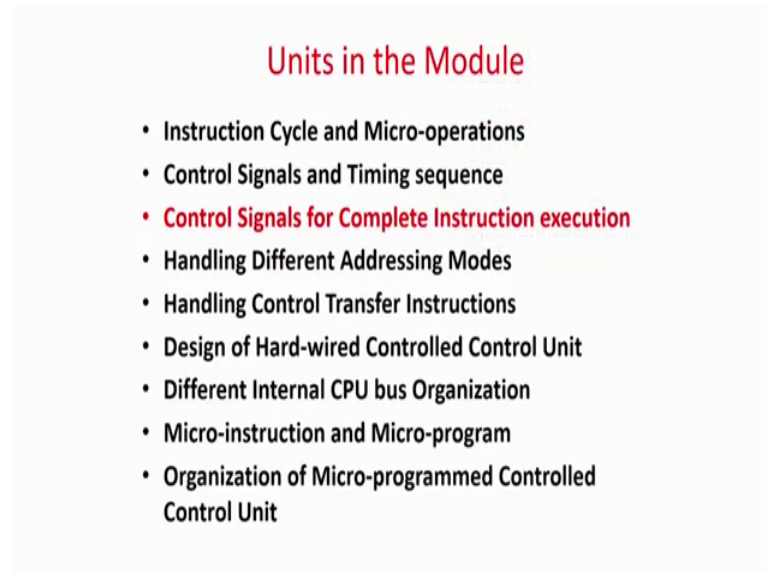


Computer Organization and Architecture: A Pedagogical Aspect
Prof. Jatindra Kr. Deka
Dr. Santosh Biswas
Dr. Arnab Sarkar
Department of Computer Science & Engineering
Indian Institute of Technology, Guwahati

Lecture – 17
Control Signals for Complete Instruction Execution

Hello and welcome, to the third unit of the module on control and this unit is concern with control signals, for complete instruction execution.

(Refer Slide Time: 00:39)

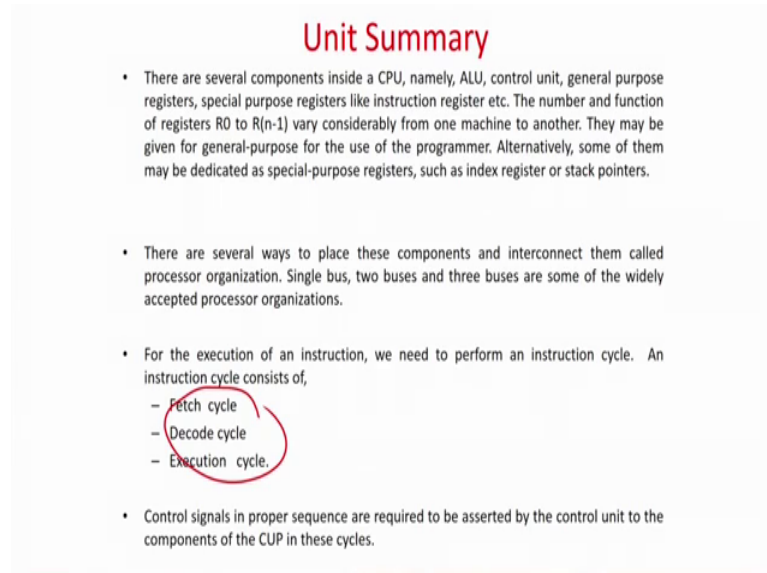


So, if you look at flow, in last basic two units we mainly covered about. What is single bus organization? How different components are connected? and then we have looked into that, for a given instruction what are the very broad kind of control signals require and what are the timing diagrams, require involved in generate the control signals, to execute the instruction.

In this unit, as we have seen we will actually look into depth, of the control signals how they are generated in single bus architecture? And how and we will look in details? That how these control signals are required or executed to implement a complete instruction? That is, we will take some instructions and we will see, how different control signals

generated or required for the complete instruction execution? That is what is the contain of today's unit.

(Refer Slide Time: 01:23)



The slide is titled "Unit Summary" in red text. It contains a bulleted list of five points. The third point, "For the execution of an instruction, we need to perform an instruction cycle. An instruction cycle consists of," is followed by a sub-list of three items: "Fetch cycle", "Decode cycle", and "Execution cycle." These three items are circled in red. The other points discuss CPU components, processor organizations, and control signals.

Unit Summary

- There are several components inside a CPU, namely, ALU, control unit, general purpose registers, special purpose registers like instruction register etc. The number and function of registers R0 to R(n-1) vary considerably from one machine to another. They may be given for general-purpose for the use of the programmer. Alternatively, some of them may be dedicated as special-purpose registers, such as index register or stack pointers.
- There are several ways to place these components and interconnect them called processor organization. Single bus, two buses and three buses are some of the widely accepted processor organizations.
- For the execution of an instruction, we need to perform an instruction cycle. An instruction cycle consists of,
 - Fetch cycle
 - Decode cycle
 - Execution cycle.
- Control signals in proper sequence are required to be asserted by the control unit to the components of the CUP in these cycles.

Basically, what will be covering in this unit, the unit summery will basically we will have quick revisit, as you can see in the first point. We will quickly revisiting what is single bus architecture? Because that is what is mainly we are dealing with all the examples and most of a study is basically on a single bus architecture. Then we look at where the ALU is connected? What are the different types of registers? What are the program counters? Instruction register their inter connects, all this things will have a quick reconnect recollect. Basically, and then because, as we know there are multiple bus structure also like 2 and 3, but it is slightly more advanced and we are not going to look at in, much details in this course.

Then, we click quickly jump to the different cycles, of an instruction like fetch decode execute and there will exactly see that what are the basic control signals require? In each of the cycles and then we will see that, for which for any cycle, any instruction like fetch decode and execute we will see that for some cases, which part of the control instructions or control signals are similar and for which part, basically is differ like for fetching an instruction, it will be same for all this instructions because, you have to fetch it from the memory.

So, more or less the control signal sequence will be similar for any instruction in the fetch phase, decode phase is nothing but basically, you take the instruction from your memory data register to the instruction register, and try to find out what happens? So, initial part may be similar and the next part will be different, if for example, if the direct instruction or if it is an immediate instruction, if it is an indirect instruction for more complicate instruction, we have already seen in the last unit, that you require more cycles I mean execution in terms of control units and etc or.

In fact, in the last class or last unit we are looking at these control instructions, or control signals in terms of micro instructions. So, more the number of micro instructions or different of the types of micro instructions, for a given instruction cycle different will be control signals. So, that we will see; how it differs mainly for decode and execute cycle.

(Refer Slide Time: 03:19)

Unit Summary

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.

In the first step the control signals 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.

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

In the second step the control signal loads the updated value of PC (from temporary register to PC) and halts the CPU activity till memory read signal is active.

