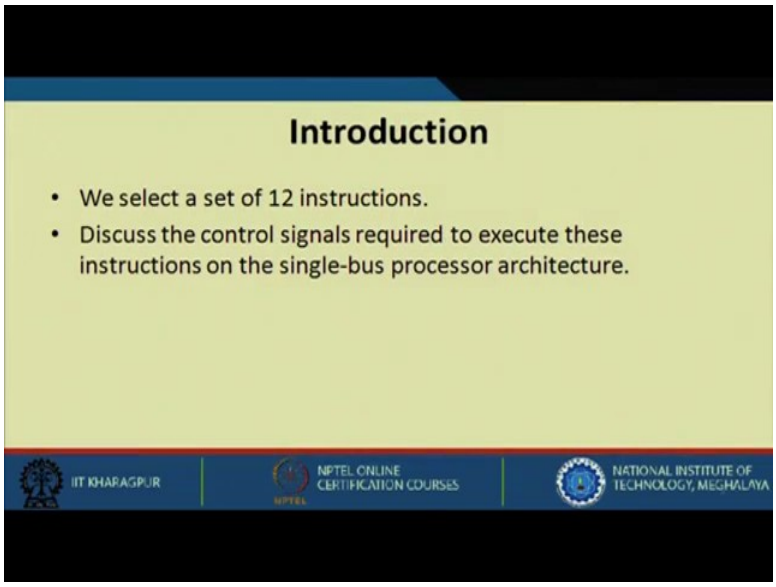
Computer Architecture and Organization
Prof. Kamalika Datta
Department of Computer Science and Engineering
National Institute of Technology, Meghalaya

Lecture – 19
Design of Control Unit (Part 3)

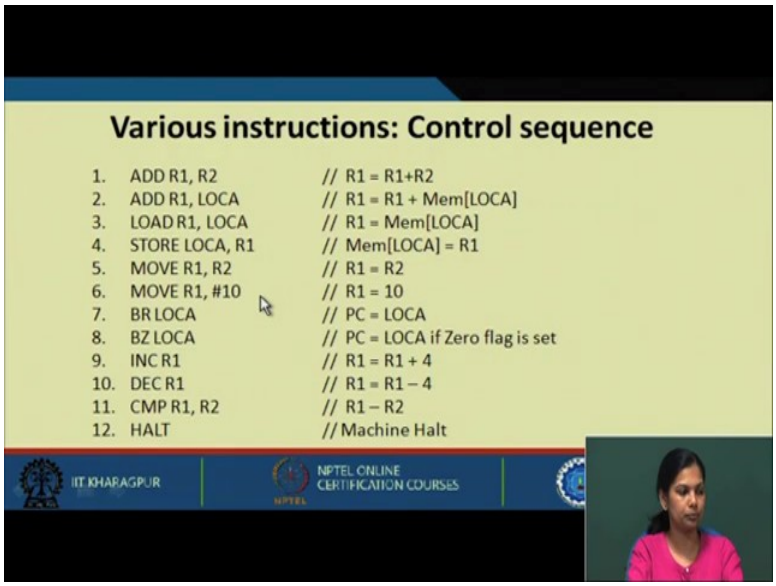
Welcome to the next lecture. In this lecture, we will continue with the design of control unit. Till now we have seen single bus organization internal to CPU, and multi bus which is a three-bus organization again that is internal to CPU. Now, we will be looking into how we can execute various kinds of instructions using a typical single bus architecture.

(Refer Slide Time: 01:06)



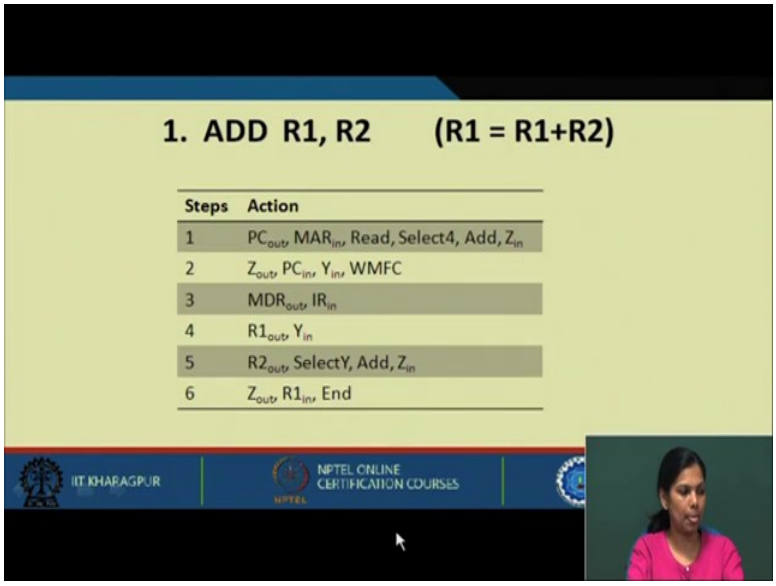
So, for this what we have done; we have selected a set of 12 instructions. And we will discuss the control signals required to execute these instructions on a single-bus processor architecture.

