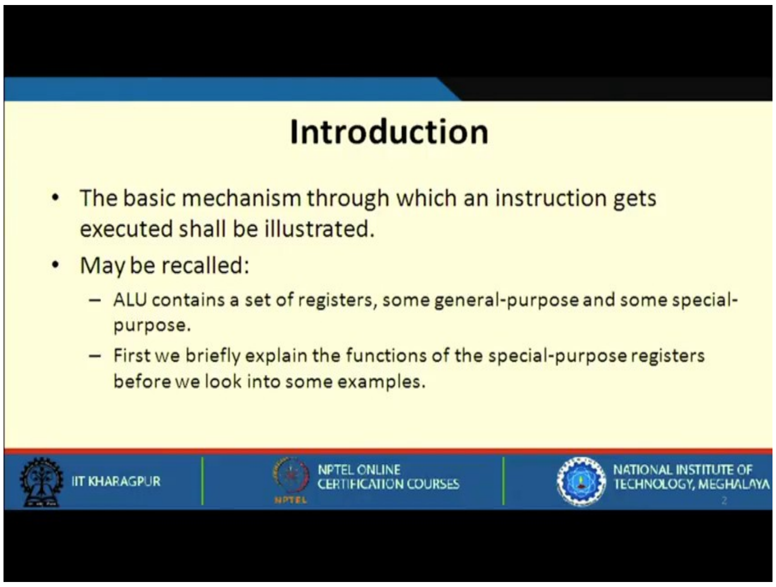
Computer Architecture and Organization

Prof. Kamalika Datta
Department of Computer Science and Engineering
National Institute of Technology, Meghalaya

Lecture - 02
Basic Operation Of A Computer

(Refer Slide Time: 00:27)



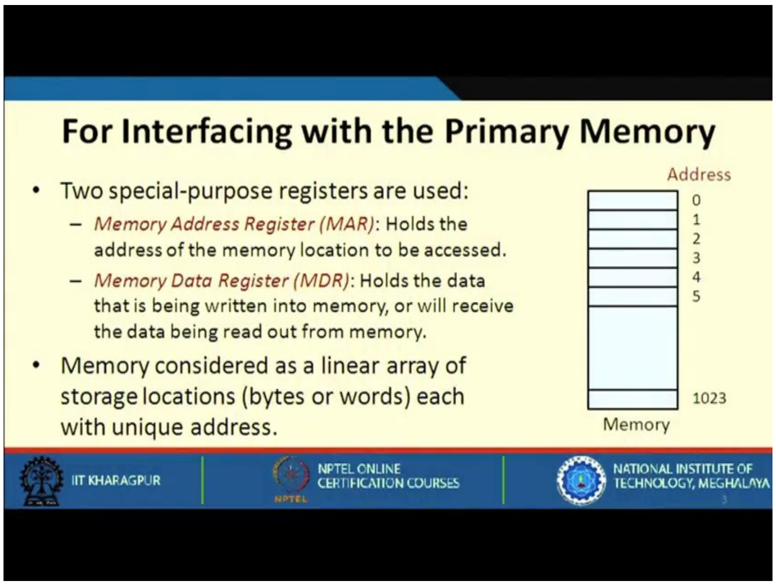
Introduction

- The basic mechanism through which an instruction gets executed shall be illustrated.
- May be recalled:
  - ALU contains a set of registers, some general-purpose and some special-purpose.
  - First we briefly explain the functions of the special-purpose registers before we look into some examples.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA
2

Welcome to the next lecture on basic operation of a computer. So, the basic mechanism through which an instruction gets executed shall be illustrated. And just recall that what we discussed in the previous lecture. Your ALU is having some registers, some registers are called special purpose registers and some are general purpose registers. And firstly, what we will do we will discuss some function of special purpose register.

(Refer Slide Time: 01:01)


For Interfacing with the Primary Memory

- Two special-purpose registers are used:
  - *Memory Address Register (MAR)*: Holds the address of the memory location to be accessed.
  - *Memory Data Register (MDR)*: Holds the data that is being written into memory, or will receive the data being read out from memory.
- Memory considered as a linear array of storage locations (bytes or words) each with unique address.

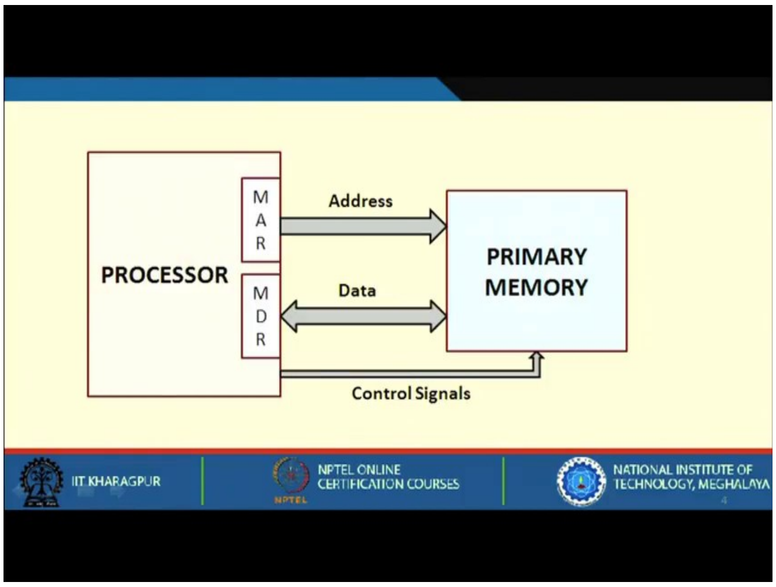
Address
0
1
2
3
4
5
1023
Memory
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

Instructions and data are stored in memory. And from memory you have to bring the data and the instruction to the processors and then you have to execute it. For this purpose, we require two special purpose registers they are called Memory Address Register(MAR) and Memory Data Register (MDR). So, let us see first what is memory address register. So, memory address register holds the address of a memory location to be accessed. So, when I say that it holds the address of memory location that is to be accessed that means, I can access a memory location for reading an instruction, I can access a memory location for reading a data and I can also access a memory location for writing back the data. So, memory address register holds the address of the instruction that is to be read or it holds the address of the data that is to be read from the memory or the address of the memory where the data is to be written.

