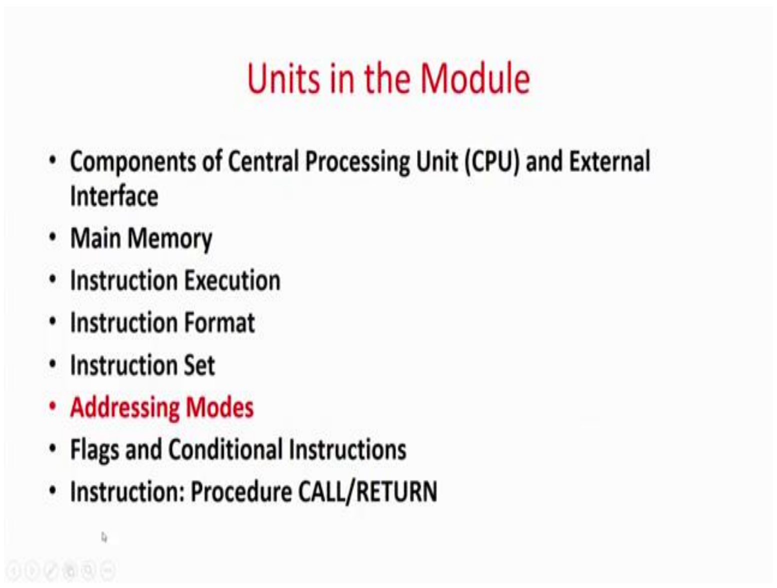
Computer Organization and Architecture: A Pedagogical Aspect
Prof. Jatindra Kr. Deka
Dr. Santosh Biswas
Dr. Arnab Sarkar
Department of Computer Science & Engineering
Indian Institute of Technology, Guwahati

Lecture – 12
Addressing Modes

Ok so welcome to the next unit on the module on Addressing Mode, instruction Set, and Execution Flow.

(Refer Slide Time: 00:34)



Units in the Module

- Components of Central Processing Unit (CPU) and External Interface
- Main Memory
- Instruction Execution
- Instruction Format
- Instruction Set
- Addressing Modes
- Flags and Conditional Instructions
- Instruction: Procedure CALL/RETURN

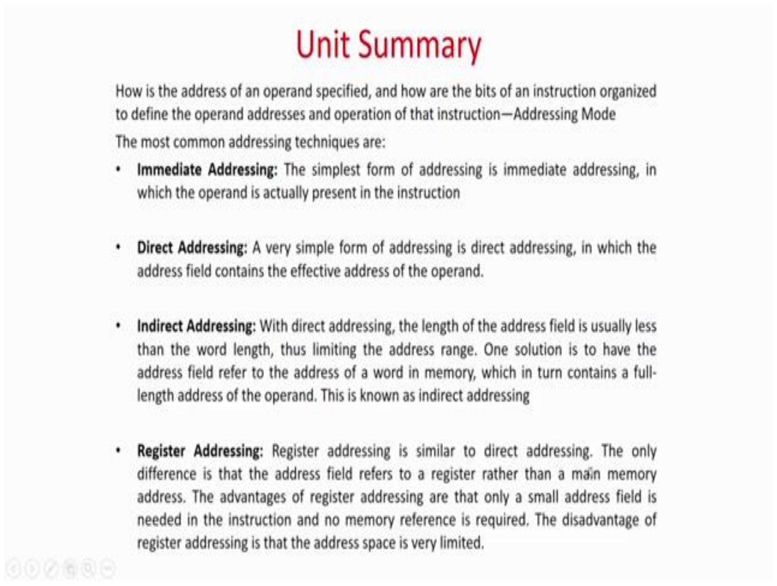
So, till now we have covered 5 units of the module in which we are discussing basically the basic architecture of the central processing units, how it is interfaced with a memory? Then how what are the basic instructions are and how an instruction is executed. Then we have also seen different format of instructions and then we have seen how instructions basically can be clustered together based on their formats.

Now, today we are going to focus on the next unit which is actually very important in this part which is called the addressing modes. That is, in other words or in an Indian language what happens that all instructions as we have discussed in the previous units they require an opcode that which tells what it has to do and of course, we require the operands.

