

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

**Lecture - 17**  
**Designs Of Control Unit (Part 1)**

Welcome to week 4. In this week we shall be looking into the design of control unit. So, till now what we have seen? We have seen how an instruction gets executed; what are the hardware blocks that are required for the execution of the instruction. And for executing that instruction, basically some steps are required. That we are saying that, the content of the PC will go to MAR. And then that particular content in MAR will hit the memory through the address bus, and then we will get the data. And then something else will happen. So, for any instruction certain steps are required to be executed. And for executing those steps we require some hardware, like the registers, the ALU, and other interconnecting blocks, other registers like IR, PC and all other.

So, in the design of control unit, we will be seeing what are the signals that needs to be generated for executing an instruction. So, we will be seeing this in course of the lectures that are there in week 4.

(Refer Slide Time: 01:54)

**Instructions**

- Instructions are stored in main memory.
- Program Counter (PC) points to the next instruction.
  - MIPS instructions are 4 bytes (32 bits) long.
  - All instructions starts from an address that is multiple of 4 (last 2 bits 00).
  - Normally, PC is incremented by 4 to point to the next instruction.

12							
8							
4							
0							

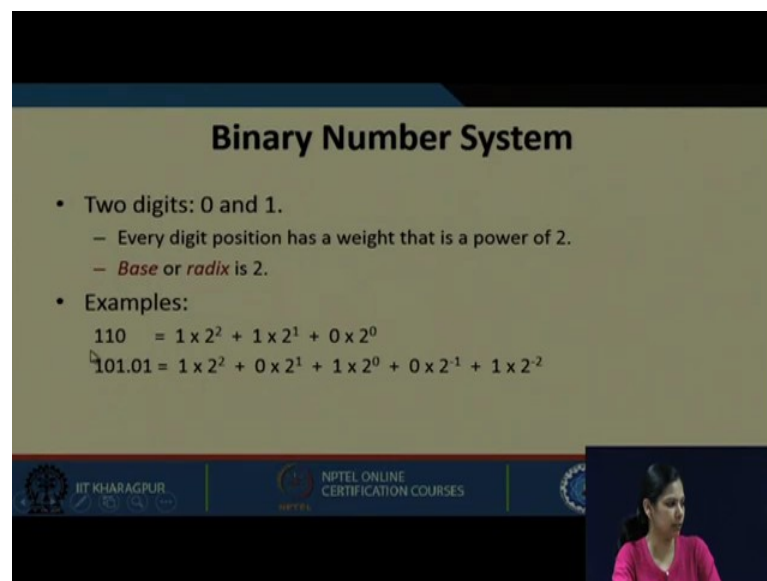
The diagram shows a grid with 8 columns and 4 rows. The rows are labeled on the left as 12, 8, 4, and 0. The first row (12) is empty. The second row (8) is empty. The third row (4) is labeled 'instruction word'. The fourth row (0) is labeled 'instruction word'. Vertical dashed lines are present in each column.

IT Kharagpur | NPTEL ONLINE CERTIFICATION COURSES | NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

We already know that instructions are stored in main memory. The program counter points to the next instruction. So, here we have an instruction, next the PC points to this

location, then the next one, and so on. So, each time an instruction is fetched, the PC gets incremented to the next one, such that once the execution of this particular instruction is completed, the next instruction can be fetched, and again the PC get incremented and so on. Generally, MIPS instructions are all 4 bytes or 32 bits long. All instruction starts from an address that is some multiple of 4. Last two bit is will be 0. And normally the PC gets incremented by 4 to point to the next instruction. We know about this particular thing from the very beginning.

(Refer Slide Time: 03:03)



**Binary Number System**

- Two digits: 0 and 1.
  - Every digit position has a weight that is a power of 2.
  - *Base or radix is 2.*
- Examples:
  - $110 = 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$
  - $101.01 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2}$


The slide also features logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES, and a small video inset of a woman in a pink shirt.

Let us recall that in binary number system we have 2 digits 0 and 1. This is how it is represented; both in binary and in decimal; addressing a byte in memory. We know that memory is byte addressable.


(Refer Slide Time: 03:19)

### Addressing a Byte in Memory


- Each byte in memory has a unique address.
  - Memory is said to be *byte addressable*.
- Typically the instructions are of 4 bytes, hence the instruction memory is addressed in terms of 4 bytes (word length = 32 bits).
- When an instruction is executed, PC is incremented by 4 to point to the next instruction.



