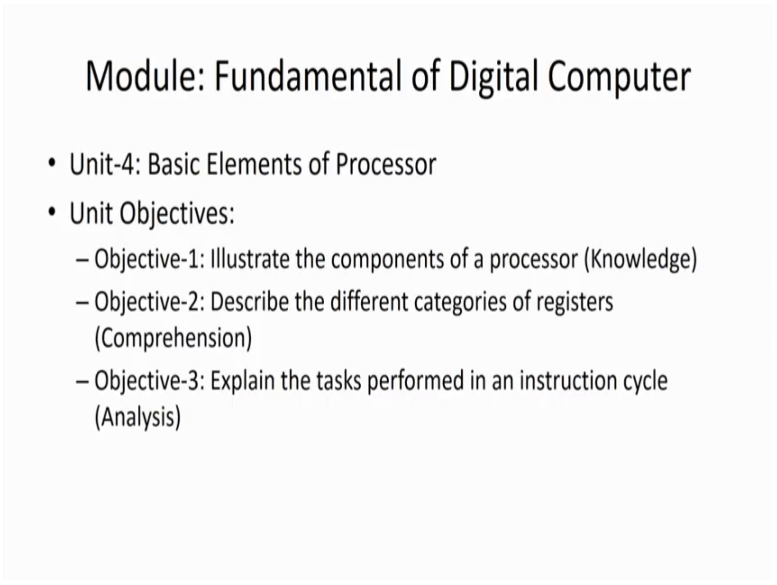
Computer Organization and Architecture: A Pedagogical Aspect

Prof.Jatindra Kr.Deka
Dr.Santosh Biswas
Dr.Arnab Sarkar
Department of Computer Science and Engineering
Indian Institute of Technology, Guwahati

Fundamentals of Digital Computer
Lecture - 04
Basic Elements of Processor

(Refer Slide Time: 00:47)



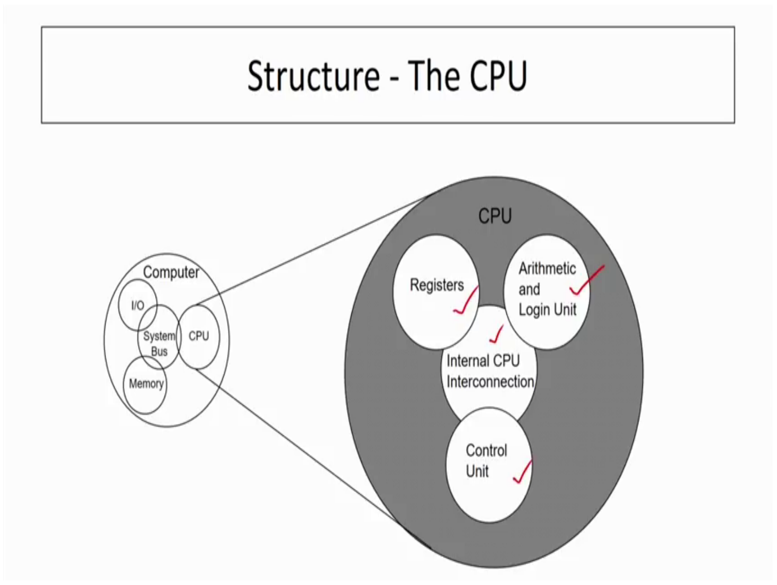
Module: Fundamental of Digital Computer

- Unit-4: Basic Elements of Processor
- Unit Objectives:
  - Objective-1: Illustrate the components of a processor (Knowledge)
  - Objective-2: Describe the different categories of registers (Comprehension)
  - Objective-3: Explain the tasks performed in an instruction cycle (Analysis)

Hello everybody welcome back to the online course on computer organization and architecture. We are in a module fundamentals of digital computer. And now we are in unit 4, the unit 4 is related to basic elements of processor. So, as usual now first we are going to talk about the objective of this particular module. We have identified three objectives for this particular unit on basic elements of processor. The objective -1 - illustrate the components of a processor. So, we are going to see what are the elements we have in a processor and the level is your in knowledge level. So, in this particular unit we are going to discuss everything in knowledge level only. So, we are going to just give idea about it, but in subsequent module these things can be covered in higher level maybe in the design level and synthesis level.

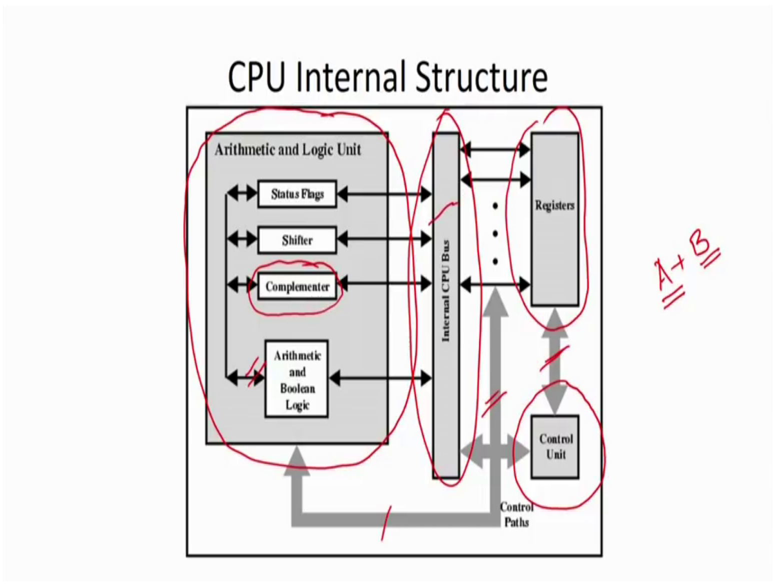
Objective-2 - describe the different categories of registers. This is in comprehension level, what is register we are going to see and what are the different categories that we have we are going to look into it. Objective-3 - explain the tasks perform in an instruction cycle. So, already in our earlier unit, I have talked about the instruction cycle, how to exit an instruction. Now, we are going to see what are the different tasks or sub tasks that we are going to perform in an instruction cycle. So, this is in analysis level that means once you see an example of an instruction then at least you will be able to analyze what are the tasks that will be performed by this particular instruction, and finally, what is the effect that we are going to have after exiting this particular instruction.

(Refer Slide Time: 02:10)


Structure - The CPU

So, we have seen that top level view of your computer structure. So, in this particular case, you are saying that we are having four basic element CPU, I O, memory and they are connected with system bus. So, this is the computer that we can see components of a computer. Now, in this particular unit, we are going to talk about the processor CPU central processor unit. Now, if you look at it then you can bubble up this particular bubble of CPU and we are going to see that basically it is having three major component one is your arithmetic and logic unit, then we are having the registers which are nothing but the storage unit and along with the control unit. Controlling unit having the responsibility to synchronize the operation inside the processor and all those components are connected through this internal CPU interconnection. Because sometimes or in many a time you transform information from one component to the other component like the transfer information from registers to the ALU or maybe the result from ALU to the registers. So, for that we need to have the interconnection, so we are having the internal CPU interconnection.

(Refer Slide Time: 03:21)



Now, this is the internal structure, it is very generic one, in most of the cases you are going to see this is the internal structure of a processor, but it may varies is from processor to processor, but in general we are going to have this component. So, just see what are basic components that we are having, we are talking about the registers, these registers are nothing but the storage element. So, here we can store our information. Another component we are having called ALU - arithmetic and logic unit. So, this is the main processing element inside the processor, here we can perform some operation like in ALU inside that we are having a component called which have a have arithmetic and logic operation.

So, some of the arithmetic operation you can perform like addition, subtraction, multiplication, division etcetera along with that we are having a provision to perform some operation Boolean operation like AND, OR, exclusive OR etcetera. So, these are the basic operation that we can perform along with that sometimes we have some other functionalities also one is known as your complementer basically it is going to complement a given number. Another one we are having a shifter, already have talked about the shift registers that means, sometimes we have to shift the contents of a register either it is your left shift or right shift. So, for that also you are having this particular shifter. And along with that we are having status flag will going to see or going to elaborate what is those particular status flag, but already have discussed some of the issues related to those particular status flag when we talked about the arithmetic operation inside the ALU. So, we are going to see how what is the effect of those particular component.

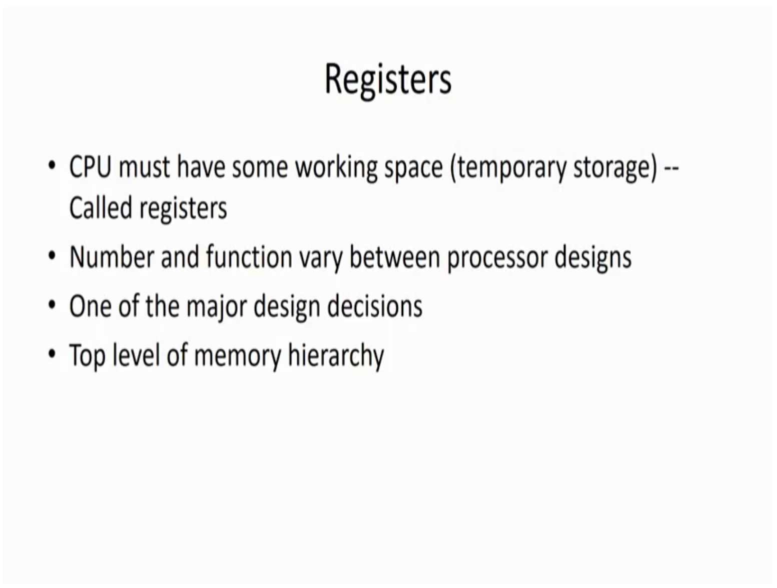
Along with that now say we are having registers and arithmetic and logic unit; now we are having a interconnection of this particular component. So, here we are having a interconnection network which is basically bus we will see what is bus now. So, basically it is a connecting where. So, from this registers we can place the information in this particular internal connecting where bus from that it will go to the ALU. ALU will perform the operation and result will be transferred to again through bus to the registers if we want to store the result.

Now, we have to now perform this operation in a systematic way. Just that if I want to add two numbers A plus B, then what we are going to do possible transfer A from this register to the input of the ALU then will transfer B from the register to the ALU then I am going to perform the addition operation. Whatever result we are getting after performing the addition operation, we are going to transfer it to the registers. So, for that we need appropriate control signals. So, this control unit is having the responsibility to generate this particular control signal. So, these are the control lines connected to different component.

