Computer Organization and Architecture: A Pedagogical Aspect

Prof. Jatindra Kr. Deka
Dr. Santosh Biswas
Dr. Arnab Sarkar
Department of Computer Science & Engineering
Indian Institute of Technology, Guwahati

Lecture - 18
Handling Different Addressing Modes

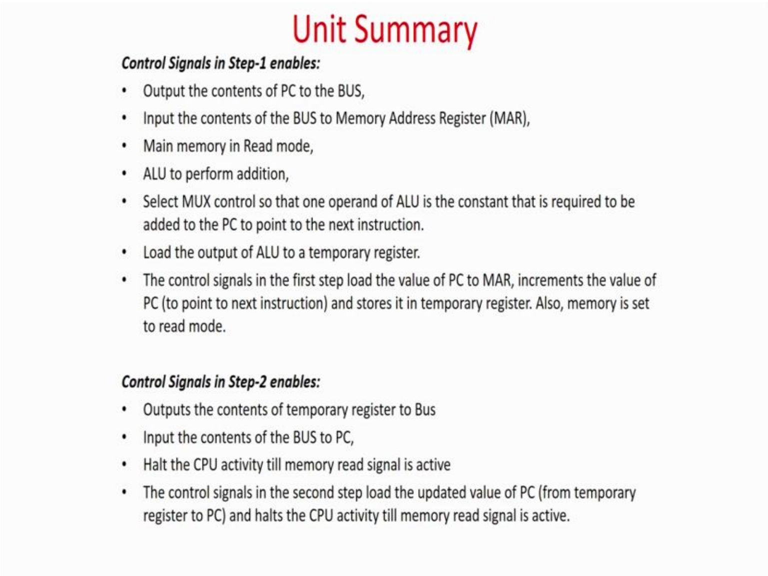
Welcome to the next unit, that is unit 4, on the control unit module. Where we will be discussing the control instructions or basically control signals and the micro instructions, which are required for handling different addressing modes. So, in the last unit basically which was on control signals for complete instruction execution, which we have discussed in the last unit we have taken some temporary instructions and then we have seen how different control signals are generated.

(Refer Slide Time: 00:49)

And we have also considered a single bus architecture and we saw of that we have seen in a very detailed manner that for any such given instruction, like fetch, decode and execute for the complete sequence, how what are the different control instructions or control signals and the corresponding micro instructions required and generated.

Now, we are going to look at, now we are just going to extend the last unit, where we will be looking at different addressing modes like immediate direct, indirect, register indirect some generic addressing modes and then we will see what are the different type of control signals generated. In fact, we will be again looking at the single bus architecture so. In fact, this unit is a smaller unit which is just an extension of the previous unit and now we will be looking at different addressing modes in particular.

(Refer Slide Time: 01:36)

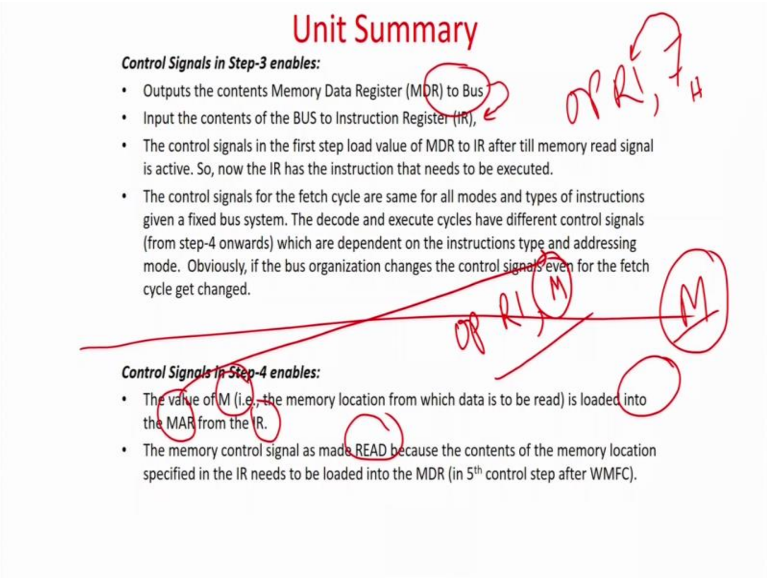


So, basically as we are handling with a pedagogy sense. What is the basic unit summary? So, will be basically looking at the different steps of instruction execution and we look at different addressing modes and what are the different type of control signals generated.

As we have already discussed in the last unit that basically the first 3 steps that is step 1, step 2 and step 3, basically consist of instruction fetch. So, what happens in step 1? The program counter value is loaded into the bus. That bus basically or the program counter is read into memory address register. Memory is given in read mode and then actually in these 3 steps, ALU actually adds program counter value which is now in the bus with a constant value, which is the next memory location and then it is again waited to be loaded into the program counter in the next iteration.

