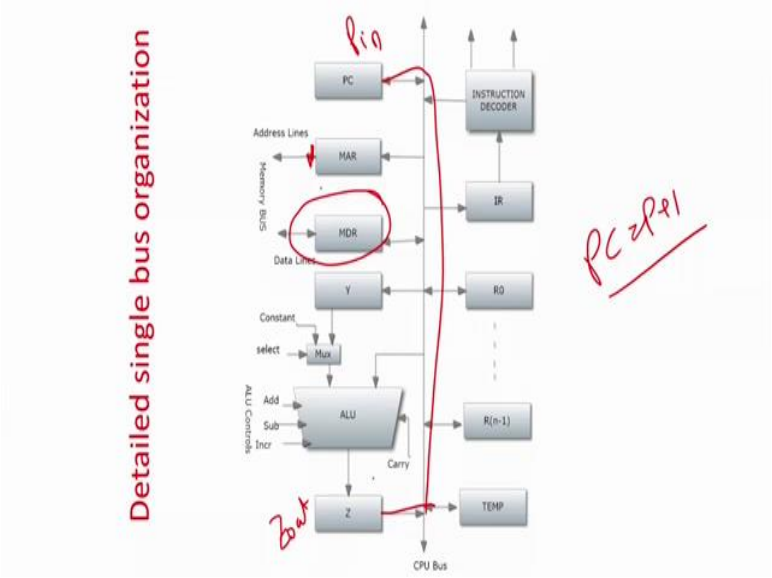
Let us again clean it up, because we will have to revisit this figure many times. So, again I am cleaning it up. So, next is what? Now what now actually next stage is till now we have seen, that the output of this $PC = PC + 1$ is memory is in register IR and memory has you have given the command to read the memory. In second stage what we do? so whatever I told you about the first one is written over here, you can read it now what is it says Z_{out} PC_{in}. So now, what this IR has, if you look at the initial last slide then IR had the value of $PC = PC + 1$, but at that time it was Z_{in}.

Now, I am making as Z_{out} and PC_{in}; that means, the value of IR will go to PC program counter, via the bus because Z_{out} and PC_{in} and we are waiting for $WFMC$ so are waiting till the memory says that, I am ready and whatever you asked in the first stage it has been dumped to the memory buffer register in fact, again revisiting. So, in this stage what I am doing? You are making Z_{out}. So, the value of IR is over.

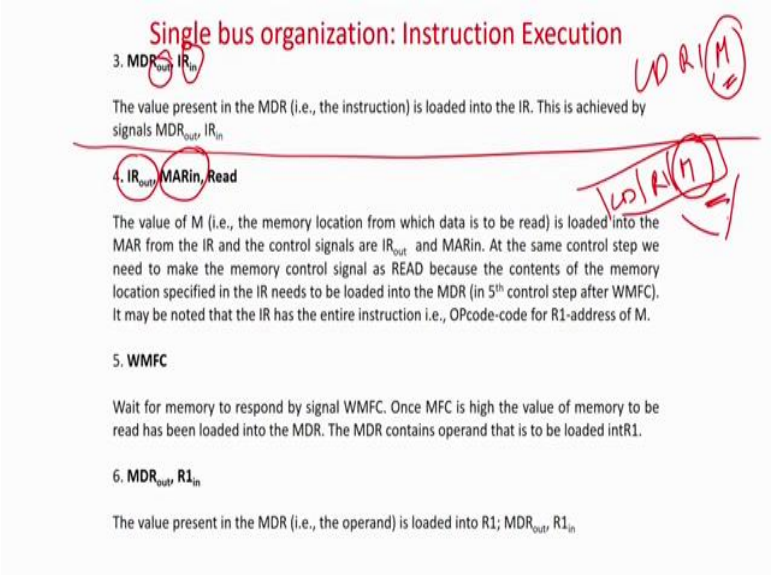
(Refer Slide Time: 26:04)



So, this is Z_{out} and PC is now becoming PC_{in}. So, the incremented value of PC is going to this 1 by this path. So, $PC = PC + 1$ or the constant, is loaded into the PC and also I am waiting for $WFMC$; that means, if the signal is one; that means, what the value of the memory location, where the instruction was there is loaded into the memory data register or the memory buffer register, and now you can read the instruction to the instruction register.