PC = PC + Const

Basically, in a nutshell, what we will see? Any first step of the instruction basic instruction flow that is, basically your the fetch. So, fetch basically what happens? You take instruction from the memory and basically bring it to the instruction register that is first part of the instruction.

So, we will see what basically in the unit, we will first see what are the basic type of control signal require to do that, basically what happens first, you see it will output the it shows that output the contains of the PC to the BUS, there because of the program counter already will point out, that fetch instruction has to be executed, then the contains

of the bus will be loaded in to the memory address register, because when the program counter value will be loaded to the memory address register and the memory is in the read mode.

So, what will happen basically based on the contents of the program counter, basically you will load the value of your instruction, that is because from the memory address register, it will tell you where the instruction is obtained and it will be loaded to the address it will be loaded to the memory, buffer register of the memory data register and. In fact, also in this instruction you will also have to increment the program counter to the point to point to the next instruction. So, what we do? We also instruct the ALU to perform addition, in this case it will add the value of program counter which is now in the bus, with the increment.

So, if the program if the instructions are 1 bit, 1 memory with sorry in 1 memory 1 then, you will add 1 and in other case, if it a 2-odd instruction you will add 2 and so, forth. So, basically the first step of the instruction, basically loads the instruction from the memory to the instruction register, it initiates and it will also increment value of program counter to point to the next location, the you selects the mux control that is one operate of ALU to the constant, that is you are going to add program counter whose value is in the bus to a content.

So, we have already seen in the last class that basically, the ALU one of the operates can be a constant or a register value. In this case we keep it as a constant, the constant value here is nothing but it is the length of the new instruction; that means you are by it to the present value of the program counter; we will point to the next instruction.

Then we will load the value of ALU to a temporary register. So, the temporary register now has the value of PC, equal to PC plus this content. So, what happens in the first stage, basically what we have done you can look at the last comment, what it says? In the first step load the value of program counter in the memory address register, increment the value of PC and store it in a temporary register, also the memory is set to read mode; that means, now your memory is pointing towards the at the memory location, where the current instruction is there.

So, that it can be rate into memory buffer register in the next step, and program counter has be incremented, but the increment value of program counter is now in a temporary

register, not yet uploaded to the program counter. Next what you do? Next output the content of the temporary register to bus and input the content of the bus to the P C, because at present the temporary register is holding the value of PC plus content, you have to dump it to the bus and the bus value will be dump to P C. So now, the program counter will have the value of PC equal to PC plus 1.

Now you have to wait till the memory signal is ready, basically what happens, whenever we are giving a read command and you have given the data that is the PC value to the memory address register, you have to wait for some amount of time, till the memory says that I am ready and the instruction is now loaded into the memory buffer register. So, basically in this second step, this control signal loads the updated value of PC, from the temporary register to PC and halts the CPU till the memory read signal is active, that is what it is done output the content of the temporary register to the bus, because temporary register as you have seen in the previous instruction has the value of PC plus content.

Then input the value of bus to PC. So now, PC is incremented because we do not no longer this require PC value as of now, because the PC has been loaded to the memory address register.

(Refer Slide Time: 07:07)

Unit Summary

Step-3 enables:

- Outputs the contents Memory Data Register (MDR) to Bus
- Input the contents of the BUS to Instruction Register (IR).

In the third step the control signal loads value of MDR to IR after the memory read signal is active. So, now the IR has the instruction that needs to be executed.

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 is 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).

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.

Step-6 enables:

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

In the third stage, what happens? Outputs the memory data register to the bus, because in second step you already know that, the memory has is now ready after the memory says that it has it is vary, that it has dump the control dump the content of the memory value,

which was pointed by the memory address register, to memory data register and whenever it says that I am ready, the second stage starts, in second stage what will happen, third stage starts, whenever the memory says that I am ready. So, the memory data register value will be content to will dump to the bus. So, what was in the memory data register? It was the point of the memory that was being pointed by program counter, basically it was containing the instruction.

So, the bus value will now go to the instruction register. So, in this third stage what happens, the instruction is now loaded into the instruction register, from the memory data register. So, the third stage the control signal loads the value memory data register, to the instruction register after the memory read signal is active; that means, the memory has set that whatever was required by me, what was what was asked from me to be dumped to the memory data register is now ready, you can read it.

So, after the third stage the instruction has the I R has the instruction, that need to be execute. One important point to be note that, step 1 and step 2 and step 3 are actually impelling the memory fetch, that is instruction fetch. Instruction fetch means you load using the value of the P C, point to the location in the memory where the instruction is stored, increment the value of PC and then dump the increment the value of PC by the arithmetic logic unit, by adding a content the PC is incremented at the same time you have to wait, till the memory says I am ready. Once it is ready, take the value of the memory data register sorry memory data register or the memory buffer register and dump it into the I R register, instruction register via the bus, this will be same this three stages or the three micro instruction and the control signal will be same for any instruction.

Because, they correspond to fetching a instruction, after that step 4 5 6 actually depends on what are the type of instruction it is? What is the addressing mode? And what is it pretends to? And what is to be done? For example, the value of MI the memory location from which the data is to be read, is located is loaded into the memory address register from the instruction register, what is it saying? It is saying that now your instruction register IR, is having the value of your off code and also you have some say R1 some instruction example I am taking and some m.

So, basically what it is saying? Or. In fact, let us make it simple instead of R1, let us call it accumulator. So, it is saying that you have to take one operand from the memory location which is M. So, in this case what happens? This address will be loaded to the memory address register so, that in the next cycle, you can fetch the value from the memory location which is given in this point.

So, you can see 4 step 4 says that, the memory location memory made read made read, because you have to read the operand and the value of M is loaded into the memory address register, from the I R; that means, the I R is now being decoded and it is going to execute. So, before execution, it is decoded that it has to read the value of the operand from the memory, whose location is M. So, that value is loaded to the memory address register, this part is done by step 4, but (Refer Slide Time: 10:25) you have to remember that it is very, very instruction specific.

If it is an immediate mode of operation, then such a stage will not occur so; that means, 4 5 6 7 8 like that, it will depend on different type of instruction more or the instruction type, again you are in a read mode. So, in the step 5 you have to wait till the memory says that, I am ready because in this first 1 2 stage you are reading the instruction, from step 4 5 and 6 in these steps are involved with reading the, operate from the memory. So, you are saying that wait for the memory to respond. So, again once the memory has responded, now the value of M that is you have to load the value that is the operand, which was stored in the memory location M it is now loaded into the memory data register.

Now, in fact, if you remember that whatever data now you are loading in step 5 is a data and not an instruction. So, will not go to I R basically, it will be added with if we op code is load. So, basically it will actually load the value, in accumulator or R1 in this case it is called R1, but our example for was accumulator 6th is the present value in the memory data register, is loaded into the register; that means, here I have mentioned accumulator, but it can be R1 R2 anything. So, basically 6th stage actually executes the instruction, it takes the value from the memory location memory M which is now present in the memory buffer register. So, one so, this one is actually presented in memory data register of the memory buffer register.