The next register is called memory data register. Memory data register or MDR holds the data that is been written into memory or the data that, we will receive when read out from the memory location. So, when we write we always write a data into the memory, but when we read as I said, I can read an instruction or I can read a data. So, this memory data register will contain the instruction that is to be read from the memory or the data that is to be read from the memory or the data that is to be written into the memory, it can contain any of these three.

Now, you can see this that addresses it is spanning from 0 to 1023. So, the number of memory locations that we can address is starting from 0 to 1023, hence 1024 memory locations in total. So, a memory is considered as a linear array of storage locations, bytes or words, each with a unique address. So, these are the addresses of the memory location where it is incremented one by one and it has got 1024 locations. Now, what I am trying to say is that this memory address register will hold one of such address that is either 0, 1, 4 or 5 or anything 1011 anything and the memory data register will hold the content of that location. If it is 5, whatever content will be there in that particular location that will be present in MDR.

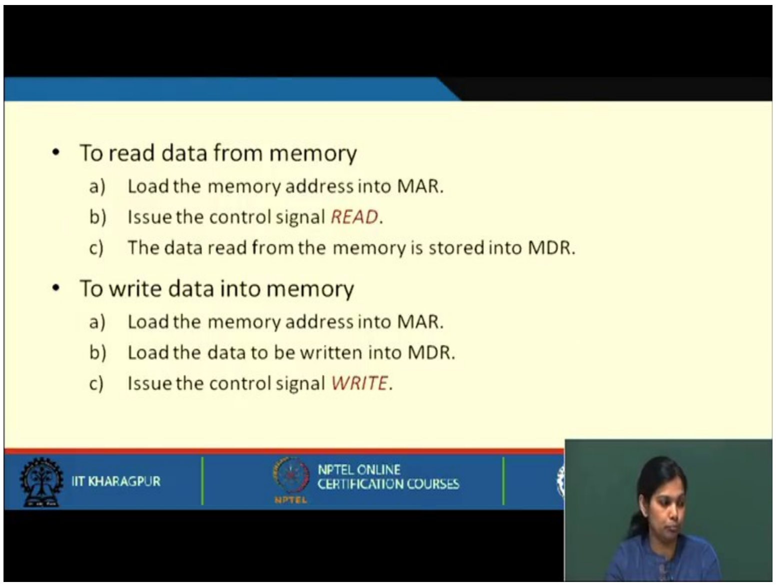
(Refer Slide Time: 04:33)


PROCESSOR
M A R
M D R
Address
Data
PRIMARY MEMORY
Control Signals
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

Now, just see this diagram it shows the connection between processor and main memory. So, I talked about MAR and MDR. MAR is Memory Address Register and MDR is Memory Data Register. Memory Address Register is connected with primary memory through address bus. Memory Data Register is connected with primary memory through data bus. And some control signals are also required. So, why control signal are required. So, when I say that this particular data will be read from the memory or this particular data will be written into the memory, so we need to specify that information to the memory that from this particular location we need to read a data or from this particular location we need to write back data. Control signals are for this purpose.

First we provide the address in the address bus that hits to the primary memory then we provide the control signals either Read or Write depending on that, a particular value is read from that particular address and it comes to MDR. Or if I want to write a value, I have to put that value in MDR and the address where I want to write in MAR, and then I activate the write control signal. Hence according to read or write a word is read or written from or into memory.

(Refer Slide Time: 06:40)

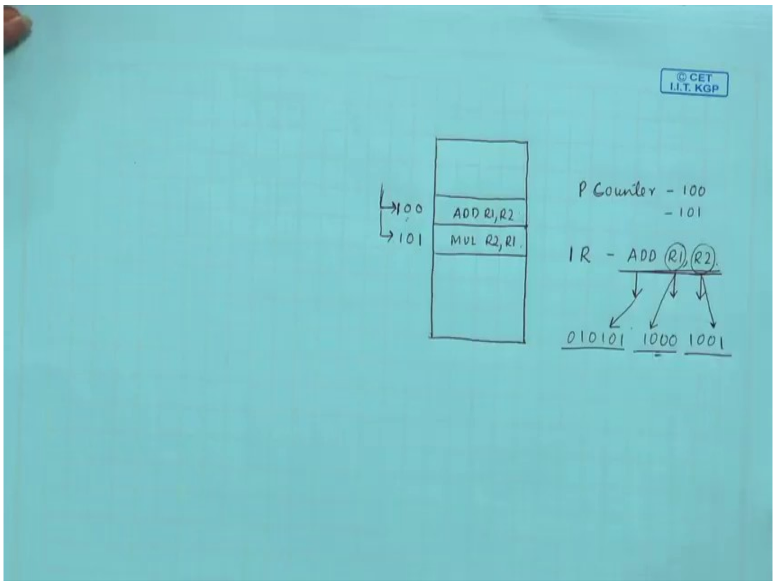


- To read data from memory
  a) Load the memory address into MAR.
  b) Issue the control signal *READ*.
  c) The data read from the memory is stored into MDR.
- To write data into memory
  a) Load the memory address into MAR.
  b) Load the data to be written into MDR.
  c) Issue the control signal *WRITE*.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

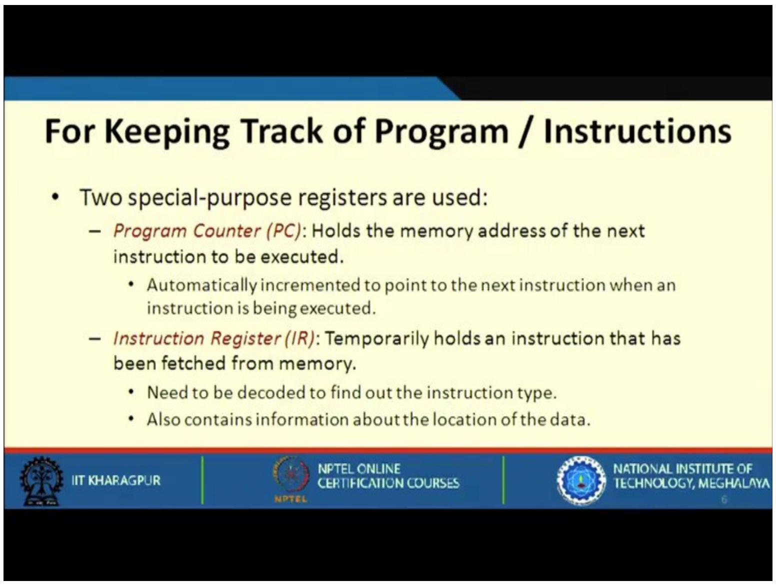
So, as I said this slide will summarize it. To read the data from memory, we first load the memory address into MAR then we issue the control signals i.e., read then the data from memory is read and it is stored into MDR. Now, to write into memory what we were doing as I said we have to load the memory address into MAR the data that is to be written must be loaded into MDR and then we issue write control signal. So, for reading these are the following steps that are required and for writing these are the following steps we have to issue.

