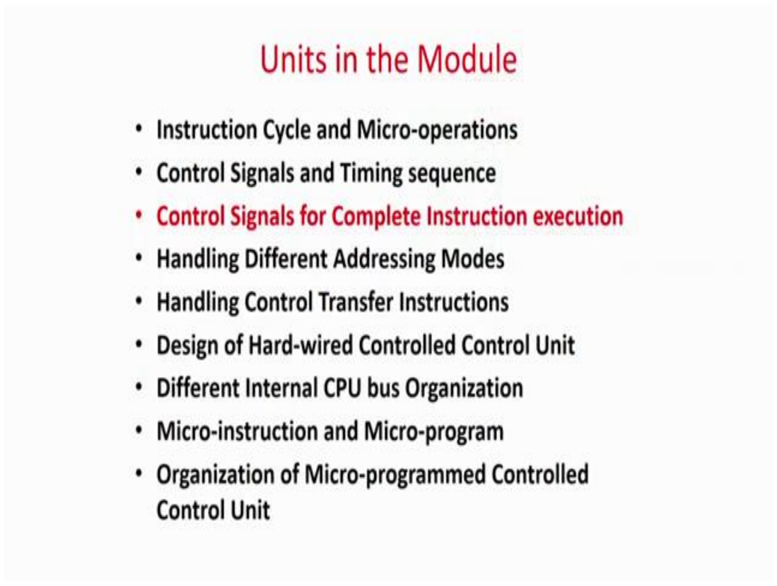
Computer Organization and Architecture: A Pedagogical Aspect

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 concerned with control signals, for complete instruction execution.

(Refer Slide Time: 00:39)



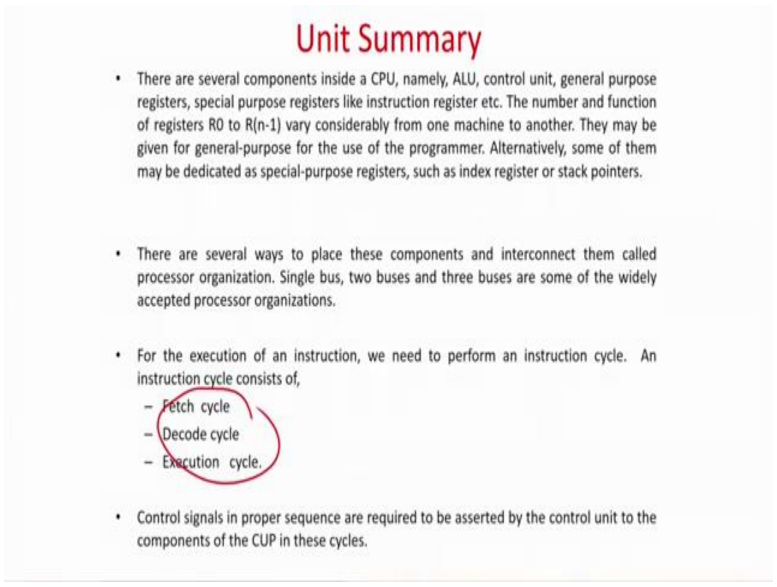
Units in the Module

- 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

So, if you look at flow, in last basically two units we mainly covered about what is a 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 required and what are the timing diagrams required involved in generating 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 a 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 content of today’s unit.

(Refer Slide Time: 01:23)



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.

