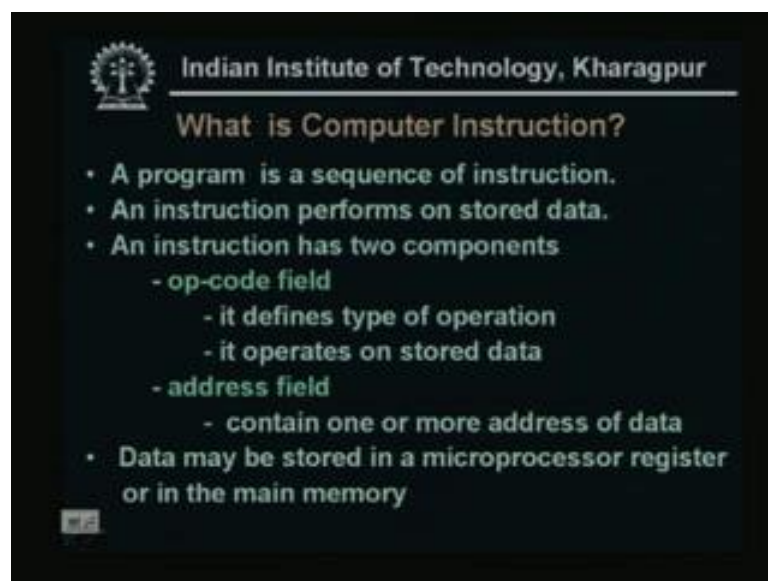


Digital Systems Design
Prof. D. Roychoudhury
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 33
Design of Computer Instruction Set and the CPU

Today, we will start discussion on the Design of Computer Instruction Set and the CPU. First we will see how we can design the computer instructions.

(Refer Slide Time: 00:59)



Now, before stating the design what do you mean by computer instruction, all for you know that a software program is consists of a sequence of instructions. Now, an instruction performs on stored data; that means a program is nothing but, a sequence of operations to be performed on the data and instruction is one single operation on the data. So, an instruction has two components, normally we called the op code field and the address field.

Now, op code field defines the type of operation and it operates on the stored data, address field contain one or more addresses of data, on which the operation is to be performed and data may be stored in a microprocessor register or in the main memory. So, instruction has two field an op code field and the address field, so when we want to design a computer instruction or to represent a instruction, so actually we have to design the op code field and the address field.

That means, which operation is to be performed and the operation to be performed on some data, where the data is stored the address. So, these two fields are actually to be designed.

(Refer Slide Time: 03:02)

Indian Institute of Technology, Kharagpur

Instruction Fields

Example

MOVE A0, A1

Op-code field Address field

Source Destination

- The instruction moves the contents of the register A0 to A1
- The number of instructions supported by a microprocessor depends on the size of Op-code field
- 8-bit op-code can specify $2^8 = 256$ unique instructions

Now, take one small example say I have one instructions the move A 0 to A 1, I am considering or assuming A 0 is the source and source address and A 1 is the destination. That means, I want the content of the address A 0 is to be moved to the or as the content of the destination address A 1, so my op code field; that means, which operation is to be performed that is my move, so this is my op code. And the source address and the destination address that is the A 0 and A 1 these are my address fields.

So, the instruction moves the contents of the register A 0 to A 1 the number of instructions supported by a microprocessor depends on the size of op code field. Now, all of we know the computer can understand only a 0 or 1, so this op code field depends on how many number of operations are allowed that depend that what is the size of this op code field. So, the number of instructions or the number of operations to be performed or a number of operations allowed depends on the size of op code.

So, this is very important when we will design the instruction set, so this is one of the important things to be considered that what is the size of the op code field. Say I have a 8 bit op code, means a 1 8 bit number, so by 8 bit op code we can specify 2 to the power 8; that means, 256 unique instructions.

(Refer Slide Time: 05:55)

