

Computer Organization and Architecture: A Pedagogical Aspect.

Prof. Jatindra Kr. Deka

Dr. Santosh Biswas

Dr. Arnab Sarkar

Department of Computer Science & Engineering

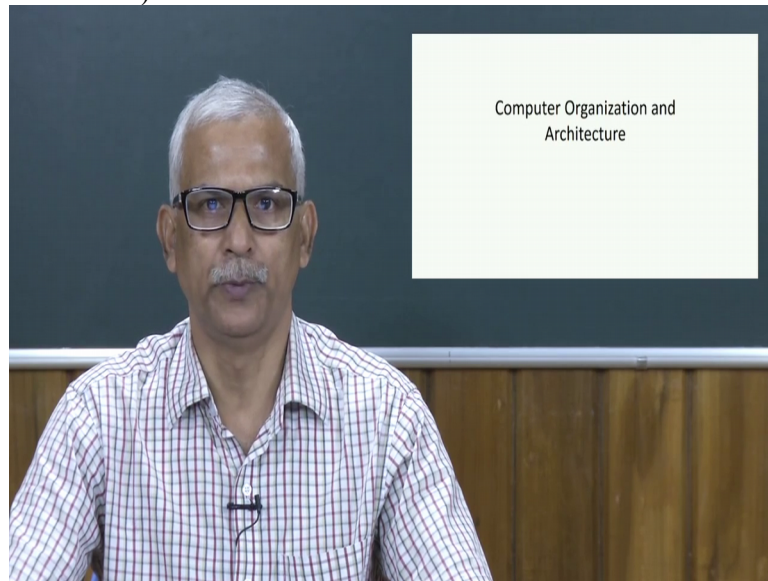
Indian Institute of Technology, Guwahati

Lecture – 06

Execution of Program and Programming Languages

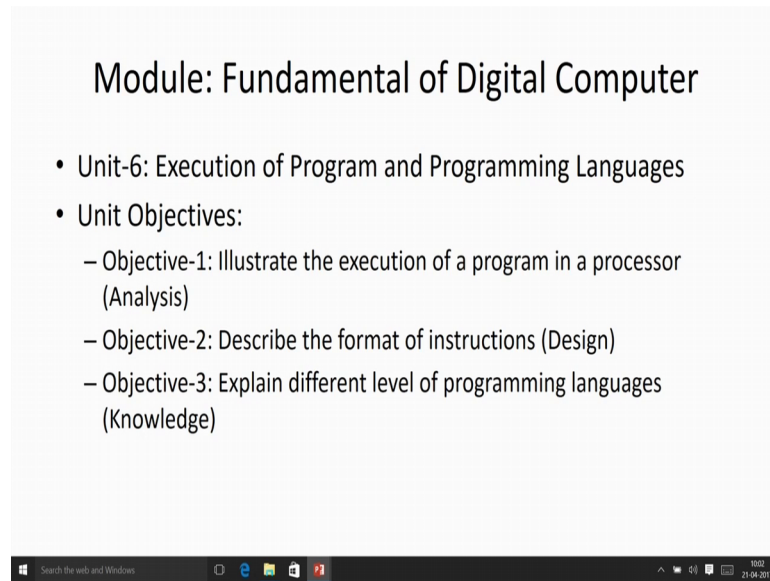
Hello everybody.

(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 like as usual.

(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)

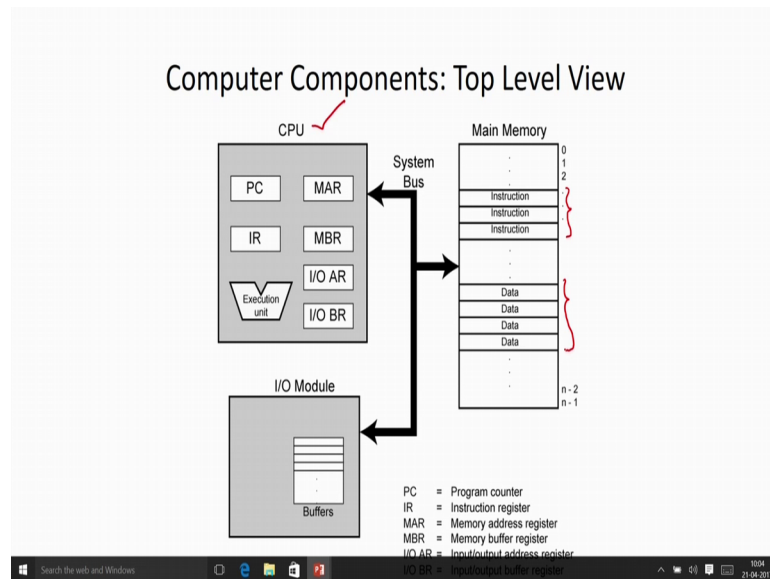
Now, you are going to define all Objective. What are the Objective of this particular module and after completion of this particular unit we are going to achieve those particular Objective.

So, the Unit Objective of this particular unit is. Objective 1 we have to it as define like that illustrate the execution of program in a processor. 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 a 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. In a knowledge level just you 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 you 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 a processor, which is your Central Processor Unit. So, this CPU is going to perform or task depending on the program that you are going to execute.

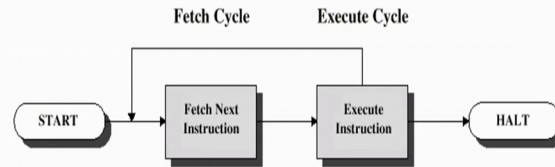
But, the processor cannot walk alone. So, we have to connect the Main Memory which is the storage device because processor works on von Neumann stored program principle. We have to store our program and data in the Memory. This is the Main Memory component. Here we are having the program just we are writing as Instruction and in some Memory location we have all data. So, processor works with the contents of the Main Memory, but how to take the information to the Main Memory and how you 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. 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)

Instruction Cycle

- Two steps:
 - Fetch
 - Execute



So, we know that while going to execute an Instruction. That Instruction Cycle is having basically 2 steps. Many 2 steps. 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 you get the Instruction or job is to execute that particular Instruction. So, we are going to Execute that particular Instruction inside a processor.

