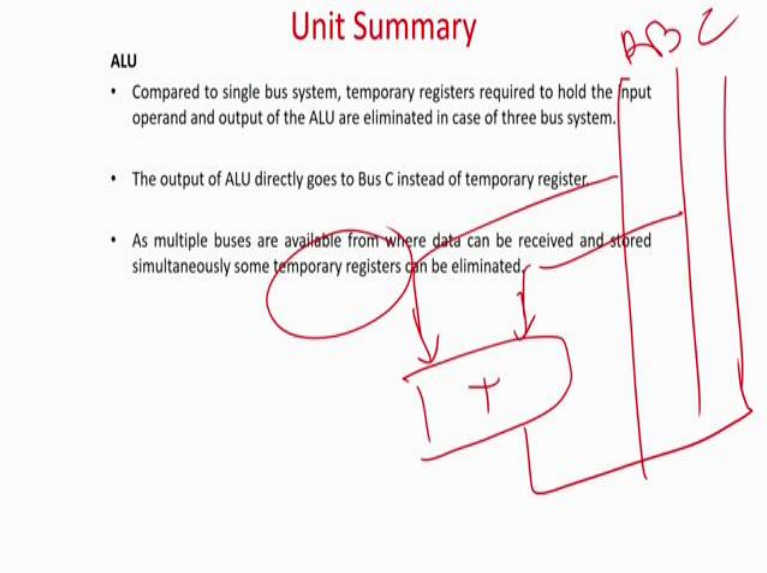
(Refer Slide Time: 15:37)



Of course; similar because we are not handling multiple instructions. So, ALU can do only one operation at a time.

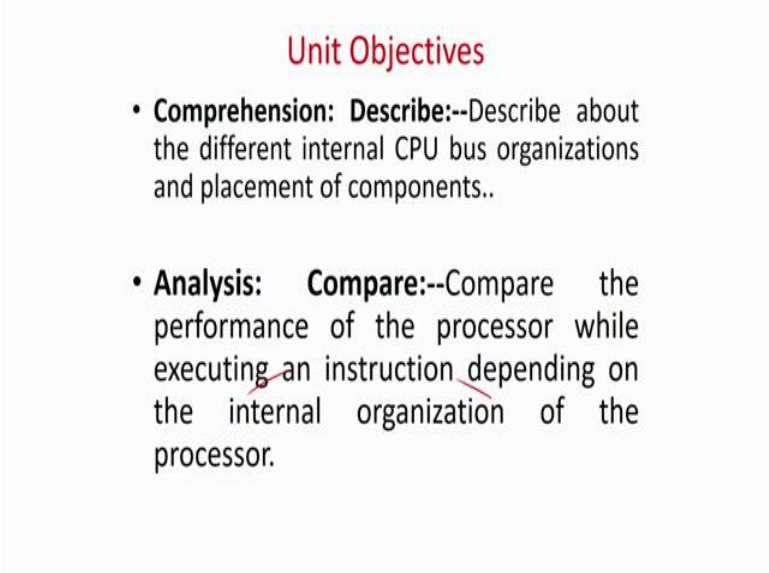
So, ALU just like this if you look at it, it is something like this and something like this. You cannot there is no a means point in putting multiple input output ports and because we are actually not going to handle multiple instructions, but one actually changes here, this is something interesting.

So, as I told you if you remember if it's a single bus architecture then first the data comes here and gets stored in a temporary variable. Second stage this value gets directly fed over here the addition or subtraction is done, and the output is also stored in a temporary register. And then in the third phase this one will go over there this one will be erased the erase means it is nullified, and the value will travel to the bus and go to the register. But if you look at if there are multiple registers multiples wires, then actually all these things all these temporary registers can be done away. So, we can have three. So, this one will go over here, this one will go over here and this one will go over here ABC. So, that is the only difference in the context of ALU, that there is no buffering registers or there is no temporary registers are eliminated. So, that will be the difference.

So, in summary we are going to look at three bus architecture, how control signal changes and how the different components basically requirement of different components like ALU,

program counter, memory data register, memory buffer registers change in this context that is what we are going to learn in this unit.