Now, what is the control signal. So, first I am going to say the transfer the A to ALU. Now, we are going to select the register a inside that ALU we having several registers in register bank. So, controlling unit will generate appropriate signal to select this particular register A, then will transfer the information to the input of this particular ALU one of the input of this particular ALU. Then control unit is going to generate the next signal, so that it is going to select the register B. And after selecting a register B will transfer the information to the ALU again. Now, after getting both the information A and B inside ALU now what that responsibility of the control unit to generate the signal to trigger the additional operation inside this particular ALU, because there have been several operations inside ALU and all ready we have discussed it all ready explained how we are going to select a particular processing element.

So, control unit is going to generate the signals, so that it is going to trigger that particular addition module of this ALU and finally, it is going to get as the result. So, once they are getting the result, then what we are going to do we are going to transfer it to some of the registers. So, again control unit is going to generate the signals. So, the responsibility of the control unit to generate the appropriate control signal at the appropriate point of time, and these control signals are given through this particular control part or control bus. And through this data bus we are transferring the data from register to ALU or from ALU to registers. So, these are the internal structure of a processor; in most of the processor we are having those components, but the placement of the components may vary from design to design.

(Refer Slide Time: 08:11)



Registers

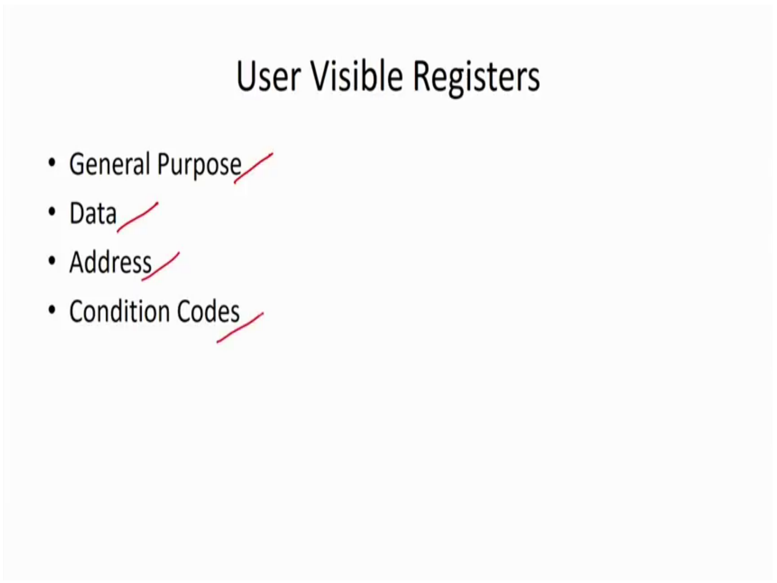
- CPU must have some working space (temporary storage) -- Called registers
- Number and function vary between processor designs
- One of the major design decisions
- Top level of memory hierarchy

Now what is the first component that we are going to talk about it is the register. What is register? These are nothing but some storage space. So, CPU must have some temporary working space and which are known as your registers. So, first we are going to take the information and put it into this particular registers. So, after that once we have the information inside that processor that now processor can carry out the work according to our requirement. So, the number of functions vary between processor design. So, the what are the number of process register that we may have inside a processor, what are a function of those particular register, it depends on design of the processor. So, it varies from design to design. So, this is a one of the major decision in processor design. And we have to see how many resistors we are going to keep and what are the purpose of this particular registers, and secondly, we can consider this as a top level of memory hierarchy.

What is the memory? These are storage element, so I can store information. So, this is top level we are saying that this is inside the processor itself processor is going to walk with the information available on these particular registers. Now, where from we are going to get that information into those particular registers if you see the working principle or Von Neumann stored program principle so; that means, you must have storage element, which is the memory modules, so that will be connected to the processor. So, this is the next level of storage in the memory hierarchy.

So, in a memory hierarchy, top level you are having the register where you can store all information, but where from we are going to get information through that particular memory or registers it is from the main memory. So, main memory is the next level. So, we can have an hierarchy of memory. And in this particular course, we are going to see what are the define level of memory that we have been a computer. So, just I am telling that it is a top level of memory hierarchy. So, registers are nothing but a temporary storage unit which are place inside the processor itself and processor is going to walk with the information available in the register.

(Refer Slide Time: 10:21)



User Visible Registers

- General Purpose
- Data
- Address
- Condition Codes

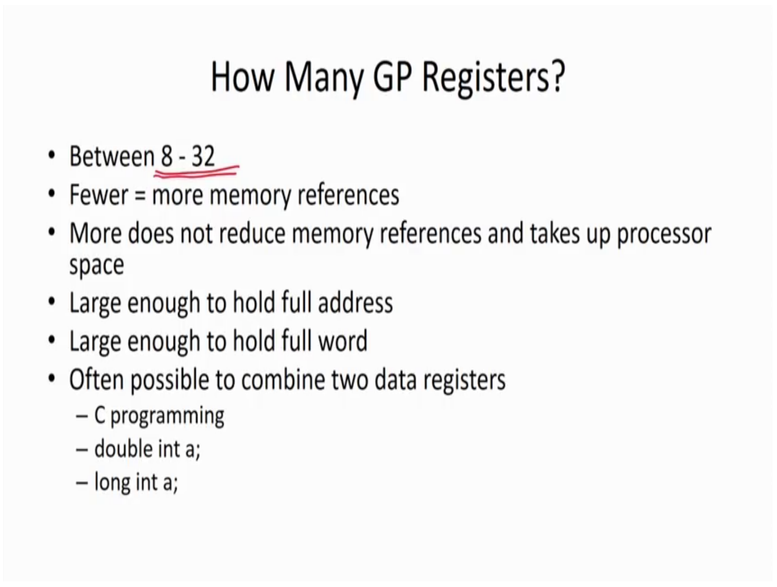
So, what are those register. So, registers can be categorized into different part, one is known as your user visible registers. Why you are saying this is user visible, because these are the architectural issues and we know that there are some register inside the processor, and while writing our program we can use those particular registers, so that is why we are saying this is the user visible registers. And these are this can be used by a programmer while writing his program. Maybe when I am going to talk about the writing a program we will see in what level basically I am mentioning about it. So, these are the registers and user or a programmer can use. Again in this user visible resisters can be categorized into four different categories one is a general purpose, data registers, address register and condition codes.

So, general purpose register, it can be used for any purposes you can keep or you can store an information or any kind of information inside that general purpose registers, but if you are going to say it is a data register then what will happen we can store only data that we are going to work on. So, data resister can keep data only. When we talk about the address resister basically this is going to keep the information about address. Now, what is an address? So, you all ready have mentioned that our computer works on Von Neumann stored program principle. So, for that we are having a storage in it which is known as your memory unit; now in the memory we are going to store our information.

Now, how to get this thing? So, for every memory location we must have to assign an unique address likewise say if I want to reach you at this I must know your address. So, if you are a student and you are staying in and hostel that is I must know the room number that you are staying, then I have to know in which hostel you are staying; of course, from outside world, I must have to know in which institute you are there. So, like that all those components is going to tell me your location and I can go to your location. Like that when we are going to take going to take an information from a memory we must know this particular address.

So, we are going to give some address to each and every location, and this address register can keep on the addresses of this memory location. Secondly, when you are talking about addresses several IO devices will be connected to the computer. So, to identify each and every IO devices, we must have to assign some addresses. So, those addresses can also be kept in our address register. And we are having one more registers called condition codes. So, we will elaborate it what does it means and what information we are going to store it condition code.

(Refer Slide Time: 13:04)



How Many GP Registers?

- Between 8 - 32
- Fewer = more memory references
- More does not reduce memory references and takes up processor space
- Large enough to hold full address
- Large enough to hold full word
- Often possible to combine two data registers
  - C programming
  - double int a;
  - long int a;

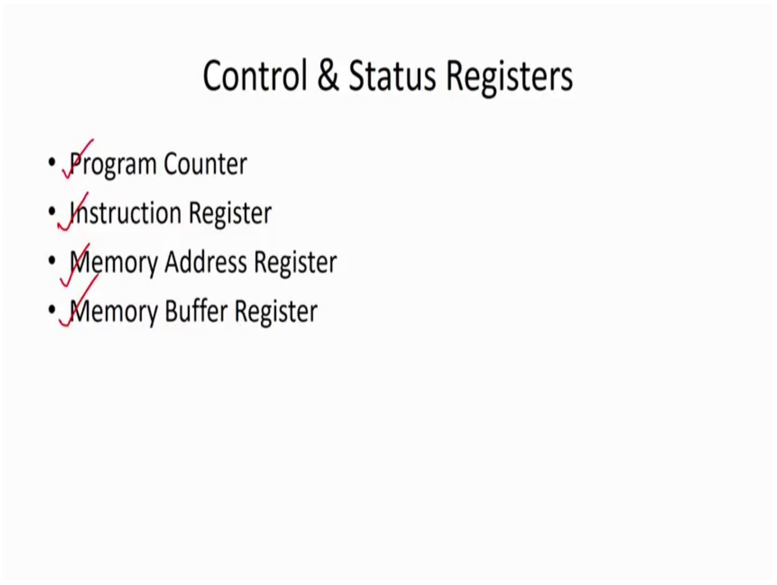
Now, one of the major decision while going to design a processor is the number of registers, how many general purpose register we are going to keep. This is a major design decision because the performance of a processor depends on this particular number. If we are having a pure registers then what will happen, there will be lot of memory offenses, you are going to keep one going to bring one information from memory working with that particular information. But if we are having a fewer number of registers in all you are keeping some information in all the registers then what will happen when we are going to bring another information to the processor, we cannot keep in any of the register because all are occupied. Then we have to transport some of the information from register to the memory to make a (Refer Time: 13:56) space and in that particular (Refer Time: 13:58) space, you can keep this particular new information.