So, how the operands are present in the instructions will be actually referred to as the addressing modes. Like for example, some instruction may have the data to be operated on in the instruction itself like add 5. Then you can call it as an immediate instruction mode, but sometimes you may have the instructions where the data is deeply embedded in a very indirect manner. Like for example, the address of the address of the data may be available in the instruction which is something called a memory indirect that is there are different spectrum in which case in which the data is present in an instruction.

So, in this unit we are going to discuss in detail what are the different addressing modes?

(Refer Slide Time: 01:57)



Unit Summary

How is the address of an operand specified, and how are the bits of an instruction organized to define the operand addresses and operation of that instruction—Addressing Mode

The most common addressing techniques are:

- **Immediate Addressing:** The simplest form of addressing is immediate addressing, in which the operand is actually present in the instruction
- **Direct Addressing:** A very simple form of addressing is direct addressing, in which the address field contains the effective address of the operand.
- **Indirect Addressing:** With direct addressing, the length of the address field is usually less than the word length, thus limiting the address range. One solution is to have the address field refer to the address of a word in memory, which in turn contains a full-length address of the operand. This is known as indirect addressing
- **Register Addressing:** Register addressing is similar to direct addressing. The only difference is that the address field refers to a register rather than a main memory address. The advantages of register addressing are that only a small address field is needed in the instruction and no memory reference is required. The disadvantage of register addressing is that the address space is very limited.

So, as we are dealing the whole course in a very pedagogical manner. So the unit summary we will talk about different instruction modes. So, the first one which will have we are going to see is the immediate addressing.

So, immediate addressing means the instruction itself will have the data. Then, there is something called direct addressing means the instruction will not have the operand itself, but it will be pointing to a memory address where the data will be present. Already we have seen in the fewer previous units that if you want to keep a very accurate or a data whose accuracy is high it requires a large number of bits to be represented.

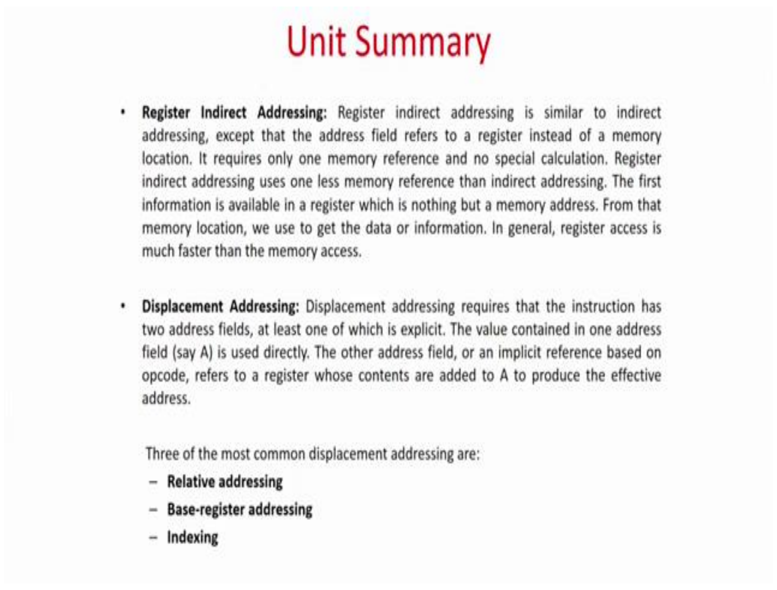
So, if you want to do it is as an immediate format then the instruction size will become larger, to avoid that we go for direct addressing that we give the address of the memory where the data is present and in fact, the data can be accessed from there.

Then there is something called indirect addressing. The details and advantages and disadvantages we will see as we will go along the units. So, what is an indirect addressing? In this case the data is not present in the address which is mentioned in the instruction it’s an indirect way of referring that is the one hop reference. The instruction will have the reference to a memory and in that memory location there will be another address which will be pointing to the exact data in the memory. We will later see that it allows you to expand the memory space or the size of the data to be represented.

Then there is something called we will see there is something called register addressing. So, register addressing is very similar to direct addressing, but in that case the only difference is the address refers to a register than a main memory. In this case there are as I told you there are different variations in where the data is present.