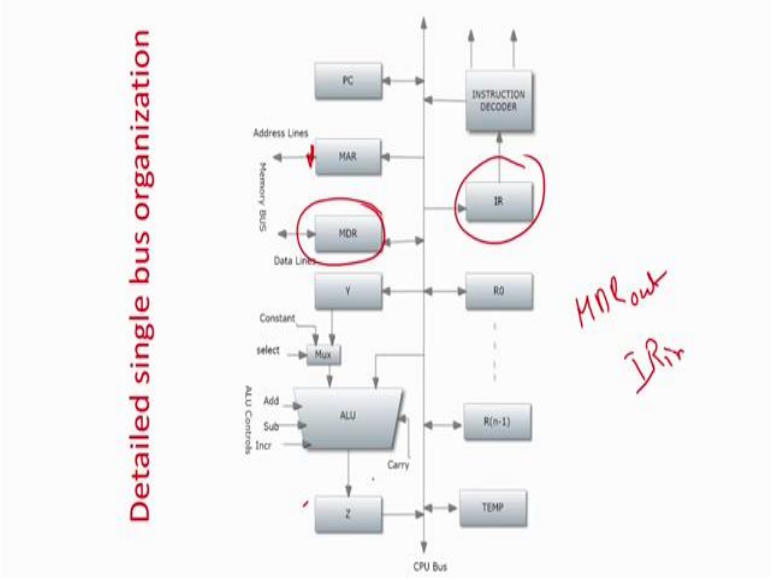
So, in the second stage PC_{in}, this data will be read from this memory this IR, which actually nothing but $PC = PC + 1$ and it will be read to the PC by this bus. So, the 2 signal Z_{out} and PC_{in} accomplishes that and we are waiting till basically, our memory is ready. So, that is over here.

(Refer Slide Time: 26:56)



After that what happens? Now the memory is ready, now what you have to do you have to load it into load it, load the value of this instruction into the instruction register, very simple you will make memory data register out and register in as simple as that just have a very quick look. So now, your instruction is over here, you have to load it to the instruction register.

(Refer Slide Time: 27:18)



What will you do very simple MDR_{out} and IR_{in} it will serve the purpose basically, what is being done in the 4th instruction, 4th step these are the 2 control signals which is generated in the 4th stage, this one then again what?

Now, what is you instruction, that is 1, 2 and 3 will be same for all instructions, you know that is instruction fetch. Now instruction has been fetched, it is in the instruction register now you have to tell what I have to do. So, what was the instruction the instruction was basically, load $R1$, M that is whatever is present in the memory location that is $LOAD\ R1, M$; that means, in memory location M whatever value is, there data is there you have to load it into $R1$.

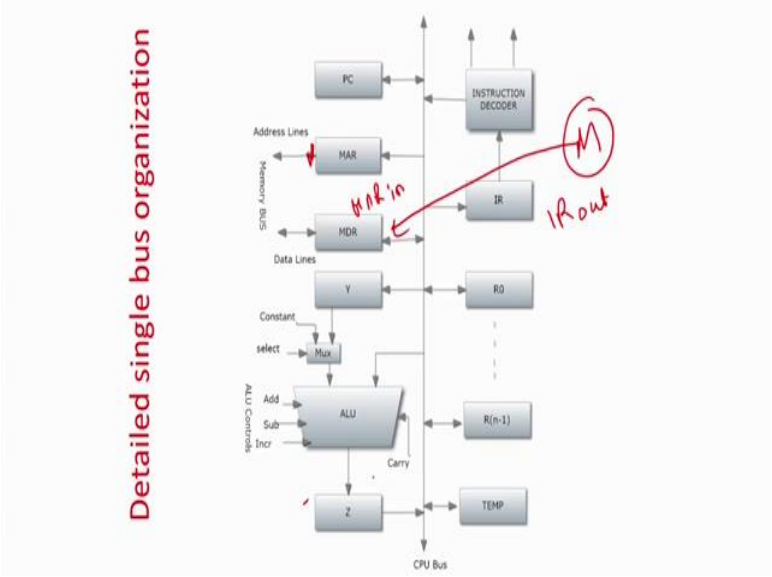
So now, what? So, you have to take this value M and loaded it into the memory address register, because that part is going to tell where the operand exists so obviously, first instruction will be IR_{out} because the value of the instruction, which is present in the instruction register has to be given into the bus, and then your memory address register basically, we read the value from M, now what there is some subtle thing over here, we are not and memory has to be in read mode of course, but there is slight thing which we have skipped over here, that is the whole instruction register will have load then $R1$ and M.

We need to load only this part of the instruction IR to the memory address register because, we don't require the opcode we don't require the address for $R1$. In fact, the instruction decoder will take care of that. So, that part we have slightly obstructed, because we just required to keep this part, which is a very simple digital operation but only this part has to be loaded into the memory address register. So, we will do that and then, again wait till your memory read operation is complete. So, you can see the $R1$ has the entire instruction opcode $R1$ address and address for M, that is the 3 parts basically as I have told you, but in fact, you are going to only use this part, to be loaded into the instruction memory address register. So, that clipping part we have dropped over here.