(Refer Slide Time: 01:23)



So, for showing the control signals for various instructions, we have chosen these set of instructions. This set includes ADD with register, ADD with memory operand, loading a word, storing a word, moving data between register, moving an immediate data to a register, unconditional branch, conditional branch, increment, decrement, compare and of course, halt. We shall be showing the various control signals that are required to execute the following set of instructions.

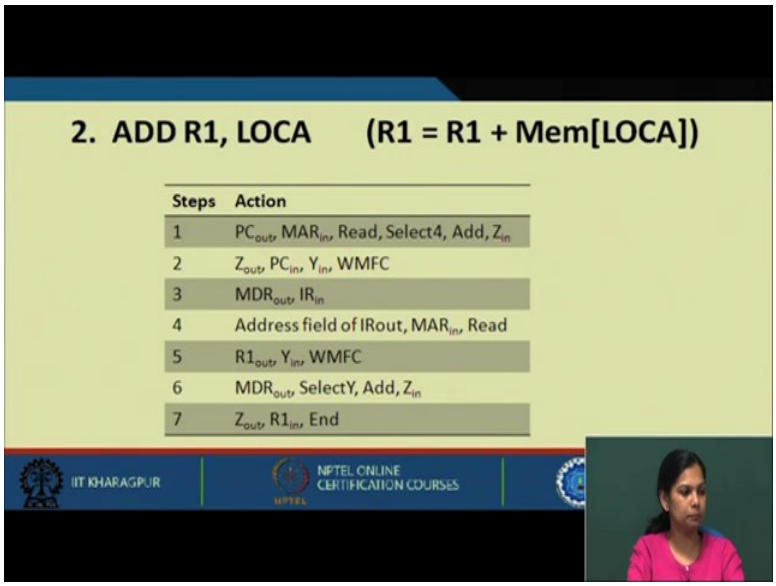
(Refer Slide Time: 02:20)



Let us see this. As we know that for all the instructions that I will be showing you, the first three steps will be the same. And I have taken single bus architecture to show all the instructions. So, let us see this; PCout, MARin, Read, Select4, Add, Zin. This set of control signals needs to be activated at time step 1 for execution. So, what it will do, it will output the value of PC into the bus. And then it will be put it into MAR, then we activate the Read control signal, Select4, PCout. So, it will be available to the A input, and when we do Select4 then A input will be having 4. We perform add, and after addition we put the result in Zin. Now the incremented value of PC is in Zin. We do Zout, and we again put it in PC using PCin signal, and then we also put it in Y. We will be seeing course of time why we are putting it in Y, and then we wait for MFC.

In the next step we do MDRout and IRin; that means, after the MFC is set by the memory unit. What we can do is that we know that the instruction is now available in MDR. Once the instruction is available in MDR, it can be brought into IR. Now, at this stage it is ready for decoding and execution. So, from the next step the instruction gets decoded, and after decoding it is ready for execution. So, this instruction basically adds the content of R1 and R2 and stores back in R1. So, for this one input should directly come through bus and the other input should come through Y. So, we are doing R1out, Yin. So, one of the register value is coming through Y. Next we are doing R2out, SelectY, Add, Zin. So, now R2out is available directly from the bus, we perform SelectY, Y will get selected we perform Add and put the result in Zin. And finally, we do Zout and the data is now available in R1 and finally, End.

