

Computer Organization and Architecture: A Pedagogical Aspect.

Prof. Jatindra Kr. Deka

Dr. Santosh Biswas

Dr. Arnab Sarkar

Department of Computer Science & Engineering

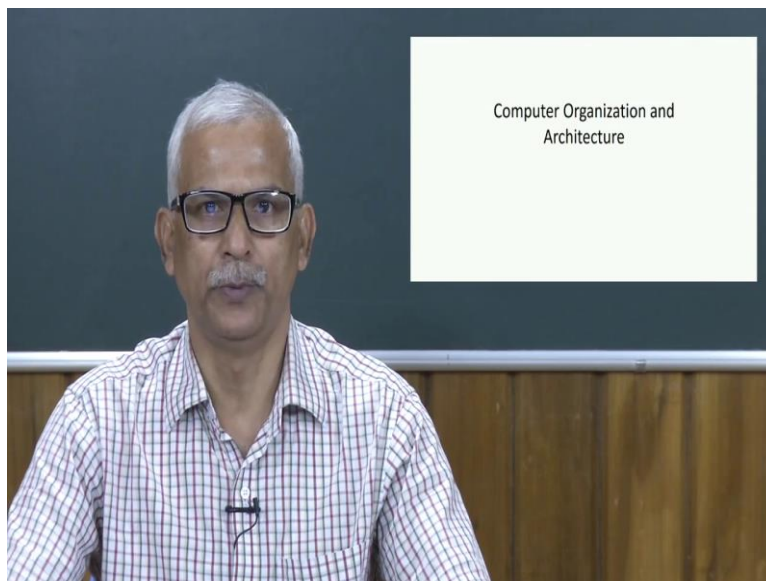
Indian Institute of Technology, Guwahati

Lecture – 06

Execution of Program and Programming Languages

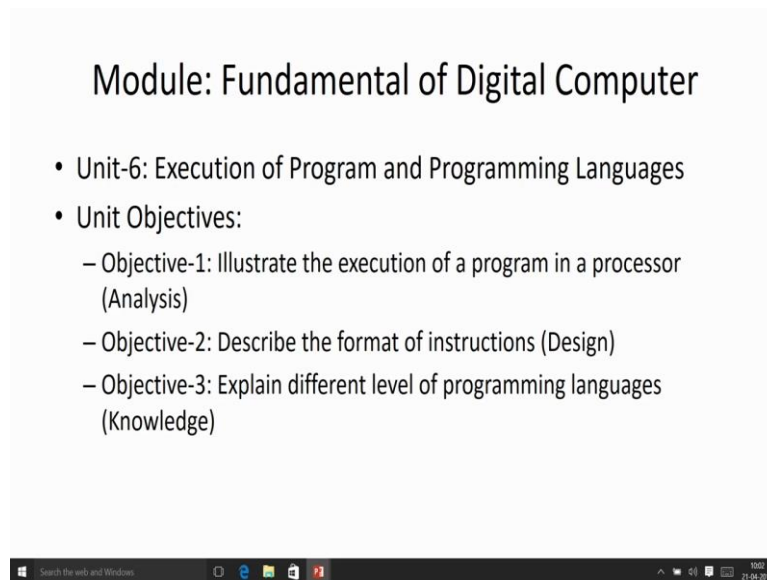
Hello everybody.

(Refer Slide Time: 00:28)



Welcome back to the online course on Computer Organization and Architecture. We are in the Module Fundamentals of Digital Computers and today we are going to discuss Unit 6. And this Unit 6 is related to execution of program and programming languages. Ok like as usual now we are going to define all Objective. What are the Objective of this particular module and after completion of this particular unit we are going to achieve those particular Objective.

(Refer Slide Time: 00:48)



Module: Fundamental of Digital Computer

- Unit-6: Execution of Program and Programming Languages
- Unit Objectives:
 - Objective-1: Illustrate the execution of a program in a processor (Analysis)
 - Objective-2: Describe the format of instructions (Design)
 - Objective-3: Explain different level of programming languages (Knowledge)