So, in the next step basically what happens we means the accumulator, the ALU basically now holds the value of $PC = PC + 1$, which we dump basically to the program counter in the second step also in the second step basically we wait till the memory says that I have dumped the values which or the instruction, which was available in the memory location or pointed by the PC to the memory data register or the memory buffer register.

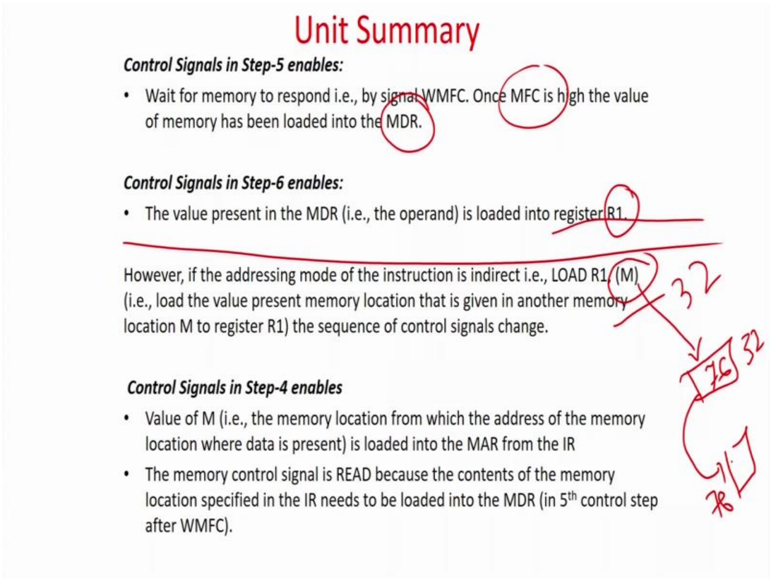
(Refer Slide Time: 02:47)



In step 3, basically the memory tells that it is ready and the memory data register now has the instruction which is now loaded because the memory is ready then actually this memory data register transfers the instruction to the instruction register via the bus. So, now, the instruction register has the instruction which is to be executed. So, the first 3 steps are similar for all type of instructions because they corresponds to instruction fetch.

Now, from 4 onwards depending on different addressing modes or the instruction types they will vary. So, for example, like for example, if the instruction is a direct instruction that is you have to read something from the read or write, something from the memory location. Then in the fourth stage, what is going to happen basically, you are going to take from the instruction register the value of M which corresponds to the memory location for an instruction because it's a direct instruction. So, the value of the operand will be present in memory location M which will be specified in the instruction itself.

So, in step 4 from the instruction register, you will get the value of M. So, instruction may be something like opcode may be some register R_1 etcetera and this memory location. So, the value of memory location will be taken from the instruction register and it should be loaded into the basically memory address register. So, memory address register will now have the value of M you will have to set the memory to read mode and in the next instruction, (Refer Slide Time: 04:06).



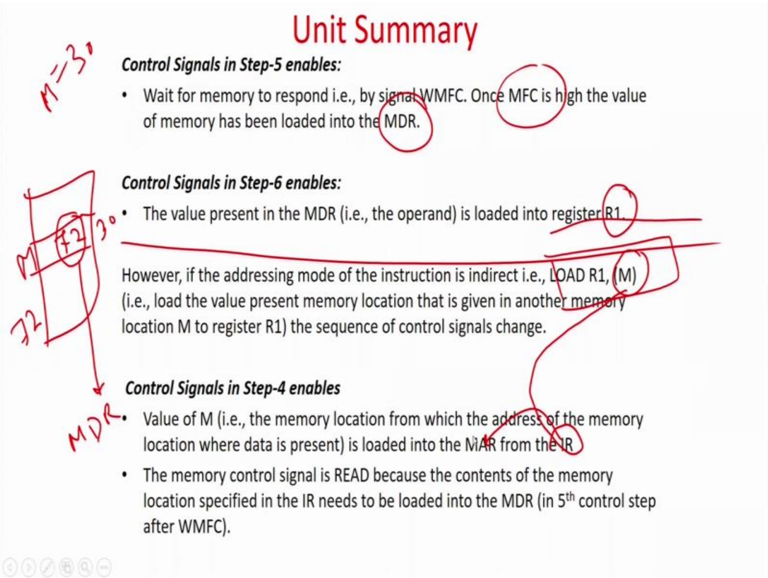
Next cycle basically what happens, you will wait for the memory to be ready. So, once the memory is ready the value of M that is the operand in M will be loaded into the memory data register and in the last that is the last stage, in the sixth stage we will actually transfer the value of memory data register, which is now the operand to some register or to accumulator or to any other point, where you want to require where you want it to be. Basically in fact, these 6 steps are required in case of a direct mode of instruction, direct mode of addressing and if the value is available or the operand is available in the memory location M, but in case say for example, if it is a indirect instruction sorry if it is, in this case, it is an indirect mode where the memory location M actually points to another memory location basically which has the operand.