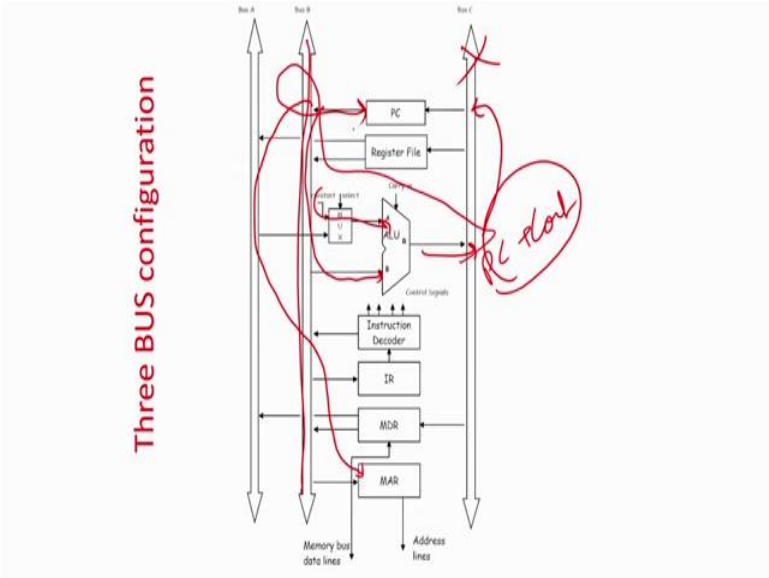
(Refer Slide Time: 17:02)



So, what are the basic objectives of the unit, the basic objective of the unit is one is a comprehensive objective, that is we will be able to describe about different internal CPU bus organization and placement of components. That is if I give you a single bus architecture, two bus architecture, three bus architecture, you will be able to design the entire system and place the different components like ALU, registers etcetera.

And then analysis is there you can compare the performance of the processor, while executing an instruction depending on the internal organization of the processor. That is whether this is the single bus architecture, two bus architecture, three bus architecture then if I am executing some instructions, what are the number of steps required, how the control signals will change, what are the advantages etcetera you will be able to compare so, that is these are the two basic objectives of this unit.

(Refer Slide Time: 17:47)



Before we start this one let us pay a very careful attention on this three bus architecture. So, again I will first show you that is the three bus architecture, we have three buses A, B and C and these are the internal components like PC, register, ALU, decoder, instruction register etcetera. But now we will try to assume here that bus A and B if you look at it, bus A and B are going to take the data from the output of the registers and blocks and bus C mainly basically is going to take the data and write it to the registers. And for ALU it is just the reverse, again I am repeat it general trend in this architecture whatever values you have to dump from program counter any other register will be dumped to bus A and B. So, mainly bus A and B are going to read the values and you are going to write the values sorry bus A and B are going to take the values from program counter, register files etcetera and the registers are going to read from bus C.

And only for the arithmetic and logic unit it will be just the reverse, it will read from bus A and B and write to bus C and we will see why it will help. So, with this assumption we will take some time and look at this bus in elaboration. So, there are two buses A and B. So, you can just assume that for all the registers they will be writing the values from the registers to bus A and B and they have two ports. So, if you see the register file that mean R_1, R_2, R_3, R_4 and so forth. So, they have two output ports.

So, same value can be dumped at register A and. So, bus A and bus B. So, same register values or the register output will be now available in two bus simultaneously. And bus C is generally

going to give the out int input to the registers. Program counter generally is writing to bus B you can also think that I am going to write to bus A, but generally as the program counter give the values only to memory address register to find out the next instruction is not advantageous or not necessary that you dump the values in both A and B not necessary. But in bus C it is going to give the inputs through program counter or any other register.

Now, let us look at the ALU here is slightly different. So, in the in all other cases basically the registers they are going to dump the values to A and B and read from C, but ALU is slightly reverse they will take the values from A and B and the output will dump to C this is obvious basically it's a mirror image kind because you are going to give one some registers we are going to give the values which has to be computed upon. So, the register will write the value to this bus and this bus they are going to take the inputs and the output is going to be dumped to C. So, that is very obvious that registers are connected to the inputs of the ALU and the output of the ALU is again going to writes write to the registers or the variables that is A+B=C.

