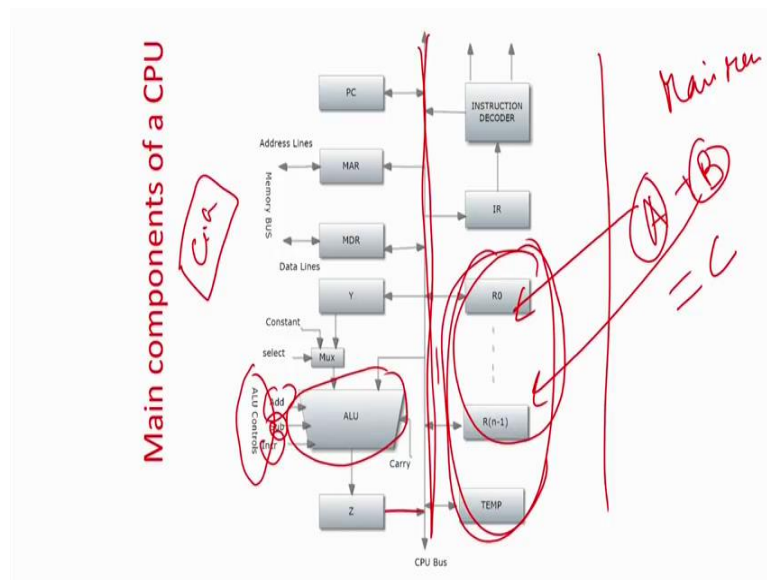


(Refer Slide Time: 25:13)



So, this is the basic set of elements which is in a CPU and their functions. So, till now I am telling you so much stories about different components of a CPU, registers, ALU and so many things, but without looking at a picture it seem just like a story without a movie, just like a I am telling you a story without showing the photographs. So, this is actually a picture of a broad picture of a main components of a central processing unit. So, how it looks? So, as I told you there is something called the arithmetic and logic unit; so this part is responsible for performing all type of a arithmetic and logic operations, but as I as I told you lot of controls are required because we have an adder here, subtractor here, multiplier here and what not.

So, you see like there are some control signals add, subtract, increment. So, this control values will come from a controller when I have to do what. Then as I told you what there are many other components will come out slowly.

So, this is the register bank that is $R0$ to Rn let n be something we don't require right now in this discussion. So, the registers there are n registers over here that is whenever you want to take some as I told you $A + B = C$. So, this A value will be say loaded to register $R0$, B may be register this one there main memory which is external to the CPU and the registers are nothing but same memory location you can assume, but at a inside the processor. So, that will has to be loaded here because the processor generally the central processing unit will look at the internal of it and directly not talk with the main memory.

So, this is the register bank this is the CPU controller is here I am not drawing it. So, this is the controller, it will send all the control signals when to do what and in fact you can see all the connections of the interconnecting buses. So, this is a main where it is connecting the ALU to all the registers to other registers as well, as was as I was saying there are two type of registers generalize general purpose register and specialized registers, R_0 to $R(n - 1)$ mainly which I was talking about you load the data from the memory etcetera are general purpose registers.

Which is open to the programmers to use right and there are some specialised instruction some special registers like *IR* that is an instruction register, program count we will come to it memory address register, memory data register, they are specialized registers you have to access, but you cannot directly use as your choice like the general purpose registers, they have a special meaning and they are used for a special purpose.

So, the registers you can see are general purpose, special purpose you have an ALU and you have a controller to generate the signals. So, this is, this is the quite familiar diagram if you look at computer architecture any standard book or any standard lecture will tell you about the basic design. So, whenever somebody asks you to draw CPU basically the ALU, the buses, the general purpose registers the special registers are actually drawn, to show basically different components and basic what are the jobs of each of the block.

(Refer Slide Time: 28:02)

Storage Elements

- Processor registers are generally divided into two categories:
 - General Purpose Registers (GPR) and Special purpose Registers.
- Von Neumann Stored Programmed principle—data and program in main memory.
- General purpose registers** are used to keep the data needed for an operation and can be used to store the user data.
 - During processing, data are brought to the processor and kept in a register. Processed data is updated in a register.

Special Purpose Registers

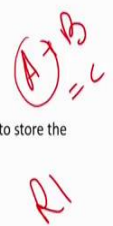
During program execution, we need to fetch instructions and data from the main memory.

Specify the memory location of the data and instruction. To provide the memory address, a special register is used which is known as MAR (**Memory Address Register**).

MBR/MDR (**Memory Buffer/Data Register**) is used for data transfer.

Execution of a program is carried out by executing the instructions in sequence. After fetching one instruction, we have to keep ready the address of the next instruction. This information is kept in a special purpose register known as **Program Counter (PC)**.

After fetching the instruction, we keep the instruction in a special purpose register called **Instruction Register (IR)**.