So, in register addressing what we do instead of saying that the memory is or the data is locating so and so memory location we will tell the data is present in such and such register. So, as the number of registers are small in number. So, the data or the space in the instruction required for such type of addressing will be lower as well as the access to data will be faster because you need not go to the memory you can directly get it from some other register. But again, the tradeoff is the number of registers are small so you cannot have a large number of data present in this addressing mode.

(Refer Slide Time: 04:11)



Unit Summary

- **Register Indirect Addressing:** Register indirect addressing is similar to indirect addressing, except that the address field refers to a register instead of a memory location. It requires only one memory reference and no special calculation. Register indirect addressing uses one less memory reference than indirect addressing. The first information is available in a register which is nothing but a memory address. From that memory location, we use to get the data or information. In general, register access is much faster than the memory access.
- **Displacement Addressing:** Displacement addressing requires that the instruction has two address fields, at least one of which is explicit. The value contained in one address field (say A) is used directly. The other address field, or an implicit reference based on opcode, refers to a register whose contents are added to A to produce the effective address.

  Three of the most common displacement addressing are:
  - **Relative addressing**
  - **Base-register addressing**
  - **Indexing**

We have seen direct, indirect then we have seen register similarly there can be register indirect. So, register indirect is very similar to normal indirect, but in this case the address refers to a register in a rather than a memory location.

So, what happens in indirect in indirect addressing? In the instruction you tell that this is the memory location where the data is present, but indirect means you tell the memory location in that memory location again there will be a redirection to the exact data. But in case of register indirect you are not going to refer to any memory location rather you are going to use a register to do the indirection.

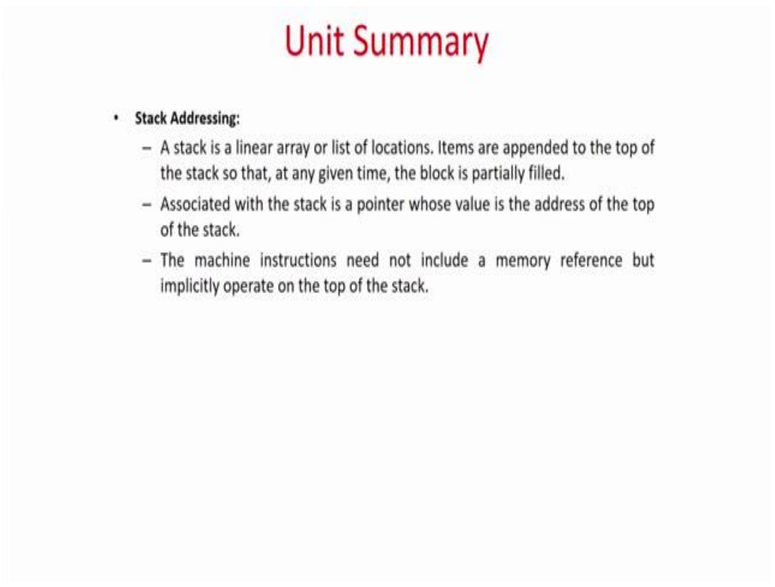
Another next very important is something called a displacement addressing. This is slightly different from all other three we have studied. In displacement addressing basically there are two parts of the address: one part is exactly fixed which is mentioned in the register in the instruction and there is another part actually which says to a whose contents are basically can be modified is some similarly something like that say may be we have a opcode and two parts of the address one parts is fixed and the other parts actually can be modified or it can be in fact they are added and you can play on with this two to this variable addressing part so that you can go for a wide spectrum of addressing like looping or you can go for a displacement then you start from 0 then you put a buffer or you put a displacement to that. Then you can move some 10 hex, start from 10 hex plus the kind addressing.

So in fact, you can understand we will see in details when we are going into the module. The displacement means there are two components of an address they are added to find out the effective address and one of them is modified so that or you can go as for example, you want to increment a loop. So, one part of the address can be changed or incremented is added to the other. So, you can get lot of displacement you can go in a loop and in that manner.

So, displacement addressing is slightly different from all the others we have discussed till now. So, all others are basically kind of a static way and here actually it is something like which is updated so you can go for a shift you can go for a loop etcetera.

And the most common displacement addressing are relative addressing, base register addressing and index register addressing, so that is we will see. Means based on the register in which you will be trying to do all this displacement the names have been given.