(Refer Slide Time: 04:05)

Example of Program Execution

Step	Memory	CPU Registers
Step 1	300 1 9 4 0 301 5 9 4 1 302 2 9 4 1 940 0 0 0 3 941 0 0 0 2	PC: 3 0 0 AC: 1 9 4 0 IR: 1 9 4 0
Step 2	300 1 9 4 0 301 5 9 4 1 302 2 9 4 1 940 0 0 0 3 941 0 0 0 2	PC: 3 0 1 AC: 0 0 0 3 IR: 1 9 4 0
Step 3	300 1 9 4 0 301 5 9 4 1 302 2 9 4 1 940 0 0 0 3 941 0 0 0 2	PC: 3 0 2 AC: 5 9 4 1 IR: 2 9 4 1
Step 4	300 1 9 4 0 301 5 9 4 1 302 2 9 4 1 940 0 0 0 3 941 0 0 0 2	PC: 3 0 3 AC: 0 0 0 5 IR: 2 9 4 1
Step 5	300 1 9 4 0 301 5 9 4 1 302 2 9 4 1 940 0 0 0 3 941 0 0 0 2	PC: 3 0 2 AC: 2 9 4 1 IR: 2 9 4 1

Handwritten notes:

- Left side:
 - 1. MAR ← PC
 - 2. MBR ← memory
 - 3. PC ← PC + 1
 - 4. IR ← MAR
 - IR: 1940
 - 1001 1001 010 0 000
 - 16 bit
 - 2 bytes
- Right side:
 - PC AC: IR Accumulator
 - Hexadecimal
 - 300
 - 1001 000 000 1
 - 12 bit
 - 2¹² = 4K
 - 2941

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. 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 awesome store programming (Refer Time 04:36). 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.

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 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. This 12 bits basically having address bus of side 12 bit. Then what is the maximum Memory that we can address over here? This is your 2 to the power 12 . That means, 4K Memory location we are having. So, 2 square is equal to 4 and 2 power 10 is your K. So, total you are having 4K Memory location.

And what is the size of this particular Memory location? How many bits you 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 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 you 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 an 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 you are going to look into it then in CPU Register we are having 3 Register PC, SC 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 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 all information. Just see now if I look into it you just say that we are storing all 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 this 300 address of the Memory location where storing in this particular program counter. Now program counters 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 you 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 plus 1. So, if this is my time second t1, this is my time second t2, this is my time second t3 sorry. So, this both will be done in t2 and in time second t3 what we are doing we are transferring the information from MBR to IR.

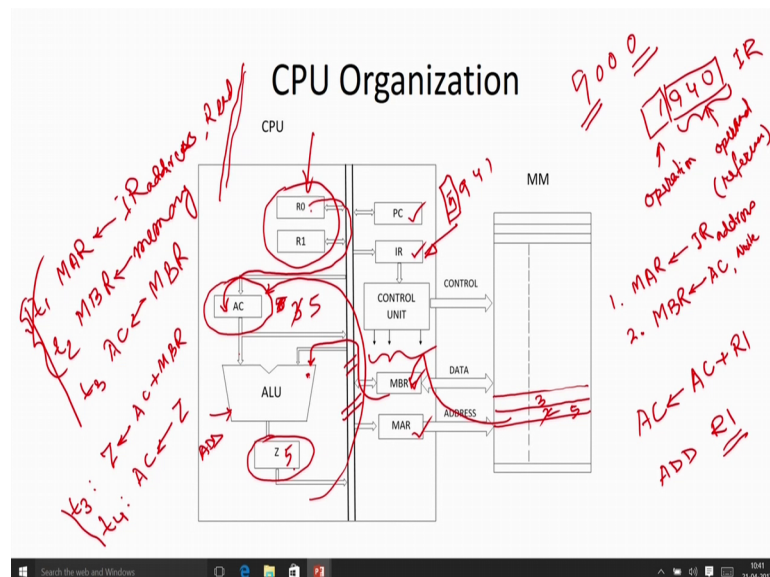
So, you just see we are going to fetch the information. After fetching what you are going to do we are bringing this particular information the contents of this particular Memory location 300 to the processor and we are placing it in the IR. This is the fetch phase is over now. We are fetching it. After fetching it, now what will happened we have to increment the values of the program counter. So, that will be knowing from which Memory location we are going to fetch the next Instruction. So, that is why when we complete this particular fetch that 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 locations 300. After completion of the exhibition you have to fetch the Instruction from 301.

Now, what does this Instruction mean 1940? we do not know. We have to see now. What exactly you 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 it into the Register 300. Now how I will be knowing that we have to bring it from Memory location 940 that 3 is in and bringing it over here now I have to bring it from 940. So, in that particular case you just see the contents what we have in your Memory 1940. In 1940 now we can part into 2 different parts.

1940 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 you have to take the information. You 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. I am going to write in the next slide, but to know the effect we have to know the internal Organization of this particular process also then only you 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 you 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 Registrar. General we say where we are going to accumulate our region. So, that is why we are going to say it is an Accumulator and the Organization that we are having over here is known as your Accumulator with best processor for 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 or 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 Accumulators so; that means, what will happen the effect of this things is your Accumulator is nothing, but Accumulator plus R1. Maybe from Register R1 having to take this thing. So, in Accumulator based Register processor what generally we use 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 have restoring 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. In first clock cycle what we are going to do we are going to do the MAR is equal to address part of IR. Now in IR, I am having this particle contents. So, this is the address part and we are going to put it into the IR and after that we will receive the read signal so; that means, you are going to read it what you are going to get in MBE we are going to get from Memory.

So, this is in time step t1 and in time step t2 we are getting the information from the Memory and in time step t3 what we are going to do Accumulator is equal to MBR. 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 you have object the contents of this particular program counter and we are loading the information Instruction in Instruction Register. When we will go to the fetch execution cycle. So, these are the 3 steps you 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 to the Accumulator.

So, this is the way we can say that now Accumulator is getting it over here. This is the way we are going to do it. 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 you have fetching the Instruction from Memory location 301 because this is the contents of a 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 have 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 plus 2 are going to put it in 5; that means, you can say that initially my Accumulator is having value 3. In Memory location I am having 3 and 2 and 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. 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.

So, now according to that what will happen. Now the execution step of this particular Instruction is also be similar. First 2 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 equal to your Accumulator plus MBR. Because one of the input to the ALU is your Accumulator and second input you 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 you are going to have Z equal to AC plus MBR. And t4 what we are going to do we are going to transfer information from Z to AC. To execute this particular second Instruction now we need 4 clock cycle. First 2 will come as it is and third and 4 will come like that there is equal to AC plus MBR and AC is equal to Z. This is the completion of the second Instruction. Now after completion of the second Instruction, now you have to go for the next Instruction. The program counter is having 302; that means, you have to fetch the Instruction from 302. We are fetching it we are getting 2941. So, this is the contents of my Instruction Register.

