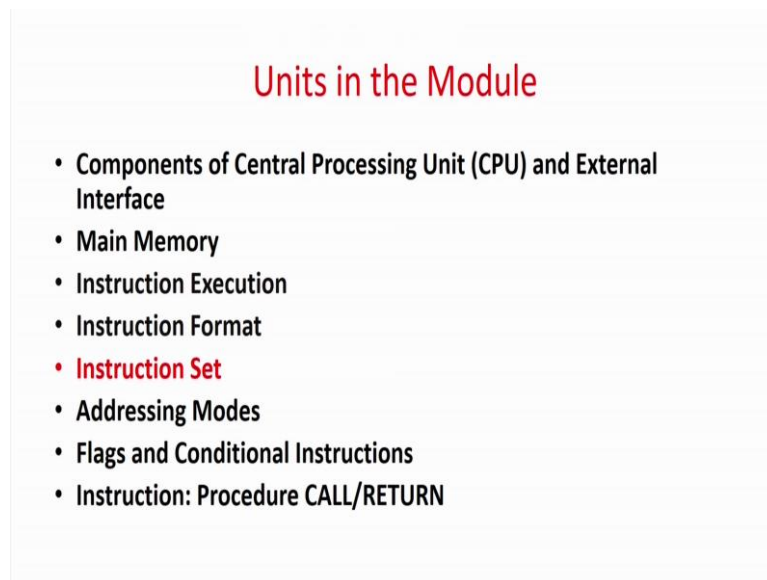


**Computer Organization and Architecture: A Pedagogical Aspect**  
**Prof. Jatindra Kr. Deka**  
**Dr. Santosh Biswas**  
**Dr. Arnab Sarkar**  
**Department of Computer Science & Engineering**  
**Indian Institute of Technology Guwahati**

**Lecture – 11**  
**Instruction Set**

So welcome to unit number 5 of the module on addressing mode instruction set and instruction execution flow.

(Refer Slide Time: 00:34)



So, till now basically we have mainly concentrating on how what is the basic instruction how it looks like and how it basically executes, now from now onwards we just try to go in more depth of basically how instruction works, how it is designed, how it can be clustered, what are the different type of sets in which you can club them etcetera.

So, in the last unit we basically saw about what is the basic instruction format that what it has, it has opcode different maybe depending on the instruction format it can have 1 address 2 address or even 0 address; now in this unit basically on instruction set we are going to see in more depth of how the instructions can be clubbed based on their functionalities. So, this is a small module sorry it is a small unit of the module in which case we will try to categorize the

say instructions based on their functionalities, many times we have discussed while discussing in this module that it can be of arithmetic type, logical type or data transfer type.

But in this case we will look into more depth of those classes and basically if you are executing an instruction, what are the different registers that are set, so slightly in more depth we will go in this ok.

(Refer Slide Time: 01:46)

The slide is titled "Unit Summary" in red. It contains a bulleted list of points. The first point is about instruction set design. The second point, "Data Transfer", is highlighted with a red box. The third point, about CPU tasks for data transfer, is also highlighted with a red box. The fourth point, about actions for memory operands, is highlighted with a red box. Within this fourth point, the sub-points "Calculate the memory address, based on the addressing mode.", "If the address refers to virtual memory, translate from virtual to actual memory address.", and "Determine whether the addressed item is in cache." are circled in red. The sub-point "If not, issue a command to the memory module." is underlined in red.

### Unit Summary

- One of the most interesting, and most analyzed, aspects of computer design is instruction set design. The instruction set defines most of the functions performed by the CPU. The instruction set of a CPU can be categorized as follows:
- **Data Transfer:** The most fundamental type of machine instruction is the data transfer instruction.
- The CPU has to perform several tasks to accomplish a data transfer operation. If both source and destination are registers, then the CPU simply causes data to be transferred from one register to another; this is an operation internal to the CPU.
- If one or both operands are in memory, then the CPU must perform some or all of the following actions:
  - Calculate the memory address, based on the addressing mode.
  - If the address refers to virtual memory, translate from virtual to actual memory address.
  - Determine whether the addressed item is in cache.
  - If not, issue a command to the memory module.

So, in this case basically what is this unit summary about. So, in this unit basically we will be classifying the instruction based on their functionalities and in each class what are the different type of instructions available we will be looking in depth. So, basically first is the data transfer, we all know that if the some instructions like load, store etc. Transfer data from basically based on one memory location to other, the memory location can be a register; it can be an accumulator, it can be a memory location in the main memory or it can be a cache. So, anyway we are not concentrating on the cache that way because, already we have told you and in between the registers and the main memory there are element of memory called cache, in which case wherever you want to execute some code a part of that main memory; where you are going to executive or the temporal data are loaded into the cache.

