

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. 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 because 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)

Units in the Module

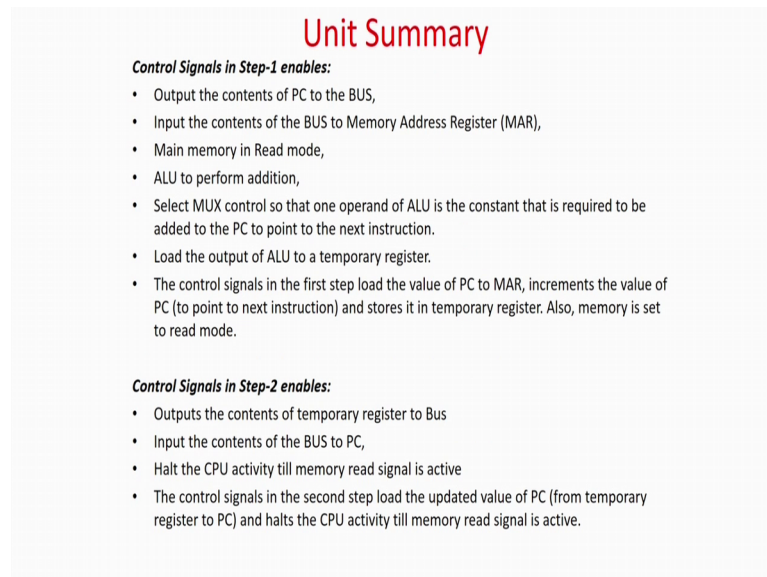
- Instruction Cycle and Micro-operations
- Control Signals and Timing sequence
- Control Signals for Complete Instruction execution
- **Handling Different Addressing Modes**
- Handling Control Transfer Instructions
- Design of Hard-wired Controlled Control Unit
- Different Internal CPU bus Organization
- Micro-instruction and Micro-program
- Organization of Micro-programmed Controlled Control Unit

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 given instruction, like phase 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 you are just going to extend the last unit, where we will be looking at different addressing modes like immediate direct indirect restrain,

direct 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)



Unit Summary

Control Signals in Step-1 enables:

- Output the contents of PC to the BUS,
- Input the contents of the BUS to Memory Address Register (MAR),
- Main memory in Read mode,
- ALU to perform addition,
- Select MUX control so that one operand of ALU is the constant that is required to be added to the PC to point to the next instruction.
- Load the output of ALU to a temporary register.
- The control signals in the first step load the value of PC to MAR, increments the value of PC (to point to next instruction) and stores it in temporary register. Also, memory is set to read mode.

Control Signals in Step-2 enables:

- Outputs the contents of temporary register to Bus
- Input the contents of the BUS to PC,
- Halt the CPU activity till memory read signal is active
- The control signals in the second step load the updated value of PC (from temporary register to PC) and halts the CPU activity till memory read signal is active.

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 you 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 a program counter value which is now in the bus with a constant value, which is the next memory location and then it is again weighted 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 equal to PC plus 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 dump the values which or the instruction, which was available in the memory

location called pointed by the PC to the memory data register or the memory buffer register.

(Refer Slide Time: 02:47)

Unit Summary

Control Signals in Step-3 enables:

- Outputs the contents Memory Data Register (MDR) to Bus
- Input the contents of the BUS to Instruction Register (IR),
- The control signals in the first step load value of MDR to IR after till memory read signal is active. So, now the IR has the instruction that needs to be executed.
- The control signals for the fetch cycle are same for all modes and types of instructions given a fixed bus system. The decode and execute cycles have different control signals (from step-4 onwards) which are dependent on the instructions type and addressing mode. Obviously, if the bus organization changes the control signals even for the fetch cycle get changed.

Control Signals in Step-4 enables:

- The value of M (i.e., the memory location from which data is to be read) is loaded into the MAR from the IR.
- The memory control signal as made READ because the contents of the memory location specified in the IR needs to be loaded into the MDR (in 5th control step after WMFC).

In step 3, basically the memory tells that it is ready and the memory data register now, have 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 times they will value. 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 is 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 OP code may be some register R1 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).

Unit Summary

Control Signals in Step-5 enables:

- Wait for memory to respond i.e., by signal WMFC. Once MFC is high the value of memory has been loaded into the MDR.

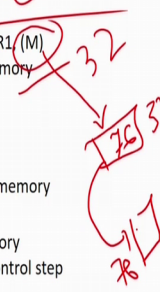
Control Signals in Step-6 enables:

- The value present in the MDR (i.e., the operand) is loaded into register R1.

However, if the addressing mode of the instruction is indirect i.e., LOAD R1 (M) (i.e., load the value present memory location that is given in another memory location M to register R1) the sequence of control signals change.

Control Signals in Step-4 enables

- Value of M (i.e., the memory location from which the address of the memory location where data is present) is loaded into the MAR from the IR
- The memory control signal is READ because the contents of the memory location specified in the IR needs to be loaded into the MDR (in 5th control step after WMFC).