Step 5 clears, that memory is (Refer Slide Time: 11:50) memory is ready, data is available in memory data register and then in the 6th stage, you will dump the value of

the memory data register basically, is containing this operand into the accumulator or register R 1 so. In fact, 1 2 3 again I am repeating this is actually summery, which will be covering in this unit for different type of instructions. So, step 1 step 2 step 3 that is instruction fetch will be similar for everything, 4 5 6 7 8 9 will depend on the instruction time.

(Refer Slide Time: 12:20)

Unit Objectives

- **Comprehension: Explain:--** Explain the generation of control signals that is driven by the internal organization of the processor.
- **Synthesis: Design:--** Explaining the design of complete control steps to execute the instructions like ALU operation, Data movement, etc.

Which case you can explain the generation of control signals, that is driven by the internal organization of the processor; that means, given a single bus architecture which is the main focus of this unit, even single bus architecture you will be able to basically explain, how different signals are generated? For each of the micro instructions in a very detailed manner; which will require for a complete instruction execution; Then next the design objective, you can explain the design of complete control steps to execute the instructions like ALU operation data movement etc.

And this different operation style like load, data movement, store, auto increment, decrement for such type of different type of instruction and operand means instruction type of instructions and their addressing mode, you will be able to design a complete steps, that needs to be followed, for generating the control signals for executing the instructions.

(Refer Slide Time: 13:12)



Now, we are going to again revisit the details system bus architecture, single bus architecture in details, but again you have to see the slight details we have compare to the pervious unit. So, this program counter this is your single bus, then you have a memory address register which is connecting. In fact, actually it is a slight stake basically. In fact, the memory address register is a unit directional bus.

So, you should not be there it a unit directional bus so in fact, which is connecting. So, it is connect giving the input to the memory. So, whenever in depending on read and write mode, the your memory bus it will dump the value in the memory data register, if it is a read and it is a write, the data from the memory register will go to the memory.

This is your instruction decoder which is basically decoding instruction and generating the control circuits, this is very, very important and who is going to feed to the instruction decoder? The instruction register the instruction register will again get the value from the bus and it will go to the instruction decoder, which will decode by in terms of control signals based on the instruction to be executed, you can see there are different registers R 0 to R n, there all the registers like general purpose registers, it will depend on 16 32 48 I mean sorry fixing 32 64 depending on the type of person you are using.

So, these the general purpose use there is a temporary register or scratch pad and this is very important part you have to look at it. So, this is your arithmetic logic unit part. So,

you can see this is your ALU. So, in the ALU one input is coming from the bus. So, generally it will be service it will actually take one operand, which available in the bus. So, this part or this part of the ALU the right-hand side part of the input to the ALU, basically taking fetch the data from the bus, we general is an operate, but the left-hand side you can see, it can take data from a register Y the special register sometimes we call accumulator. So, it is going to take the data from the Y as an input or in some other case, is can also take as a constant. So, it is a multiplexer the multiplexer is select claim.

So, you have to in very importantly keep this in mind that this part. So, either the accumulator or the ALU will take data from register Y or a constant. So, when is a constant, where is basically you have to increment the value of P C. So, whenever you have to increment the value of P C, at that time the constant will be given, constant will be length of the instruction and of course, there are different modes of operation in the ALU like add, subtract, increment there will again tell you based on the operation type, is an add instruction, load instruction, and unit sorry load instruction, subtract instruction, increment instruction any type of ALU instruction, that will be commanding the signals will determine that for example, if you have add 2 numbers it will be in add mode subtract multiple and so, forth.

But most important is this part, if the value is coming from Y then it is generally an operand which is already loaded in Y, but if it is you have to increment the value of PC at that time generally, what we do? We take the value from this constant. So, constant is basically used the length of the instructions. So, that $PC = PC + \text{constant}$; that means, is points to the next instruction, in that case select line in the multiplexer will determine that. So, again I request you that please look at this part in more carefully manner, it is these parts please look at it very carefully ok.

(Refer Slide Time: 16:21)

Detailed single bus organization

Program Counter
The program counter (PC) keeps the track of the current instruction being executed. It is a register that contains the address (location) of the current instruction. After any instruction is fetched to the instruction register, program counter increases its value by 1 (or a constant). So, after each instruction is fetched, the program counter points to the next instruction in the sequence.

Memory Address Register
The memory address register (MAR) holds the value of the address location which is to be read or written. The MAR holds two kinds of addresses—(i) the address of an instruction (ii) address of operand.

Memory Data Register
The Memory Data Register (MDR) is a buffer that stores data temporarily which is being transferred from memory to a register or vice versa, I/O device to memory etc.

Arithmetic logic unit
ALU carries out all arithmetic (addition, subtraction etc.) and logic operations (comparison, XOR etc.). Typically, the ALU has two inputs-- input from the CPU bus (connecting to main memory, input/output devices etc.) and accumulator (or a register). The output of ALU is connected to the CPU bus via a register (shown as Z).

So, again we keep on revisiting this, there is not a problem. Now, let us go to this in details. So, there is a program counter. So, this is program counter. So, what else I was telling is written in this. So, basically program counter as we have discussed so many times basically, holds the present location which has to be executed. So, generally in the first step of control instruction, or first micro instruction the value of program counter is loaded to the memory address register as it is incremented.

So, based on that that instruction is fetch and the (Refer Slide Time: 16:44) then there is something called memory address register, as I told you it actually tells the memory that what value? What register? What location of that memory? Has to be read or written. So, generally when you are fetching an instruction, the value of the program counter will be loaded into the memory address register. If there is a store instruction, then when you are saying that store accumulator to memory location n, in that case the M that, will be actually present in the instruction that value will be taken from the instruction register the M and it will be loaded to the memory address register. So, basically memory address register tells that basically, means which memory (Refer Slide Time: 17:19) has to be read or written.

So, therefore, actually the two kinds of addresses, one the address of an instruction that is generally taken from the program counter and address of an operand. So, address of operand address, of an operand is available in the instruction. So, it is generally taken from the instruction register ok. So, next is the data memory data register as you look at

this slide. So, memory data register is nothing but a buffer, which will take the data from the memory, to the bus before that if you think that, this is your memory and this is data bus. So, I generally have a buffer over here, which is the memory data register or memory buffer register.

(Refer Slide Time: 17:48)

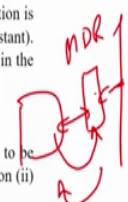
Detailed single bus organization

Program Counter
The program counter (PC) keeps the track of the current instruction being executed. It is a register that contains the address (location) of the current instruction. After any instruction is fetched to the instruction register, program counter increases its value by 1 (or a constant). So, after each instruction is fetched, the program counter points to the next instruction in the sequence.

Memory Address Register
The memory address register (MAR) holds the value of the address location which is to be read or written. The MAR holds two kinds of addresses—(i) the address of an instruction (ii) address of operand.