So, if having a very less number of registers then what happen that memory referential be more; that means, lot of information has to be take from memory and in term lately we have to store into the memory. But again it does not mean that if we are going to increase that register they are going to put more number of register again it is not going to reduce the memory references drastically, it will reduce the memory references up to a certain extent. So, it is not like that we are going to walk in a very less number of register or very big number of registers. In typical ways is observed that in most of the processor we are having a register somewhere between 8 to 32, 32 is also in some higher side. So, it is somewhere near to 8 only or in general I can say that register number of register varies with an 8 to 32

Now, when we are talking about the registers, when we are going to talk about this storage element, temporary storage element instead of processor, we must have to store some information, and those information should be stored properly. So, for that we have to see you what will be the size of those particular registers. If we are talking about the address register then at least it should able to store the full address of the memory location or full address of the IO devices. So, resistors address registers has to be long enough or large enough to keep the complete information of a memory location or a device. And what will be the size of a data register, it should be long enough to or large enough to hold a complete word.

So, if we are going to talk about a processor say we are going to see that it is a 16 bit processor; that means, we can walk with 16 bit at a time so that means, the size of the register should be at least 16 bit. When we talk about a 32 bit processor then we are going to walk with a 32 bit data, so that size of the registers should be you 32 bits. Also it is possible to combine two data register together to work with more information. So, like that in C programming, you focused on the C languages you will find that sometimes you are going to use that double integer. So, integer is a data type we can have a double integers. So, in case of doubling integers, we are going to increase or runs. So, in some cases depending on the processor, it may happen that sometimes we have to combine two register together to keep a double integer inside the processor. So, this provision is also you have to provide while designing the processor.

(Refer Slide Time: 16:33)



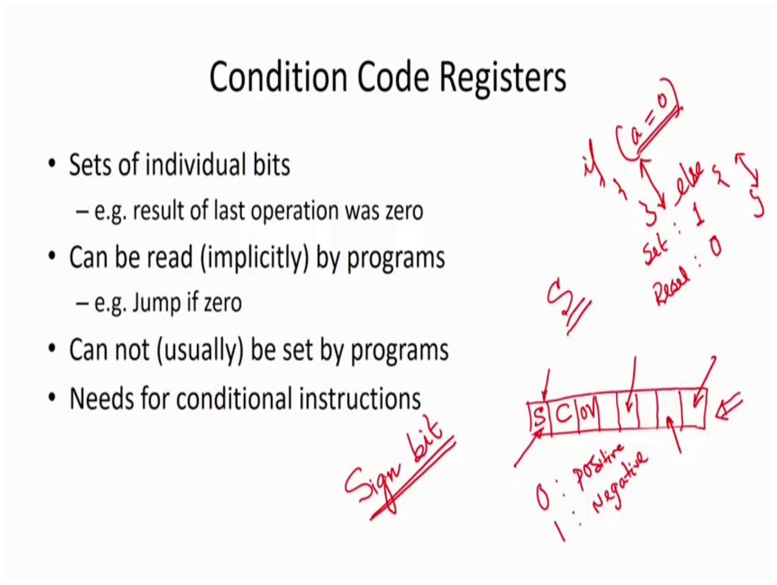
Control & Status Registers

- Program Counter
- Instruction Register
- Memory Address Register
- Memory Buffer Register

So, these are the general purpose register, where we are going to store information data and addresses and these are visible to the programmers. So, while we are going to write a program, we may use those particular register to store our information, but there are some registers or storage elements inside the processor which are known as a control register or status register, these registers are not visible to the programmer. Directly we cannot use those particular registers, but for the PRATAP operation of the computer or processor we have to have those particular registers. So, programmer cannot use those register, but to control our operation while designing we are going to use those particular registers.

So, some of those resistances is one is your program counter, second one is your instruction register, third one is your memory address register, fourth one is your memory buffer resister. So, I am citing four registers over here, and these are bare essential register inside the processor to controller operation inside the processor. So, in subsequent lectures, we are going to discuss in details about the use of those particular resisters thus here I am introducing that these are the four status registers or control register we have inside our processor.

(Refer Slide Time: 17:57)



Condition Code Registers

- Sets of individual bits
  - e.g. result of last operation was zero
- Can be read (implicitly) by programs
  - e.g. Jump if zero
- Can not (usually) be set by programs
- Needs for conditional instructions

if (a=0)
else
Set : 1
Reset : 0
S
C
ov
Sign bit
0 : positive
1 : negative

There is another one register called condition code register, this is an resistors, but here we are going to work with individual bit. So, say that if we are working with an 8 bit processor then what will happened the size of the registers are 8 bit. So, we can consider an 8 bit register where we can store 8 different information. So, in case of your condition code register, we can work with individual bit either we can set or reset those particular individual bit or we can use those particular individual bit. So, this is one of the issues. And now what we can store or what we are going to store over here. So, this bit will be safe or decide, all ready I have mentioned that set means we are storing information 1, and reset means we are resetting it to 0.

So, these are basically made up flip flop. So, flip flop can be set or reset. And when we are going to set and reset, it depends on some operation inside the processor. So, generally it cannot be set by the programmer, but some of the bits you may have which can be set by the programmer in general you can say that it is not shared by the programmer. But those bit will be set by the operation of a processor, we are performing some operation and on depending on the result of those operation, we are going to either set or reset those particular bits.

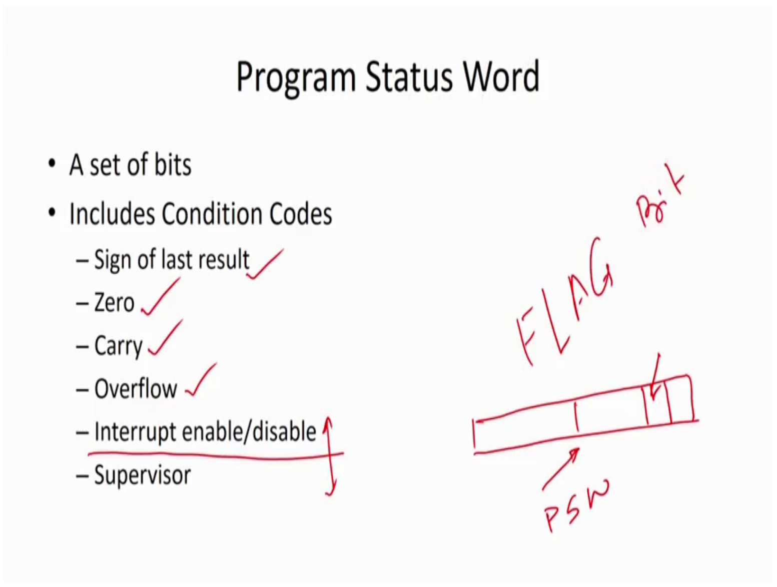
And when we are going to use those particular condition bits to make some decision. So, when we are going to design some conditional instruction say in high level language generally you write something like no, if a equal to 0 then do something else do some other thing. So, now, here we are comparing. So, depending on this comparison if it is 0 then we are going to perform this code segment; and if it is false then we are going to perform the other code segment.

To design such type of instruction in programming languages - conditional instruction, we must have some instruction inside the processor in the instruction set of the processor we have to provide some conditional instruction. So, while going to design those particular conditional instruction then we are going to use those particular condition code bits. So, one of the code bits I can give her like that sign bit. So, when we discussed about the representation of integers or representation of your real number we have seen that we can work with positive number and negative numbers. So, for that we are having an sign bit.

So, when we perform an operation say like that a plus b then depending on the result that sign bit will be positive or negative. So, depending on that say if I am going to have a sign bit over here that sign bit will be set or reset depending on the result of my ALU operation. So, if it is a positive number, if the result is positive, then we say that 0 will be source. So, 0 is going to indicate the positive; and if the result is negative then we are going to store 1, so it is going to indicate that the result coming out from the ALU is a negative 1. So, this is one of the condition code.

Like that another condition code I can say carry flag; already I have said that we are having a carry out of an inter circuit similarly we can have overflow bit condition code because you have seen that the range of numbers is restricted by the design of the processor, we cannot go from minus infinity to plus infinity. So, depending on the early result sometimes it goes beyond those particular events and in that case we are going to say it is an overflow. How we are going to identify the overflow? So, in that particular case what we are going to do we are going to keep this information in one of the condition right, and to take the decision we are going to see what is stored in this particular position.

(Refer Slide Time: 22:23)



Program Status Word

- A set of bits
- Includes Condition Codes
  - Sign of last result
  - Zero
  - Carry
  - Overflow
  - Interrupt enable/disable
  - Supervisor

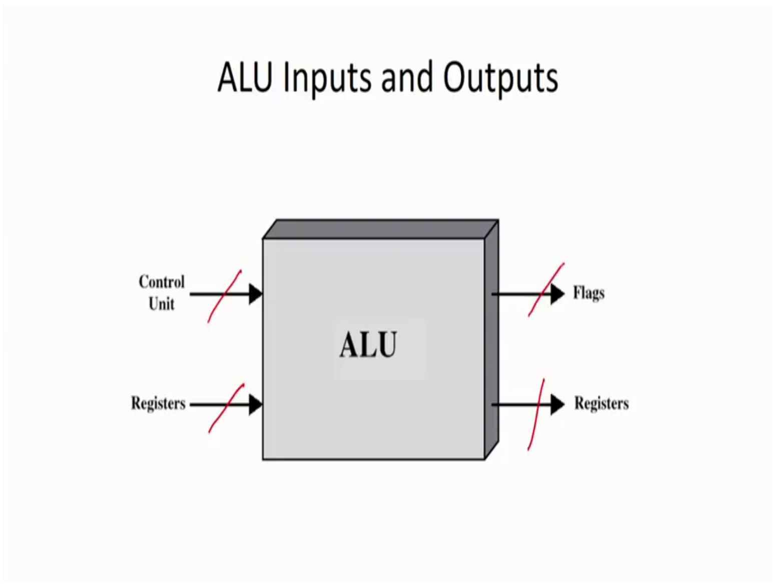
FLAG
Bit
PSW

So, this is a condition codes. And condition codes are basically known as your flag also flag bits, you can say these are flag bits or condition bit. And already I said that we having a condition code register, and what about value we have in this particular condition code register this is also known as my programs status word PSW - program status word. Basically it says that what is the status of my program at that particular point because at some point of time my result may be your negative or at some other point whatever value I have getting it may be an overflow condition, sometimes my result may be 0. So, like that we can have flag bits inside this particular PSW - programs status word condition code one bit for your sign bit the one bit to indicate the zero bit, whether the result of the processor is zero or not. One bit to indicate weather carry out is generated or not, another bit may be saying about what is a overflow occurs in the computation or not.