So, as I told you now we will start looking at the storage elements, now we will elaborate on each of the components 4 components we have seen. So, first let us start with the ALU. So, as

I told you ALU basically has lots of adders, multiplier, subtractors internally. So, mainly if you have the concept of digital design and which was also discussed in the basic module on which were the prerequisite and which are started by professor Deka that you have learned or you have revised basically some of the concept of digital design. So, ALU design mainly the logic and arithmetic units mainly consist of adder, multiplier, subtractors which are from the digital design principles.

Now, let us look at the registers that is your memory or the storage elements in the CPU perspective. So, as I told you, you have seen some of the registers like R0 to R1 where you generally keep your daily to daily to use variables like A, B, C, D. So, they are general purpose register, along with that you observe and remember the names say something called IR that is the instruction register, say something called the program counter say something called the MAR memory address register and MDR memory data register. So, they are used for some specialized functions, so you cannot directly access them they are used for some specific job they have to be accessed in an indirect manner and in fact you never require to use them while executing any code.

So, let us now go about it. So, there are 2 type of storage elements general purpose and special purpose. So, as I told you we are using a Von Neumann architecture. So, both data and program are in the memory. So, memory concept has to be very important for understanding a CPU operation and our case as you have dealing with the central processing unit, our data or our memory or our program everything is stored in a memory and internal to that one is a register.

So, as I told you general purpose registers are used to keep data needed for operation and our user data basically. So, generally data are brought from the memory to the processor kept in a register they are accessed, updated by the CPU or by the arithmetic logic unit and then again they are written back to the memory which is the main memory we are talking about.

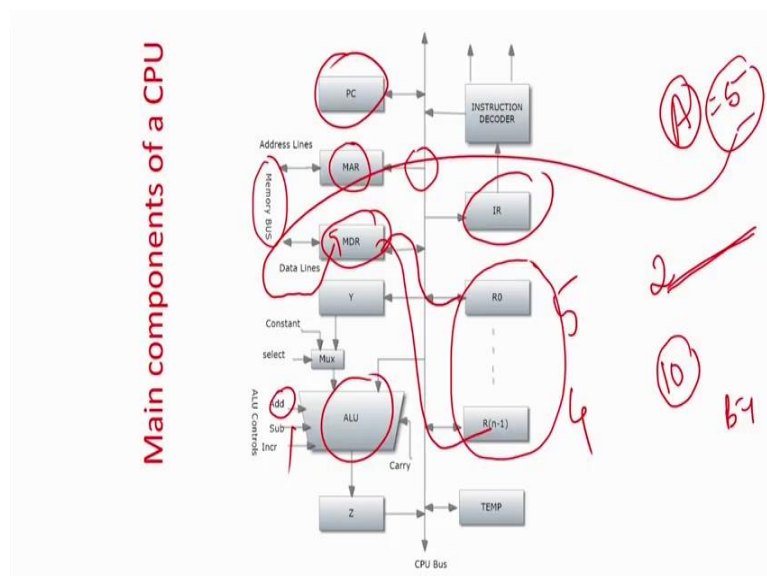
But that is simple, but generally there is something called as I told you lots of special purpose register. So, we will see one by one, so there is something called main memory address register. So, there is a memory, as I told you if we are doing $A + B = C$. So, A is nothing but a memory location. So, if you have done assembly level programming. So, basically we light load say R1 register 1 from memory location A. So, you have to fetch the value of A to register R1, then you need to know what is the value of memory location A.

So, now I have to load the value of memory location A to the memory then the memory will fetch the data from memory location A and dump it in R1. So, somewhere I should specify where is the address. So, that is the special purpose address register which is called the memory address register that is memory address register. Though your arithmetic and logic unit will generate or sometimes the instruction decoder will generate the address from where the value of the memory has to be fetched. So, this actually is the memory address register.

So, whenever you are saying $A = B + C$ or whatever the memory location address is generated in the or kept in the memory address register which actually goes to the memory bus. We will come what is the memory bus, basically the address is stored over the memory address register this one will talk to the main memory, bring the data from that address to memory buffer register and then write to the temporary register. So, memory address register you specify where is the address of the data that has to be fetched.

Next is memory buffer or memory data register, now you have given the address the memory has read the value from the address and you have to dump the data, then where it is dumped? Basically again a special purpose register called memory data register. So, say memory location A has been read A was having the value of 5.

(Refer Slide Time: 31:53)



Let us assume so, now, the memory will get the value of A memory location A, let it be 5 and it will where it will give it to the processor, how it will give it to the CPU. So, it will actually dump the value of A in something called a memory data register. Memory data register will

temporarily hold the value of 5 then via the system bus it will be give you to *R1* or any general purpose register. So, this *MDR* is a memory data register which is special purpose register to store the value of the temporarily the memory location that has been read.

Another important general special purpose register is the program counter that was you have written in this point a program basically execute in steps. From memory location all the instructions are also stored in the memory as well as data is also stored in the memory because it's a Von Neumann architecture. So, you first take the value of memory location say 1, execute it memory location 2, instruction 3, 4.