So, what are these are the outputs A, B and this is the going to be the output with the C. So, just you are remembering as default that register files are writing register bank is writing to bus A and B which are inputs to the ALU or some other blocks and the computation output will go via C and it will go to and write the register blocks. Again this is very similar to the single bus concept mux with a select it is a constant, then program counter will be incremented, otherwise it is going to take the operand from bus A. Bus second operand B is directly connected to the ALU.

Instruction register is going to take the value from bus C again for instruction register slightly the other way around it is going to take the data from some instruction register from bus B instruction register is going to take the value is going to for the instruction decoder, and it is going to write the values of the instruction decoder value to bus B. So, this is the cycle over here we will later see how it is advantageous and why the connection is in this manner.

Memory data register again it will take the value from bus because they memory data register is also a register. So, it will take the value from bus C and it will dump the value together to bus A and B. So, in fact, basically what happened all the registers excepting instruction register and ALU are connected in a very similar kind of a fashion. That is the dump the values to bus A or B or both and read from bus C. ALU is just the reverse it will take the values from A and B and write to C, because its A+B=C. So, whatever our output for the registers will become

input from the ALU and what are the inputs for the register file it's the output for the ALU that is obvious.

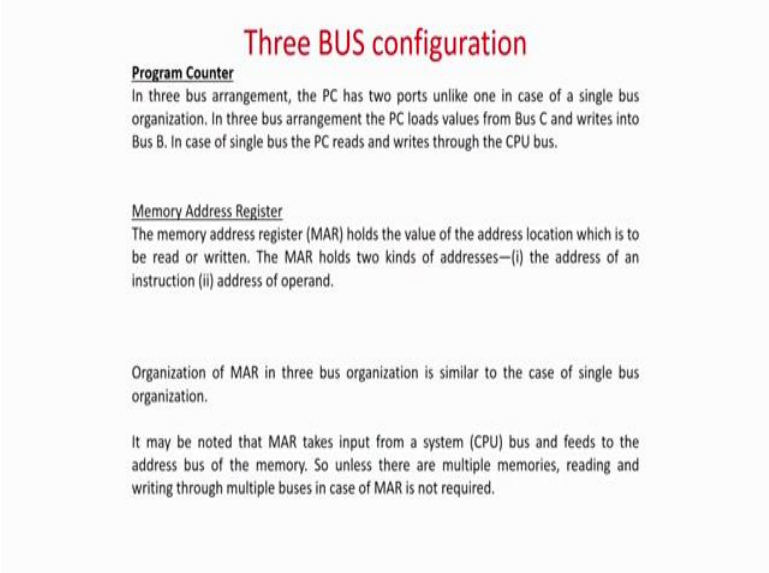
Just memory data register is also the same thing and the interconnectivity of the instruction register is slightly different, it takes just the value from IR from bus B and it dumps the value basically to bus B or C in fact, I mean this is in fact, I mean if you say, this is in fact, not very much required because what happens is that it generally generates this multiple signals. So, basically it reads from bus B and generally it generates multiple signals.

So, this signal basically whatever in this part is not that very relevant, then basically if you look at it the last block which is important here is the memory address register. So, sorry in memory data register there is one more port basically which is the input which is coming from the memory. So, memory data register it has four ports, it dumps the value to two register two lines A and B takes the input from line numbers C and also it can the bidirectional bus if you see it goes to the memory. Of course, if you remember what is the memory data register? Memory data register writes to the memory by this line and also reads from the memory, if it reads from the memory so, this will be the bus. So, it's a four port architecture and it's a bidirectional bus.

And there is something called the memory address register. So, memory address register has nothing but it just reads the value from bus B and then it feeds this to the memory. So, memory address register has only a single port, because we are assuming a single memory kind of in a block here. So, it takes the address register from one from a bus B it will take the address from where you have to read the instruction or data and this is the single address line because there is a single multiplier block sorry single memory block.

So, that this in a nutshell shows what is the basic architecture we are following, but in fact, slight changes you can yourself do like for example, memory address register is reading from bus B. You can make it to read from bus A and drop bus B like. So, this is why some interconnection changes can easily be done over this. But in our examples will be strictly following this three bus architecture, and then we will on this one will try to see basically what are the signals what are the number stacks etcetera required to execute the instruction. So, for the time being look at this slide for one minute and then drawing your mind that this is going to be the basic architecture and how I am going to actually execute the instruction on this three bus architecture.