(Refer Slide Time: 06:25)

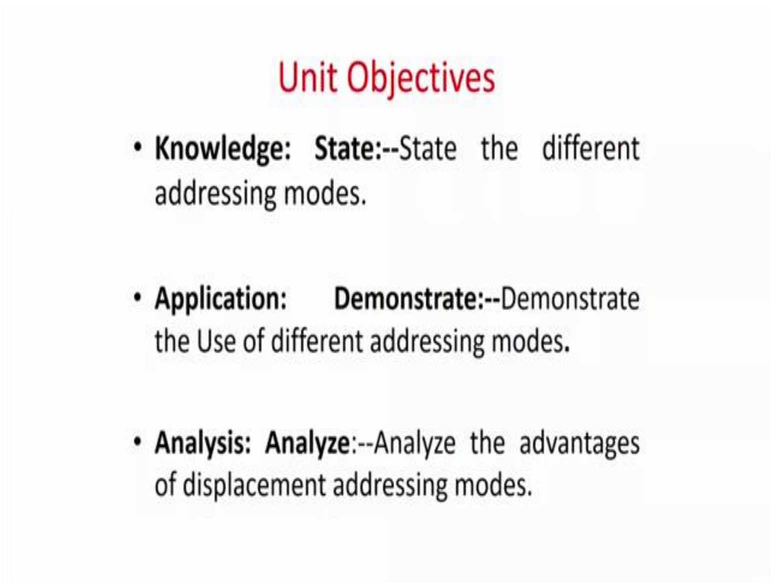


Unit Summary

- **Stack Addressing:**
  - A stack is a linear array or list of locations. Items are appended to the top of the stack so that, at any given time, the block is partially filled.
  - Associated with the stack is a pointer whose value is the address of the top of the stack.
  - The machine instructions need not include a memory reference but implicitly operate on the top of the stack.

And then finally, we will see stack addressing. So, it is very simple we have already discussed in the previous unit in case of stack addressing the opcode or the instruction will not have anything basically rather than the only the opcode and all the data will be present in a stack the top two elements of the stack one element of the stack will be used to do the operation.

(Refer Slide Time: 06:41)



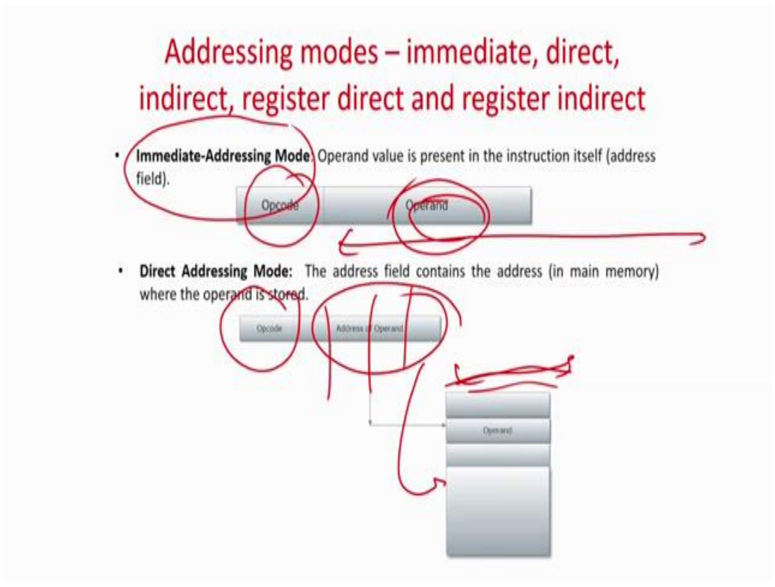
Unit Objectives

- **Knowledge: State:--**State the different addressing modes.
- **Application: Demonstrate:--**Demonstrate the Use of different addressing modes**.**
- **Analysis: Analyze**:--Analyze the advantages of displacement addressing modes.

And then what are the objective of this unit the objective of this unit is first knowledge you will be able to state different type of addressing modes. Then next is an application objective you will be able to demonstrate the use of different addressing modes when it is better, which one has a tradeoff, which is faster, which is slower, which takes a larger space in the memory and so forth. And then finally, analyze then you will be analyze the advantages of different addressing mode in particular the displacement addressing mode.

Because all other addressing mode as I was telling is some kind of a static either the data is present in the instruction or the instruction has a memory reference where the data is there. But in case of the displacement you add you increment and all those some kind of dynamism is present. So, this one you will be you will be able to analyze that what are the advantages of such a kind of a addressing mode.