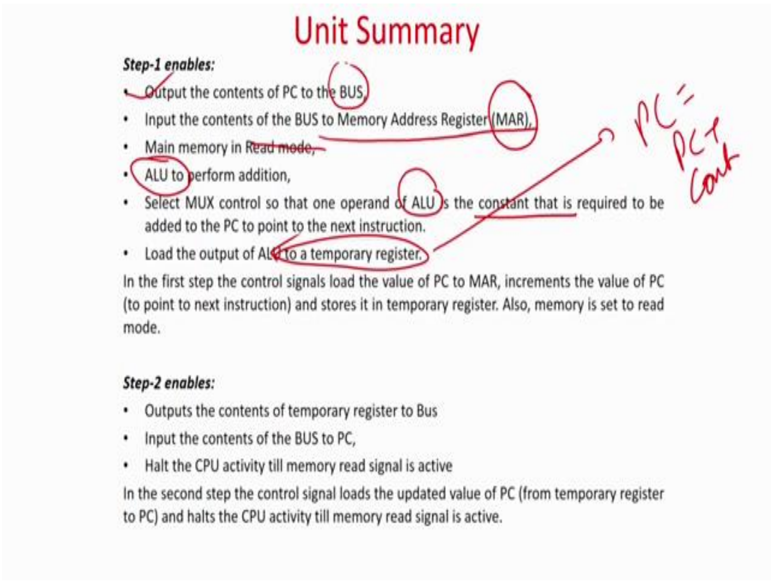
Basically, what will be covering in this unit, the unit summary will basically we will have quick revisit, as you can see in the first point. We will be quickly revisiting what is a single bus architecture? Because that is what is mainly we are dealing with all the examples and most of our study is basically 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 interconnects, all this things we will have a quick reconnect recollect. Basically, and then because, as we know there are multiple bus structures 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 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 required 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 it differs 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 instructions, we have already seen in the last unit, that we require more cycles I mean execution in terms of control units and etc.

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 types of micro instructions, for a given instruction cycle different will be the 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.

$PC = PC + Cont$

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.

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 the 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 requires to do that, basically what happens first, you see it will output the it shows that output the contents of the PC to the BUS, there because of the program counter only will point out, that which instruction has to be executed, then the contents 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 loaded to the address it will be loaded to the memory buffer register or the memory data register and. In fact, also in this instruction you will also have to increment the program counter to point to the next instruction. So, what we do? We also instruction 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 word then, you will add 1 and in other case, if it a 2 word instruction we 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, then you select the mux control that is one operand of ALU to the constant, that is you are going to add program counter whose value is in the bus to a constant.

So, we have already seen in the last class that basically in the ALU one of the operand 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 adding it to the present value of the program counter; we will point to the next instruction.

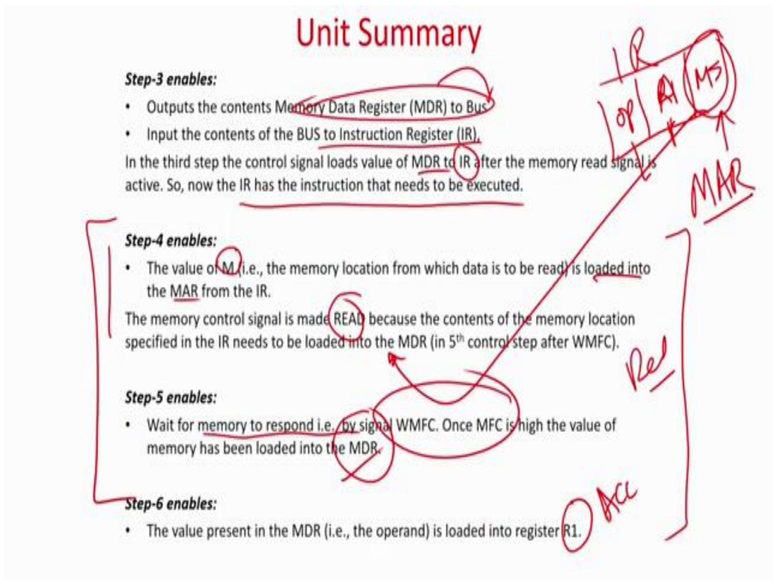
Then we will load the value of ALU to a temporary register. So, the temporary register now has the value of $PC = PC +$ this content. So, what happens in the first stage, basically what we have done if you can look at the last comment, what it says? In the first step load the value of program counter to 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 read 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 the bus and input the content of the bus to the PC, because at present the temporary register is holding the value of $PC +$ constant, you have to dump it to the bus and the bus value will be dump to PC. So now, the program counter will have the value of $PC = PC$ + 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 + constant.