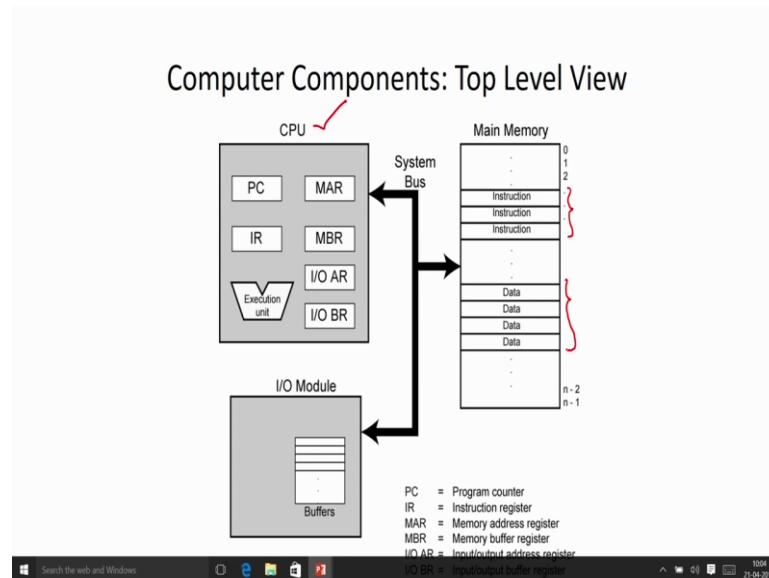
So, the Unit Objective of this particular unit is. Objective 1 we have to define it like that illustrate the execution of a program in a processor. So this is in Analysis level. That means, once you get a program then you will able to analyse exactly what task is performed by this particular program and how it is going to execute in a computer.

Objective 2 describe the format of Instruction. So, program is nothing but a set of instruction which will be executed in sequence. So, we are going to see; what is the format of instruction that we are having in the Instruction set of the processor. It is slightly in the design level in the higher level. Once you know the principal then you will be able to design an Instruction set for a processor.

Objective 3 explain different level of programming languages. So in a knowledge level just we are going to give information what are the different kinds of programming language we have

in for computer programming. So, it is in the knowledge level just we will give the brief idea about it.

(Refer Slide Time: 02:01)



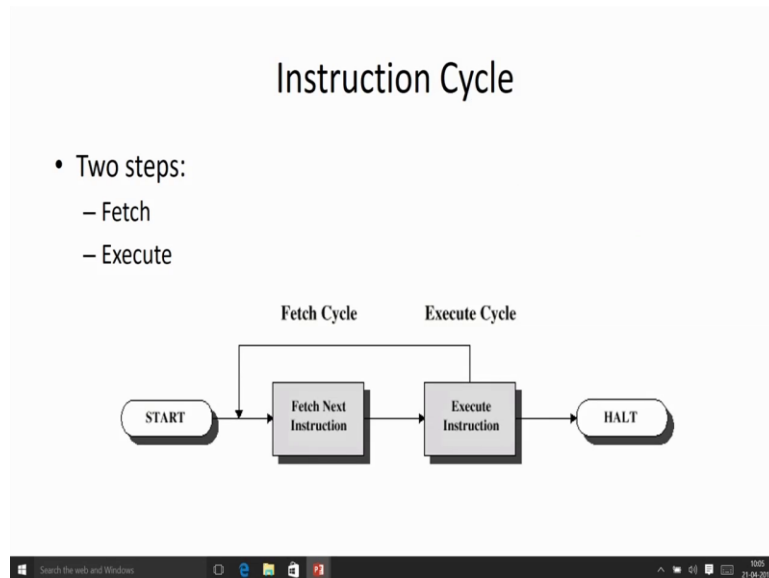
Now, in this particular module we have discussed about the working principle of Computers and the basic Components of the Computer. In the top level view of Computer Components are coming like that the main Component of a Computer is the processor, which is your Central Processor Unit. So, this CPU is going to perform our task depending on the program that we are going to execute.

But, the processor cannot work alone. So, we have to connect the Main Memory which is the storage device because processor works on von Neumann stored program principle. So we have to store our program and data in the Memory. So this is the Main Memory component. Here we are having the program just we are writing as Instruction and in some Memory location we have our data. So, processor works with the contents of the Main Memory, but how to take the information to the Main Memory and how we are going to get the result to the users. For that we need some I/O devices Input Output devices.

So, to control the Input Output devices we have this particular I/O Module. So the Main Memory and I/O Module will be connected to the processor through this particular system bus. Already I think in my last lecture we have mentioned that system bus is having 3 component

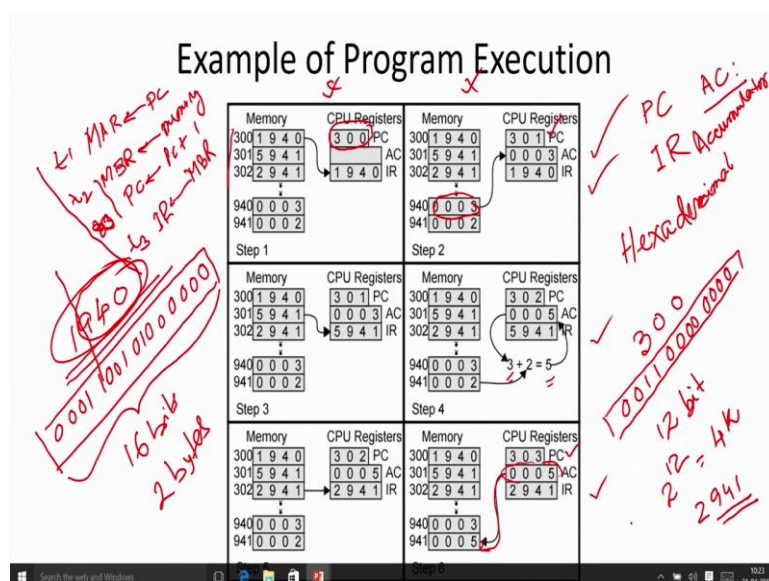
address bus, data bus and control bus. Now, in this particular frame work now we are going to see how we are going to executive a Computer program.