(Refer Slide Time: 24:36)



So, again just read through the slides. So, what are the difference in between a single bus architecture and a multiple bus architecture? Let us look at the program counter you can read the text over here. So, I will just tell you; what is the difference? And then again we can come back. So, this is the program counter. So, what the program counter does? So, the program counters basically writes the value to bus B, which can actually feed the value to the instruction register. So in fact, this program counter is writing something and it is going to the value of instruction register.

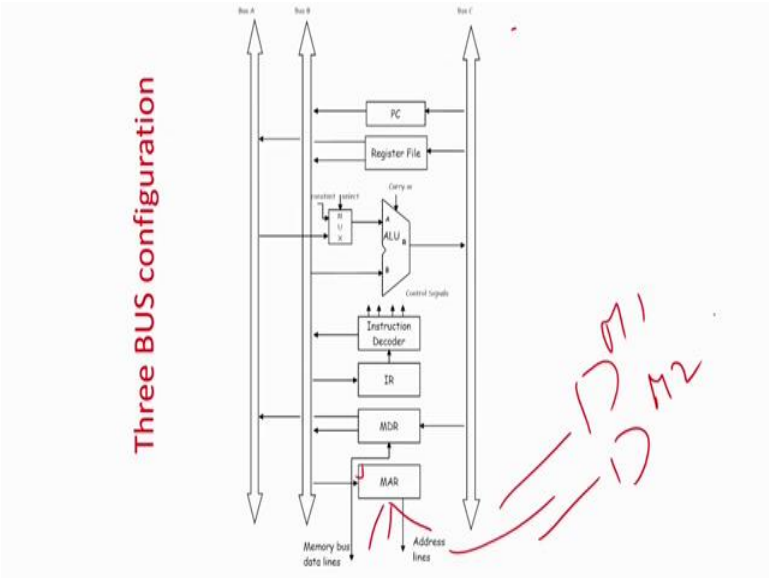
At the same time if you have to also increment the value of program counter equal to program counter plus 1. So, very easily same time you can pass this value to B and you can take this constant, and you can dump the value over here and it is can be directly going to the PC. So, single cycle you can do it. You write the value to instruction register sorry this one will go to the memory address register I am sorry this line will not be there sorry ok. So, this line will be not there PC will write to the memory address register same time you dump the value to B as the second operand from part A, you take the value of constant and output here will be equal to PC plus constant and same line you can directly feed it to the program counter.

If you compare the single bus architecture, this part was not there basically. So, if this part is not there, in one step you can get the value of program counter to the memory address register, this one you can dump it over there PC will be PC + 1, but this line is not there. So, you have to wait till this bus is free because that is the only available bus, then only you can route it from here to here. So, that was two steps if there is a single bus architecture, but in this as there is a

three bus. So, you can very easily do it in a single stage and also if you look at. So, there is a two port program counter.

Next it is saying that the memory address register. So, memory address register basically if you look at it, it will take some value from bus B and write it to the memory address to the memory. Of course, single memory is there. So, we cannot have much more ports to help us.

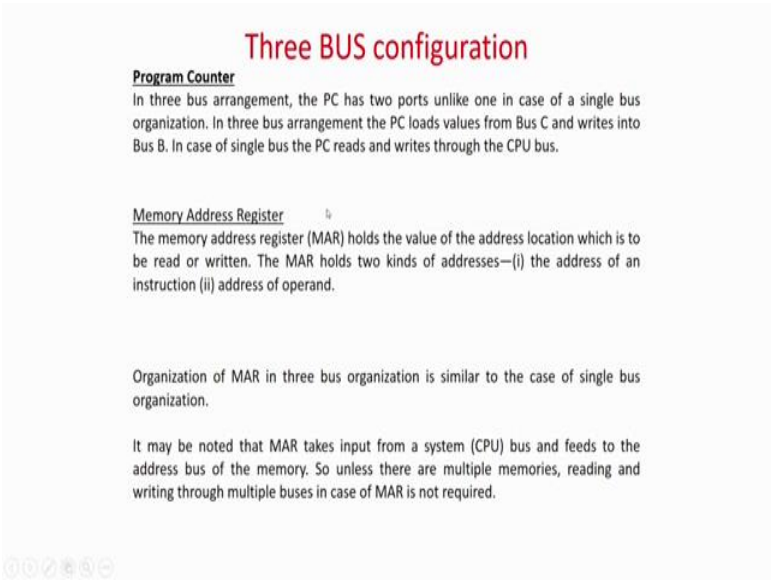
(Refer Slide Time: 26:47)



Because if there is a multiple ports over here; so in fact, in this and they connect to different bus to different lines to the address. So, you should have multiple memory blocks memory 1, memory 2.

