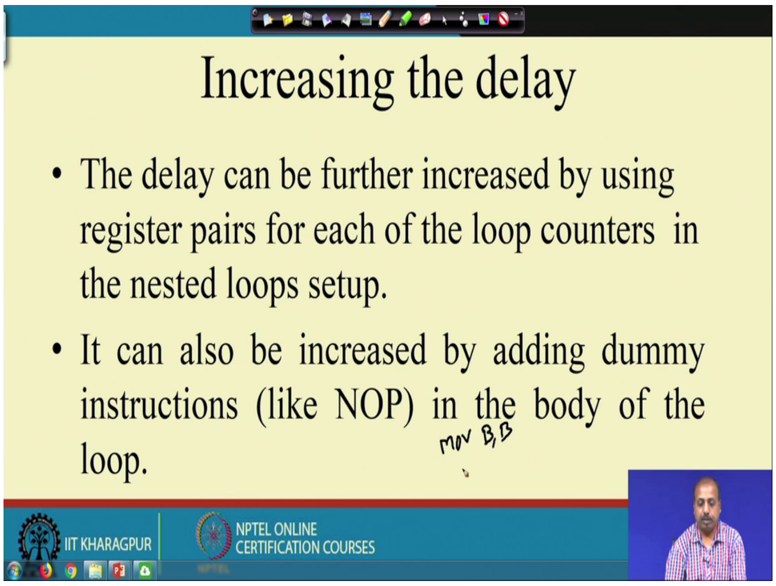
Digital Circuits

Prof. Santanu Chattopadhyay
Department of Electronics and Electrical Communication Engineering
Indian Institute Of Technology, Kharagpur

Lecture- 55
8085 Microprocessor
(Contd.)

So, up to increase the delay, so delay can we increase further by putting some further instructions.

(Refer Slide Time: 00:21)



Increasing the delay

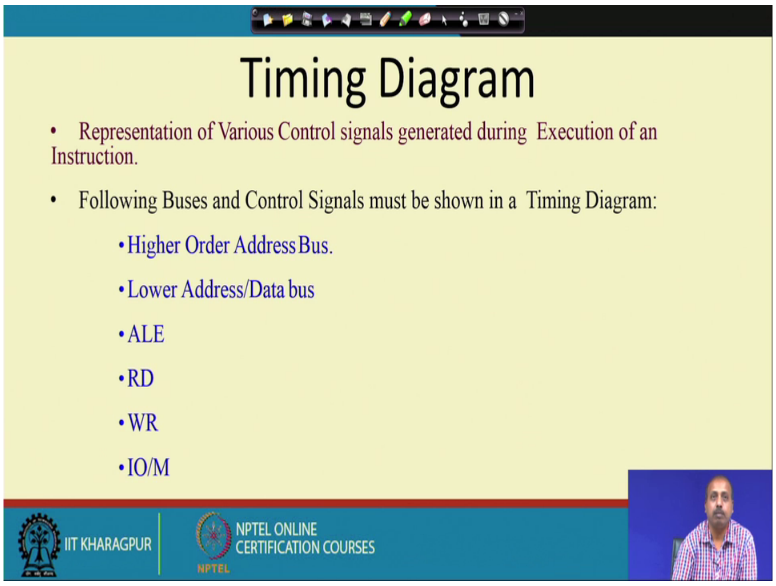
- The delay can be further increased by using register pairs for each of the loop counters in the nested loops setup.
- It can also be increased by adding dummy instructions (like NOP) in the body of the loop.

MOV B,B

So, one possibility is that we can increase the count value of this register pair that we have putting into. So, that way delay will increase significantly, sometimes what is required is that we can you do not need that much increase ok. So, it may be just very small amount of delay has to be inserted. So, far that purpose we can put some instruction with does not have any effect, so like this NOP type of instruction. So, another so we can we can think about the different NOP type of instruction like you can say I put on instruction like MOV B comma B. So, effectively does not have any operation that it is doing.

So, we does not do any do any complication are any movement as such. So, that way the it is similar to the NOP, but anyway. So, this NOP is much more understandable, so that way many a many a time will put this NOP instructions inside the delay loops to increase the delay.

(Refer Slide Time: 01:24)



Timing Diagram

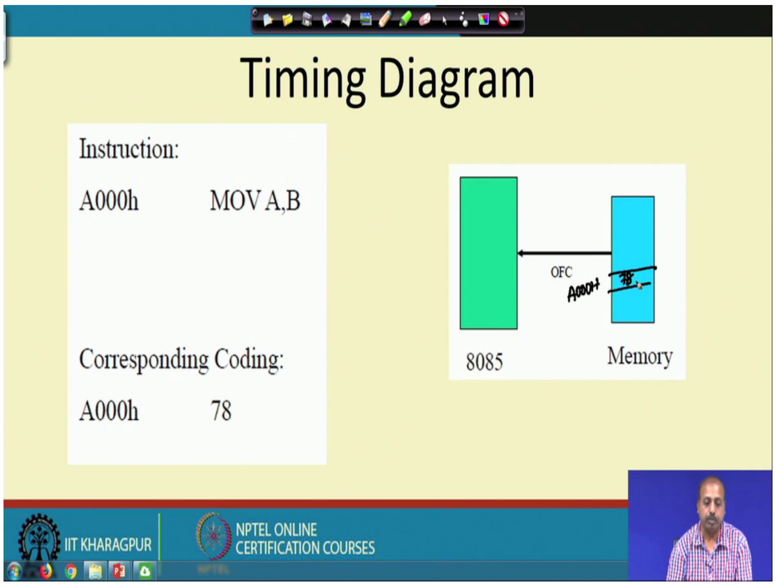
- Representation of Various Control signals generated during Execution of an Instruction.
- Following Buses and Control Signals must be shown in a Timing Diagram:
  - Higher Order Address Bus.
  - Lower Address/Data bus
  - ALE
  - RD
  - WR
  - IO/M