(Refer Slide Time: 03:34)



So, we know that while going to execute an Instruction that Instruction Cycle is having basically 2 phase mainly 2 phase. One is your Fetch, second one is Execute and we have seen in the Fetch Cycle what we are going to do we are going to Fetch the Instruction from the Memory and going to bring it to the processor and once we get the Instruction our job is to execute that particular Instruction inside the processor.

(Refer Slide Time: 04:05)



Now, here I am giving an example of a program. Here I have just mentioned or presented something over here in some tabular form. So, if you look into it what you are going to get over

here some numbers only and along with that we are talking about Memory and CPU Registers. So, basically this CPU Registers is a part of my processor. So inside the processor we are having storage element these are the Registers and it is going to take the information from this particular Memory. So, it is works on stored program principle. So, if you simply look into those particle figures I think it is difficult to understand or difficult to figure out what we are going to do. Now, we are going to see basically what does it means again I want to mention that whatever numbers we are writing over here all are in the hexadecimal notation.

So that means, in base 16 system. So, here you just see that we are having that Memory we are talking about 300, 301 and 302. These are basically the location of the Memory and 1940 these are nothing but the contents of that particular Memory location. So, 300 if this is 300 in hexadecimal, then what is the binary representation I know that this is your 001100000000. So, this is the exact address representation of the Memory location and this is basically in hexadecimal 300, but in binary representation 001100000000. So, it is having total 12 bit.

So, the address size of this particular Memory Unit is your 12 bits. And we can now access those particular Memory location. So this 12 bits basically if I am having an address bus of size 12 bit. Then what is the maximum Memory that we can address over here? This is your 2^{12} . So that means, 4K Memory location we are having. So, $2^2 = 4$ and 2^{10} is your K. So, total we are having 4K Memory location.

And what is the size of this particular Memory location? How many bits we can store inside this particular Memory location? You just see one of the content is your 1940. So, this is in hexadecimal. What is the binary representation 0001100101000000. So, this is the information that we have in the Memory location 300 in hexadecimal.

So, this is basically we are having total 16 bits. Basically it is your 2 bytes; that means, in every Memory location we can store 2 bytes of information. Now what this 1940 represents? Now we have to see because by just looking into it we cannot understand anything, but if we are going to interpret it properly then we are going to get some meaningful information. Now, here in a tabular form we are having 3 rows. Basically these 3 rows means we are going to execute 3 Instruction and in every row we are having 2 columns. So, these 2 columns indicate basically the execution phase of this particular Instruction.

So, first one is the Fetch phase and second one is the Execution phase. Now, when we are going to look into it then in CPU Register we are having 3 Register *PC*, *AC* and *IR*. I think already I have mentioned about *PC* which is nothing, but the program counter it is going to keep the address of the next Instruction that we are going to execute. We know *IR* which is your Instruction Register after fetching the Instruction we are going to put it into the Instruction Register, along with that we are having one more Register here which is your *AC*. *AC* stands for Accumulator. So, it is a working Register where we are going to keep our information. Now

just see now if I look into it you just see that we are storing our Instruction in Memory location 300, 301 and 302; that means, we must start our execution from the Memory location 300.

So, in that particular case in 300 this 300 address of the Memory location we are storing in this particular program counter. Now program counter knows which Instruction we need to Fetch or from which Memory location we need to fetch because if you remember this execution of fetch Instruction then what you are going to do we are going to put the content of *PC* in *MAR*. We are not showing this *MAR* and *MBR*. Then what we are going to do, in *MBR* we are going to get the contents of Memory location. In this particular case the Memory location is 300 and along with that we are going to increase the value of program counter. So, $PC + 1$. So, if this is my time cycle t_1 , this is my time cycle t_2 , this is my time cycle t_3 . So, this both will be done in t_2 and in time cycle t_3 what we are doing we are transferring the information from *MBR* to *IR*.

