

Computer Organization and Architecture A Pedagogical Aspect

Prof. Jatindra Kr. Deka

Dr. Santosh Biswas

Dr. Arnab Sarkar

Department of Computer Science and Engineering

Indian Institute of Technology, Guwahati

Unit – 2

Control Unit

Lecture - 16

Control Signals and Timing Sequence

Hello, and welcome to the second unit on control signals and timing sequence, which is the second unit on the module on control block of the CPU. So, in the last unit, basically we chose the first unit on the control unit module, we have discussed that basically how for a given set of instructions, what are the microinstructions involved in executing that macro instruction, and what are the basic kind of control signals required to do it. And we got a very broad idea that how these macro instructions are broken down into microinstructions and they are executed.

Basically in today's module, now we will see basically for given is each of the microinstructions, what are the control signals required, exactly which block of the CPU generate those signals, and what are the timing sequence for that. And we will be understanding that in a more depth or a more what do I say that more in a digital fundamental manner in which digital design fundamentals using timing diagrams which signals are generated by which blocks, what are the inputs to the registers in that manner.

So, in fact, today we will see how microinstructions are basically executed, what are the control signals in terms of digital design, and what are the timing involved, you will understand the timing sequence using timing diagrams.

(Refer Slide Time: 01:44)

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 it today, we are in the second unit of this module, which is on control signals and the timing.

(Refer Slide Time: 01:46)

Unit Summary

- The control unit is responsible for generating signals for data flow control within the components of the CPU, memory and I/O devices.
- Also, the functionalities of the different Arithmetic and Logic units are determined by the control signals.
- The control unit takes input from the flags (representing the result of the previous instruction(s)), instruction register (Opcode), control signals from the control bus and internal signals.
- The output of the control unit comprises signals that determine which functionality (out of all possible ones) is to be performed by a sub-module and what would be the data flow.
 - For example, the control signal to the ALU determines if addition or subtraction is to be performed on the inputs.
 - In case of data flow, the control signals decide whether a register is to take data from the bus or write into the bus.

So, as we are going in a pedagogical manner, so for a pedagogical form, so let us first look at the unit summary. So, basically we already discussed that the control unit is responsible mainly for generating the signals for data flow within the CPU, there is internal to the CPU data transfer, data transfer between the CPU and the memory or the IO devices. So, basically in this unit, we will be covering what type of signals are

required to do that and mainly we will be taking a very simple architecture that is a single bus architecture.

Also we will see that how basically the different functions of the arithmetic and logic block like whether it to be added, to subtract to, go for a chip, how signals are generated at by the instruction register, and how ALU is controlled by that. Then we will see basically we will also see in a black box manner that what is the control unit, what are the inputs it takes from, it take basically input from the flag registers, it will also take inputs from the opcode that is from the instruction register. If you take an instruction which is noted in the instruction register basically the Opcode decides that what the functionality now the CPU or the control unit has to do. So, basically the control unit will take some inputs from the flags.

It also because for example if you have a jump instruction, so you need to know what are the previous values of the flags. So, it has to read the values of the flags. Then you will also see basically that of the main heart of the control unit to function is the input from the operations that are required to be performed by the opcode of the instructions. And also we need some other of inputs to do some functionalities like the control signals which will be coming from the may be the memory, but because it when you are sending a data to the memory or receiving the data from the memory. There is a handshaking signal involved which will actually the signals will be coming from the memory block to the actually your bus.

So, basically if you look at it, we will see what are the inputs and outputs of the control unit and what the output manner, where the data comes from and the basic formats will be looking into. Then also as I told you this was the inputs. Also we will be looking at the outputs of the control unit actually that generates signals which will instruct that where the data has to move from whether it is from one register to another or whether the data has to go from memory to ALU or vice versa.

So, we will see basically what are the outputs of the control units, and how it basically determines the function will be like whether there should be an addition, whether there should be a data transfer between memory to register, or from register to register etcetera. So, we will study the input output functionality and the signals involved in a controlling that is one of the main part we will study.

(Refer Slide Time: 04:17)

Unit Summary

The groups of input signals of control unit are from the following blocks:

- Clock: The clock is used for synchronization of all the modules of the control unit. Also, the clock determines the minimum duration of time (or frequency) that can be considered as an atomic unit. Micro-operations that constitute an instruction are executed in one clock pulse. This is sometimes referred to as the processor cycle time, or the clock cycle time.
- Instruction register: The instruction fetched from the memory is loaded in the instruction register. The opcode of the instruction determines the sequence of micro-operations that are required to be executed.
- Flags: Flags store the status of the processor and the outcomes of earlier ALU operations. The values of flags are used in control transfer instructions.
- Control bus: The control unit sends signals to different modules of the CPU (using control bus) for selecting their functionality. Similarly, the modules also communicate to the control unit by sending control signals through the control bus.

Then basically we will see, what are the very important chunks of signals for that? Of course, very important chunks of signals will be as I told you the instruction register. Instruction register which will consist of the instruction is the main heart that will actually command the control unit that this is the opcode of the instruction now you have to do this, for example add. So, the opcode for the add, which will represent in the instruction register will ask the control unit to generate the signal, so that the ALU goes to the addition mode. So, in fact, so the in instruction register is a main command mode for the control unit.

Then of course, as I told you that if you have to use a jump instruction or sometimes you have to use the values of the flags, which was said by some other old earlier instructions may also decide on the functionality of the control unit that is for example, if you want to jump on a zero. So, you have to also look at what are the results of the previous instructions. So, the flags are also very important source of signals control signals as input to the control unit.

Of course, then there is something called the control bus because not only from the flags and the instructional registers also the control unit will depend on signals from the memory from some I O devices etcetera. So, the control bus basically will take care of all this. So, it will shift the different signals from other than the central processing unit like memory I O devices and the control unit has to act on that. For example, if you want

to interface with a keyboard, when the keyboard is ready to send a signal or ready to then only the memory will be reading your data from the keyboard so in fact the control bus will take care of that.

And finally, there is a clock which is actually the whole synchronization part of the entire control unit or the central processing unit. So, basically as we know it synchronizes all the modules in the control unit, and in fact we assume that you know in one clock pulse one microinstruction is occurred or I can be processed.

(Refer Slide Time: 06:06)

The slide is titled "Unit Summary" in red. It contains a bulleted list of control signal groups. The first group is "Control signals within the CPU: These are two types", with two sub-points: "Data transfer from one register to another" and "Select the specific ALU function for the current micro-operation for an instruction". The second group is "Control signals to other modules through the control bus: These are also of two types:", with two sub-points: "Control signals to Memory" and "Control signals to the I/O modules". Red handwritten circles and lines highlight the sub-points and connect them across the slide.

Unit Summary

- The groups of output signals of control unit are as follows:
 - Control signals within the CPU: These are two types
 - Data transfer from one register to another
 - Select the specific ALU function for the current micro-operation for an instruction
 - Control signals to other modules through the control bus: These are also of two types:
 - Control signals to Memory
 - Control signals to the I/O modules.

Then basically we say that if you want to chunk out that what are the two different basic classification of the signals the out these were all about the input signals like instruction register flag and control bus. So, what happens actually your control unit basically takes signals from the clock, instruction register then your opcode that is your I mean is I mean basically it takes signals as I told you from the clock instruction registers which is the main flag and control bus.

Using that, it basically generates an output signals. So, output signals can be classified in two types control signals which are inside the CPU like for example, you want to ADD two numbers. So, the ALU has to be commanded. So, data transfer from so you can select the specific ALU function for the current micro operation like ADD. So, on fact, it was saying that specific ALU function. So, in that case the control unit will generate

some signals which are required only for the ALU to function. So, it is an internal control unit signal.

Similarly, if you want to transfer some data register from data from one register to another, so you need not send any control signal outside a control unit. So, basically this is some kind of internal control signals. Of course, the control unit will also generate some controls for the external parts like your memory, your IO modules. For example, as I told you that you want to write something to the memory, you give some address and then you give the data, but they may have to wait then the memory said that I have already read the data, now I am free because reading the data from the memory buffer register takes some time.

So, in that case, there will be some control signals like it has to be generated saying that you start the read, the read signal then sometimes it has to wait till the memory has done the operation. So, basically some of the control signals will be generated which will go to outside the control unit like to the memory to the IO modules in that case they are drawn through a to the control bus.

Because when the control signals are internal to the CPU then or sorry in terms the control unit or the mainly the CPU when the signals generated by the control unit and reserved only for the CPU that you did not go to memory etcetera then they are internal. But if you are going outside the CPU, then you have to talk to memory you have to talk to other IO modules etcetera. So, in that case we go through a control bus. So, inputs here. So, basically that is the input output organization in terms of our control unit.