So, as we are assuming single memory block, they are not going to help you. So, we are having a single port two port basically MAR which takes the value from memory from bus B and directs through the address lines of the memory. But in fact, you instead of bus B you can also take the input from A that you can easily do.

(Refer Slide Time: 27:15)



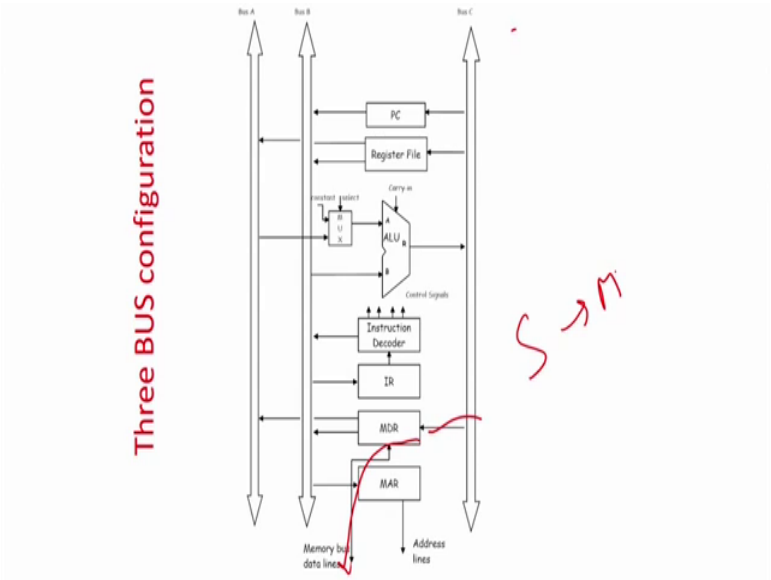
So, that what is says. So, it is made in bold it says that this one is going to change, if you are assuming a 3 bus architecture. Memory address register is similar and does not require any change. So, to explain it emphasize on that I have not made it bold sorry.

Next memory data register. So, let us look at it. So, this is a MDR Memory Data Register. So, memory data register basically I have put multiple ports over here, because if you consider a single bus architecture it takes some data in and out basically it can take this is the case, if you assume that there is a single bus you have to take a bidirectional port and of course, these two buses are not there. So, what happens? It takes input output from the memory bi directionally, and also it connects to a single line. So, that is what is the architecture. So, it's a two port architecture, and both the directions are bidirectional.

But in multiple port architecture, we are not giving a bidirectional bus here, but we are writing to two different lines that is the A and B. So, the memory data or memory instruction can go to two different places. In fact, it is if it is an instruction generally directly goes to the instruction register, there is no point in having two lines, but mainly we are handling also data in the same memory component. So, if the memory data register has any data. So, it can directly go to two different memory two different locations or two different registers or it can go to a register, it can also go to an ALU or two different locations basically simultaneously because it feeds bus A and B together and it reads from bus C and basically this is the bidirectional bus.

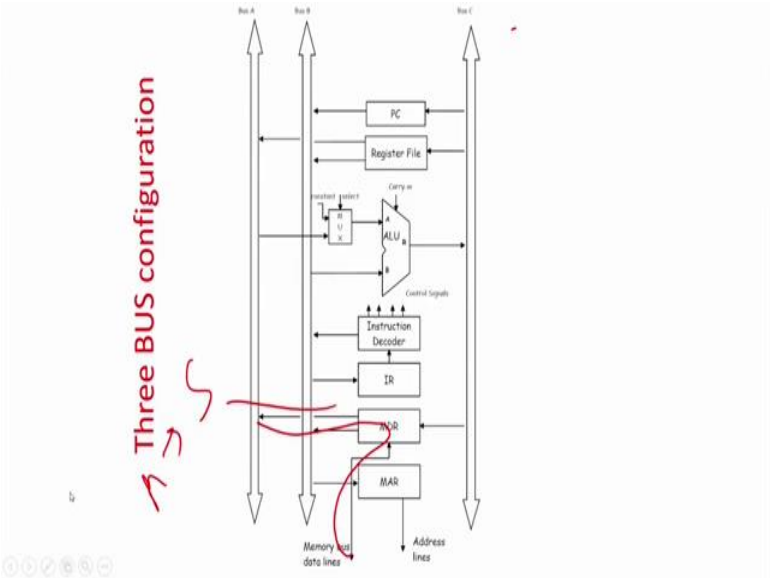
So, for example, if you want to read from the memory. So, it will go over here and then it will go to this bus and this bus. If you want to write from the memory or if you want to write to the memory basically you will take the data from here and it will go here.

