

Microprocessors and Microcontrollers
Prof. Santanu Chattopadhyay
Department of E & EC Engineering
Indian Institute of Technology, Kharagpur

Lecture - 13
8085 Microprocessors (Contd.)

Next, we will look into an example of using register pair as loop counter for generating some delay.

(Refer Slide Time: 00:19)

Register Pair as a Loop Counter

- The following is an example of a loop set up with a register pair as the loop counter.

```
LXI B, 1000H
LOOP DCX B
      MOV A, C
      ORA B
      JNZ LOOP
```

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

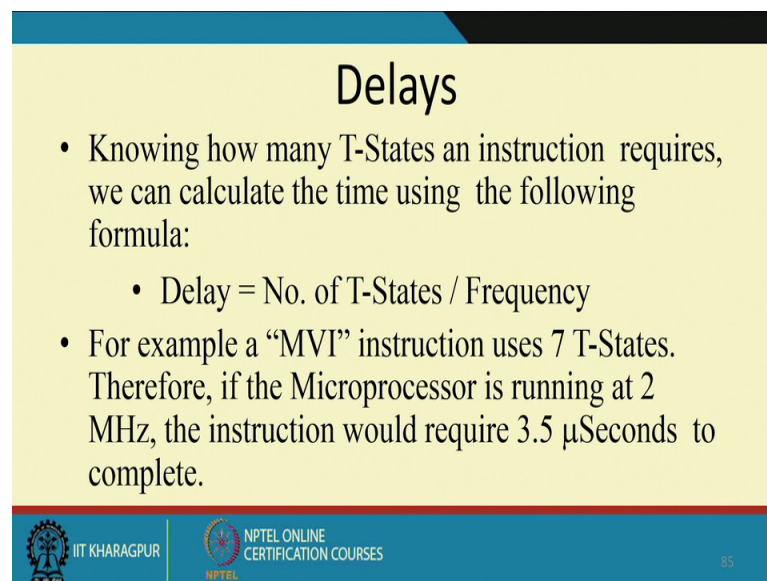
So, you see that in this example. So, first this LXI B instruction, it will load the register pair B C with the value 1000 hex. So, B register will get this one 0 value, and the C register will get the value 0 0. So, as a result total B C pair is now holding the value 1000 hex.

Now, next instruction is DCX B. So, that will decrement the B C pair by 1, and this is actually this is decrementing the pair value by 1. So, actually the C value will become will become FF and this B value will be modified accordingly. So, next we need to check whether the B C pair has become 0 or not. And as we said that this DCX instruction does not affect the 0 flag, whereas that decrement the single digits right DCR. So, that was decrement that was affecting the status flags. So, this DCX instruction does not affect the flag register.

So, we have to take a roundabout way. So, we move the content of C register to a register. So, move A comma C, and with the idea that when this B C register content will become 0, both the B and C register will have 0. So, we do or A B so, or accumulator with the B register. So, be a register has got the value of C. So, if both the registers B and C were 0, then this or accumulator operation. So, this result will be 0, and the or accumulator the or a instruction this affects the status register or the status flags. So, the 0 flag will get affected. So, we can have a loop like JZN loop. So, there that will go back to this instruction whose label is loop.

In different assemblers you will find slight variation like, sometimes these labels are written directly in this fashion where there is just a space between the opcode and the instruction. In some cases, you will find that there is a there will be a colon after this loop. So, that we have got this we have got this loop as a label. So, that is identified in some assemblers like that.

(Refer Slide Time: 02:47)



Delays

- Knowing how many T-States an instruction requires, we can calculate the time using the following formula:
 - $\text{Delay} = \text{No. of T-States} / \text{Frequency}$
- For example a “MVI” instruction uses 7 T-States. Therefore, if the Microprocessor is running at 2 MHz, the instruction would require 3.5 μ Seconds to complete.