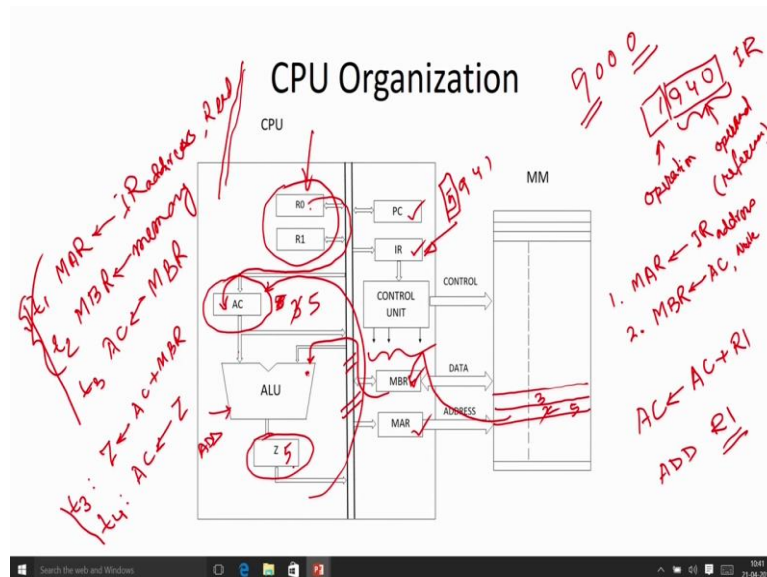
So, you just see we are going to fetch the information. So after fetching what we are going to do we are bringing this particular information the contents of this particular Memory location 300 to the processor and we are placing it in the *IR*. So this is the fetch phase is over now. We are fetching it. After fetching it, now what will happen? We have to increment the values of the program counter. So, that we will be knowing from which Memory location we are going to fetch the next Instruction. So, that's why when we complete this particular fetch then we are increasing the value of the program counter. So, now, value of program counter become 301 so that means, we are fetching the Instruction from Memory location 300, after completion of the execution we have to fetch the Instruction from 301.

Now, what does this Instruction mean 1940? We don't know. We have to see now what exactly we are going to do. Now it says that after performing this particular Instruction then what will happen the 3 is loaded into this particular Register Accumulator; that means, we are bringing some information from our Memory location to a working Register and in this particular case working Register is Accumulator.

Now, from this particular Memory location 940 we are bringing the information and keeping in into the Register 300. Now how I will be knowing that we have to bring it from Memory location 940 that 3 is there and we are bringing it over here now we have to bring it from 940. So, in that particular case you just see the contents what we have in your Memory 1940. So in 1940 now we can part into 2 different parts ok 1 and 940. That means, this address of the Memory location is available in the Instruction itself. So, in the Instruction after fetching the Instruction we got the information from which we have to take the information. So we have to take the information from Memory location 940 and after bringing the Instruction what we need to do we are going to put it into the Register Accumulator; that means, if I am going to see this things what is the effect of this particular thing so I am going to write in the next slide,

but now the effect we have to know the internal Organization of this particular processor also then only we will be knowing.

(Refer Slide Time: 12:18)



So, if you see that what is the CPU Organization in that particular case we are having this Register *PC*, *IR*, *MAR* and *MBR* along with that we may have some working Register may be *R0*, *R1*, *R2*, *R3* like that and along with that we are having an Register called Accumulator. This is a special Register generally we say where we are going to accumulate our result. So, that's why we are going to say it is an Accumulator and the Organization that we are having over here is known as your Accumulator based processor so this processor is based on Accumulator. Now what is the Accumulator? You just see that we are having the ALU. One of the input is coming from the Accumulator and second input is coming from some maybe some Register or may be Memory and is coming as a second input and after performing the operation we are storing the result in some Register *Z* which is a temporary Register. It is not visible to the users and after from *Z* we are storing the result in Accumulator so; that means, what will happen the effect of this things is your Accumulator is nothing but Accumulator + *R1*. Maybe from Register *R1* we are going to take this thing. So, in Accumulator based Register processor what generally we used to do. We used the Register Accumulator to accumulate our result and after performing any operation result will go to the Accumulator.

So, one of the Register is implicit over here so; that means, to perform this operation I can simply write `ADD R1`. In that particular case what will happen take the contents from the Register *R1* now add it to some numbers to which number we are going to add whatever number we are having in a Accumulator we are going to add this number *R1* to that particular Accumulator and storing the result back into Accumulator. This is the effect. So, we have to specify only one operand over here. This is your Accumulator. Now, you just see what is

happening over here with 0001 we are just so, in 940 we are having 0003 and we are storing it over here.