(Refer Slide Time: 29:06)



That is a system to memory and then if you are telling the other way around, that is from memory to system.

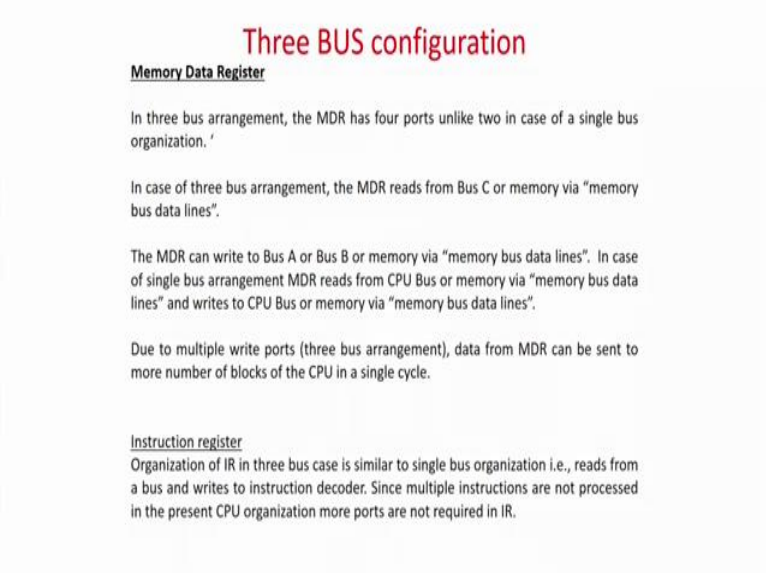
(Refer Slide Time: 29:13)



So, this is your system and then from memory sorry then it will memory it will come from here and it will that is from the memory to system, system means here you are having two buses over here.

So, that is the change that if from 2 ports now it has become 4 ports and you have to observe that this is a now a unidirectional bus not a bidirectional bus because output is this direction, output is this direction input is this direction from the MDR, but in a single bus system which we have to generally make it a bidirectional port because these two port lines are because A and B these two lines are not available, that is these two lines are not available actually.

(Refer Slide Time: 29:49)

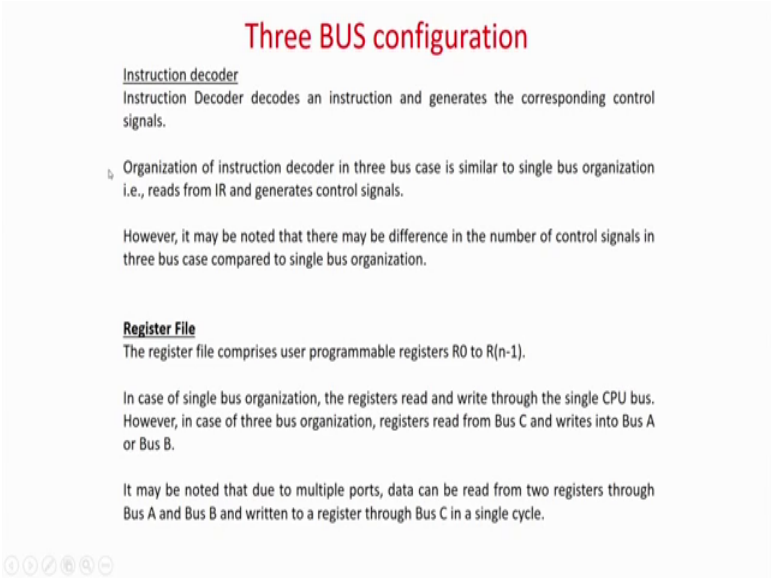


So, MDR has a four port block. So, if you look at it memory data register now has two ports. So, it can write two bus A or B or memory data lines and basically all the differences has been mentioned over you can read the slides. And instruction register as I told you so, basically if you look at the figure. So, instruction register we have a we have single instruction, that is you take the data from the data means it is the instruction from the memory is taken from the memory data register via it will go this.