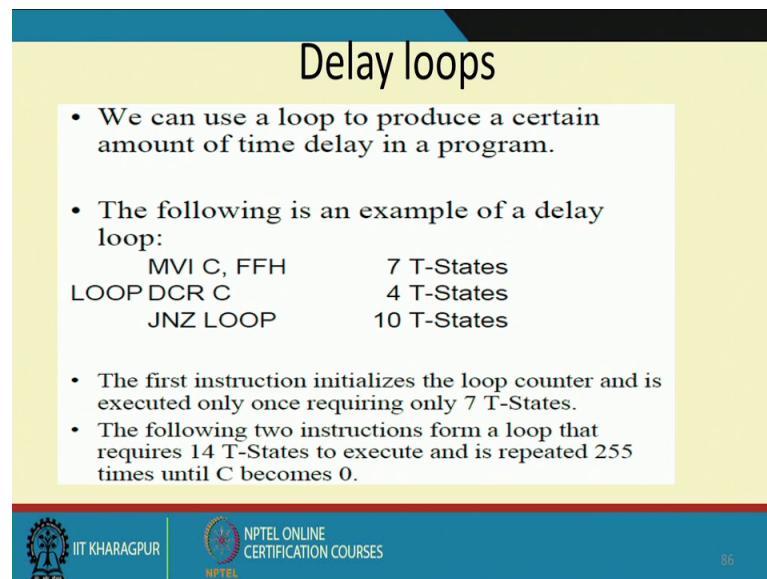
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | 85

So, next we will look into how to calculate the delay that is introduced as I said that this looping is used mainly for this delay generation, because the adapters thus code segme segment that we have seen that is not doing any other meaningful job that way. So, for this purpose we need to know what is the time required for the individual instructions. In 8085 architecture; so this individual clock cycle. So, they are called the T states, or time steps or individual clock steps, you can say. So, as you know that there is a crystal

connected to the 8085 that generates a clock signal for it. And then so, how many clock cycles are taken. So, that is that will that is documented in the manual for every instruction, what we will do will is that we will essentially add all those clock cycles that are needed in different instructions, and knowing the frequency of the oscillator connected crystal connected to the chip. So, we can find out what is the actual delay produced by such a codes code fragment. So, the overall delay that is calculated is given by number of T states divided by the frequency.

So, if we have got this this MVI instruction for example, if we look into the manual you will find that this requires 7 such clock cycles or 7 T s states. So, if the microprocessor is running at 2 megahertz, then this instruction will require about 3.5 microseconds to complete so that you can calculate because that 7 clock cycles. So, that will turn out to be 3.5 microsecond for the micro process. So, using this type of information we can calculate the time needed for any code segment.

(Refer Slide Time: 04:46)



Delay loops

- We can use a loop to produce a certain amount of time delay in a program.
- The following is an example of a delay loop:

MVI C, FFH	7 T-States
LOOP DCR C	4 T-States
JNZ LOOP	10 T-States
- The first instruction initializes the loop counter and is executed only once requiring only 7 T-States.
- The following two instructions form a loop that requires 14 T-States to execute and is repeated 255 times until C becomes 0.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | 86

So, let us go back and to our first example of delay loop that we had, and try to see; what is the amount of delay that it can generate. So, if this code fragment is put inside a bigger code, then it will introduce some delay in that code. So, we will and as I said that this is necessary many a times particularly when you are doing some display part, say for which the display part it is meant for human being. So, our eyes will not be able to change even

locate that changes. So, I have to keep the display for some time when if it changes very fast then the eye will not be able to see that change.

So, these delay loops are used for that purpose. So, we can use a loop to produce certain amount of time delay in a program. So, for example, if we look into this program MVI C FF hex, then because decrement C and jump or not 0 loop. So, if you consult the manual you will find that this MVI instruction takes 7 T states then this decrement C. So, this instruction takes 4 T states, and JZN loop it takes 10 T states.

Now, the first instruction so that is a loop initialization instruction that for a loop counter initialization. So, that is executed only once, if you look into this code fragment like this, but MVI instruction is executed only once, the following instructions that is DCR C and JZN loop. So, that is they will be repeated 255 times, this FF value is the 255. So, it will be repeated 255. So, if you add this number 2 values of T states 10 and 4. So, total one iteration of this loop will take 14 T states. So, that is repeated 255 times. So, you can find out what is the time required. So, 255 into 4, that way it will come.

(Refer Slide Time: 06:47)

Delay Loops (Contd.)