Then input the value of bus to PC. So now, PC is incremented because we no longer require the 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 ready, that it has dumped the content of the memory value, which was pointed by the memory address register to the memory data register and whenever it say that I am ready, the second stage starts, in second stage what will sorry third stage starts, whenever the memory says that I am ready. So, the memory data register value will be content will dumped 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 happened, the instruction is now loaded into the instruction register, from the memory data register. So, the third step 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 said 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 IR has the instruction, that need to be executed. One important point to be note that step 1 and step 2 and step 3 are actually implying the memory fetch, that is instruction fetch. Instruction fetch means you load using the value of the PC, 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 constant 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 IR register, instruction register via the bus, this will be same this three stages or the three micro instruction and the control signals 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 it pertains to? And what is to be done? For example, the value of M 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 opcode 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 called 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 will happen? 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.

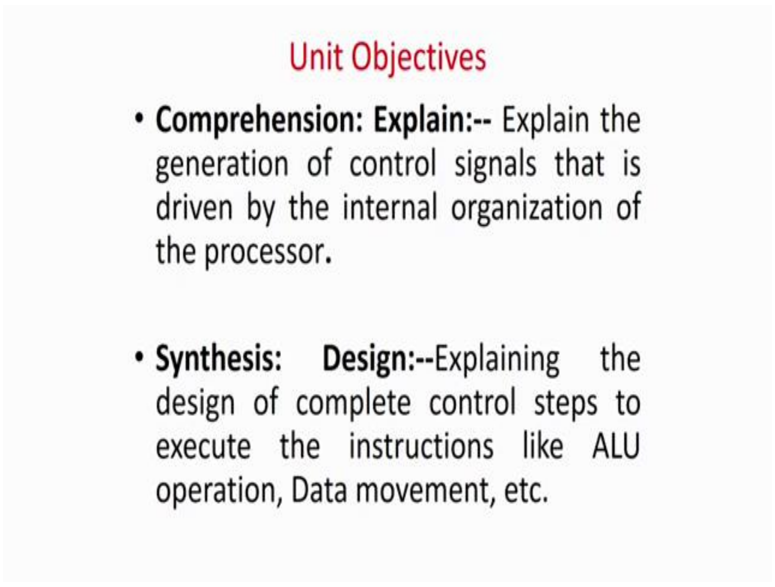
So, you can see 4 step 4 says that, the memory location memory is made read, because you have to read the operand and the value of M is loaded into the memory address register, from the IR; that means, the IR is now being decoded and it is going to execute. So, before execution, it is decoded that is 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 in fact 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 mode 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 now in this first 1 2 3 stage you are reading the instruction, from step 4 5 and 6 in these steps are involved with reading the operand 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 we are loading in step 5 is a data and not an instruction. So, will not go to IR basically, it will be added with if the opcode is load. So, basically it will actually load the value, in accumulator or $R1$ in this case it is called $R1$, but our example was for 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 step 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 or the memory buffer register.

Step 5 clears, that memory is searching if memory is ready, data is available in the memory data register and then in the 6th stage, you will dump the value of the memory data register basically, which is containing this operand into the accumulator or register $R1$ so. In fact, 1 2 3 again I am repeating this is actually summary, which we 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 type.