Then at 5th stage, we wait till the memory says that, I am done with it. So, once the memory says that I am done with it; that means, the data this M data is now loaded in to the memory data register. So now, what you will have to do you have to just dump the memory data register value that is $MDR_{in}$to R_{in} that is $R1$. So, MDR_{out} means whatever data is available in the memory data register, it will out and it will be present in $R1$. So, in 6 stages I complete the instruction, let us quickly look at the three controls in this figure again. So now, what happened

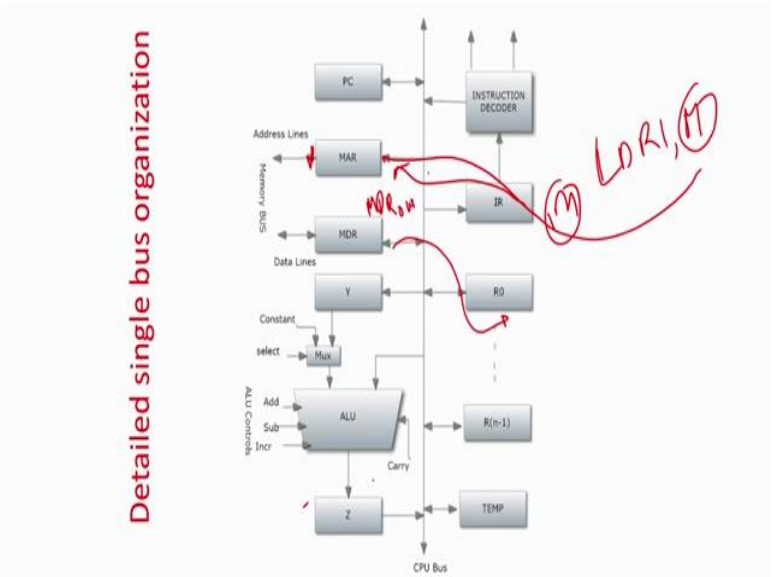
the instruction decoder via instruction register will load the value of M. So, this one will go to the memory data register.

(Refer Slide Time: 29:55)



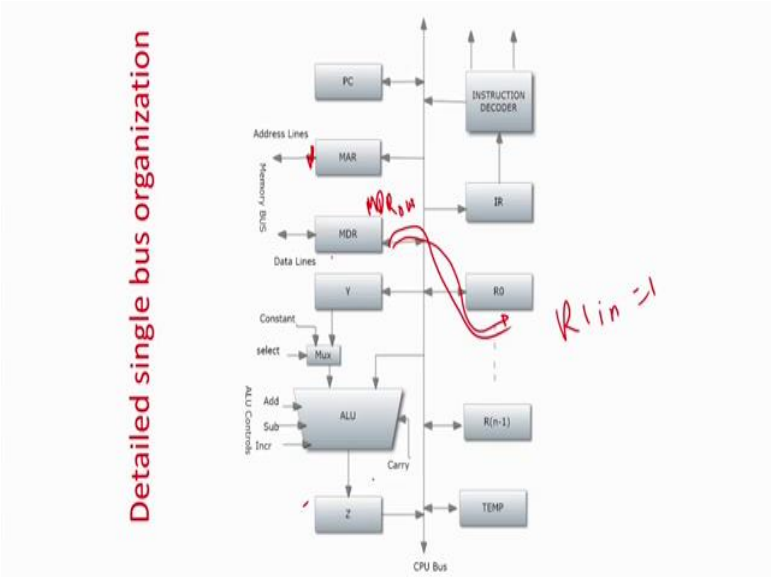
So, of course, it is IR_{out} with slight abuse of notation, which I am not going to take the entire IR only the M part of IR and MDR_{in} and memory is in READ mode, wait for some amount of time that is stage 5 and then once it is done, the data has come sorry the just a small mistake the M part of this one, has gone to the memory address register not the data register that is $LOAD\ R1, M$.