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL

Next will be looking into something, something called the timing diagram; So, this timing diagram this actually tells that when are the different control signals generated as we are proceeding through the operation of instructions. So, following buses and control signals must be shown in a timing diagram for if you the drawing the timing diagram of 8085. So, we must show these line higher order address bus, lower order address bus and data bus then the ALE signal the read signal, right signal and IOM bar line.

So, these are the minimum signals that you must show when you are trying to when you are after draw the timing diagram of any 8085 instruction, because that will tell us what a how are the how are the things going on inside the processor for executing the instruction.

(Refer Slide Time: 02:17)



Timing Diagram

Instruction:

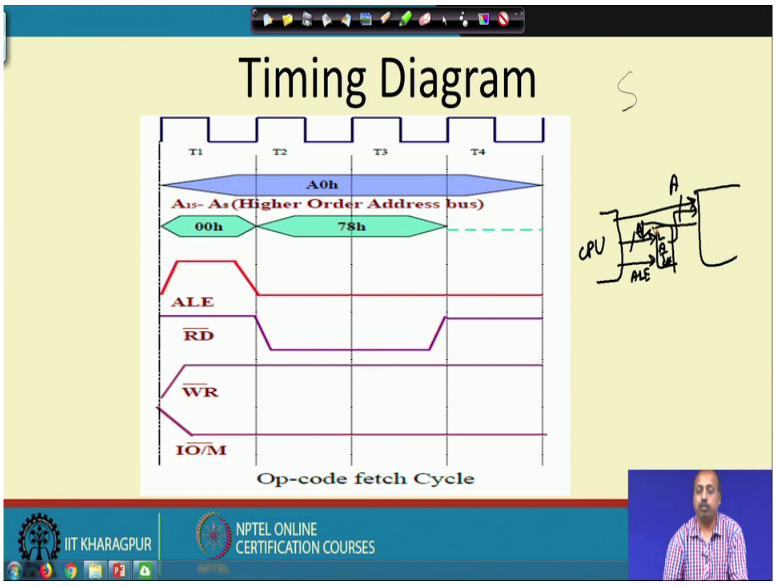
A000h MOV A,B

Corresponding Coding:

A000h 78

So, let us consider this instruction MOV A comma B ok, so here the opcode is 78 so, this opcode how as a how this will be done. So, this is we assume that this instruction is that memory location A 000 hex and that also so if you can say that at if this is the location A 000 hex, then here the 78 available. Now, how is it going to be fetch to the process and going to be executed, so that will see.

(Refer Slide Time: 03:06)



Timing Diagram

T1
T2
T3
T4
A0h
A15- A8 (Higher Order Address bus)
00h
78h
ALE
RD
WR
IO/M
Op-code fetch Cycle
S
A
CPU
ALE
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

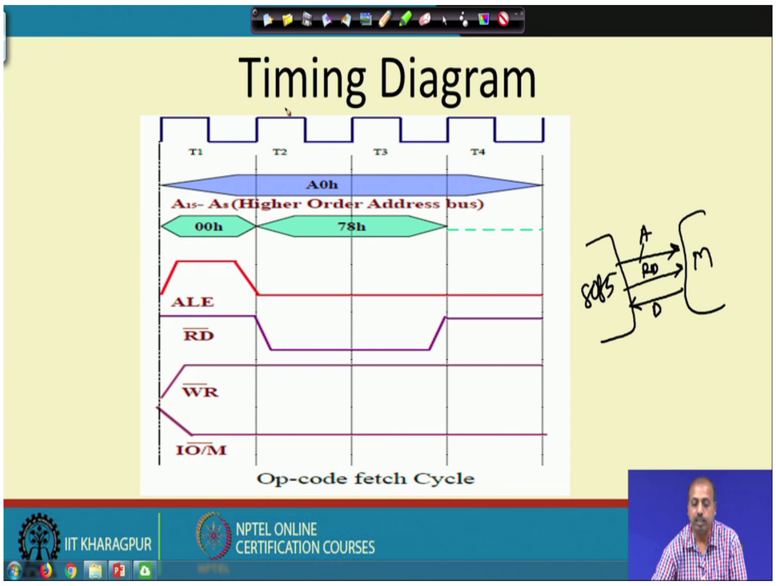
Now, so this is the timing diagram for a 80 for this execution of this instruction. So, in the first clock as this requires only four clock cycles which is which forms a part of the opcode fetch cycle. So, in the so this is the address bus content. So, this A A 15 to A 8 higher order address bus. It contents the higher order part of the address that is A 0 and the next part the so this the lower order address part A 0 to A 7. So, they are contained in this the lower order address bus cum data bus.

So, that is 00 hex and when this lower order data is lower order address is available on the lower order address bus this ALE signal is activated ok. So, this ALE signal is activated as we know that there is a latch outside the processor and it is expect that this if this is your if this is the processor, if this is the processor and then there is a latch there is a latch here. So, this lower order address bus comes here and the ALE signal goes there. So, ALE signal is used to latch this lower order address bus and then if this is the memory and this lower order address bus and the higher order address bus together.