So, after fetching will go to the execution phase. 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. Again in 2941, 941 is the address of 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.

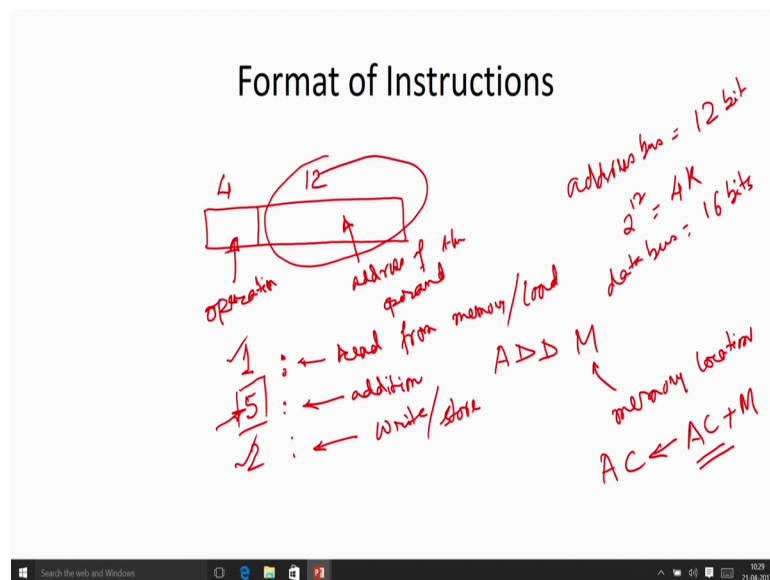
So, now you just see what you are doing first you are getting one data and putting it in our Accumulator in second Instructions we are taking the one more data and adding it to the contents of the Accumulator and stored back the result in the Accumulator. And in a third Instruction what you are doing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

So, for this one also we can say that 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 about here, now that first step is your MAR. 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. When you 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. This is the execution phase of this particular Instruction. You just see now we are having some numbers. Now if we can properly interpret it then we can get the effect what exactly you are doing and to get the effect at list 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 of the 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 to the power 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 you 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 base processor. So, we need to this is only one operand second operand is your implicit.

Now, in this particular example what are the operation we have done one 5 and two; that means, Op-code is 1, Op-code is 5 and Op-code 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. You are going to take the operand from one operand from Accumulator and second operand from the reference by 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 Op-code 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.

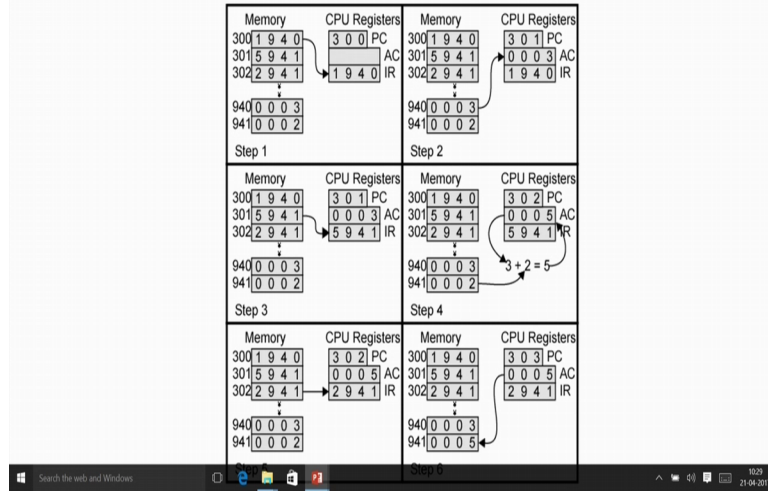
So, one Instruction is load another Instruction is stored and along with that we are having an Arithmetic operation Instruction which is going to use the ALU thus see you first and third Instruction you are not using the ALU because this is simply bringing the data or storing the data, but in the second Instruction we are using the ALU where you have perform in a 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. Here we are having a Op-code. Instruction Register having a Op-code.

So, once you are getting this particular Op-code 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 as an add that control unit will generate this particular add signal then whatever input we are having in this ALU it will perform the addition operation and stored in this particular temporary Register set. Now again I am showing this

(Refer Slide Time: 28:05)

Example of Program Execution



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 a processor. Now, this information we are going to

(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

give up some meanings. So, we are executing 3 Instruction 1940, 5941 and 2941. 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 for 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 50101 and this is your address and 2941 in case of operations is 20010 and this is the address 941.

Now, I am giving some symbolic name to that here I am saying that this load m; that means, load the Accumulator from Memory location M, ADD M and store 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 you 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 Op-code is 5 and what is stored M? It is basically stored 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 Op-code and Memory reference. So, this is the way we are going to interpret our Instruction. So, here we are going to said it this is some code we are giving and we are going to said these are the assembly level code and these are the Instruction which is combination of zero's and one 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)
Y = X + Y	LDA X ADD Y STA Y	1940 5941 2941

Handwritten notes:

$$18 \times \frac{1}{2 \times 10^9} = 18 \times 0.5 \times 10^{-9} = 9 \times 10^{-9} = 9$$

$$3 + 3 = 6$$

$$3 + 4 = 7$$

$$3 + 2 = 5$$

2GHz

Now, finally, what we are going to get say basic operation what we are going to do say Y is equal to X plus Y because we are going to add 2 numbers 3 plus 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 custom with c or some other high level language generally we use to define some variables and these variables are place holder. They can keep or store some values.

So, in that particular case in high level language we are going to write Y is equal to X plus 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 is why we are saying Y is equal to X plus 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 12, I am going to execute convert this particular program for the processor we have to use 3 Instruction LDA X, ADD Y and STA Y. So, these are the 3 Instruction that we are going to execute. If we are writing in this particular code generally say these are the assembly code or we said 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 step required. So, in fetch we need 3 step. In your execution some are in a 3 step some are in a 4 steps and some are in a 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 said it your computer works on say 2GigaHertz. This is the operating frequency. Now we will 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 set. So, total time required will be your 18 into one upon 2 to the power 10 to the power 9. 2GigaHertz. If 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 into 0.5 into 10 to the power minus 9. 18 fives are 90. So, 90 into 9 into 10 to the power minus 9; that means, 9 what will be the unit second, then millisecond, micro second, nanosecond. 10 to the power minus 9 is your nanosecond.