(Refer Slide Time: 05:37)

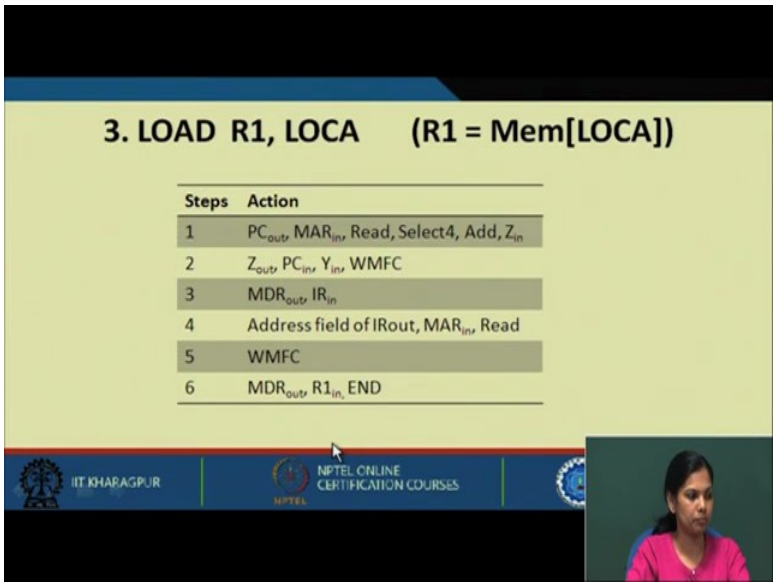


Let us move on. Now, this is for an extra memory access. So, what this instruction will do, it will load the value from LOCA in memory, it will add it with content of R1, and it will store back to R1. Let us see how we can do it. Now, I will not be repeating this again because first three steps are again the fetch, which is same for all instructions. Let us see from next step what we need to do. In the first step, this is a memory location. So, this particular value should be put in MAR and we need to activate the Read control signal. Once WMFC signal will arrive, then the data available in MDR will be brought in, added with R1 and will store back in R1.

Again let us see what are the steps that are required for this operation. First you need to make available this; this can be available from the address field of IRout. So, address field of IRout will make available this content which goes to MARin at the same cycle, and we perform Read, we activate the Read control signal. After this is done then it is hit to the memory and it start reading the data. In the next step, we need to wait for the MFC. At the same time, we can also make available the value of R1 into Y. So, we can do R1out, Yin. So, R1 value is now available in Y register.

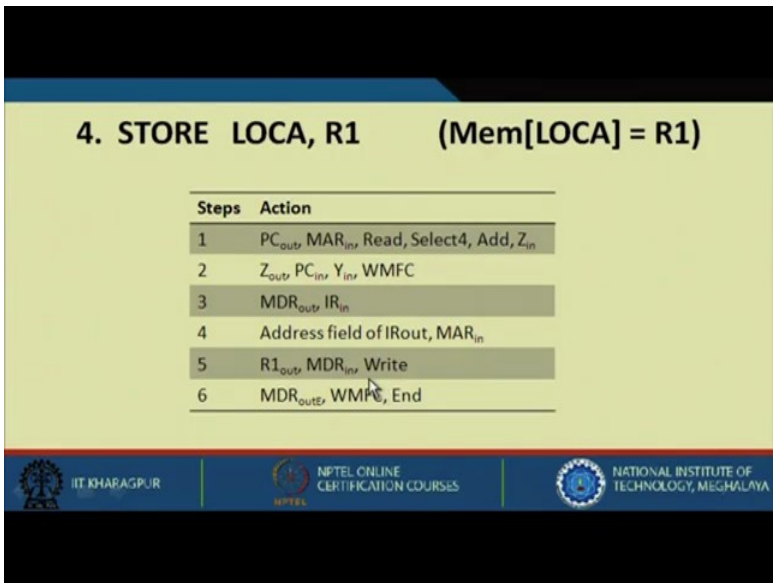
Next what we do after this MFC signal is provided by the memory, then the data will be available in MDR, we do MDRout, we do SelectY because already in Y the R1 value is present, we perform Add and we do Zin. So, after performing this Zin the result is now in Z register, but the result has to be in R1. So, we perform Zout, R1in and finally, we End.