So, these are the bits which will be set or reset depending on my result of the ALU. Along with that we may have some other bits also condition code which may be said by your programmer while that is program. So, one of these things is maybe it is talking about the interrupt enable and disable. So, just to give an idea I am just giving you one example the interrupt means like that we can interrupt or we can disturb the working of the processors. A processor is executing a particular program, but at some point of time I want to give some input from the keyboard or maybe I want to control some other devices to the computer; in that particular case that problem will be interrupted and the processor service will be given to that interrupted request. So, that interrupt rather we are always going to allow it or for some cases we are going to disallow it. So, again to have these things we may have one bit which is going to indicate about that when interrupt is allowed or it is disabled.

So, basically sometimes it may happen that when we are performing some high priority job we feel that or we expect that no one should disturb me at the particular point; in that particular case what will happen we can be disable this particular interrupt after completion of the touch we can enable it. So, we are having some more flag bit which can be set or reset by the programmer.

(Refer Slide Time: 25:02)



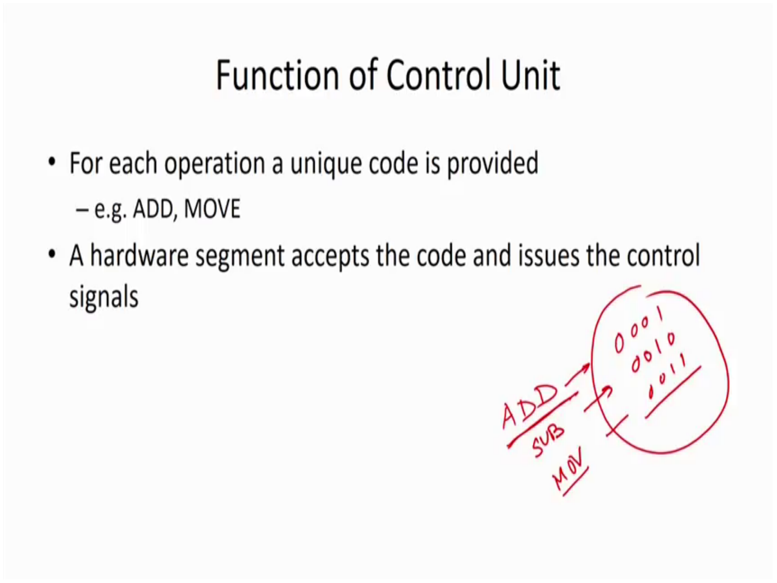
ALU Inputs and Outputs

So, these are the registers. So, what resistance we are getting basically storage element. One we are having the general purpose registers, secondly, we may have some dedicated start to store only data and store only address. We are having some special purpose register here I have mentioned for resister called program counter, instruction register, memory address register and memory buffer resister. And along with that we are having a special register call condition code register, where we are going to store the status of the processor. So, these are some storage unit or element we have inside the processor and these are known as our register.

And what is the another component inside the processor, this is the ALU - arithmetic and logic unit. So, ALU is going to perform the basic operation inside the processor. So, it is nothing but a combination of processing element. So, in an (Refer Time: 25:59) simply you can say that it is like that inside that ALU we are not going to look into it will just see it is a black box. So, what we are having we are going to give input from the registers, it will perform the operation depending on our request. So, control unit it something else will come from the control unit if you identify which operation we are going to perform. After performing the operation it is going to give us the result whatever result we are getting we can again store it into the registers. So, I have mentioned that depending on the result it is going to send some of the flag bits like that after performing some operation if overflow occurs then what will happen that overflow bit will be said by those particular signals.

So, some crack signals will also come from the ALU and according to the status of those particular signal if that we are going to set or reset those particular flag bit. So, in a higher level, you can see that ALU in this particular way we are having input, we are having output, we are having some control signal and its input and some of the data or bit flag bits as an output. So, this is another component inside a processor which is ALU, and this is the basic processing element inside the processor.

(Refer Slide Time: 27:10)



Function of Control Unit

- For each operation a unique code is provided
  - e.g. ADD, MOVE
- A hardware segment accepts the code and issues the control signals

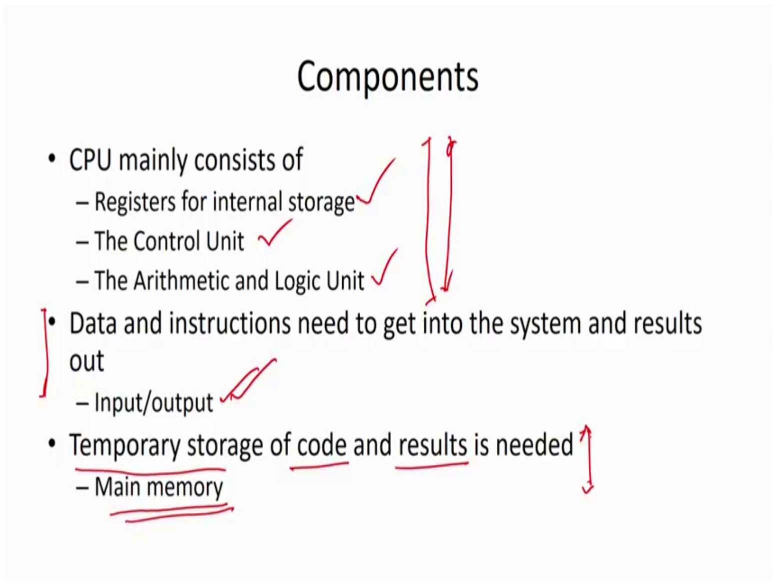
ADD
SUB
MOV
0001
0010
0011

Now another component is your control unit. So, what is the function unit or sorry what is a function of this particular control unit? So, what will happen inside a processor we are having several operation. And we say this is the instruction of the particular processor and we are having an instruction set. So, for every instruction, we are going to provide an unique code like that here we are talking about that I can perform an add operation. So, for that particular add operation, we are going to give a unique code to it, because we are going to work with the numbers or which can works with the binary digit only zeros and ones. So, one unique code will be givens I can say that this is 001 is going to give it for add operation.

So, similarly if I am going to perform a subtract operation then I am going to give another unique code. So, if I want to move the information form say register to ALU then what will happen we must have an instruction, you must have an operation for that and this operation is also going to get an unique code. So, I am going to say that 001 is the code for that if we are going to work with prove it. So, this is the way we can say that we are going to give an unique code to each and every operation. And when we get this particular unique code, we know what operation we need to perform that control unit is going to generate the appropriate signal to carry out those particular operations or carry out those particular talks.

So, when we talk about that add instruction going to perform the add operation, it may have several sub steps. So, in every sub step that control unit is going to generate appropriate signal and that will go to the appropriate component inside the processor to enable that particular component. So, this is the function of our control unit.

(Refer Slide Time: 29:05)



Components

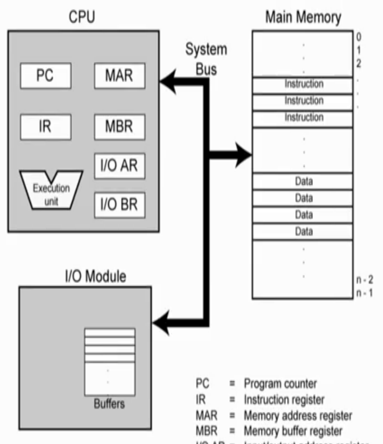
- CPU mainly consists of
  - Registers for internal storage
  - The Control Unit
  - The Arithmetic and Logic Unit
- Data and instructions need to get into the system and results out
  - Input/output
- Temporary storage of code and results is needed
  - Main memory

So, now in a nutshell now we can say that what are the components that we are having inside the processor. So, CPU mainly consists of register for internal storage, it is having an control unit, and it has an arithmetic and logic unit. So, these are the three major components that we have inside a processor; and these components are connected to a interconnection network. Now, to work with this particular processor, what will happen we have to take bring the information inside that processor. So, for that somehow we need to bring this particular information and somehow we have to give the output to the users. So, for that we need this particular input output mechanism.

So, in a simple example, I can say that keyboard is my input device. So, true keyboard I can give the input to the processor and monitor is an output device to monitor I am going to get it. Secondly, an another way we need one more component which is known as a temporary storage for code and result and it is known as my main memory, because the computer works Von Neumann stored program principle. So, you have to keep those information in the main memory. So, along with the processor we have input and output and we have main memory or storage unit.

(Refer Slide Time: 30:36)

Computer Components: Top Level View



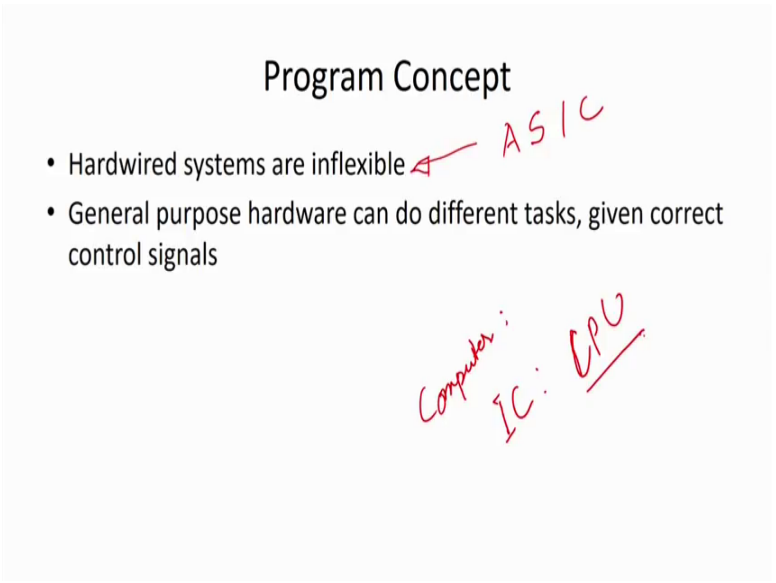
PC	=	Program counter
IR	=	Instruction register
MAR	=	Memory address register
MBR	=	Memory buffer register
I/O AR	=	Input/output address register
I/O BR	=	Input/output buffer register

So, in this particular way now we can see the components of my computer it is a top level view. So, this is the processor CPU. So, CPU we having some internal registers called already have talked about PCs is nothing but program counter, IR is nothing but instruction register, MAR is your memory address register, MBR is your memory buffer resistor along with that we are having two more additional register. We are talking about IO AR that means IO address register, and IO BR IO buffer resistor. So, these are the special purpose register along with that we are having some general purpose register also and this is the execution in basically ALU.