(Refer Slide Time: 07:49)



Now, for keeping track of the program what I said like let us say this is my memory, these are some locations. So, this is say 100 location, this is 101 location and in these locations we have some instructions. Now, suppose you are executing at this particular location how will you move to the next location. So, for that reason there must be some counter that will point to location 100. So, you will go to location 100, you will fetch the instruction, execute the instruction. Then your counter should automatically get incremented to 101; such that after executing this instruction, you move to the next instruction and execute the other instruction.

(Refer Slide Time: 09:11)



For Keeping Track of Program / Instructions

- Two special-purpose registers are used:
  - *Program Counter (PC)*: Holds the memory address of the next instruction to be executed.
    - Automatically incremented to point to the next instruction when an instruction is being executed.
  - *Instruction Register (IR)*: Temporarily holds an instruction that has been fetched from memory.
    - Need to be decoded to find out the instruction type.
    - Also contains information about the location of the data.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

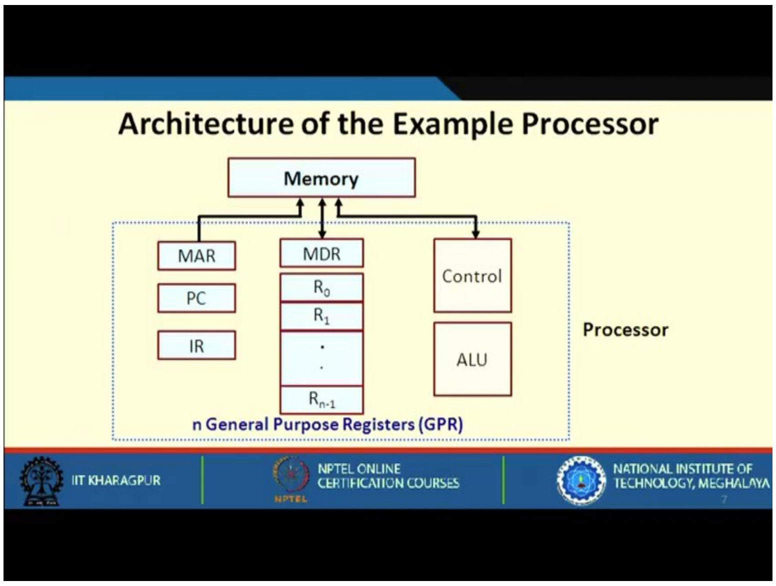
So, now we see for this purpose we have two special purpose registers. One is called Program Counter (PC) that holds the memory address of the next instruction to be executed, automatically it is incremented to point to the next instruction when an instruction is fetched and about to be executed. So, PC holds the memory address of the next instruction to be executed. So, once we read and execute an instruction, PC now must point to the next instruction that I have to execute next and so on.

Next, another register is the Instruction Register. So, now you see in this diagram, when I say that this is the location. So, PC will contain the location from where I have to read the instruction. Now, I have to read this instruction and store it somewhere, the register where it will get stored is known as Instruction Register where this entire instruction ADD R1,R2 is stored. So, this register Instruction Register temporarily holds an instruction that has been fetched from memory. Now, once an instruction is fetched from memory, you need to decode that instruction. See we understand this is add, but a computer is a layman. So, you have to instruct the computer do this, do that, then only the computer will do that.

So, when I say that ADD R1,R2. So, add is an instruction saying that add the content of register R1 with R2, and store back in R1. So, the instruction needs to be decoded and then the computer must understand, now it has to execute the instruction add the content of R1 and R2 and, store back the result in R1. So, the instruction register temporarily holds an instruction that has been fetched from memory, needs to decode to find out the instruction type, which contains information about the location of the data.

IR contains the entire instruction. So, after we get this instruction we need to decode to know this is add, and we also need to know locations of R1, R2 registers. So, ultimately after decoding we will understand this add R1, R2 is nothing but some bits of 0's and 1's. If a total of 16 registers are present then we need four bits to specify any one of the 16 registers and we can just name that R1 is a four bit register and it is represented by 1000. You can also say R2 is another register which is represented by 1001. And add is an opcode which is represented by some bits. So, ultimately this set of instruction is nothing but bits of zeros and ones. So, we will see this in near future.

(Refer Slide Time: 13:31)

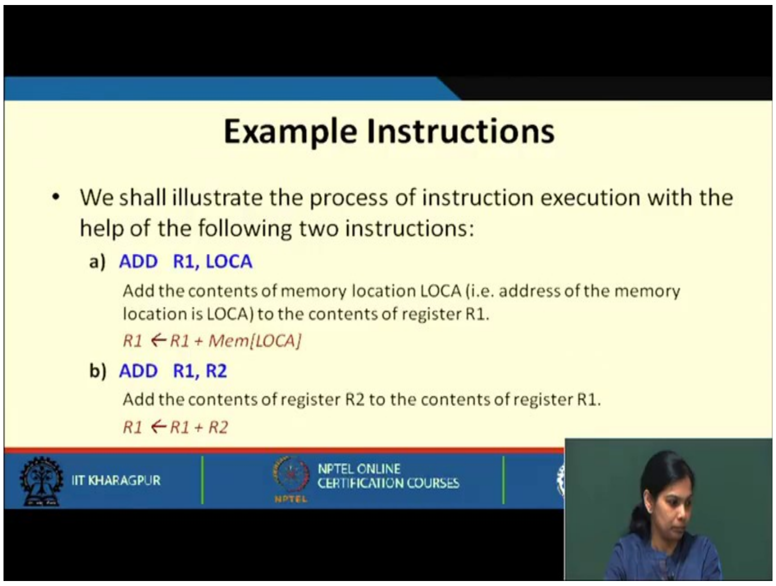


Architecture of the Example Processor

Memory
MAR
PC
IR
MDR
R_0
R_1
R_{n-1}
Control
ALU
Processor
n General Purpose Registers (GPR)
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