(Refer Slide Time: 08:22)



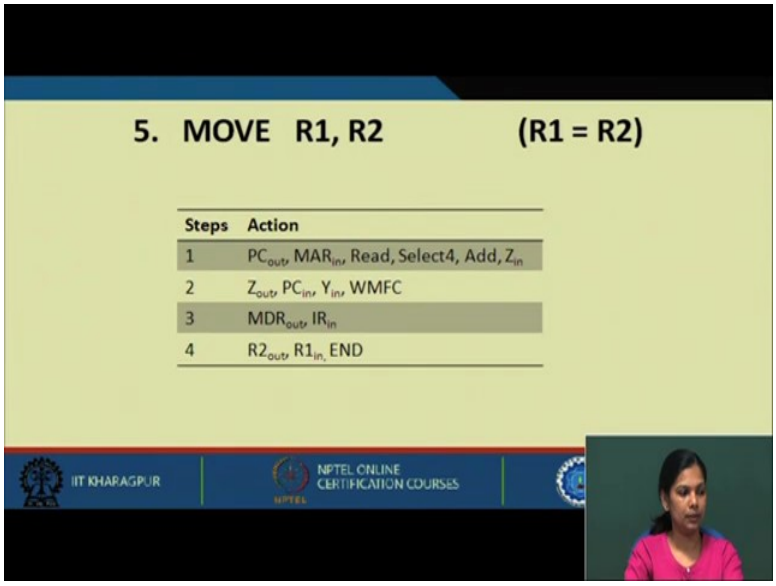
Next let us see the instruction to load the value from a memory location. We already did it in the previous instructions, but just for loading let us see what to do. In the same way first three steps will be same, and from the fourth step this will be present in the address field of IRout that is put in MAR using MARin, reactivate the Read control signal. And then we wait for MFC. Once MFC is performed then the value is available in MDR; and from MDR we have to put it in R1. So, we do MDRout, R1in and End for loading a word from memory.

(Refer Slide Time: 09:16)



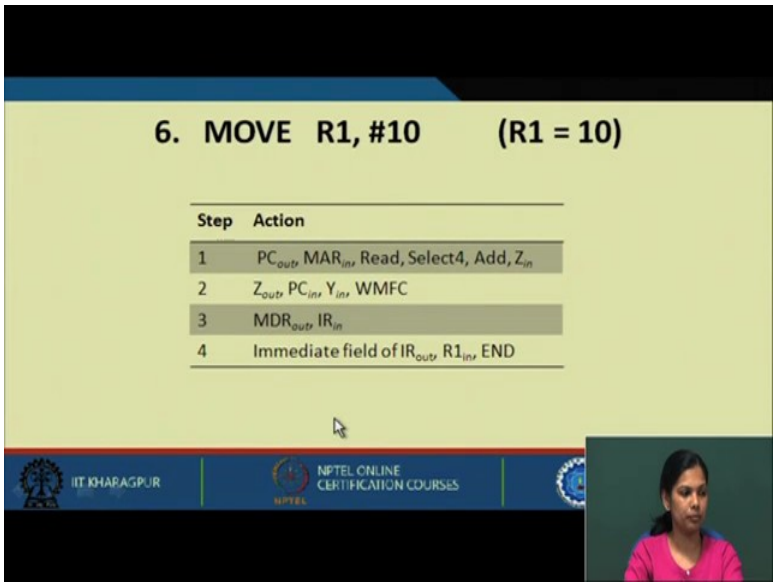
Let us move on how to store a word into memory. Once we activate the Read control signal, the data is read, and it is stored in MDR. For write operation, similarly we know that we have to put the address in MAR and the data that needs to be written will be put in MDR, and then only we can activate Write control signal. So, we need to do these two things; earlier we were just putting the value that the address that needs to be read into MAR, and we were activating the Read control signal, but for Write, you need to put the data in MAR as well as in MDR and then you activate the Write control signal. So, what we are doing here from step 4, address field of IRout, MARin, R1out, MDRin because the content of R1 needs to be written in to LOCA. So, R1 content should be in MDR and activate Write control signal. And in next step we wait for MFC, after the MFC has arrived then we can End it.

(Refer Slide Time: 10:40)