(Refer Slide Time: 08:12)

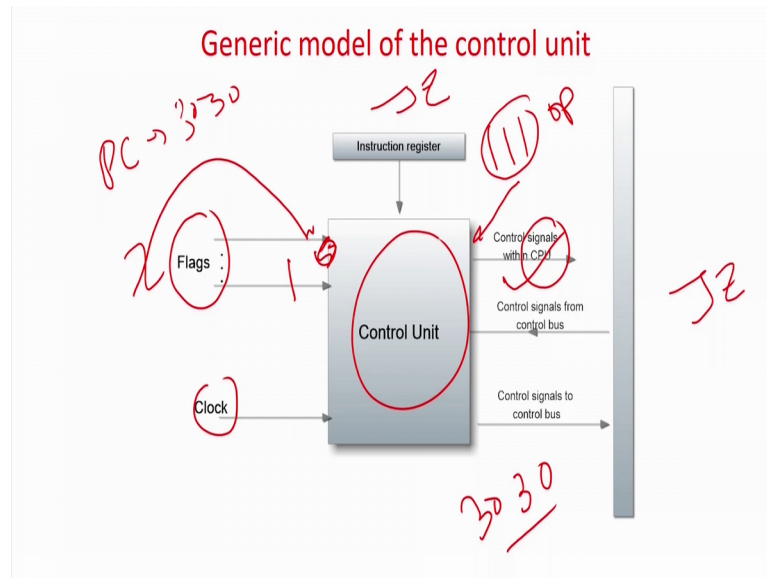
Unit Objectives

- **Knowledge:** Describe--Describe the different categories of input and output signals of the Control Unit of the processor.
- **Comprehension:** Indicate--Indicate the control signals to synchronize the speed of the memory module and the processor.
- **Synthesis:** Design--Design the timing sequence generator to carry out the proper micro-operation at appropriate time.

So, what are the basic objectives of this small unit, basically you will be able to first is the knowledge objective, you will be able to describe the different categories of input and output signals of the control unit. That if you take a black box of the control unit, then you have to take you will be able to tell which are the inputs and the outputs. Makes the comprehension objective, you will be able to indicate the control signals to synchronize the speed of the memory module and the processor that is how basically our clock occurs on basis of the clock, how basically are the control signals are generated or taken as input by the controller. So, you will able to indicate the use of clock and synchronization.

Finally, the synthesis objective you will be able to design timing sequence to carry out proper micro operations at an appropriate time. That is you will be able to design micro operations and in fact, you will be able to design at the granular level that for functionality or what signals will generate or what signals to expect as the input or output at appropriate exact timing of the clock. That means, you will be able to represent or design microinstructions in terms of timing diagrams so that is about the basic objectives of this do it.

(Refer Slide Time: 09:14)



Now, as I was saying we are coming to the basic idea of the unit, we are saying the general model of a controlling me. So, these are control unit which will be actually which will be your new central processing unit. It will actually do all the commands of the arithmetic logic unit, to the registers, to the cache memory etcetera. So, basically this is your control unit which will take care of all the points. So, the inputs as I told you is a clock, the clock is the synchronization, everything will be synchronized in terms of clock. As I told you there will be flags we have already know they the zero flag, non zero flag, carry flag etcetera. So, they will also be input to the control units.

Now what are the why is flag so important because I told you if you have the instruction say here call jump on z. So, in fact, say that is the instruction register will have the instruction called jump on z. Now, the instruction register for jump on there is an off code say for example, the opcode is 1 1 1 in this case. So, this input signal 111 to the control unit will specify that it is a jump control jump instruction, this is not an unconditional jump is a conditional jump.

So, this input from the instruction register that is the opcode part, because we assume that the jump instruction jump on zero to some memory location is already loaded in the instruction register, the opcode corresponding to that will generate the opcode which will actually control unit, it is an input to the control unit telling it what to do. So, this is an obvious input to the control unit that is the opcode for the current instruction. And in

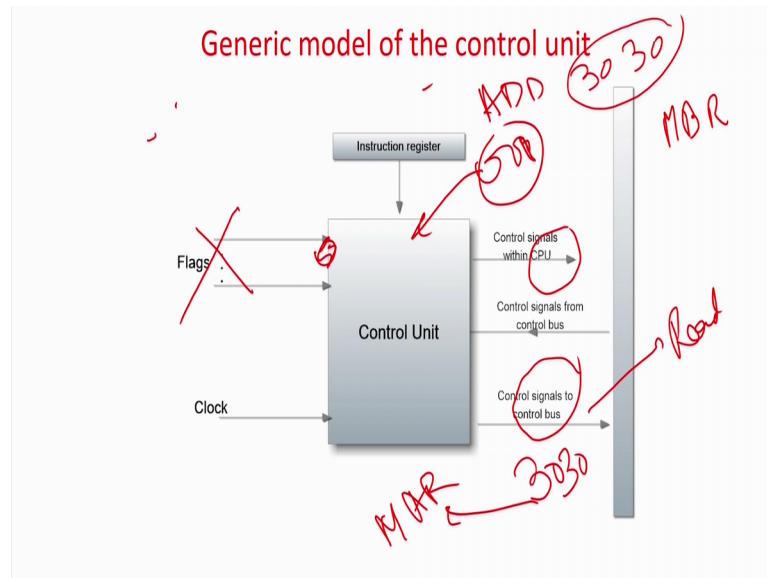
fact, that is the heart that heart of the inputs to the control unit because that tells you what to do now.

Now, the flags, now as I told you is a jump instruction. So, you want to know previously what was my input or what exactly was the operation done whether if I have subjected to number, whether the answer was a 0; if it is again then I will jump other I otherwise I will just go to the next instruction. So, the zero flag has to be set if the previous abstraction operation resulted in zero operation or any operation resulted in a zero operation. So, if the zero flag is said that signal will be sent zero over here, and the control unit will know that jump on zero is 1 1 1 that is the off code, and of course, zero flag was set that means in fact, it was 1. Sorry I mean it said because the answer was 0, so zero flag is set.

And accordingly whatever you want to do such signals will be generated by the CPU like for example, in this case you are saying there jump on 00 to 30 30. So, in fact, the program counter will be now loaded with say 30 30, the program counter will be loaded with the operation 30 30. And you already know that program counter is a register which is the internal to the CPU.

So, in that case the control signals will be generated within the CPU. Now, the program counter will be loaded with the value of 3030. So, the control signals generated by the control unit will generate the signals which is internal to the central processing unit, it will be something like it will configure the program counter in a load mode and we load it with 30 30. So, in that case simply your next instruction will start from 30 30 memory location.

(Refer Slide Time: 12:09)



If you take a another instruction let us assume that is a simple ADD operation ADD operation ADD say we call it 30 30 from the memory location you have to load 30 30. So, in that case what happen maybe the off code of ADD is 001. So, in this case, the 001 will be fed from the instruction register three control unit. In that case flag is not required because you are just doing for an ADD and you have to load from memory location 30 30. So, it will first generate some input signals to the CPU that the ALU has to be configured to ADD mode fine, then you have to read some data from 30 30.

Now, it will generate some data 30 30 which will be fed to memory address register because you have to read from memory location 30 30 which is be fed to the memory address register. They are now all internal CPU instructions, but actually if you also generate a signal called read because you have to read from the memory of course, then you have also put the memory address of 30 30 in the memory address register all those things are there like what memory address is an internal register CPU. So, it is an internal signal you have to load memory address register with 30 30.

But you have to make the signal to the memory saying there is a read signal. So, it will some generate some signal to the control bus, which will actually ask the memory saying that now we have to go to a read mode. Of course, after that, what will happen the memory will dump the value of 30 30 in the memory buffer register. Now, what happens now you have to wait for certain time till when the memory will load the value of 30 30

the whatever is minimal to 30 30 memory buffer register; till that time the control unit has to wait because immediately it cannot read the memory buffer it takes some time.

You are giving the 30 30 in memory address register, asking the memory in a read mode, and then you have to wait for some amount of time till which the memory gives the value whatever is present in 30 30 memory location to the memory buffer register. Once it is done the memory will give a signal saying that I am done now you can read the memory buffer register, so that will be a control signal which will be coming from the control bus and the control unit will read it. Once this is enable then you can read the value from the memory buffer register which will be again taken by the data bus, it will be going to the accumulator or and then it will be added.

So, basically in this case you can see there are some signals which are for the internal use of the CPU, and there are some thickness which is for example, memory or IO which are for the external use. For such external use we are using the control bus; and for our internal use, there is an internal bus of the CPU like register transfer, transferring considering the arithmetic logic unit etcetera, so that is basically the generic model of a control unit.

(Refer Slide Time: 14:37)

Generic model of the control unit

The groups of input signals of control unit are explained as follows:

Clock: The clock is used for synchronization of all the modules of the control unit. Also, the clock determines the minimum duration of time (or frequency) that can be considered as an atomic unit. Micro-operations that constitute an instruction are executed in one clock pulse. This is sometimes referred to as the processor cycle time, or the clock cycle time.

Instruction register: The instruction fetched from the memory is loaded in the instruction register. The opcode of the instruction determines the sequence of micro-operations that are required to be executed. For example, if the instruction is "LOAD R1, 32", then 32 (i.e., address of data which is to be loaded in register R1) is loaded into MAR and the control line of the memory is made READ. These sub-operations are achieved by executing the corresponding micro-operations

So, as I was telling now whatever I discussed they are actually written in the text, so that you can read it. So, they already told you clock is nothing but a synchronization module as I told you we have not talked anything about the clock over here. But whenever all the

instructions or all these signals are generated they are all either at the positive edge or the negative edge of the clock means everything has to be synchronized with some clock edge.