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 sixth stage, in the sixth stage you 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 an indirect instruction sorry if it is, in this case, is an indirect mode well 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 is equal to 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. So, all addressing. So, in this case actually slightly the number of steps from 4 onwards is basically going to

change after 3, basically till 3 switch and after that it will depend on the addressing mode of 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, is a very rudimentary operation in which case you just transfer the value basically it will be OP code say R1 and we have 7 x 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 R1 that is you are going to load the value of 7 from the instruction register to register R1 by a bus.

So, there will be nothing called step 5, but now even in indirect mode easiest way to always 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)

Unit Summary

M=3

Control Signals in Step-5 enables:

- Wait for memory to respond i.e., by signal WMFC. Once MFC is high the value of memory has been loaded into the MDR.

Control Signals in Step-6 enables:

- The value present in the MDR (i.e., the operand) is loaded into register R1.

However, if the addressing mode of the instruction is indirect i.e., LOAD R1, (M) (i.e., load the value present memory location that is given in another memory location M to register R1) the sequence of control signals change.

Control Signals in Step-4 enables

- Value of M (i.e., the memory location from which the address of the memory location where data is present) is loaded into the MAR from the IR.
- The memory control signal is READ because the contents of the memory location specified in the IR needs to be loaded into the MDR (in 5th control step after WMFC).

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 n say it has the value of say assume that the value of 72 may be this is 72, assume M we can assume n equal to 30 say for example.

So, basically this is address of the memory location address is 30, which is equal to n. 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 your 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 this 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)

Unit Summary

- Control Signals in Step-5 enables**
 - Wait for memory to respond i.e., by signal WMFC. Once MFC is high the value of memory (i.e., memory address of the operand) has been loaded into the MDR.
- Control Signals in Step-6 enables**
 - The value present in the MDR is loaded into MAR.
- Control Signals in Step-7 enables**
 - Wait for memory to respond i.e., by signal WMFC. Once MFC is high the value of memory (i.e., operand) to be read has been loaded into the MDR.
- Control Signals in Step-8 enables**
 - The value present in MDR (i.e., the operand) is loaded into R1.

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 or 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 a 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 R1.

So, in add 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)

Unit Objectives

- **Comprehension: Explain:**--Explain the addressing mode with respect to internal structure of the processor and instruction format.
- **Synthesis: Design:**--Explain the Design of complete control steps to execute an instruction that involves different addressing modes, such as, Memory Direct, Memory Indirect, Register Direct, Register Indirect, Immediate, etc.

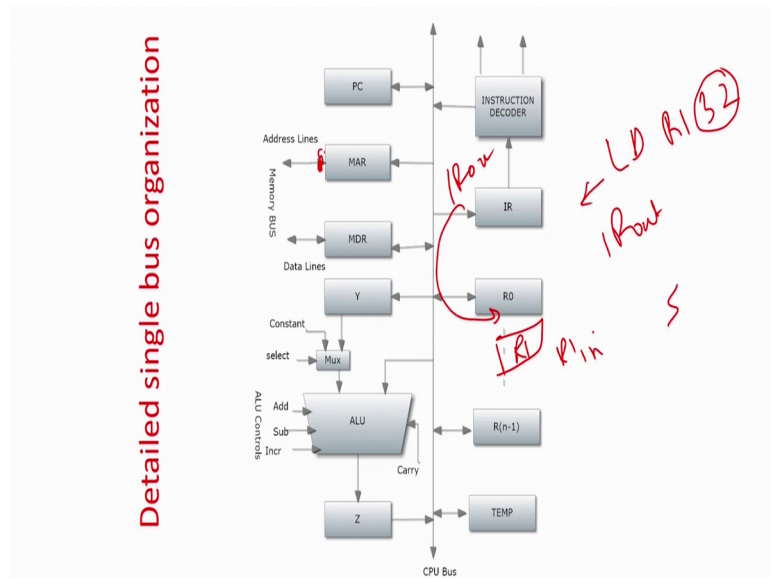
And if we look at what are the different types of basically objectives, we are going to support in this unit. So, if you look at we have the first is a comprehension objective in which case we will be able to explain the different addressing modes with respect to internal structure of the processor and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

So, basically the objective of this unit, that is that after will be 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)

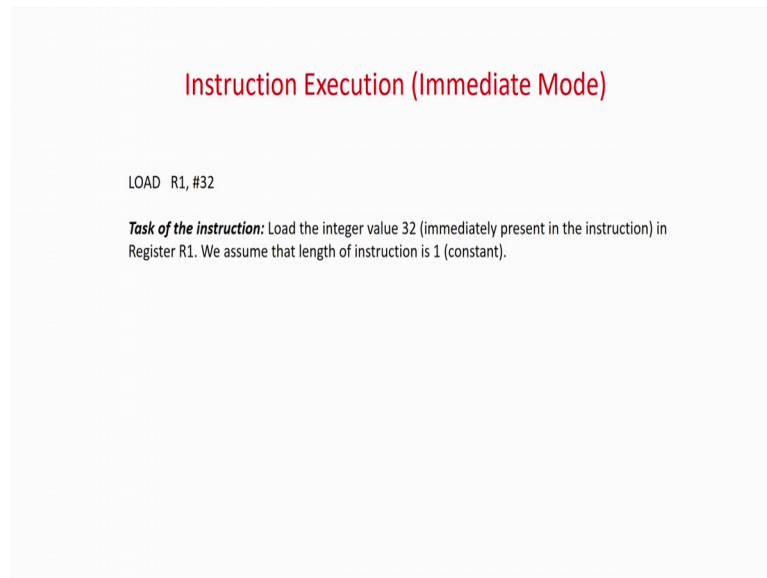


Basically already we have seen yesterday, when as I am just giving a recap because we are going to go for extending the instruction, micro instruction and the control signal based on the single bus architecture. So, if you remember we have a program counter we have memory address the distant. In fact, 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, an 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 store calls Z.