(Refer Slide Time: 30:19)



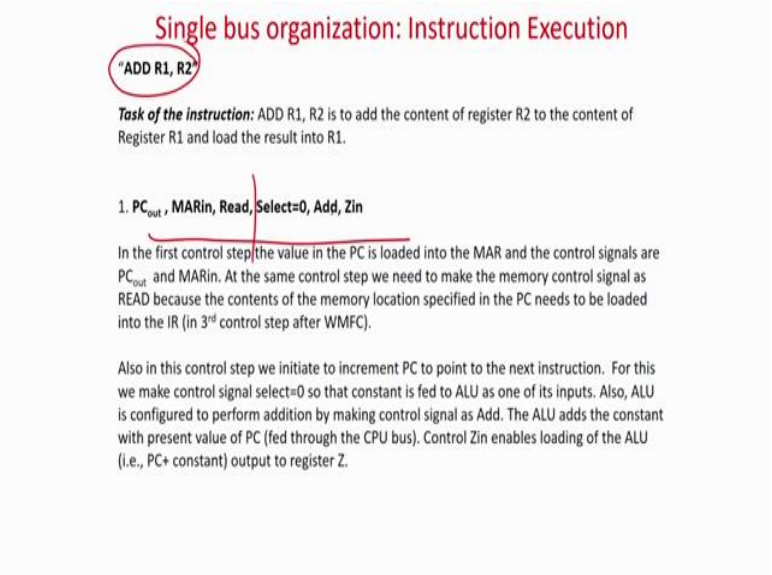
So, this M part actually goes to memory address register via the instruction register. So, IR_{out} and MAR_{in} you wait for some time, which is signal number stage number 5 after that the value has come over memory data register, now it has go to it has go to register $R1$. So, as simple as that now we are going to make an MDR_{out} of course, all these things are now have become 0 that all these things has to go off. So, MDR_{out} will be 1 and also R_{in}, R1in has to be 1.

(Refer Slide Time: 30:53)



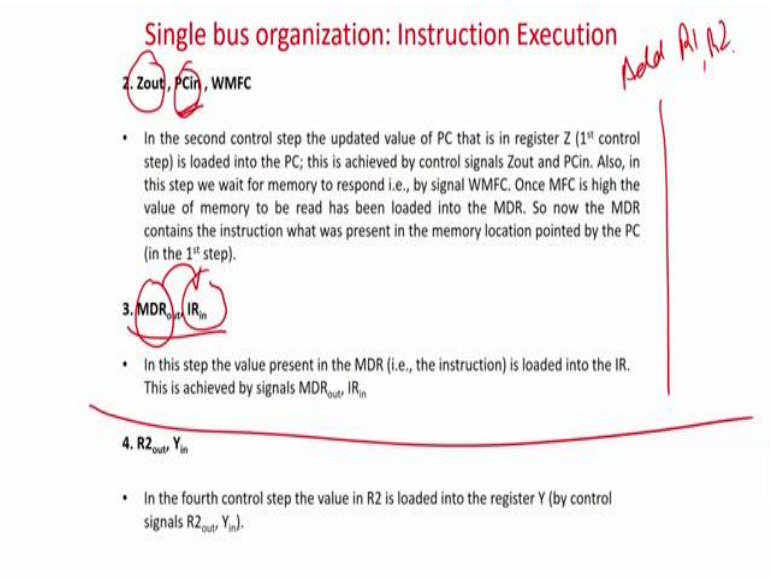
So, in fact this one will go from here to here. So, it is done. So, all the stages are complete. So, last 3 stage basically reads the value of M from the instruction register, writes into the memory address register, waits till the value of M is dumped to the memory data register and then the value MDR_{out}, will take the value from MDR and it will load to $R1$. So, your job is done. So, in 6 micro instructions and the corresponding control signals, what we have done we have shown how a complete instruction is fetched, decoded and executed. So, this was about the instruction $LOAD\ R1, M$.

(Refer Slide Time: 31:29)



As I told you, we will look at different instruction. So, that was just a load instruction. So, in this case, we are going to see another arithmetic operation. So, in this case we are saying $ADD\ R1, R2$; in this case the value of $R1$ will be added to $R2$ and stored in $R2$. Again as I told you in the first stage PC_{out}, $memory\ address\ register_{in}$, $Read, select, Add\ and\ Z_{in}$ this stage.

(Refer Slide Time: 31:50)



Program counter incremented stored in IR, move to PC and wait for $WFMC$, once it is done then MDR_{out} and put in IR that is first stage points to the memory location, where the data is