As I told you instruction register heart of the controlling unit, it tells exactly what to do say like for example, it is saying load R 1, 32 that means, the whatever is present in the location 32 has to be loaded in R 1. As I told you, so the op-code corresponding to load will actually direct the control unit to do that. So, as I told you so it is a heart of the control.

(Refer Slide Time: 15:22)

Generic model of the control unit

Flags: Flags store the status of the processor and the outcomes of earlier ALU operations. For example, if the instruction "SUB R1, 31", results in a zero then the zero flag will be set. This information can be used later to check if value in R1 is equal to the value in memory location 31; based on this condition, jump or loop instructions may be executed.

Control signals from control bus: The control unit sends signals to different modules of the computer (using control bus) for selecting their functionality. Similarly, the modules also communicate to the control unit using control signals through the control bus. For example, the controller may instruct the I/O module to start the printing task (by appropriate signals). However, the printing requires some time (i.e., clock cycles) to complete the operation. Once the operation is complete the I/O notifies the control unit using a signal through the control bus.

There is a flag and I told already explained that flag is very important because you have to remember what happened previously. Like for example, if there is a sub instruction and it said the sub R minus 31 may have been 0, so the zero flag has been said, but next after that you are using a conditional instruction that you should jump if R 1 is equal to 31. That means, whether the value of 31 is equal to the whatever value is in the register R 1 if that is set then you have to say I want to jump to some other location; otherwise I just want to do not jump I just go to the next location.

So, in this case you have to reuse, you have to read the value of flag. So, of course, the control signal or the control unit will require the value of flag signals to take a decision. So, they are also the inputs. And finally, as I told you there are control signals from the

control bus. They are basically which are taken externally compared to the CPU like from a memory.

As I told you memory is saying that I have done the right operation with memory buffer register now you can read, so that signal will be taken from the controller. For example you want to make the in a read mode or a write mode. So, the control signals will be given as output in the control bus. So, this is a very important control bus for that like for example, here is a printing operation etcetera.

(Refer Slide Time: 16:36)

Generic model of the control unit

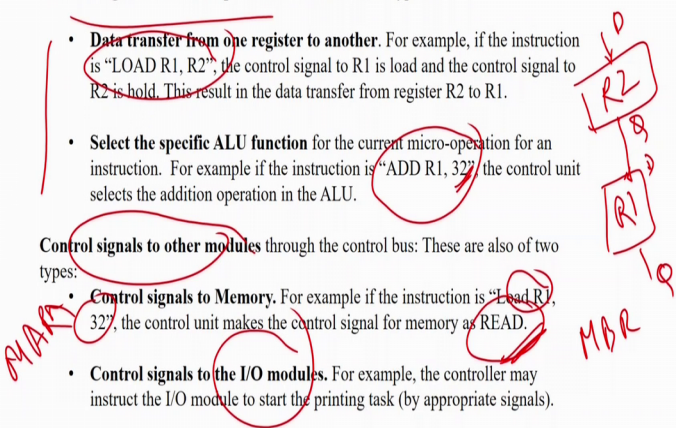
The groups of output signals of control unit are explained as follows:

Control signals within the processor: These are two types

- **Data transfer from one register to another.** For example, if the instruction is "LOAD R1, R2", the control signal to R1 is load and the control signal to R2 is hold. This result in the data transfer from register R2 to R1.
- **Select the specific ALU function** for the current micro-operation for an instruction. For example if the instruction is "ADD R1, 32", the control unit selects the addition operation in the ALU.

Control signals to other modules through the control bus: These are also of two types:

- **Control signals to Memory.** For example if the instruction is "Load R1, 32", the control unit makes the control signal for memory as READ.
- **Control signals to the I/O modules.** For example, the controller may instruct the I/O module to start the printing task (by appropriate signals).



Now, as we are already telling that these are all the inputs. What are the outputs? Already I have given example that basically the output signals are of two types. Therefore, signals can be within the processor that is CPU or they can be to the other modules like the memory or the I O. For example, we are already say for example, there is a instruction code load R 1, R 2. You can very easily understand that both the registers are in the CPU, so the control unit will just make R 1 into load mode and R 2 in the output mode.

For example because registers are nothing but this is one set of flip flop which is R 1 and then another set of flip flop which is R 2 because they are registers. The control unit will make it in such a fashion, so that this may be Q, I am I just assuming a D flip-flop, this is D, and this is D, and this is Q. So, it will make in such a fashion, so that the output of all the D registers of R 2 that is W will be made connected to the input D of the R 1. So, R 1 output will be loaded into R 2 at the clocking. So, simply such connections will be done.

In fact, you digital fundamentals, if you have forgot you forgotten you should go understand go and read about multifunctional register that means, a multifunctional register can do multiple things in can load from other register, it can load from input, it can freeze its import, it can go for increment, it can go for shifting. So, you have a multi function register basically this is a register if it is D, there is a multiplexer over here which actually do lot of interconnections and then you can set it in the functionality you require. This is the digital design fundamentals, if you have forgot you can just go and recollect it.

But in fact, for this load what happens you will connect the multiplexer in such a fashion, so that the output of R 2 that is q will be connected to d of R 1, so you will note R 1 to sorry R 2 to R 1. So, in that case in a simple load operation, so the control limit will generate these signals for the multiplexers of R 1 and R 2. So, as registers are internal to the CPU, there is nothing no signal to be given to the output in the control bus.

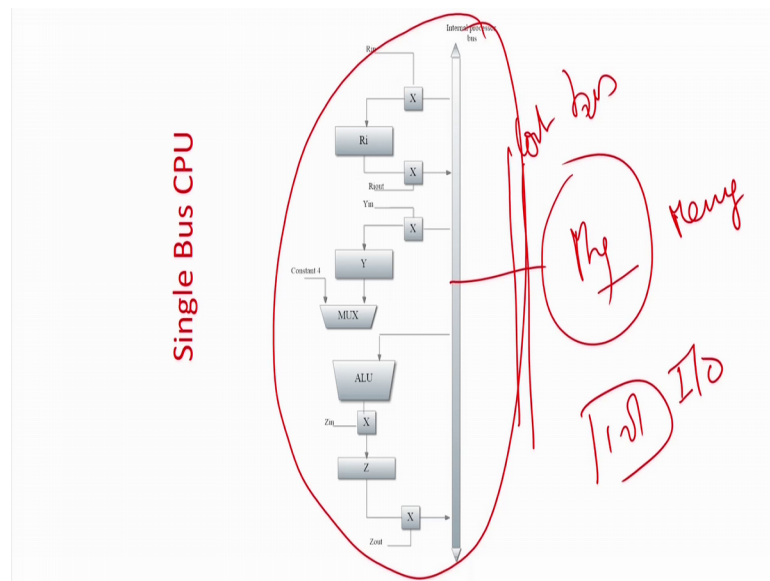
Like for example, if you say that at R 1, 32, in this case, for example, if is a specific instruction say as I assume that needs an immediate operation that 32 is basically nothing but you have to ADD R 1 with the absolute value 32 and ADD it. So, in this case also again if 32 is a memory operation then it will be an external signal, but if I assume 32 be an immediate operand then you have to ADD R 1 and 32. So, you need not generate any signal to these control bus only what will happen ADD will be ADD opcode immediate will configure the ALU to be in the read mode and then 32 will be loaded to the accumulator and you will ADD it.

So, basically for such type of operation like load data transfer within the register, some immediate mode of addressing, you need not generate any signal from the control unit to the control bus. Like that example for example, if you say that load R 1, 32 in this case if as I told you if 32 is not an immediate operand, it is a memory location. So, in this case, you have to make a control signal read that in the control box you have to say that I want to read then you have to put 32 in the memory address register, but memory address register is an internal register, so that will that load command or load signal is an internal load. But we read which is commanding the memory to be in read mode is an external signal, so that has to go to the control bus. But when memory address register is loaded with 32 that is an internal one, but this is redesign external come out to the control bus.

Then after some time what will happen the memory will load the value whatever is available in 32 memory buffer registered in that case again the control unit has to read via the control bus that when the memory has said that I am ready you can read from the memory buffer unit, so that will be an input. Similarly, if you have connection IO bus; obviously control signals has to be there from the IO bus for synchronization.

As I told you for example, if I am using this mouse then when I am making a mouse click then your control signal will be read from the control bus by the CPU, it will find out that the mouse click is there then we will it will give command for display. So, whenever the IO devices involved, memory devices involved, which is out of the CPU then the control bus comes into picture which is taking signals in and out from the control unit.