IIT KHARAGPUR



NPTEL ONLINE CERTIFICATION COURSES

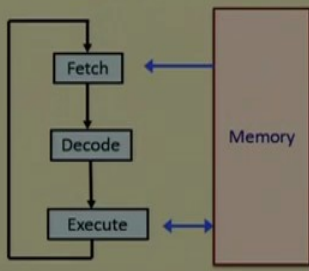


So, each byte in a memory has a unique address, not each bit rather each byte. So, typically the instructions are of 4 bytes. Hence the instruction memory is addressed in terms of 4 bytes, that is word length is 32 bits. When an instruction is executed, PC is incremented by 4. What if it is 64 bits or 8 bytes; it will be incremented by 8 and so on.


(Refer Slide Time: 04:05)

### How an instruction Gets Executed?


```
repeat forever
  // till power off or
  // system failure
{
  Fetch instruction
  Decode instruction
  Execute instruction
}
```




```
graph TD
    Memory[Memory] --> Fetch[Fetch]
    Fetch --> Decode[Decode]
    Decode --> Execute[Execute]
    Execute <--> Memory
```



IIT KHARAGPUR



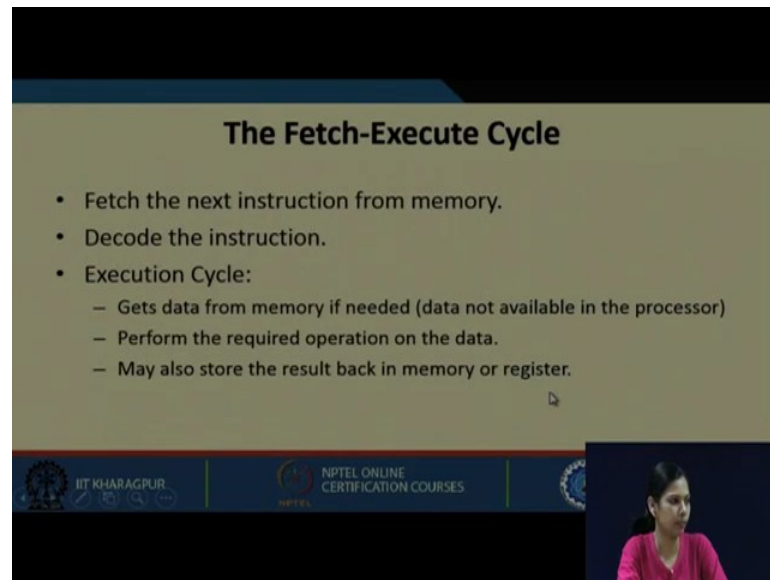
NPTEL ONLINE CERTIFICATION COURSES



Now, let us see how an instruction gets executed. So, what happens this is memory. We fetch the instruction and then we decode that particular instruction. After decoding, whatever necessary thing needs to be done for that instruction, are performed. That is, we

execute that particular instruction. And then we move on to the next one, from the value pointed by PC. And again we do the same thing. Again we fetch, we decode and we execute.

(Refer Slide Time: 04:51)



**The Fetch-Execute Cycle**

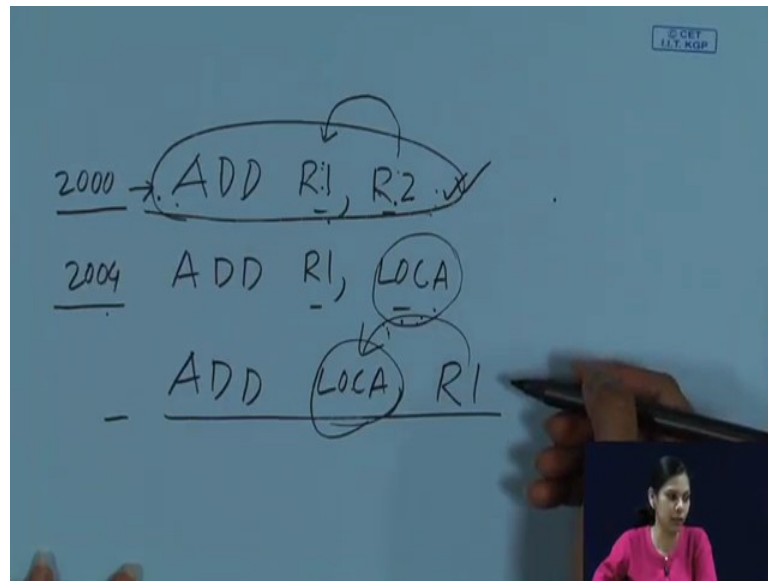
- Fetch the next instruction from memory.
- Decode the instruction.
- Execution Cycle:
  - Gets data from memory if needed (data not available in the processor)
  - Perform the required operation on the data.
  - May also store the result back in memory or register.

Logos: IIT KHARAGPUR, NPTEL ONLINE CERTIFICATION COURSES, NPTEL

Video inset: A woman in a pink shirt.