This is your register bank, instruction register and I click go into the instruction decoder to generous the instructions. It is control signals for the instruction basically this is the structure we are explained in details in the last unit.

(Refer Slide Time: 12:04)



Instruction Execution (Immediate Mode)

LOAD R1, #32

Task of the instruction: Load the integer value 32 (immediately present in the instruction) in Register R1. We assume that length of instruction is 1 (constant).

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 to the program counter to the bus and vice versa.

So, first instruction what the most simplest one is the immediate mode. So, we are just saying load R1 32; that means, the value of 32 constant will be loaded to R1 and we are assuming that this instruction takes a single one. So, in this case the PC has to be incremented by 1.

(Refer Slide Time: 12:37)

Instruction Execution (Immediate Mode)

1. PC_{out} , MARin, Read, Select=0, Add, Zin

In the first control step the value in the PC is loaded into the MAR and the control signals are PC_{out} and MARin. At the same control step we need to make the memory control signal as READ because the contents of the memory location specified in the PC needs to be loaded into the IR (in 3rd control step after WMFC).

Also in this control step we initiate to increment PC to point to the next instruction. For this we make control signal select=0 so that constant is fed to ALU as one of its inputs. Also, ALU is configured to perform addition by making control signal as Add. The ALU adds the constant with present value of PC (fed through the CPU bus). Control Zin enables loading of the ALU (i.e., $PC + \text{constant}$) output to register Z.

2. Zout, PCin, WMFC

In the second control step the updated value of PC that is in register Z (1st control step) is loaded into the PC; this is achieved by control signals Zout and PCin. Also, in this step we wait for memory to respond i.e., by signal WMFC. Once MFC is high the value of memory to be read has been loaded into the MDR. So now the MDR contains the instruction what was present in the memory location pointed by the PC (in the 1st step).

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 you have already seen in details in the last unit then in a second stage Z out; that means, the output of the ALU that is Z which is now equal to PC equal to PC plus constant will be going to PC in; that means, the output that is PC equal to PC plus constant will be noted into the PC where a bus and we are waiting till the memory cell to signal that I am ready.

(Refer Slide Time: 13:33)

Instruction Execution (Immediate Mode)


3. MDR_{out} , IR_{in}

In this step the value present in the MDR (i.e., the instruction) is loaded into the IR. This is achieved by signals MDR_{out} , IR_{in} .

4. IR_{out} , $R1_{in}$

In the fourth control step the value 32 (immediately present in the instruction) is loaded into register R1 and the control signals are IR_{out} and $R1_{in}$. It may be noted that the IR has the entire instruction i.e., OPcode-code for R1-32 in binary. The instruction decoder understands that in the current mode of instruction (immediate) operand is in the instruction itself and only "32 in binary" is loaded from IR to R1.

↳ Also, another point is to be noted is that there is no direct output of the IR to the CPU bus. The "32 in binary" part of the IR is loaded to R1 through CPU Bus and the output is given by instruction decoder. The IR cannot give any output of its own and the instruction needs to be decoded before a meaningful output can be produced. However instead of making the signal as "instruction decoder_{out}", with slight abuse of notation we denote the signal as IR_{out} .



So, once it is ready the output of the memory data register, which now has the instruction that is equal to load R1 32 we 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 if the immediate instruction we just require just one more control part that is in the instruction register we now have the value called load R1 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 instructions that R out and R1in that is as simple as taking this step if you look at it this. Now basically you have loaded your instruction that is some load R1 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 R1. So, this is may be this register is called R1.

So, it will be R1 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 we inst 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)

Instruction Execution (Direct Mode)

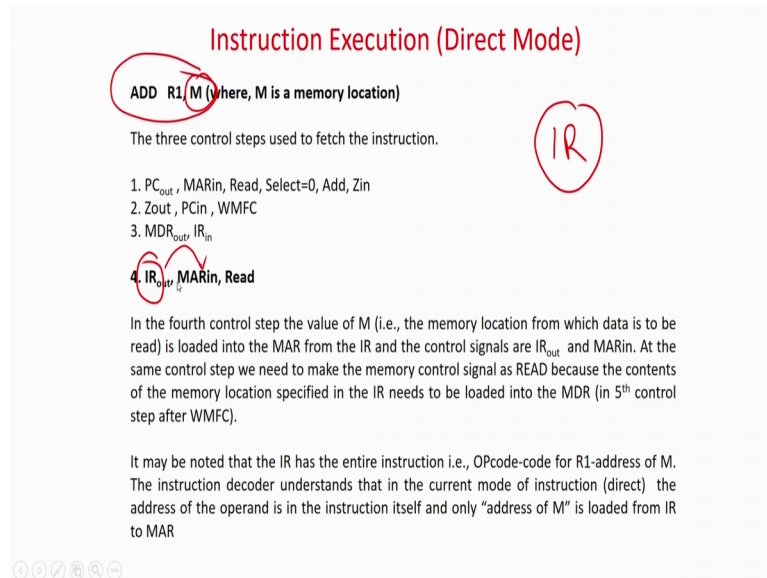
ADD R1, M (where, M is a memory location)

The three control steps used to fetch the instruction.