So, whatever are the instruction in memory location 1, then 2, then 3 you keep on sequentially executing it and how do you know what is the next instruction to be executed the value or a part basically if you look at it a part of the memory is you draw it like this, a part of the memory is data and a part of the memory is program.

(Refer Slide Time: 33:02)

Storage Elements

- Processor registers are generally divided into two categories:
 - General Purpose Registers (GPR) and Special purpose Registers.
- Von Neumann Stored Programmed principle—data and program in main memory.
- **General purpose registers** are used to keep the data needed for an operation and can be used to store the user data.
 - During processing, data are brought to the processor and kept in a register. Processed data is updated in a register

Special Purpose Registers

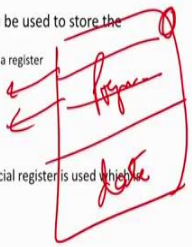
During program execution, we need to fetch instructions and data from the main memory.

Specify the memory location of the data and instruction. To provide the memory address, a special register is used which is known as MAR (**Memory Address Register**).

MBR/MDR (**Memory Buffer/Data Register**) is used for data transfer.

Execution of a program is carried out by executing the instructions in sequence. After fetching one instruction, we have to keep ready the address of the next instruction. This information is kept in a special purpose register known as **Program Counter (PC)**.

After fetching the instruction, we keep the instruction in a special purpose register called **Instruction Register (IR)**.



So, first instruction, the first location it is executed second instruction is executed and so forth.

But then who knows that what is the address location of this instruction, actually that is a special memory special register which is called the program counter which is *PC*, it will remember. So, it will first start with the memory location one, then that instruction is executed *PC* is automatically incremented.

Now, it is 2, the second memory location instruction is brought and executed and where it is brought it is brought to a special instruction called special register called instruction register, *IR* is an instruction register. So, whenever you execute an instruction the value of *PC* is the address of the instruction that has to be executed. So, *PC* will give the value to memory address register, memory address register will ask the memory to give the data it will give the data in the memory data register.

If it is an instruction the instruction register will take the instruction decode it and execute it, but the *PC* is always auto incremented, first instruction second instruction third instruction. So, it keeps remembering and when a jump instruction comes then the *PC* will not incremented by 1, it will now point to the new instruction location which is to the jump see for example, these after second memory location instruction, you have to execute the instruction which is the memory location number 10 then after executing the second instruction *PC* will not be 3, it's a jump. So, the *PC* will have the value of now 10.

So, basically program counter is also specialised register which holds the value of the next instruction to be executed, as well as instruction register is another specialised register which is actually holds the instruction and for the decoder to decode what the instruction has to do.

So, as I told you apart from general purpose register, there are lot of specialised registers which have some special job to do and which is not accessible to the user for programming and many others are there. We are just giving a broad overview that what are the storage elements, registers 2 types general purpose and specialised, let us go ahead.

(Refer Slide Time: 35:17)

Processing Units of a processor

- Every processor has one or more processing elements.
 - The most important is ALU (Arithmetic and Logic Unit).
 - Responsible to carry out any arithmetic and logic operations
- Any operation in hardware, performs better.
 - How many operations that will be pushed into the ALU is a design decision issue.
 - Limited number of operations in the ALU, implies lower capability of the processor.
 - To enhance the capability of the processor, some of the instructions are implemented with the help of programs that are known as micro routines.
- ALU which has addition and subtraction operations. But multiplication is not included.
 - A simple micro routine which performs the multiplication by repeated addition method.
 - The processor that is equipped with the multiplication block is faster but has larger area

Handwritten notes: "MUX" circled in red, "5x2" written in red, and a box with an "x" drawn in red.

So, ALU as I was say is generally responsible for computation. So, there is one very important thing, as I told you ALU should have a arithmetic and logic unit that is mandatory which is carrying out the logical operations as well as arithmetic operations. But there is very one important thing we have to understand there is a trade-off, I would like my ALU to be extremely super fast I would like to have adder, multiplier, subtractor, graphics processing as well as may be what not I want a oracle ALU you give an instruction immediately it is executed you can do that, but it will actually make the size of the ALU extremely large and very expensive for the cost of the processor.

So, here we have to make a decision that how many operations will be pushed in the ALU is a decision issue that whether you will your ALU will do only add and subtract or it can also do add, subtract, multiply and divide because you know as the multiplication can be implemented by multiple number of additions. Similarly division can also be implemented by combination of addition and subtraction and even subtraction can be implemented in a two's complement arithmetic by only addition.

So, now it depends on how much hardware you give it to the processor. So, if your processor has a multiplier, hardware multiplier then you give 2 numbers immediately it will be multiplied and you are going to get the answer. But if I say that I don't have a multiplier unit, I only have an adder and subtractor then is your processing unit not able to multiply no, processing unit can do anything that is possible by algorithm.