So, if processor is going to work in 2GigaHertz, then we need 9 nanosecond to carry out this particular operation. So, this is the way will be knowing what is the time require to execute an 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

- Number of Instruction: $2^4 = 16$
- Number of address space: $2^{12} = 4096 = 4K$
~~4096~~ $2 \text{ bytes} = 16 \text{ bits}$

Instruction format that total we having 16 bit of information to represent one Instruction out of that 4 bit will go for the Op-code 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 to the power 4 equal to 16; that means, you can design 16 different Instruction.

And what is the address space that how many Memory location we can referred in this particular processor this is your 2 to the power 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. This is the information that we 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 Instructions. So, we are having 16 Op-code. So, these Op-code 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 F we can design. Now, currently we have discussed about 3 Instructions say load Accumulator, store Accumulator, load Accumulator from Memory, store Accumulator to Memory and add Accumulator or add Memory to the Accumulator. So, these are the 3 Instructions we have designed and the code assign is your 1, 2 and 5. Other codes are now still available to me.

So, now we can design some more Instructions. Now, we are saying that already we have designed

(Refer Slide Time: 36:41).

Instructions

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

AC ← AC - M

these 3 Instruction. Now, we have seen that we are designing one more Instruction call SUB M. So, it means subtraction. So, what is this Instruction this is basically nothing, but Accumulator is equal to Accumulator minus contents of the Memory. You are going to give the Instruction format is same whatever we are going to design for all the Instruction it is going to follow this particular pattern.

So, now along with this 3, I am going to use one more code this is code 4. Which your subscription 4. Like that you can now add more and more Instruction now along with that now again I am going to give designing 4 more Instruction. This is a similar Instruction load store sub m, but my referring is different initially you are talking about the memories now we are talking about the Registers.

(Refer Slide Time: 37:34)

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

Handwritten notes on the slide include:

- 0001 (next to op-code 1)
- 0101 (next to op-code 5)
- LDA M → R ← 12 bit → A096
- AC ← AC + R7
- 8 (with a checkmark)
- 1001 (next to op-code 9)
- 1101 (next to op-code 5)
- 9000 (in a box)
- D007 (in a box)
- 8, R0, R1, R7 (vertical list)

So, now what will happen whatever operation we are having say here I am going to say that this is again a load operation this is your 1 is your 0001, what is your 9, 1001; that means, for load I am returning this particular 0001, but with that most significant bit 0 is going to say that it is a Memory reference and one will indicate that it is a Register reference. So, when it is 9 then I can give the reference of the Memory. So, in that particular case generally already I have mentioned that number of Registers is limited very less number Registers. So, if I am having say only 8 Registers that we are going to use say R0, R1 to R7 say these are the Registers.

Then what will happen in that particular case we do not require that 12 bit. So, these are basically do not gave, but I can keep all those things is 0 and along with that here I am going to give the Register number. So, if I say that this is your 9000 means it is going to refer to this particular Register R0. The value of the Register R0 will be loaded to the Accumulator.

So, similarly if my Op-code is your 9001 it is going to say that take the value of the Register R1 and bring it to the Accumulator. So, if you see the Organization, we are saying that we are having some general purpose Register now we are having R0, R1 like that we are saying 8 different Register. So, contents of this Register can be taken out from this Register and put it to the Accumulator with the help of this Instruction 9000. So, it is referring to the Register 0 and what is the Op-code says take his information and bring it to this Accumulator.

So, like that we can have the effect of those particular Instruction. Now, if I am going to write say 5 is your 0101 and what is your this things when I am going to have 5 this is your 1101. So, now, if my Instruction is like that this is your Op-code is your D and say it is your 0 07. Now, what it basically is the it is the edition of bar.

So, these Instruction is basically nothing, but Accumulator is equal to Accumulator plus R7 they are referring this particular Register 7 or R7 . So, the effect of these Instructions like that. So, similarly can now interpret data also. So, we are having similar Instruction one is referring to the Memory location and second one is referring to the Register.

So, in this particular case you just see that how many different kind of combination you have? You are going to have only 8 different combination. Because, that Registers values can go from 0 to 7 totally. So, for all those Instruction you are going to get 8 different variation maybe load R0, load R1 like that.

But what will happen in this particular case when I am going to talk about LDA M. So, M is going to take all the 12 bit address. So, this is your 2 to the power twelve; that means, 4096 Memory space so; that means, we are going to have 4096 variation of this particular LDA M. Because, we can get or you can gives any Memory location to take our information, but for LDA R we are going to get 8 variation only because you are having 8 general purpose Register.

So, similarly LDA R, STA R, SUB R, ADD R know you just see that 4 plus 4 we have design 8 Instruction still we can design 8 more Instruction. Now, in that particular case

(Refer Slide Time: 41:41)

Instructions

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

Handwritten notes around the table:

- 900 | 15 | 14
- 6900
- INC: increment
- DEC: Decrement
- B002
- R2 - 1
- counter

We are designing some more Instruction over here. So, this is INC is basically in increment and DEC is your decrement. So, what will happen we are having one increment operation and one decrement operation we can increment the value of my Register or Memory or we can decrement the value of my Memory or Register.

So, in that particular case say if I am going to say 6900. So, in that particular case what will happen 6 is my decrement. So, whatever value we have in the Memory location 900 say if this is my Memory and in my Memory location 900 say we are having say 15 then what will happen it will decrement of value of this particular Memory location and after execution we are going to get 14.

Similarly, if I am going to have say B002 now what will happen it is basically nothing, but we have been the Register R2 to the value of Register R2 is nothing, but R2 minus 1. So, this is the way we can look into it now when we are using increment and decrement operation in processor general we are not going to use the ALU for such type of Organization. Because, here we are having some values in the Accumulator.

If I am going to use Accumulator for decrement that and you have to bring it from the Register to the Accumulator. So, it will get this type; that means, temporary you have to store the value of Accumulator. So, instead of doing it what generally we use to do we are going to put specialist circuit over here which is going to increment it or decrement it. One simple way you can think about that how to implement it maybe we can use a counter which is an up down counter because already we have discuss up down counter. So, we put the value once we going to execute the a increment Instruction what we are going to do we are going to count up and one we are going to implement a decrement operation we are going to count down.