So, basically it’s an address to another location may be say for example, we say that we know that what do you mean by direct instruction. So, maybe we are having may be M = 32. So, will point to memory location number 32, inside 32 may be the value is say 76. So, basically in the memory location whose address is 76 we are going to get the real operator. We know that is an indirect mode of operation or addressing. So, in this case actually slightly the number of steps from 4 onwards is basically going to change after 3, basically till 3 is fetch and after that it will depend on the addressing mode or the instruction type.

Like for example, if it is an immediate mode where the value is directly available in the instruction itself basically, we can just fetch, we also see in terms of examples the basic first 3 steps will be there and then basically in the fourth stage, basically we, it is a very rudimentary operation in which case you just transfer the value basically it will be opcode say R_1 and we have 7 hex which is the immediate value. So, in this case from up to step 3, everything will be similar in step 4 just you transfer the value of 7 to R_1 that is you are going to load the value of 7 from the instruction register to register R_1 via a bus.

So, there will be nothing called step 5, but now in indirect mode easiest way to shortest number of micro instructions required to execute an instruction is the immediate mode where the operand is available in the instruction register itself. Now, it’s a direct. So, you require till step 6 in which you case you get the operand from memory location 7.

(Refer Slide Time: 06:32)



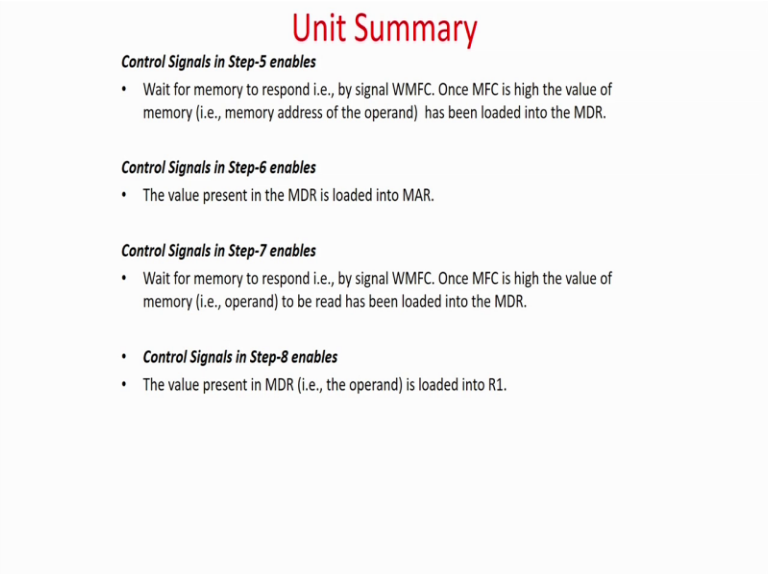
But in indirect one you have to have two memory location access. So, basically what happens in step 4, what you do basically you load the value of M, because this is your instruction you load the value of M to the from the instruction register to the memory address register. So, if you look at, this is your memory. So, let us assume that this is your memory location M say it has the value of 72, assume we can assume M = 30 say for example.

So, basically this is address of the memory location address is 30, which is equal to M. So, from the instruction register the value of 30 will be given to the memory address register, and the memory will be made in a read mode, and then you have to wait till the memory says that I am ready and it is going to dump the value of 72 into the memory data register. So, this is step 5.

So, unlike in the previous case the 72 itself is the operand, but in this case it is the indirect mode. So, in this case it will not be the in this case it is not going to be the operand itself, but again from 72, basically you have to get the value of 72 in the memory data register and then you have to again look at the memory location number 72 where exactly the operand will be present.

So, in this case in step 4 you load the value of M that is 30 in our example let and it will go to the memory instruction register from the instruction register the value of 30 will go to the memory address register and then you have to wait.

(Refer Slide Time: 07:55)



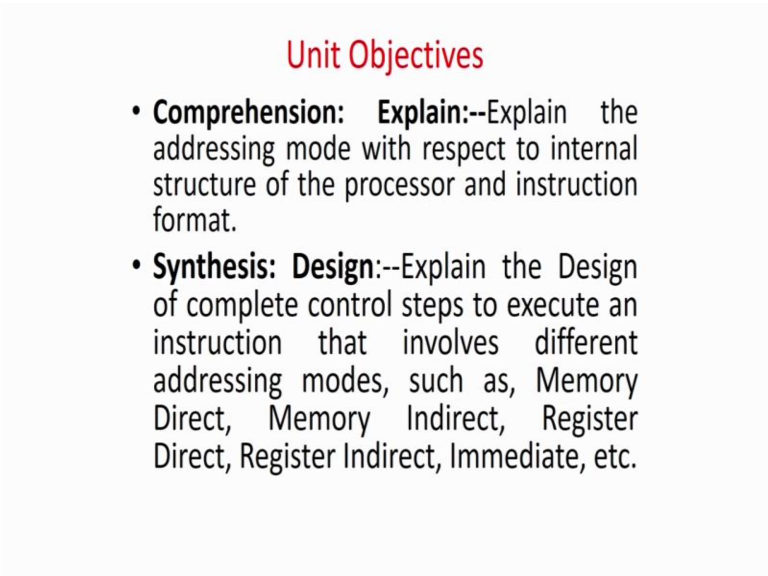
So, after you wait in step 5 what happens, you are it is saying that the instruction is ready to execute I mean, so, the memory is ready. So, whenever the memory is ready. So, whenever we say that the memory is ready that is by WMFC signal. So, once the MFC is ready, so what happens the value of the memory that is in our example basically if you look at our example, we are considering 70, the 70 will be dumped to the memory data register.