(Refer Slide Time: 20:34)

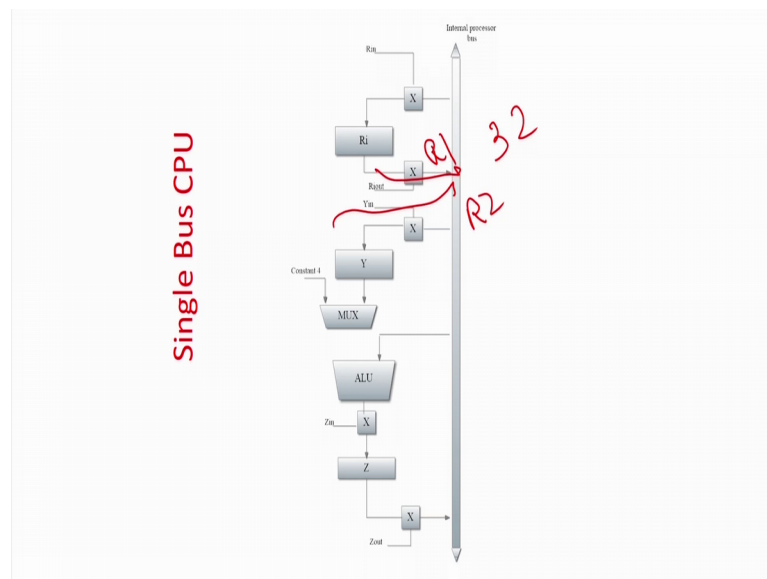


Now, very important thing that is we are going to look at what is the basic architecture of a single unit bus. So, let me zoom it. So, if you look at it, it is basically again let me escape. So, if you look at in a broad picture, so this is a single bus. So, in one part of the bus this side, you can have your, you can assume that there will be an internal bus, there will be some control buses etcetera, there will be your memory, there will be your IO. So, all these devices will be there and it is an internal and of course, there can be some control bus and several other buses which we are at present we are not talking about. So, internal control bus mean this is this part is basically nothing but your CPU.

And this part is your memory, these parts use your IO devices, this part is your memory devices and in fact this is a control bus to for the synchronization. Now, what we will do now initially we look into details on the internals if you bus because for the external devices when you will be covering entire modules on IO, they will be entire module on memory, so we will be handling that in details.

So, for example, for the time being let us just look at the details of the internal bus. So, there are some registers R_1 to R_{32} , R_{64} four how many registers you have. So, if you want to take from any input from the register from the internal bus, then what actually have to do you have to make R_{in} enable that is R_{in} equal to 1. If R_{in} is equal to 1, whatever data is available in the internal processor bus will be fed to R_1 . If it is $R_1 R_2$ we have just drawn it in a single R_i , but in fact in the 32 you have to replicate this part in 32 ways. But one very important thing is that so like for example, if I want to get the value of output of R_i to this processor bus, I have to make R_{out} equal to 1.

(Refer Slide Time: 22:17)



So, what happens see if for example, I have got the value 32 in the bus. Now, what I want to do I want to read this 32 into R_1 , and R_2 and R_3 . So, what you have to do R_{in} for R_1 has to be made 1, R_{in} for R_2 has to be made 1, and R_{in} for R_3 has to be made 1. So, in this case all the registers will be enabled in a read mode. So, 32 will go to R_1 , R_2 and R_3 this is fine.

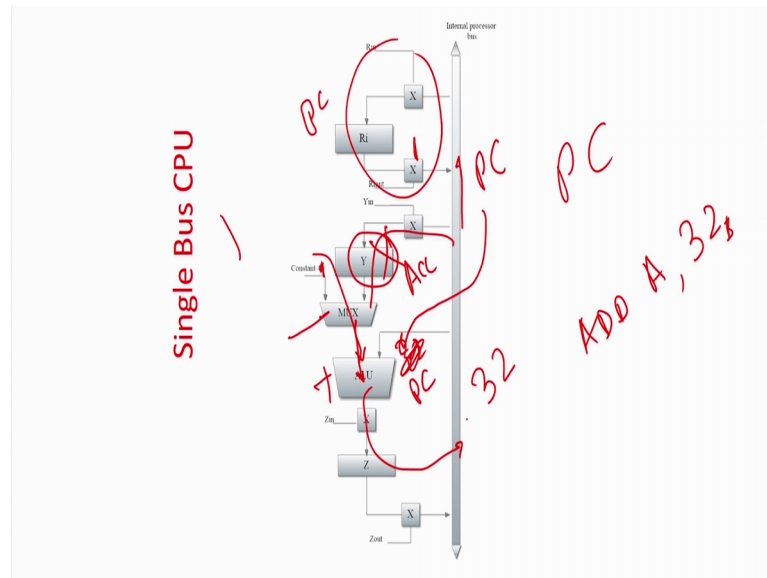
But be very, very careful that R I out cannot be more than one for any block which is giving output with resistor. For example, if I say that somehow I make R i out that is R 1 out equal to 1, and R 2 out equal to 1. What will happen the data from register R 1 will also go to the output and somehow in this case some R 1 will also go to the output R 2 will also go to the output, there will be a contention so that we cannot have.

So, while giving any output to the control unit sorry what output to the internal CPU bus, we have to be very, very careful that only one register or one ALU or one memory buffer register etcetera is loading into the internal bus. Multiple parties cannot output at a single go in the internal CPU bus that is very, very important. Now, who takes care the control unit, because the control unit will generate the signals R 1 in 1 1, R 2 out equal to 1 something like that.

Because whether resistor R 1 is going to read or whether this one or whether register R 1 is going to give the output will all depend on the control signals R in and R out, which will be ended by the control unit. So, contrary very judiciously takes care that, no two guys, no two registers or not ALU and register not memory buffer or resistor together our dumps at the same time in the internal bus that is out signal of any register to registers or one register or a memory buffer register can be one at a time. So, these are all registers organized in this manner; these are I forgot to draw this.

Now again if I zoom this next part of it. So, you can see that is basically second part is an ALU. So, either you can get the value from Y, so that is means whatever this is an input from the control bus sorry it is from the internal bus where you can get the data values. So, either you can get the data value Y in, so it is a multiplexer in the ALU. So, you can get one as I told you ALU basically does all the mathematical and logical operation. So, there are two operands for this. So, one operand basically comes from this CPU bus, and the operand basically Y is the register, so sometimes you may have to hold the value temporary.

(Refer Slide Time: 24:51)



Like, for example, if I say that I want to make ADD, I want to go for say ADD accumulator and 32 immediate. So, in this case, what is going to happen very interesting simple, so 32 value that is what is seeing that ADD 32 if you see, so it can be fed over here by this line direct connection because of the controller will set in such a fashion, so the value of 32 will be loaded over here. The control signals also will make the ALU to be ADD mode for example, when I am going for going to execute this command ADD accumulator 32 immediate.

So, it will be converted to ADD mode by the signal for the signals of the control unit 32 will be loaded in the bus, so you will have the value of 32 over here. But previous value of activate one has to be added to this. So, in fact, actually the accumulator will be loaded in the previous cycle value of accumulator will be loaded to Y or in some sense you can also think that in my case Y is an accumulator. So, basically sometimes this is what is the idea.

So, in the first go, the value of accumulator will be or if the accumulator has if you have to note some particular value the accumulator. In first case, you will be loading the value of the accumulator from memory or some block to Y, I am assuming A as an accumulator. And in the second unit clock, you will actually ADD this accumulator with 32, but now you can see a marks over here, because sometimes we require to ADD some constants also like program counter. Program counter will be program counter plus 1. So,

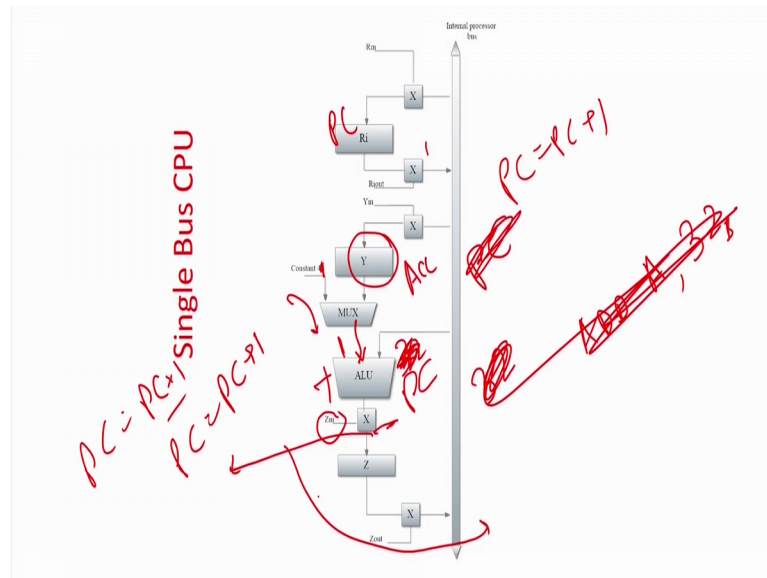
in this case, I have kept the value 4, because I am assuming in this figure that assuming that the number of instruction length is 4, basically I can also say that the sorry I can make this constant as one.

So, why I am making this constant as one; basically sometimes when I have to increment the PC. So, in this case what happens I will make the control unit will make marks set as this part that is so for example, if this was about the ADD, so if I go for a. So, for example, if I say that I want to go for a PC increment mode. So, in the PC increment mode, what is going to happen, you have to go for an increment of the PC. So, in this case I will this is the flow from the accumulator, so in this case I will not go by this path. So, what I will try to do is that is the constant. So, the multiplexer will be set in such a fashion by the control unit that this signal will enable this to go to the ALU.