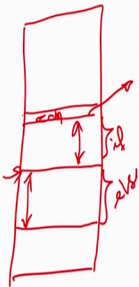
So; that means, value of R1 will be pressed to counter. So, basically what we are going to do when we are going to have this thing say decrement your R2 then what will happen we may use a counter over here. So, a value of R2 will be loaded over here then we are going to count down operation then B is your increment. So, we are going to count up. Then value of counter will be incremented by one and going to store it back to Register R0.

So, like that now we say out of 16 we have now consumed 12 operations. Still 4 are remaining left till now we have designed this particular 12 Instruction still 4 more for the remaining left 087 and F. So, that we can design 4 more Instruction now just see what we are designing.

(Refer Slide Time: 44:52)

Instructions

Op-code	Instruction	Op-code	Instruction
0	JMP	8	HALT
1	LDA M	9	LDA R
2	STA M	A	STA R
3	INC M	B	INC R
4	SUB M	C	SUB R
5	ADD M	D	ADD R
6	DEC M	E	DEC R
7	IJZ	F	JNZ



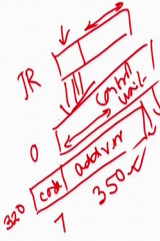
Z - flag zero

Result of ALU operation + zero

JZ: Jump on Zero

JNZ: Jump on not zero

JMP: Jump (unconditional)



if (a=0)

{

i

j

k

l

m

n

o

p

q

r

s

t

u

v

w

x

y

z

}

Now, here I am saying that Op-code 0 is your JMP. It is a jump Instruction. It is a halt Instruction; that means, it is going to say that halting the program or stopping the program execution this means at the end of the program we have to give this whole Instructions just to say that now stop execution need not fetch any more information.

Similarly 7 is given as your JZ and F is your JNZ. This is JZ is your jump on 0 and JNZ is your jump on not 0 and this is JMP is your jump. So, these are the control Instruction we are defining this jump is your unconditional jump without any condition we are going from one Memory location to another Memory location, but jump Z and jump not Z and Z jump 0 and not 0 these are conditional jump Instruction. It depends on some condition and we are going to jump to some other Memory location. So, basically what will happen we are talking about the 0 know how we are going to take a decision you know that we are having an Z flag which is basically 0 flag now when we are going to set this particular 0 flag when the ALU operation is 0.

Now, you just see that in your high level language sometimes you write if a equal to 0 do something otherwise do something else. So, if you are having such type of program. So, in Memory what will happen we are storing our problem like that you are said this is the conditional Instruction and this is one part and this is the other part.

So, I can say this is if part this is else part. In that particular case what will happen first we are going to check this particular condition depending on the condition if it is true we are going to execute this particular set of Instruction and if it is false then we are going to execute the other set of Instruction. It is making a say choice.

Now, how we are going to making a choice depending on the condition. So, such type of condition can be implemented with the help of this conditional jump Instruction. So, one of the things you are talking about jump 0 so; that means, we are having this particular 0 flag. Zero flag will be set to one if the a result of ALU operation is 0 result of. So, we perform an ALU operation if the result of ALU operation is coming as 0 then it is going to take a decision and it may take come to this particular Memory location. We have to specify the Memory location along with Instruction. We know the format. Basically this is your Op-code and this is address.

So, what will happened here I am going to say that Op-code is 7 and stay address is 350. So, in that particular case when I am going to execute this thing if this condition is true we are going to jump to the Memory location 350 and say here it is a now we are in set 320. So, next Instruction should be we need to fetch from 321. But when this condition satisfied then we are not fetching it from 321, but we are going to fetching it from 350.

Similarly, that Op-code 0 it is unconditional jump. So, whatever address you provide it will simply go to that particular Memory location then how you are going to get it; that means, you are going to load the program counter with this particular of value then only we will be knowing that next Instruction you have to fetch from that particular Memory location and it is halt it is going to say that stop the execution of the program. So, we need not to fetch any more information from Memory now you just see that we are having 16 different operation with and we are assigning in this particular 16 different operation to those particular code 16 code.

Now, when we fetch it then what will happen it will come to the IR and in IR this is the basically code and this is the other 12 bits is different. These code what this 4 bit we are going to give it to the control unit and according to the nature of this particular Instruction, now control unit will generate the appropriate control signal and this will go to the appropriate component inside the processor. So, this is the way we can see how we are going to execute this particular program.

(Refer Slide Time: 50:00)

Program

- Write a program to calculate the total marks scored by a student in an examination having six subjects.
- Program segment:


```

t_m = 0, n_s = 6, m[6]
do { t_m = t_m + m[n_s];
    n_s = n_s - 1;
} while ( n_s > 0)

```

Now, you just see that taking a small example which is I am saying that I have to write a program to calculate the total marks scored by a student in an examination having 6 subjects. So, say in your semester you are having 6 subject you scored say marks in the 6 subject and finally, you can get the add them together and find out what is the total marks that you have scored.

So, in some high level language you can write such type of program code what is that now we are having we need some (Refer Time 50:32). You are defining one t underscore m. This is going to say what is the total mark initial this value is 0, n underscore s number of subject now you are talking about the 6 subject and we are having much of 6 different subject. So, we are all saying m 6. So, in high level you know that you can define array.

So, you are defining arrays. So, we having six element 1, 2, 3, 4, 5, 6. So, here I am having some marks like that 25, 97 something like that. So, this marks whenever defining arrays. So, this marks will be stored in our Memory in 6 consecutive Memory location.

Now, what you are doing now do what I am going to do t underscore m is equal to t underscore m plus m[n underscore s]. So, if this is 1, 2, 3, 4, 5, 6 location of this thing first I am going to add the contents these things to total marks. So, total marks will become 0 plus something than we are decrementing n s by 1. n s minus 1. So, now, n s becomes 5. So, I am going to perform this particular operation while n s is greater than 0 after that I am going to check now it is 5 not it is still greater than 0.

So, we will go back and we add the next number will decrement it like that when it becomes you come out from the loop and it will go to the next Instruction. So, I am getting after coming out from this loop what you are getting we are adding all those numbers and the result is available in the t m t m is equal to t m this array element.

And this array will be stored in consecutive Memory location in my main Memory. Now this is a high level language now when we are going to write a program. So, in low level.