Memory Data Register
The Memory Data Register (MDR) is a buffer that stores data temporarily which is being transferred from memory to a register or vice versa, I/O device to memory etc.

Arithmetic logic unit
ALU carries out all arithmetic (addition, subtraction etc.) and logic operations (comparison, XOR etc.). Typically, the ALU has two inputs-- input from the CPU bus (connecting to main memory, input/output devices etc.) and accumulator (or a register). The output of ALU is connected to the CPU bus via a register (shown as Z).



So, basically immediately you cannot read and write from the memory. So, you give a read command, then you have to wait for certain amount of time then the data from memory will come to the memory buffer register, and then it will say that I am ready that is data has been given, then only we bus and read it from the memory buffer register or the memory data register.

Similarly, if you want to write also, you put the value in memory data register and then you give a write command to the memory, but you have to wait for sometime till the memory save ok, then you know that the data has been read from the memory data register to the memory. So, that one is a buffer in between.

Then arithmetic logic unit, nothing to tell much either it can do all arithmetic and logic operation, but one important thing basically you can see that it is connected by a Z; that means, what? That is because you have to in I mean, because the output of the ALU will be dump to Z which is a register. So, that it holds for some amount of time that is for one block unit of time.

So, that it can be given to the respective place. So, the output of ALU is stored in a register called Z, which holds it for sometime before you take the value and the ALU can be reused. Because if I do not store the value of register, if I do not store the value of the output of ALU in the Z, then you can have some kind of problem like there can be overwritten, that may be I am storing the value of 3 which I have bought in the last block part, but if I do not store over here, then actually it is a value come over here, there can be a production in the output. So, therefore, we have a temporary register or a register of a Z, which holds the value of ALU output for some amount of time that is what the arithmetic logic unit.

(Refer Slide Time: 19:30)

Detailed single bus organization

Instruction register
Instruction register (IR) is a register that stores the instruction currently being executed or decoded.

Instruction decoder
Instruction Decoder decodes an instruction and generates the corresponding control signals. For example, if there is an ADD instruction then the IR generates signals that enable the ALU to perform addition.

Registers
As shown in the figure there are user programmable registers R0 to R(n-1).

Multiplexer and Constant
One input of the multiplexer is from register Y and the other is a constant. The input from Y is taken in case of an ordinary arithmetic/logic operation. On the other hand, if the adder of ALU is used to increment PC then instead of the input from Y, a constant is taken as input. The value of the constant depends on the memory organization.

Handwritten annotations: "push" in red ink above the Instruction decoder section, and a red circle around "registers R0 to R(n-1)" in the Registers section.

Instruction register, nothing to tell more it take the instruction and knows what to do. Instruction decoder, basically it is a normal decoder circuit it will take the instruction, because all instruction has first is off code. So, will take the off code and according to generate control signals, corresponding to that off code register suggest as I told you shown from R 0 to R n minus 1, there all general common registers, it depends basically on what is your processor type? And what is your processor?

So, it can range from 4 to 32 registers. And so, forth then we have multiplexer and constant. So, this part already I have told you many times right now, but again let us force. So, what it says is this part. So, this is your ALU and this is your register, basically this is your marks and this one. So, basically as I told you, the ALU can either take input

from Y the, it will be an operate, or it can take a via multiples constant. So, if this constant made it is for the implementing of it P C. So, that is what has been told over here, that multiplex and constant what are the meaning? Basically, instated of Y it is an operant, it will take the value from the constant which is from incusing the value of program counter.

(Refer Slide Time: 20:36)

Single bus organization: Instruction Execution

"LOAD R1, M" (where, M is a memory location)

Task of the instruction: Load the content of Memory Location M in Register R1. We assume that length of instruction is 1 (constant).

The fetch operation basically involves loading the value of PC to MAR and fetch the instruction from the memory location (specified in PC) to the IR.

The slide features a red circle around the instruction "LOAD R1, M" and a red arrow pointing from it to the label "PC".

So, which is basic background in mind, we will now see for dewfall instructions, what are the control signals generated? In full flow of instruction, now one thing before we go one thing we have to remember, that for example, there is something for everything, there is a command involved in and out. So, for example, if I want to say that the value of register Z has to be dump to the CPU bus, then you will write Z output; that means, the value of Z would be dump to the bus for example, the value of P C, but PC has to read the value from the bus. So, in that case what we will do? You will write PC in; that means, what PC is going to take the input from the bus, as already discuss in the last unit you cannot have simultaneously Z out and PC out, in this case PC will be dumping over here Z will be dumping over there is a conflict.

So, those things has to be actually avoided right? So, sorry so now, what we will do? I think I slide missed in bus. So now, what will do? Now basically we will take this instruction load R 1 M. So, R 1 is register user general purpose register, M is a memory

location. So, what will happen you read the value of memory location M, whatever variable value is this is operant has to be loaded to R1. So, that is what are going to do.

So, first we will face this instruction, decode it and then execute we will see all this step-in details.

(Refer Slide Time: 22:01)

Single bus organization: Instruction Execution

1. PCout, 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 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 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. The ALU adds the constant with present value of PC (fed through the CPU bus). Control Zin enables loading of the ALU (i.e., $PC + constant$) output to register Z.

2. Zout, PCin, WMFC

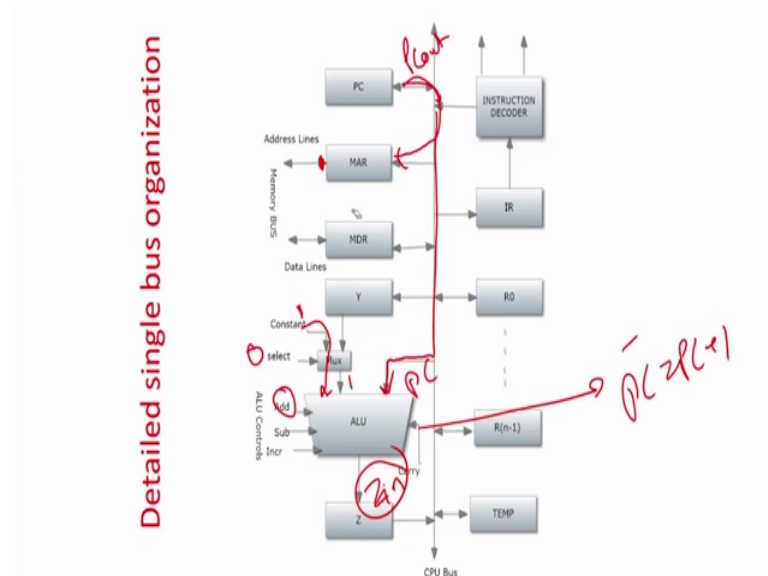
In the second control step the updated value of PC that is in register Z (1st control step) is loaded into the PC; this is achieved by control signals Zout and PCin. Also, in this step we wait for memory to respond i.e., by signal WMFC. Once MFC is high the value of memory to be read has been loaded 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, first what is the first step? That you have to as I already told you I the summery, the first stage actually will involve writing, because the 1 2 your you want to basically first stage is you want to fetch this instruction. So, which in which memory location load R 1 n is there that, been actually known by the program counter because, program counter always points to the next instruction. So, only the program counter can tell you where basically this instruction is at present in.