After fetching this Instruction, 1940 we are fetching it. So, what is this particular 1940 means it is having 2 parts. One part is saying this is the operation that we are going to perform and this is going to give the operand. This is not directly operand. I can say that reference of the operand. And what we doing over here; that means, after fetching the Instruction we come to know the address of the operand.

So, you have to fetch the data from that particular operand. So in first clock cycle what we are going to do we are going to do the *MAR* is equal to address part of the *IR*. Now in *IR*, I am having this particular contents. So, this is the address part and we are going to put it into the *IR* and along with that after that we will issue the read signal so that means, we are going to read it what we are going to get in *MBR* we are going to get from Memory.

So, this is in time step t_1 and in time step t_2 we are getting the information from the Memory and in time step t_3 what we are going to do Accumulator is equal to *MBR*. So these are the 3 steps that we need to execute this particular Instruction. So, what we are getting basically now first we are fetching it. These are the 3 clock steps that we need to fetch the Instruction. After fetching the Instruction we have to update the contents of this particular program counter and we are loading the information Instruction in Instruction Register. When we go to the fetch execution cycle. So, these are the 3 steps we are performing. First, we are getting the address of my data then we have bringing it to the *MBR* and from *MBR* we are transferring it to the Accumulator.

So, this is the way we can say that now Accumulator is getting it over here. So this is the way we are going to do it. So in this particular case we are having this particular Instruction. After completion of this particular Instruction 1, now we are going to execute the Instruction 2. So, in the similar fashion what we are doing we are fetching the Instruction from Memory location 301 because this is the contents of our program counter bringing it to the 941. Now, again it is having 2 part 5 and 941. 5 is going to indicate the operation that we are going to perform and 941 is going to give me the reference to the data or operand what operand we have going to give.

.So that means, in 941 we are having this particular operand 2. Now what is the effect say during execution, you see what it is happening it is showing the error. So, whatever we have in the Accumulator, initially in Accumulator we have 3 now whatever we are getting from the Memory location 941 we are going to add it to the contents of the Accumulator and what about the result we are getting $3 + 2$ we are going to put it in 5 so that means, you can say that initially my Accumulator is having 5 so sorry accumulator is having value 3. So in Memory location I am having 3 and 2 and I suppose 3 and 2.

Now, what this Instruction is doing, it brings this particular 2 to *MBR*. From *MBR* it is going out and coming as an input to the ALU. So, now one input is 2, other input is 3. Now in Instruction Register we are having the Instruction which is the code is your 5941. 5941 so this

5 indicates what operation we are going to perform. We are going to perform the contents of the Accumulator and going to add the value that is bringing from this particular Memory and we are going to get this information in this particular result in temporary Register 5. And from this particular temporary Register we are going to transfer it to the Accumulator. So that now Accumulator value becomes 5.

Ok so, now according to that what will happen? Now the execution step of this particular Instruction will also be similar. First two step it is similar. Once I am going to have this things my data in the *MBR*, then t3 is nothing but it will be your $Z = \text{Accumulator} + \text{MBR}$. Because one of the input to the ALU is your Accumulator and second input we are transferring it from *MBR* to your second input. It is coming through this particular internal data. It is transferred to the internal data bus or internal bus of the processor and from bus it is coming to this thing. So, this result we are storing in this particle temporary Register because directly I cannot give it to this particular bus to transfer it to the Accumulator. Because then there will be data conflict already I am having an input data along with that I cannot give the output data to the bus. So, they are at the conflict of data.

So, in t3 we are going to have $Z = AC + MBR$. And t4 what we are going to do we are going to transfer the information from *Z* to *AC*. So to execute this particular second Instruction now we need 4 clock cycles. First two will come as it is and third and 4th will come like that they are equal to $AC + MBR$ and $AC = Z$. So this is the completion of the second Instruction. Now after completion of the second Instruction, now we have to go for the next Instruction. The program counter is having 302; that means, we have to fetch the Instruction from 3002. We are fetching it we are getting 2941. So, this is the contents of my Instruction Register.

So, after fetching we will go to the execution phase. So what is the contents of now *PC* it is 3003. Now we are updating the values of the program counter. So, that next Instruction can be fetched from Memory location 3003. Now what this Instruction is doing that 2941. So again in 2941, 941 is the address reference of my operand and 2 is the operation. Now what operation we are performing with respect to this particular 2, you just see whatever value we have in the Accumulator we are transferring it back to the Memory location 941. .

So, before execution of this particular Instruction contents of the Memory location 941 is your 0002, but after completion of the Instruction it is becoming 5. So, this is the way we are going to execute our Instruction.

So, now you just see what we are doing first we are getting one data and putting it in our Accumulator in second Instruction we are taking the one more data and adding it to the contents of the Accumulator and store back the result in the Accumulator. And in your third Instruction what we are doing we are storing the information from Accumulator to this particular Memory location.