Now, we can see the architecture of an example processor or in other words we can say that how memory is connected with the processor. So, this is a processor where we have some special purpose registers MAR, MDR, PC and IR you already know the functions of these registers. We have some general purpose registers and we have control unit and ALU. Now, what is the function of this ALU? When I say that ADD R1,R2, R1 and R2 are registers that are present inside your processor. So, these registers we need to bring to ALU so R1 must be brought to ALU and R2 must be brought to ALU and then the particular instruction like add or mul is executed on R1 and R2 and we get the result. So, this is an example processor how it looks like, but there will be many more things that we will see later.

(Refer Slide Time: 15:02)



Example Instructions

- We shall illustrate the process of instruction execution with the help of the following two instructions:
  - a) ADD R1, LOCA

    Add the contents of memory location LOCA (i.e. address of the memory location is LOCA) to the contents of register R1.

    R1 ← R1 + Mem[LOCA]
  - b) ADD R1, R2

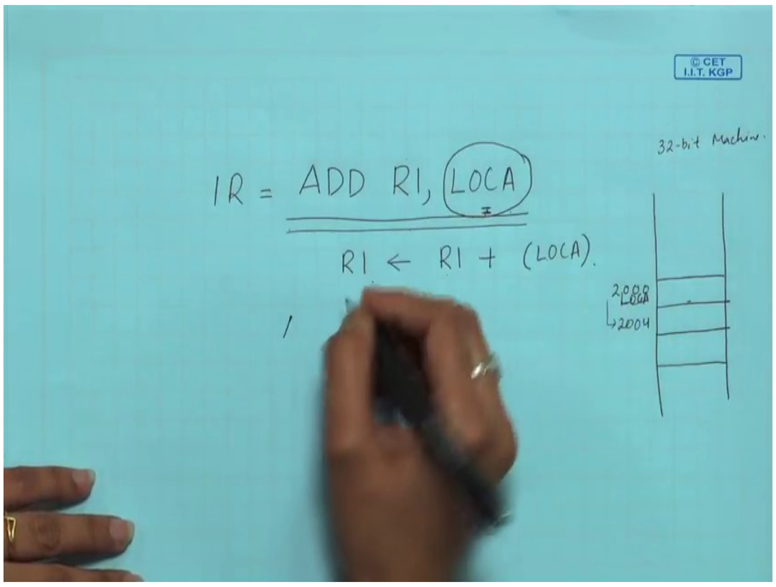
    Add the contents of register R2 to the contents of register R1.

    R1 ← R1 + R2

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

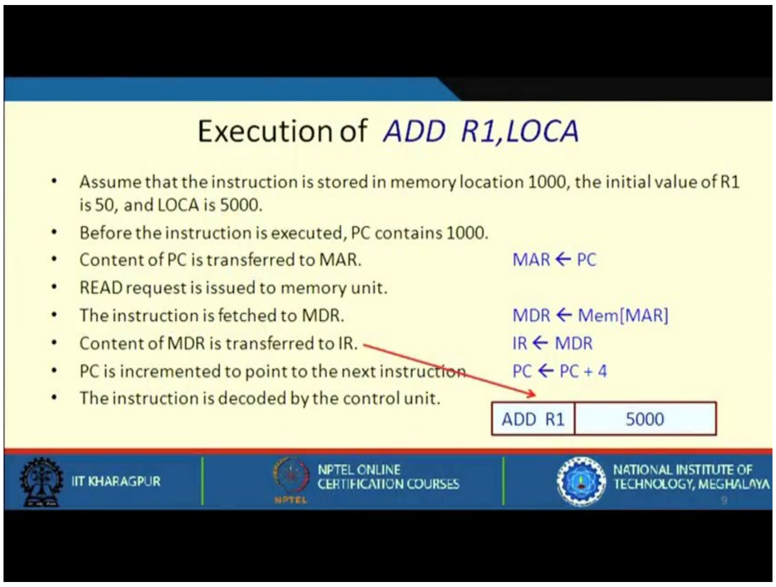
Now, we shall illustrate the process of instruction execution with the help of following two examples. So, the two examples that we will be taking the first one is ADD R1, LOCA.

(Refer Slide Time: 15:24)



The instruction ADD R1, LOCA will add the content of R1 and location A and store back the result in R1. LOCA is a location in memory. So, this is your memory and this is some location. The content of this will be added with R 1 and will get stored back in R 1. The next one is simply add the two registers that is R1 and R2 both are present in processor and store back the result in R1.

(Refer Slide Time: 16:45)



Execution of *ADD R1,LOCA*

- Assume that the instruction is stored in memory location 1000, the initial value of R1 is 50, and LOCA is 5000.
- Before the instruction is executed, PC contains 1000.
- Content of PC is transferred to MAR. MAR ← PC
- READ request is issued to memory unit.
- The instruction is fetched to MDR. MDR ← Mem[MAR]
- Content of MDR is transferred to IR. IR ← MDR
- PC is incremented to point to the next instruction. PC ← PC + 4
- The instruction is decoded by the control unit.

