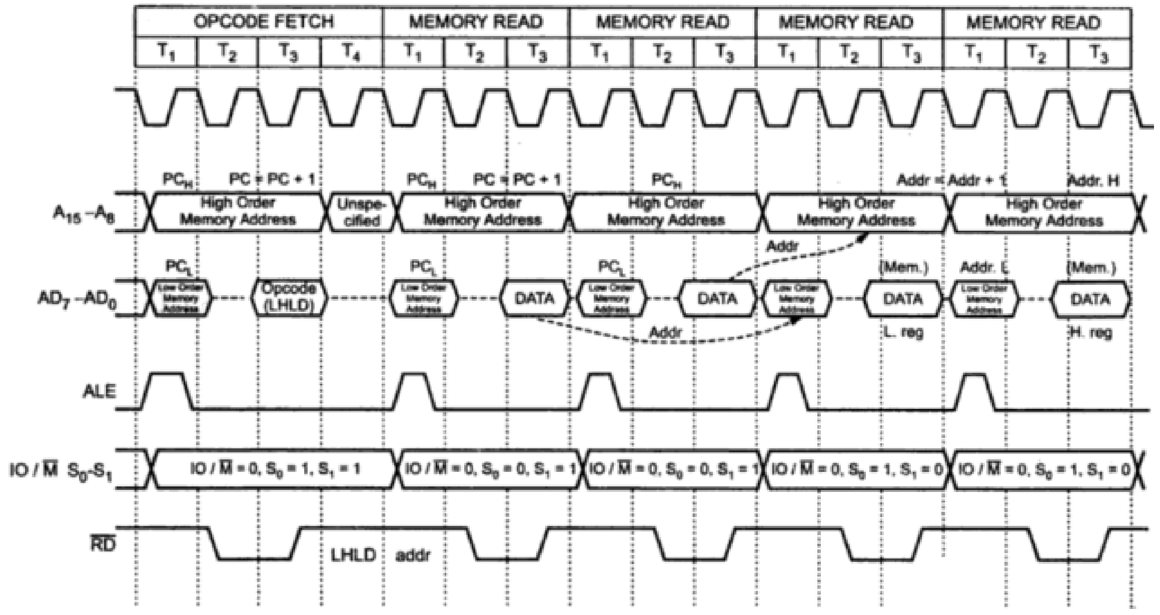
So, the first machine cycle will be OFMC, during which the microprocessor will read the opcode. During the second machine cycle, the microprocessor will read the operand, which is the 8-bit number 45H. So, the second machine cycle will be MRMC. You can refer to the timing diagram of the MVI instruction above.

LHLD Instruction

LHLD instruction does the following two tasks:

Copies the contents of the memory location with the address specified by the 16-bit operand (in this case, 2040H) to register H.

Copies the contents of the memory location next to the address specified by the 16-bit operand (in this case, 2041H) to the register L.



The first step of execution of any instruction is fetching the opcode. Here also, OFMC is the first machine cycle. Now, LHLD is a 3-byte instruction. The first byte is the opcode, the second byte is the lower bit of the address (40H in this case), and the third byte is the upper address bit (20H).

The first machine cycle is OFMC. The microprocessor also needs to read the 16-bit operand. So, OFMC is followed by two MRMC for reading the operand. Refer to the timing diagram below.

Now, the microprocessor has the 16-bit address. It reads the content of the address in an MRMC, stores it in register L, and then increments that address by 1. Then, it reads the contents of that incremented address in another MRMC and stores it in register H.

Thus, the number of machine cycles required by LHLD instruction:

Sr No	Machine cycle	Purpose	No of T states
1	OFMC	To read the opcode of the instruction	4
2	MRMC (twice)	Two MRMC to read the 16-bit operand, byte by byte.	3×2

3	MRMC (twice)	Another two MRMC to read the content of the address and the contents of the location next to it.	3×2
Total number of T states			16 T states

Opcode fetch machine cycle (to read the opcode)

Two memory read machine cycles to read the 16-bit operand, byte by byte.

Another two memory read machine cycles to read the content of the address and the contents of the location next to it.

How to calculate the number of T states in a given instruction?

In this section, we discuss a systematic way of determining the machine cycles and operations taking place during the execution of a given instruction when we are provided with an instruction and are informed about what that particular instruction does.

We will proceed in steps.

Step 1

The first step is to determine the byte size of the instruction. On the basis of size, there are 3 different types of instruction:

Sr No	Type of Instruction	Description
1	One-byte instruction	There are some instructions without an operand. Such instructions consist of just one byte which corresponds to an opcode. E.g. MOV A, B; DAD
2	Two-byte instruction	Some instructions are of two bytes. First byte for the opcode. And the second byte is an 8-bit operand. E.g. MVI A, 23H
3	Three-byte instruction	Three-byte instructions have the opcode as first byte and two bytes of a 16-bit operand, divided into two parts of 8 bits each. E.g. LDA 2034H

Step 2

After determining the size of the instruction, we can determine the machine cycles required to read it. The first machine cycle is always the OFMC to fetch the opcode. Now, depending on the opcode, the microprocessor knows the number of bytes of operands required and executes the MRMC for the required number of times (either once or twice).

Keep in mind that an OFMC is usually of 4 T states but for some cases, it extends up to 6 T states. Think about and consider any such exceptions before proceeding.

Step 3

After reading the instruction and operand completely, the further process depends on what that instruction does. The instruction may do the following –

Sr No	Task	Machine Cycle	No of T states
1	Read data from memory or IO	MRMC or IORMC	3
2	Read data from memory or IO	MWMC or IOWMC	3
3	Carry out some arithmetic process inside CPU without any data transfer	BIMC	6
4	Respond to an interrupt	IAMC	6

According to the task that an instruction performs, we can allocate the respective machine cycles and count the number of T states.

Now, let us take STA instruction as an example and perform the following steps.

Counting the number of T states in STA – an example

Step 1

The format of STA instruction is ‘ STA 16 bit address’. So, we know that it is a three-byte instruction – one byte for opcode and two bytes for the 16-bit operand.

Step 2

The first machine cycle will be the opcode fetch machine cycle. Besides that, the microprocessor will have to read the 16-bit operand. So, two MRMC are required.

Step 3

STA instruction stores the content of the accumulator to the memory location with the address provided as the 16-bit operand of the instruction. Suppose our instruction is STA 2050H. To execute this instruction, the microprocessor will copy the contents of the accumulator to the memory location with address 2050H.

So, we can conclude that a MWMC is required. This completes the execution of the instruction as well as our analysis. Let’s tabulate the observations.

Suppose the instruction ‘STA 2050H’ is to be executed, which is stored at 4045H.

This instruction is converted into hex code and stored in the memory. The microprocessor reads it from there during execution. The manner in which this instruction is stored in the memory in the form of hex codes is depicted in the table below.

Memory Address	Value stored at that address	Description
4045H	32H	Opcode of the instruction
4046H	50H	Lower byte of the operand
4047H	20H	Higher byte of the operand

Our conclusions regarding the instruction cycle of STA instruction

Sr No	Machine Cycle	Purpose	Number of T states
1	OFMC	To fetch the opcode and decode it	4
2	MRMC	To read the lower 8 bit of the 16-bit operand (50H)	3
3	MRMC	To read the higher 8 bit of the 16-bit operand (20H)	3
4	MWMC	To write the contents of the accumulator to the desired memory location	3
Total T states			13 T states

We conclude our post here. If you have any doubts or want to discuss more on this topic, just leave a comment below and we will get back to you.

https://technobyte.org/timing-diagrams-machine-cycles-8085/