So, it can perform work then we have to bring the information from main memory. So, this memory will be connected to this particular processor we call this is the system bus. So, two bus we are going to connect this particular memory. Already I have talked about bus is nothing but the connecting where inside the processor also you are having an internal bus through that internal bus we are going to transfer information from one component to the other component. In case of your computer through this system bus we are going to transfer information from main memory to the processor.

Again that IO module input output model will be connected through this particular processor through this particular system bus and all the input output device will be connected to this particular IO module. So, you are going to discuss these things in detail in subsequent modules. So, this is the top view of our computer and these are the components having processor main memory and IO module.

(Refer Slide Time: 32:17)



Program Concept

- Hardwired systems are inflexible ← ASIC
- General purpose hardware can do different tasks, given correct control signals

Computer: IC: CPU

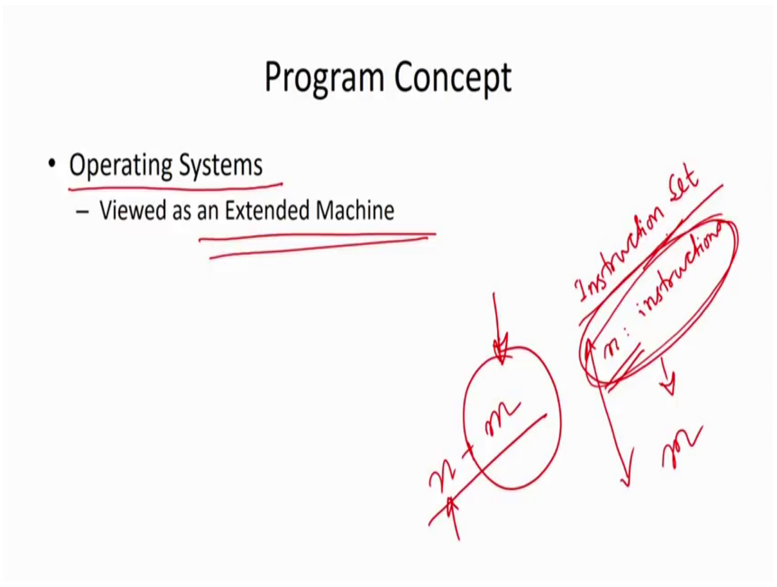
Now, what is a program concept? Now, why you say talk about a program what is a computer program it is nothing but collection of instruction and we are going to perform those particular instruction. So, here we are talking about programming concept and why this programming concept is coming into picture. It says that that I have written it like that hardware system are inflexible but computer is also a hardware device. We build the computer with the hardware which is nothing but your electronics component, but in general you can say that hardware systems are inflexible.

You know I can give you a simple example if you talk about the television, TV is also electronic device. So, we are having several components electronics components inside the television, but television is performing on particular tasks only. It receives the signals and according to the signal received, it is going to display it in the monitor. But along with that what will happen we are having some provision to sense the channel, sense the brightness or contrast all those things, but it is only going to perform that particular tasks only receive the signals and display into the monitor. And you can see, but we can there is some provision to channel to channel help of dimensional like that but we cannot do any other work.

But if you are going to loop for a general purpose devices then what will happen we must have a provision to configure it according to our requirement that means, you can program it according to our requirement. So, in that particular case we are going to say it is a general purpose hardware. So, computer can be treated as a general purpose hardware because we are having the programming facility and with the help of program we can carry out our tasks so that is why it says that in general hardware systems are inflexible.

So, if we are going to design and hardware circuit for to perform specific tasks in that case we are going to say this is your ASIC - application specific integrated circuit. So, like that we are having an ICs for our TV television. So, this is basically ASIC the application specific integrated circuit, but when you go for general purpose or when you go for computer then what will happen in that particular case we are having an IC, but this IC said this is the CPU or processor. So, this processor can be programmed according to our requirement. So, from that the concept of program is coming into picture.

(Refer Slide Time: 34:58)


Program Concept

- Operating Systems
  - Viewed as an Extended Machine

Instruction Set
n : instructions
n + m
m

So, when we are talking about the programming concept one of the issues or all of you know that we are having an operating system. So, when you are going to switch on your computer generally we said that you are booting up our computer. And when you say that we are booting our computer, you are booting it with the help of some operating system. You are having several operating system, and you know that basic one is your Windows, but you know about Unix you know about your Linux like that.

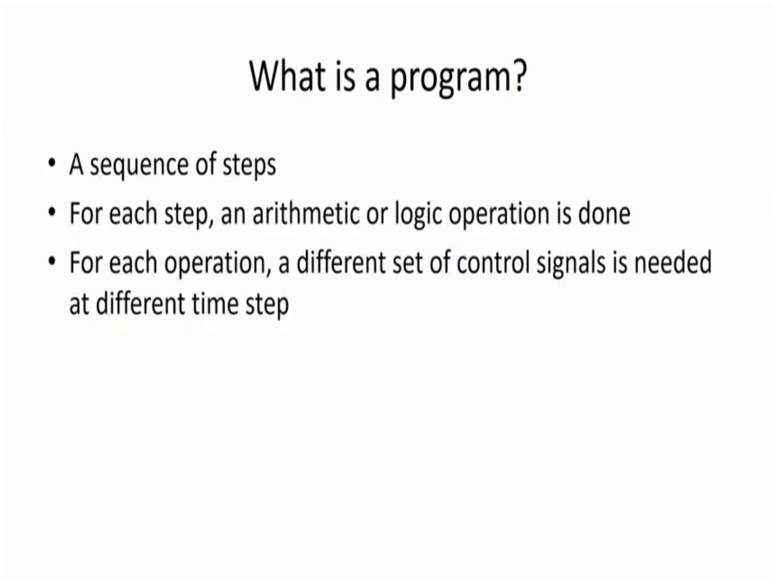
Now, what is an operating system? So, for this programming concept and for this particular you walk when we are going to discuss about a computer organization and architecture here we can view this operating system as an extended machine. Why you are talking about that extension machine, because every computer or every processor is having a specific instruction set. So, it depends on the all the processor and you are having n number of instruction. So, whatever we are going to do we are going to do everything with the help of those and instruction only. So, we have to write a program to perform any operation in that particular processor with the help of those particular n instruction only.

Now, one of the issue I can say that now whatever we have inside a memory say I am storing some information in the memory, I want to display those information to the monitor so that user can see it. So, now, when I am going to perform this particular task, taking the information from memory and displaying it into the monitor, I have to write a program. And when I am going to write a program then what will happen I have to know this particular instruction set and with the help of those instruction set, I am going to write a program to display information from memory to the monitor.

Now, what happen if you see as a common user, you have to write this particular program every time. And you have to know that complete instruction set. So, for that what will happen such type of routines, we are going to write in one go and we keep everything in one place and we say that this is that component of my operating system. So, when we would tell all those instruction will be in place, so that I can use it. So, now, that saw where I am saying we are going to view this operating system as an extended machine.

So, whatever an instruction we are having with the help of those instruction, we can create another say m number of instruction to do define operation. So, total instruction I can say that n plus m, this m instruction are basically some software written with the help of those n instruction, and they are integrated together and kept in the part of this particular operating system. So, this is why we are saying that opening system can be treated as an extended machine and what we are having basically in operating system. It is a collection of some programs and these programs are made out with the help of that particular processor. So, these are the programs concept that we have in computer.

(Refer Slide Time: 38:22)

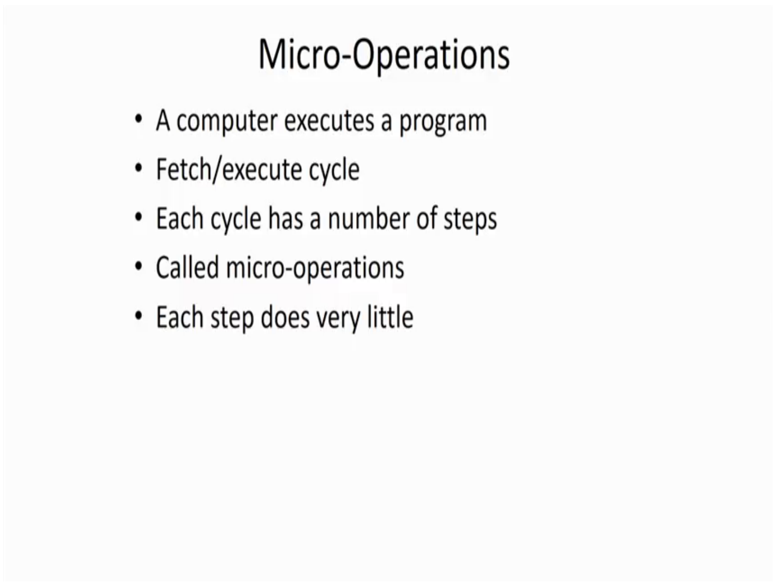


What is a program?

- A sequence of steps
- For each step, an arithmetic or logic operation is done
- For each operation, a different set of control signals is needed at different time step

Now, what is a program? Now, in a nutshell you can say that it is a sequence of steps. So, we are having sequence of step, we have to write in proper sequence. And when computer is going to execute this program then it is going to execute those particular step or instruction one by one. And for each step an arithmetic or logic operation is done because we are having the ALU, we are going to perform some operation with the help of this processing element. And for each operation a different set of control signal is needed at different times stamp, because already I have said that when I am going to performing or execute an instruction, we have to perform some tasks. And to performance some tasks means we have to transfer information from one point to the other point, we have to perform some operation. So, control unit need to generate those particular signals at appropriate time.