ADD R1	5000

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

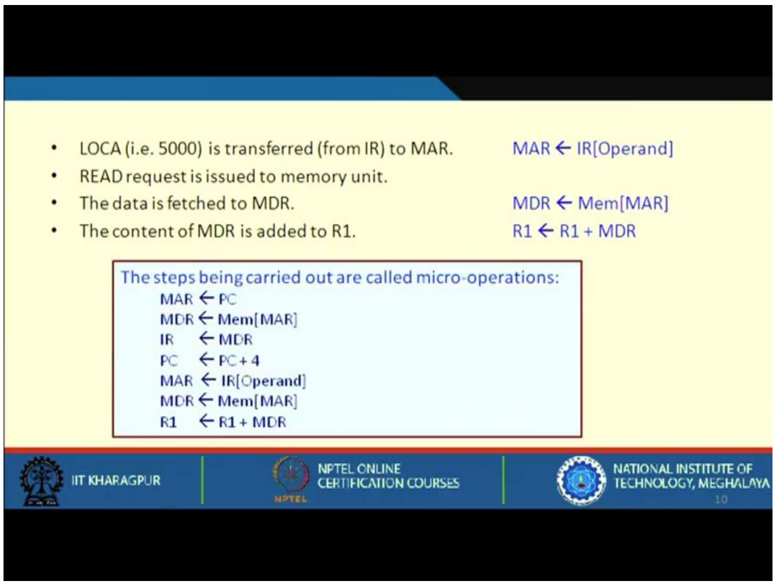
Now, let us see how we will execute this instruction. So, we have to do some assumption here. As I said that firstly, when we say this instruction, this instruction is stored in some memory location. We assume here that the instruction is stored in location 1000, and the initial value of R 1 is 50, and LOCA value is 5000. Before the instruction is executed PC contains the value 1000. Now, the content of PC is transferred to MAR as we need to read the instruction. To read the instruction from memory what I said earlier that the address needs to be loaded in MAR, and we need to activate the read control. Hence 1000 now should be transferred to MAR, then a read request is issued to memory unit.

After the reading is performed, the instruction is in MDR; and then from MDR, it should be transferred to IR Instruction Register. And while doing these steps, we have to increment the PC to point to the next instruction. As I said in computer there will be sequential execution of instruction and the instructions are stored one by one. So, here first PC was pointing to 1000, we fetch the instruction from location 1000 and then the PC should point to the next location that is 1001. So, PC will now point to the next location which is 1001. And next whatever instruction that we have read from memory is loaded in MDR; and from MDR, it will be transferred to IR. And we know after it has come to IR the instruction needs to be decoded and then it has to be executed. We will see step by step how it happens.

So, from this we can say that firstly, PC value is transferred to MAR. Read signal is activated. The content specified by the memory location(MAR) is read and it is stored in MDR. From MDR, it is transferred to IR and at the same time PC is incremented to 4. Why we have said here 4, it is depending on the word size, we will be coming to this little later, but it depends on what will be the word size. So, I will just explain it once. Let us consider a byte addressable. So, let us say if this location is 2000, and the next location is 2004 that means, we are using a 32-bit machine; and the PC is incremented by 4. If it is a 64-bit machine, the PC will get incremented by 8. So, here the PC is incremented by 4, so it is a 32-bit machine.

Now, once the instruction comes to IR it needs to be decoded and executed. Let us see how it is done. Now, location is for example, it is 5000 it is transferred from IR to MAR. So, if you consider the same instruction that is ADD R1, LOCA. Now, this is in IR. After decoding it has understood that one of the operand is in memory. If one of the operand is from memory then you have to read that particular operand the data which is there in that particular location from the memory.

(Refer Slide Time: 21:54)



• LOCA (i.e. 5000) is transferred (from IR) to MAR.	MAR ← IR[Operand]
• READ request is issued to memory unit.	
• The data is fetched to MDR.	MDR ← Mem[MAR]
• The content of MDR is added to R1.	R1 ← R1 + MDR

The steps being carried out are called micro-operations:

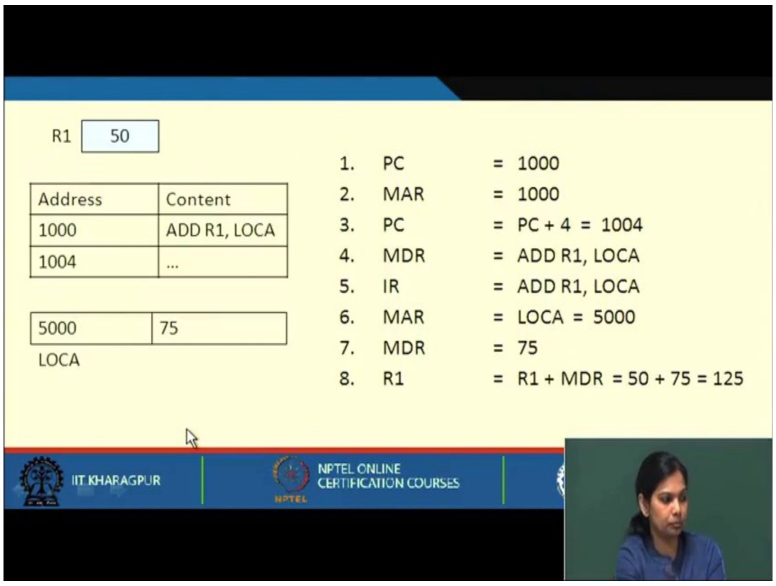
```
MAR ← PC
MDR ← Mem[MAR]
IR  ← MDR
PC  ← PC + 4
MAR ← IR[Operand]
MDR ← Mem[MAR]
R1  ← R1 + MDR
```

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA
10

So, you have to load that location into MAR, then activate the read control signal then the data that is present in that location LOCA that is 5000 will be read and will be transferred to MDR. Now the MDR contains one of the operand and R1 contains one of the operand. These two will be brought into your ALU which will be added and stored back in R1. So, these are the steps that we follow to execute the instruction ADD R1, LOCA. In this instruction, how many read operation we have to do the first read operation we perform to fetch the instruction. And after decoding we again see that one of the operand is a memory location. So, we again have to read that particular location from the memory, we again read that particular data from that memory location, we execute the instructions and store back the result. So, two memory operations were required in this particular instruction.

Let us move on with the next one. So, these are the steps that are being carried out and these are called micro operation. We will see in more detail later, but just to see first PC value is transferred to MAR. Then whatever value of MAR is read from memory and it is stored in MDR. Now, from MDR the value is moved to IR. The PC gets incremented to point to the next memory location. If we see that there are some operands that is to be read from memory, then it has to be read and then transferred to MAR. And then read signal is activated and data is read and is brought into MDR. And finally, now all my data are present in processor both R1 is present in processor after reading MDR the data of LOCA is also present in MDR. So, both are processor register we need to add this and store the result in R 1.