(Refer Slide Time: 12:20)



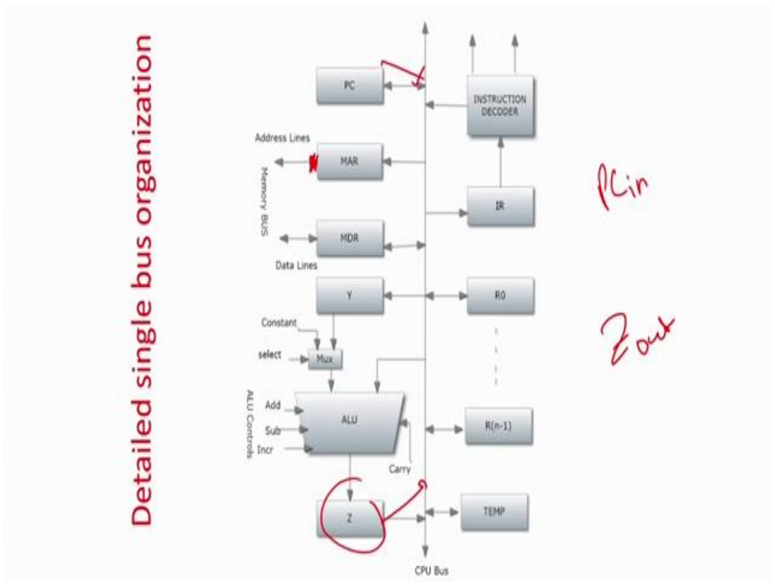
Unit Objectives

- **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.

So basically what are the basic objectives of the unit the first is a comprehension objective in 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, given a 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 there is 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 compared to the previous unit. So, this is program counter this is your single bus, then you have a memory address register which is connecting. In fact, actually it is a slight mistake basically. In fact, the memory address register is a unidirectional bus.

So, it should not be there it's a unit directional bus so in fact, which is connecting. So, it is giving the input to the memory. So, whenever 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 data register will go to the memory.

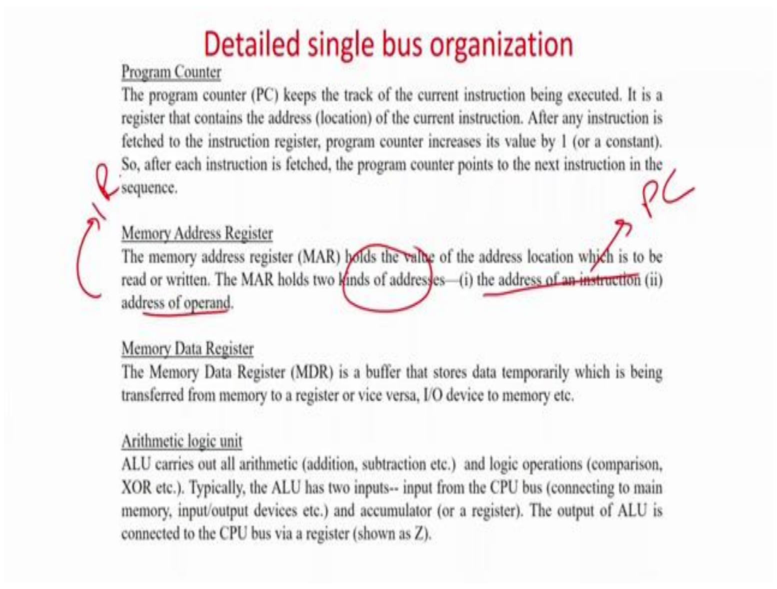
This is your instruction decoder which is basically decoding instruction and generating the control circuits, this is very, very important and who is going to feed to the instruction decoder? The instruction register. The instruction register will again get the value from the bus and it will go to the instruction decoder, which will decode in terms of control signals based on the instruction to be executed, you can see there are different registers $R0$ to Rn, they are 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 processor you are using.

So, this is the general purpose registers they are 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 takes the data from the bus, which general is an operand, but the left-hand side you can see, it can take data from a register Y the special register sometimes we call it 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 has select line.