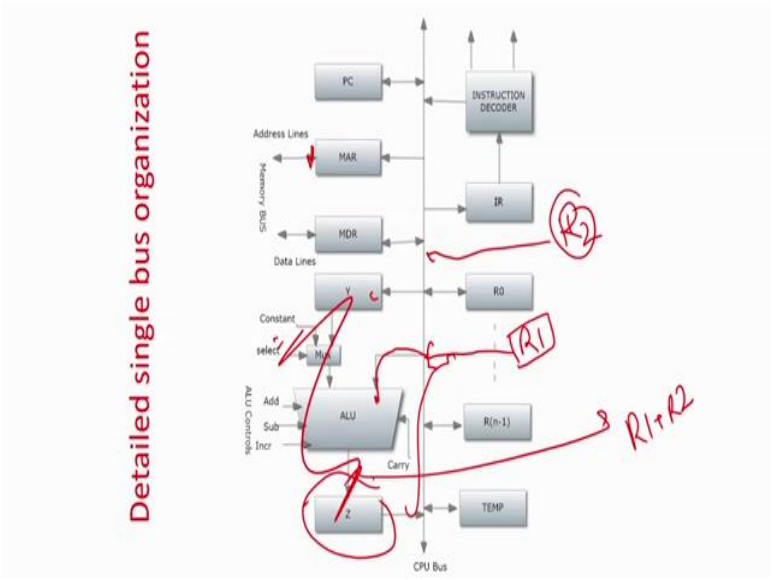
to address is where the instruction is stored, plan to increment the program counter, really increment the program counter by ALU, dump it in PC and wait for some amount of time, once it is done take the instruction from instruction register sorry take the instruction from MDR and put it in.

So, these three stages basically correspond to any instruction fetch, whether it is add $R1$ $R2$, whether it is load $R1$, M1 the first 3 stage, basically this one this one and this one will remain constant for everything. Now, from here based on the addressing mode and the instruction type things will start becoming changed. So, this one is all correspond to fetch. So, no changes over here correspond to compared to the previous case. So now, in this what I am having. So, next instruction is add $R1$ and $R2$. So, it was add $R1$ and $R2$. So now, what is going to happen?

So, how I will do it? Basically there is one operand in $R2$, one operand in $R1$, both has to be added and the value has to stored in $R1$ so. In fact, here there is no question of accessing the memory again. So, unlike the previous case here we will not take any we will not take the instruction from IR and put it into the memory address register and again read the memory not required, because everything is in the registers.

So, basically what is does, it says that $R2_{out}$ and put it in Y_{in}. So, what it is doing taking the value of $R2$ and putting it in the Y, Y let us quickly look at the figure again.

(Refer Slide Time: 33:39)

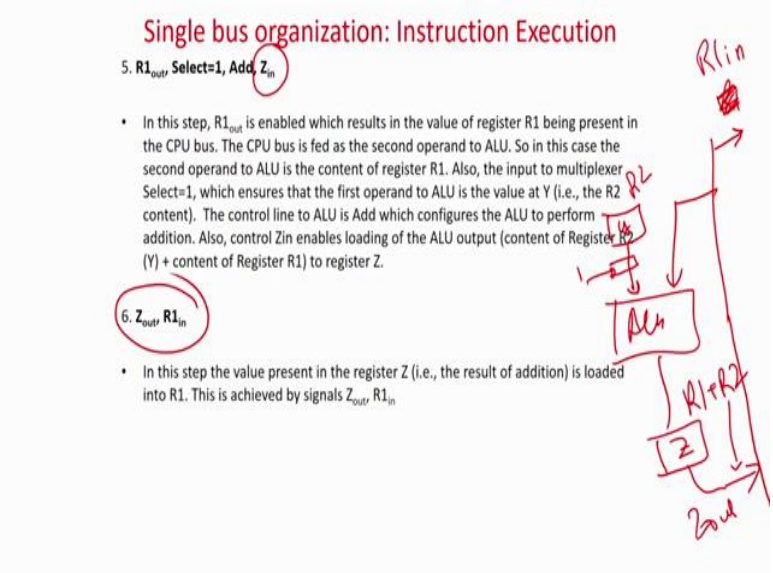


So, what it is doing it is taking the value of $R2$ some $R2$ which is connected over here some register. So, that it is dumping it to Y. So, this value I am putting it over here, now how what will happen basically. So, what I will do that $R2$ value I will first store it into Y, then what I will do then I will take the value of $R1$, which is again another register over here and I will connect it to here. So, first I will what I will do, the value of $R2$ I will store it in Y for a temporary, then instruction Y will have the value of $R2$.

Then next is what I am going to do, next is I am going to just connect it to $R1$. So, that is goes to the ALU directly, and in the mux select will be equal to 1 so, basically sorry. So, select will be equal to 1, if $select = 1$ then $R1$ will be directly fetched to the ALU to this part and Y which, is nothing but in your case in your case $R2$ will be coming to the ALU and; obviously, in add mode.

So, here you will have the value of $R1 + R2$. So, that is how I will do, I will store the value of $R2$ to Y first then next stage I will directly connect $R1$ to ALU by the bus, as simple as that select will be 1. So, I will have the value $R2$ to $R1$ which I have to store in a very temporary manner to register IR and then again, I will dump the value of IR to register $R1$. So, this will be the stage it will go through this is over all idea, now we will individually take the steps and clear 1 at a time, now let us again go step by step.