(Refer Slide Time: 24:32)



R1 50

Address	Content
1000	ADD R1, LOCA
1004	...

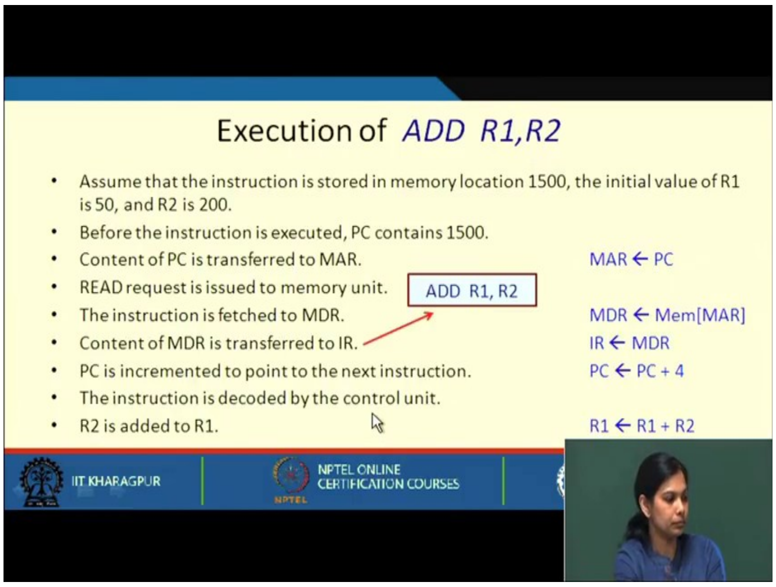
5000	75

LOCA

1. PC = 1000
2. MAR = 1000
3. PC = PC + 4 = 1004
4. MDR = ADD R1, LOCA
5. IR = ADD R1, LOCA
6. MAR = LOCA = 5000
7. MDR = 75
8. R1 = R1 + MDR = 50 + 75 = 125

So, whatever I have explained is that PC contain this value, MAR contain this value. PC get incremented it points to 1004 then MDR initially have the instruction ADD R1, LOCA that is moved to IR. Then after decoding LOCA value will be in MAR the data will be read and in that location the value is 75 which is stored in MDR. And finally, after adding this the value is stored back in register R 1.

(Refer Slide Time: 25:14)

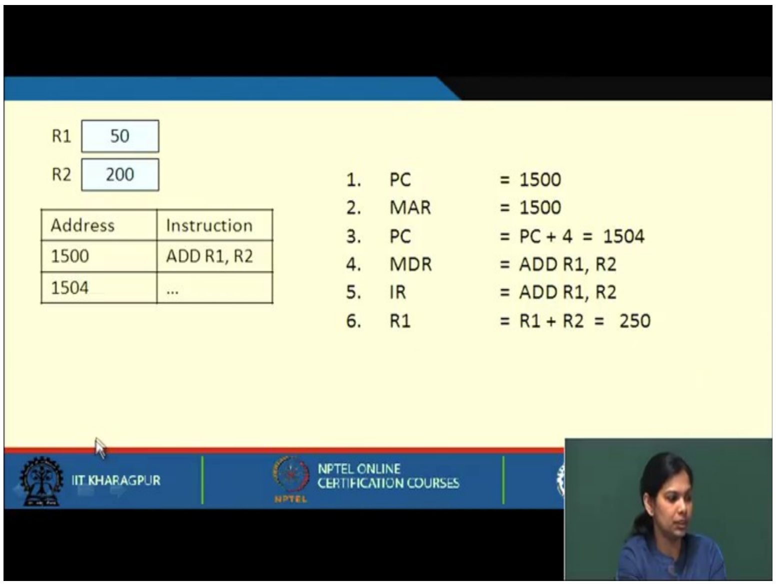


Execution of *ADD R1,R2*

- Assume that the instruction is stored in memory location 1500, the initial value of R1 is 50, and R2 is 200.
- Before the instruction is executed, PC contains 1500.
- Content of PC is transferred to MAR. MAR ← PC
- READ request is issued to memory unit.
- The instruction is fetched to MDR. MDR ← Mem[MAR]
- Content of MDR is transferred to IR. IR ← MDR
- PC is incremented to point to the next instruction. PC ← PC + 4
- The instruction is decoded by the control unit.
- R2 is added to R1. R1 ← R1 + R2

In a similar way, let us see execution of another instruction that is ADD R1,R2. Here you can see that in this instruction both R1 and R2 are processor register. So, you need not have to bring anything from memory. So, both the operands are present in your processor register; all you need to do is that you have to read this particular instruction from the memory, and then you will execute this instruction. So, assume that the instruction is stored in memory location 1500, the initial value of R1 and R2 are 50 and 200. Before the execution, PC contains 1500 and then the content of PC is transferred to MAR. We activate the read control signal the instruction is fetched and it is stored in MDR. Then the content of MDR is transferred to IR, it is decoded. And then finally, the content of R1 and R2 that is 50 and 200 are added and stored back in R1.

(Refer Slide Time: 26:34)



R1 50

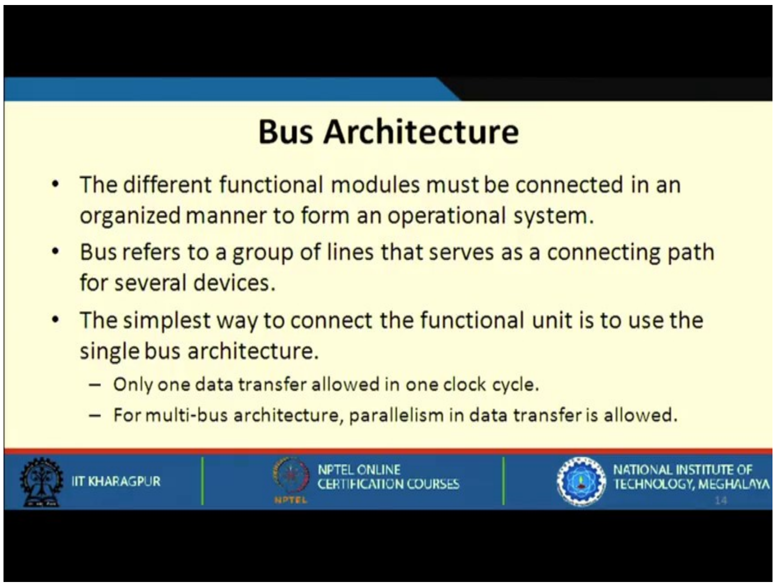
R2 200

Address	Instruction
1500	ADD R1, R2
1504	...