(Refer Slide Time: 52:17)

Assembly Level Program

	Assembly code	Remarks
	LDA M1	// No. of subjects
	STA R1	// Keep it in registrar R1
	LDA M	// mark of first subject
	DEC R1	
Loop	ADD M	// marks of subsequent subject
	DEC R1	
	JNZ Loop	
	STA M2	// store the total score
	HALT	// stop execution

R1 → 6
 = 6
 A ← 75

75 → 6
 75
 701 → 67
 702 → 85
 92
 78
 89
 75+6

Or assembly level, I can say like that. So, this is basically related to this particular example here we are saying what first I am writing load Accumulator M1.

So, basically what I am thinking say this is the Memory and we are having some Memory location in this Memory location we are storing that number of subject then I am having say 6 marks in some consecutive Memory location this is say 75, 67, 85, 92, 78, 89. So, this is a number of subjects say here basically we are storing 6.

So, initially first I am bringing it this particular number 6 then store Accumulator R1. We are storing this particular thing in Register R1. So, in R1 what we are having 6 this is the number of subject then load Accumulator Memory. So, I am now going to give the address of this particular Memory location in this possible Instruction and we are getting this is now what we are having this information is in Accumulator now. So, in Accumulator we are having 75.

Already we are taken care of the first number. So, we are decrementing R1 now value of R1 become now 5 because already have taken care one number then what you are doing ADD M what ADD M will do it will take the contents of this particular Memory location 67 adding with the Accumulator and store the result in a Accumulator. So, we are taken care of the second number. So, we are decrementing R1.

So, now R1 will become 4 we will say sum not equal to 0 then go back to this particular Memory location and fetch this operation. So, it will be in this particular loop till we are going to adjust this particular 6 number once it is over then say store Accumulator and I can say that the result we are going to store in some Memory location and after getting the result in the Memory location now we halt it. So, this is the way we are going to do it.

Now, this program cannot be written for the processor that we are discussing and for the Instruction set we are having because here you just see that here I am talking about that add m you are talking on Memory location when you go back in the next time we have to go to the next Memory location, but here in this Instruction set we do not have any provision to manipulate this particular address Memory. So, that is why this program is cannot be executed in this particular processor if you are going to write this particular program what will happen is going to add this particular 75, 6 times because we do not have any provision to sense manipulate this particular address m we do not have any Instruction.

So, address manipulation is not available in this particular Instruction set. So, if you are going to execute this particular program in this particular processor we are going to add a 75 in 6 times. So, result will be your 75 into 6 because we do not have an Instruction to manipulate this particular address m because if I am storing it in say 700 second one I have to take it from 701 then 702; that means, we need to increase this particular M after adding that particular number.

But in my Instruction set I do not have any such provision whatever we are doing incrementing and decrementing it is going to increment and decrement the contents of the Memory location it is not going to manipulate the address. So, we need some more Instruction to manipulate the address also. So, we will see in a subsequent lecture, but in this processor we cannot design it because we are having 16 code and we have exits all the 16 code. So, we need more bits to design more number of Instruction.

(Refer Slide Time: 56:20)

Computer Program

	Assembly Level	Machine Level	
	LDA M1	1 701	
	ADD M2	5 702	
	ADD M3	5 703	
	ADD M4	5 704	
	ADD M5	5 705	
	ADD M6	5 706	
	STA M7	2 700	
	HALT	8 000	

C
a b c d
g c c
1 a m b t
7 a m t

So, one simple program I can write for this particular machine. So, load Accumulator m one. So, first number I am getting it in Memory location the way I am saying that 700 so; that means, load is from this particular Memory location an m 2 take the number from the second location add it and store it in Memory location like that ADD M3, ADD M4, ADD M5, ADD M6 then store the result in M7 and halt. So, we are going to write individual Instruction for each and every number.

So, for 6 of the give it is fine, but if I say that going to add thousand numbers then we cannot write thousand Instructions somehow we have to use such type of loop only. So, for that we need Instruction to manipulating address also now finally, I can say that now what we are saying that this numbers we are storing in Memory location 701, 702 like that.

So, in assembly level we are writing it then in machine level we are going to have this particular effect. So, now, you just see that let we are talking about the assembly level machine level and like that we can have high level also. So, here I can write high level program the way I can writing over here. So, this is a high level program. So, the same problems I can see this is the assembly level and this is a machine level.

Now, when I am going to write an high level program now how I am going to execute it. So, somehow I have to convert it to the machine level program. So, for that we are having a software which is known as your compiler. So, compiler is there. So, we are going to compile it and after compilation you are going to get such type of machine level code and this machine level code will be executed in program.

So, if you are accustomed with your C program. So, if you have written any C program and say if you are working with a unique system on a Linux system generally we use some compiler like GCC. Like that, say if you are writing a program called say abc.said dot c then what will happen generally you compile something like that GCC abc dot say and it is going to give me a dot out. So, this is the executable file and these executable files are not having such type of machine level code only.

Now, we can execute this particular code. So, high-level language can be converted to machine-level code similarly we are having an assembly level also we can write the code in the assembly level and with the help of assembler we can convert it to the machine-level code finally, we need this particular machine-level code to execute in the computer.

So, assembler is going to convert assembly code to the machine code and compiler is going to convert the high-level language to the machine code of a particular machine along with that we are having another term which is known as your interpreter. So, again interpreter is used for high-level language. So, in case of compiler we are going to compile the whole program and going to get an executable file and going to execute this particular file, but in case of interpreter it is going to interpret instruction by instruction and it is going to first interpret the instruction first then going to execute it then it will interpret the instruction to and going to execute it.

So, if I am writing a high-level program it is going to interpret the instruction by instruction and execute it one by one now finally, we are going to get this particular machine-level code now you have to execute it now one how you are going to execute it generally you give the command like that a dot out what will happen when I am giving the command a dot out it will load this particular program to main memory.

(Refer Slide Time: 59:57)

Computer Program

pg 150

	Assembly Level	Machine Level	Memory Location
	LDA M1	1 701	150
	ADD M2	5 702	151
	ADD M3	5 703	152
	ADD M4	5 704	153
	ADD M5	5 705	154
	ADD M6	5 706	155
	STA M7	2 700	156
	HALT	8 000	157

So, you just see that we are loading this particular programs say from main Memory location 150 to 157. So, we are loading it and Memory location; that means, in Memory location hundred fifty I am having one 701 in 151 I am having 5 702 like that 157 we are having 8000 the other bits are immortal weather it is the all function. So, we are loading it to the Memory location and when we load it to the Memory location then we are having the program counter that program counter will be load with the value 150.