So, this is the fetch-execute cycle. We fetch the next instruction from memory, decode the instruction and execute the instruction. In the execution process what we do. It might happen we have various kinds of instruction. Let us say, we can have an instruction which is `ADD R1,R2`. We can also have an instruction called `ADD R1,LOCA`. In this particular case we can say the operands are present in the processor register as well.

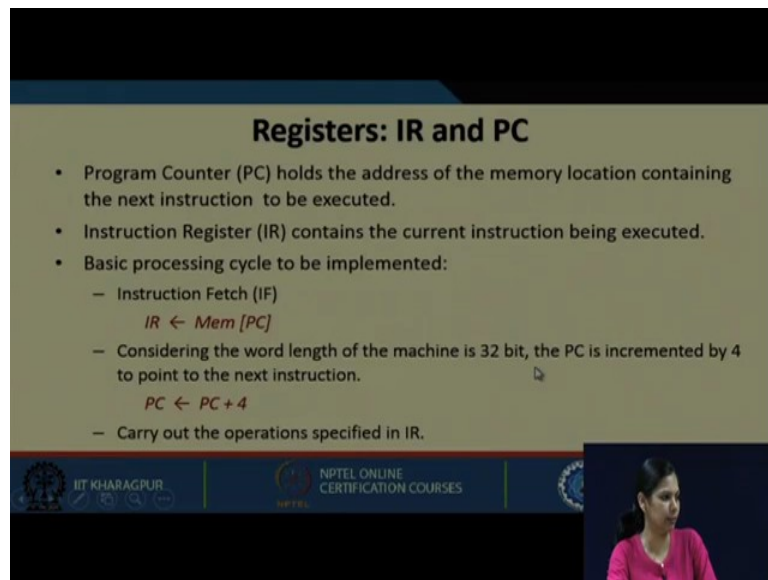
(Refer Slide Time: 05:14)



In this case, we need to bring a data from memory and then we can process this data. So, in this case no further memory access will be required, but for this particular case memory access will be required. So, in the execution cycle once we decode the instruction, we know that this is the addressing mode, and so on and so forth. Now we need to execute it. So, we get the data from memory, if needed data is not available in the processor. We perform the required operation on data, and may also store the result back in memory or in register as and when it is required.

So, in these two particular cases, we need not have to store back in memory. But in this particular case, it might happen we need to store the result back in memory. So, we first fetch this instruction, then we fetch this particular data, then we perform the operation, and finally, we store back in this particular memory location. So, this is the fetch-execute cycle.

(Refer Slide Time: 06:54)



**Registers: IR and PC**

- Program Counter (PC) holds the address of the memory location containing the next instruction to be executed.
- Instruction Register (IR) contains the current instruction being executed.
- Basic processing cycle to be implemented:
  - Instruction Fetch (IF)  
 $IR \leftarrow Mem[PC]$
  - Considering the word length of the machine is 32 bit, the PC is incremented by 4 to point to the next instruction.  
 $PC \leftarrow PC + 4$
  - Carry out the operations specified in IR.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

The register we already know about this is the instruction register (IR) and this is program counter (PC). Program counter holds the address of the memory location containing the next instruction to be executed. Instruction register contains the current instruction being executed. So, if this is my current instruction, then PC value is 2000. Next PC will get incremented and it will have 2004, and so on. So, the PC will point to the next instruction. Once we fetch this instruction, IR will contain this particular instruction.


So, instruction register contains the current instruction being executed. Basic processing cycle to be implemented is after PC points to the memory location where it is. So, memory location of the PC will be transferred to IR. Now IR contains the current instruction. Considering the word length of the machine is 32 bit, the PC is incremented by 4 to point to the next location. Now PC will have  $PC + 4$ . Then we carry out the operation specified in IR. So, whatever is specified in IR, we decode that and we then need to perform the specific operation.

(Refer Slide Time: 08:24)


**Example: Add R1, R2**

Address	Instruction
1000	ADD R1, R2
1004	MUL R3, R4



- a) PC = 1000
- b) MAR = 1000
- c) PC = PC + 4 = 1004
- d) MDR = "ADD R1, R2"
- e) IR = "ADD R1, R2"  
(Decode and finally execute)
- f) R1 = R1 + R2