So, that is again a theoretical perspective I don't want to go, but if your processor has an adder, subtractor and some registers then it can be shown that any algorithm or any problem that can be solved by an algorithm can be executed by the simple processor, but anyway that is towards the theoretical angle.

So, I don't have a multiplier, but still my processing unit can do a multiplication that is you have to write a small routine which is actually called micro routine, a small routine will be there for the time being you can understand the small program which will say it is 2 into 5. So, what it will do? It will add 2, 5 times. So, but now your processor or your arithmetic logic unit is able to do 5 into 2 but it will take more time because you are going to add 5 two times or 2 five times.

So, it's a code will be execute to implement that, but if I have a dedicated multiplier unit any type of multiplier you can think about array multiplier or whatever nature, then you need not execute any microcode. You give 5 and 2 immediately it will multiply and it will give the answer, but in that case the area of the microprocessor of sorry, the area of the processing unit or arithmetic and logic unit will be larger. So, there is always a trade-off that what to put in the software and what to put in the hardware.

Software means you have to write micro routines like as I told you multiplication can be done by addition or you want to put a dedicated hardware to execute it in that case the ALU will be more complex it will be faster, but it will be more expensive too. So, therefore, there is lot of trade off in between this. So, there is actually something called micro routine that you will have the instruction MUL, multiply A cross B, but actually multiply will be internally represented by a micro routine which will break it up into smaller instructions and actually it is implementing multiplication by multiple additions.

So, broad idea of ALU is very simple, you design it for addition, multiplication, subtraction using digital design fundamentals, but how much you put and how much you put in a software like a micro routine depends on the decision of design. So, in the next few modules and lectures we will see such type of designs, in fact the next module will handle those things that if you have to design a processor and you do not have a multiply unit then how to write a micro routine which will execute it.

So, as again very important thing, what is the number size you can represent in a processor? Of course, it's not infinite we always go by something called 32 bit machine, 64 bit machine, we have heard the term.

(Refer Slide Time: 39:01)

The slide is titled "Processing Units of a processor" in red. It contains a bulleted list with handwritten annotations in red ink. The first bullet point states: "Processor cannot take care of the complete range of any number systems; it depends on the size of input and output." The second bullet point states: "For example, if the size of input is 8 bits, then it can handle numbers up to 255 = $2^8 - 1$ (for unsigned integer). For signed integer, the range is half." Handwritten red circles highlight the words "complete range", "size of input", "8 bits", "255", and " $2^8 - 1$ ". To the right of the text, there are handwritten red numbers "8" and "2" stacked vertically.

Processing Units of a processor

- Processor cannot take care of the complete range of any number systems; it depends on the size of input and output.
- For example, if the size of input is 8 bits, then it can handle numbers up to 255 = $2^8 - 1$ (for unsigned integer). For signed integer, the range is half.

Exactly it depends on the range of numbers, it actually depends on the input and output size of the processor or the arithmetic logic unit. Say for example, if the input size is 8 bits then the number it can handle is 255 that is 2^8 for unsigned number, if you are using a signed number then one bit will be reserved for sign or other way. So, the range is high, but again you can use as I told you that you can use multiplication, in multiple type of additions to do it similarly you can represent the very large number also, but not directly.

If your, if you have a input of 8 bits you have to truncate it, you have to go for the first 8 bit LSB, then next LS set of 8 bit LSB and so on. But there is it will going on an iterative manner, initially you cannot have everything and directly do. 8 bit input means 2^{255} unsigned and half of that in signed can be directly used for computation, larger than that you have go in iteration first LSB 8, then next, then next and so forth.

So, in fact that is what is I am going to say is that processor can do a lot of things, in fact everything which is possible by a algorithm, but what is directly possible is what is the hardware implementing it, like 8 bit processor like directly two 8 bit numbers can be multiplied or whatever, but if it is the 16, 16 bit numbers then you have to go it in phase first 8 LSB and next and so forth.

So, either you go in a iteration and take a more longer time or you have a more expensive ALU, with a more hardware in it as a design choice. Then as I told you we are coming to the third component which is called the interconnections and controller design is more complex, I will just give you the idea, but not going to elaboration because controller signals are assume that it will come. So, in fact let us not go interconnections directly.

So, we can go for first and very basic idea of what is basically the control signals, say for example, and why do you not require elaborate at this position is something like that. Say I want to add $A + B$, the value of A has to be first j first the program counter PC has to tell that instruction has to be executed, then the value of A has the A has to be put to the memory address register, the address a memory address register will get the value to the address memory, memory will get the value of variable A it will dump into the memory data register say and it will go to a general purpose register from R0 to R1 similarly you have to repeat it for B. So, 1 one register will have the value of 5 that is A and may be you can assume that B is having the value of 4. So, another register will have it.