Because we must know the address of an Instruction since this program is loaded from Memory location 150 to 157. So, program counter will be loaded with this particular value one fifty now we are going to fetch the Instruction from 150 will execute it after execution of first Instruction next Instruction we have going to fetch it from the 151 will execute it like that one you fetch the Instruction from 157 we all the program we stop the program execution; that means, we are not going to fetch anymore Instruction from 158.

(Refer Slide Time: 61:04)

Assembly Level	Machine Level	Memory Location
LDA M1	5 701	150
ADD M2	5 702	151
ADD M3	5 703	152
ADD M4	5 704	153
ADD M5	5 705	154
ADD M6	5 706	155
STA M7	2 700	156
HALT	8 000	157

Memory location; it stop; this is the way we are going to execute all program in our computer.

(Refer Slide Time: 61:13)

Test Items

Q1. Why there are three levels of programming languages
(Objective 3)

Q2. What is assembler and compiler. What is an interpreter.
(Objective 3)

Q3. Why it is not possible to implement the loop with the given
instruction set of the processor discussed in this lecture.
(Objective 1 & 2)

Now say I feel that now you are having an idea how a problem is executed in a pro computer whether we are writing it in a high level language or assembly level language or machine level language.

So, now you just I am giving some very simple test item here I am saying that first question is why there are 3 levels of programming language now I am talking about the machine level assembly level and compiler you just see if I am going to give it a processor and just saying that you just program it then what will happen you have to write everything in machine level; that means, you must remember the code of each and every Instruction which is not possible always have to take a reference.

But instead of remembering Instruction or numbers slightly it is easy to remember some code like that add subtract multiply load store. So, for that what will happens we are giving a code to each and every Instruction which is known as your mnemonic codes now once we have the mnemonic codes now we can use this code to write a program.

So, if we write our program with the help of mnemonic codes then we are going to say this is the assembly level code after writing in the assembly level code with the help of assembler we are going to compile it to the machine code and that machine code will be executed, but again remembering the assembly code on mnemonics is not that easy because now it is I am talking about the 16 Instruction.

But if I am going to design an processor where the size of the Instruction is a 8 bit that Op-code then I can design 256 defining Instructions. So, remembering the define Instruction mnemonics is difficult. So, for that we are coming up with the high level language you write your program the way you think, but we have to follow the syntax of a particular programming language

So, when we are going to write c program we must follow the syntax of the c program once we write it then can compile it to the machine level. So, this is the requirement of the high level language because it is difficult to code in machine level it is against slightly difficult to code in assembly level also because we have to remember many more thing. So, that is why you write in high level then converted to the machine code.

Question 2 what is assembler and compiler and what is an interpreter. So, already I have mention it what we use to do it assembler and compiler and what is an interpreter question 3 why it is not possible to implement the loop with the given Instruction set of the processor discussed in this lecture.

So, we are just saying that as small processor Instruction is 4 bit already I have explain why you cannot implement that loop. So, for implementing the loop we have to have some Instruction to manipulate the address even if you are going to take the information from our Memory.

So, this is required. So, that is why here it is not possible.

(Refer Slide Time: 64:09)

Module: Fundamental of Digital Computer

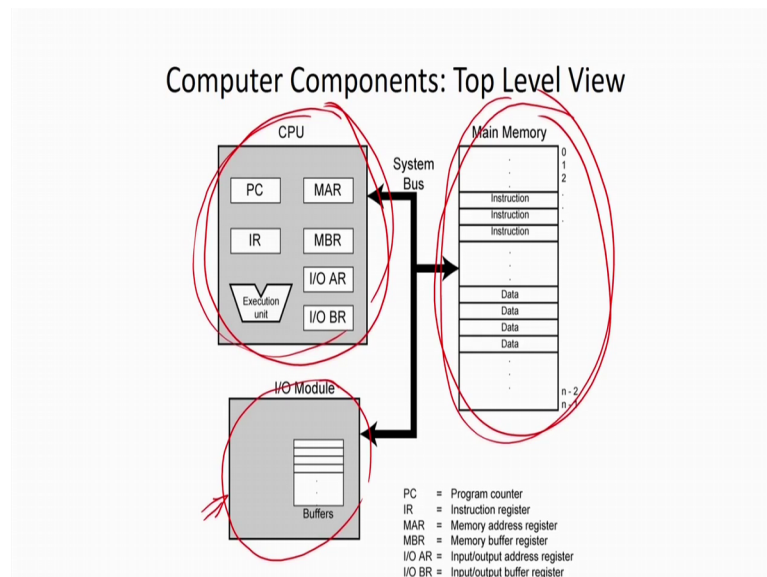
- Module Units
 - Unit-1: Model of Computer and working principle
 - Unit-2: Digital logic building blocks
 - Unit-3: Information Representation and Number system
 - Unit-4: Basic elements of the processor
 - Unit-5: Storage and I/O interface
 - Unit-6: Execution of program and programming languages

Now, this unit is the last unit of this particular model fundamental of digital computers so; that means, we are coming to end of this particular module. So, in this particular module we have decided this module into 6 unit. So, in first unit we are talking about model of computers and working principle. So, we have discuss about the how we are going to model a computer and how a computer works unit 2 is about digital logic building blocks because we are going to. So, you are going to take help of many more digital building blocks. So, because we are going to construct a digital computer. So, we are going to take help of many more digital building blocks. So, just we are giving an brief ideas about those particular digital building blocks and we are going to use those building blocks

So, we are not discussing anything about the design issues of those particular digital building blocks in knowledge level we have discuss it unit 3 basically bit infor information representation and number system. So, how to represent numbers integer and real we have discussed how to represent other information like that how you are going to write your name store your name in a computer all those things we have discussed in unit 3.

Unit 4 is basic element of the processor. So, we are discussed about what are the basic elements that we have in the processor in the top level then unit 5 we have discuss about storage and I/O interfaces because processor works on von Neumann stored program principle some we have to give the information to the processor. So, that is why we have to interface or connect storage unit and as well as I/O unit and unit 6 that last unit that today have covered we have discuss about how you are going to execute a program and what are the different levels of programming languages.

(Refer Slide Time: 65:48)



Now, So, what idea till now we have this is a top level view of our computer. So, main element is the processor it works on von Neumann stored program principle. So, we have the main Memory then processor is going to work the content available to the main Memory, but how we are going to bring the information to the main Memory for that we need input output device that will be connected through I/O modules and all those components are connected through this particular system bus. So, we are having these idea till now. So, in first module we have just have a very top level view about computer now in subsequent module. So, we are going to discuss about the design of those particular processor. So, we are having modules to discuss about the design issues of this particular processor or CPU.