(Refer Slide Time: 35:10)



So, next is $R2$ Y_{in}. So, already we have told $R2$ means the value of register 2, is dumped in Y in. Next one next already your Y that is your if you look at it this is your ALU and this is your

register Y and it is coming here, by a multiplexer this is your mux and this in fact, is your mux. So, next stage what I have done I have said $R1$ out. So, $R1$ was somewhat connected over here. So, I am making $R1_{out}$. So, the value of $R1$ is going in that direction already we have discussed, but again I am just redrawing.

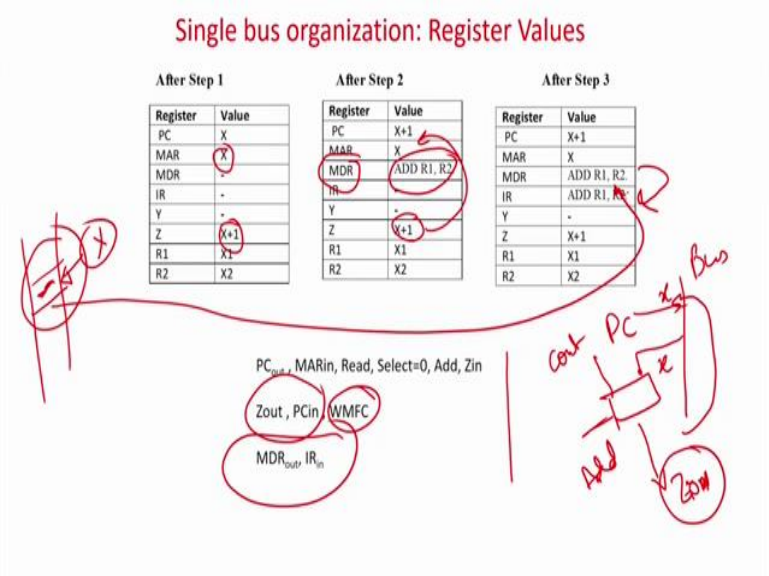
So, value of $R1$ is going over here, which is our next operand which is connected to the ALU and now Y which is having the value of $R2$, previously it is actually temporarily holding the value of $R2$, now this one you have mux you have made it one. So, it is being fed over here. So now, it is your doing $R1 + R2$ that you are going to add now, where I am going store as you already seen that we have Z_{in}. So, Z_{in} will actually temporarily store the value of $R1 + R2$.

Now, what next stage the IR is also connected over here in this 6th stage you will make Z_{out} the value of Z_{out} will come over here and now $R1$ will be $R1$ in. So now, this will no longer be in this mode it will be going in 6th stage that is will be going in the reverse mode. So, it will be R_{in} and it will be Z_{out}.

So, value of $R1$ and $R2$ via IR will go to R_{in} and your job is done. So, in the 6th stage IR will be out and $R1$ will be taking in. And so, these are the 6 signals, which I have taken here add $R1$ and $R2$, if you compare load from a memory location and load from $R1$ and $R2$, this is a register mode instruction and that involved a memory mode. So, there was a memory.

So, in the second stage also you have to load the memory, you have to in first stage you fetch the instruction from the memory in the second stage actually you have get you have get the operand from memory location M, but in this case what happened we have never gone to memory for the second time, because everything was available in the registers right?

(Refer Slide Time: 37:02)



So now let us quickly have a look at, what are the different register values while we are executing this instruction. So, first instruction like as I told you, the first 3 instructions that is $program_{out}, memory\ address_{in}, Read, select\ 0, Add\ and\ Z_{in}$ then again the value of Z_{out} is going to the program counter, wait for some time and the memory data register is going to the register in; that means, first one corresponds to reading the instruction from the memory, as well as setting to increment the program counter.

This one actually increment the program counter and 3rd one actually reads the value of the instruction from the memory to the instruction register, this corresponds to instruction fetch they are same for everything, let us see step 1 2 3 what are the values of the registers?

So, let us assume that the program counter has the value of X, X will be 1 2 3 4 depending on which position of the program you are in so. In fact, what happens. So, program counter I am assuming the value of X. So, you are dumping the value of program counter to memory address register. So, what is going to happen? So, if the value of program counter is X, the memory address will also have the memory address register will also have the value of X memory is in READ mode, select mode, nothing to do and you are in ADD mode. So, what is going to happen?