IIT KHARAGPUR



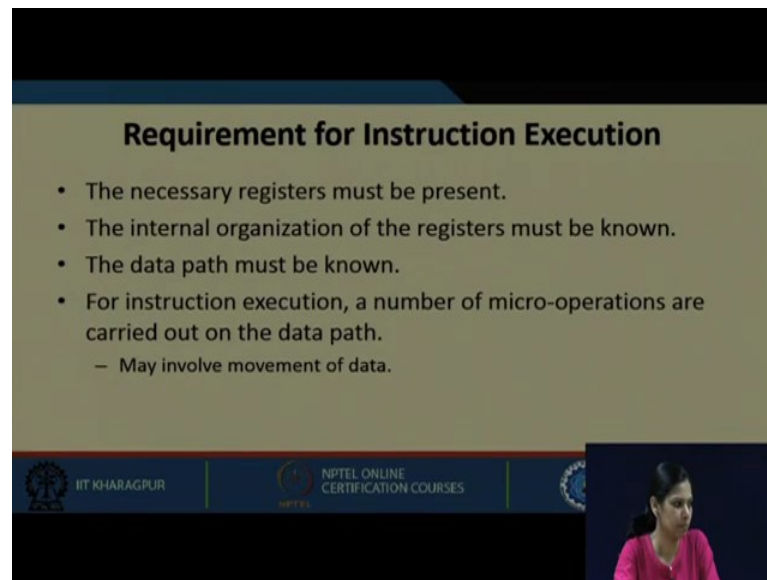
NPTEL ONLINE  
CERTIFICATION COURSES



Let us take an example. ADD R1,R2 and MUL R3,R4. So, PC initially contains 1000. MAR contains 1000. PC now contains 1004; MDR will contain this entire instruction. IR will also contain this instruction. Finally, it will get decoded and executed. And then after adding these two, the result is stored back in R1.

Now, you see these are some steps that are happening. We can see this in terms of some values because we know that this particular address is having this value, now it will go to MAR. But, if we require our computer to do this, some signals need to be generated in proper sequence, to perform this particular operation depending on certain hardware that is present.

(Refer Slide Time: 09:26)



**Requirement for Instruction Execution**

- The necessary registers must be present.
- The internal organization of the registers must be known.
- The data path must be known.
- For instruction execution, a number of micro-operations are carried out on the data path.
  - May involve movement of data.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, what is the requirement for instruction execution? The necessary registers must be present. We require to have the registers for that operation. Internal organization of the registers must be known. This is very important and this is what we will be looking into in this particular lecture, that internal organization of the registers.

What do you mean by internal organization? I say that, you will be having registers, you will be having an ALU. You will be having other registers like PC, MAR, MDR. How these are connected? We need to know internally how these are connected. MDR and MAR are connected to the memory bus. But we need to know the internal structure of the organization of the hardware, such that we must know that how the registers are connected, how the ALU is connected, to perform an operation in ALU what needs to be brought in, how it should be brought in and everything

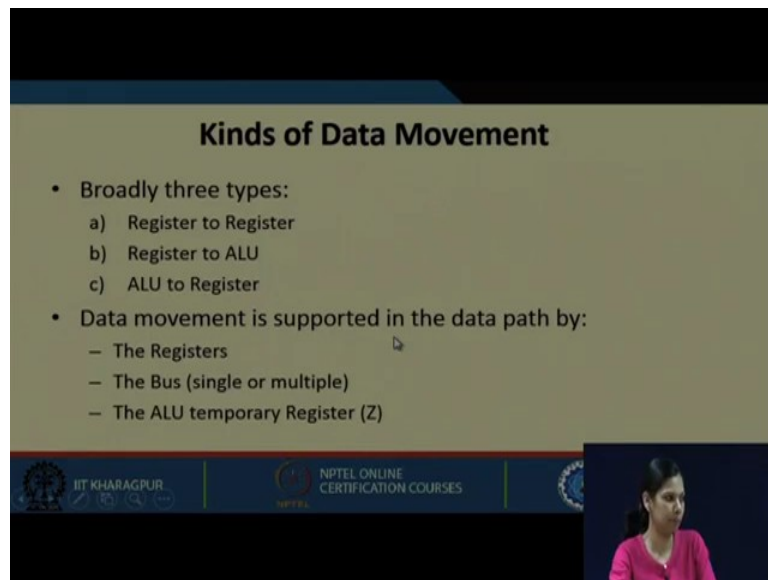
So, we need to know a complete picture of what is there inside. So, the internal organization of the registers must be known. The data path must be known, that is, we will be seeing what is data path. So, for instruction execution a number of micro operations are carried out on the data path, may involve movement of data; that means, when we are performing ADD R1,R2; how this data is actually moving? So, all these operation how it is happening we need to know. The steps that are required for execution are known as micro operations. So, micro operations should be carried out on the data path provided.