So, we are going to discuss in details in subsequent module how we are going to design a processor how we are going to design an Instruction for that particular processor how we are going to interpret those particular Instructions. So, all Organization and architecture this will be discussed with respect to the designing of this particular processor we will have one module where we are going to discuss about the Memory module or Memory unit.

What are the components that you are having because here we have seen the top level only we have the data bus and we have that address bus with the help of this thing we can fetch information from Memory and stored information in Memory. So, what are the basic principle to design this particular Memory how you are going to construct the Memory you are going to design all those in this is another module on Memory model we will have one module where we are going to discuss all the design issues of the I o module ok.

So, we will get another module and in this particular case whatever we are going to discuss over here we are going to discuss the basic things on the this is unique processor system we are having one processor and you are going to work with this particular processor only we are going to execute in this particular process, but to announcement of the performance we are having some advanced topics also advance feature also.

But the something can be done in parallel or not whether it can be done in stages or not. So, some of the features to enhance the performance of the computer will be discuss in one module here we can we will discuss only in the information purpose only in the knowledge level only so

(Refer Slide Time: 68:12)

Module: Fundamental of Digital Computer

- Module Objectives
 - Objective 1: Describe the Model of Computer and working principle of Computer (Analysis)
 - Objective 2: Preliminaries of Digital Building Blocks (Knowledge)
 - Objective 3: Describe the representation of Information and Number Systems (Knowledge)
 - Objective 4: Explain the components of Processor (Comprehension)
 - Objective 5: Describe the Interfacing mechanism of storage unit and I/O devices (Comprehension)
 - Objective 6: Explain the execution of Program in a processor and categories of computer programming languages (Application)

When we are discussing about this particular module we have define some objective in a module level also because in every unit we have define unit level objective and we have seen that after going through that particular unit we have receive those particular objective.

Now, we have completed the complete module of fundamentals of digital computer let see where are you could achieve those particular objective that we have already mentioned for this particular module. So, the objective first objective mention like that describe the model of computers and working principle of computer and we have define it in the analysis level I think we are now having some idea what is the model of computer and how it works.

So, I think we have achieve this particular objective the objective type 2 we have define it like that preliminaries of digital building blocks. So, it is in the logic level. So, what will happen in that particular case we are just giving the ideas of those particular building blocks and we are going to use those things?

So, objective 3 describe the representation of information and number system. So, in this particular objective I think we have made that particular objective also because now we are having some idea about a number system and how to represents those number in computer real numbers and integer along with that how to represent the array information also objective 4 explain the components of processor. So, it is in the comprehension level.

Now, at least now we are having an idea what are the component that we are having in processor and how it works objective 5 describe the interfacing mechanism of storage and I o devices this is also in comprehension level. So, here now we are having that we have to connect I o devices we have to connect storage and I have to take information from those things.

So, in comprehension level we are having some idea, but in subsequent module we are having to discuss all those issues in the design level like for the processor also in subsequent level you are going to discuss all those things in our design level and objective 6 explain the execution of program in a processor and categorise the computer programming language this is in the application level or I can say that up to some extent we are going we have gone up to the analysis level also.

So, now we place if we get and program we will be able to analyse that particular program and. Secondly, if we know the Instruction set of a processor I think you will be able to write a assembly code for that particular processor. So, it is an application and analysis level we have a shift and in this particular course we are not going to discuss anything about this particular objective.

In 6 we are not going to discuss anything about the programming languages like that all those things we will be discuss in some other courses maybe that compiler is another course where you are going to discuss about the design of compiler. So, in this particular course we are not going to discuss anything about those particular programming languages and execution of the program now when we are completing this particular module now just look into some problems in the module level.

Now, what will happen in unit level problem you must know the concept of the particular unit then you will be able to solve the particular problem, but when I come to the module level problem then you must know the entire concept of this particular module, then only you will be able to solve those particular problems?

(Refer Slide Time: 71:17)

Module Level Problems

Q1. Consider a processor having an ALU with 4 Arithmetic and 4 Logic operations. The size of the data bus is 8 bits. Indicate how to use a particular operation of the ALU. Construct some examples to carryout some of these ALU operations and check the CARRY, SIGN and OVERFLOW. Provide the circuit to set the CARRY (C), SIGN (S), OVERFLOW (OV) and ZERO (Z) flags.
(Objective 2, 3 and 4)

So, here I am giving a Question 1 it says that consider a processor having an ALU with 4 Arithmetic and 4 Logic operations. The size of the data bus is 8 bits. Indicate how to use a particular operation of the ALU. Construct some examples to carry-out some of this ALU operation and check the CARRY, SIGN and OVERFLOW. Because, you can perform some Arithmetic operation. After performing the Arithmetic operation you can see the result. In the result can see whether it is generating the carry, whether what is the sign is it positive or negative or whether it is an overflow condition.

So, you just formulate some examples and try to carry out those particular example again it says that provide the circuit to set the carry sign and overflow and 0 flex. So, you are having flag which carry flag, sign flag, overflow flag, and Zero flag. So, this flag which will be set or reset depending on the ALU operation. Now you come up with an circuit to set or reset those particular flags.

Now, once you can able to solve that particular problem in this problem then what will happen we are going to meet the objective 2, 3 and 4 of this particular module. Similarly, look for the second question it is says that what are the different categories of Instruction in a processor design an Instruction set in such a way that some of the Instruction are having Memory indirect cycle indicate the format of the Instruction and the tasks performed by the Instruction.

Now, try to write a program having loop with the help of this Instruction set. So, basically just take a simple example take the design the Instruction set; that means, what are the Instruction that we are going to put after that come up with the format of those particular Instruction like that of Op-code address or whatever it may be come up with the format.

Now, once you know this particular format now you will be able to write a program with the help of those particle Instruction. Once you write the program say may be in the assembly level you write it then you just convert it the machine code of that particular processor because you are going to assign a machine code for each and every Instruction. So, if you are going to able to do solve this particular problem then what will happen you are going to meet the Objective 1, 4, 5 and 6.

So, with this we are coming to the end of this particular module Fundamentals of Digital Computer hope you are having some idea know how computer works and what are the components of the computer and in subsequent module we are going to discuss in detail about all those particular components.

Thank you very much.