So, now 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 it is a constant? where is basically you have to increment the value of PC. So, whenever you have to increment the value of PC, 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 that will again tell you based on the operation type, is an add instruction, load instruction, and 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 multiply 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 your the length of the instructions. So, that $PC = PC +$ constant; that means, it 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 this part 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

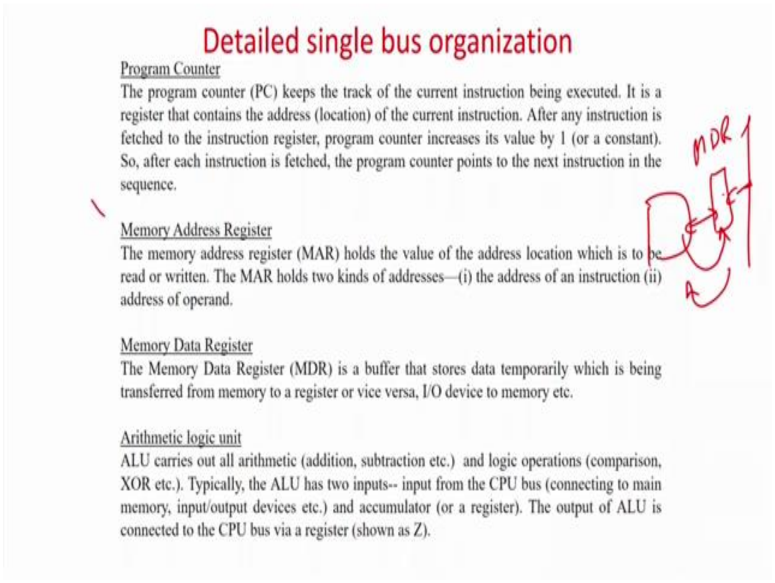
ALU carries out all arithmetic (addition, subtraction etc.) and logic operations (comparison, XOR etc.). Typically, the ALU has two inputs-- input from the CPU bus (connecting to main memory, input/output devices etc.) and accumulator (or a register). The output of ALU is connected to the CPU bus via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

So, again we will keep on revisiting this that is not a problem. Now, let us go to this in details. So, there is a program counter. So, this is your 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 and it is incremented.

So, based on that that instruction is fetched and the word stops 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 loaded into the memory address. If there is a store instruction, then when you are saying that store accumulator to memory location M, 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 loaded to the memory address register. So, basically memory address register tells that basically, means which memory word has to be read or written.

So, therefore, actually the two kind 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 your 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).

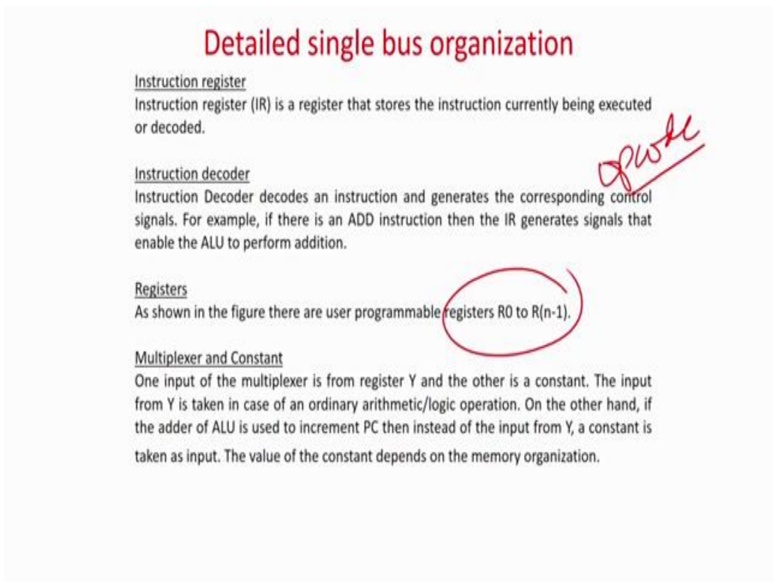
So, basically immediately you cannot read and write from the memory. So, you give a read command, then you have to wait for certain amount of time then the data from memory will come to the memory buffer register, and then it will say that I am ready that is data has been given, then only the bus and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

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

So, that it can be given to the respective place. So, the output of ALU is stored in a register called IR, which holds it for some time before you take the value and the ALU can be reused. Because if I don't store the value of register, if I don't store the value of the output of ALU in the IR, 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 got in the last clock part, but if I don't store over here, then actually some new value comes over here, there can be a corruption 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).