And in fact, so this one will be added. Now, the value of PC, so you are going to add it to 1, and then the PC will be say for example, one because this is the chunk of registers. So, for the time being assume it to be the PC. So, in this case, what happened you will enable in such a fashion that this is enable will be 1. So, in this part, now the value of PC will be there. So, the PC will be connected over here. So, now, it is not 32, it is the value of program counter, and program counter will be program counter plus 1. So, now, after that you can dump the value output here, and you can again feed it back. So, again let me clean the whole picture.

So, now, again what let me do it in steps? So, what is the first step, the first step is basically for ADD, actually I go by this configuration that the configuration is this, but now for PC to be incremented, let us assume this is to be PC.

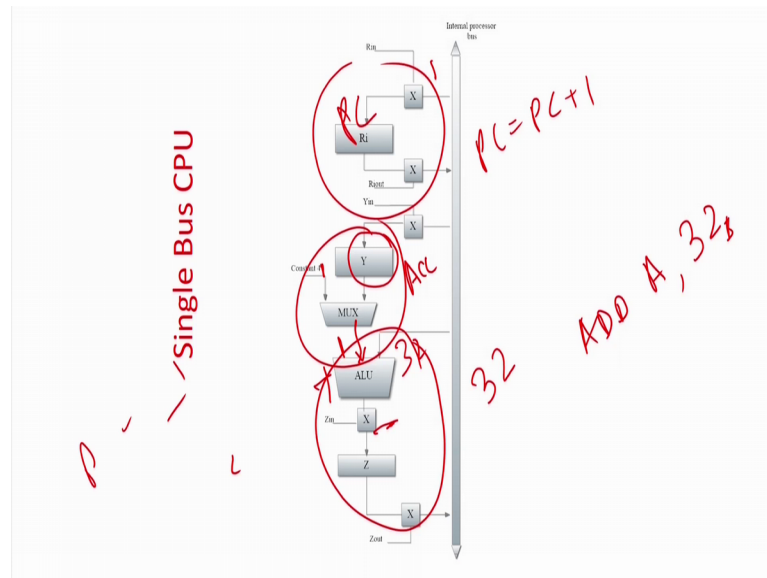
(Refer Slide Time: 27:58)



So, first we are going to go for PC equal to PC plus 1. So, what you are going to do, for adding basically we are for adding two numbers or some other operation, we are going to take this path, but actually for the PC because it is to be incremented by a constant. So, what we are going we are going to take this path. So, the mux is set in such a fashion. So, this one this one or constant one in case if the size of the instruction is one, it may be 4, 8 or 2, but in size of the instruction that value will be fed over here, accumulator in plus mode. So, in this case, we are adding a value of 1, but now it should not be the value 32.

So, it is not the case you should have the value of the PC fed over here. So, what I will do assuming that this is the PC I will make this PC out equal to one. So, the value of the PC will be in this bus. So, now, the value of PC is fed over here. And addition is that and then what happens this is the Z in plus will be made one or in other words this is the path to control the output of the ALU to the bus. So, now, it is added is PC equal to PC plus 1 is added, and it is now waiting over here. Now, we are it is PC equal to PC plus 1. Now, after some time what you will do you will actually now it is having the value of PC equal to PC plus 1, because now the output is fed over here.

(Refer Slide Time: 29:17)



Now, after that what you will do now you will make this as 1, because now PC equal to PC plus 1. So, now, it will be loading the value of PC and PC will be incremented. So, in this fashion basically we require a multiplexer for that. So, these are basically if you want to understand what happens basically in this case, so this is basically the architecture single bus. So, we have some registers which have input and output control that is register can be fed in, register can be fed out from the control bus.

Similarly, this is an ALU, ALU is also the facility you can load, and also you can write out the value of the ALU to the control bus. And we are having multiplexing arrangement. So, in one (Refer Time: 29:51) of the multiplexing arrangement, you can take some operand from the control bus; and the other operand is a constant like increment, increment by one, but some other constant value. So, this is in a nutshell a very broad idea of a single bus CPU.

So, now we will tell you whatever I have discussed is given in the text, so I am saying that for each register including the program counter, memory buffer register etcetera there are two signals R in and R out. Basically if R in is 1, the register is going to read from the bus; if R out equal to one then is going to write into the bus, but you have to be very very careful that we should not basically enable two R outs at a time then they will be contention. But there can be multiple R ins.

(Refer Slide Time: 30:35)

Working of Single Bus organization

- For each register (shown as R_i) two control signals (R_{in} and R_{out}) are used to place the contents of the register on the bus or to load the data on the bus to the register.
- To elaborate, the input of register R_i is connected to the bus via control signal R_{in} and output of R_i is connected to bus via R_{out} . When R_{in} is 1 then data on the bus are loaded into R_i and when R_{out} is set to 1 then contents of the register are placed on the bus. It is assured that at a given time no two registers can have their control signal R_{out} as 1; this ensures that there are no conflicts.
- Example: In case of instruction MOV R2, R1, first we set R_{out} to 1 so that data of R1 is placed on the bus and then by setting R_{in} to 1 (and R_{out} to 0) data from the bus is loaded to R2.

*R2in = 1
R1out = 1*

For example, if you are saying that move R 1 and R 2 then what will happen basically is R 2, R 1 is feeding to R 2. So, what is the control unit is going to generate the control unit is going to generate that R out equal to 1, because R out is going to give the value to the bus, and R in R 2 register is reading. So, R 2 in should be equal to 1. So, R 2 in is equal to 1, and R 1 out equal to 1. So, R out R 1 is giving the value the bus, and R is reading. So, of course, other thing has to be made 0, because R 1 out is equal to in this first case. So, R 1 out is equal to 1. So, R 1 R 1 in has to be made 0 that is the case. For example, R 2 is reading, so basically R 2 in is 1 and R 2 out has to be basically set to 0 that is what is very simple logic.

(Refer Slide Time: 31:22)

Working of Single Bus organization

"MOV R1,32"

- 1) IR_{out}, MAR_{in} , Read
- 2) MDR_{in} , WMFC(control signal that causes processor to wait for MFC signal)
- 3) $MDR_{out}, R1_{in}$

- First, the address of the memory which is to be read (32 in this example) is loaded into the MAR. This is achieved by making signals IR_{out} as 1 and MAR_{in} as 1. As the contents of MAR are always on the address bus, when a new address is loaded into MAR it will be on the memory address bus at the start of next cycle.
- The memory control signal is made READ, which results in the data in the corresponding memory location to be loaded in the MDR. So, in the second cycle MDR_{in} is set. However, the system needs to wait for the delay involved in memory reading (indicated by MFC signal). After the delay (i.e., WMFC becomes 1), the data from memory is loaded into MDR.
- In next clock cycle MDR_{out} is made 1 and $R1_{in}$ is made 0 which transfers data to register R1 from MDR.

Now, we will take some more important basic ideas other instructions to make it more clear. So, we are taking a instruction called move R 1, 32. So, basically what is the steps, what happens. So, let us assume that this is your control box then what happen then this is a register R 1. So, as I told you, you can read from the register and also the register can be making an output this is your control bus. So, either you can read or you can write. So, in this case, the 32 has to move, in fact, here 32 is a memory location given the assumption. So, now, what happens 32 is not an immediate operand, it is a memory location. So, you have to read the value whatever is available in memory location 32 to register R 1.

So, of course, register R 1 in should be equal to 1, these we are going to read it and then basically other steps will follow. So, what are these steps, first step is that the instruction register should give the command or the microinstructions following that. So, what are the microinstructions for that? For example, the value of 32 has to be first loaded into the memory address register, why, because the memory address register is going to give the value to the memory, and it will say that I want to read the value from 32, and it will dump the value in memory data register or memory buffer register.

And then finally, what is the first step, you have to put the value of 32 in memory register, memory address register make the signal read that is the first signal. Then what happens then the memory will see the memory address register is 30, whatever value is

memory location 30 will be dump to memory data register or memory buffer register and you have to wait for some time as I told you this signal is called WMFC.

Control signal that causes the processor to wait for the MFC signal. And you will have to wait for some time, when the memory says that I have right written the value what was the memory location 32 to the memory data register or the memory buffer register it will become 1 then you know the memory has given the data now everything is stable. Now, the memory buffer register will dump the value to R 1.

So, these are the three microinstructions. The first microinstruction we lead to basically loading the value of 32 in the memory address register, second thing it will read the value in the memory data register, third microinstruction will load the value of memory location in the R 1. So, what are the steps, let us write down the signals and the registers.

(Refer Slide Time: 33:46)

Working of Single Bus organization

• "MOV R1,32"

- 1) IR_{out}, MAR_{in} , Read
- 2) MDR_{in} , WMFC(control signal that causes processor to wait for MFC signal)
- 3) $MDR_{out}, R1_{in}$.