- We need to keep in mind though that in the last iteration of the loop, the JNZ instruction will fail and require only 7 T-States rather than the 10.
- Therefore, we must deduct 3 T-States from the total delay to get an accurate delay calculation.
- To calculate the delay, we use the following formula:
 - $T_{\text{delay}} = \text{total delay}$
 - $T_0 = \text{delay outside the loop}$
 - $T_L = \text{delay of the loop}$
- T_0 is the sum of all delays outside the loop.

$T_{\text{delay}} = T_0 + T_L$

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

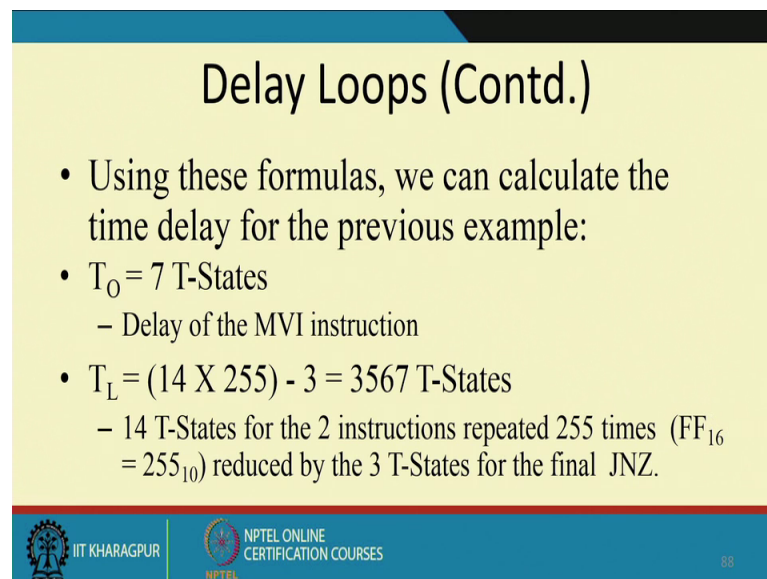
Of course, there is another important issue that the last iteration of the loop. So, JZN will fail, and require only 7 T states rather than 10 T states. So, this is this is a very I should say minute delay calculation, because this JZN loop instruction. So, it takes 10 T state when the loop is successful, because in that case the program counter has to be loaded

with the value from the memory the what whatever is the instruction this offset the loop. So, that has to be loaded into the program counter. So, that takes time.

Whereas, if it is not so if the loop fails that is if the 0 flag is set, in that case that lasts part of the instruction is not executed where the program counter will be loaded with the value of this loop address. So, that is not executed. So, that requires some less number less clock cycles and that is B C that requires 7 T s T states rather than 10; so to calculate correctly. So, we should deduct these 3 T states from the total delay from the loop, and the total delay of the program fragment is given by T delay is T o plus T l where T o is the delay outside the loop T l is the delay of the loop and T delay is the total delay.

So, if there are a number of instructions which are not in the loop, which are outside the loop so, they will contribute to this T o calculation. In our case there is only a single instruction which is initializing the C register with the value.

(Refer Slide Time: 08:34)



Delay Loops (Contd.)

- Using these formulas, we can calculate the time delay for the previous example:
- $T_0 = 7$ T-States
 - Delay of the MVI instruction
- $T_L = (14 \times 255) - 3 = 3567$ T-States
 - 14 T-States for the 2 instructions repeated 255 times ($FF_{16} = 255_{10}$) reduced by the 3 T-States for the final JNZ.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | 88



So, in our case this T o is equal to 7 T states delay of the MVI instruction, and T l is 14 into 255 minus 3, because for the last iteration. So, it will the JNZ will not require that 3 cycles. So, it will require 7 cycles instead of 10. And that boils down to 3567 T states; so 14 T states for the 2 instructions. So, that is repeated 255 times reduced by 3 T states that is for the final calculation of T l.

(Refer Slide Time: 09:07)

Register Pair as a Loop Counter

- The following is an example of a delay loop set up with a register pair as the loop counter.