So, you can easily access them because the cache is inside the processor, but anyway that we will take in more details in our future module on memory. So, basically for us right now the classification of instruction of data transfer means you have to transfer data from 1 memory

location to another and in the last unit, we have seen that the how many such operations can be done in an instruction depends on the number of addresses.

If it is a 3 address then we can have more number of data transfers corresponding to a 2 addresses and so forth. So, basically what happens in this case so it has to calculate the address of the operands based on the addressing mode; so if the addresses then actually those things will be detailed out in a future module then basically you first find out whether the value of this memory location is available in the cache and if it is in the cache you can directly fetch it and if it is not in the cache then you have to read from the memory module; but in fact, this will be more dealt in more details in a future module as we have discussed just told right now. But in our as of now we have to just understand that it's a transfer of data from one memory location to another.

(Refer Slide Time: 03:40)

Unit Summary	
Operation Name	Description
Store	Transfer word from processor to memory
Load (fetch)	Transfer word from memory to processor
Exchange	Swap contents of source and destination
Clear (reset)	Transfer word of 0s to destination
Set	Transfer word of 1s to destination
Push	Transfer word from source to top of stack
Pop	Transfer word from top of stack to destination

So, as I told you we will be going into depth of basically or we will look into the more details of what are the exact type of memory, such type of data operation data or means what do I say that is data transfer operations. So, basically we have store so transfer one word of the transfer word from transfer to memory that is some operations you have already done in the processor and you store the result to a memory.

So, that memory is generally a main memory sometimes it can also be a register, there something called load fetch that is you actually transfer the word from memory to a processor; that can be passed to the register it can be passed to the register and accumulator. Exchange

such type of instructions are also there in which we can swap the contents. Clear, reset, set and reset also very important operations, there are some instructions which you can make all the values of memory location 0 or 1.

So, set and reset are also a class of operations which fall under this set, push and pop as I told you that you have to push a value to a register means stack and pop a value from the stack is also falls in the class of data transfer based or data operation or data movement based instruction. So, that is data transfer based instruction. So, as I told you that push and pop basically corresponds to a stack based machine or a 0 address based instruction; next is basically the arithmetic.

(Refer Slide Time: 05:00)

Unit Summary	
Operation Name	Description
Add	Compute sum of two operands
Subtract	Compute difference of two operands
Multiply	Compute product of two operands
Divide	Compute quotient of two operands
Absolute	Replace operand by its absolute value
Negate	Change sign of operand
Increment	Add 1 to operand
Decrement	Subtract 1 from operand

**Arithmetic:** Most machines provide the basic arithmetic operations like add, subtract, multiply, divide etc. The execution of an arithmetic operation may also involve data transfer operation to provide the operands to the ALU input and to deliver the result of the ALU operation.

So as I told you that overall we have been discussing throughout in many of the units over here, that there are 3 types of operable basically like mainly heart of all the computation is arithmetic and logic; that is you have to add 2 numbers, you have to multiply 2 numbers there is all the mathematical operations like add, subtract, multiply, divide, absolute, negate, increment, decrement. So, whatever things are whatever we know about standard mathematical operations the all the instruction sets or the instructions dedicated to it will be called as arithmetic operation.

But again as I highlighted in the last unit that add can be of several types the that add immediate; that means, you will have to add the value of 1 operand will be available instruction itself, add

two memory locations value of the 2 memory locations will be loaded then it can be add indirect.

So, as I idea is that even for add, subtract, multiply at each particular also particular operation also; there can be lot of variations as simply add immediate and add from memory immediate means the operand value will be will be given in the instruction itself. So, each can have a lot of different variations itself increment 1, increment, decrement, increment by 1, increment by 2 they are quite standard like negate, absolute, they do not have much variations like add, subtract, multiply you have lot of variations.

(Refer Slide Time: 06:16)

**Unit Summary**

Operation Name	Description
AND	Performs the logical operation AND bitwise
OR	Performs the logical operation OR bitwise
NOT	Performs the logical operation NOT bitwise
Exclusive OR	Performs the specified logical operation Exclusive-OR bitwise
Test	Test specified condition; set flag(s) based on outcome
Compare	Make logical or arithmetic comparison Set flag(s) based on outcome
Set Control Variables	Class of instructions to set controls for protection purposes, interrupt handling, timer control etc.
Shift	Left (right) shift operand, introducing constant at end
Rotate	Left (right) shift operation, with wraparound end

**Logical:** Most machines also provide a variety of operations for manipulating individual bits of a word or other addressable units.