So, if it is a direct mode we actually end we can end in this step itself or just one more step will be required to transfer the data from the memory data register to the other register or whatever you want to do, but now in this case what happens now in this case if you look at, we require some more steps now again the value of 72 which is in the memory register has to be again loaded into the memory address register, because this 72 has to be again looked into the memory, that is the 72 will be loaded into the memory the address register and from the memory address register you will look at this second memory location that is 72 in the example and there you can get the operand.

So, very easily you can see in step 6, what happens the memory address register data register basically in this example it is 72, will be loaded to the memory address register. So, we are going an indirect search and after that it is similar 7 step says that the memory has to wait, because now the memory address register has the value of 72 and so, whatever is available in the memory location 72 will be loaded to the memory buffer register where you have to wait for some amount of time which is actually step 7 and once memory is ready it will give a signal WFMC; that means, the value of whatever is contained in memory location 72 that is the real operand will be loaded into the memory data register and then you can load it to R_1.

So, in indirect mode of operation you require 7 steps, in the direct mode of operation, you require 6 steps in immediate mode, you just require 4 steps to do this operation. So, in a summary we have discussed how different type of modes, actually take different type of control instruction that is going to be discussed elaborately with different examples in this unit.

(Refer Slide Time: 09:56)



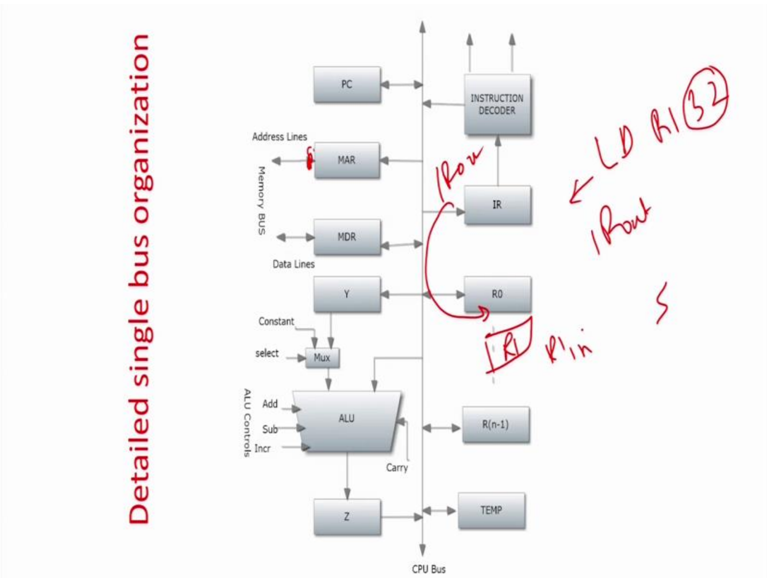
And if we look at what are the different types of basically objectives we are going to support in this unit. So, if you look at we have the first is a comprehension objective in which case we will be able to explain the different addressing modes with respect to internal structure of the processor and instruction format.

That is given an instruction or addressing mode, you will be able to tell what are the micro instructions and what are the formats and then you will be able to design complete control steps to execute those instructions for different addressing modes like memory direct, memory indirect, register direct, register indirect, immediate etcetera with respect to a single bus architecture.

So, basically the objective of this unit, that is that after doing the unit you will be given a single bus architecture mainly you will be able to design instructions based on the or you can be able to design instructions in exact terms of the control steps or the macro micro operations and the control signals which will be required in different addressing modes mainly like direct, indirect, immediate etcetera.

So, this is again the recap of the single bus organization, I am not going to spend more time on this.

(Refer Slide Time: 11:03)