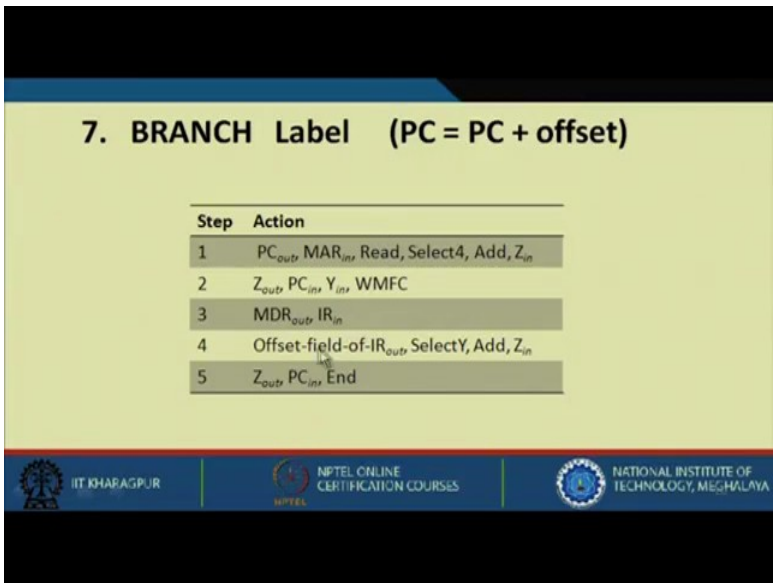
This is very simple moving R2 to R1. So, R2out, R1in --- the content of R2 will be moved to R1.

(Refer Slide Time: 10:57)



Many cases we require to store an immediate value into a register. So, immediate value is stored in the immediate field of IRout. So, the immediate field of IRout value we need to put it in R1. So, we simply do immediate field of IRout. R1in. So, value of this that is 10 will be moved to R1.

(Refer Slide Time: 11:32)

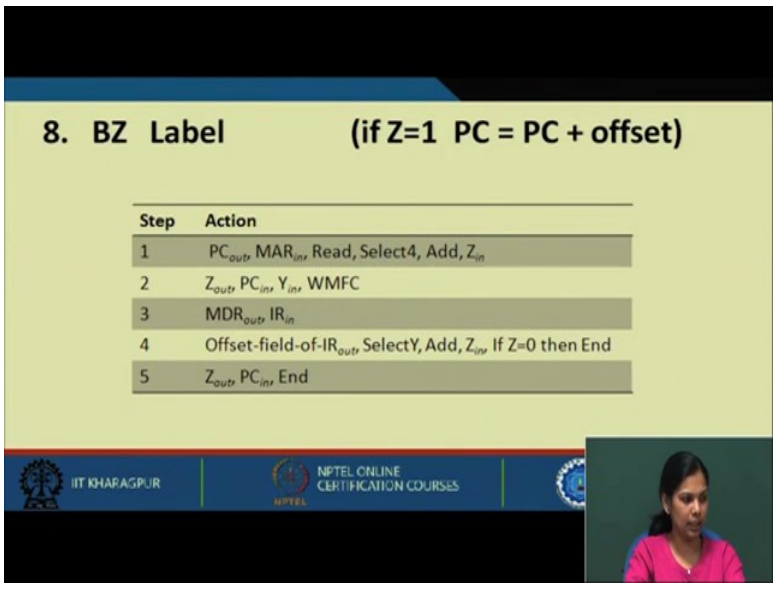


Now, coming to branch instruction. So, in unconditional branch there is no condition; if branch is there we need to load the PC with the branch address. So, we know that every time we fetch an instruction the PC value gets incremented to point to the next

instruction. In any instruction in the fetch phase we do not know whether it is a branch instruction or not. So, in any case we are incrementing the PC value, but after decoding the instruction, we will know that this is a branch instruction. Once we know after decoding that this is a branch instruction then what we need to do, we need to load the value of the PC with the branch address that needs to be calculated, and then it should be loaded into this again.

So, let us see what the steps that are required for this purpose. Now, recall our discussion where we said that in step 2, what we are doing, we are doing Zout, PCin and we are also doing Yin. And I told you at that point that this Yin is required for branch. Now, let us see why it is required for branch. Branch to Label that means content of the PC value should be added or subtracted depending on the offset, and we get a new PC value where we have to go and fetch the instruction. So, in this case, what we are doing offset field of IRout will be available; and if we SelectY because you see in Y, PC value is incremented. PC value is already there. So, in Y as incremented PC value is already there, we need to add that PC value with offset, so that is what we are doing, offset field of IRout, SelectY, because Yin already contains the updated PC value we Add and we do Zin. And finally, we do Zout, PCin, and End. So, these are the following steps required when it is an unconditional branch.