Next is basically logical one logical means they are mainly basically bit wise operation so like and, or, not, exclusive or, then actually very important these are the standard ones, but there are some important ones like left shift, right shift, compare that is this test and compare actually these things are very important as we will see more on the in the future module, future unit we will be looking at the jump instruction or conditional instruction execution.

So, in that case actually test and compare this will be very very important that whenever some mathematical operations are done basically some flag values are set. So, you can test those flag values that whether the 0 flag is set then you take a jump instruction, you compare two arithmetic operations and then some set some values. So, if you compare 2 values like comp a, b. That means compare the register value a register value b if they are equal some flags will be set.

So, this type of instructions basically fall under the category of a logical instruction, like and or not are basic logics, left shift right shift are very simple, test and compare are very very important logical instructions which will be used for execution of jump instructions mainly; then some control variables can be set for some kind of protection purpose like interrupts etc, which will be dealt in later. In a nutshell these are some of the instructions basically which fall into the class of logical instructions. That is till now we were saying that and, or, not are mainly the logical instructions, but apart from the basic ones these are other very important logic instructions.

(Refer Slide Time: 07:46)

Unit Summary	
Operation Name	Description
Input (Read)	Transfer data from specified I/O port or device to destination (e.g., main memory or processor register)
Output (Write)	Transfer data from specified source to I/O port or device.
Start I/O	Transfer instructions to I/O processor to initiate I/O operation.
Test I/O	Transfer status information from I/O system to specified destination

**Input/ Output :** Input/ Output instructions are used to transfer data between input/output devices and memory/CPU register.

And then there are some instructions for I/O generally many of the cases we say that the I/O is a part of the data transfer operation, but for many cases we can also classify them as the input output; basically you read from some port, you write from some port that is the input output devices are available. So, there will be a whole module on I/O which will be taught by the other faculty members who are dealing with the courses. So, in that case it is more or less a data transfer like input is read output is write, but in this case it will not be exactly from a memory.

So, it will be from some output devices like it can be a mouse it can be a keyboard etc. So, there we have full unit dedicated to that maybe I can say start of I/O; that means, say I want to read from the keyboard. So there will be one instruction for that, test an I/O. So, whether the device from where I am going to read or write is functionally proper or not. So, there is another full sector of the operations or instructions basically which are dedicated to I/O sometimes

mean till now it means as a separate unit dedicated for I/O. So, till now we are not discussing much about input output, because sometimes in a very broad sense or an abstract sense we can also call it the data transfer operation. Because you are going to read, you are going to write, start. Start can be a control instruction, test can be again a control instruction or it can be a in fact logical operation you can tell like that. In a read write from a memory is very simple data transfer operation. But when you are talking to some input output devices it will be a I/O instruction like start and test I/O can be control or you can think of logical, but when you are talk about specifically about connection to an into other devices apart from memory then actually it's an I/O instruction. So, these part will be delta more details in a future module on I/O.

Then in the last part actually of this classification in the control instructions, as I told you so generally the instruction goes in sequence, but based on some conditions of an operation some flags may be set based on the value of the flag you can take the next instruction or some other instruction that is the conditional instructions.

(Refer Slide Time: 09:48)

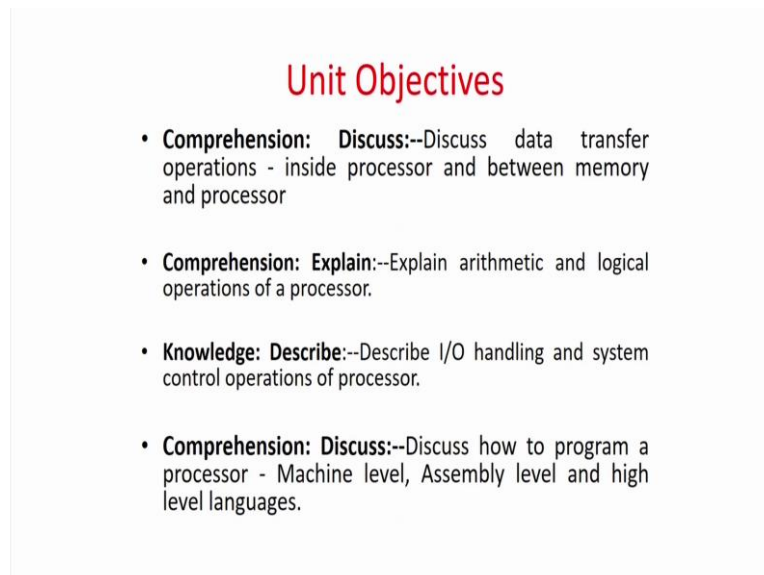
Unit Summary	
Operation Name	Description
Jump (branch)	Unconditional transfer, load PC with specific address
Jump conditional	Test specific condition; either load PC with specific address or do nothing, based on condition
Jump to subroutine	Place current program control information in known location; jump to specific address
Return	Replace contents of PC and other register from known location
Skip	Increment PC to skip next instruction
Skip Conditional	Test specified condition; either skip or do nothing based on condition
Halt	Stop program execution