- First, the address of the memory which is to be read (32 in this example) is loaded into the MAR. This is achieved by making signals IR_{out} as 1 and MAR_{in} as 1. As the contents of MAR are always on the address bus, when a new address is loaded into MAR it will be on the memory address bus at the start of next cycle.
- The memory control signal is made READ, which results in the data in the corresponding memory location to be loaded in the MDR. So, in the second cycle MDR_{in} is set. However, the system needs to wait for the delay involved in memory reading (indicated by MFC signal). After the delay (i.e., WMFC becomes 1), the data from memory is loaded into MDR.
- In next clock cycle MDR_{out} is made 1 and $R1_{in}$ is made 0 which transfers data to register R1 from MDR.

So, basically let us assume this is a control bus. So, first is the instruction register. As I told you, it is very, very important; the instruction register is the heart. So, what it will load, it will load the value of 32 in the memory address register. So, as I told you these are memory address register - MAR.

So, in this case, memory address register in will be one and instruction register out will be equal to 1, because instruction register will dump the value of 32 to the memory address register. So, in fact these are the two internal signals that is R out instructional

out equal to 1, so instruction set will dump the value control bus; and memory address register will in is equal to 1, so it will read the value of 32. Now, the 32 has gone to the memory address register. These are the two internal control signals.

(Refer Slide Time: 34:31)

Working of Single Bus organization

- "MOV R1,32"
 - 1) IR_{out}, MAR_{in} , Read
 - 2) MDR_{in} , WMFC(control signal that causes processor to wait for MFC signal)
 - 3) $MDR_{out}, R1_{in}$.
- First, the address of the memory which is to be read (32 in this example) is loaded into the MAR. This is achieved by making signals IR_{out} as 1 and MAR_{in} as 1. As the contents of MAR are always on the address bus, when a new address is loaded into MAR it will be on the memory address bus at the start of next cycle.
- The memory control signal is made READ, which results in the data in the corresponding memory location to be loaded in the MDR. So, in the second cycle MDR_{in} is set. However, the system needs to wait for the delay involved in memory reading (indicated by MFC signal). After the delay (i.e., WMFC becomes 1), the data from memory is loaded into MDR.
- In next clock cycle MDR_{out} is made 1 and $R1_{in}$ is made 0 which transfers data to register R1 from MDR.

Now, next what next we have let say control bus, this is your control bus which is an external control bus. So, the control unit will generate the value of read to the memory, because this control bus is connected to the memory, this is the memory which is an external, but the first two signals are internal, this is an external signal. Now, let us go to the second signal. So, in the second signal what happens it is saying that now after some amount of time, the data will come to the memory data register.

(Refer Slide Time: 35:05)

Working of Single Bus organization

- "MOV R1,32"
- 1) IR_{out}, MAR_{in} Read
- 2) $MDR_{in}, WMFC$ (control signal that causes processor to wait for MFC signal)
- 3) $MDR_{out}, R1_{in}$

- First, the address of the memory which is to be read (32 in this example) is loaded into the MAR. This is achieved by making signals IR_{out} as 1 and MAR_{in} as 1. As the contents of MAR are always on the address bus, when a new address is loaded into MAR it will be on the memory address bus at the start of next cycle.
- The memory control signal is made READ, which results in the data in the corresponding memory location to be loaded in the MDR. So, in the second cycle MDR_{in} is set. However, the system needs to wait for the delay involved in memory reading (indicated by MFC signal). After the delay (i.e., WMFC becomes 1), the data from memory is loaded into MDR.
- In next clock cycle MDR_{out} is made 1 and $R1_{in}$ is made 0 which transfers data to register R1 from MDR.


We know that you know already the memory address has been set so some data will be coming to the memory data register is an internal part or a memory buffer register whatever. So, as the data from memory location 32 is going to come over here this in signal I have to make it one, these are internal signal. But then if you wait for some external control bus, you have to wait for some of the signal which is coming from the memory that is called wait for MFC this signal is MFC we are waiting for MFC. Whenever MFC is going to be one at that point of time basically already MDR is equal to one input. So, whenever MFC is equal to 1 then you know that there is a valid data over here, I have already made it 1.

So, whatever is the memory buffer register is coming over here, but I cannot immediately read the value in the memory data register, from the memory data register to R 1. I have to wait for some amount of time. Whenever MFC is 1, which is an external signal then again the third the micro cycle instruction control signals come up. Now what, so now basically MDR is already one signal has come. So, now, what is going to happen already memory data MDR has already read.

(Refer Slide Time: 36:07)

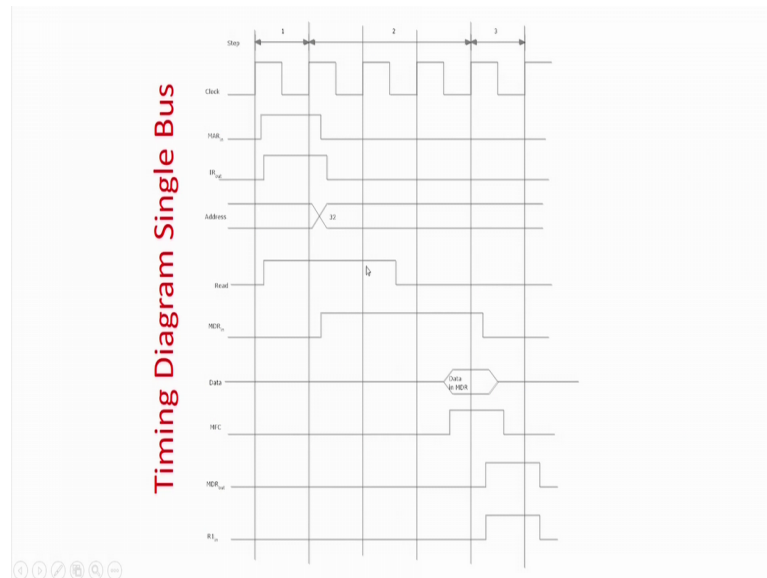
Working of Single Bus organization

- "MOV R1,32"
 - 1) IR_{out}, MAR_{in} , Read
 - 2) $MDR_{in}, WMFC$ (control signal that causes processor to wait for MFC signal)
 - 3) $MDR_{out}, R1_{in}$
- First, the address of the memory which is to be read (32 in this example) is loaded into the MAR. This is achieved by making signals IR_{out} as 1 and MAR_{in} as 1. As the contents of MAR are always on the address bus, when a new address is loaded into MAR it will be on the memory address bus at the start of next cycle.
- The memory control signal is made READ, which results in the data in the corresponding memory location to be loaded in the MDR. So, in the second cycle MDR_{in} is set. However, the system needs to wait for the delay involved in memory reading (indicated by MFC signal). After the delay (i.e., $WMFC$ becomes 1), the data from memory is loaded into MDR.
- In next clock cycle MDR_{out} is made 1 and $R1_{in}$ is made 0 which transfers data to register R1 from MDR.



So, now this if this is your MDR it has already read. So, now, initially it was in was one because it was reading from the memory, now it has to make it out. So, my memory data out signal is be 1, so out signal is equal to 1, because already we know that the memory has dumped the value in the memory data register and it will read to R 1. So, this in signal has to be 1. So, the memory data will be given over here. So, again in this case, these are the two control signals generated, memory data register out equal to 1 and R in equal to 1. And both our registers basically they are all internal signals. So, this is how in a single bus memory operation that is reading from 32 into R 1 works. So, basically what I have told you written in the text over here which you can go through.

(Refer Slide Time: 36:51)



Now, we are going to look at in timing, as I told you that will be all discussing it in terms of timing sequence. So, let us look at the timing sequence. So, this is your clock and we are doing everything in the positive edge or in the positive edge of the clock. So, this is the positive edge, this is the positive edge and so forth. So, first as I told you first one read R 1, 32. So, what you have to do, first you have to read the value of 32 from the instruction register to the memory address register. So, what I am doing just after the first clock edge I am making m b R equal to one and R out equal to 1.

So, what is going to happen R out that is instruction register which is containing now the memory location 32 will be done to memory address register. So, I have made both the control signals equal to one. So, in the next clock edge, what is going to happen that is the synchronization. At this clock is what is going to happen the value of 32 from instruction register will go to the memory register that is at this rising clock edge, the value of 32 will go from instruction register to memory data register, it will happen at this clock edge.

So, now again let me erase this, and let us study the future in future what happens basically that is the first clock edge. Secondly, so already we have seen that in this clock edge the value will be dumped. So, we have actually given the value of address is 32 is as gone to the memory data register. So, in this clock edge has got this address of 32 is now loaded into this memory buffer register sorry memory address register. Now, already

we know that you have to give a read signal. So, anyway I have given the read signal over here, you could have also given the read signal over, I have given the read signal initially. So, now, at this clock edge, these 32 is loaded in the memory address register as well as the read signal is the one.

So, now the memory is configured that in this case in the first in this clock edge, the instruction register has given the value of 32 to the memory address register which is this case, read signal was already one. So, now, the memory knows very well what it has to do and. In fact, already I have, so this is happening in this clock edge. So, by this clock edge the memory data should come, because in this case I have sent the value 32 the address register in this clock edge that works.