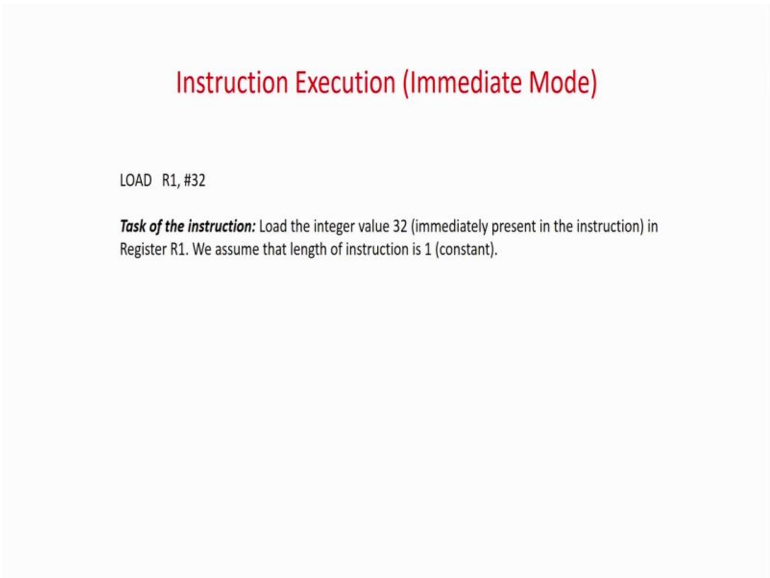


Basically already we have seen yesterday, when as I am just giving a recap because we are going to go for extending the instruction, micro instruction and the control signals based on the single bus architecture. So, if you remember we have a program counter we have memory address register. In fact, this is a unidirectional bus. So, there was a slight I mean a small error which we had taken thought of it.

So, just yesterday, we have to just this one and this is the unidirectional bus which goes to the output. So, in this case, there is a slight mistake and then basically we have a memory addresses data register which is taking the data from the memory then you have this ALU, which is having a Y as the input from the bus, multiplexer which takes a constant as the input which is required to increment the program counter output of the ALU is stored in a temporary register called Z.

This is your register bank, instruction register and directly go into the instruction decoder which generates the instructions control signals for the instruction basically this is the structure we have explained in details in the last unit.

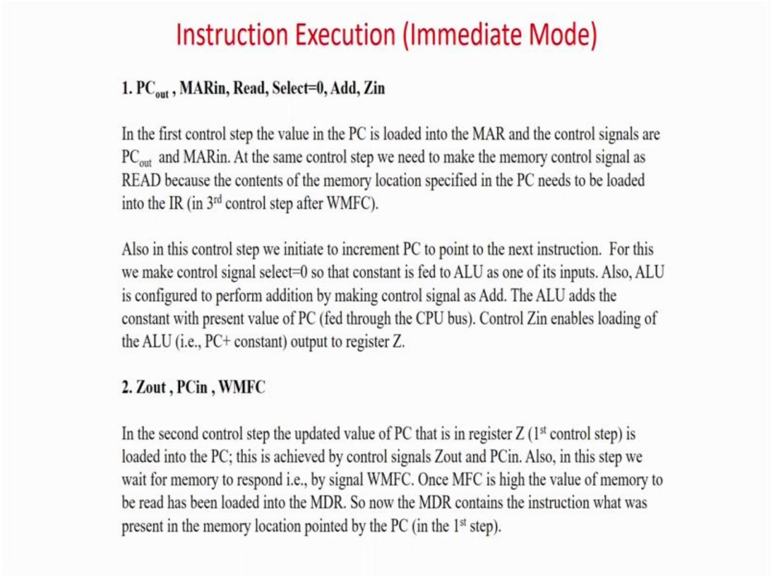
(Refer Slide Time: 12:04)



So, now first we are going to give an example for a immediate mode of instruction. So, just we have to just keep we remember that whenever we say that PC_{in} the value is going to the PC and whenever we say the PC_{out} the value is going from the program counter to the bus and vice versa.

So, first instruction mode the simplest one is the immediate mode. So, we are just saying LOAD R_1, 32; that means, the value of 32 constant will be loaded to R_1 and we are assuming that this instruction takes a single word. So, in this case the PC has to be incremented by 1.

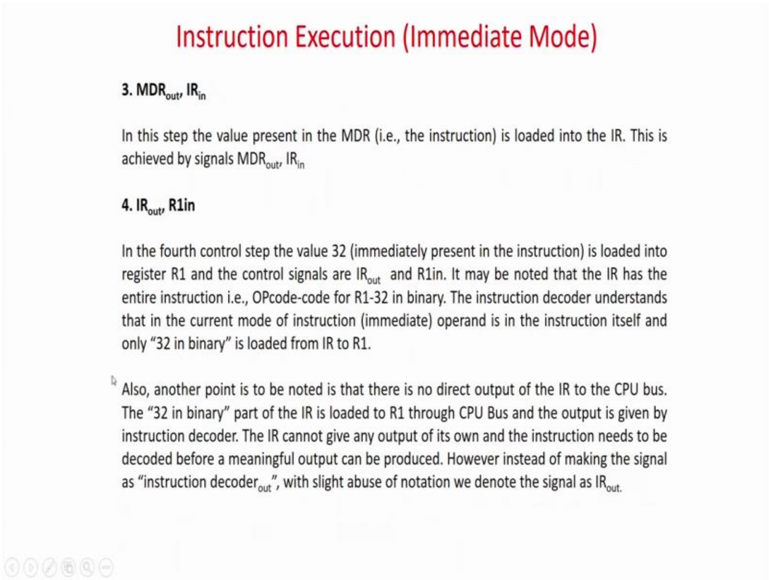
(Refer Slide Time: 12:37)



So, what are the steps? So, the first step if you look you know the program counter will be the output it will go to memory address in. So, what does it mean already we have discussed with the figure in the last class. So, this I am going to keep a bit brief. So, PC out and memory register addressing; that means, what the output of the program counter will be loaded to the memory address, will be in a read mode select will be 0; add and Z_{in}. So, up to this PC out, memory register address in and read this corresponds to the fetch part select zero; that means, you are going to add the constant to the program counter and in the output will be of the ALU will be going to register Z.