1. PC_{out} , MAR_{in} , Read, $Select=0$, Add, Zin
2. $Zout$, $PCin$, $WMFC$
3. MDR_{out} , IR_{in}
4. IR_{out} , MAR_{in} , Read

In the fourth control step the value of M (i.e., the memory location from which data is to be read) is loaded into the MAR from the IR and the control signals are IR_{out} and MAR_{in} . At the same control step we need to make the memory control signal as READ because the contents of the memory location specified in the IR needs to be loaded into the MDR (in 5th control step after WMFC).

It may be noted that the IR has the entire instruction i.e., Opcode-code for R1-address of M. The instruction decoder understands that in the current mode of instruction (direct) the address of the operand is in the instruction itself and only "address of M" is loaded from IR to MAR

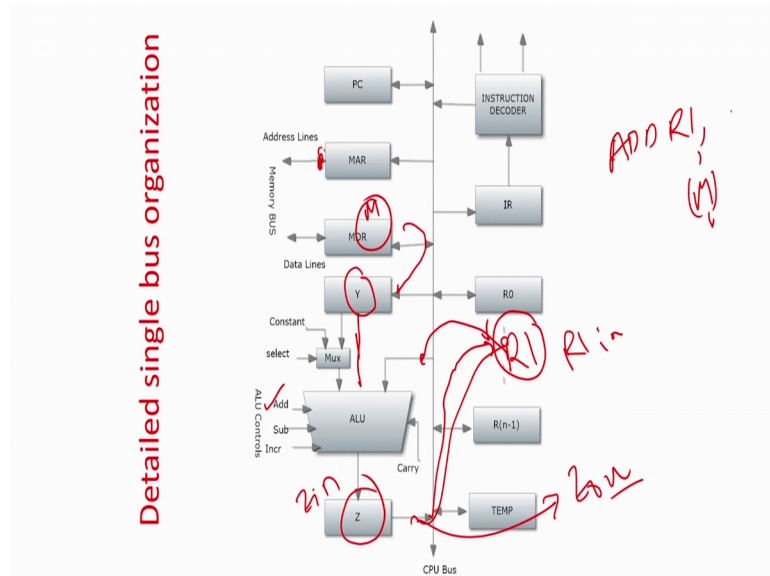


Then next is the direct mode. Direct mode already also we have discussed yesterday, but in this case you are going to just see a comparison that what is the more complexity involved. So, in this case we will see that we require 6 stages, for 6 micro instruction in sequence. So, the first three like program counter value to memory address, read select an increment of PC and this 3 basically and MDR out to restrain, 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 at R1M, 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 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 damping 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 R1 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 come as an opera and it will go to register R1 or wherever you want to dump there 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)

Instruction Execution (Direct Mode)

5. WMFC

In the fifth control step we wait for memory to respond i.e., by signal WMFC. Once MFC is high the value of memory to be read has been loaded into the MDR.

6. $MDR_{out}, Y_{in}, Select=1, Add$

In this step the value present in the MDR (i.e., the operand) is loaded into register Y. This is achieved by signals MDR_{out}, Y_{in} .

Also, the input to multiplexer $Select=1$, which ensures that the first operand to ALU is the value at Y (i.e., the content of M). The control line to ALU is Add which configures the ALU to perform addition.

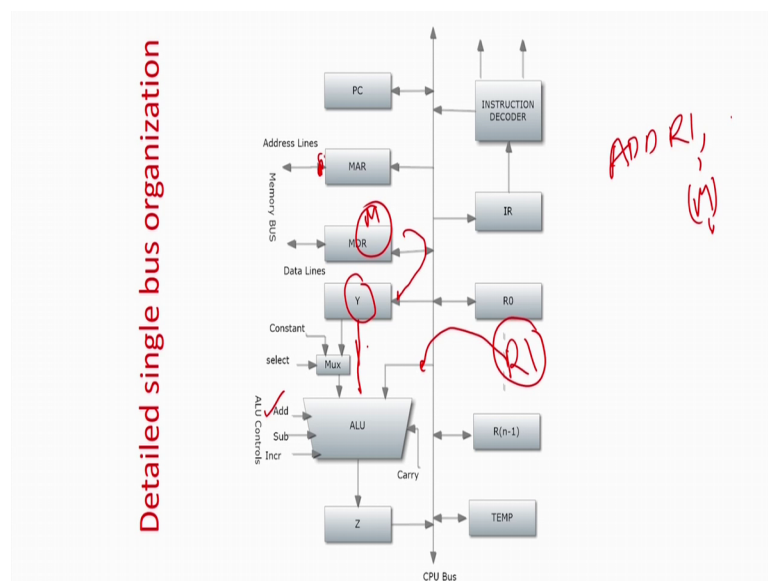
Handwritten red annotations include: 'MDR' in a circle with an arrow pointing to the MDR; 'M' in a circle with an arrow pointing to the MDR; and 'ADD R1, (M)' written at the bottom right.