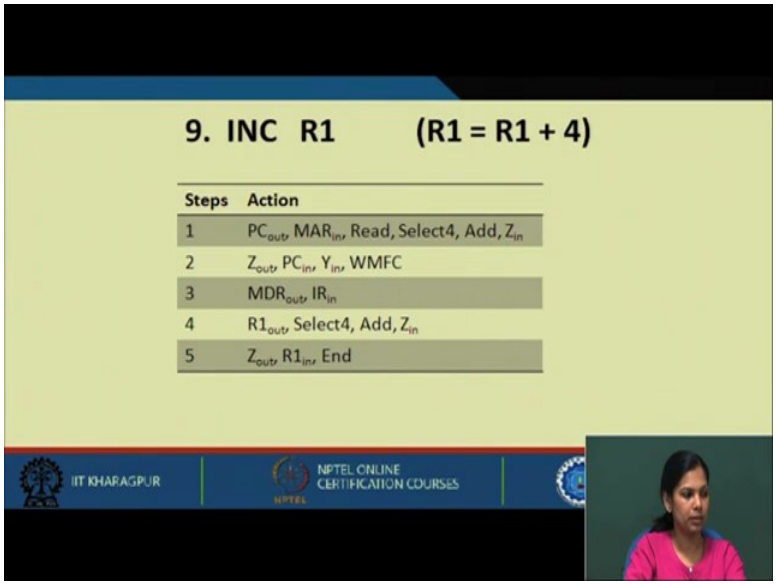
(Refer Slide Time: 14:33)



Let us move on and see for a conditional branch. So, for a conditional branch what happens if the condition is met then only it will branch; otherwise normally it will proceed to the next instruction. So, similarly here first three steps will be same and finally, here offset field of IRout, SelectY, Add, Zin. Here we need to check if Z is equal to zero or not, that means if the zero flag is set or not. If Z is equal to 0 that means, zero flag is not set. If the zero flag is not set then there is no need to do anything you can stop here, and then the PC value that already got incremented to the next one. So, it will fetch the next instruction as usual and it will do it because the branch condition is not satisfied.

Let say if Z is equal to 1, then what happen we have already calculated Z. This register Z value here which is offset value is added with content of PC; we will do set out PCin and then End. Now, PC will be loaded with the conditional branch address. With this that we have calculated using this particular offset. So, branch if zero which is a conditional branch, a condition needs to be checked that is here the flag, and then accordingly it will be Z if the condition is not satisfied it will End here and the PC will next fetch the next instruction. And if it is satisfied then it will load the PC with the branch address.

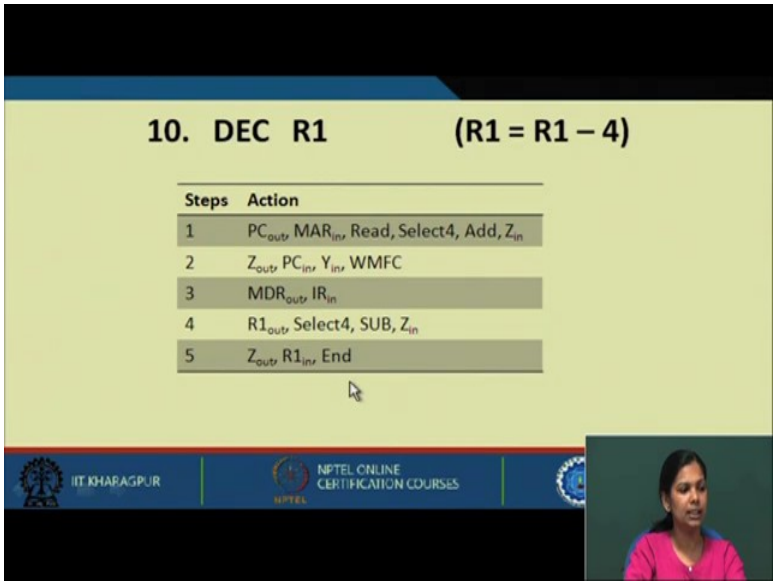
(Refer Slide Time: 16:43)



Let us see the next instruction INC R1, where R1 is incremented by 4 and stored back in R1. First three steps will be again same, what we have to do R1out, Select4. If you Select4 then 4 will be selected from the MUX, which will go to one of the inputs of