If you look at the architecture this was you main bus, PC you are dumping over here, which was which has the value of X and this is your ALU. So, X is going over here and you are making it ADD mode, correct is it an add mode and you are saying that this select is equal to 0, select

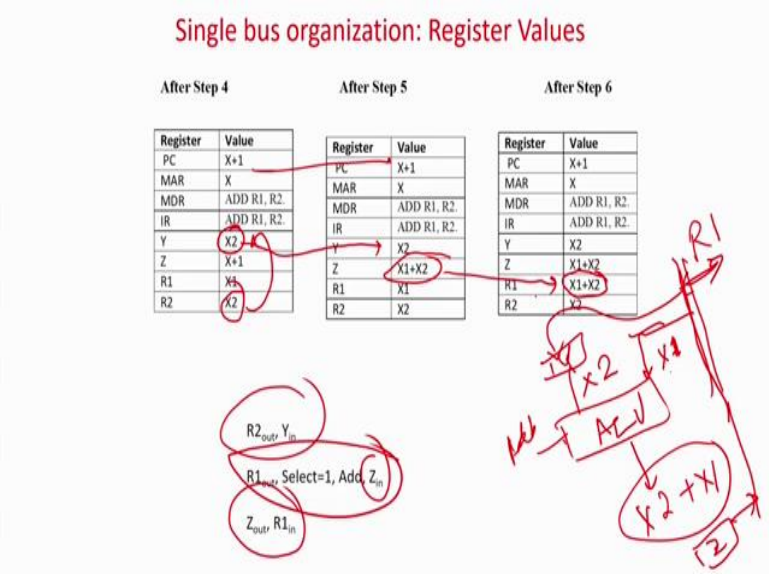
is equal to 0 means, you are going to add a constant. So, it will be constant plus X. So, in this case we have said that and the output is also to Z_{in}.

If it is Z_{in} so, $Z1$ will be written by the value of the output of the arithmetic logic unit. So, in this case we are assuming the constant to be equal to 1. So, you are going to get $IR = X + 1$, that is PC is incremented by one and we are assuming that, register $R1$ and $R2$ are having the value of $X1$ and $X2$, which will be 5 6 7 10 11 14 whatever which one we are going to add. So, after the first stage you can see the value of program counter, which was X is incremented and memory address register, has the value X that means, now in the next stage basically, the program the memory location X will have the instruction will be loaded to the instruction register.

Now next what happens next stage you can see $Z_{out}\ PC_{in}$; that means, the value of IR will be now out, and it will actually write the program counter in a reverse manner. So, of course, the value of IR will be written to the program counter $PC = PC + 1$ and also, we are waiting for $WFMC$ that is you have to wait until the memory saturated, saying that the data has been peacefully read from the memory to the memory data register. So, the instructions $ADD\ R1, R2$ was available in the memory. So, if you look at the memory. So, here the instruction was there, that is $ADD\ R1\ R2$ and this was actually memory location M.

So now it has come over here, that is in the second stage and finally, in the third stage basically where you are saying MDR_{out} register in; that means, memory data register value will actually equal to the instruction register; that means, now what we have done in 3 stages this memory location M basically. In fact, it should not be M basically it is X I am sorry, memory location X we are making it X this memory location X which is having the instruction, add $R1\ R2$ is being fetched memory location X and it is dumped to the instruction register. So, instruction has been fetched now we are looking to the next slide.

(Refer Slide Time: 40:42)



Now, you are going to go to the ADD. So, this was the stage so, next what you are doing? You are saying that $R_{out} = Y$ in already we have seen that, this is your ALU and this in your $R2$, $R2$ should be fetched to Y means a temporary register over here. So, I have here the mux will be set in such a manner this is Y. So, that it will not take a constant it will take the value of $R2$ as a operand. So, in this second stage you are saying that $R2 = Y$ in. So, $R2$ basically had the value of X, which will be dumped to memory location Y I am sorry register Y. So, you can see this has been dumped over here.

So now, this is actually having the value of $X2$, this is Y and another if you remember, this part of the ALU directly is connected to your bus. So, in the second stage the value of $R2$ is now dumped into Y, which is nothing but $X2$ is 1 operand of the ALU, which is read. Second stage what you are doing here, that is 5th stage in this case, you are saying $R1$ out select 1 ADD and Z_{in}. So, you are saying $R2$ out. So, if you are saying $R2$ out means what sorry $R1$ out; that means what? This is $R1$ which is now feeding the bus.