LXI B, 1000H	10 T-States
LOOP DCX B	6 T-States
MOV A, C	4 T-States
ORA B	4 T-States
JNZ LOOP	10 T-States

 IIT KHARAGPUR  NPTEL ONLINE CERTIFICATION COURSES 89



Now, once this is calculated, so you can find out like what is the time needed like, this is the T total that is required is 7 plus 3567 that is 3574 T state. Now if you know the frequency of operation of the processor. So, you can calculate, what will be the total time very corresponding to this value. So, this way we can calculate the time that is that is the time delays that are introduced by the delay loops. So, to if we want more delay if we want more delay, then one way out is to use this register pair as a loop counter.

So, let us look into the other program that we have taken as an example, where first this LXI B 1000 hex. So, the B C register pair is loaded with 1000 hex then DCX B move A comma C ORA B and JZN loop. So, if you if you consult the manual then this LXI B instruction takes 10 T states, DCX takes of 6 T states, MOV A comma C for T states, ORA 4 T states and JZN 10 T states. Of course, with the exception that for the last iteration JZN will take 7 T states.

(Refer Slide Time: 10:26)

Register Pair as a Loop Counter

- Using the same formula from before, we can calculate:
- $T_0 = 10$ T-States
 - The delay for the LXI instruction
- $T_L = (24 \times 4096) - 3 = 98301$ T-States
 - 24 T-States for the 4 instructions in the loop repeated 4096 times ($1000_{16} = 4096_{10}$) reduced by the 3 T-States for the JNZ in the last iteration.

 IIT KHARAGPUR |  NPTEL ONLINE CERTIFICATION COURSES

90

So, once you know this so, in a similar fashion. So, you can calculate the total number of clock cycles. So, T_0 that is the delay what is introduced by the instructions which are not in the loop that is we do to the LXI in this example. So, that is $T_0 = 10$ T states then inside the loop so, we have got so, we have loaded with 1000 hex 1000 hex corresponds to 4096 in decimal. So, this is and this sum of these T states is 6 plus 4 plus 4 plus 10, that is equal to 24. So, that 24 into 4096 minus 3. So, 3 is reduced because the last iteration will not take 3 cycle 10 cycles it will take 3 cycles less. So, total is 98301 T states.



So, 24 T states for the 4 instructions in the loop repeated for 96 times reduced by 3 T states for the JNZ instruction in the last iteration. So, this way this calculation of T_L will be done. And now you can calculate the total delay by adding T_0 and T_L .

(Refer Slide Time: 11:36)

Nested Loops

- Nested loops can be easily setup in Assembly language by using two registers for the two loop counters and updating the right register in the right loop.
- –In the figure, the body of loop2 can be before or after loop1.

```
graph TD; Start([Start]) --> Init2[Initialize loop 2]; Init2 --> Body2[Body of loop 2]; Body2 --> Init1[Initialize loop 1]; Init1 --> Body1[Body of loop 1]; Body1 --> Update1[Update the count1]; Update1 --> Dec1{Is this Final Count?}; Dec1 -- No --> Body1; Dec1 -- Yes --> Update2[Update the count 2]; Update2 --> Dec2{Is this Final Count?}; Dec2 -- No --> Body2; Dec2 -- Yes --> End([End]);
```

 IIT KHARAGPUR |  NPTEL ONLINE CERTIFICATION COURSES

91

Another way of incorporating delays higher delay is by means of nested loops. So, what we can do like this is the overall structure of a nested loop.

So, we initialize that loop 2, we initialize loop 2, then the this is the body of loop 2, then in the body of loop 2 initialize loop 1, and then we have a body of loop 1 then we update the counter for the inner loop. And if this is the final count, if it is no it goes back the inner loop is repeated, if it is yes then the counter for the outer outs outer loop is in the updated. So, the counter is updated, and then we check whether these are the outer loop has the final count, or not if not, it will go here and do the next iteration of the outer loop.



So, in this way we can have these nested loops, we can implement in assembly level programs by 2 registers for the 2 counters. So, for outer loop 2 and loop 1 we can put 2 different registers to hold the count values.

(Refer Slide Time: 12:45)

Nested Loops for Delay

- Instead (or in conjunction with) Register Pairs, a nested loop structure can be used to increase the total delay produced.