So, they form the address for the memory and the data comes to the data bus directly like this. So, you have got this ALE in the signal activated here, and the ALE signal is activated when the address lower order address data bus it contains the lower order address part this. So, the process, so now to the memory the address has been given properly, now since this is a memory read operations.

So, fetch is a memory read operation, so this read bar signal is activated at the beginning of T 2 ok. So, during T 1 ALE was during T 2 this read bar signal is active. So, this read bar signal go slow telling that it is asking for the instruction or the data from the memory and getting this read bar signal.

(Refer Slide Time: 05:30)


Timing Diagram

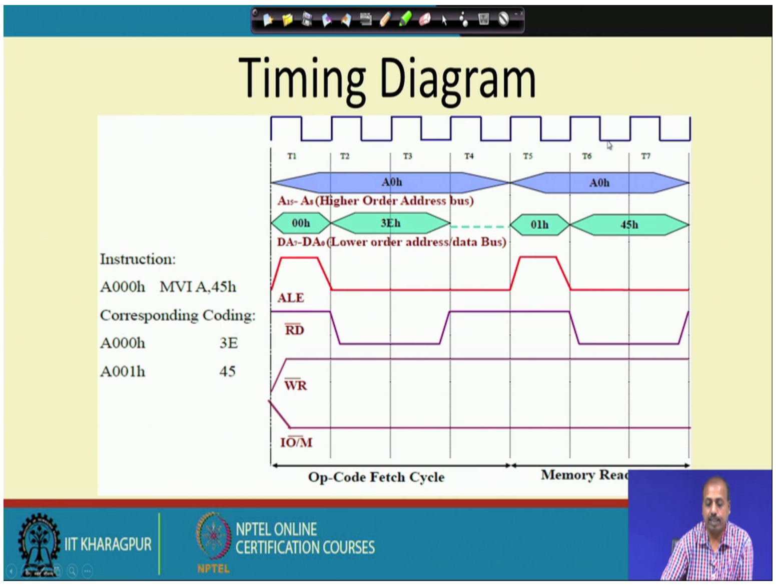
T1
T2
T3
T4
A0h
A15- A8 (Higher Order Address bus)
00h
78h
ALE
RD
WR
IO/M
8085
A
RD
D
M
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

Op-code fetch Cycle

So, again if this is the CPU this is the 8085 CPU and this is the memory so far we have been able to set the address bus and we have been able to give the read bar signal. So, when you give this read signal then the memory will put the data onto the data bus which happens to be the lower order address bus itself. So, the 78 value which is the content of the location A 000 hex is put on to the data bus and then this processor will read it from the data bus.

Now, the right bar signals is this is not a memory right operations. So, this is get continually high throughout the operation and is IOM bar signal. So, this is made low because this is the memory operation and not an IO operation. So, that way this opcode fetch a part a opcode fetch cycle is executed for the MBI. So, if you draw this timing diagram this tells us explicitly how different control signals are being activated by the processor to get the operation done.

(Refer Slide Time: 06:34)



Let us take a slightly more complex example MBI A comma 45 hex. So, here you can understand that in the first machine cycle. So, is the fetch machine cycle and in the fetch machine cycle. So, it will fetch the instruction and since it can fetch only 8 bit of data at a time. So, if it will in the first cycle it will get the opcode parts this MBI A 45 hex this instruction. So, this instruction has got a coding of 3E for the MBI A part and the next part is 45 hex.

So, if the instruction is again at location A 000 hex. So, it will take two bytes to hold the instruction. So, location A 000 hex we have three and A 001 will have 45. Now the way the operation is done is similar like it is in the opcode fetch cycle it is similar to the previous case. So, this higher order address bus contains A 0 hex lower order address bus if the first clock cycle contains 00 the ALE signal is activated and the then.

So, ALE signal is activated then the 00 gets latched on to the memory on the address bus of the memory. So, we have got the address memory address bus contains A 0000 hex total 16 bit address. Now, read bar signal is given so the memory will put the content of that location 3E on to the data bus.

Now, the process also the read the right bar signal is high and IOM bar signal is low that is as previously. Now, getting this 3E the processor understand that the controller part to the instruction decoder and the control unit. So, it will understand that this is a MBI instruction MBI. So, MBI A instruction, so for that it needs to get the immediate value.

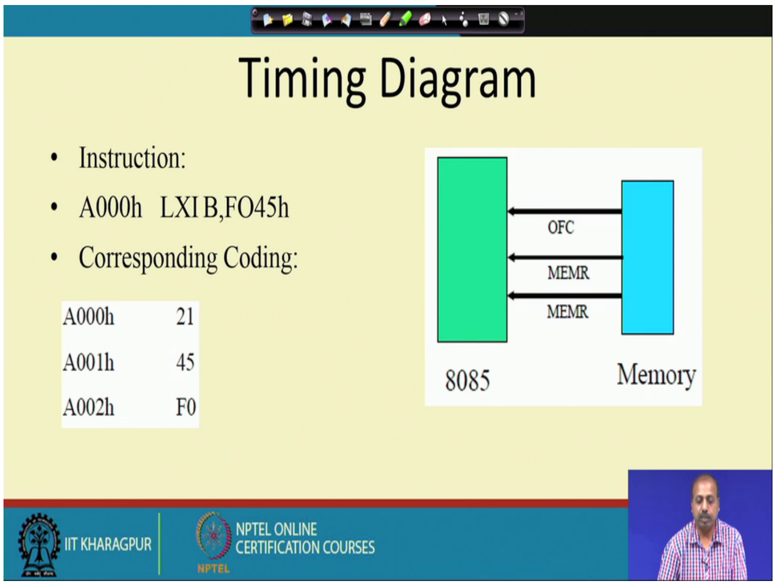
So, it has to read the memory once more. So, that is the it has to go into a memory read cycle in the memory read cycle it is similar to the fetch, but one is get less than the fetch. So, it is putting this higher order address bus higher order address on the higher order address bus, lower order address on to the lower order address bus and it is giving the ALE signal.