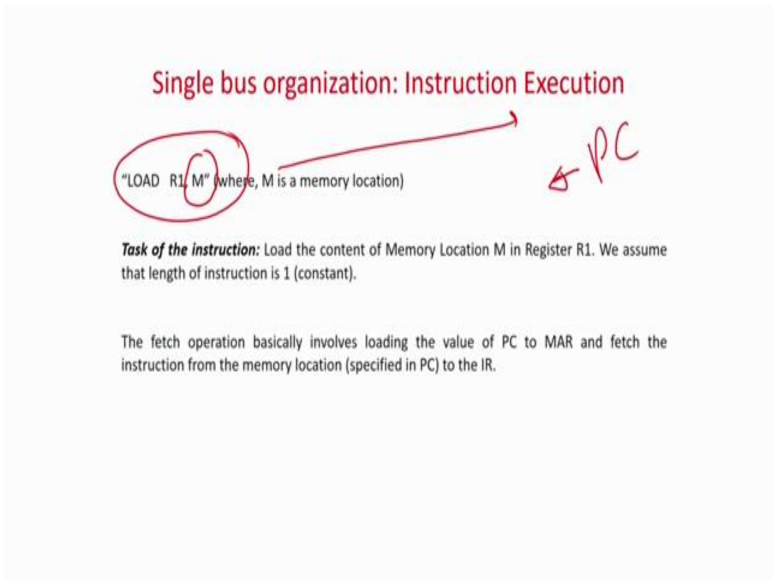
Multiplexer and Constant

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.

Instruction register, nothing to tell more it take the instruction and knows what to do. Instruction decoder, basically it's a normal decoder circuit it will take the instruction, because all instruction has first is opcode. So, will take the opcode and accordingly generate control signals, corresponding to that opcode. Registers as I told you shown from $R0$ to $R(n-1)$, they are 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 focus. So, what it says is this part. So, this is your ALU and this is your register, basically this is your mux and this one. So, basically as I told you, the ALU can either take input from Y then it will be an operand, or it can take a via multiplexer it is constant. So, if it is constant means it is for the incrementing of the PC. So, that is what has been told over here, that multiplexer and constant what are the meaning? Basically, instated of Y if it is an operand, it will take the value from the constant which is for increasing 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).

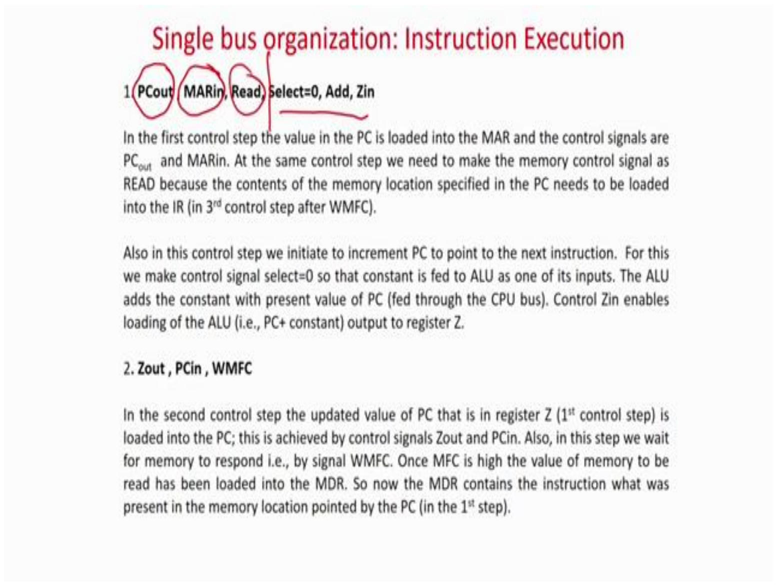
The fetch operation basically involves loading the value of PC to MAR and fetch the instruction from the memory location (specified in PC) to the IR.