(Refer Slide Time: 11:59)

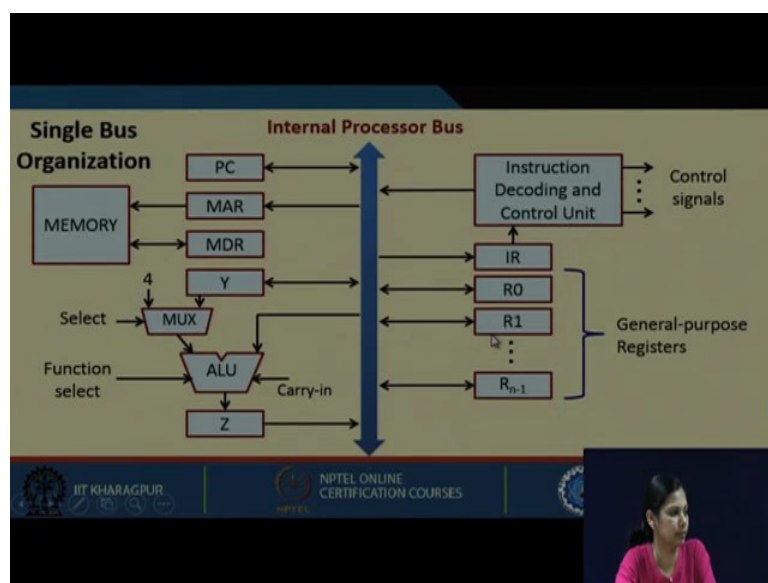
### Kinds of Data Movement

- Broadly three types:
  - a) Register to Register
  - b) Register to ALU
  - c) ALU to Register
- Data movement is supported in the data path by:
  - The Registers
  - The Bus (single or multiple)
  - The ALU temporary Register (Z)



So, let us see the kinds of data movement. Broadly it can be register to register, it can be register to ALU, or ALU to register. We will be seeing all these kind of transfers in course of time. So, the data movement are supported by the data path. And the data path contains what the registers, the bus through which the data will move, the ALU, and of course, some of the temporary registers. Some temporary registers are needed for this. So, all these together are supported in the data path. Coming to the single bus organization.

(Refer Slide Time: 12:38)

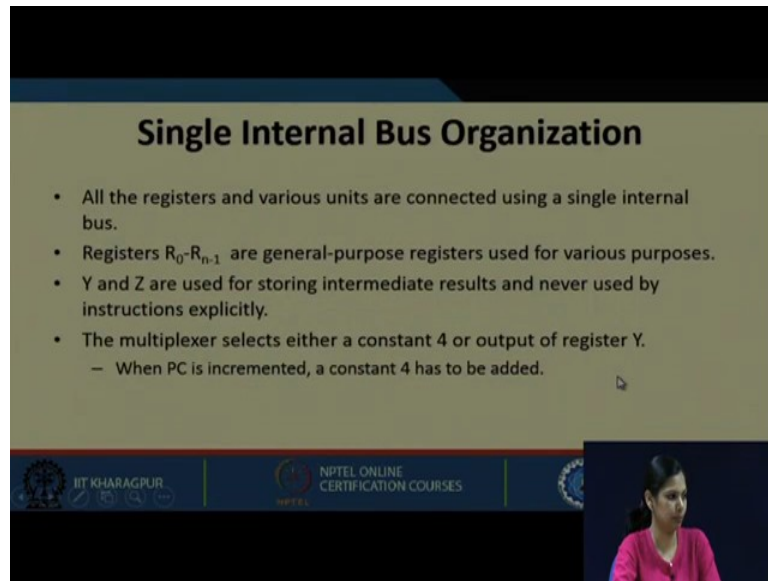


This is a very simple single bus organization that we are showing, which is internal to the processor bus. And now see, these are the buses. This is the data bus and this is the address bus, which are connected to the memory, through these two registers MAR and MDR. In this internal processor bus we have PC. So, data from this bus can come in here. And from PC also the data can be available in this bus. We have MAR and MDR.

So, this line is missing. So, there will be a connection, between internal processor bus as well, and then this connection will be there. So, a two-way connection will be there. So, data from this bus can also come into MDR, and from MDR the data can come into this processor bus. Now you see this ALU. This ALU performs the required operation. And there are two inputs of the ALU and one output. We see that one input of the ALU is directly coming from the bus. So, whatever data is there in the bus, can be directly connected to this input of the bus. If we say this is A input and this is B input, in this B input we can see that it is available. And another data is coming through this Y register. And from Y there is a MUX. Now see that MUX is selecting either the output of Y or it is selecting this 4. Why this 4 is required? We will be seeing little later.

But let us understand for the moment, that when this select line is 1, either we select 4 if the select line is 0, or we select Y depending on how you have implemented it. So, at a time either 4 comes into ALU, or Y comes into ALU. After any particular function that is performed by ALU, the data is transferred to Z register, and from this Z register the data can be available in the internal processor bus; and from this internal processor bus now the data can move to any of these registers. This is the IR, and this is the instruction decoding and control unit. Instruction decoding and control unit is required to generate the control signals. So, ultimately this unit will be generating the control signals necessary to execute an instruction.