(Refer Slide Time: 07:36)



Addressing modes – immediate, direct, indirect, register direct and register indirect

- **Immediate-Addressing Mode**: Operand value is present in the instruction itself (address field).
- **Direct Addressing Mode:** The address field contains the address (in main memory) where the operand is stored.

Opcode
Operand
Opcode
Address of Operand
Operand

Now, this was about the basic objective of the modules, the summary of the modules. Now we are going to look start the unit and we are going to look at real concrete examples and more in a more elaborate in a pictorial representation what are these type of addressing modes are?

So, the first one as I was telling you was the immediate addressing mode, immediate addressing mode means in the same instruction you will have the opcode as well as the data will be present over here.

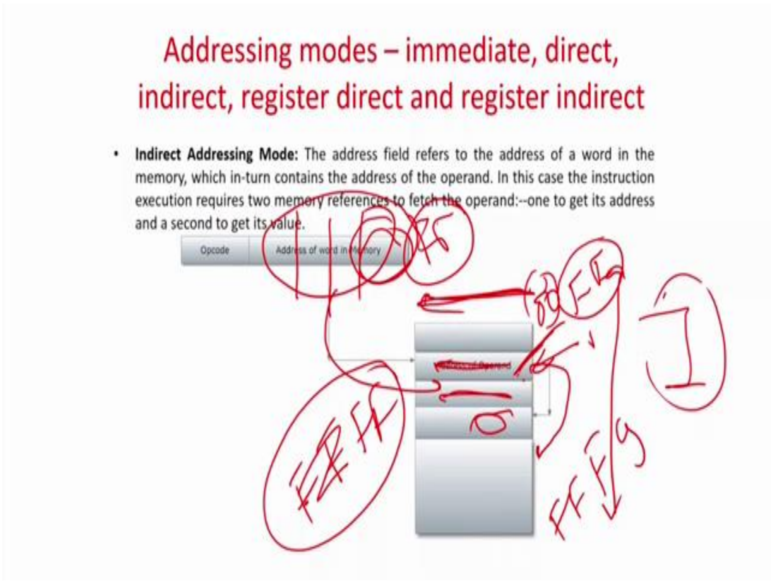
So, whenever this instruction will be fetched from the memory you need not have to fetch any others stuff or any other memory location or any other register to get the operand, the operand is present in the memory location itself. Then the next most, but the problem here is that as I told you if your data is to be more precise that you require more width to representation and this is not going to be a very handy kind of a addressing mode because the size of the instruction is has to be then made 2 words, 3 words, which is not actually very fast to be implemented.

Then what is another way of doing it? Then what you can do then you can say this is called the direct mode of addressing in which this is the opcode and here you will refer to a memory location and we know that this can be quite this is this why this is the width of the memory location. So, you have to just tell that which memory location the data will be present. So, this is direct way of addressing. So, if you say opcode something add 32 it means the data is available in the 32th memory location. But in this case there is a memory operation required.

But in fact, as you can see; obviously, you can have a wider space to do your operation. Because in a general instruction format there will be an opcode there is not only a 1 operand, there can be multiple operands. So, I mean you have to divide this space into multiple spaces.

So, you will have a very less space for each operand, but if you refer to a memory location then it will be just the address of the memory location and the data can be quite wide quite wide or you can have a more precise data in that manner. So, this is called the direct mode of addressing.

(Refer Slide Time: 09:24)



Addressing modes – immediate, direct, indirect, register direct and register indirect

- **Indirect Addressing Mode:** The address field refers to the address of a word in the memory, which in-turn contains the address of the operand. In this case the instruction execution requires two memory references to fetch the operand:--one to get its address and a second to get its value.

Then there is something called indirect mode of addressing. So, indirect mode of addressing is very similar to direct mode in direct mode what we have seen that this one will refer to a memory location where the data will be present, but in this case what happens the data will not be present over here rather it will refer to another memory location where the data will be present.

So, in this case to get a data you will have to have two memory addresses. So, first you have to address the first location which will tell you the location of the data which is available in the; that means, the data is available actually in this memory location.

So, first one you will have first the instruction will tell you which memory location to address which has the address the memory the memory address of the data is not directly present in the instruction itself which was the case in the direct addressing.