1. PC = 1500
2. MAR = 1500
3. PC = PC + 4 = 1504
4. MDR = ADD R1, R2
5. IR = ADD R1, R2
6. R1 = R1 + R2 = 250

So, these are the steps that are shown. PC will contain 1500 then MAR will also contain that. PC will get incremented by 4. And then the entire instruction is brought into MDR. IR will also contain this instruction, this instruction will now get decoded and the value of R1 and R2 will be added, and will be stored back in R1 that is 250.

(Refer Slide Time: 27:39)



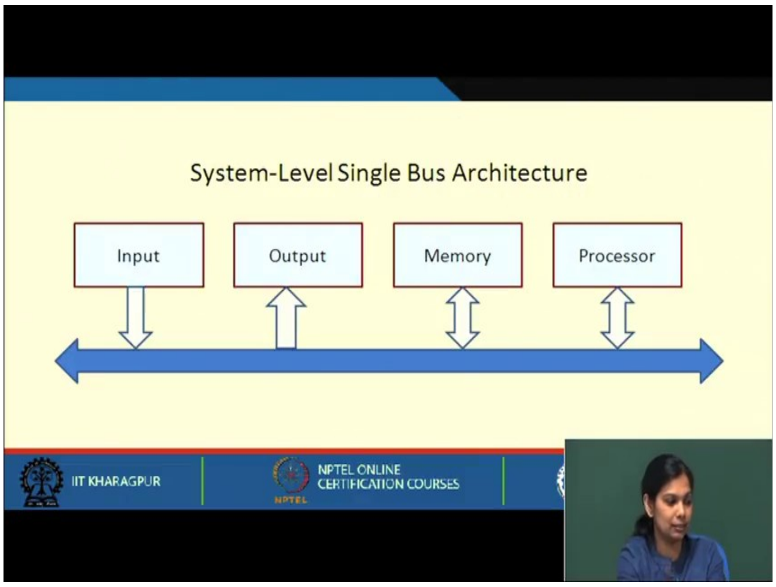
Bus Architecture

- The different functional modules must be connected in an organized manner to form an operational system.
- Bus refers to a group of lines that serves as a connecting path for several devices.
- The simplest way to connect the functional unit is to use the single bus architecture.
  - Only one data transfer allowed in one clock cycle.
  - For multi-bus architecture, parallelism in data transfer is allowed.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

Now, coming to the bus architecture. So, we were discussing about how an instruction will get executed. When we say that processor is a module, memory is a module, input output is a module, so all these are communicating between each other. So, they need a communicating pathway to communicate between each of these functional units. So, the different functional modules must be connected in an organized manner to form an operational system. By bus what we refer is to a group of lines that serve as a connecting path for several devices. So, the simplest way to connect all these is through a single bus architecture, where only one data transfer will be allowed in one cycle. For multi bus architecture parallelism in data transfer is allowed.

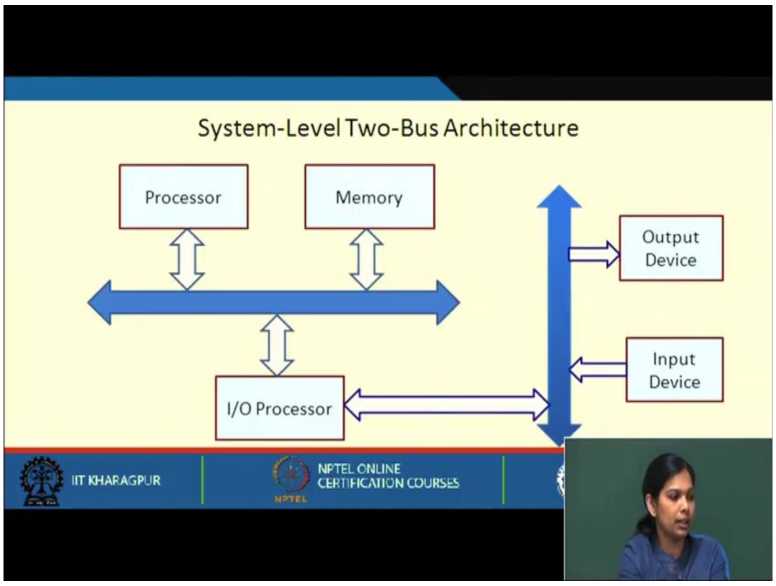
(Refer Slide Time: 28:15)



System-Level Single Bus Architecture

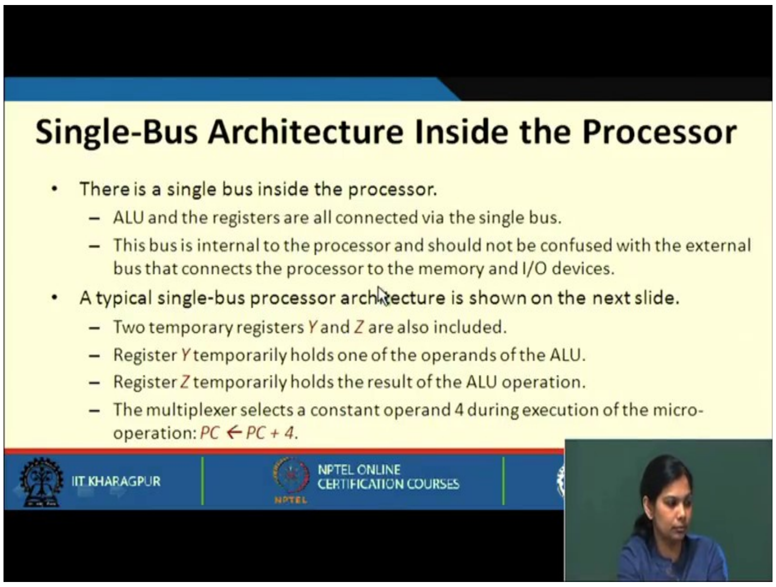
So, let us see this. So, this is a single bus that we can see where all our modules, memory, processor, input and output are connected to a single bus. So, if the processor wants to communicate with memory, it has to use this bus and no other modules will be using that at that moment. In the same way if from memory something needs to be sent to output device, no other module can communicate. This is a bottleneck of a single level system bus.