And again in this clock edge, the value of 32 will be taken in by the memory and the read signal is also one at this place. So, in fact, now it will start dumping the value of the memory data register the memory will start dumping the value whatever is the 32 that will actually happen in the next clock in this clock, it should start happening. Because in this clock edge 32 is loaded in this clock edge because everything happens in positive clock edge.

So, if you find in the first clock edge nothing happens in the second clock edge basically in this edge the instruction register value as gone MAR address register, and in this clock edge 32 is loaded in that case and read instruction is also 1. So, in this memory, in this clock edge the memory is fully configured to deliver the value of memory location 32 the memory data register, already read signal was there. And if you see MDR in was already set, so now, in this clock pulse if you compare the value will be after this clock pulse that is this clock pulse the value will be fed to the memory data register because the signal is already one.

And only after this the MFC will signal will be set as high. So, what is this MFC signal, only after this clock pulse? The MFC signal will be high means it will say that now I have done the value of memory location 32 in the memory data register in a very peaceful manner everything is stabilized. Now, you can read, so that is the signal, so that will happen in this positive clock edge. And after that has happened that MFC has been given as out, so what you will do this says that I have done or I have given everything which was the memory location 32 to the memory data register.

Now, what you will do now we have to read of memory data register to the register R 1 that is you have to do this part that memory data register value will has to be dump to register R 1. So, only after that MFC signal has become 1, you can make the memory data register signal as out. Because before that if you see the memories data signal was a 1 over here in one that is memory data register in was a 1 that means it was reading from the memory.

This MFC signal is saying that the reading is over. So, now, you make memory data register out signal equal to 1 that means now it will dump the value whatever was in the memory data register which is taken from the memory to the bus. And then R in 1 equal to 1 that means, whatever was in the memory data register will dump to the register R 1 and this instruction of move R 1, 32 will be over.

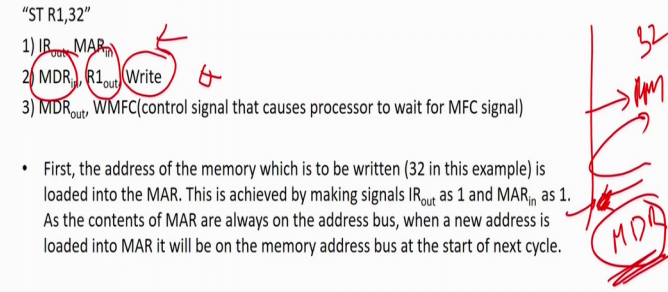
So, if you can recollect it look at the figure in a slightly you can think it for some time then everything will be very clear to you. Then this was a move instruction. Now, let us say that is a store instruction move means what we have done, we have taken the value of memory location 32 whatever was there, we have moved to R 1.

(Refer Slide Time: 42:05)

Working of Single Bus organization

"ST R1,32"

- 1) IR_{out} , MAR_{in}
- 2) MDR_{in} , $R1_{out}$, Write
- 3) MDR_{out} , $WMFC$ (control signal that causes processor to wait for MFC signal)



- First, the address of the memory which is to be written (32 in this example) is loaded into the MAR. This is achieved by making signals IR_{out} as 1 and MAR_{in} as 1. As the contents of MAR are always on the address bus, when a new address is loaded into MAR it will be on the memory address bus at the start of next cycle.
- Next, the data in R1 is transferred to MDR which is achieved by making MDR_{in} and $R1_{out}$ to be 1. The memory control signal is made WRITE, which results in the data present in MDR to be written to the corresponding memory location.
- So, in the third cycle MDR_{out} is set. However, the system needs to wait for the delay involved in memory writing (indicated by MFC signal) for the instruction to be complete.

Now, let us very quickly see that if this is the reverse one that is if there is something in memory register R 1 sorry if there is some value in R 1, we want to dump it to memory looking at 32; one was the read operation, next was the right operation very simple. Of course, first the value of R 1 has to be written to 32. So, the register value our out

instruction register has to be made 1, because the default idea is that whatever instruction is there will be first in the instruction register. So, therefore, any instruction in a general thumb rule, what is there you have to first make the instruction register out that means, the value of the instruction register have to go to the memory address register.

Because whenever there is a non immediate mode of operation, so what happens like in this case the memory location is 32; that means, you have to do something with memory location 32, but when is that values 32 will be there in the very initial case it will be in the instruction register. So, generally the first micro operation always says that the instruction register out that means, you have to get the value of the memory location from the instruction register. And it has to go into the memory address register that is a very (Refer Time: 43:05) standard for most of the cases. So, the value of 32 from the instruction register will be dump to the memory instruction register.

Even if you have looked in this solution also, the same read or write from the memory the first microinstruction will be more or less similar. So, the first microinstruction register instruction register out will dump the value to the memory register a memory address register; that means, instruction register 32 value will be now dump to the memory address register. So, now, the memory address knows that value of 32 is there. Now, we have to do something with 32.

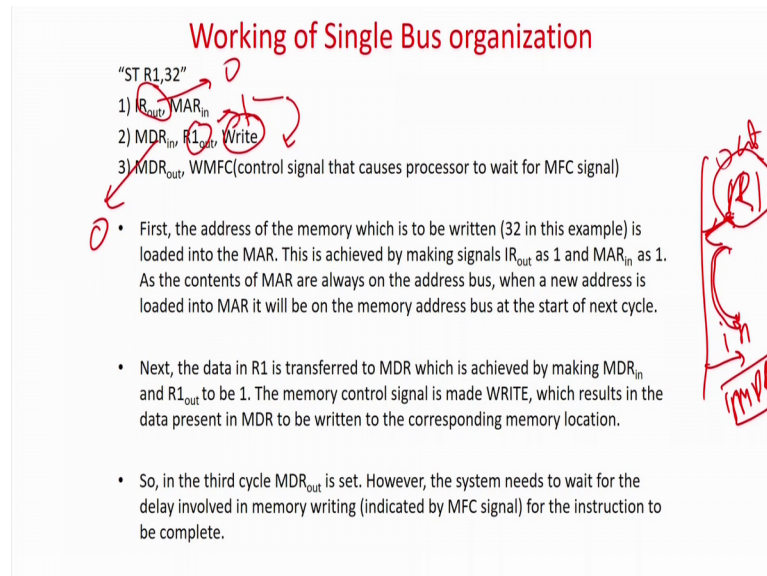
Now, in this case, what happens, you have to write; in the previous, case what happened it was a read now it is basically a write; that means, whatever is in the register R 1 has to go to the memory. So, in now in this case what happens this is the memory, this is the 32 memory location has to be read and in fact, there is a memory data register.

So, in the read mode from the memory, the memory register is to be read, but is the right operation, so you have to dump the value of here. So, in this case we will just the reverse compared to the previous analysis. So, MDR will be actually equal to in. So, now, the memory read it has to read. So, the memory data register will be not read mode because it will read something from the register R 1 and that has to be dumped to the memory.

So, in the previous case memory data register in was a 1 that means, what basically what happens again as I was telling you that the first microinstruction in the second microinstruction more or look less look similar in case of a read write mode. Because in any case, memory data register is to be read either it is from the memory or it is from the

CPU register. If you are going for a write-read operation, then the memory register will read from the memory if you are going to go for a write operation to the memory is going to read from the register. But anyway memory data register has to read some value which is to be transferred to the memory or it has to be transferred to the register. So, in this case, memory register, data register is equal to 1 and R out equal to 1.

(Refer Slide Time: 45:12)



So, in case what happens in the second case, so if you look at it, so this is your register memory data register, which is in mode, and your register R 1 is in the outward. So, the register will dump the value which will go to the memory data register, but this is a memory write operation. If it is a memory read operation, then actually in fact this it will not be a R 1 in fact it will be a memory, which will damp it to the value to the memory buffer register, which will go to the memory there are the previous case.

But it now second case, what happens, memory data register is in mode and who is going to write it the reading instruction the register R 1. So, R 1 will be out will be one. So, in that case, it will be going in this case. And very important we have to know one point over here, which I want to emphasize basically first microinstruction R out equal to 1 that is you are dumping the value of 32 to memory address register. Before going to the second microinstruction, R out IR out has to be made to 0. Otherwise what is going to happen others there will be a conflict what the instruction register as well as R 1 is write together to the CPU bus that should not happen. Before going to the value here this has

made be made 0, but now in this case R out is 1, this one is in 0. So, basically now R out is going to write to the memory data register. And now we are giving a signal called write because the memory has to be in right mode.

(Refer Slide Time: 46:40)

Working of Single Bus organization

"ST R1, 32"

- 1) IR_{out}, MAR_{in}
- 2) $MDR_{in}, R1_{out}, Write$
- 3) $MDR_{out}, WMFC$ (control signal that causes processor to wait for MFC signal)

- First, the address of the memory which is to be written (32 in this example) is loaded into the MAR. This is achieved by making signals IR_{out} as 1 and MAR_{in} as 1. As the contents of MAR are always on the address bus, when a new address is loaded into MAR it will be on the memory address bus at the start of next cycle.
- Next, the data in R1 is transferred to MDR which is achieved by making MDR_{in} and $R1_{out}$ to be 1. The memory control signal is made WRITE, which results in the data present in MDR to be written to the corresponding memory location.
- So, in the third cycle MDR_{out} is set. However, the system needs to wait for the delay involved in memory writing (indicated by MFC signal) for the instruction to be complete.