Important conditional instructions are branch there is unconditional, load this specific address wherever you want to jump. Conditional means you have to check the value of the flag; if it is true you take that location or you continue as procedural. Next step jump to subroutine basically if there subroutines in the code you jump to that procedure, return, means after completing this subroutine or the interrupt subroutine ISR you go back to from where the procedure was called,

so in that case all the values of the PC etc has be brought back from the stack and it has to go some other instructions has keep conditional, that is next instruction may be skipped next instructions may be skipped on some conditions.

So, basically these are in a nutshell the broad classification of conditional instruction; jump, jump conditional, jump to route subroutine, return from the subroutine, skip an instruction, skip an instruction based on some condition and halt. Halt is also a very important control instruction, where you stop the code. Again as we are discussing everything for pedagogical perspective.

(Refer Slide Time: 10:42)

A slide titled "Unit Objectives" in red text. Below the title, there are four bullet points, each starting with a category in bold: "Comprehension: Discuss", "Comprehension: Explain", "Knowledge: Describe", and "Comprehension: Discuss". Each bullet point describes a specific objective related to processor operations, I/O handling, and programming levels.

**Unit Objectives**

- **Comprehension: Discuss:**--Discuss data transfer operations - inside processor and between memory and processor
- **Comprehension: Explain:**--Explain arithmetic and logical operations of a processor.
- **Knowledge: Describe:**--Describe I/O handling and system control operations of processor.
- **Comprehension: Discuss:**--Discuss how to program a processor - Machine level, Assembly level and high level languages.

So, again this is basically a recall and a knowledge based kind of an objectives mainly in this case, you will be able to after doing this unit you will be able to discuss the different type of operations inside a processor mainly between a processor and a memory that is data transfer operation, you will be able to explain about the arithmetic and logical operations.

You will have some idea and you will describe about I/O handling system and control operations of a processor, you will be able to describe the basic idea and in fact, a separate module we will discuss in depth and comprehension means here; you will be able to discuss how to program a processor in a machine level assembly language high level languages.

Basically the idea is that after doing this unit and also from the knowledges of some of the previous units, you will be able to tell that given a code a high level code, how what will be the



assembly language level, what will be the instruction at the assembly level, what will be the binary version of the machine level and what will be the corresponding high level; that means, given a high level code you will be able to translate it into assembly language and a machine language and discuss how it basically executes.

(Refer Slide Time: 11:45)

**Step wise instruction execution and CPU registers**

Example: The content of memory location FF0 is 5 and FF1 is 7. We want to add these two numbers and store the result in memory location FF2. Assume that we store this program from memory location 3F0.

To perform this task, following operations have to be accomplished.

- Contents of memory location FF0 have to be loaded into accumulator. As it is given that 5 is present in FF0, it should be loaded into accumulator.
- Contents of memory location FF1 have to be read and should be added to value in accumulator. Result of the addition should be stored in accumulator. 7 has to be added and the result should be stored in accumulator.
- The result of the addition which is stored in accumulator must be written into memory location FF2. i.e., result of addition (12) have to be written into memory location FF2.