So, what happen if program counter value will be loaded into the memory address register, and you have to make the memory in the (Refer Slide Time: 22:36) mode. So, that is what is been said. So, program counter value is PC out, PC out means the value of program counter will be dump to the memory bus, now where it will go it will go to the memory address register e; that means, the value of program counter will go to the memory address register; that means, now you are pointing to the next instruction, which has to be fetch you are making it READ that the memory is in READ mode.

Now, this part actually corresponds to PC out, memory address register in and READ this 3-control signal basically specifies that, I have to read next instruction from the memory which is pointed by P C. Now, you can see it is making select 0 add and Z in. So, what does that mean if select here means select 0 means, will again see it is correspond to multiplexer, select 0 means you have to add the constant and not the value of Y, add here means what? Add here means the ALU is in add mode at Z, Z in Z in means the register Z will take the value in from the ALU. So, let us look at it what does it mean, we will be revisiting it many times. So, what is was saying? It will say that PC out.

(Refer Slide Time: 23:45)



So, PC out means, it is going to give value over here in the bus, it was said memory address register in; that means, the value of program counter is going to the memory address register at the same time it saw saying select the 0, if the select the 0 means the constant will be fail over here. So, the memory address register PC value is going to the memory address register, as well as it is directly coming to the ALU by this part because, it is in the bus PC of PC program value PC counter is in the bus.

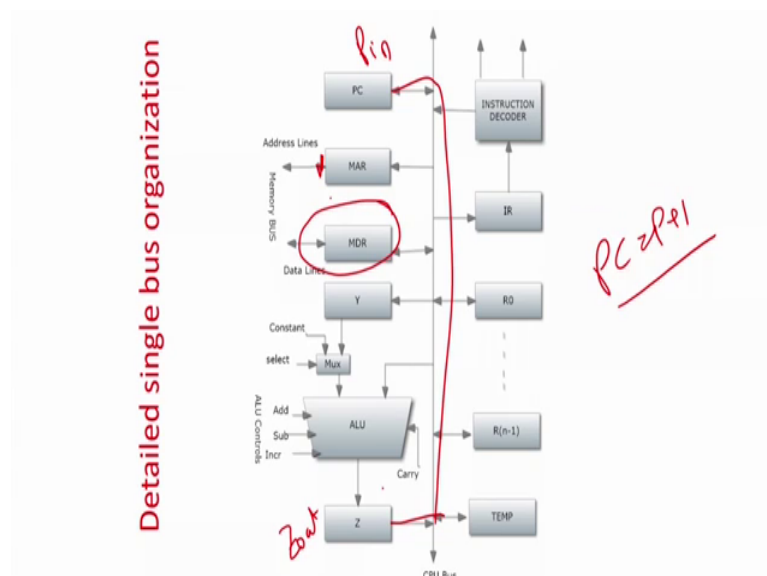
So, this is what is the case and we are say that is why select is equal to 0, and at the same time we are saying that add; that means, you are going to add it and we also saying Z in; that means, what? Your PC out value is going to the memory address register, at the same time the or input to the ALU is P C, this part you are going to say select 0. So, your select

a constant, if the instruction size is a 1. So, you are going to get put it 1, and you are going to get as 1 and Z in; that means, the output of ALU which is nothing but PC equal to PC plus 1 is going to the said. So, you are executing 3 basic things, you are incrementing the value of program counter by the ALU, loading the value in Z and also you are loading of program counter to the memory address register. So, that the instruction can be fetch, at the same time memory also has been made in a READ mode. So, this are the signals which corresponds to the first stage.

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 equal to PC plus 1 is memory is in register Z 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 Z has, if you look at the initial last slide then Z had the value of PC equal to PC plus 1, but other than it was Z in.

Now, I am making a Z out and PC in; that means, the value of Z will go to PC program counter, via the bus because Z out and PC in and we are waiting for W F M C. So, are waiting will be memory says that, I am ready and whatever you asked in the first stage it is been dump 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 Z 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 part. So, PC equal to PC plus 1 of the constant, is loaded into the PC and also I am waiting for f W F M C; that means, if the signal is one; that means, what the value of that memory location, where the instruction was there in loaded into the memory data register are 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 Z, which actually nothing but PC equal to PC plus 1 and it will be read to the PC by this bus. So, the 2 signal Z Z out and PC in accomplishes that and we are rating to basically, our memory is ready. So, that is over here.

(Refer Slide Time: 26:56)

Single bus organization: Instruction Execution

3. MDR_{out}, IR_{in} (M) (M)

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. $IR_{out}, MAR_{in}, Read$ (M) (M)

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.

5. **WMFC**

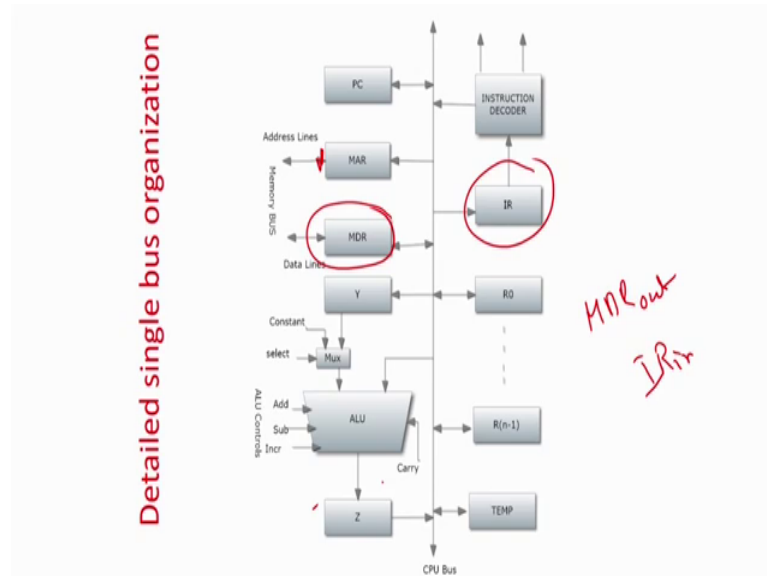
Wait for memory to respond by signal WMFC. Once MFC is high the value of memory to be read has been loaded into the MDR. The MDR contains operand that is to be loaded into R1.

6. $MDR_{out}, R1_{in}$

The value present in the MDR (i.e., the operand) is loaded into R1; $MDR_{out}, R1_{in}$

After that what happens? Now the memory is ready, now what you have to do you have to load it into loaded, 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 M D R out and I R in it will serve the purpose basically, what is being done in the 4th instruction, 4th 4th step this 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 instruction, 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 R 1 in that is whatever is present in the memory location that is load R 1 in; that means, in memory location M whatever value is, there data is there you have to load it into R 1.