	MVI B, 10H	7 T-States
LOOP2	MVI C, FFH	7 T-States
LOOP1	DCR C	4 T-States
	JNZ LOOP1	10 T-States
	DCR B	4 T-States
	JNZ LOOP2	10 T-States

 IIT KHARAGPUR  NPTEL ONLINE CERTIFICATION COURSES 92

So, this is an example. So, here what we are doing is that MVI B. So, 10 hex B is loaded with 10 hex that is 16, then C is loaded with FF hex that is 255. And then we do so this C is this this B register. So, this is used as the outer loop index and C register is used as the inner loop index; now this decrement C so they were decrementing the value of C. And then we are checking whether the value has become 0 or not if it is not 0. So, we go back to loop 1. So, this decrement C and JZN loop 1. So, they continue to be executed and then once this C register becomes 0 then B is decremented. And then if B is not 0, then we go back to loop 2 where this C register is again initialized with FF hex. And the inner loop continues for some time.

So, this way we can have nested loops implemented, and this can have more delay compared to say single register, or if the values are properly chosen. So, we can we can get better delay than the register pair as a delay.

(Refer Slide Time: 14:05)

Delay Calculation of Nested Loops

- The calculation remains the same except that the formula must be applied recursively to each loop.
 - Start with the inner loop, then plug that delay in the calculation of the outer loop
- Delay of inner loop
 - $T_{O1} = 7$ T-States
 - MVI C, FFH instruction
 - $T_{L1} = (255 \times 14) - 3 = 3567$ T-States
 - 14 T-States for the DCR C and JNZ instructions repeated 255 times (FF=255) minus 3 for the final JNZ

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | 93



So, for nested loops the calculation is similar, excepting that we have to the same the formula is applied recursively for each loop. So, we have to start with the inner loop, calculate the delay of the inner loop, and then we have to go to the outer loop delay calculation where this inner loop delay will become a component.

So, if we look into the previous program, then this inner loop that is DCR C then JZN loop 1 etcetera; so that they will take a number of clocks cycles. So, T_{O1} that is the outer loop 1 is the MVI MVI C FF hex. So, that is this is it is that C register instru initialization instruction that takes 7 T states then for the loop 1 the looping part. So, that we will take as we have seen previously 255 into 14 minus 3. So, that is 3567 T states. Now so, this is the total time needed for the inner loop. So, 7 plus 3567 so, 3574 that is the total number of cycles T states that are needed for doing the inner loop, for one iteration of the outer loop.

(Refer Slide Time: 15:22)

Delay Calculation of Nested Loops

- Delay of outer loop
 - $T_{O2} = 7$ T-States
 - MVI B, 10H instruction
 - $T_{L1} = (16 \times (14 + 3574)) - 3 = 57405$ T-States
 - 14 T-States for the DCR B and JNZ instructions and 3574
 - T-States for loop1 repeated 16 times ($10_{16} = 16_{10}$) minus 3 for the final JNZ.
 - $T_{Delay} = 7 + 57405 = 57412$ T-States
- Total Delay, $T_{Delay} = 57412 \times 0.5 \mu\text{Sec} = 28.706 \text{ mSec}$

IIT KHARAGPURNPTEL ONLINE
CERTIFICATION COURSES94

Next this outer loop calculation has to be done. So, in the outer loop we have got this MVI B 10 hex as the instruction outside the loop body. So, that will take 7 T states, and this this loop 1. So, what is the total time needed now? So, this is basically 16 times. So, the value was one 0 that is that is so, it will be repeated 16 times 16 into so, this 14 is the initialization part of this inner loop. And plus, this 3574 into 14 plus 3574. So, that will be multiplied by 16.