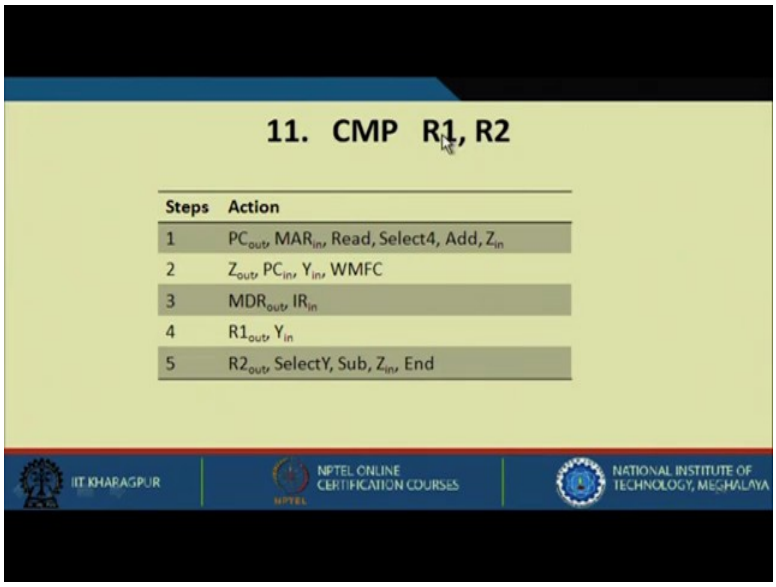
ALU. We add it and then we put it in Z. And finally, the value of Z is put in R1 by Zout and R1in and finally, we End. So, this is for increment.

(Refer Slide Time: 17:25)



Similarly, we can have for decrement where everything will be same the only difference being this operation that is SUB. Here we will do R1out, Select4, SUB and then Zin and then Zout and R1in; the value of Z will be put it into R1. The decremented value will now will be present in R1.

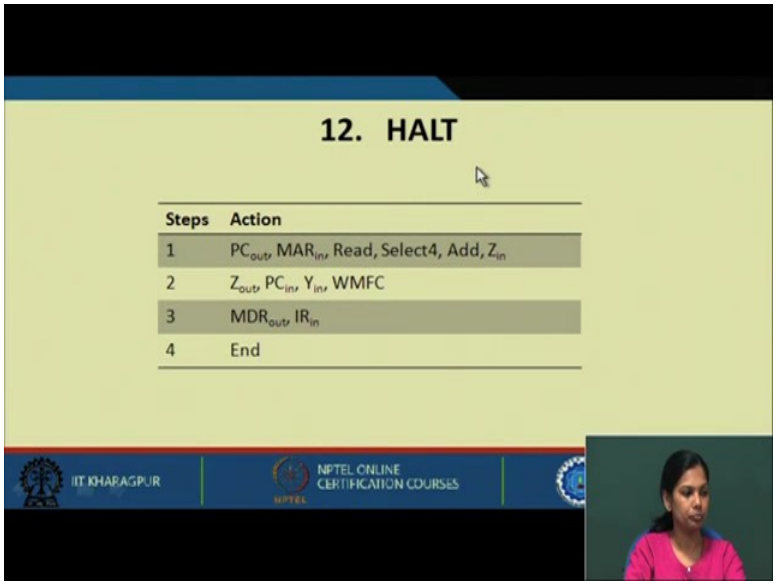
(Refer Slide Time: 17:52)



Compare R1 and R2. So, generally when we perform an operation depending on that operation the flags gets affected, the flags are set or reset. So, how do we compare? If you subtract these two numbers, if we subtract R1 - R2 and the result is zero, then the zero flag will be set. Let say we subtract R1, R2 and the result is positive; that means, R1 is greater than R2; let us say we subtract R1, R2 and the result is negative, that means R2 is greater than R1.

So, depending on on the subtraction various flags will get affected, and accordingly we can compare it. So, what we are doing here, we are doing R1out, Yin, R2out, SelectY, and then we do SUB and then Zin. So, after SUB and Zin, after the operation of ALU, flags will get affected; and accordingly we can see based on the flag value. So, after performing this, we have to check a flag value to see whether R1 is greater than R2, or they are equal, or R2 is greater than R1 depending on this.

(Refer Slide Time: 19:36)



The last instruction is HALT. HALT will stop the execution of the instruction. But HALT itself is an instruction. So, for every instruction, what we do we fetch the instruction and then we decode the instruction. So, the first three steps will be same for HALT and after at the end of step 3 once it is decoded, it is known that this is a HALT instruction. So, for a HALT instruction finally, what it will do, it will End. In step 4 it will End.

So, in this lecture we have shown basically the various instructions, we have taken a set of 12 instructions and we have shown you that how using single bus architecture we can execute these instructions.

Thank you.