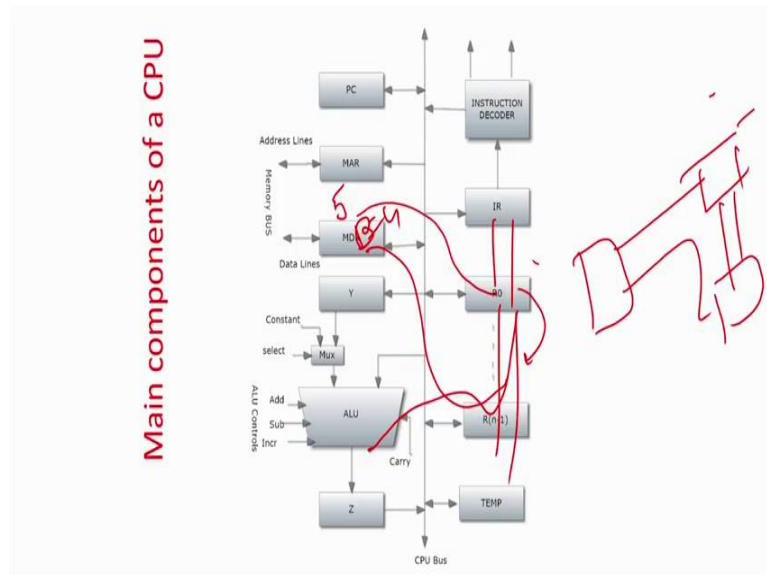
Now, you have to add it. So, $A + 5 + 4$ will happen and this it will be stored in another temporary register may be R3 and again the value of R3 will be written to the memory data register and now the memory address register will have the value of C and the from the data. Now, from the memory data register which is now 9 will be written back to the value of C, in fact memory data register both holds the value when it is coming from the memory as well as when it is going from the central processing unit to the memory, it's a buffer.

So, lot of control signals are required over here like for example, when the memory data register is writing the value of A to memory register 0, a connection has to made like this. When the value of B is being read from the memory data register a connection has made like this. So, lot of multiplexing signals has to be there, when you have add it you have to make this line 1 and all others are 0 because the ALU is responsible for addition, subtraction, multiplication everything. So, I assume that there is a block which is a controller, which will generates signals on appropriate time, how to design the controller is a big job that we are going to see later.

So, therefore, you assume that there is an oracle sitting which will generate all the signals as and when required. So, that is the basically the idea of a controller. So, I at this position point it's not wise to go more depth on the design of controller because it will involve 4 to 5 units totally dedicated on that. Now, basically we are going to go for the interconnections as I told

you I cannot put one very simple way you can tell that, I can have a matrix side of an arrangement which is the very best like this one I can because you see as I told you. So, there is a single memory data register.

(Refer Slide Time: 43:15)



It will get the value of A first then will get the value of B that is equal to 4, then the value of B will be fed to R0, the value of B will be fed to R some value. So, what I can do? I can say that this is memory data register and these are all the registers.

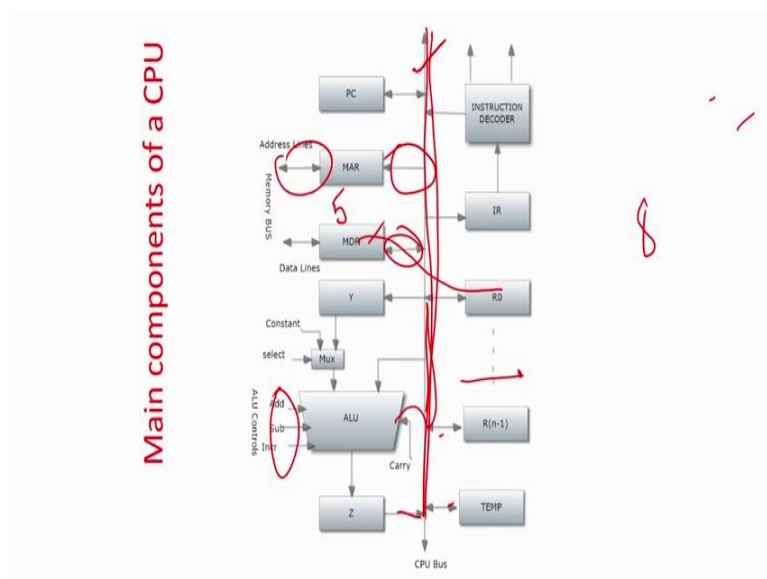
So, I can make a connection like this, every point to every point because value of R0 may also be loaded to R3. So, in between these also lot of interconnections. So, everything may be interconnected, ALU may be able to write it directly.

So, I can make a pure matrix connection, that any point of time any 2 guys can be connected R1 can be connected to R5, R1 can be connected to ALU, R1 can be connected to memory data register. In fact there we will have a very nice arrangement because everybody can talk to everybody in at any point of time, but you have to understand that it will make a very curve type or a very chaotic type of type of a interconnection network.