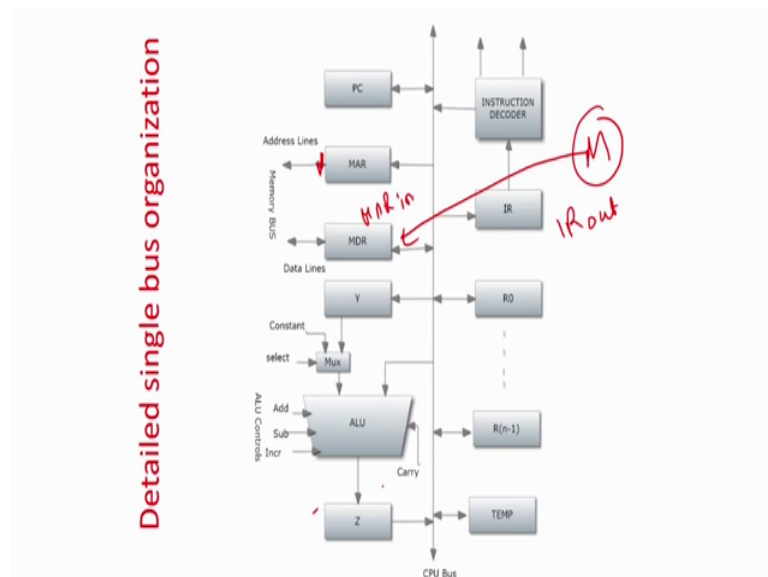
So now, what? So, you have to take this value M and loaded into the memory address register, because that part is going to tell where the operant exist so; obviously, first instruction will be I R out because the value of the instruction, which is present in the instruction register has to be given into the bus, and then your memory address register basically, we read the value from M, now what there some settle 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 R 1 and M.

We need to load only this part of the instruction I R to the memory address register because, we do not require the off code would not require the address for R 1. In fact, the instruction will take care of that. So, that part slightly obstructed, because we just require

to keep this part, which is a very simple digital operation already this part has to be loaded into the memory address register. So, we will do that and then, again wait till your memory op read operation is complete. So, you can see the R 1 has the entire instruction op code R 1 address and address for M, that is this 3 part basically as I have told you, but. In fact, you are going to only use this part, to be loaded into the instruction memory ad address register. So, that clicking part we have drop over here.

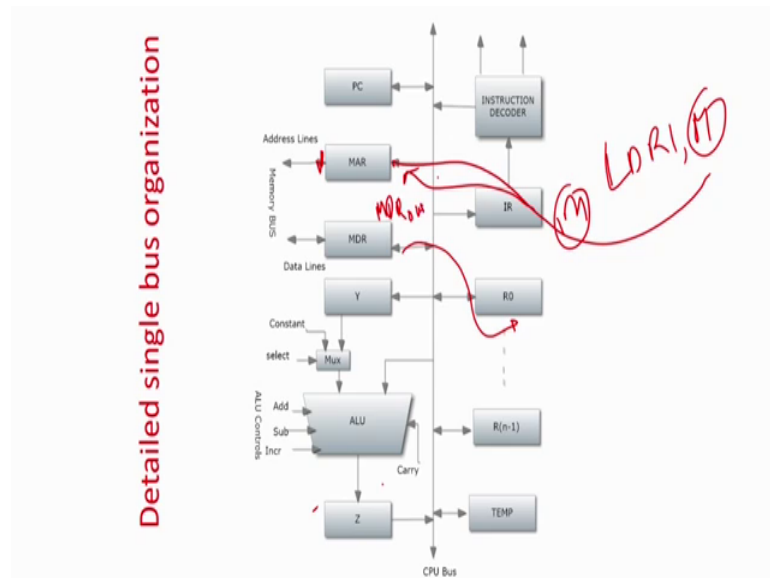
Then at 5th stage, we wear till the memory says that, I am done with it. So, once the memory says that I am done with it; that means, the data this M data is now loaded in to the memory data register. So now, what you will have to do you have to just term, the memory data register value that is M D R into R in that is R 1. So, M D R out means whatever data is available in the M memory data register, it will out and it will be present in R 1. So, in 6 stages I complete the instruction, let us quickly look at the c controls in this figure again. So now, what happen the instruction recode by an instruction register, we will we will load the value of M. So, this one will go to the memory data register.

(Refer Slide Time: 29:55)



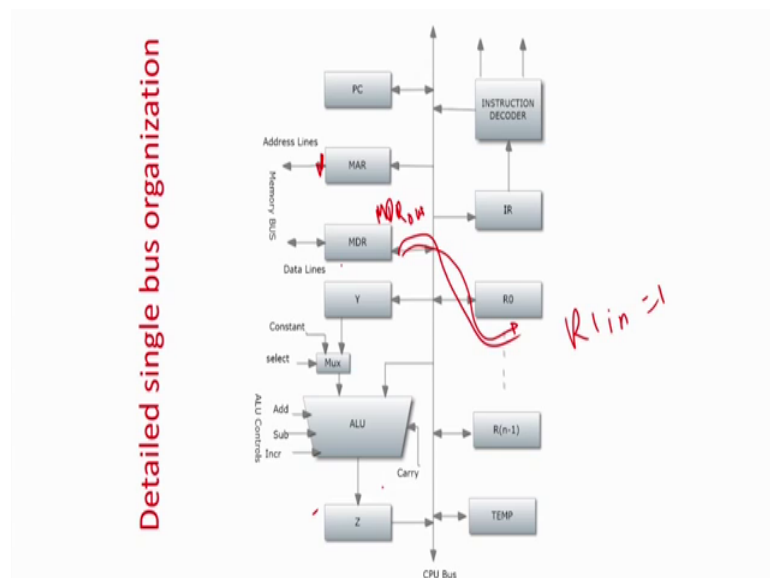
So, of course, it is I R out which (Refer Slide Time: 29:59) at use of notation, which I am not going to take the entire I R only the M part of I R and M D R in and memory is in READ mode, wait for some amount of time that is phase 5 and then once it is done, the data has come sorry the just the small 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, where the instruction register. So, I R out M D R 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 have go to register R 1. So, as simple as that now we are going to make an M D R out of course, all these things are now all become 0 that I have all this thing has to be go off. So, M D R out will be 1 and also R in R 1 in n is 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 this register completes. 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 dump to the memory data register and then the value M D R out, will take the value from M D R and it will load to R 1. So, your job is done. So, in 6th micro instructions and the corresponding control signals, what we have done we have shown how a complete instruction is fetch decoded and execute. So, this was about the instruction load R 1 M.

(Refer Slide Time: 31:29)

Single bus organization: Instruction Execution

ADD R1, R2

Task of the instruction: ADD R1, R2 is to add the content of register R2 to the content of Register R1 and load the result into R1.

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+ constant) output to register Z.