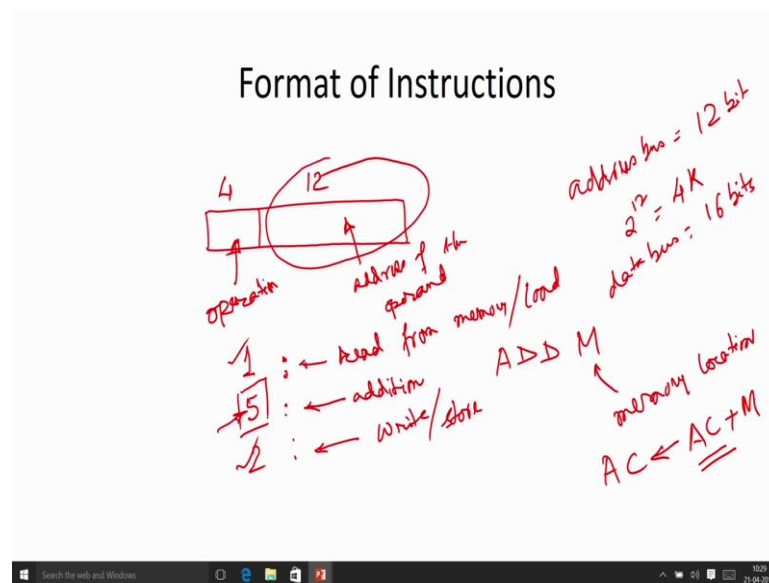
So, for this one also we can say that first we are executing it oh sorry first we are fetching it. These are the 3 clock step. This is required to fetch it. Now what will be the execution phase now over here, now that first step is your *MAR*. So for the execution of the third Instruction I

can say that I am going to put the information into the *MAR* address basically this is basically address part of the *IR*. This is the *IR* this is the address part we are putting it *MAR*.

Now, we are identifying this particular Memory location now what we need to do in second clock step we are having some value in your Accumulator. So, from Accumulator we are going to bring it to the *MBR* and we are going to give the write signal. So when we are going to give the write signal then what will happen whatever value we are having in the *MBR* that will be written in this particular Memory location with this particular address. So, this value will become now 5. So this is the execution phase of this particular Instruction. So you just see now we are having some numbers. Now if we can properly interpret it then we can get the effect what exactly we are doing and to get the effect at least we must know some internal details of the processor also.

So, by looking into this particular scenario we have just say that this may be the probable Organization of the processor it is Accumulator based processor and in the particular processor we can carry out this particular Instruction. Now in this particular example.

(Refer Slide Time: 24:01)



Now, what we have seen that say address bus is your of 12 bit. So, we can go up to 2^{12} Memory location which is your 4K Memory location. What is the size of data bus? The size of data bus is your 16 bits. So, we are going to work with 16 bit of information. So, whatever information we are getting 16 bit. Now, how we are going to interpret it if it is an Instruction already we

have seen that my Instruction is having a 16 bit of information. So, we are dividing into 2 parts. This is your 4 bit and this is your 12 bit total 16 bit.

So, this 4 bits are going to indicate my operation what operation we are going to perform and this 12 bit is going to give me the address of the operand and. Secondly, this is a Accumulator based processor. So, we need to specify one operand second operand is your implicit.

Now, in this particular example what are the operation we have done 1, 5 and 2; that means, opcode is 1, opcode is 5 and opcode is 2; that means, in 5 this is basically read from Memory. Second one is addition operation. So, what is the addition operation basically we are doing basically we are performing ADD M. Where M is your Memory location. So, we are giving the address of the Memory location we are going to take the information from that particular Memory location and we are going to add it to Accumulator and going to store the result in Accumulator. So, effect is your Accumulator is equal to Accumulator plus contents of this particular Memory location.

So, the second operand is my implicit reference. We need not to mention categorically always. We are going to take the operand from one operand from Accumulator and second operand from the reference we are giving along with this particular Instruction. So, for this particular processor to interpret the information that Instruction we must know the Instruction format and by looking into the scenario we have come up with this particular Instruction format and the opcode 2 is basically nothing but we can say that this is your write operation.

So, we are going to write or store something to some Memory or some location. This is read from Memory or we can say read or loading something from Memory to the processor. So, general we use the term load also.

