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 a 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.

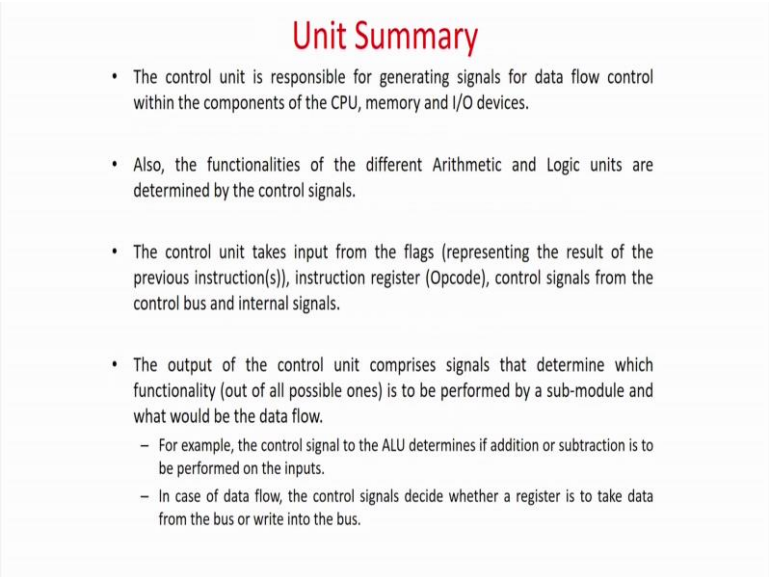
Basically in today's module, now we will see basically for given each of the microinstructions, what are the control signals required, exactly which block of the CPU generates those signals, and what are the timing sequence for that? And we will be understanding that in a more depth or a more what do 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)

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)



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 that is internal to the CPU data transfer, data transfer between the CPU and the memory or the I/O 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.

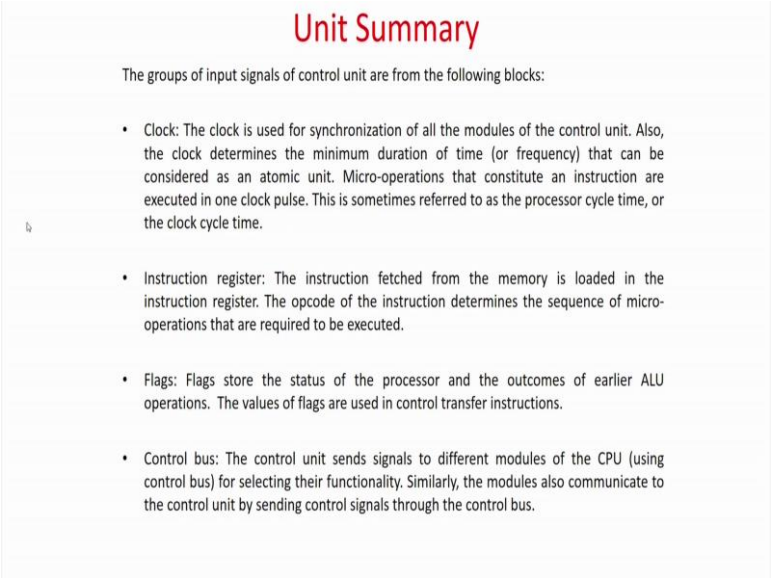
Also we will see that how basically the different functions of the arithmetic and logic block like whether it to be added, to subtract to, go for a shift, how signals are generated at by the instruction register, and how ALU is controlled by that. Then we will see basically we will also see in a black box manner that what is the control unit, what are the inputs it takes from, it takes basically the inputs 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 loaded in the instruction register basically the Opcode decides that what are 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 has to 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 we will also see basically that of the main heart of the control unit 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 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 we 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 functionality 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 control unit that is one of the main part we will study.