Now, so this address parts gets latch to the address bus of the memory and then it gives the read signal. So, in response to that the memory will put this 45 hex on to the database and this 45 hex comes to the processor ok. So, this way this 45 hex, so I get this value into the memory into the processor and the processor will now put this 45 hex into the A register that is done immediately after that 45 hex value has been obtain. So, it is put immediately into the A register, so it does not require any more clock cycle.

So, one interesting thing to note is that this fetch and at this memory read. So, both of both this memory both this cycles are similar. So, both of them are doing some sort of memory read operation, but in case opcode fetch cycle it takes one 1 T state more than the memory read. So, this is most probably due to this decoding operation that the processor has to do after getting the opcode part, though it is not documented in the manual.

So, it is not very difficult to understand that the decoding take some time and this extra clock cycle that is put in the opcode fetch cycle. So, that is most probably for doing that decode operation careful. So, that is the timing diagram of this MBI A type of instruction.

(Refer Slide Time: 10:14)



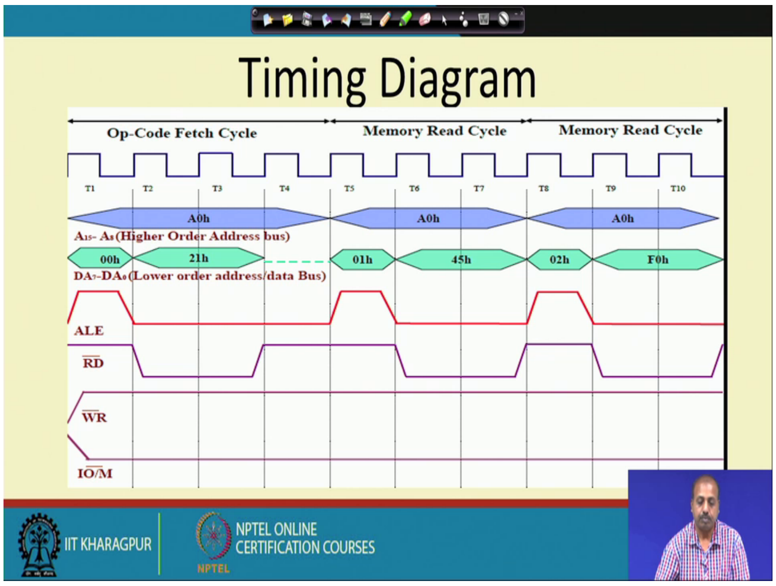
Timing Diagram

- Instruction:
- A000h LXI B,FO45h
- Corresponding Coding:

A000h	21
A001h	45
A002h	F0

Next we look into another instruction which is LXI instruction. So, LXI B comma F045 hex. So, that is that 3 byte instruction has we know this LXI B. So, the its opcode is 21 hex then this F045, so 45 will be put at location A 001 hex and F0 will be put at A 002 hex. So, this is the 3 byte coding of this instruction now how is it going to be executed in terms of timing diagram, so we would like to see.

(Refer Slide Time: 10:50)



Timing Diagram

So, this is the whole operation, so we start at. So, is first this higher order address bus contains A 0 lower order address bus contains 00 ALE signal is given, so and this read bar signal is given, so these memory will get the content of this location that is 21 hex and putting on to the data bus.

So, here put it on to the data bus when the read bar signal is low. So, this I have the right bar and IOM bar both are I have read the right bar is high and IOMR is low, because is not a memory right operation and this is the memory operation, so that way. So, after that so this T 4 is for the decoding part, so in the T 5 clock cycle. So, it will understand the processor understand that now, I have to these are these are code of LXI this 21 hex is the code of LXI it is the code of LXI B instruction; so, it as to get two more bytes and accordingly initialize the B and C registers.

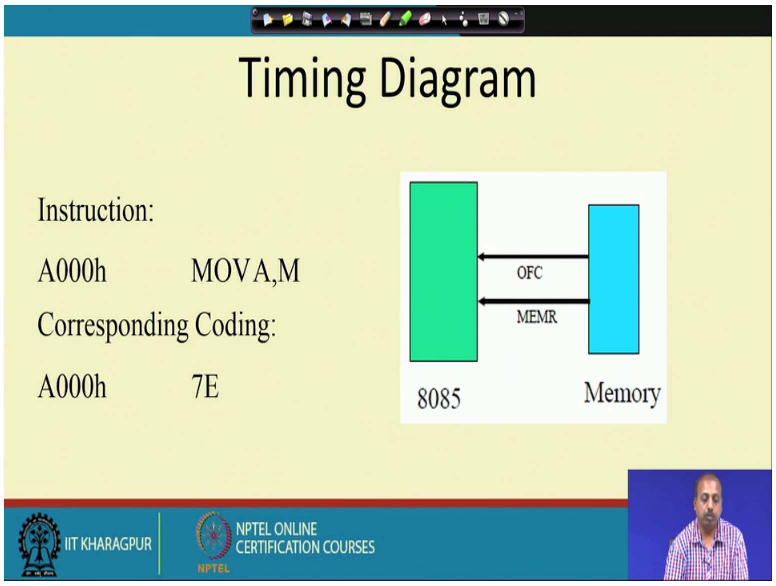
So, it will do that so it will put the higher order address part A 0 on to the higher order address bus, it will put the lower order address part 0 1 hex on to the lower order address bus give the ALE signal and then it will give the read bar signal. So, as a result memory will put this 45 hex on to the data bus and then this 45 hex comes to the processor and the processor put this 45 hex into the C register, because that is the lower order bytes, that goes the C register.