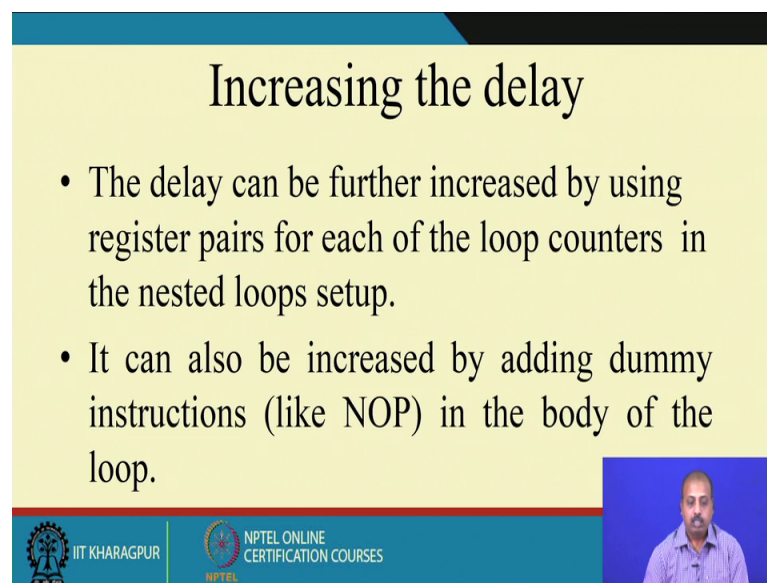
So, these 14 T states are coming from the DCR B instruction and the JZN instruction. So, this is basically from this; so DCR B and this JZN. So, they are actually taking that extractor that 14 cycles of 10 plus 4 14. So, that is a part of the outer loop. So, they are included here. And this 3574 so, this is basically the T states for loop 1. So, that is repeated one 16 times. So, this is gives the 3 14 plus 3574 this is the one iteration of the outer loop. And that is that is multiplied by 16 so that we can get the delay for the outer loop minus 3 because for the last iteration that DCR B the JZN instruction will require less clock cycles. So, that gives a total of 57405 T states.

So, total delay produced by that nested loop pair is 7 plus this 57405; so 57412 T states. So, total delay introduced. So, if we assume that the clock period is 0.5 micro second. So, that this frequency is accordingly 1 upon 0.5 micro second. So, it is total delay is 28.706 millisecond. So, this is the precise delay produced by this instruction, this that

particular program how it nested loop. So, this particular program it has got the precise delay of 28.706 milliseconds.

So, this is the beauty of assembly language programming. So, if the same thing you do with high level language program. So, we are not sure how much time these instructions high level instructions we will take. Because that depends on the way the compiler translates the program into machine code, but if you are looking into the assembly language or you are writing the program in assembly language, then the programmer has will know the exact instruction that is put into the program and accordingly can calculate the delay calculate the time needed for a program. So, that way this assembly language programs are very much important when you are looking for very precise timing information.



(Refer Slide Time: 18:21)



The slide has a yellow background with a blue header and footer. The title 'Increasing the delay' is centered at the top. Below it are two bullet points. In the bottom right corner, there is a small video inset showing a man speaking. The footer contains the IIT Kharagpur logo and the NPTEL Online Certification Courses logo.

Increasing the delay

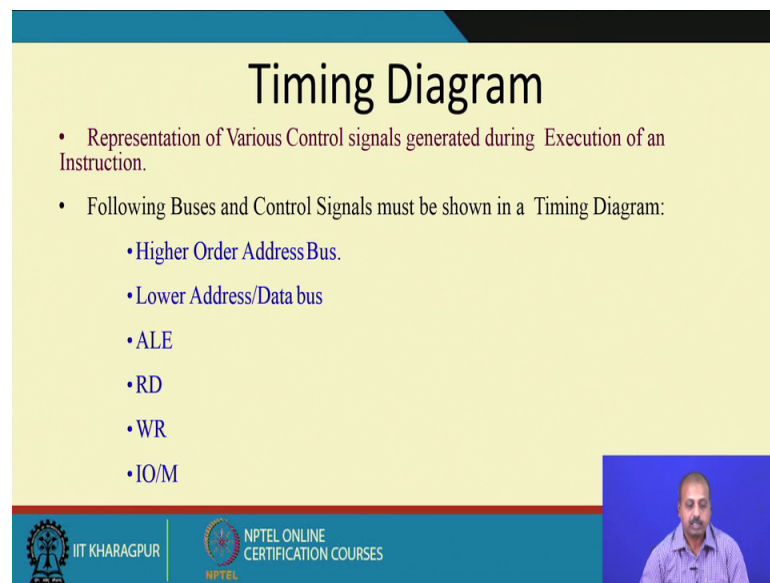
- The delay can be further increased by using register pairs for each of the loop counters in the nested loops setup.
- It can also be increased by adding dummy instructions (like NOP) in the body of the loop.