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 R1 or M or load R1M 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 R1. So, having had the instruction being load R1M, you could have done there is simply that after this in 6 stage we could have made memory data register out and we could have made R1 in, but in this case instead of load if we then 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 this structure again. So, now, if you look at it so, now, the value of M has been noted for the operand which is present in M is present over here the content of M is available over here.

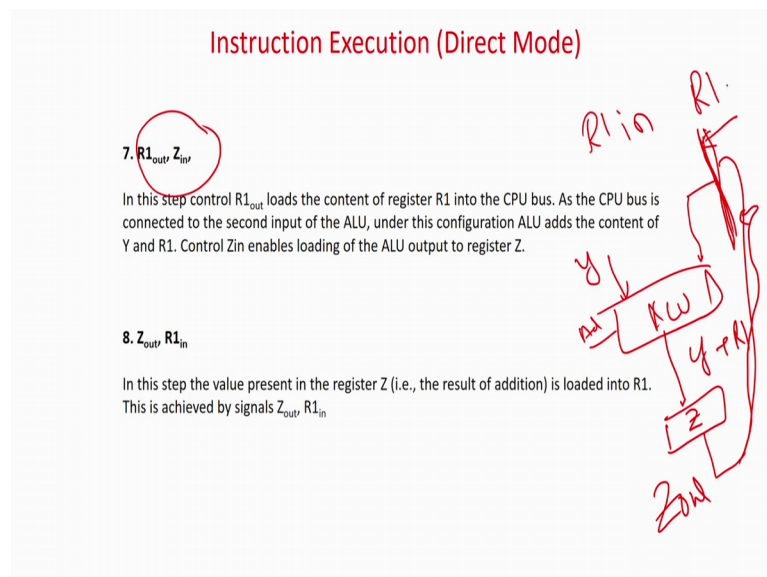
(Refer Slide Time: 18:12)



So, now the instruction was add R1 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 R1 which will directly feed over this operand. So, that I can add, 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 equal to 1 and I will make this select equal to 1. So, that the value of M via Y can be coming to the, a b.

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 equal to 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 state 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 this is what this is R out, R1, you are making R1 out. So, the value of R1 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 R1 which I am directly taking where the bus is your single bus by making R1 out. 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 R1.

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 R1 that is the content of Y plus register R1 would be dumped to the bus and I will make R1 in. So, if I make R1 in. So, the value will be fed from Z out to register R1 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 load R1 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 R1 via R out here and z in is there. So, the value of R1 plus the value of content of M by via Y is now in z in last stage is basically we make the z out z out. So, the value of z is dumped over here and it is going to R1. So, we are going to make 1 in R1 in. So, in that case the value of then by virtue of signals it out will be dumped to register R1. So, Z out equal to 1 and R in equal to 1 will dump the value of Y, then is you actually the content of M plus content of R1 into R1 itself. So, that is the basic path. So, this, what has been explained.

(Refer Slide Time: 21:48)

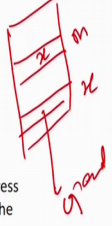
Instruction Execution (Indirect Mode)

"Load R1, (M)" (where, M is a memory location)
Task of the instruction: Load the content of Memory Location M1 to Register R1, where address of M1 is specified in memory location M. We assume that length of instruction is 1 (constant).

The three control steps used to fetch the instruction.

1. PC_{out} , MAR_{in} , Read, $Select=0$, Add, Z_{in}
2. Z_{out} , PC_{in} , $WMFC$
3. MDR_{out} , IR_{in}

4. IR_{out} , MAR_{in} , Read



In the fourth control step the value of M (i.e., the memory location from which the address of the memory location where data is present) is loaded into the MAR from the IR and the control signals are IR_{out} and MAR_{in} . At the same control step we need to make the memory control signal as READ because the contents of the memory location specified in the IR needs to be loaded into the MDR (in 5th control step after WMFC).

It may be noted that the IR has the entire instruction i.e., Opcode-code for R1-address of M. The instruction decoder understands that in the current mode of instruction (indirect) the indirect address of the operand is in the instruction itself and only "address of M" is loaded from IR to MAR

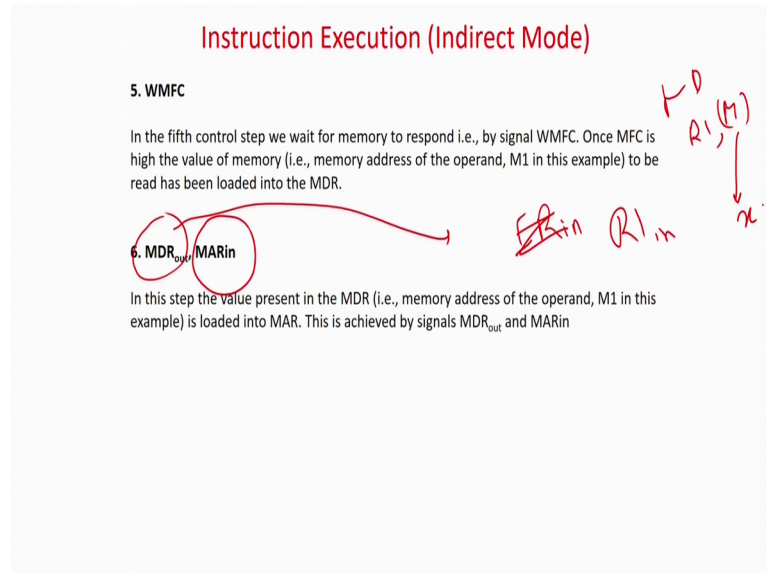
Now, we will go to the other mode, which is more complex in nature which is the indirect mode. By indirect mode already we mean that whenever is the indirect mode of M; that means, we say this is the memory loc, this is the memory here is M, at the address of M here there may be some addresses called x is some content over here, then again you have to look at the content in x and basically this is your operand, this is what is the idea we all know about it. So, if you now look at basically your first 3 stages. So, first 3 stages; as I already were discussing like PC out MAR in and this is control stage basically are only for fetching the instruction.