(Refer Slide Time: 16:35)



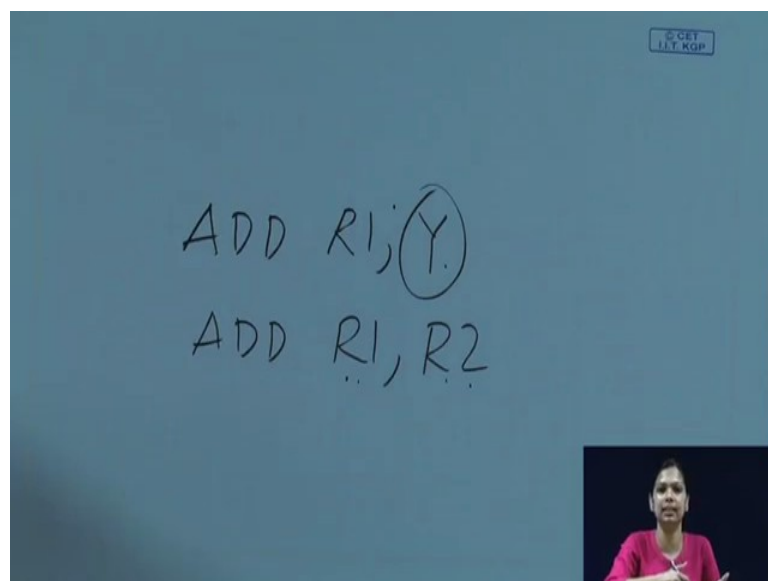
**Single Internal Bus Organization**

- All the registers and various units are connected using a single internal bus.
- Registers  $R_0-R_{n-1}$  are general-purpose registers used for various purposes.
- Y and Z are used for storing intermediate results and never used by instructions explicitly.
- The multiplexer selects either a constant 4 or output of register Y.
  - When PC is incremented, a constant 4 has to be added.

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

So, this is our single internal bus organization. So, let us see some of the features that I already discussed. This is the single internal bus organization. All the registers and various units are connected using a single internal bus. We have only one bus through which it is connected. Registers are  $R_0$  to  $R_{n-1}$ . So, we have  $n$  general-purpose registers used for various purposes. Y and Z are used for storing intermediate results. The intermediate result of any operation is stored in these registers, and they are never used by an instruction.

(Refer Slide Time: 17:31)

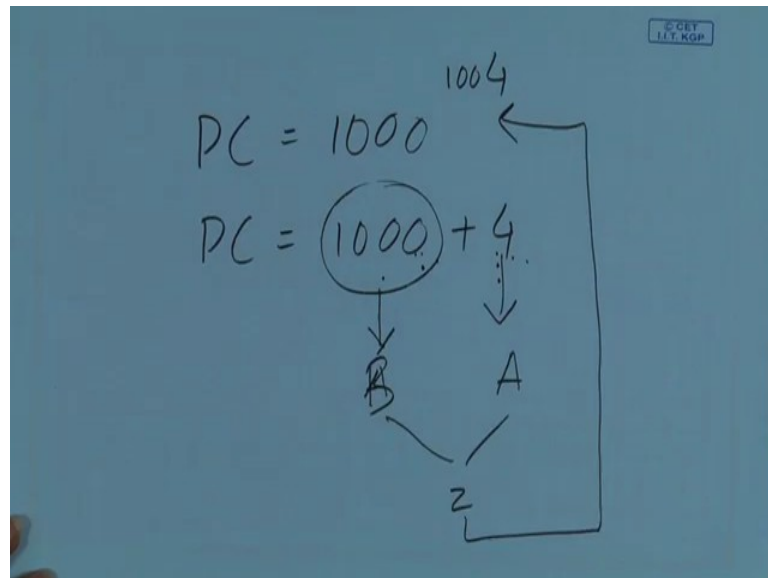


©, C.E.T. I.I.T. KGP

ADD R1, (Y)  
ADD R1, R2

This means, we will never see that we are doing something like ADD R1,Y. We will never do that. We will only have ADD R1,R2 etc, which is the general-purpose register or memory location. The MUX selects either a constant 4 or the output of register Y. When PC is incremented a constant 4 has to be added. Now understand this. What happens, when we see that this is my PC? PC is now 1000.

(Refer Slide Time: 18:25)



Now, the PC needs to get incremented. So, you have to do  $1000 + 4$ . How will you do this? We cannot simply do this; we need a circuit to do this. And for doing this, this input must come into one of the inputs of ALU. So, if it comes to any one input of the ALU, then we can add 1000 plus 4, and then the result can be stored in Z. And from there it can be again put it in PC. So, it becomes 1004 again. So, this is how it can be done. So, when PC is incremented a constant 4 has to be added.

(Refer Slide Time: 19:20)

- The instruction decoder and control unit is responsible for performing the actions specified by the instruction loaded into IR.
- The decoder generates all the control signals in the proper sequence required to execute the instruction specified by the IR.
- The registers, the ALU and the interconnecting bus are collectively referred to as the *datapath*.