After that it comes to the next memory read cycle for again it put this A 0 hex on to the higher order address bus, 02 hex this the next address is put on to the lower order address bus actually this is constitute this constitutes the program counter. So, this higher order address and this lower order address it combine together into the program counter register and after every memory fetch or this memory read operation. So, this program counter is implemented.

So, that way it is doing this after every fetch. So, this is an instruction fetch operation that is going on. So, till the instruction complete instruction has been fetch this program counter value goes on incremented. So, that is the it becomes A 002 and this is A 0 is put on to the higher order address bus from the PC high and the 02 hex that is for a PC low that comes to the lower order address bus, and similarly when this ALE signal is given.

So, address is latched and when this read bar signal is activated. So, this net is rate from the memory. So, this F0 hex value that comes from memory and it is loaded into the B register. So, this way this way LXI B instruction is executed and we have got all entire timing entire timing diagram so, that elasted how this A side B is done.

(Refer Slide Time: 13:44)



Timing Diagram

Instruction:

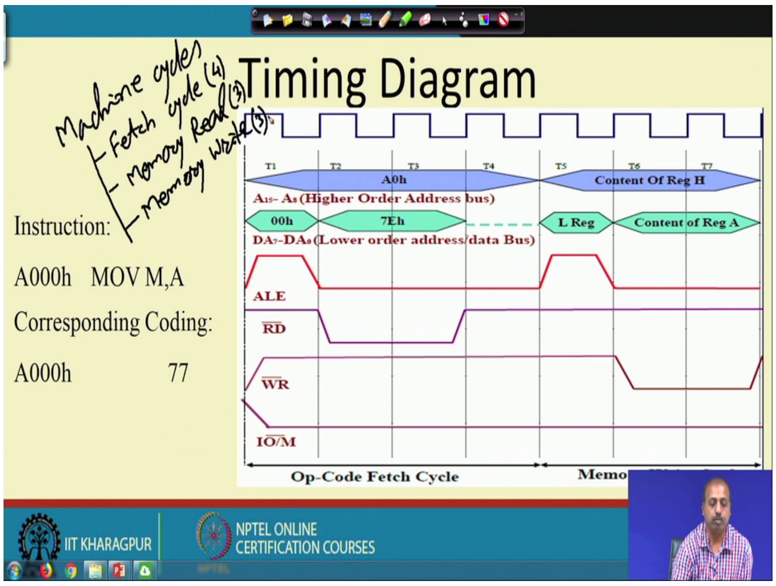
A000h MOV A,M

Corresponding Coding:

A000h 7E

Next we look into another interesting instruction which is MOV A comma M. So, this is slightly tree key because you see that the meaning of this instruction is that your this content of memory location pointed 2 by HL register pair will come into the accumulator, so the corresponding code for this is 7E ok.

(Refer Slide Time: 14:08)



Timing Diagram

Instruction:

A000h MOV M,A

Corresponding Coding:

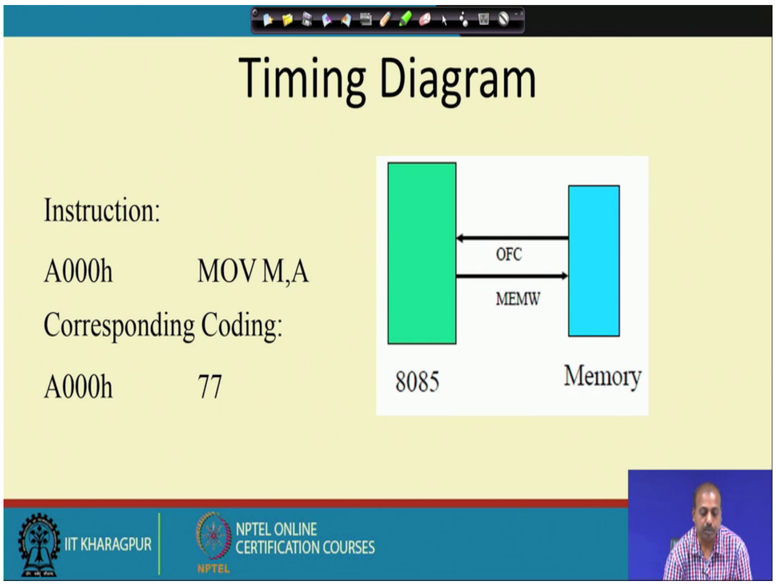
A000h 77

So, this is 7 E and how this instruction is going to be executed is like this first it has to get first part is same. So, that is the opcode fetch cycle and for the opcode fetch cycle. So, the A 0 and 00 they are put on to the address bus ALE signal is activated in T 1 read bar is activated in T 2 and T3. So, this is we get the value 7 E into the instruction decoder. Now this instruction decoder finds that this is an MOV M A comma M instruction, now it has to start a memory right operation.