(Refer Slide Time: 04:17)



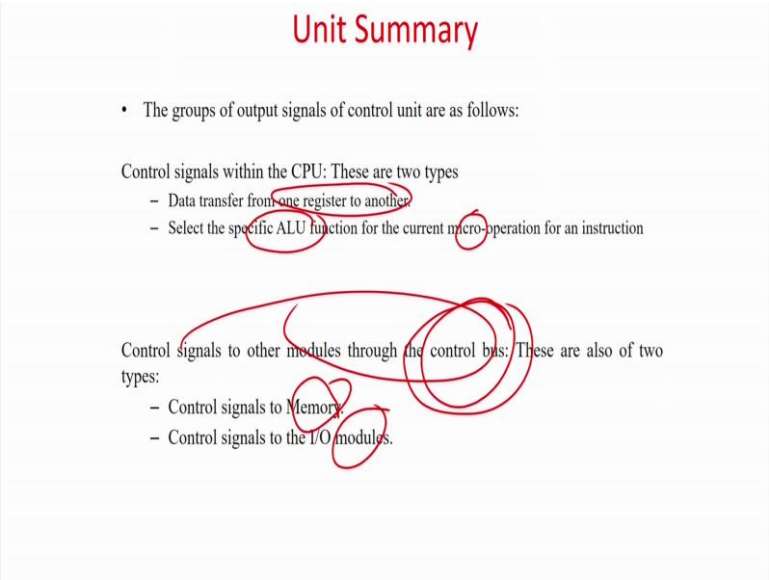
Then basically we will see, what are the very important chunks of signals for that? Of course, very important chunks of signals will be as I told you the instruction register. Instruction register which will consist of the instruction is the main heart that will actually command the control unit that this is the opcode of the instruction now you have to do this, for example ADD. So, the opcode for the ADD, which will be present in the instruction register will ask the control unit to generate such a signal, so that the ALU goes to the addition mode. So, in fact, so the 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 set 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 instruction registers also the control unit will depends 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 ship 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 module in the control unit, and in fact we assume that in a one clock pulse one microinstruction is occurred or can be processed.

(Refer Slide Time: 06:06)



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 are 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 some 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.

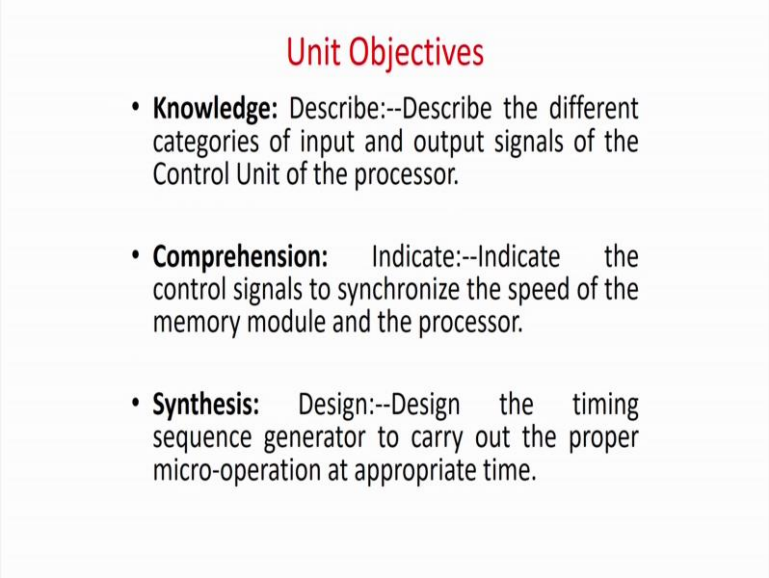
Similarly, if you want to transfer some data register from data from one register to another, so you need not send any control signal outside the control unit. So, basically this is some kind of internal control signals. Of course, the control unit will also generate some controls for the

external parts like your memory, your I/O modules. For example, as I told you that you want to write something to the memory, you give some address and then you get the data, but then you have to wait then the memory says 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, that is 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 I/O modules in that case they are drawn through the control bus.