As I told you, we will look at if an 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 R 1 R 2, in this case the value of R 1 will be added to R 2 and stored in R 2 again, as I told you the first stage program out PC out memory address register in READ, select ADD and Z in this stage.

(Refer Slide Time: 31:50)

Single bus organization: Instruction Execution

2. Z_{out}, PC_{in}, WMFC

Add R1, R2.

- 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 Z_{out} and PC_{in}. 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).

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. R2_{out}, Y_{in}

- In the fourth control step the value in R2 is loaded into the register Y (by control signals R2_{out}, Y_{in}).

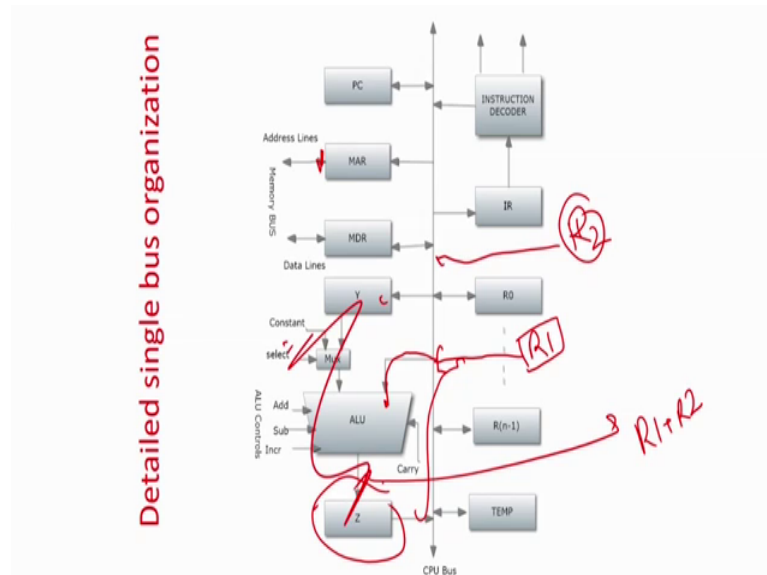
Program counter incremented stored in Z, move to PC and wait for W F M C, once it is done then M D R out and put in I R that is first stage points to the memory location, where the data is to address is where the instruction is stored plan, to increment the (Refer Slide Time: 32:04) program counter really 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 M D R and put it in.

So, these three stages basically correspond to any instruction fetch, whether it is add R 1 R 2, whether it is load R 1 M 1 the first 3 stage, basically this one this one and this one will remain content 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 compare to the pervious case. So now, in this what I am having. So, next instruction is in add R 1 and R 2. So, was add R 1 and R 2. So now, what is going to happen.

So, how I will do? It basically there is one operate in R 2 one operate in R 1, both has to be added and the value has to stored in T 1 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 I R and put it into the memory address register and again read the memory not require, till everything is in the registers.

So, basically what it does, it says that R out 2 and put it in Y in. So, what is doing taking the value? Of R 2 and including 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 R 2 some R 2 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 R 2 value I will first stored it into Y, then what is will do then I will take the value of R 1, 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 R 2 I will store it in y for a temporary, then instruction y will have the value of R 2.

Then next is what I am going to do, next is I am going to just connect it to R 1. So, that is goes to the ALU directly, add in the mass select will be equal to 1 so, basically sorry. So, select will be equal to 1, if select is equal to one then R 1 will be directly face to the ALU to this part and Y which, is nothing but in your case in your case R 2 will be coming to the ALU and; obviously, in add mode.

So, here you will have the value of r 1 plus R 2. So, that is how I will do, I will store the value of R 2 to Y first then next stage I will directly connect R 1 to ALU by the bus, as simple as that select will be Z 1. So, I will have the value R 2 to R 1 which I have to store in a very temporary manner to register Z a then again, I will dump the value of Z to

register R 1. So, this will be the stage it will go through this is over all idea, now we will individually take this steps and clear 1 at a time, now let us again go step by step.

(Refer Slide Time: 35:10)

Single bus organization: Instruction Execution

5. $R1_{out}$, $Select=1$, Add , Z_{in}

- In this step, $R1_{out}$ is enabled which results in the value of register R1 being present in the CPU bus. The CPU bus is fed as the second operand to ALU. So in this case the second operand to ALU is the content of register R1. Also, the input to multiplexer $Select=1$, which ensures that the first operand to ALU is the value at Y (i.e., the R2 content). The control line to ALU is Add which configures the ALU to perform addition. Also, control Z_{in} enables loading of the ALU output (content of Register R2 (Y) + content of Register R1) to register Z.

6. Z_{out} , $R1_{in}$

- In this step the value present in the register Z (i.e., the result of addition) is loaded into R1. This is achieved by signals Z_{out} , $R1_{in}$.

So, next is R 2 Y in. So, already we have told R 2 means the value of register 2, is dump in Y in. Next one next already your Y, that is your if you look at in this is your ALU and this is your register Y and which is coming here, by a multiplexer this is your marks and this. In fact, is your marks. So, next is what I have done I have said R 1 out. So, R 1 was somewhat connected over here. So, I am making R 1 out. So, the value of R 1 is going in that direction already we have discusses, but again I am just redrawing.

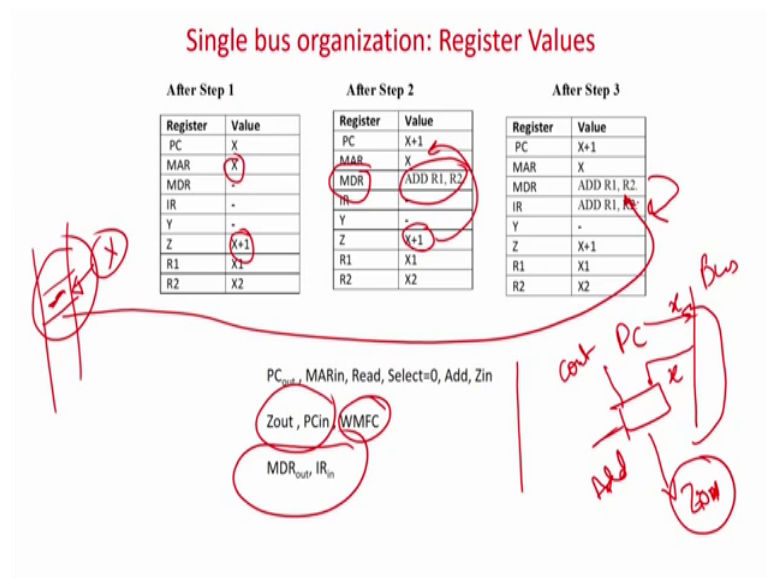
So, value of R 1 is going over here, which is our next operand which is connected to the ALU and now Y which is having the value of R 2, previously it is actually temporarily holding the value of R 2, now this one you have marks you have made it one. So, it is been fed over here. So now, it is your doing R 1 plus R 2 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 stored the value of R 1 plus R 2.