So, so this is the memory read operation because content of that memory location will come to the A register there is a memory read operation, but what is the higher order address now? As I said that move in MOV A comma M instruction the address is given by the HL pair. So, this H register pair value goes to the higher order address bus and L register value goes to the lower order address bus.

So, that way and the ALE signal is generated. So, it is in this value this address is latched and then the read bar signal is given. So, that the content of the memory location pointed to by HL comes to the data bus, and then when it comes to the data bus where the end when it is available. So, the processor will put the value into the A register, so this way it goes into an opcode fetch cycle followed by a memory read cycle.

(Refer Slide Time: 15:34)



Timing Diagram

Instruction:

A000h MOV M,A

Corresponding Coding:

A000h 77

The other way the MOV M comma A, so that is that is another operation where we are time to use this indirect addressing And here the content of the accumulator register will be saved on to the memory location pointed to by the HL pair and the coding for this is 77.

So, that is available from the manual of 8085 that the code for a MOV M comma A is 77 hex. So, how is it done, so initially we put this higher order address bus contain the A 0, lower order address bus contain 00 and then ALE is given in the first clock cycle read bar is given in second part. So, as a result we get the instruction 77 into the sorry this value should be 77 not 7 E.

So, this comes to the data bus then the processor will understand the, this is a MOV M comma A instruction. So, then it will do a memory right operation because now their all content of the A register should be put on to the memory, so this, so address of that memory location given by the HL pair. So, this H register contains the higher order address part L register contains the lower order address part ALE signal is given to, so latch this L register value to the lower address bus of the memory.

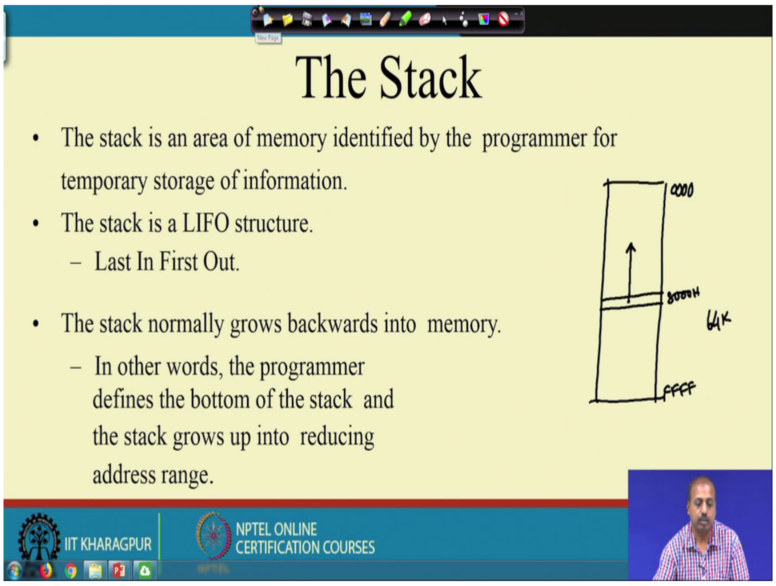
Then you see the right bar signal has been made low. So, right bar signal has been made low and the content of register A is put on to the data bus by the process. So, the memory when it gets the right bar signal, so it whatever is there in your in its data bus it will write it on to the memory location pointed to by its address bus. So, that is done, so content of the, that location comes to the content of A register goes to that particular location pointed to by the address bus of the memory.

So, in to summarize you can see that, so far we have seen different types of cycle. So, these cycles they are known as they are known as machine cycles. So, there are different types of machine cycles that we have seen the first one that we have first machine any instruction to start with. So, it has got a fetch cycle or opcode fetch cycle.

So, that is common for all the instructions. So, after that, so they are may be a number of memory read it can be a memory read it can be a memory right. So, this type of cycles can be there now this fetch cycle. So, it takes 4 clocks clock state or t states where are this memory read and memory right cycles, so they take 3 clock cycles. So, so far we have got introduced with three different types of machine cycles will see some more as we proceed ok.

So, this machine cycles are used for this machine cycles are actually part of this is a combination of this basic clock cycles or T states and that way it constitutes different operation to be done by the processor.

(Refer Slide Time: 18:58)



The Stack

- The stack is an area of memory identified by the programmer for temporary storage of information.
- The stack is a LIFO structure.
  - Last In First Out.
- The stack normally grows backwards into memory.
  - In other words, the programmer defines the bottom of the stack and the stack grows up into reducing address range.

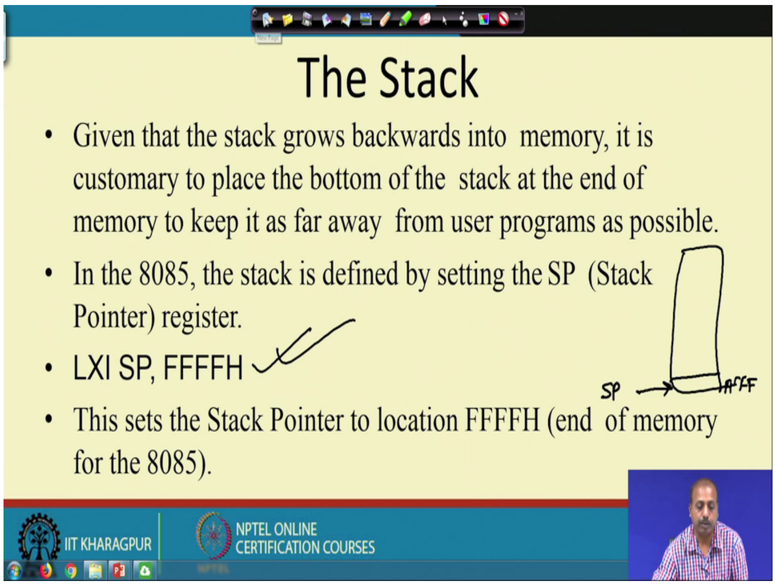
Next will be looking into another very important aspect that we that we that we have in any process are based program execution that is called the stack. So, if you, if you think in terms of the architecture or organization of the system then the stack is an area of memory identified by the programmer for temporary storage of information. So, very often we give do some we want to have some, some space where we do some book keeping job.