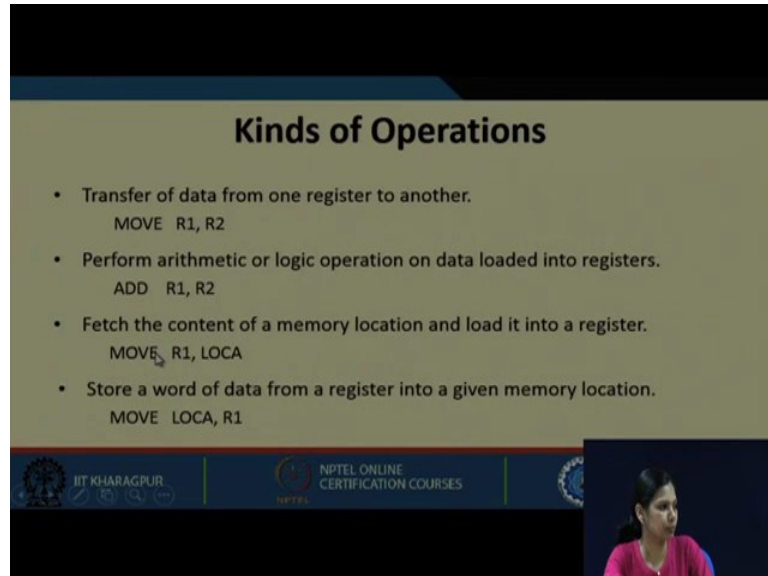
The instruction decoder and control unit is responsible for performing the action specified by the instruction loaded into IR. Now once the instruction is fetched from the memory, it comes through MDR, and then it goes to IR using that single bus. Once it is loaded in IR, it is the responsibility of the decoder unit to decode that particular instruction. And then it generates whatever needs to be done; if it has to bring the data from memory again it will do the required operation, if the data is already present in the processor register, then it has to add it or multiply it with whatever action is specified it needs to be done.

So, the instruction decoder in the control unit is responsible for performing the action specified by the instruction loaded into IR. The decoder generates all the control signals in proper sequence required to execute the instruction specified by IR. Now the decoder decodes the instruction. After decoding the instruction it generates the control signals that are required for that particular instruction in a proper sequence. Now what is data path then? The registers, the ALU, and the interconnecting bus are collectively referred to as the data path; that means, through this particular path, the data are moving for performing the operation.

So, for performing the operation it has to come to ALU. Then from ALU it has to again go to some register. So, how it is going? The registers are involved, the ALU is involved,

and the connecting bus through which the data is moving. So, data are moving through all these places. Collectively this is referred to as data path.

(Refer Slide Time: 21:40)



**Kinds of Operations**

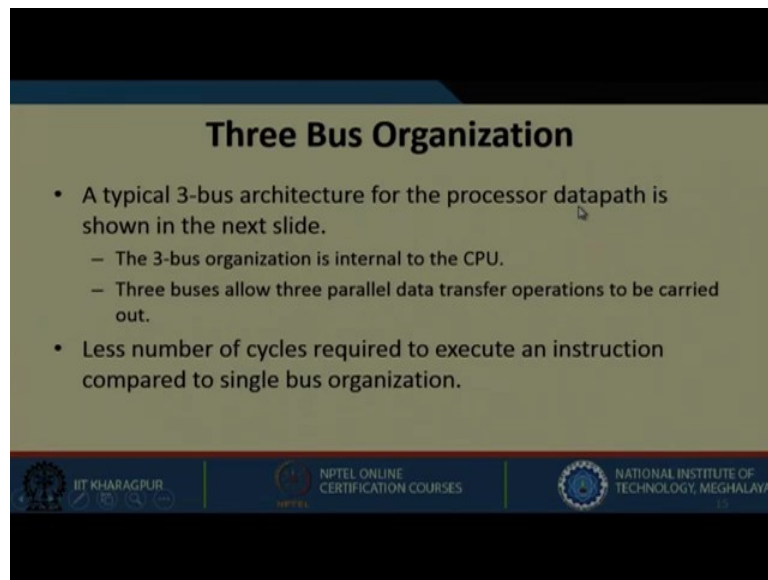
- Transfer of data from one register to another.  
MOVE R1, R2
- Perform arithmetic or logic operation on data loaded into registers.  
ADD R1, R2
- Fetch the content of a memory location and load it into a register.  
MOVE R1, LOCA
- Store a word of data from a register into a given memory location.  
MOVE LOCA, R1

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

So, now let us see what are the kinds of operations that are performed. Transfer of data from one register to another. Let us say moving a data from R2 to R1 is required. Perform arithmetic or logic operation on data loaded into register. Let us say the data is loaded in R1 and R2; all we need to do is that we perform such an operation, and store it back here. So, here also this is a kind of operation that is required. Fetch the content of memory location and load it into register, basically load.