(Refer Slide Time: 28:54)


System-Level Two-Bus Architecture
Processor
Memory
Output Device
Input Device
I/O Processor
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

Now, coming to the system-level two-bus architecture. In this two-bus architecture, what we can see is that there is a bus dedicated to processor memory and of course, the IO processor is also connected to it. But for input and output there is a separate bus and this bus will be communicating with the IO processor in turn and the IO processor will then be communicating with the processor or the memory as and when it is required. So, when there is more communication between processor and memory then we must have a bus dedicated for this.

(Refer Slide Time: 29:43)

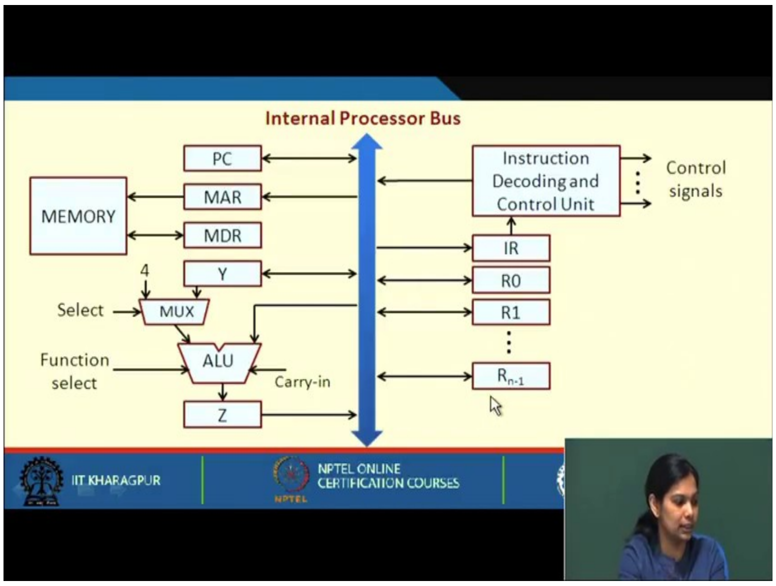


Single-Bus Architecture Inside the Processor

- There is a single bus inside the processor.
  - ALU and the registers are all connected via the single bus.
  - This bus is internal to the processor and should not be confused with the external bus that connects the processor to the memory and I/O devices.
- A typical single-bus processor architecture is shown on the next slide.
  - Two temporary registers *Y* and *Z* are also included.
  - Register *Y* temporarily holds one of the operands of the ALU.
  - Register *Z* temporarily holds the result of the ALU operation.
  - The multiplexer selects a constant operand 4 during execution of the micro-operation: $PC \leftarrow PC + 4$.

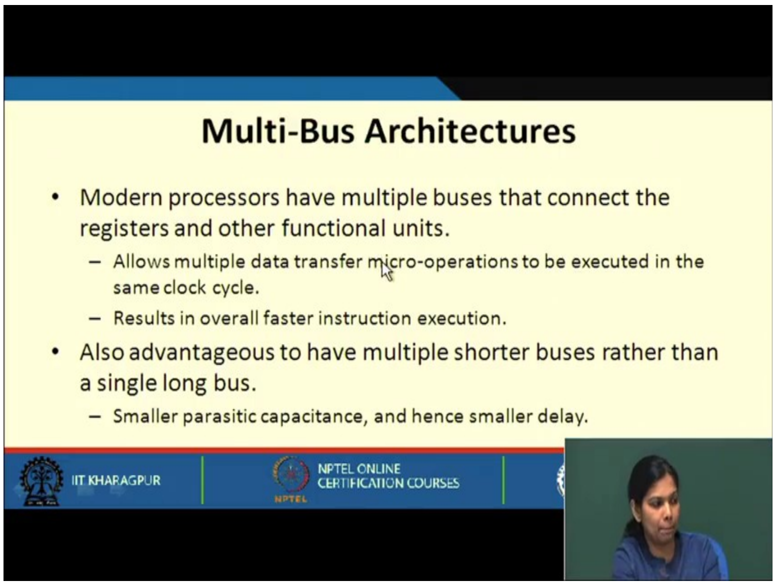
Now we can also see a bus that is required within the processor. Within the processor, we have seen that there are many components many transfers are taking place. So, whenever we are moving some register value within the processor, we also need a bus within the processor. So, in that bus, there are other components like, ALU and the registers that are all connected via this single bus. The bus is basically internal to the processor. So, I am talking about a bus which is inside the processor. Typical single bus processor architecture is shown in the next slide.

(Refer Slide Time: 30:49)



This is an internal processor bus. So, all these things are inside the processor PC, MAR, MDR, Y register is there and then ALU is there. There are some temporary registers Z, Y and there are some general purpose registers; IR is also there and there is a instruction decoding and control unit. So, information is getting transferred within all these modules. If I have to transfer a data from R1 to R2 or I have to transfer a data from PC to MAR or have to transfer a data from MDR to R1 or R2 or to ALU, some control signals needs to be generated. So, how they will be communicating, they will be communicating through this internal processes bus.

(Refer Slide Time: 31:44)



Multi-Bus Architectures

- Modern processors have multiple buses that connect the registers and other functional units.
  - Allows multiple data transfer micro-operations to be executed in the same clock cycle.
  - Results in overall faster instruction execution.
- Also advantageous to have multiple shorter buses rather than a single long bus.
  - Smaller parasitic capacitance, and hence smaller delay.

Now, multi-bus architectures are also there. So, modern processors use this multi-bus architecture. Here, within your processor to communicate between various registers you have multiple-bus. The advantage we get is that more operations can be performed and we get results much faster. So, there will be a overall improvement in instruction execution time. Some smaller parasitic capacitance can be there and hence smaller delay.

So, we have come to the end of lecture 2. In lecture 2, what we have studied how an instruction actually get executed, various processor registers, various registers that it present inside the processor. And finally, how we can increase the performance using either single-bus architecture or multi-bus architecture.

Thank you.