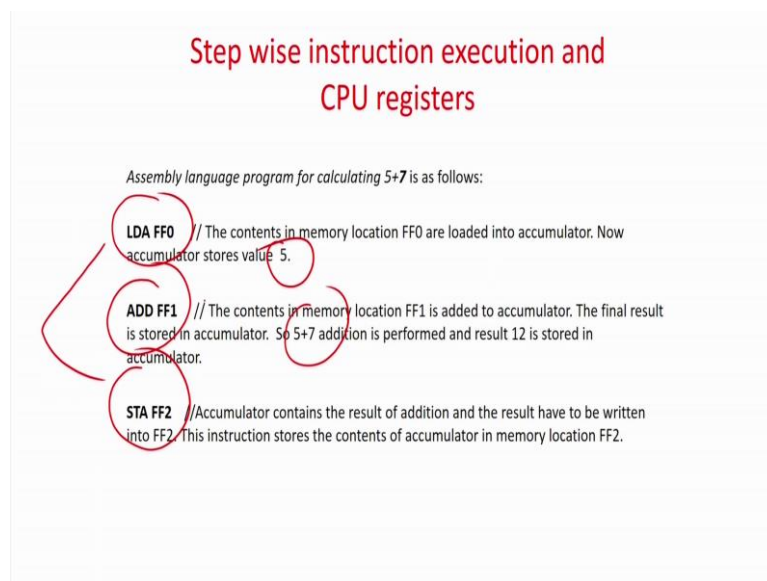
So, without I mean as I told you this unit is basically you can think as a second part of the last unit on the what about the instruction set, basically we are discussing on the instruction format. So, these last 2 units that is you so in fact if you look at it. So, instruction format and instruction set you can call that at the 2 small units there one is covering about the different formats and this one this unit is to talking about the basic classification based on some operations.

So in fact, that is why without going into much more theory in this unit let me give you in more details by some examples, that will set the that will make you help in understanding more on this units; basically because most of the theory related to this unit has been discussed in the last unit, anyway discussing about the instruction formats, because formats means what are the basic structures or what are the basic components and here we are discussing on the instruction set how to classify based on it is functionality like data transfer, I/O, control or arithmetic logic right.

So, we take a very simple example like there is 2 memory locations FF0 it has 5 and FF1 has the value of 7, we have to add these 2 numbers and the result has to be written in a memory location FF2.

Ok so, already such type of program we have discussed in a previous lecture previous unit, but here we are going to look at it more on a more from the perspective of the instruction set and instruction format as well as also we will look at how the memory is handled and how internal registers are handled and what are the control signals; it will be a same type of program now in more depth from the architectural concept as well as the instruction format and addressing mode concept. So, we are saying that the job is simple two memory location have 2 values you have to add them.

(Refer Slide Time: 13:33)



So; obviously the first instruction and yet we are taking a format what is the machine thing, the example you are solving here is a single address instruction. So, as I told you the same good we have seen maybe in a previous unit, but now we are going to focus more on it from the last 2 unit the current unit and last unit perspective, that is on the addressing mode and the instruction set.

So, the first one is LDA that is load accumulator FFO, so that means what? Whatever is value in memory location FFO will be loaded to the accumulator. So, this is a data transfer instruction as well as it's a single address instruction. Single address because as I already told you single address instruction means, one of the operator or the operand is basically a sorry the operand is basically accumulator. So, it is done and then as I told you whenever one thing I missed to tell you basically.

So, whenever you are writing a code you have to assume what is the instruction what are the machine type. So, for example, in this case we are assuming that the architecture of the CPU or the machine type is a 0 sorry single address format. So, in this case one is de facto a accumulator then we are saying add FF1. So, what is it's a arithmetic operation arithmetic instruction.

So, it says that whatever is in the value of accumulator that is value 5 which was FF0 for will be added to the value which is available in FF1 and it will be stored back to register, finally you have to store the value of STA FF2 that is the value of the accumulator you have to store it to FF2 memory location. So, now accumulator has  $5 + 7$  so it will be stored over there. So, to do this operation we have 2 I/O operations sorry 2 data transfer operation and 1 is A arithmetic operation and this machine is a single address machine.

(Refer Slide Time: 15:16)

**Step wise instruction execution and CPU registers**

Converting the assembly code to machine code :

Let the Opcode for LDA instruction is: 0000  
 Let the Opcode for ADD instruction is: 1000  
 Let the Opcode for STA instruction is: 0001

Hence,

LDA FF0 is equivalent to 0000 1111 1111 0000 in machine language.  
 LDA F F 0

ADD FF1 is equivalent to 1000 1111 1111 0001 in machine language.  
 ADD F F 1

STA FF2 is equivalent to 0001 1111 1111 0010 in machine language.  
 STA F F 2

*Handwritten notes: A circled '16' with an arrow pointing to the opcode field, and '16 bits' written next to the machine code representations.*

Now, as I told you so that now if you look at the machine code. So, as I told you machine code is always a binary code, but generally what happens we do not write it too much in the textual literature or the books or the slides, because it becomes very difficult to read and understand. So, we will feel guilty so what it says it says load FFO. So, single so this is for the opcode and this is for the operand single address. So, another is de facto is the accumulator which is not mentioned. So, maybe I am telling you that maybe this machine has 3 instructions. So, I think 2 bit codes were enough to do this, but let us assume that the machine has some more it has 16 operations to do. So, they have kept 4 bit as the size of the opcode.