Next one is what I have to do. So, this is the instruction load R1 into memory from indirect memory location, that is the content of M you have to again go to that memory location and there will get the operand it has to be loaded to R1 that is we say that load the control memory location M1 to register R1 where M1 is specified in the memory location if that is indirect and we assume that the length is one.

So, the next stage is IR out memory register in and read if we are fetching the basically. So, what we are doing here. So, now, the after executing the third instruction the third control step register instruction register R IR is having the value of load R1M. So, in the fourth stage what I am going to do I am going to take the value of M and I am going to feed it into the memory address register so, that you can read the value of memory location M that is x in this example.

So, IRout that is you are going to take the value of M and dump it into the memory register address register. So, that now I can read the value of x.

(Refer Slide Time: 23:24)

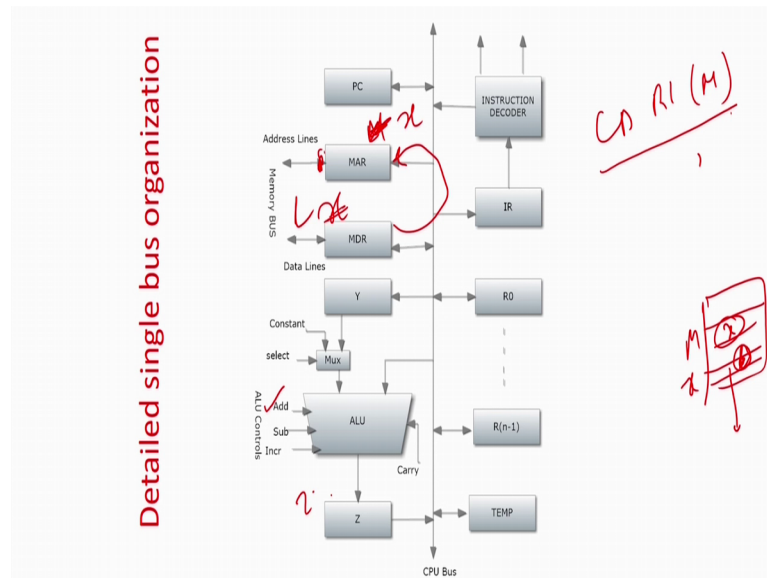


Of course, in stage 5 we have to wait till everything is ready. So, once it is ready basically we know that the value of memory location M that is x in the example is now loaded into the memory data register now interestingly for all cases what we have seen for most of the cases after the memory data out, if it's a direct instruction we generally take this memory data out from the memory buffer register we actually say IR in, if is a fetch or sometimes we call R1in, in that way.

That means, the operand that is present over here the x we are directly loading at some place or we are directly using it as an instruction, but in this case it is very interesting that the memory data out will be fed to the memory address register in basically; that means, what I am going to take this value of x and I am again dumping it to the memory address register because exact value will be found out in memory location x.

So, if you quickly look at what it happens in the bus it will be more clear. So, look at the bus. So, many times you have to refer to this diagram. So, will recollect the phase we are in. So, basically we are in this stage when say load R1.

(Refer Slide Time: 24:40)



So, we just recall the instruction is something like say load R1 indirect M and your M we are having something like this M this is x, this is your memory location x and this is the exact content. So, now, your instruction register basically has load R1 M that is already there now what we are doing. So, now, we are making IR out and we are loading the memory address register with M. So, this is done.

So, now, it is done means, now the content of memory location that is x will be given into the memory data register after you have to wait for MFC, now what happened now this content x generally is used by some of the registers if the data or it is an instruction it will go to IR, but in this case interestingly what is going to happen this x will be again going to fed back to the memory address register. So, in this case now it will be an x.

So, if it is an x, the content we can say that now the content is 1. So, that content will again come here and which is actually your real operand. So, in next case what we do we say, memory data register out and memory address register in. So, that the value of x will be fed to the memory address register now the memory buffer register after a wait will give the exact value, which is present in memory location which is your exact data. So, it will, you will come to the memory data register and then you can load it to your R1 or wherever by a simple signal sequence that is MDR out register R1 in.