In this case it will tell the address which will refer to a memory address which will exactly which will absolutely have the address of the location of the data in this and not here. So, this is actually the address and not here.

So, there are 2 memory addresses required to get a data, but what is the advantage as I told you generally in opcode there is a number of operand present in any instruction. So, this is not quite large.

So, what happens actually so in fact, I mean if the memory size is quite large. So, you may not be able to put the full a full bandwidths of the, or I should not put full range of the memory cannot be encoded in this small space. For example, if you require FF, so if the memory has 00 to FFFF number of locations. So, 4 into 4 so it is FFFF then actually you require 16 bits to represent the memory location.

But in fact, if you understand that it is divided into 4 number of operands. So, you may not have the space to represent a 16 bit operand over here that is the memory address of this. So, in this case we can restrict that my data path will be a smaller part of the memory maybe I can say that 00FF will be limited to the range of the data.

So, in this case the 00 will be explicitly given it is implicit and the only FF has to be kept over here. So, I am actually making the size of the opcode size of the size of the operand means address of the operand in the memory smaller then you will apply FF00 will be explicit. So, you can go over there, but that is one way of doing it. The other way of doing is the in that case now you have you we will refer to this one.

Now, now in, but you need to actually address the full memory for that because memory the data can be anywhere in the memory. So, it may refer to this one, but the exact, but to access the full memory you required to access the 16 bit that is 4 4 4 4 the 16 bit address space of the memory. Now where it will be present? The exact location of the data will be I mean that data may be available in say FFF9. So, the data may be present over here.

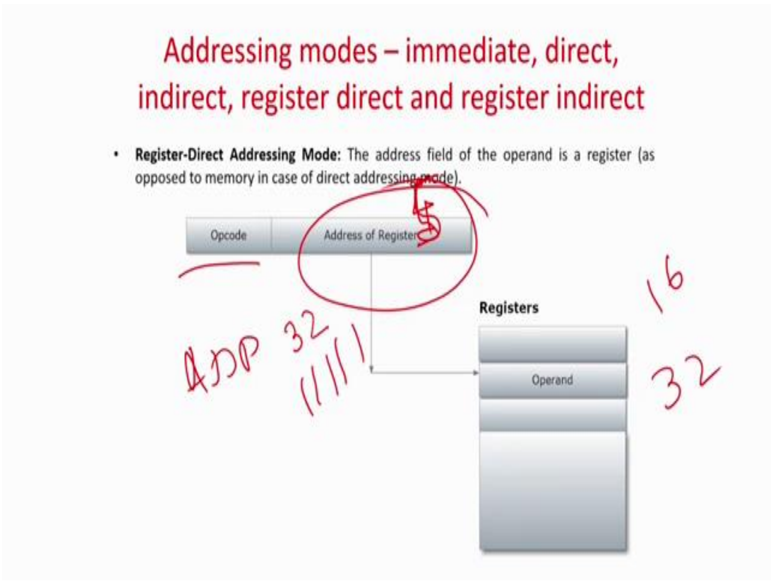
So, what it will do? So you are referring only FF because the exact location that is FFF9 is actually present over here so this one is having the value of FFF9 where exactly the data is present.

And in fact, that FFF9 we will actually take 16 bits to be represented that much space I don't want to keep in the instruction, but in the instruction I give the only say value FF that means, it refers to the first I mean the 00 is explicit. So, it just refer to this memory location and in this memory location I have the because this is much is quite wide compared to this smaller space in the memory.

So, now can you can access the whole range of the memory itself so that is one advantage of the indirect addressing mode. There are several others like we will see then actually we require a indirect addressing mode for implementing or accessing arrays and all those things.

But as we are I have not gone into that much of details in the course as of now so for time being this understanding is enough, to get a more wide more full range of the memory access such an addressing mode is useful.

(Refer Slide Time: 13:23)



Addressing modes – immediate, direct, indirect, register direct and register indirect

- **Register-Direct Addressing Mode:** The address field of the operand is a register (as opposed to memory in case of direct addressing mode).

Then next we are going to this was all about the direct, indirect and immediate addressing mode. But there is another way of doing is that is the instead of referring to the memory locations the data can also be present in the registers because register access is more faster it is inside the CPU itself and then other advantages are there regarding speed etcetera. But again this is what is the disadvantage the disadvantage is the number of registers are smaller in number.