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

(Refer Slide Time: 08:12)

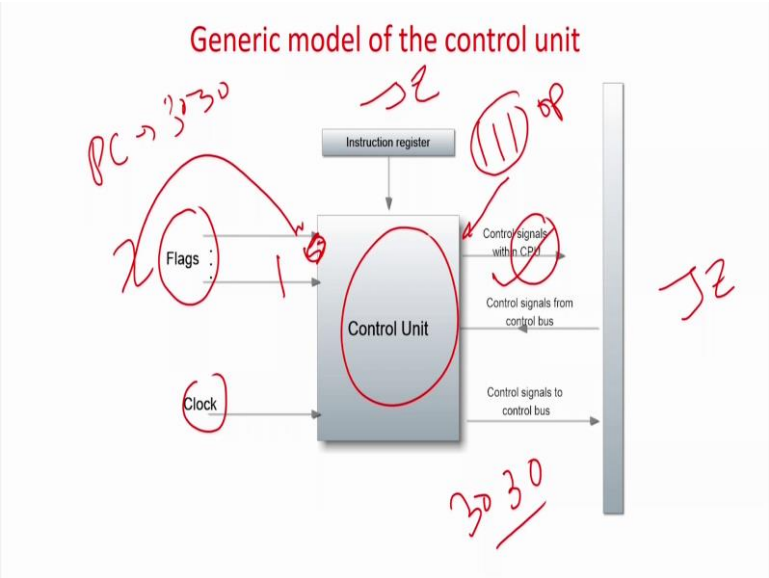


So, what are the basic objectives of this unit, basically you will be able to first is the knowledge objective, you will be able to describe the different categories of input and output signals of the control unit. That if you take a black box of the control unit, then you have to take you will be able to tell which is the inputs and the outputs. Next 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 to 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 unit.

(Refer Slide Time: 09:14)



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

Now why is flag so important because I told you if you have the instruction say called 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 z is an opcode. Say for example, the opcode is 111 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 it's a conditional jump.

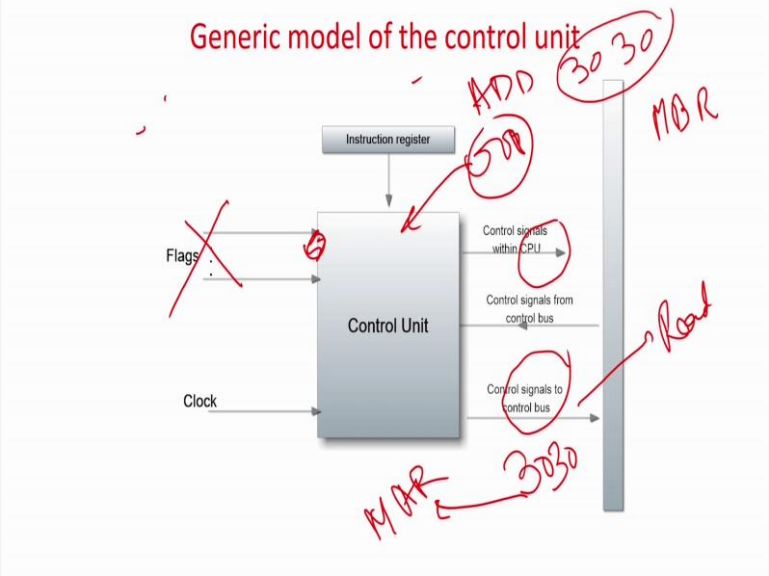
So, this input from the instruction register that is the opcode part, because we assume that the jump instruction jump on zero to some memory location is already loaded in the instruction register, the opcode corresponding to that will generate the opcode which will actually control the control unit, 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 it's a jump instruction. So, you want to know previously what was my input or what exactly was the operation done whether if I have subtracted two number, whether the answer was a 0; if it is the case 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 subtraction 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 111 that is the opcode, and of course, zero flag was set that means in fact, it was 1. Sorry I mean it is set 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 3030. So, in fact, the program counter will be now loaded with say 3030, the program counter will be loaded with the operation 3030. And you already know that program counter is a register which is 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 3030. So, in that case simply your next instruction will start from 3030 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 3030 from the memory location you have to load 3030. So, in that case what happens maybe the opcode of ADD is 001. So, in this case, the 001 will be fed from the instruction register to the control unit. In that case flag is not required because you are just going for a ADD and you have to load from memory location 3030. 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 3030.