(Refer Slide Time: 39:19)

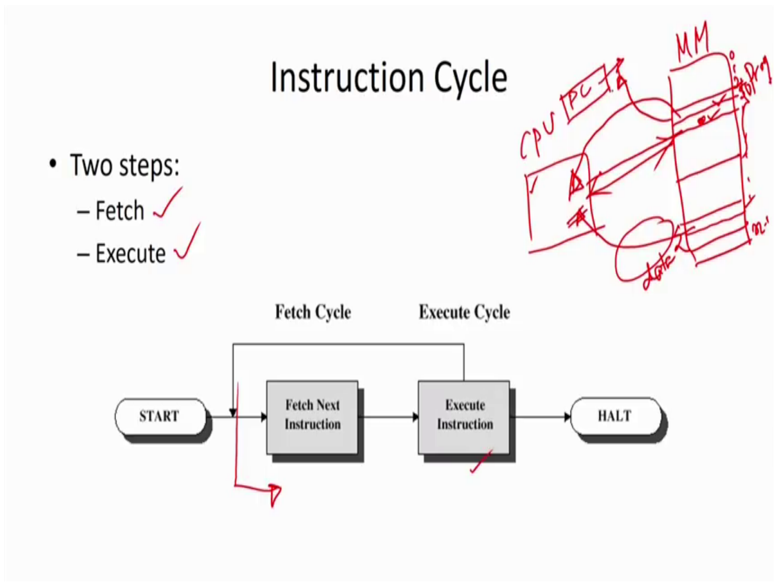


Micro-Operations

- A computer executes a program
- Fetch/execute cycle
- Each cycle has a number of steps
- Called micro-operations
- Each step does very little

Now, when we talk about an instruction in one go we cannot have the effect of the particular instruction, it is going to execute this particular instruction; and at some point of time, I have mentioned that instruction cycle can have two cycle once fetch and second one is execute. So, we can say that two different task we are performing to complete that particular instruction. Now, in every cycle, we may have again several subtypes and combining all those particular subtask, we can say that we are completing the fetch cycle. Similarly, in execution cycle, we are going to perform different subtasks, after completing those particular subtasks, we are going to say that execution cycle is over. So, each cycle like fetch cycle and execute cycle has number of steps; and all those steps are doing very little work and we are saying that these all little works are basically some micro operation.

(Refer Slide Time: 40:32)



Instruction Cycle

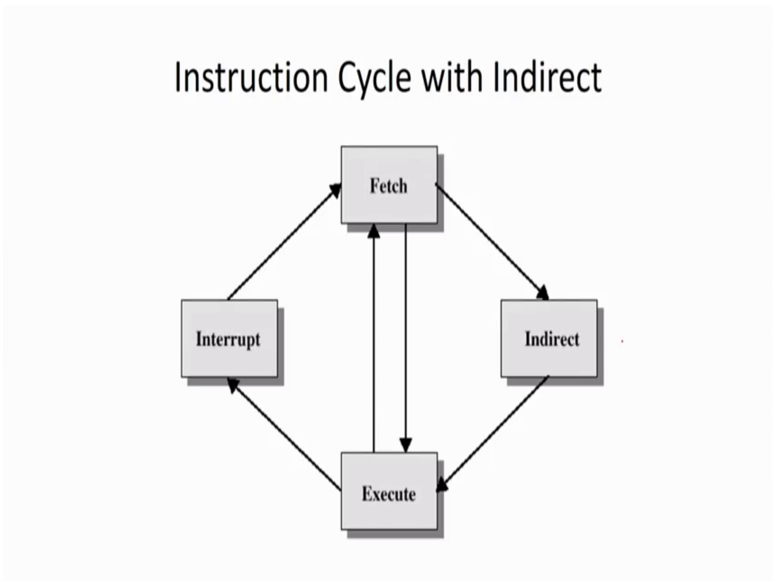
- Two steps:
  - Fetch ✓
  - Execute ✓

So, after combining all those micro operation, we are going to effect of the operation. So, in this way we can see now here we are going to see how we are going to look for those particular micro operation. So, this is the instruction cycle in nutshell I can say that it is having two cycle, one is your fetch, second one is execute. So, what fetch means we are fetching the information, we are fetching the instruction then processor will be knowing what we need to do, and accordingly processor is going to carry out the job and we said this is the execution fetch.

So, basically where from we are getting it computer works on Von Neumann's stored program principle that means, we are having this particular processor. And I am having this particular main memory, and it is connected through this particular system bus. So, in main memory we are having this particular program we are storing it. So, somewhere we are having this particular data. So, when I am going to execute an instruction that instruction in this particular memory location, I have to bring it to the processor. So, this is called fetch. I am fetching the information. And after that once I fetch it now I know what to do accordingly I am going to execute it; and after completion of this instruction then I have to go for the next instruction because program is a collection of instruction.

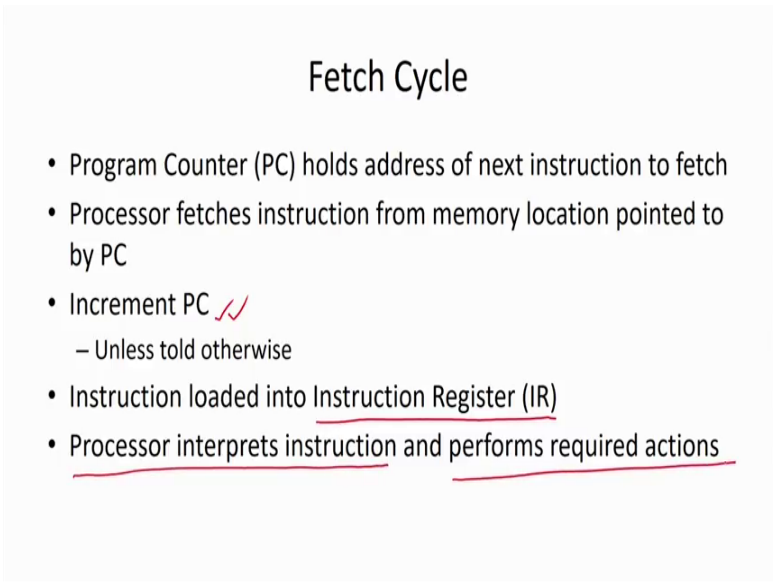
And we have to execute those instruction in sequence, so that is why this cycle is repeat. So, now, this is the fetch and this is execute.

(Refer Slide Time: 41:54)



Another one we are having indirect cycle all ready I have mentioned that sometimes when I am fetching the instruction before execution I have to need the data. So, for any getting the data I can go through this particular indirect cycle. So, in this particular case, I just say I am fetching the instruction to the processor. After getting the information that I need to perform some operation and processor now see that we have to act on some data, but these data are not available inside this particular processor. Then it will go to the indirect cycle and from to this indirect cycle since I know that my data is somewhere in this particular memory location it is going to bring this particular data instead of processors. So, this is basically indirect cycle. So, in indirect cycle we are going to bring the information inside a processor. Now, once my instruction is available once data are available inside the processor then processor can carry out those particular operation. So, this is indirect cycle.

(Refer Slide Time: 43:01)



Fetch Cycle

- Program Counter (PC) holds address of next instruction to fetch
- Processor fetches instruction from memory location pointed to by PC
- Increment PC
  - Unless told otherwise
- Instruction loaded into Instruction Register (IR)
- Processor interprets instruction and performs required actions

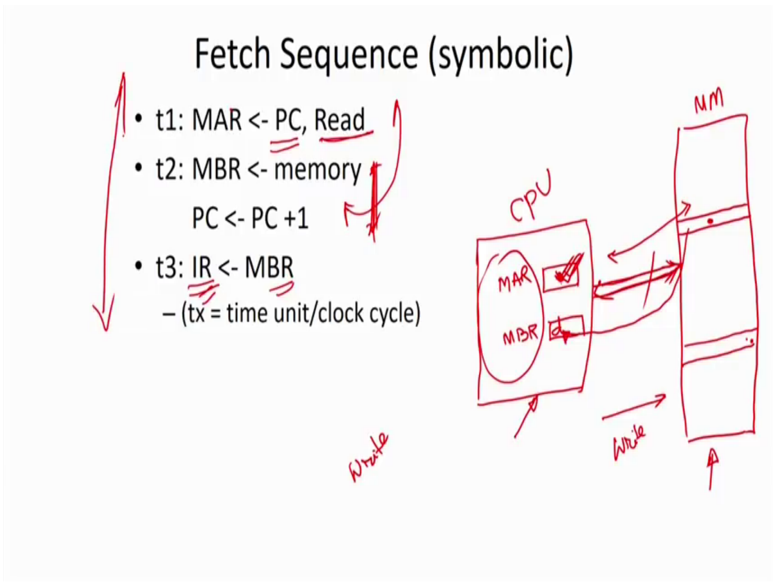
Now, basically what we are going to do in a fetch cycle. So, it is a fetching and information from memory to the processor. Now, what we must know when we are going to fetch an instruction, at least we have to know the memory location where we have the instruction. Now, where I am going to get this particular information. So, already I have mentioned that we are having a special purposes register are called program counter, PC - program counters. So, in that particular case, what will happen I am having a call register called program counter, and program control will have the address of this particular memory location.

Thus say that it is a number, (Refer Time: 43:41) you just think as a number said this is the zero at address first location, second location like that we are having total n minus one location. So, 0 to n minus 1 total n location. So, we note that address is say 50, then program counter will have the value fifty over here that means, program counter is going to give us the information from where we need to fetch the instruction. So, we are having a register called program counter, and this program counter is going to give us the information from where we need to fetch the instructions.

Now, after fetching one instruction then what will happen we have to after completion of this particular instruction, we have to fetch the instruction from next memory location, because it is in the sequence so that is how you can say that sometimes we have to increment the PC also. First it is going to have the in address of an instruction processor fetch this information from memory to the processor, and along with that it will increment PC because after completion of this particular instruction, what will happen we have to go to fetch of the next instruction, and next instruction will be available in the next memory location. So, after fetching information, generally we update this particular program counter, we just increment it.

After that whenever we are getting this particular instruction, this instruction will be loaded to instruction register. We have said that we are having a special register called instruction register. So, when we fetch an instruction, after fetching it, we are going to keep it instruction register. Once we have the instruction in the instruction register, then processor is will be knowing what operation we need to perform, so that information will be given to the control unit, and control unit is going to generate the appropriate signals. So, once it is having in that instant instruction register. So, now we are saying that processor interprets instruction and perform require action, so that means processor interpreter instruction. So, after getting the instruction, we know that we have to perform say addition operation then processor is going to perform the required actions; that means, control unit is going to generate the correct signal at correct time, so that operation can be performed in correct way. So, this is the fetch cycle.