This corresponds to increment of the PC which we have already seen in details in the last unit then in the second stage Z_{out}; that means, the output of the ALU that is Z which is now equal to $PC = PC + constant$ will be going to PC_{in}; that means, the output that is $PC = PC + constant$ will be noted into the PC via a bus and we are waiting till the memory sends the signal that I am ready.

(Refer Slide Time: 13:33)

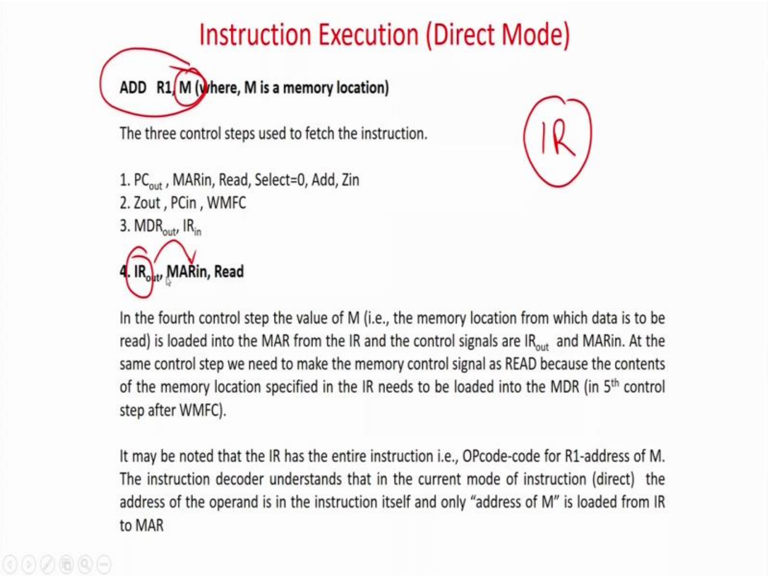


So, once it is ready the output of the memory data register, which now has the instruction that is equal to $LOAD\ R_1, 32$ will be loaded into the instruction register. So, this up to these 3 stages as we have already seen basically corresponds to fetch. Now as it is the immediate instruction we just require just one more control part that is in the instruction register we now have the value called $LOAD\ R_1, 32$.

So, that value of 32 will have to be taken from the instruction register and dumped into register 1. So, very obvious signal will be R instruction register R_{out} and R_1in that is as simple as taking the step if you look at it this. Now basically you have loaded your instruction that is some $LOAD\ R_1, 32$ this is already it is present in your instruction register. So, this part has to be extracted that is nothing but equal to IR_{out} and it has to be loaded in R_1. So, this is may be this register is called R_1.

So, it will be R_1 in an IR. So, that is done basically. So, it will give the value. So, there are basically 3 steps involved in this case extremely simple, that is fetch and after this just you require one stage in which case, the value of the instruction register immediate part will go over here. So, basically for an immediate mode we just have 4 micro instructions and your actually 4 micro instructions and your job is done. First micro instructions or sequence of control fetch and last one you take the value of the operand from the instruction register and dump it into the respective register.

(Refer Slide Time: 15:06)

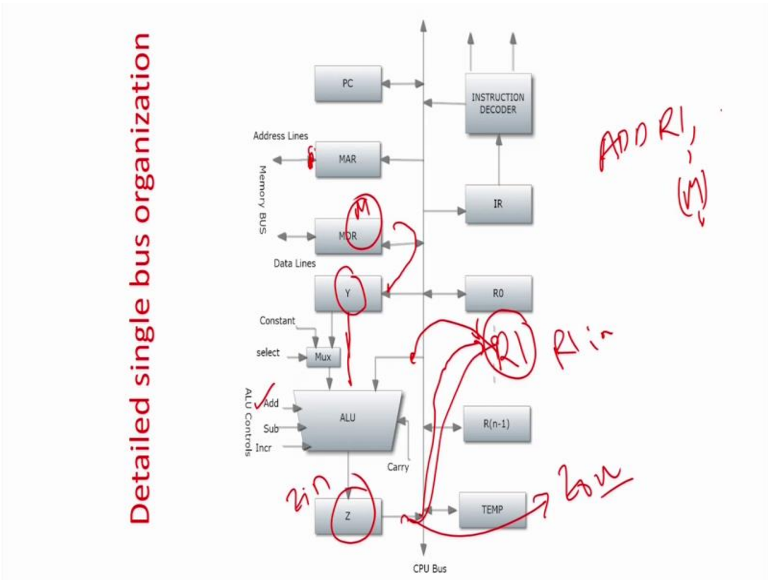


Then next is the direct mode. Direct mode already also we have discussed yesterday, but in this case you are going to just see a comparison that what is the more complexity involved. So, in this case we will see that we require 6 stages, or 6 micro instruction in sequence. So, the first three like program counter value to memory address, read select an increment of PC and this 3 basically and MDR_{out} to register in, these three instruction or three controls basically corresponds to instruction fetch. So, now, what the instruction has been fetched. So, you will have the value $ADD\ R_1, M$ this will be in your in register which is your actually nothing but your instruction register. So, this one will be available in your instruction register.