So, if you want to read an instruction it will come from the MDR, basically it will come from the MDR and it will go to the instruction register. This will be basically the path it will come from memory to MDR from MDR it will directly go over here. Basically and we are as you are telling that we are not in implementing multiple instruction case. So, the instruction register is

very similar compared to a single bus architecture ok. So, in this case again the instruction register is very similar to the single bus architecture.

(Refer Slide Time: 30:48)



Now, instruction decoder I just need not elaborate here, the instruction decoder is also a very similar stuff because you are considering a single instruction. So, single instruction will be taken in the instruction register, it will go to the decoder and signals will be generated. Of course, the internals of the instruction decoder will change because here the outputs of the instruction decoder will be different because of course, the control signals here we will now we will have to control three buses instead in old case it was just controlling one bus.

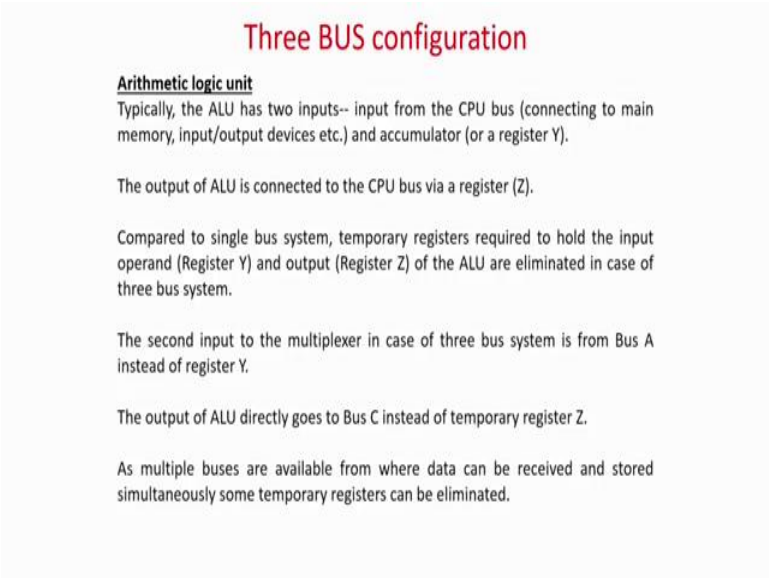
So, in that case the instruction decoder internals will change, but the input output ports or the I/O behavior of the instruction decoder will be similar. Because the instruction decoder, decodes an instruction and generate the corresponding control signals. So, in this case multiple number of control signals will be generated and it will be feed to different parts, but basically if you look at it that if you look at this part not much we will change, because it is going to take the data from the instruction register and it will generate multiple instructions and instruction values or control signal values basically.

So in fact, the control signals of course, will be different because there are different buses. So, there will be different input output configurations, but the overall topology will look very similar. That it will take some data, instruction from the instruction decoder and generate the corresponding inst corresponding control signals.

So, therefore, they say that not nothing much changes in instruction decoder. Of course, the register file registers have lot of changes. If you look at it again let us go to the figure sorry, if you look at the figure the register changes because in the old case there is a single bus like say bus C, and the registers file have a bidirectional bus. It reads and writes from the same bus, but in this case number of ports are increased because the registers are writing to bus A and B simultaneously. So, you can dump the values in two different locations simultaneously and it can, but and the input comes from a register.

So in fact, the number of ports have increased from 1 to 3 and secondly, there is no bidirectional bus requirement here because in this case it reads from port this port and dumps the value the output port. So, simultaneously you can read from here and the old value can go over here like a shift register kind of a thing, you read from this port and write it to this variable. So, this is the slight change in the architectural register files.