Now, it will generate some data 3030 which will be fed to memory address register because you have to read from memory location 3030 which is to be fed to the memory address register. They are now all internal CPU instructions, but actually if you have to 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 3030 in the memory address register all those things are there like that memory address register is an internal register CPU. So, it is an internal signal you have to load memory address register with 3030.

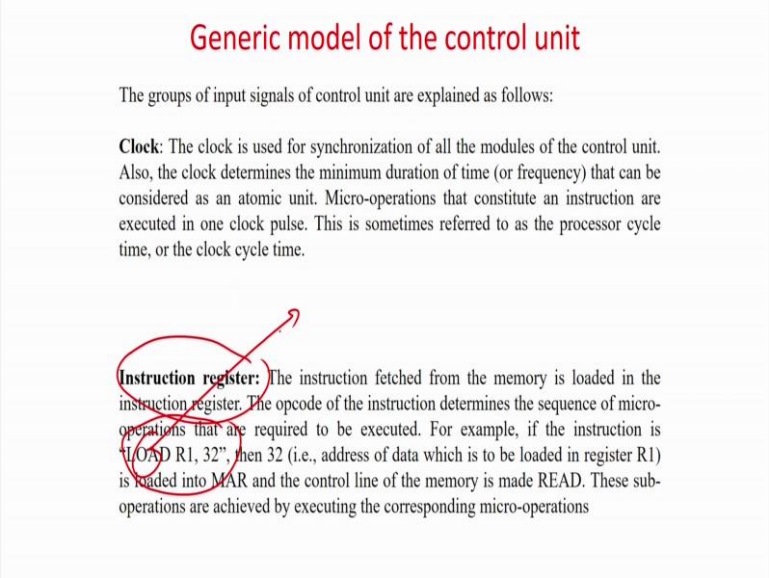
But you have to make the signal to the memory saying that it's a read signal. So, it will then generate some signal to the control bus, which will actually ask the memory saying that now we have to go to a read mode. Of course, after that, what will happen the memory will dump the value of 3030 in the memory buffer register. Now, what happens now you have to wait for certain time till when the memory will load the value of 3030 the whatever is maintained by

3030 to the memory buffer register; till that time the control unit has to wait because immediately it cannot read the memory buffer register it takes some time.

You are giving the 3030 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 3030 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 enabled then you can read the value from the memory buffer register which will be again taken by the data bus, which 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 signals which is for example, memory or I/O 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 configuring the arithmetic logic unit etcetera, so that is basically the generic model of a control unit.

(Refer Slide Time: 14:37)

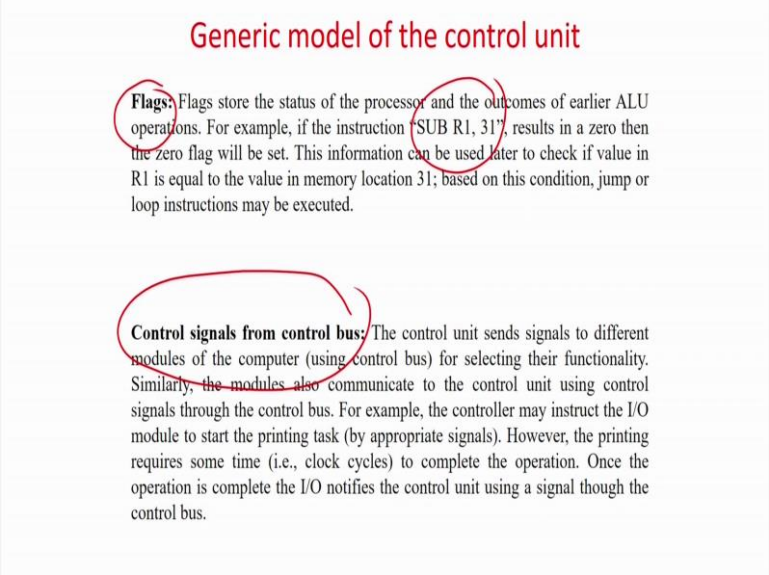


So, as I was telling now whatever I discussed they are actually written in the text, so that you can read it. So, as I already told you clock is nothing but a synchronization module as I told you we have not talked anything about the clock over here. But whenever all the instructions