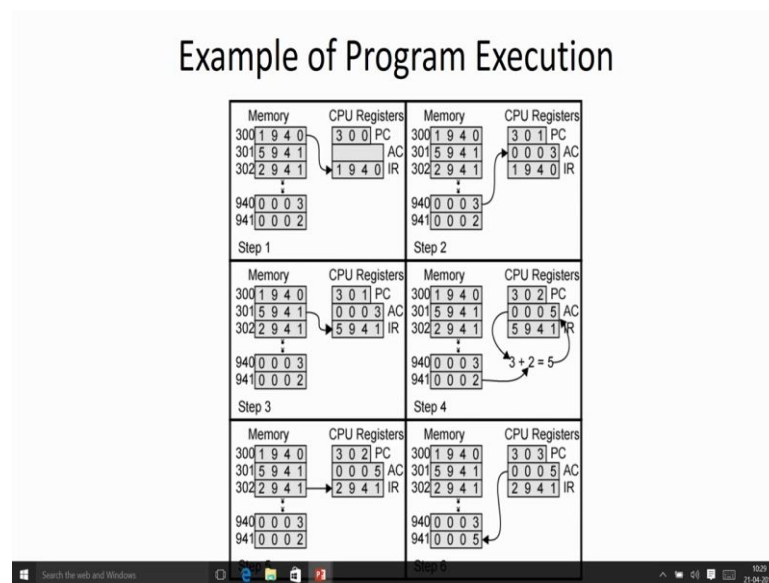
So, one Instruction is LOAD another Instruction is STORE and along with that we are having an Arithmetic operation Instruction which is going to use the ALU. Just see in first and third Instruction we are not using the ALU because this is simply bringing the data or storing the data, but in the second Instruction we are using the ALU where we are performing addition operation. Now this information we are getting it that now we have to perform the addition

operation by looking into this particular code 5. So here we are having that opcode. So instruction Register is having the opcode.

So, once we are getting this particular opcode that will be given to the control unit and control unit is going to generate those particular control signal and it will go to the appropriate location.

Now, one of the things will come here is an ALU, the control unit will generate this particular add signal then whatever input we are having in this ALU it will perform the addition operation and store it in this particular temporary Register Z. Now again I am showing this

(Refer Slide Time: 28:05)



particular slide. Now, I think after knowing the details how to interpret this particular information, what is the processor Organization. So, after knowing all those things now very well now we can interpret this information and we know what are the things that we are

performing inside the processor. Now, this information now we are going to give some meaning.

(Refer Slide Time: 28:30)

Machine Instruction

Machine	Instruction Format				Assembly
Instruction	Operation ✓	Address ✓			Code ✓
1940	0001	1001	0100	0000	LDA M ✓
5941	0101	1001	0100	0001	ADD M ✓
2941	0010	1001	0100	0001	STA M ✓

(LDA M) LOAD AC: Load the accumulator by the contents of memory location specified in the instruction

(ADD M) ADD AC: Add the contents of memory location specified in the instruction to accumulator and store the result in accumulator

(STA M) STORE AC: Store the contents of accumulator the memory location specified in the instruction

So, we are executing three Instructions 1940, 5941 and 2941. So in that particular case the Instruction 1940 we can say that it is having 2 part. One is your operation part and second one is the address part or reference part. So, this is 1940 the operation code is 1 along with that we are giving some reference address 5941 operation code is your 5 - 0101 and this is your address and 2941 in case of operation is 2 - 0010 and this is the address 941.

Now, I am giving some symbolic name to that, here I am saying that this load M (LDA M); that means, load the Accumulator from Memory location M, ADD M and store M (STA M). So, we are designing 3 Instruction also we are having 3 Instruction in this particular processor one of the Instruction is load M. So, what it says load the contents from the Memory location M and store the result in Accumulator. So, it is loading the loading some values to the Accumulator, second one is we are saying that ADD M.

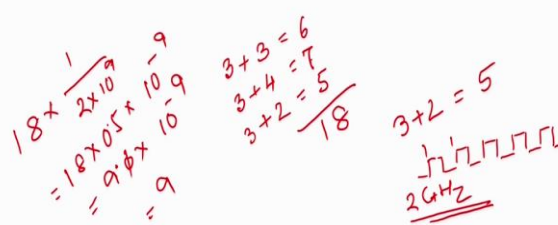
So, what it does basically. So take the information from Memory location M, add this to the contents of the Accumulator and store the result back into the Accumulator. So, this is the effect. So, this opcode is 5 and what is STA M? Store M, it is basically store the value of Accumulator to the Memory location M. So, we are having some value in the Register Accumulator inside a processor and we are going to store this information to some specified Memory location M which will be given as an input in the Instruction itself.

So, it is having the format opcode and Memory reference. So, this is the way that we are going to interpret our Instruction. So, here we are going to say that this is some code we are giving and we are going to say these are the assembly level code and these are the Instruction which

is combination of zero's and one we will say this is the basically machine level Instruction machine understand only this particular bit pattern zero's and one and nothing else.