(Refer Slide Time: 46:13)



Fetch Sequence (symbolic)

- t1: MAR <- PC, Read
- t2: MBR <- memory
  PC <- PC +1
- t3: IR <- MBR
  - (tx = time unit/clock cycle)

Now, in a symbolic way, now I can say how we are going to do the fetch cycle. Now, already have talked about PC that program counter that we are having. And we know IR instruction register after fetching it we are going to put it into the IR. Now, how we are going to do it. Again we are having two special purpose register, one is known as MAR - memory address register, and second one is your MBR - memory buffer register. So, these two register are basically the interfacing register of my processor.

So, now, what I can say this is my processor I am having a register called MAR and another resister called MBR memory address register and memory buffer register thus said that this processor is connected to main memory or maybe to IO devices also. Now, how we are going to interact, how we are going to fetch it. You just see that somehow I have to give the address of the memory location, and whatever data we are having over here we are going to bring it into the processor. So, these two resistors are going to act as an interfacing register.

So, if I want to be the information from a particular memory location then first that address we have to put it into the MAR. After that this address will go to this particular memory need through this system bus then we are going to read the information, so that is why saying that first I am going to place the information from PC what we have in PC. In PC, we have the address of the memory location from where we need to fetch the instruction. So, first that the information of PC's transfer to MAR now in MAR we are getting the address of the memory location. Now, I will generate the read signal. So, control unit will generate read signal what it will say that now we want to read the contents of this particular memory location. And once we are getting this particular information then reading it one read is complete, then what will happen the information will come to your MBR.

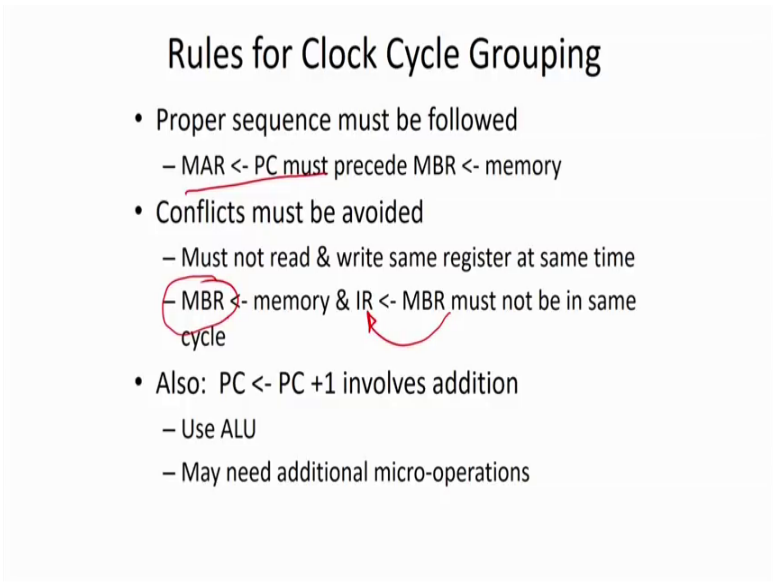
So, now say first cycle clock we are placing the PC to MAR then giving a read signal we are going to read it. Now, in second cycle what we are going to do, we are going to take the information from memory to MBR. And along with that what we are doing we incrementing the programs on the PC equal to PC plus 1. Now, one this time cycle, two is over then what we are going to do now that information is inside my MAR. Now, what we are fetching now we are fetching the instruction; that means, we have to execute this particular instruction. So, in the third clock cycle we are transferring the information from MBR to IR instruction say register. So, now, I know the instruction what instruction we need to execute, now accordingly now control unit is going to behave control unit it is going to act and going to execute this particular instruction

You just see here that means, to fetch itself we need three clock cycles or three step. Now, I can see in the second step, what we are doing we are covering two operation memory to MBR and PC plus one adding the PC and keeping it up updating the program counter value. Now, why it is possible basically say I want to mention it said this is a processor CPU, it is an electronic device, this is a memory unit, this is also a semiconductor device, but the speed of the processor is not same with the speed of the memory, memory is always slower than the processor.

So, for that while I am transferring this particular information, it takes more time. So, since it is taking more time, so during the time inside the processor also I can carry out some work. So, the terminal PC is equal to PC plus 1. So, this is the way we can do it. Again this is a read we are reading it from memory to the processor, similarly we can have the write operation also; in write operation what happens we are going to write the information from a register to the memory.

So, in the particular case, what will happen first will give the address to MAR that means, we are going to identify the memory location, where we are going to write or store the information. Then we will put our data in to the MBR - memory buffer register then will give the write signal. Then what will happen whatever information we have in our MBR that will be stored in this particular memory location. So, we have been two operation, one is call read operation and write operation. And while we are interacting or interfacing with external to the processor mainly that a MAR and MBR is going to act as my interfacing register.

(Refer Slide Time: 51:34)



Rules for Clock Cycle Grouping

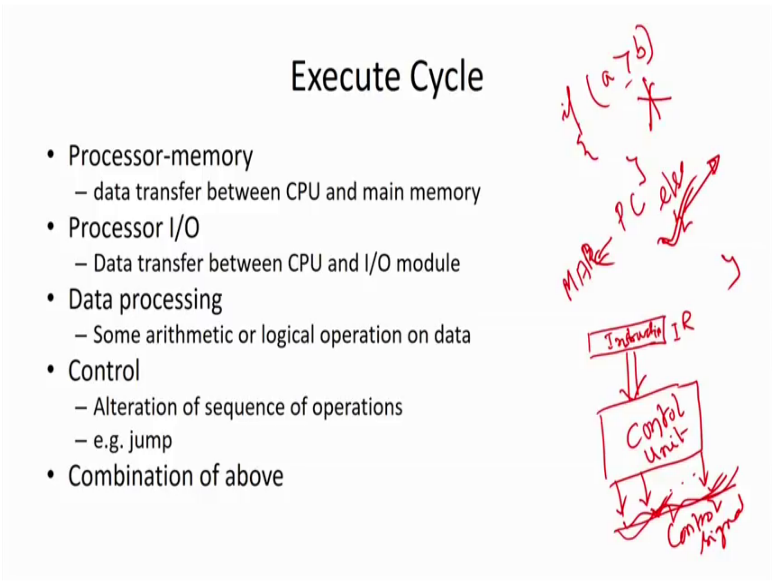
- Proper sequence must be followed
  - MAR <- PC must precede MBR <- memory
- Conflicts must be avoided
  - Must not read & write same register at same time
  - MBR <- memory & IR <- MBR must not be in same cycle
- Also: PC <- PC +1 involves addition
  - Use ALU
  - May need additional micro-operations

Now, here what will happen in this particular case, we are doing some operation parallely or sometimes you can say that we are performing this step then only coming to new step why I cannot perform both together. So, there is some research conflict that is why everything cannot be done in one clock cycle we have to perform in different step. So, for that there we have to follow some rules what are the signals that can be grouped together and we said this is a clock looping first thing is that we have to maintain a proper sequence.

So, what will happen PC to MAR always precede memory to MBR. So, basically is associated after knowing the address only I can get the information from memory to MBR. So, these operation cannot be performed before the first operation. So, this is the proper sequence. So, first I have to place the program counter to MAR then only I will be knowing. So, p PC to MAR must precede the memory to MBR. Again some conflict must be avoided why I am saying that everything cannot be done it one go because they are some conflict may occur. So, we should not must or we should not read and write the same register at the same time, because if we are reading from some memory location or reading from a register, at the same time we should not write the same in some information to the register because it will change the contents of the register. So, read and write should be in two different clock cycle.

Again memory to MBR, and MBR to IR cannot be done in the same cycle because here we are storing some information, it will take some times after getting the proper result only I can transfer it to IR. So, they cannot be done in the same time. So, like that we have to see what are the things, what are the signals can be generate the same times what other subtasks can be done in the same time, they will be done in one clock cycle. If there is some resource conflict then surely we have to go into two different step, so that is why by looking into all those particular scenario, we are saying that fetch cycle is going to take three different clock step, it cannot be done in one clock step or two clock step.

(Refer Slide Time: 53:56)



Execute Cycle

- Processor-memory
  - data transfer between CPU and main memory
- Processor I/O
  - Data transfer between CPU and I/O module
- Data processing
  - Some arithmetic or logical operation on data
- Control
  - Alteration of sequence of operations
  - e.g. jump
- Combination of above

if (a>b)
MAR ← PC else
Instruction IR
Control Unit
Control signal

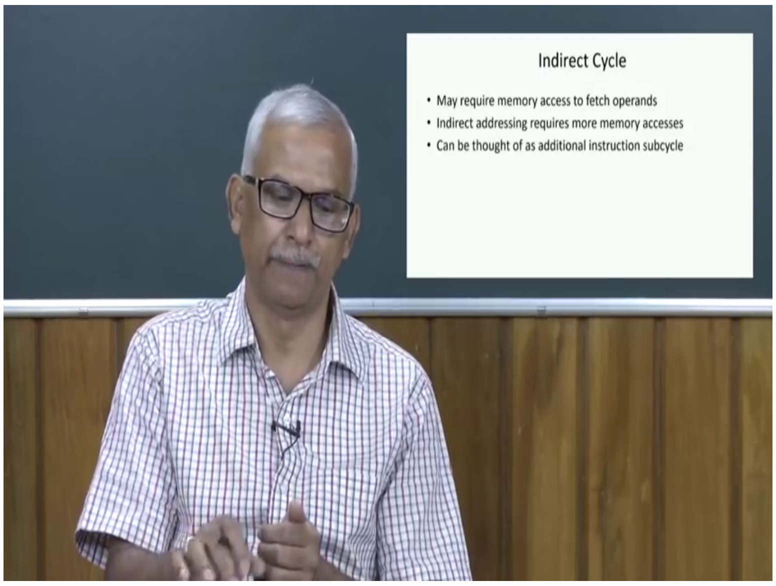
Now, what is an execution cycle? Now, say once we are getting the information in IR in instruction register, now I am having that instruction, some instruction we have fetch and you are having it. Now, according to this instruction what will happen we have to perform our operation, so this information will this really go as an input to your control unit. So, we are having a control unit inside a processor. Now, this control unit is going to generate those particular different control signal. And those control signal is going to control define components or basically going to control perform different task. Like that so first in the fetch cycle which we are talking about it is a PC is transfer to MAR.