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 control unit, it tells exactly what to do say like for example, it is saying $LOAD\ R1$, 32 that means, the whatever is present in the location 32 has to be loaded in $R1$. As I told you, so the opcode corresponding to load will actually direct the control unit to do that. So, as I told you so it's a heart of the control.

(Refer Slide Time: 15:22)



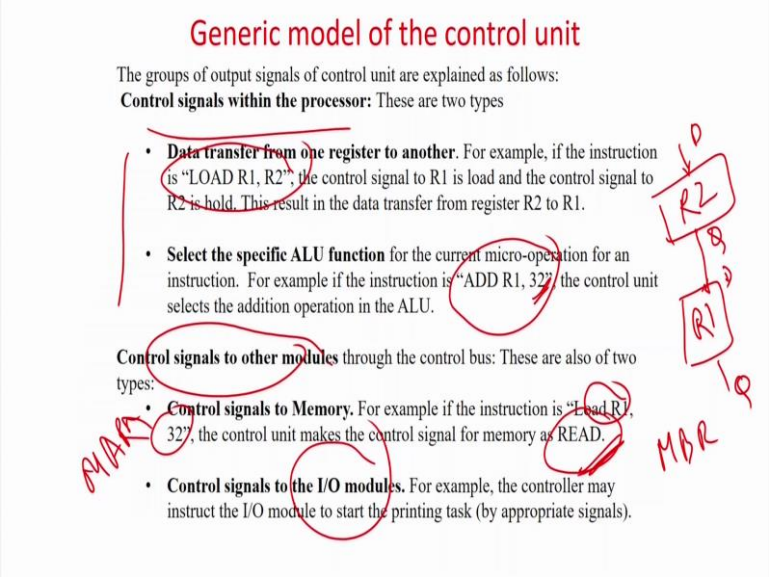
There is a flag as I told already explained that flag is very important because you have to remember what happened previously. Like for example, if there is a sub instruction and it said that $R1$- 31 may have been 0, so the zero flag has been set, but next after that you are using a conditional instruction that you should jump if $R1 = 31$. That means, whether the value of 31 is equal to the whatever value is in the register $R1$ 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 write operation with memory buffer register now you can read, so that signal will be taken from the control unit. For example you want to

make the memory 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)



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 $LOAD\ R1,R2$. You can very easily understand that both the registers are in the CPU, so the control unit will just make $R1$ into load mode and $R2$ in the output mode.

For example because registers are nothing but this is one set of flip flop which is $R1$ and then another set of flip flop which is $R2$ 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 $R2$ that is Q will be made connected to the input D of the $R1$. So, $R1$ output will be loaded into $R2$ at the clocking. So, simply such connections will be done.

In fact, you digital fundamentals, if you forgot you forgotten you should go understand go and read about multifunction register that means, a multifunction register can do multiple things it can load from other register, it can load from input, it can freeze its input, 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 $R2$ that is Q will be connected to D of $R1$, so you will load $R1$ to sorry $R2$ to $R1$. So, in that case it's a simple load operation, so the control unit will generate these signals for the multiplexers of $R1$ and $R2$. 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 $ADD\ R1$, 32 in this case, for example, if it's a specific instruction say as I assume that it's an immediate operation that 32 is basically nothing but you have to add $R1$ 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 $R1$ 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 $LOAD\ R1$, 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 bus 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 signals is an internal load. But the 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 read external command to the control bus.

Then after some time what will happen the memory will load the value whatever is available in 32 memory buffer register 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 register, so that will be an input. Similarly, if you have connected an I/O bus; obviously control signals has to be there from the I/O bus for synchronization.