So, from here I have to extract the value of M and I have to put it into the memory address so that I can get the operand from there. So, what I am doing I am taking instruction register value out and I am making MAR_{in}. So, basically it will take the value of M which is your memory location basically to your. So, what will happen? So, in this fourth stage basically what is going to happen, you are going to take the value of M and you are dumping into the memory address register so, that you can read the value of the operand. So, if you look at it in the bus architecture.

So, what it's simply what you are doing, in the fourth stage the instruction value is say add or load whatever R_1 in to memory location 32.

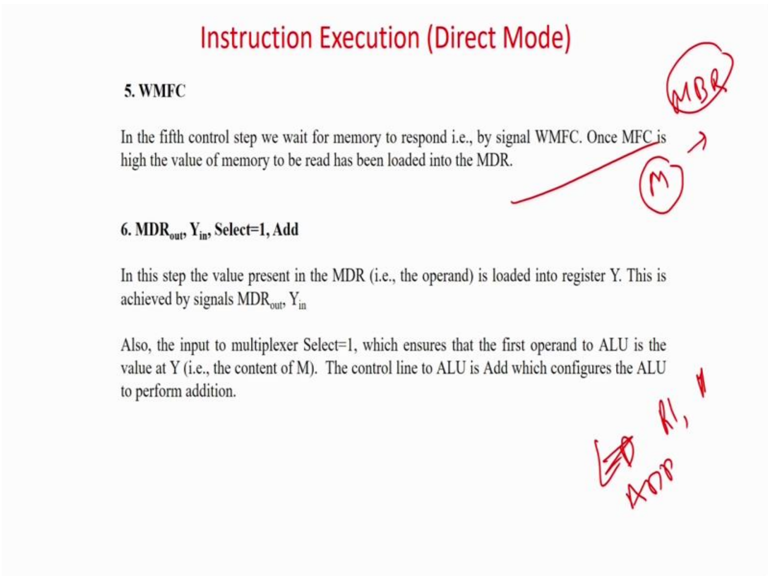
(Refer Slide Time: 16:21)



So, this 32 I am taking it from the instruction register by IR_{out} and I am putting into this memory address register. So, that the memory location value 32 whatever is in this can be com come as an operand and it will go to register R_1 or wherever you want to dump that is again you want to read this value. So, this value of 32 from the IR is going to dump to memory address register.

So, IR_{out} $memory\ address\ register_{in}$. So, that is what I am doing make the memory to a read mode.

(Refer Slide Time: 16:54)

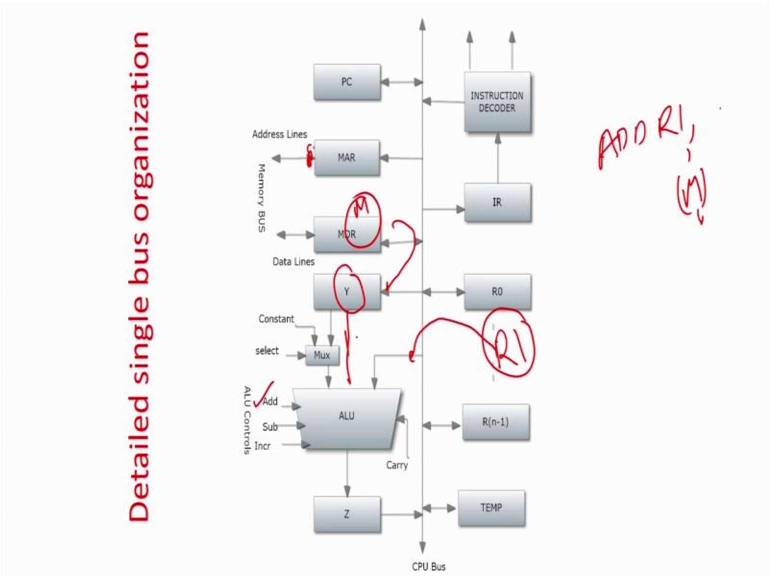


In fifth stage you have to wait till everything is ready. So, once it is ready; that means, the memory data register now has the value of this instruction $ADD\ R_1, M\ or\ LOAD\ R_1, M$ whatever may be the case, but is a direct mode of addressing. So, now, after this one basically you are having the value of, after this after the fourth stage and the fifth stage when memory has been read. So, the value of value present in memory location M is now dumped to basically your memory buffer register, that is done after 5 says that the memory has memory reading is complete.