 IIT KHARAGPUR  NPTEL ONLINE CERTIFICATION COURSES

How to increase the delay further? So, if you are not satisfied with whatever we have done. So, you can introduce some dummy instructions like NOP, no operation the there is a there is a special instruction called NOP that is there in almost all the processors as well as in 8085. So, in the body of the loop you can put a NOP instruction. So, if you put a NOP instruction. So, that will introduce some delay into the system and that will be. So, one NOP instruction actually takes 4 clock cycles; so in this program if I put a NOP here or there. So, if I put a NOP so; that means, the inner loop will that the body of the inner loop will get extended by 4 more T states.

So, instead of this being 255 into 14. So, it will be 14 plus 4 18. So, this value will become 18, as a result this value will change. And that will finally, affect the total clock. So, this value will get modified. So, total value this one will get changed. And that will effect that delay. So, you can put say a number of NOP instructions. So, if you want slightly more extended delay. So, you can put a number of NOP instructions in the innermost loop so that that delay value is increased.

(Refer Slide Time: 19:44)



The slide titled "Timing Diagram" contains the following text:

- 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

The slide also features logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES, and a small video inset of a speaker in the bottom right corner.

Next, we will look into another important issue like the sequence of operations that are done in the microprocessor when some instructions are being executed. So, this is documented in the in the manual of any processor. So, if you if you remember in the architecture initial architecture lectures we have seen like for doing operations like ADD R 1 comma R 2 we have to you have to activate some of the control signals in a proper sequence.

So, here also the same thing like the now we know the 8085 architecture, now here also for doing some operation executing the instructions so, we will need to you will need to control those number of control signals and 8085 the decoder and sequencer that exactly does this operation. So, we will look into some of the instructions, and how they are executed over number of T states. So, that will be seen.

So, this timing diagram actually this is a representation of the various control signals generated during execution of an instruction. So, these are actually the signal lines that

are monitored. The higher order address bus lower order address bus address data bus. So, this is as you know that is a multiplexed, then the ALE signal read write and I O M bar. So, these are the signals that will be monitoring for the timing diagram purpose.

And as you remember that this read write. So, they are all complimentary signal read bar write bar. So, they are active when the value is low, and this particular I O M bar. So, here also this is there is a bar. So, that signal means if the operation if it is to in your memory access in that case this I O M bar line will be equal to 0, and if it is doing and input output from a device, then it will do this I O M bar line equal to 1.

(Refer Slide Time: 21:47)

The slide titled "Timing Diagram" contains the following information:

Instruction:	
A000h	MOV A,B
Corresponding Coding:	
A000h	78

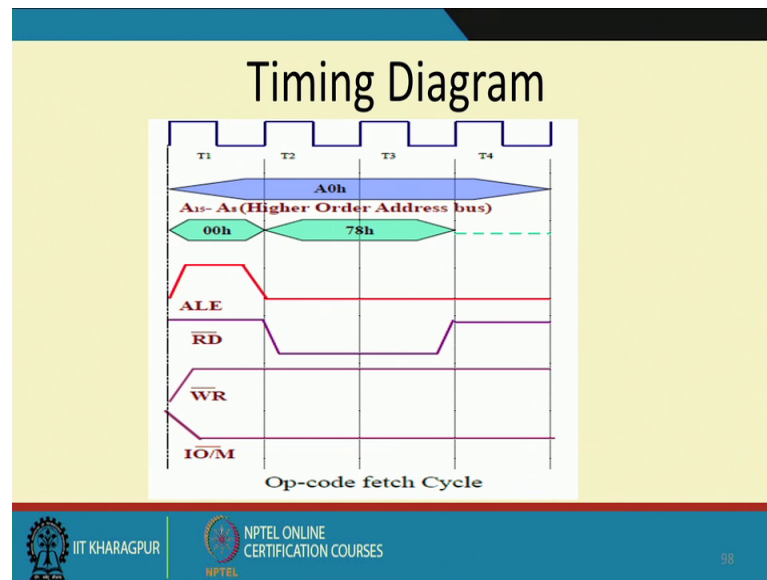
The diagram shows a green box labeled "8085" and a blue box labeled "Memory". An arrow labeled "OFC" points from the Memory box to the 8085 box, indicating the opcode fetch cycle.

Logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES are visible at the bottom of the slide.

So, we will explain it as we proceed.

To look into the first example, let us look into the instruction move A comma B. Suppose I have got so, this move A comma B it is stored at the memory location A 0 0 0 hex. So, the first thing that has to be done is that this instruction has to be brought to the instruction register from the memory. So, if you look into the code of this so, this move A comma B the code is 78 hex. So, this operation of bringing this value 78 into the instruction register is known as the opcode fetch cycle or OFC. So, this will be called OFC or opcode fetch cycle. So, in this cycle the opcode 78 will be brought from the memory to the processor, the instruction register.

(Refer Slide Time: 22:42)



So, this is the timing diagram. So, you see that the first so, if I just go back slightly. So, you see in this address it is a 16 bit address out of that this A 0 is the higher order address byte and 0 0 is the lower order address byte. So, to put A 0 0 0 on the address bus A 0 will be put on to the higher order address bus, and this 0 0 will be put on to the lower order address bus. So, that is done. So, for this entire operation opcode fetch cycle it takes 4 T states. So, T 1 T 2 T 3 and T 4 so, T 1 starts at this point it ends at this point then T 2 starts at this point it aims at this point and these rising and falling edges of the signals they can be sampled by the memory to know the exact timing duration where the transitions are occurring.

So, this this higher order address bus it is getting the value stable value, from you can say after this T 1 has come down it has made a transition from high to low. So, this T of this higher order address bus has got this value A 0 in it. Then lower order address bus so, when this is it is. So, at this junction; this that the transition of this clock from high to low by that time the lower order address bus will contain the correct value. So, this A 0 is there in the higher order address bus, and 0 0 is there in the lower order address bus, and the processor also generates the ALE signal.

So, the as you remember that there is a latch connected. So, this lower order address bus value 0 0 will be used by will be latched on to that external latch via this ALE signal, and this ALE signal is generated at the T 1 clock state always. So, at T 1 cycle so, this is the

ALE signal will be generated. And this ALE signal will be used to latch the value 0 0 onto the address latch. And then after that ALE becomes low and so, after this perform this point onwards. So, that the address bus content is the lower order address bus content is not meaningful, because ALE has gone low. So, latch already has got the value 0 0 there.

So, for the memory now A 0 is in the higher order address bus 0 0 is in the lower order address bus. So, and the processor activates this read bar signal. So, it is read bar. So, far it was high now it becomes low. So, getting the read bar signal and the address A 0 0 0 the memory will put the corresponding location content onto the data bus. So, 78 hex it comes onto the data bus.

And so now you can say that this is actually towards the processor the from the memory to the processor. And you see that the write bar line. So, this is continually high, because this is not a memory write operation. So, of course, fetch is a memory read operation we are trying to read something from the memory. So, this write bar line is continually high, and read bar line will go become low for some time between say from T 2 to T 3 this read bar line will be low. And this I O M bar line. So, this line is low because this is a this is a memory operation memory read operation. So, this I O M bar line will be equal to 0, as a result this is a memory operation.

So, at the so, you can see that by this time by the junction of T 2 and T 3 memory is expected to put the value 78 hex on to the data bus, and the processor will be able to read it. And then in this part so, this T 4 you see that nothing very significant happens in T 4, as far as these signals are concerned. So, what happens is that it is though it is not documented, but it is expected that the processor is actually doing that decode operation at this phase to get the meaning of the instruction. So, here it is doing the opcode fetch cycle, and every instruction has got this opcode fetch in it. So, the processor so, as I said that if you reset the processor, if you reset the processor, then it will be it will be putting this program counter value on to the address bus; so it will on the higher order address bus the PC high will go lower order address bar bus the PC l the lower order 8 bits will go, and this ALE signal will be generated. Then this read bar signal will be generated.

So, that whatever is there in the content in that particular memory location, it will be copied and into the instruction register and processor will try to execute that instruction;

again after whatever be the meaning of that instruction as soon as that instruction is over. So, it will again generate the ALE signal putting the PC high and PC low values onto the bus. So, this way this opcode fetch operation goes on.