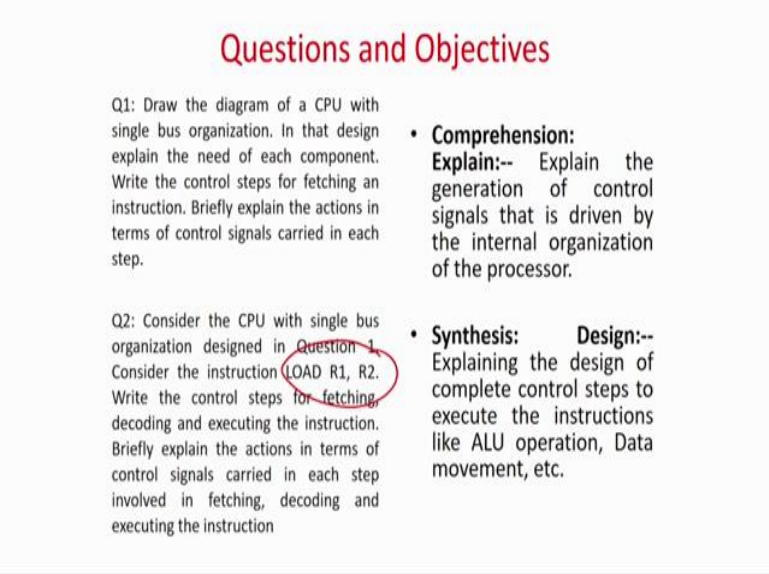
So, it is having the value of $R1$, $R1$ is nothing but your $X1$ over here. So, $R1$ is having the value of $X1$. So, $X1$ is going $X1$ is coming to the ALU, as the 2nd operand. So, this is your $X2$ sorry $R1$ sorry this is $X1$. So, $X1$ is coming over here, correct and ALU already having the data in the other operand $X2$. So, output will be $X2 + X1$, right? There is $R1$ and you are making select equal to 1, once select is made to be 1, the multiplexer will take the value of Y that is $X2$ to the ALU constant will not be added, ADD is the symbol that is going to be add it. So, that is added

now and the output that is $X1$ and $X2$ will be dump to Z_{in} because, where it is connected to a register which is IR. So, it will come over here.

So, you can see PC has been incremented. So, it is continued from here to here, memory address register is X there was no change IR is having the value of $X1 + X2$, Y is temporarily holding the value of $X2$ which is continued over here, and the $R1$ the register $R1$ that is again having the value of $X1$ is feeding the ALU directly from the bus, without any primary without any temporary register last stage what we do, we say $Z_{out} = R1$ that is, now the value of IR has to be the value of IR has to be fed to register $R1$.

So, what you will do you will say, Z_{out} and R_{in}. So, in this case the $R1$ which was initially having the value of 1, is update with the value of $X1$ and $X2$. So, that is this value actually changes and your job is done basically. So, again you can very quick look at this figure. So, I am cleaning it up then it is very simple you can understand the illustration ok.

(Refer Slide Time: 43:26)



So, with this we come to the end of this unit and before we quit, basically we always have some template questions and we see how the objectives have been achieved. So, the first question here is, draw the diagram of a CPU with single bus, in that design you need to explain each component, write down the control steps of fetching an instruction briefly explain the action in terms of control signal in each step.

So, basically if you are taking a single bus, if you are taking a multiple bus, basically the first 3 stages in case of single bus corresponds to fetch and that is constant. So, if you are able to explain this as you have already done in the first few slides, you can actually explain the generation of control signals, that is driven by the internal organization of a processor, mainly in case of single bus architecture. Also, you will be able to design the complete steps, required for a fetch phase of the instruction then, we are saying again take the same bus of this architecture and take the instruction load one $R1$ and $R2$ load $R2$ to $R1$, whatever value is available in $R2$ it will be going to $R1$. So, generate all the control signals and all this steps for this.

In fact, just we have seen one instruction for add $R1$ and $R2$ if you repeat it for load, load instruction load again if you are; obviously, able to design this, you can easily explain the generation of control signals for any kind of bus, here we are asking for single bus you can try for 2 bus or 3 bus architecture slightly complicated, but you will be able to do this, because more number of buses means less number of steps, which will again look at look in another future unit on this module, will be looking as 3 bus architecture and then we will be analyzing, but mainly here we are just going for single bus. You can just have a thought that, if you are having multiple bus, then together in one bus you can move the value of $R1$ to $R2$ one bus, you can use for loading the PC to the memory address register and so, forth.

So, it will be bit faster basically, but in if you are able to design all the steps for this load $R1$ and $R2$ then, you can explain the single bus organization, as well as you can say exact steps required to fetch decode and execute instructions. So, with this we come to the end of this unit and next unit we will be going into more depth of some slightly more complicated instructions, like jump and conditional instructions and how we are generating control signals for that.

Thank you.