So, we are loading a data from this memory location and we are storing it in R1. Or store a word of data from a register into the memory location. So, we are storing a word that is R1 with whatever value is stored in R1 into LOCA or memory location. So, this is for load, this is for store. So, these are the various kinds of operation that we can have.

(Refer Slide Time: 22:48)



**Three Bus Organization**

- A typical 3-bus architecture for the processor datapath is shown in the next slide.
  - The 3-bus organization is internal to the CPU.
  - Three buses allow three parallel data transfer operations to be carried out.
- Less number of cycles required to execute an instruction compared to single bus organization.

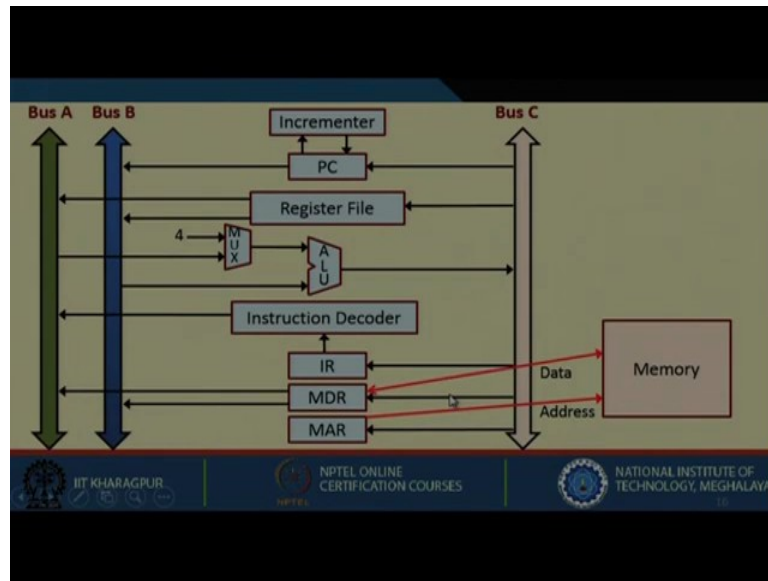
Logos at the bottom: IIT KHARAGPUR, NPTEL ONLINE CERTIFICATION COURSES, NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

Let us come to 3-bus organizations. Now in the previous case you have seen a single internal bus organization. And in that single internal bus organization we will be seeing in later lectures that only at a time, of particular value can be available in the bus. And that particular value can go to any number of registers. But at a time only one data can be available in that bus. If we want to make more data available then, what we need to do? One possible way of doing so is having multiple-bus structure.

So, what happens in multiple-bus architecture? In multiple-bus architecture we have multiple internal buses inside a processor. The MDR and MAR will be connected to the same system bus. But, internally we will not have a single bus. We will see that a single bus will restrict some operation to be done parallelly. If you want to perform some operation to be done parallelly, we require multiple-bus architecture.

So, these are just some of the features. A 3-bus organization is internal to the CPU, as I said, we will be looking into a bus organization which is internal to the CPU. We have already seen a bus organization, which is a single-bus organization. Now we will be seeing a 3-bus organization. The 3 buses allow 3 parallel data transfer operations to be carried out. Less number of cycles in turn will be required to execute an instruction, compared to single bus organization. We will be looking into this with examples later.

(Refer Slide Time: 25:05)



Now, let us see this particular multibus organization that is a 3-bus organization. In this 3-bus organization let us see what we have. This is a register file. In this register file using VLSI technology what we can do is that, we can read multiple data. But we can write in one data into the register file. So, two registers can be read at a time because we have two buses. And the data from these register file is going to two different buses. But write can happen only once, and it is coming also from a different bus. PC is incremented by a different circuitry. That is an incrementor circuit, where PC will get incremented by 4.

Now, this is the ALU; the input of ALU is coming from two different buses. One is from bus A another is from bus B. The advantage we can get here is that, we can make available the data of R1 and R2 here, and if we perform that operation and both R1 and R2 can be present at the same time. And we can also perform this ADD operation. And we can also store back here at the same time. But in a single-bus organization, only one particular data will be available in the bus at a time. As for multiple buses multiple data can be available.

This is an instruction decoder. So, after from MDR the data will be available. And then this particular data will be moved to IR, and the instruction decoder will decode the instruction and specified operation will be carried out. MAR and MDR are connected to the address and data bus of the memory as well. So, this is a 3-bus organization. We have



seen single bus organization; we have seen multiple bus organization. We will be seeing in detail what is the advantage you get in course of time when we execute a particular instruction using these bus organizations.

So, now we have come to the end of this lecture, where we have discussed about the overall internal bus organization, how internally within the CPU the buses are organized.

Thank you.