(Refer Slide Time: 33:01)



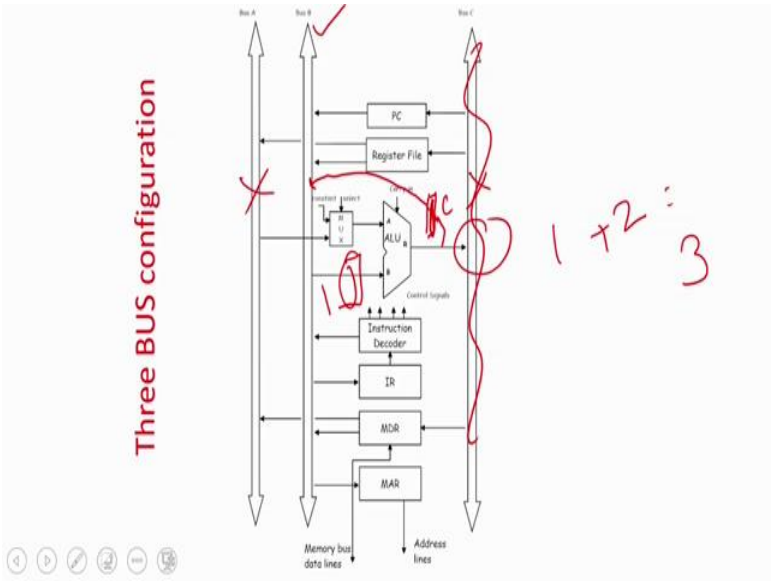
Arithmetic logic units, that is again the arithmetic logic unit is similar it comprises of an adder, subtractor, multiplier, comparator or all the logical instructions that can be executed that hardware is present, but what changes is there is no requirement of any kind of a temporary register. You can read in the slide and then we will basically look at why there is no requirement of any this slide, why there is no requirement of any temporary registers let us again look back here.

So, if you just see a single bus architecture; in single bus architecture what happens? This is the only bus available or let us assume that B is the only bus available A and C are not there. So, what will happen? So, first some data if it has to come. So, let us assume that this bus is also not there this bus is not there. So, of course, this one will have to again go back here with some multiplexing arrangement.

So, what is going to happen? So, there is only one bus that is B. So, first operand actually will come over here and say it will be stored in a temporary register, because there is only one bus which will carry the data. So, this is data 1 which is actually stored in a temporary register. Second one it can come over here and multiplexing I am not taking the constant. So, this one will be the data operand two will be directly fed over here.

So, first step you have to take a temporary variable, temporary register over here store variable number one because there is only a single bus. Next stage you connect this bus or from data from some register data and you can connect it to the ALU via mux done.

(Refer Slide Time: 34:32)



Then here we are going to have operand 1 plus operand 2 the value will be equal to say 3 and taking the value 1 and 2. So, value three or operand output 3 will be ready, it will be waiting over here see that is A+B, C or whatever the output is there.

But this bus is not available over here. So, it is waiting over here. So, why it has to wait again output you have to have a temporary register that will be Y we generally call it the register Y.

So, after this operation has been done the temporary variable or the temporary register is going to store the value of C, that is three in this case then in the third stage basically this output variable to input register temporary register C will be dumped to the single bus and it will go to the corresponding places and place basically. So, it's a multiple stage process, but in a multiple bus architecture it is very simple. So, one line is connected from the first operand, second line is connected the second operand second bus and output is going to a third bus.

So, there is no requirement of any temporary registers as such compared to a single bus architecture so, but basic ALU structure remains same that is not much changes over here and input output ports in that terms is also similar, but only thing is there is no requirement of any temporary register. So, that is what is the in a three bus configuration, the multiple registers, temporary registers job is minimized.

So, that is why we are explaining this whole unit using a three bus architecture. If you are using a two bus architecture that flavor is slightly gone because generally ALU means A+B and the output has to go to C. So, three bus architecture actually makes this things very complete from 4 bus, 5 bus things are more sophisticated and not that simple to exam explain and if we are having a two bus architecture these disadvantages cannot be illustrated.

So, therefore, we are considering a three bus architecture for explanation. Multiplexer and constant need not to say. So, this multiplexer and constant if you look at it. So, this multiplexer and constant, so these are same because the job is just to take an operand or the constant if you want to increase the value of program counter. So, there is no change in that. So, therefore, if you look at it. So, what changes basically? Configuration changes our program counter, memory data register, registers and you know arithmetic logic unit in terms of temporary registers. So, these changes are happening because there are more buses into picture.