In fact then you will land in to a situation that nothing can be debugged or nothing can be gained out of it because your interconnecting network or it will be a matrix interconnecting matrix will be more expensive to implement than the whole CPU itself. So, that is not going to be a very wise decision.

So, what people do, people implement a bus structure, that is a single wire is there or single set of wires are there for different purpose and whenever somebody has to use a wire they have to use it on a mutual exclusive basis that is; obviously, simultaneous access cannot be there. But the cost is very reasonable because say whenever say for example, the MDR is supplying data to $R0$ at the same time ALU may not be able to use this bus to write to another register it has to wait till the memory data register has transferred the value to $R0$ then only the ALU can do it.

(Refer Slide Time: 44:43)



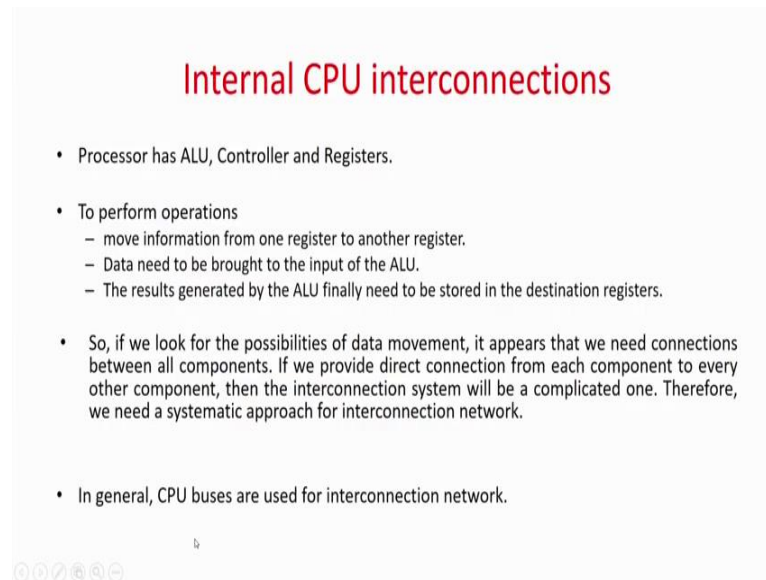
So, it's a, the whole organization is such a fashion that they are all a on a mutual exclusive basis, this wire is centralized. So, either any one of them can use it at 1 time. So, other guys has to wait for it. So, it's a time taken is slightly higher, but it's a more feasible implementation and cheaper implementation and as we are talking about not a very advanced processor. So, more advanced processor has replicate the number of buses, there is a 2 bus organization 3 bus organization and so forth.

So, generally we do not call it wire, we will be talking in terms of computer architecture will call it bus because we know that when a talking about 8 bit input. So, 8 lines are coming out. So, this is not a single line actually it is an 8 line, 8 bit bus. So, we make a sign like this and it implies that it has an 8 bit bus, 8 internal lines are going on it.

So, you should use the term bus and not to wire, but again the most important concept here is that they are all in a shared basis. So, when this if I will call it as a data bus. So, only a pair of

registers will transfer the data through a ALU then it will be first transferred from the register to the may be ALU from ALU to register, then when one transfer is going on the other all guys has to wait.

(Refer Slide Time: 46:07)



Internal CPU interconnections

- Processor has ALU, Controller and Registers.
- To perform operations
 - move information from one register to another register.
 - Data need to be brought to the input of the ALU.
 - The results generated by the ALU finally need to be stored in the destination registers.
- So, if we look for the possibilities of data movement, it appears that we need connections between all components. If we provide direct connection from each component to every other component, then the interconnection system will be a complicated one. Therefore, we need a systematic approach for interconnection network.
- In general, CPU buses are used for interconnection network.

So, that is the basic concept of a single bus processor that is we generally call it CPU interconnection. So, a processor is ALU, controller and registers and they have to interconnect among them. So, what are the interconnection operations what it required? Data need to be brought from the input to the ALU, very important that you have to get the data from the memory to the register and then to the ALU then 1 register may have to send data to the other. Result generated by the ALU permanently has to be stored to the register and then has to be return to the memory so lot of data transfer will happen.

So, if you have components as I told if you have interconnection with all components very nice, but in fact large number of number some arrangement and more cost. So, the interconnection cost will be higher than the cost of the components itself.

So, therefore, we use a bus architecture, bus means number of lines will be more than 1, it is a 8 bit address bus then it will be 8 bit data bus. So, you call the use bus so in fact that is shared. So, you can see bus, this is typically definition of a bus, bus is a communication pathway between 2 or more devices.