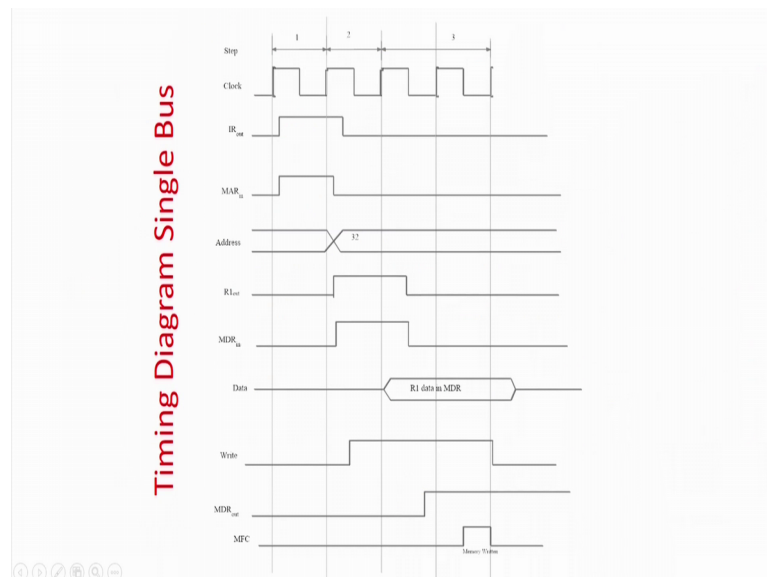
Now, what happens so now your memory data register already has the value, which is in the register. So, this phase is over. So, last microinstruction basically let us see what it will do this is your instruction this is your bus now you are already memory there instead has the value which is from R 1 and you also you have given the write command. The write command is basically an external signal; all others are internal signals which is generated by the control unit. Now, memory data register out is equal to one. So, now, it will start dumping the value.

Now, of course, the memory is already in the read mode and memory data register is writing to this will be a control bus or sorry in fact it will be the data bus which is an external bus which will be connecting the CPU to the memory and already write signal is there. But immediately you cannot we draw this signal memory data out has been made one. So, the memory register is writing to the control bus, but immediately you cannot make the memory get out equal to zero you have to wait for some amount of time. To an external bus external control bus, the memory will tell that MFC that is going to come by a external control bus to the CPU telling that my read is over or sorry yeah my read is

over; that means, I have already read the value of register R 1 in a memory location 32 now you can free your memory data register.

After that case only your memory register output will be made 0, in fact, initially it was one. So, memory data register was dump in which was going to the memory after some amount of time when this WFMC signal will be coming from the external control bus to the CPU sorry to control unit of the CPU it will relinquish the signal and MDR out will now become equal to basically your 0. No, now it is free and whole value of R 1 has been dump to memory location 32.

(Refer Slide Time: 48:10)



So, again I will quickly look at the control sequence because it is more or less similar. So, in fact, this is the positive edge of the clock. First instruction register has to be R out will be one same procedure; in the next clock edge as memory address register is already one. So, the value of 32 will be dumped in the address. From here, it is slightly changing because it is a memory write operation now R out in equal to 1. So, now from this at this clock edge basically as I have made this memory data register in one at this point at this clock edge what is going to happen is that the value of the register R 1 will be out to the control bus. And already memory data register in have been made 1 that means, at this clock edge at this positive clock edge what is going to happen, the value of R 1 will be going to the memory data register in.

Now, memory data register in will be fed with the value of register one, but where the memory data register will write to the memory which is in the memory address register which is the value of 32. So, R 1 out will make the value of register R 1 being dumped to the bus which will go to memory data resistor and memory data register will write the value that is what we are saying. There is data whatever is in the R 1 data has now moved to memory data register which is happening in this clock edge. And this memory register will be writing into the memory. So, the memory write signal has been made one.

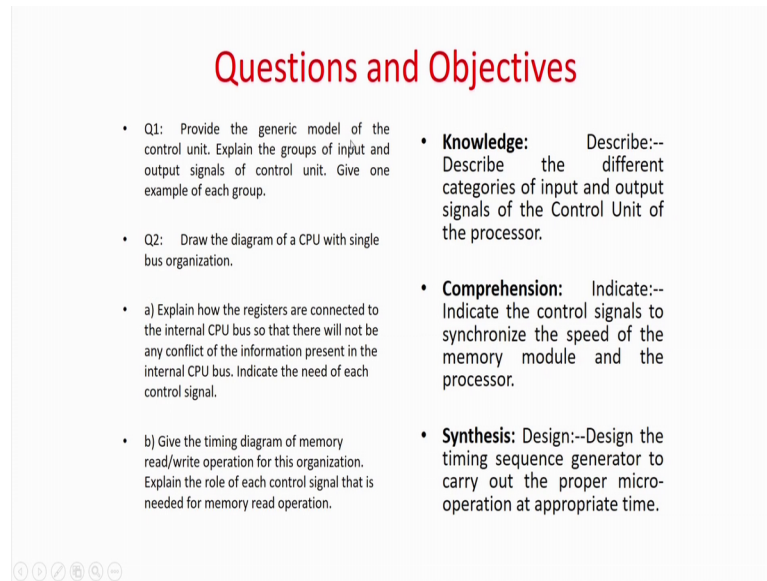
Now, at this clock edge, basically as I told you at this clock edge, the data of R 1 has been moved to the memory in register write signal has been enabled. So, now, I am making the MDR equal to 1 that means, now the memory data register is going to write to the memory. So, I am making the memory register from input mode to output mode which I am making it 1. So, in fact, if this clock is if you compare this is the rising clock edge after this signal after the MDR signal equal to out, this is the clock edge. At this clock edge, your memory will read the value from the memory to the register, because memory data register out signal equal to one and we are in a write mode. So, at this clock edge, this is the positive clock edge your memory is going to read the data from the memory data register.

But again immediately we cannot make MDR out equal to 0, we have to wait for some amount of time when the MFC signal will be seen memory has been written. So, after this clock edge basically you can again I have not shown this, but again you can in this point you can make this signal down over here. This signal can be made down at this point, because there is no point of keeping me that you over here. So, simply the memory has been.

So, that brings us to the end of this unit we have taken a single bus architecture and we have taken some very simple instructions which are macro instructions. We have broken down into the microinstructions, and we have seen what are the basic microinstructions or what are the control signals generated, and how data transfers operates happens between one register to another register, one memory to another memory and via register. And also we have seen that if you read to a memory, if you write to the memory what are basically controls signal involve.

So, in fact, in a nutshell, today we have got some idea using a very concrete digital fundamentals how basically a control unit works in terms of signals. So, before we close down as we are all discuss about some of the questions and try to understand how the objectives are satisfied. So, let us take some few examples like provide a generic model of the control unit.

(Refer Slide Time: 51:22)



Questions and Objectives

- Q1: Provide the generic model of the control unit. Explain the groups of input and output signals of control unit. Give one example of each group.
- Q2: Draw the diagram of a CPU with single bus organization.
- a) Explain how the registers are connected to the internal CPU bus so that there will not be any conflict of the information present in the internal CPU bus. Indicate the need of each control signal.
- b) Give the timing diagram of memory read/write operation for this organization. Explain the role of each control signal that is needed for memory read operation.
- **Knowledge:** Describe:-- Describe the different categories of input and output signals of the Control Unit of the processor.
- **Comprehension:** Indicate:-- Indicate the control signals to synchronize the speed of the memory module and the processor.
- **Synthesis:** Design:--Design the timing sequence generator to carry out the proper micro-operation at appropriate time.

Explain the groups of input output of these signals and give some examples, of course, which will satisfy your knowledge objective like different categories of input and output signals as a comprehension. You should be able to design it as of sorry the comprehensive objective will be able to indicate the control signals to synchronize the speed etcetera. So, if you are going if you are able to answer the first question that what are the different signals these two objects are satisfied. If I ask you to draw a CPU with a single bus organization, of course, this will again satisfy the next two objectives.

Then in this question to we have some parts we are saying that how registers are connected to the CPU and how basically signals move over there, and then we say that how they are synchronized in time. So, if you are able to design a single bus CPU, then we explain how data is moving from register to register, register to memory, and how they are all synchronizing with time as we have said that everything happens at the positive edge of a clock. So, even if I have changed these signals like R in out, memory buffer is going to in, but they will all take effect in the next coming edge.

So, if you are able to design that of course, you are going to satisfy this synthesis objective that we design the timing sequence generator to carry out the proper micro operations at a proper time. So, in fact, if the four questions you are able to answer, you are going to you are actually meeting all the objectives. With this, we come to this end of this unit; and from next unit onwards, we will be looking more into the depth of how control signals are generated if there is multiple buses, what are the changes expected and so forth.

Thank you.