(Refer Slide Time: 30:52)

Computer Program		
High Level Code	Assembly Code	Machine Code (HEX)
<u>Y = X + Y</u>	LDA X	1940
	ADD Y	5941
	STA Y	2941



The image shows handwritten calculations in red ink. On the left, a multiplication is shown: $18 \times 2 \times 10^9 = 18 \times 0.5 \times 10^9 = 9 \times 10^9 = 9$. In the middle, two addition problems are shown: $3+3=6$, $3+4=7$, and $3+2=5$ with a horizontal line under the result 5. On the right, another addition $3+2=5$ is shown, followed by a square wave diagram labeled '2GHz'.

Now, finally, what we are going to get say basic operation what we are going to do say $Y = X + Y$ because we are going to add 2 numbers $3 + 2$ and we are going to get result. So, when we are going to write a program in high level language may be if you are accustomed with C or some other high level language generally we used to define some variables and these variables are place holders. They can keep or store some values.

So, in that particular case in high level language we are going to write $Y = X + Y$. So, basically what happen we are taking the information from Memory location 940, taking the information from Memory location 941. Adding them together and storing the information in Memory location 941. That's why we are saying $Y = X + Y$. This is the operation we are going to perform, in high level we are going to write in this particular way.

Now, to perform this particular addition operation when we are going to execute in this particular processor we need 3 Instructions say we are going to perform one operation in the high level this is my requirement. For that, I am going to execute convert this particular program for the processor we have to use 3 Instruction ADD X oh sorry LDA X, ADD Y and STA Y. Ok so, these are the 3 Instruction that we are going to execute. If we are writing in this particular code generally we say these are the assembly code or we say this is the assembly level program and we say this is the high level program.

Now, again you just see that when I am talking about LDA X, ADD Y and STA Y. we have seen that the number of steps required. So, in fetch we need 3 step. In your execution somewhere we need 3 step, somewhere we need 4 steps and somewhere we need 2 step only. So, to execute those particular program if you see that first one is going to take 3 clock cycle plus 3 clock cycle, second one is going to take 3 clock cycle for fetch and 4 clock cycle for

execute and third Instruction is going to take 3 clock cycle for fetch and 2 clock cycle for execution because it works in a continuous running clock and it is having some predefined frequency depending on the processor.

So, you just see that now if you are having a computer and you say that your computer works on say 2 Gigahertz. So this is the operating frequency. Now you will be able to find out what is the time required to perform this particular operation. You just see we need total 6, 7 and 5 total 18 clock cycles. So, total time required will be your $18 \times \frac{1}{10^9}$. 2 Gigahertz is your frequency is your 2 Gigahertz, then this clock pulse is your one upon frequency. So, this many times you are going to have this is nothing, but $18 \times 0.5 \times 10^{-9}$. 18 times are 90. So, 90×10^{-9} ; that means, 90 what will be the unit second, then millisecond, microsecond, nanosecond. 10^{-9} is your nanosecond.

So, if processor is going to work in 2 Gigahertz, then we need 9 nanoseconds to carry out this particular operation. So, this is the way we will be knowing what is the time required to execute a particular Instruction in the processor. Now, already I have said now if this is the

(Refer Slide Time: 34:53)

Instruction Format

OP-CODE	Address
4 bit	12 bit

16

- Number of Instruction: $2^4 = 16$
- Number of address space: $2^{12} = 4096 = 4K$

~~4 bytes~~
2 bytes = 16 bits

Instruction format the total we have having 16 bit of information to represent one Instruction out of that 4 bit will go for the opcode and 12 bit will go for the address part. So, in that particular case what is the total number of Instruction that we can design this is your $2^4 = 16$; that means, we can design 16 different Instruction.

And what is the address space that how many Memory location we can refer in this particular processor, this is your 2^{12} which is your 4096 which is nothing, but 4K. So, 4K Memory

location we can refer in every Memory location we are having 2 bytes of information this is nothing but 16 bit. Ok so this is the information that we are having.

So, Memory space is your 4K Memory location, in every Memory location we can give 2 bytes of information and total number of Instruction that we will be able to design for this processor is your 16.

(Refer Slide Time: 35:50)

Instructions

Op-code	Instruction	Op-code	Instruction
0		8	
1	LDA M	9	
2	STA M	A	
3		B	
4		C	
5	ADD M	D	
6		E	
7		F	

So, since we can design 16 different Instruction. So, we are having 16 opcode. So, these opcode will vary from 0 to F. So, 0 means binary representation is all 0 and F means binary representation is all 1.

So, like that 0, 1, 2, 3 like that up to F we can design. So now, currently we have discussed about 3 Instructions load Accumulator, store Accumulator, load Accumulator from Memory, store Accumulator to Memory and add Accumulator or add Memory to the Accumulator. So,