(Refer Slide Time: 47:05)

Internal CPU Bus

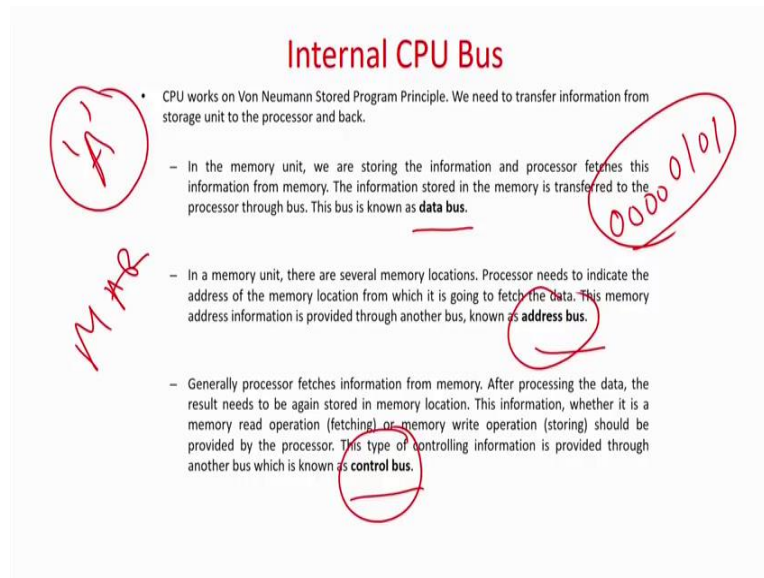
- A bus is a communication pathway connecting two or more devices. Typically, a bus consists of multiple communication pathways, or lines. Each line is capable of transmitting signals representing binary 1 and binary 0. For example, an 8-bit unit of data can be transmitted over a bus consisting of 8 lines.
- Consider a single internal CPU bus organization.
- In this case, all the components are connected through this internal CPU bus.
- For example, consider two general purpose register $R1$ and $R2$. We need to transfer the information from $R1$ to $R2$. In that case, first the data from the register $R1$ is placed into the internal CPU bus, and then the data is transferred from internal data bus to register $R2$. To enable register $R1$ and register $R2$, the control unit generates the appropriate signals.
- With single internal data bus, at any point of time only one data transfer is possible. So, for better performance, processors are designed with multiple internal buses also.

Typically bus consists of multiple communication path way or lines for example, it's an 8 bit bus means it can directly transmit 8 bit data, generally we are talking about or throughout the 2 modules we will be talking of a single bus architecture because that was a long time for long large number of a design of processing elements or processors.

Single bus was standard and everything has to be connected to the single bus architecture, then I told you it is a multiplexing arrangement because you have to wait when the other 2 guys are transferring for example, if 2 general purpose register $R1$ and $R2$ need to transform data to from $R1$ to $R2$. In this case first the data from register $R1$ is placed to the internal CPU bus and then the data is transferred from the internal CPU bus to the register 2, then you have some control signals to do that, but when this transfer has been going on other guys cannot do anything they have to wait. So, $R1$ will give the data to some register, the control signals will be appropriately set and then through the data bus it will go to $R2$ other guys are just sitting idle they cannot do any operation with the bus.

So, with a single bus at any point of time only 1 data transfer is possible. So, with better performance you can increase more number of bus, the best one I have already told you fully matrix operations. So, more have been any type of hardware more you put on hardware more benefit you are going to get.

(Refer Slide Time: 48:21)



Then 3 very important things we have to tell you, there is something in a bus when you talk about single bus or multiple bus, a bus has basically 3 sub parts data bus, address bus and control bus. So, what is a data bus? As I told you, you want to transfer some very varying of a variable from register $R1$ to $R2$. So, a bus which will do this operation is called the data bus if it will take the number say 5 that is may be 10 sorry, 0101 it will take it from register 1 and it will transmit the data to register B, then a bus which will doing it in fact is a 8 bit bus. So, the 4 LSB's will be 0 so the bus which will doing is actually called the data bus. Now, as I told you sometimes you have to get the value of some variable from the memory, like add A1, A2 or A, B. So, the value of A that is the memory location has to be sent to the register sorry memory by the memory data memory address register MAR.

So, the value of A that is the sorry the value of A will be fetched, but what is the memory location, the memory location of A that is the address of A has to be sent to memory address register now which bus will send it that is actually called the address bus. So, address bus will always consists of the address of the register. So, address of the memory locations, In fact data bus and address bus are 2 separate buses.

So, data bus and address bus are not multiplexed, some if some data transfer is going in the data bus the address bus can freely send some new data which has to be fetched from the main memory the address can be sent by that and secondly, as I told you the third type of thing is called the control bus which is generating the signal. As I told you when to add some signals

has to be changed to the ALU, *R1* has to be connected to *R2* simultaneously all other guys cannot use the address bus address sorry data bus. So, some internal signalling has to be done. So, a separate set of wires will actually do it for you that is the control bus. So, these 3 buses will totally make a system bus of the internal CPU bus. So, there is only one internal CPU bus in single bus organisation, but the address bus, data bus and control bus are segregated and they can be used simultaneously there is no multiplexor, multiplexing in between them.