Now, what next stage this Z 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 R 1 will be R in R 1 in. So now, this will no longer be in this mode it will be going in 6th stage that is will be going in reverse mode. So, it will be R in and it will be Z out.

So, value of Z R 1 and R 2 via Z will go to R in and your job is 6 data. So, in the 6th stage Z will be out and R 1 will be taking in. And so, this are the 6 signals, which I have taken care add R 1 and R 2, if you compare load from a memory location and load from R 1 and R 2, these are 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 operant from memory location M, but in this case what happen we have never gone to memory for the second time, because everything was available in the register right?

(Refer Slide Time: 37:02)



So now let us quickly have a look at, what are the different register values where we are executing this instruction. So, first instruction like as I told you, the first 3 instruction 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 3 one actually reads the value of the instruction from the memory to the instruction register, this correspond to instruction 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're 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 your main bus, PC you are dumping over here, which has the value of X and this is your ALU. So, X is going over here and you are making 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 saying that and the output is also to Z in.

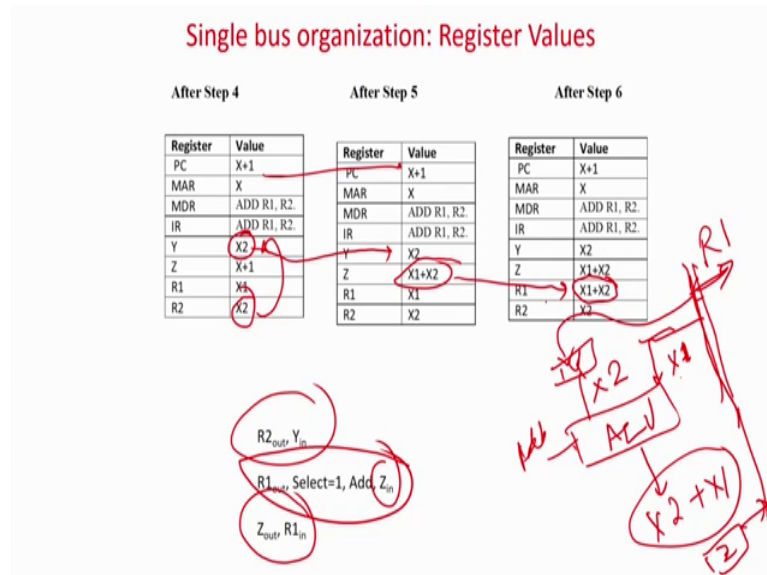
If it is Z in so, Z will be (Refer Slide Time: 38:44) by the value of the output of the arithmetic logic unit. So, in that case you are assuming the constant to be equal to 1. So, you are going to get Z equal to X plus 1, that is PC is incremented by one and we are assuming that, register R 1 and R 2 are having the value of X 1 and X 2, which will be 5 6 7 10 11 14 41 which one into 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, we program the memory location X we will have has 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 Z will be now out, and it will actually write the program counter in a reverse manner. So, of course, the value of Z will be written to the program counter PC is equal to PC plus n and also, we are waiting for WMFC that is you have to wait, but till the memory saturated, saying that the data has been peacefully rate from the memory to the memory data register. So, the instructions add R 1 R 2 was available in the memory. So, if you look at the memory. So, here the instruction was there, that is ADD R 1 R 2 and this was actually memory location M.

So now it has come over here, that 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 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 M we are making it X this memory location X which is having the instruction, add R 1 R 2 is being fetched memory location X and it is dump 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, was the stage so, next what you are doing? You are saying that R out is equal to Y in already we have seen that, this is you ALU and this in your R 2, R 2 should be fetch to Y means a temporary register over here. So, I have here the marks will set in such a manner this Y. So, that it will not take a constant it will take the value of R 2 as a operand. So, in this second stage you are saying R 2 equal to Y in. So, R 2 basically had the value of X, which will be dump to memory location Y I am sorry register Y. So, you can see this has been dump over here.

So now, this is actually having the value of X 2, this is Y and another if you remember, this part of the ALU directly connected to your bus. So, in the second stage the value of R 2 is now dump into Y, which is nothing but X 2 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 R 1 out select 1 ADD and Z in. So, you are saying R 2 out. So, if you are saying R 2 out means what sorry R 1 out; that means what? This is R 1 which is now feeding the bus.

So, it is having the value of R 1 R 1 is nothing but your X 1 over here. So, R 1 is having the value of X 1. So, X 1 is going X 1 is coming to the ALU, as the 2nd operand. So, this

is your X 2 sorry R 1 sorry this is X 1. So, X 1 is coming over here, correct and ALU already having the data in the other operand X 2. So, output will be X 2 plus X 1, right? There is R 1 and you are making select equal to 1, select is made to be 1, the multiplexer will take the value of Y that is X 2 to the ALU constant will not be added, add is the symbol that is going to be add it. So, that is the added now and the output that is X 1 and X 2 will be dump to Z in because, where it is connected to register which is Z. 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 Z is there in the value of X 1 plus X 2 Y is temporarily holding the value of X 2 which is continued over here, and the R 1 to the register R 1 that is again having the value of X 1 is feeding the ALU directly from the bus, without any primary without any temporary register last stage what we do, we say Z out equal to R 1 that is, now the value of Z has to be the value of Z has to be fed to register R 1.

So, what you will do you will say, Z out and R in. So, in this case the R 1 which was initially having the value of 1, is update with the value of X 1 and X 2. So, that is these value actually constant and your job is done basically. So, again you can very quick look at this figure. So, I am cleaning it off then it is very simple you can understand the illustration ok.

(Refer Slide Time: 43:26)

Questions and Objectives

Q1: Draw the diagram of a CPU with single bus organization. In that design explain the need of each component. Write the control steps for fetching an instruction. Briefly explain the actions in terms of control signals carried in each step.

Q2: Consider the CPU with single bus organization designed in Question 1. Consider the instruction `LOAD R1, R2`. Write the control steps for fetching, decoding and executing the instruction. Briefly explain the actions in terms of control signals carried in each step involved in fetching, decoding and executing the instruction

- **Comprehension:**
Explain:-- Explain the generation of control signals that is driven by the internal organization of the processor.
- **Synthesis: Design:--**
Explaining the design of complete control steps to execute the instructions like ALU operation, Data movement, etc.

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 own 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 stage 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, require 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 R 1 and R 2 load R R 2 to R 1, whatever value is available in R 2 you will be going it t R 1. So, generate all the control signals and all this steps for this.

In fact, just we have seen one instruction for add R 1 and R 2 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 tried 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 of this module, will be looking as 3 bus architecture and then you 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 R 1 to R 2 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 this steps for load R 1 and R 2 then, you can explain this single bus organization, as well as you can say exact steps require to fetch decode and execute instructions. So, with this we come to the end of this unit and next unit will be going into more depth of some slight slightly more complicated instructions, like jump and conditional instructions and how for generating controls for that.

Thank you.