So, that is what is happening right. So, in this case you read the instruction then you read the wait for some time, the signal that is add. So, load R1M, this instruction is already

loaded into the memory then you take in the sequence you memory data register out and basically you are going it in memory in that is a very important stage over here that is what going to take the value of content of M, which was x and you are dumping into the memory address register, next you wait for some amount of time and then in this case now exactly you are going to have the and operand which is present in the memory location x that the content we are assuming l it will be dump to register R1. So, in two indirection stage R1, we have the exact operand.

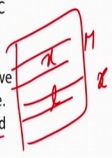
(Refer Slide Time: 26:43)

Instruction Execution (Indirect Mode)

7. WMFC

In the seventh control step we wait for memory to respond i.e., by signal WMFC. Once MFC is high the value of memory (i.e., operand) to be read has been loaded into the MDR.

It may be noted that control signals in 5th and 7th steps are same. However, in the 5th step we read the address of the memory location where operand is stored (i.e., M1) in the example. In the 7th control step we read the operand from memory location (whose value is obtained in the 5th step).



8. MDR_{out}, R1_{in}

In this step the value present in MDR (i.e., the operand) is loaded into R1. This is achieved by signals MDR_{out}, R1_{in}.

That was basically first it contain M which was x now this is x and the exact content l which will be actually dumped into R1.

So, in the indirect mode we can get this. So, now, again and our mode we are taking which is called registering indirect in this case it was a memory indirect register indirect that you have to go to the register and the content of the register will also contain the location of a memory where the data will be present like.

(Refer Slide Time: 27:09)

Instruction Execution (Reg. Indirect Mode)

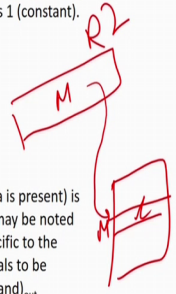
"Load R1, (R2)"
Task of the instruction: Load the content of Memory Location M to Register R1, where address of M is specified in register R2. We assume that length of instruction is 1 (constant).

The three control steps used to fetch the instruction.

1. PC_{out} , MARin, Read, Select=0, Add, Zin
2. Zout, PCin, WMFC
3. MDR_{out} , Irin

4. $R2_{out}$, MARin, Read

In the fourth control step the value of M (i.e., the memory location where data is present) is loaded into the MAR from R2 and the control signals are $R2_{out}$ and MARin. It may be noted that R2 (the register specified as the second operand in the instruction) is specific to the example instruction used in this answer and is not generic. So the control signals to be generated by the instruction decoder is (Register specified as the second operand) $_{out}$.



At the same control step we need to make the memory control signal as READ because the contents of the memory location specified in R2 needs to be loaded into the MDR (in 5th control step after WMFC).

For example, load the content of memory location M to register R1, when M is specified in register R1 that is say there is a register load, the content of memory location M to R1. So, it should be R2. So, load the content of memory location M to register R1 where M is specified in R2. So, R2 will have some value M. So, it is your memory. So, this is your value of M. So, this is your R2.

So, exact value of the operand address will be found in R2. So, R2 will have the value of M, we will have to be first into the looked into the memory this exact value say 1, will be dump R1 that is what is the register indirect mode that we are going to now do. So, as already discussed many times and the first 3 stages are basically for your basic operation of loading the instruction fetching, the instruction then what we do then we say R2 out and memory address register. So, now, in this case it is interesting. So, what happened then if you look at R2? So, R2 basically the register when the exact address of the data is present. So, in this case if you see. So, what I am trying to do. So, if you look at. So, your instruction R2 is registered.

(Refer Slide Time: 28:15)

Instruction Execution (Reg. Indirect Mode)

"Load R1, (R2)"
Task of the instruction: Load the content of Memory Location M to Register R1, where address of M is specified in register R2. We assume that length of instruction is 1 (constant).

The three control steps used to fetch the instruction.

1. PC_{out} , MARin, Read, Select=0, Add, Zin
2. Z_{out} , PC_{in} , WMFC
3. MDR_{out} , Ir_{in}

4. $R2_{out}$, MARin, Read

In the fourth control step the value of M (i.e., the memory location where data is present) is loaded into the MAR from R2 and the control signals are $R2_{out}$ and MARin. It may be noted that R2 (the register specified as the second operand in the instruction) is specific to the example instruction used in this answer and is not generic. So the control signals to be generated by the instruction decoder is (Register specified as the second operand) $_{out}$.

At the same control step we need to make the memory control signal as READ because the contents of the memory location specified in R2 needs to be loaded into the MDR (in 5th control step after WMFC).

Basically is your register R2, it has the value of M. So, that is your actually memory location address. So, this one you have to feed it directly do the memory address register. So, once you give the value of M, which is the content of R2 the memory address register, it will get the value 1 priority. So, you do register R2 from memory addresses from R2, actually you are going to dump the value of the memory address scheme because it will have the value.

So, you note in this case we are not actually reading the value to memory address register from the instruction register for all other cases, if you look. So, if you had some instruction like as I told you like say load R1 say M. So, what we used to do we used lower the value of M, from the instruction register directly to memory address register, but in register indirect mode if you observe what we are doing we are directly taking the value of R2 and you are running it the memory addressing. So, actually the instruction register very be codes, it finds that it is an indirect addressing mode.