After that what you are going to do, you are going to in this case add. So, in my case in this case specific example we are going to add had it been load we could have just transferred the value of memory buffer register to the corresponding register R_1. So, having had the instruction being $LOAD\ R_1, M$ so you could have done very simply that after this in 6 stage we could have made memory data register out and we could have made R_1 in, but in this case instead of load if is an add.

So, I have to do an add operation. So, again slightly the number of steps will be the different of the control sequence will be different. So, what I have to do. First thing is that I have to let me look at the structure again. So, now, if you look at it so, now, the value of M has been loaded or the operand which is present in M is present over here the content of M is available over here.

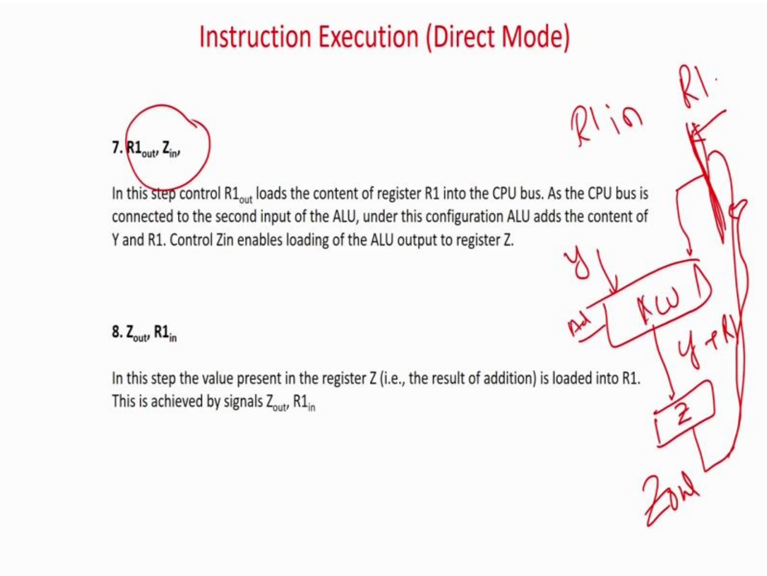
(Refer Slide Time: 18:12)



So, now the instruction was add R_1 and it is M. So, the content of M basically is now in the memory data register. So, now, I have to add it. So, for that I require the ALU. So, I have to first one control instruction will be enabling the add. Secondly, the value of M has to be put in the value of Y, because at the next stage I am going to use the value of R_1 which will directly feed over this operand. So, that I can add, so now, in this step what you are going to do you are going to take the value of MDR_{out} and you are going to dump it at Y. So, the value of M will be dumped at Y, I will make add = 1 and I will make this select = 1. So, that the value of M via Y can be coming to the ALU.

So, you just see now this step. So, it is saying memory data register out Y in. So, that I can put the value of memory data register that is M, or the content of M into Y select = 1 and add. So, that constant is now not required the value of MDR that is the content of M will be invite and it will be added.

(Refer Slide Time: 19:20)



Next basically the adding is done we are sorry the adding is not yet done basically what I have to do is that in next stage I have to make R_{out}. So, as I already told you, this is the stage R_{out} Z_{in}; that means, now one operand to the ALU, this is one operand to the ALU directly coming from Y that is your content of M and this is your bus and you are making R_1, you are making R_1out. So, the value of R_1 is the second operand to the ALU and you are making it as add mode fine and the output of this is going to the register called Z.

So, I am making Z end. So, in this stage basically your adding is done, one operand from memory location M was stored in Y and is one operand to the ALU verses via Y and the other operand is from register R_1 which I am directly taking via the bus is your single bus by making R_1out. So, Y equal to. So, now, the ALU is having the value Y which is actually the content of M plus register R_1. So, it is stored in Z in next stage is very same basically what happens in the next stage again I have to write the value of the sum back to R_1.

So, what I am going to do now I will make R out. So, sorry I will going to make z out. So, if I make Z_{out} the value of Y plus R_1 that is the content of Y plus register R_1 would be dumped to the bus and I will make R_1in. So, if I make R_1in. So, the value will be fed from Z_{out} to register R_1 and your job is done which is basically step 8. So, if you see in a direct mode had it been just the load instruction, that is load R_1 accumulator we could have done in 6 stages, but as this is an add instruction. So, you require 8 stage in the direct mode. So, depending on the type of instruction the job or the functionality of the instruction as well as the addressing mode the number of steps and sequences get change.

Last two steps basically again let us quickly look at in this example. So, basically till now, what has been done? So, the Y is over M, the R_1via R_{out} here and Z_{in} is there. So, the value of R_1 plus the value of content of M by via Y is now in Z_{in}, last stage is basically we make the Z_{out}. So, the value of Z is dumped over here and it is going to R_1. So, we are going to make R_1in. So, in that case the value of Z by virtue of signal Z_{out} will be dumped to register R_1. So, Z_out = 1 and $R_{in} = 1$ will dump the value of Y, then it is actually the content of M plus content of R_1 into R_1 itself. So, that is the basic path. So, this, what has been explained.