So, we keep some temporary data ok, so that we can put into the we can do it into the stack, stack is a last in first out structure because, in case maybe familiar with stack as a data structure. Where there are two operation PUSH and POP and this PUSH operation it writes a it puts the element on to the stack and the POP operation, so it take the element out of the stack and this always operates in the last in first out formulation.

So, stack normally grows backwards into the memory. So, what we mean is like this so this is the memory this is the full memory now we. So, this is the address as say 0 say. So, if I have got a 16 bit 16 bit address bus. So, that we goes from 0000 to F F F F, so total 64 K space now you can say that out of that my stack it will start at this location, so this is the location say 8 8000.

Now normally what is the what happens is that the stack starts growing from this point and it grows towards the upper side towards it grows towards the 0 of course, that is a convention only, but the way this stack manipulation happen implemented in 8085 processor, it happens like this that it grows backwards up to toward the lower and lower addresses. So, programmer as a programmer you can define the bottom of the stack and from that point onward it can it grows up by reducing the address value.

(Refer Slide Time: 21:17)



The Stack

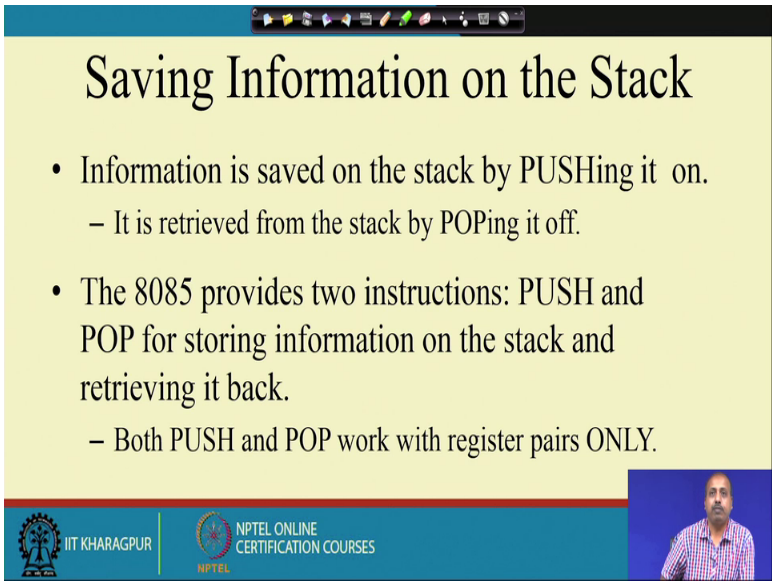
- Given that the stack grows backwards into memory, it is customary to place the bottom of the stack at the end of memory to keep it as far away from user programs as possible.
- In the 8085, the stack is defined by setting the SP (Stack Pointer) register.
- LXI SP, FFFFH
- This sets the Stack Pointer to location FFFFH (end of memory for the 8085).

So, this is the stack operation now will see how this happens in the 8085 in the 8085 processor. So, given that the stack grows backwards into memory. So, it is customary to place the bottom of the stack at the end of memory to keep it far away from the user program. Like in the previous example I said that my memory ranges from 0000 to F F F F X and I put the stack to start at 8000 hex.

Now, there is a possibility that some user program data will clash or program o code will clash with that stack just to avoid it. So, this may be a safe practice to have the stack initialized at the very bottom of the memory. So, this stack means started this F F F X and in 8085 the stack has the stack is defined by setting on special register which is known as the stack pointer register, this is this is 16 bit registers use for accessing the stack. So, what we want is that this is my if this is my memory and then the last address is F F F.

So, I want this F F F to be the stack to be the beginning of the stacks for that purpose we have to initialize the register SP to the value F F F X, and for that we have got this instruction LXI SP comma F F F X. So, the stack pointer gets initialized to the end of the last memory location. So, from last point onwards it start growing and it can grow towards the address 0.