Like if I show you this for the last time so I can tell you that this is the data bus, all this connections are data buses because they are transferring the data these are the control bus, add subtract registers are control signals which is generated like and this bus memory sorry memory address register. So, this will be the value which will be generated which is the address bus. So, it will send the value of the memory location which has to be fetched.

So, overall we are at this first lecture on this unit we are telling you the basic definition what is the basic internal meanings and we are keeping many things in very abstract manner. So, basically we are giving you an idea of what is a data bus, what is a address bus, what is a control bus, but lot of intricacies are there, like in fact how the address bus can be multiplexed and then the some address bus is doing the work, how we can utilise the data bus through other operation. It will come in the future units.

Ok so as I already told you 3 buses and again the multiple buses can be used to enhance the performance of the procedure and performance of the program. So, with this we come to the end of this unit. So, this unit basically talked about the different units or the different parts of a central processing unit, the interconnection, the registers, the arithmetic logic unit and the control, in a very broad manner which will give you an idea what is the internally it looks like.

(Refer Slide Time: 51:57)

Questions and Objectives

<ul style="list-style-type: none">• Q1: What are the main components of a Central Processing Unit (CPU) and their function. Explain roles that are performed by processor registers.• Q2: Explain the basic construction of processing elements (e.g., ALU) of a processor.• Q3: What are the issues related to internal CPU interconnections.• Q4: What is a bus and how the other devices are connected to the processor through bus.	<ul style="list-style-type: none">• Knowledge: Describe:--Describe the use of storage elements in the processor. Indicate the uses of General Purpose Registers and Special Purpose Registers.• Application: Demonstrate:--Demonstrate the use of Execution Unit.• Comprehension: Indicate:--Indicate the need of control unit in a processor.• Comprehension: Discuss:--Discuss about the interconnection of different components of processor.• Comprehension: Describe:--Describe the interconnection of Processor with the main memory and External I/O devices through system bus.
--	--

Now, as again as we have telling about pedagogical aspect. So, we have told that there are certain objectives like describe the use of processor, demonstrate the use of execution unit and so forth. So, these are the objectives we have claimed that a person should be able to do when he has completed this unit well.

So, again we put some 4 questions over here, like what are the main components of a central processing unit and their function, explain the role that are performed by the processor registers. So, once you have done the lecture I think in the very few first few after first few slides I have told you about the answers of that, again the second question explain the basic construction of a processing element ALU of a processor. As I told you it consists of digital design elements like adder, multiplier, subtractor then you have to decide whether you want to do addition, sorry multiplication by addition or you want have a hardware unit. So, we have broadly discussed about what is the basic construction of an ALU not the details, but the basic how it what, what are the issues and how we think about the design based on trade of between what power it has and what is the area and what is the cost. Similarly what are the issues related to CPU interconnections. Like as I told you single bus so you have to wait, if you are using a matrix kind of a connection it can be again make a very high end arrangement, but what are the issues means how it is interconnected because there is a multiplexed arrangement.

So, simultaneously their transfer cannot be done so that already has been discussed. What is a bus and how other designs are connected to the bus that is also very simple, we have done and

illustrated it in a pictorial fashion of how the ALU, the IR, the general purpose registers are connected. So, these are the 4 questions which you are expected to answer after going through the module and we have already discussed it, but then the you are also you appreciate that if you are say for example, if you are able to, if you are able to just explain the basic construction of a arithmetic and logic unit then you are able to demonstrate or satisfy this objective saying that demonstrate the usage of an execution unit. So, if you are able to explain the basic construction of an ALU you are actually successfully meeting this objective.

Similarly, if you are say that what are the main components of a central function unit and what are the roles. So, if you are able to tell what are the main functions of the units you will be able to indicate the need of a control unit, similarly you will able to discuss about the interconnections that is if you can tell what are the main components then and what are the functionalities then you can easily say that why controller is needed and how interconnections are there.

So, in fact if you look at the questions and if you look at the objectives answering all the questions will actually meet all the objectives and again if you can appreciate whether knowledge and mainly comprehension based objectives. So, the questions are also of that nature, that you have describe you have discuss there is no more no design in this and in fact this module is expected to give a very brief an abstract overview of a central processing unit. So, we mostly, we now in the unit we will now start dealing with how a memory external memory is organized. So, in that case will go slightly in depth of that aspect then we will try to go how a program is executed.

So, in that case the objectives start becoming more of a design and more of analysis and application based. So, the now it is knowledge based. So, it will be more on simulation based objectives will be mostly on design based. So, that once you do all those units you will able to design something ok. So, with that we come to the end of the first unit of this module and next we will start looking at the memory units of in an overview level and then we slowly go to the in depth of how the instructions are executed in a processor, which is a main role of this module.

Thank you.