So, in that particular case control unit is going to generate some signal. It will indicate that whatever value we have in the PC's a swipe that the contents of the PC and it will generate another signal, it will say that now whatever information we have read from the PC now write it in the MAR or store it in the MAR. So, these are the control signal will come out from the control unit. . So, like that after getting the instruction we are going to get giving it to the control unit, now controlling it will generate a control signal.

So, we have what are the operation basically we are going to do, one is your processor to memory; that means, data transfer between processor and main memory. So, this is basically transferring the information from processor to memory or memory to processor this is one kind of operation that we can have. Another class of a operation we may have processor IO that means, information will be transfer from processor registered to some output device or maybe from some input device we are going to take the information to the processor register. So, these are another class of instruction. So, depending on that we have to perform what task. So, in execution cycle basically are going to perform those operation.

Third categories we are talking about the data processing, basically say if I am having my data then how will find if it is an add operation now you have to perform the add operation. So, we are going to use the ALU, the adder of the ALU to perform the operation. So, control unit it is going to generate those particular appropriate signal to trigger those particular device. And another one it is control or sequence control basically in some of the cases we are having jump instruction or jump on condition. So, such type of operation we have to do for that we need appropriate instruction. Like that I have all ready mentioned that if a you are going to write a program like that a is greater than b then perform this instruction else perform (Refer Time: 56:47). So, if it is false then what will happen we are not going to execute this part of code, but somehow we have to come to this particular instruction. So, for that we need those particular control instruction or some instruction may be combination of some of those particular operation. So, we have executed execution cycle or execute cycle we are going to perform those particular tasks.

(Refer Slide Time: 57:11)

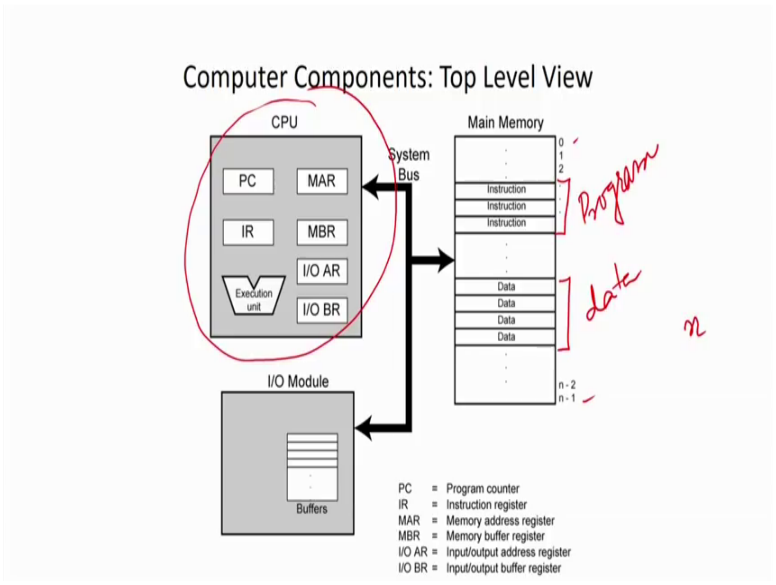


Indirect Cycle

- May require memory access to fetch operands
- Indirect addressing requires more memory accesses
- Can be thought of as additional instruction subcycle

So, in that cycle I have all ready mentioned. So, if we need data for my instruction we will go to the in that cycle and we are going to fetch the data and keep it in some register inside the processor which are nothing but the temporary storage inside the processor and accordingly we are going to use those things.

(Refer Slide Time: 57:30)



Computer Components: Top Level View

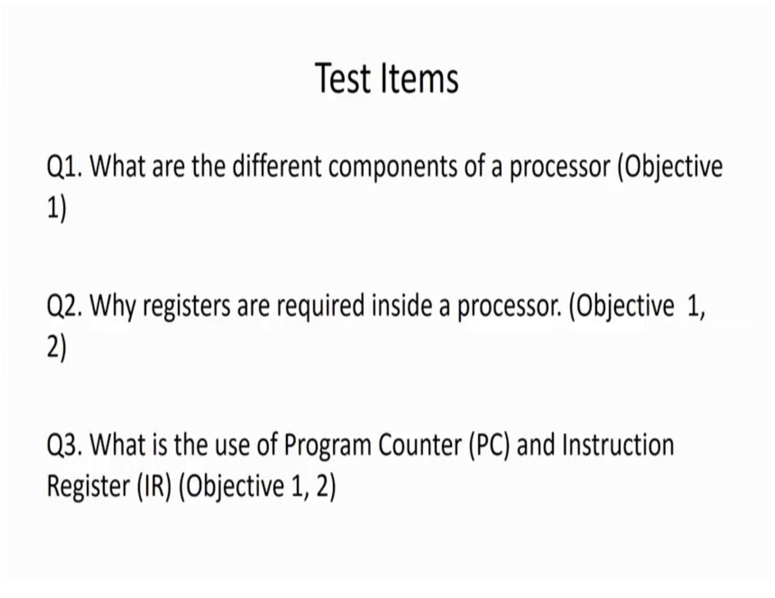
PC	=	Program counter
IR	=	Instruction register
MAR	=	Memory address register
MBR	=	Memory buffer register
I/O AR	=	Input/output address register
I/O BR	=	Input/output buffer register

Now, this is the component that we have discussed here today. So, processor is the main component central processing unit it is having a processing element and this processor is connected to main memory or IO devices through system bus and works on Von

Neumann's stored program principle. And now we have seen what are the basic components that we have inside this particular processor, till now we have not discussed anything about a main memory, we will see the construction of the main memory while they are going to discuss about the memory module. Secondly, we are going to see the complete design aspect of this particular processor when we are going to discuss about the central processing unit or the processor design, but here just in top level we have seen how we are going to perform an operation and what are the task that we are going to perform.

So, with this we are giving an having an brief idea about what we have inside the processor why memory need to be connected to the processor because it works on Von Neumann stored program principle and how a program is executes basically it is having basically two part I can say that fetching of instruction and execution of instruction. Fetching is basically done from the fetching information from memory. So, here in this case you just see that we are having 0 to n minus 1, total n memory location. So, in this particular memory location we are having those instruction. So, you can say this is my program and while I am going to execute this program it needs some data. So, this is the data that we have here in the main memory. So, processor need to take those things. So, this is the basic ideas about the components of a computer and also you have mentioned about the components that we have inside a processor and see what is a program and how we are going to execute a problem.

(Refer Slide Time: 59:27)



Test Items

Q1. What are the different components of a processor (Objective 1)

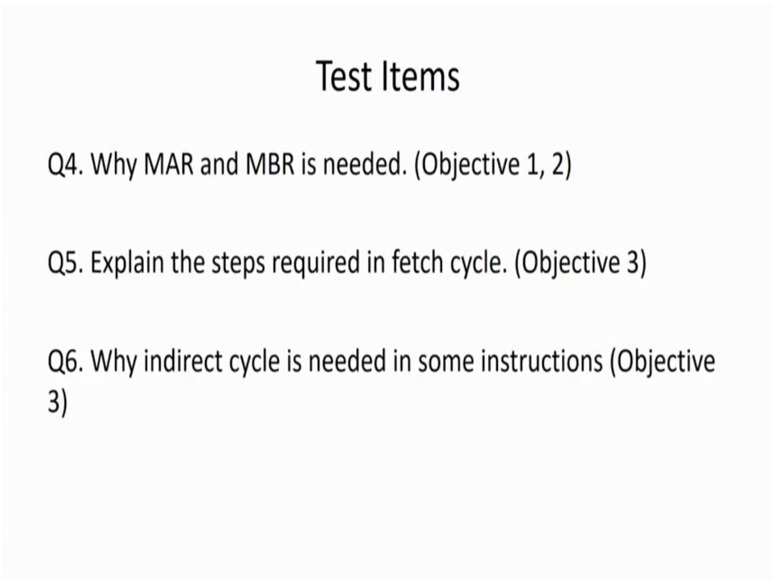
Q2. Why registers are required inside a processor. (Objective 1, 2)

Q3. What is the use of Program Counter (PC) and Instruction Register (IR) (Objective 1, 2)

Now, just look for some test item that whatever we have discussed today. So, first question I have given here like that what are the different components of a processor. I think why now you know all those things, but you do not know about the details how we are going to design it, but in top level you know it. So, we have identified some objective for this particular unit and this question is basically related to objective one. So, if you full fill the objective one then you will be able to answer this particular question.

So, basic three components you are having ALU, registers and control unit, and they are connected to a inter connection network. Why registers are required inside the processor, again it is going to meet the objective one and objected two. So, this is basically temporary storage. We have to have our information inside a processor, computer works on Von Neumann's store program principle. So, information are in your main memory. While we bring this information from main memory to the processor, we need some storage element we have to keep close information in some place. So, these are nothing but the registers. What is the use of program counter and instruction registers, I think already I have mention about it, what we are going to keep in program counter and what we used to keep in our instruction register.

(Refer Slide Time: 60:51)



Test Items

Q4. Why MAR and MBR is needed. (Objective 1, 2)

Q5. Explain the steps required in fetch cycle. (Objective 3)

Q6. Why indirect cycle is needed in some instructions (Objective 3)

Why MAR and MBR is needed again this going to fulfil objective one and two, I think in my lecture I have mentioned that these are the interfacing register for processor. So, you have to give the address of a memory location, we will give it in the MAR. And whatever data we are going to use either I am going to take it from memory or we are going to write it in the memory, it will pass through this particular MBR. So, explain the steps required in fetch cycle. So, instruction execution is having two cycle fetch and execute fetching is nothing but taking the information from main memory to processor. So, we have to perform a specific task for complete this particular fetching of the instruction. And this is same for all the instruction, and already we have discussed how we are going to do it.

Now, question 6, why indirect cycle is needed in some instruction this is objective three, because if instruction needs the data, but data is not available inside the processor then you have to take it from memory. So, with this I wind up this particular lectures, hope you have enjoyed the lecture.

Thank you very much.