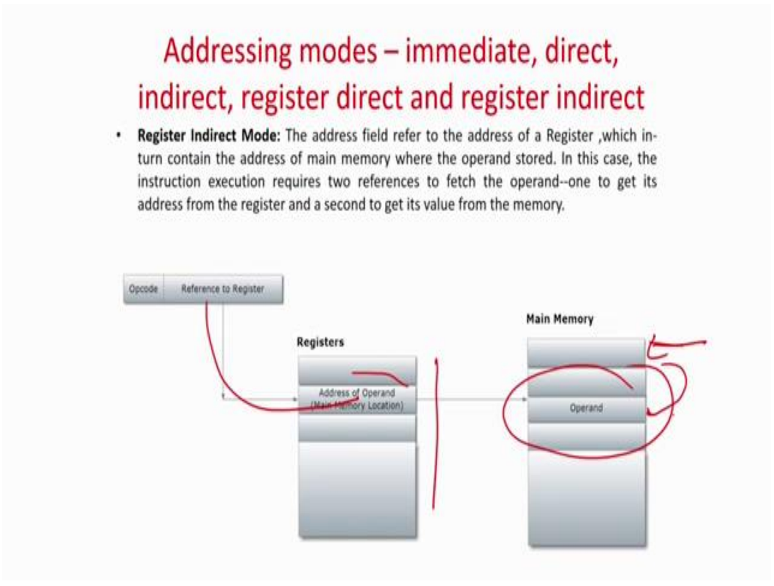
So, if you if you have you have more number of data that cannot be accommodated in a register, but it can be accommodated in a memory, but for certain access like you want to store some intermediate results where we want what the data which has to be fetched more frequently it can be temporarily stored in registers.

So, what is the register direct addressing mode an opcode will be there and there will be address of the register. In this case actually instead of referring to the memory you are going to have to refer to the register. In fact, registers are the if there is the 32 numbers of registers or in fact if there is 16 number of registers then you require just a 4 bit representation to represent any of the 4 register, 16 registers.

If there are 32 registers you just require a 5 bits for the address of the register. Say for example, if you write some add and then you write say a 32 that means what that is 2 to the power 5 that is 11111 ok. Then in that case you will be referring to the 32th register that is last register if you are having a thirty two number of registers, so you can see that the size of the address size of the instruction will be smaller and because the number of registers are less in number.

So, register direct means it's a just like a direct addressing mode the opcode will have the address of the data, but the data is present in a register and not in a memory.

(Refer Slide Time: 15:02)



Addressing modes – immediate, direct, indirect, register direct and register indirect

- **Register Indirect Mode:** The address field refer to the address of a Register ,which in-turn contain the address of main memory where the operand stored. In this case, the instruction execution requires two references to fetch the operand--one to get its address from the register and a second to get its value from the memory.

Then same thing also applies for register indirect just like a indirect addressing mode there is also indirect register addressing mode like for example, in the direct addressing of register. So, the opcode refers to a register and your data is present over the register in this register, but in this case is an indirect one. So, your opcode is there you refer to a register and the register actually refers to a part in the main memory where the data is present. So, this is register indirect.

What was the normal indirect mode? In general indirect mode means from the opcode you will actually point into one memory location which contains the location of another memory location which actually has the data. In case of register indirect you are giving the address of a register the data is not in the register, but the register is pointing to another memory location where the data is present and the advantages I told you that is before going for indirect addressing mode the you can access the full range of the memory because number of registers are in small, so if you compare to the general indirect general indirect addressing and register indirect addressing. Generally indirect addressing means you will first refer to one memory location here and in from that you are going to be redirected here. So, you require two general through main memory access. But in case of a register indirect the first you refer to a register and then you refer to a memory. So, in this case this is one register access and one main memory access. So, this is a faster way of getting the operand because register accesses are faster compared to a memory access. So, in this case there is a one memory access and one register access.

So, it is faster than your normal indirect mode where there is two memory access so that way there is advantage and disadvantage. Disadvantage is registers are less in number. So, you cannot use it very often while memory main memory is larger in larger in size. So, you can always you can have more frequent such type of indirect addressing mode ok.