(Refer Slide Time: 22:55)



Saving Information on the Stack

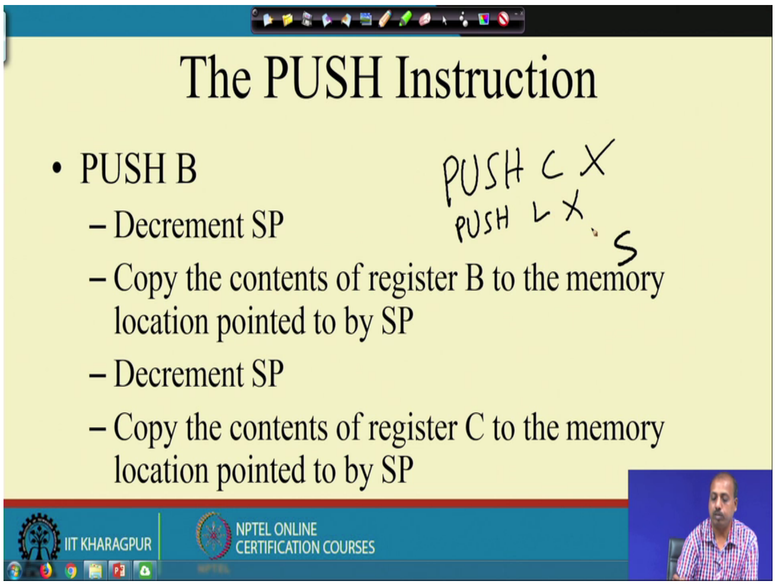
- Information is saved on the stack by PUSHing it on.
  - It is retrieved from the stack by POPing it off.
- The 8085 provides two instructions: PUSH and POP for storing information on the stack and retrieving it back.
  - Both PUSH and POP work with register pairs ONLY.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL

Now, how can you safe or get retrieve data information from stack information is saved on stack by pushing it on. So, you can PUSH data on to the stack. So, that will be saved on to the memory power memory locations pointed to by the stack pointer and then you can retrieved it by doing a POP operation. So, PUSH and POP, so these are the two operations for writing and writing to reading from the stack, 8085 provides to instructions. Similarly, PUSH and POP if we look into stack has a data structure so, will find that it defines to operation PUSH and POP in the stack.

And similarly here also we have got this PUSH and POP as two instructions in the 8085 processor. So, they PUSH and POP they work with registered pairs only, so you cannot PUSH or POP a single register an 8 bit register. So, you have to work with this register pair.

(Refer Slide Time: 23:58)

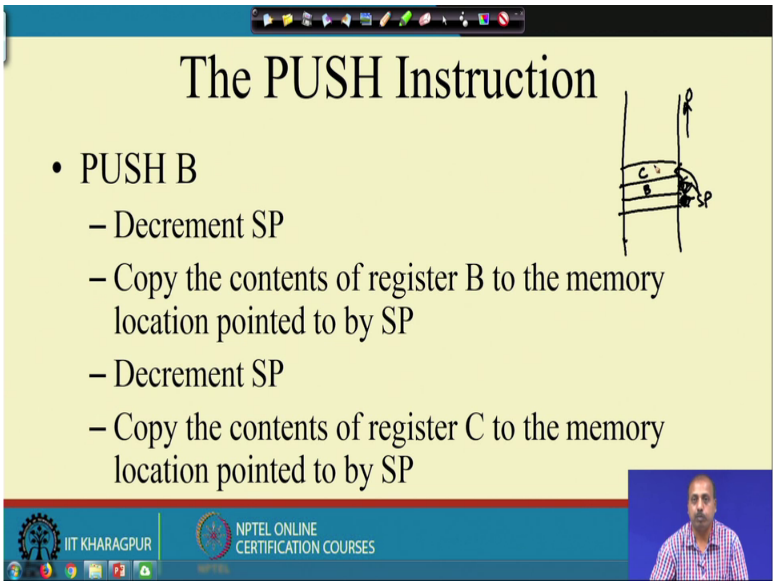


The PUSH Instruction

- PUSH B
  - Decrement SP
  - Copy the contents of register B to the memory location pointed to by SP
  - Decrement SP
  - Copy the contents of register C to the memory location pointed to by SP

For example PUSH B, so when you do PUSH B then this PUSH this actually means we are trying to save the register pair B C onto the stack ok. So, it we cannot have an instruction like PUSH C because PUSH C is not A registered pair. So, B is A register pair consisting of the registers B and C. So, we cannot have an instruction like say PUSH you cannot have an instruction like say PUSH C, so this is not possible or say PUSH E. So, PUSH L so this is not possible, so this is are not ok. Now so how it is executed is like this see I have got a stack.

(Refer Slide Time: 24:48)



The PUSH Instruction

- PUSH B
  - Decrement SP
  - Copy the contents of register B to the memory location pointed to by SP
  - Decrement SP
  - Copy the contents of register C to the memory location pointed to by SP

So, this is the memory and then suppose the at present the stack pointer is somewhere here. So, stack pointer is pointing to this location and this is towards the address 0, lower address now if I am executing this PUSH B instruction. So, what I want is somehow this B and C registered values will be saved on to the stack.

So, what is done this stack pointer is decremented, so stack pointer now points to this. So, stack pointer now points to this location and then copy the content of register B to the memory location pointed to by SP. So, at this location with copy the content of B registered into this and they need decrement the stack pointer further. So, stack pointer now points to this location it points to this location and their we copy the content of the C register.

So, this way it operates, so when it is. So, PUSH B it first decrements the stack pointer then copies the content of B register on to that and then decrement stack pointer further copies the C register content. So, you remember that in case of 8085 this lower order by it is always stored in the lower order address location. Now, out of this B C register pair, so C is the lower order register, so and B is the higher order register. So, this lower order register has gone to the lower order memory address C and this higher order address higher order register has gone to the higher order register higher order value.

So, that is the higher order memory location ok, so this is the always to so when your doing a save saving some copying some 16 bit value into some memory locations. So, this is done the other way first this higher order is copy then the lower order is copy, but that is sorry first the lower order is copy then the higher order is copy, but in case of stack in stack rows in the reverse direction it is going towards 0. So, this lower order register is copied latter first the higher order is copied and then the lower order is copy.