So, in that case the role of the instruction register is not used to directly hand over all the stuffs to register R2 and not getting involved in the picture, but if it's a direct mode or an indirect mode not involving a register. So, in this case the instruction register value the M has to be directly basically loaded into the memory address register. So, in this case the instruction, the decoder will generate such a signal. So, that instead of R2 out it will be instruction register out that is it will take the value of M and dumping the new register

here it is a memory direct or memory indirect, but even when the instruction register finds that basically.

Now, the instruction is involved with register only and no memories in picture. So, directly R2 the content of R2 will be directly donate to the memory address register; that means, basically R2, the contain M will not be the memory address register after that it is very simple you have to wait for some amount of time and in that case basically. So, once it is done the memory data you have already given, it the address of M to the area address register you have to wait for some amount of time and then this I will come to the memory data register of the memory buffer register and it is done then just you have to load it into R2. So, you have to wait for some amount of time.

(Refer Slide Time: 30:33)


Instruction Execution (~~Direct Mode~~)
Reg. Indirect

5. WMFC

In the fifth control step we wait for memory to respond i.e., by signal WMFC. Once MFC is high the value of memory (i.e., memory address of the operand specified in R2, in this example) to be read has been loaded into the MDR.

6. MDR_{out}, R1_{in}

In this step the value present in MDR (i.e., the operand) is loaded into R1. This is achieved by signals MDR_{out}, R1_{in}.



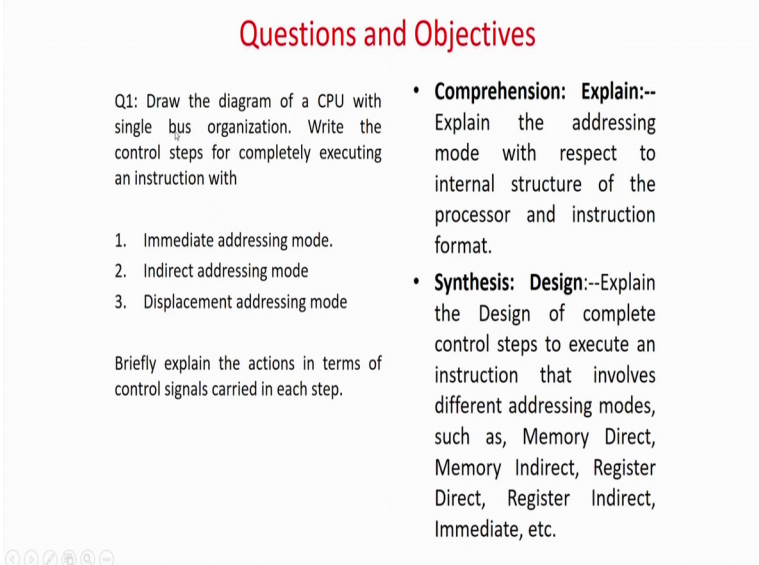
After the memory register you are going to dump it to R in.

So, you can see, it is actually in this case it should be I am just a mistake it should be basically register, indirect mode register, indirect there no copy problem. So, in this case if you can see. So, your indirect mode if the memory indirect mode you require 8 stage to do it, but if in your register indirect basically, you can solve it in 6, basically 6 steps of time. So, basically register indirect is a more faster mode compared to a or less number of steps required to a memory access, because in this case you just require one access, but in the previous case you require two memory access and in fact, also it depends on whether you want to load or you want to add and all those problems are there.

So, depending on different type of addressing modes the sequence of control at the number of steps will all differ. So, this unit basically gave you a spectrum of different type of addressing modes and with different functionality like load and add and we have shown what are the sequence of control instructions required for the micro instruction and how a complete instruction can be controlled in, with this respect to the control signals that is basically which we have covered.

So, we have taken immediate more direct, more indirect, more register, indirect mode register mode and so forth. So, we gave you a wide spectrum of designing instruction set in terms of basically our control signals given a single bus architecture as well as defined type of addressing modes under consideration. So, basically I told you these are quite small module. So, we could come to that and then we have a very simple question like draw the diagram of a CPU with single bus architecture and write different types of instructions with immediate addressing, indirect addressing, displacement addressing.

(Refer Slide Time: 32:05)



Questions and Objectives

Q1: Draw the diagram of a CPU with single bus organization. Write the control steps for completely executing an instruction with

1. Immediate addressing mode.
2. Indirect addressing mode
3. Displacement addressing mode

Briefly explain the actions in terms of control signals carried in each step.

- **Comprehension: Explain:--** Explain the addressing mode with respect to internal structure of the processor and instruction format.
- **Synthesis: Design:--** Explain the Design of complete control steps to execute an instruction that involves different addressing modes, such as, Memory Direct, Memory Indirect, Register Direct, Register Indirect, Immediate, etc.

and some other addressing modes and try to design this try to express how the instructions will execute in terms of control steps and what are the control signals for that if you were able to answer this of course, the two objectives of this course like explained the addressing mode with respect to internal structure of the processor and instruction format and the control instructions and design the complete control step to execute instruction that involved different addressing modes get satisfied.

So, once you, after doing this unit you will be able to solve this problem and you will be able to meet these objectives. So, with this basically we come to an end of this unit and in the next unit what we will be looking at, we will they are more sophisticated type of instruction like a jump instruction function call etcetera. So, which will (Refer Time: 33:25) more integrated details.

Thank you.