So, with this basic background in mind, we will now see for different 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 IR has to be dumped to the CPU bus, then you will write Z_{out}; that means, the value of IR would be dumped to the bus for example, the value of PC, 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 discussed in the last unit you cannot have simultaneously Z_{out} and PC_{out}, in this case PC will be dumping over here IR will be dumping over there is a conflict.

So, those things has to be actually avoided correct? So, sorry so now, what we will do? I think a slight mistake in bus. So now, what will do? Now basically we will take this instruction $LOAD$ $R1$, M. So, $R1$ is one register 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 operand has to be loaded to $R1$. So, that is what are going to do.

So, first we will fetch this instruction, decode it and then execute we will see all these steps 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 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. 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).

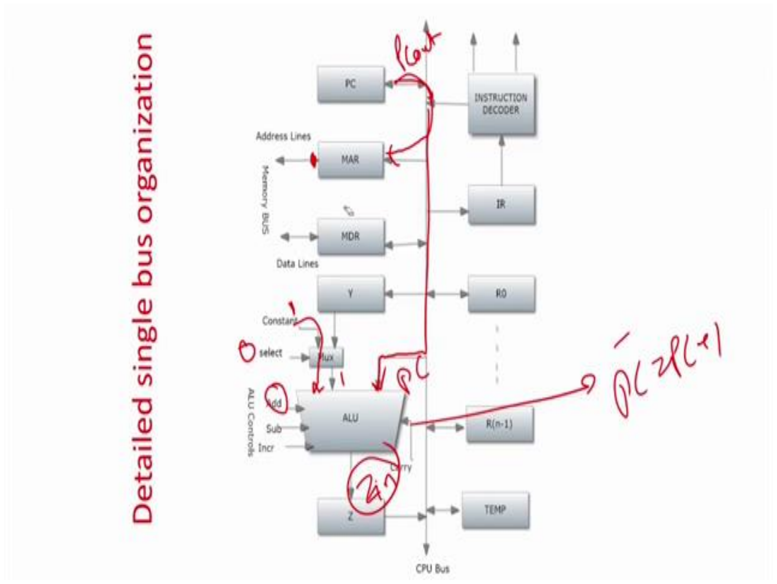
So, first what is the first step? That you have to as I already told you in the summary, the first stage actually will involve writing, because 1 2 you want to basically first stage is you want to fetch this instruction. So, which in which memory location load $R1$ n is there that has been actually known 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 happens? The program counter value will be loaded into the memory address register, and you have to make the memory in the read mode. So, that is what has been said. So, program counter value is PC_{out}, PC_{out} means the value of program counter will be dumped to the memory bus, now where it will go it will go to the memory address register in; 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 correspondence to PC_{out}, memory address register in and READ this 3-control signal basically specifies that, I have to read the next instruction from the memory which is pointed by PC. 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 corresponds 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 and Z_{in} means the register IR 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 is saying 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 saying memory address register in; that means, the value of program counter is going to the memory address register at the same time it was saying select is 0, if the select is 0 means the constant will be fed 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 path because, it is in the bus 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 other input to the ALU is PC, 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 $PC = PC + 1$ is going to be set. So, you are executing 3 basic things, you are incrementing the value of program counter by the ALU, loading the value in IR and also you are loading the value of program counter to the memory address register so, that the instruction can be fetched, at the same time memory also has been made in a Read mode. So, these are the signals which corresponds to the first stage.