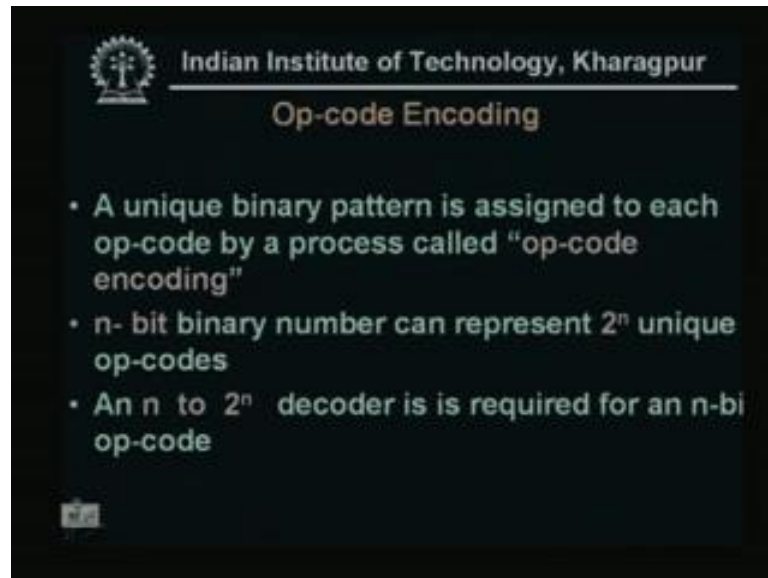
A handwritten diagram on a whiteboard titled "3-bit op-code field". It lists the 8 possible 3-bit combinations from 000 to 111, each followed by a horizontal line. To the right of these lines, the corresponding instructions are written: "instructions, add" for 000, "inst 2 sub" for 001, "3, incr" for 010, and "4, decr" for 011. For the remaining three bits (100, 101, 110, 111), there are two vertical lines to the right, with the text "8-diffⁿ opⁿ are supported." written next to them. The diagram is enclosed in a black border.

3-bit op-code	Instruction
000	instructions, add
001	inst 2 sub
010	3, incr
011	4, decr
100	8-diff ⁿ op ⁿ are supported.
101	
110	
111	

As if every 8 bit represents one unique operations, what does it mean say I have a 3 bit op code field. So, all possible 3 bits say 0 0 0 to 1 1 1, so these are my fields. Now, as if these 0 0 means 1 instruction, say I am telling add say 0 0 1 means another... This is instruction 2, say this is subtract, say 0 1 0 instruction 3 say this is some increment 0 1 1 instruction 4 this is say some decrement.

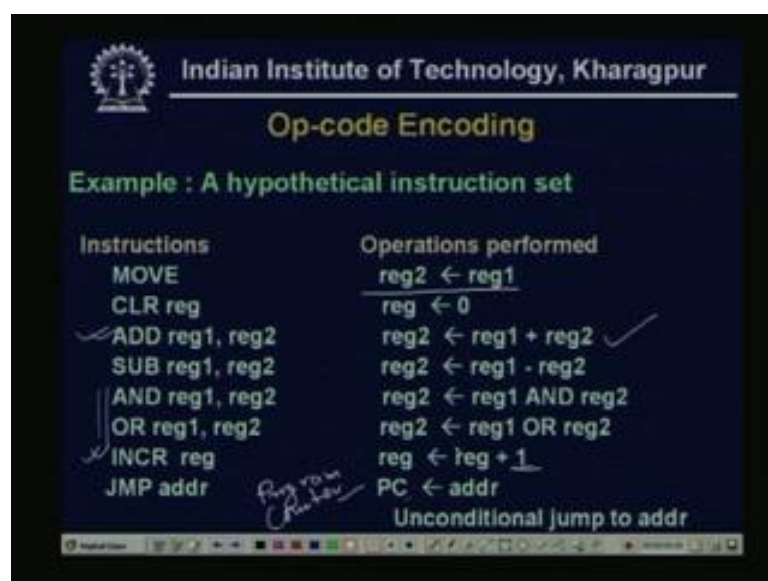
So, like 1 0 0 represent something 1 0 1, 1 1 0, 1 1 1, so for by 3 bit op code we can specify 8 different instructions or 8 different operations that are supported by this particular machine is a 3 bit by a 3 bit op code field, 8 different operations are supported. So, this is the meaning of that op code field.

(Refer Slide Time: 07:45)



Now, op-code encoding, so a unique binary pattern as all ready I mentioned that computer can understand only 0 or 1 and we use the binary systems. So, a unique binary pattern is assigned to each op code by a process called op code encode, just now I have given one example of a 3 bit op code. So, one n bit binary number can represent 2 to the power n unique op codes, so what now if I want to design this op code part, so what is it necessary I have some n bit number and I want 2 to the power n unique op codes. Now, this is nothing but, if I need a hardware all ready we have seen that an n 2 the power n decoder is required for an n bit op code.

(Refer Slide Time: 08:49)



Now, we take one example for a hypothetical instruction set, say I have instructions MOVE some CLEAR register, reg means 1 register, ADD 2 registers; that means, add the contents of 2 registers, register 1 and register 2 denoted as reg 1 and reg 2. Similarly, SUBTRACT the content of 2 registers, AND the content of 2 registers OR the value of 2 registers, INCREMENT 1 registers; that means, increment the value or the content of the registers by 1. And JUMP 1 address, means if this is a unconditional jump whatever be the condition in or irrespective of the condition, always this is a jump of jump to this address.

So, I have this 8 instructions; that means, I consider one machine which allow this 8 primitive instructions. Now, MOVE operations mean say one content of register 1, so when I keep MOVE operations, the content of register 1 is moves to content of register 2. So, this is denoted as reg 1 to reg 2, now CLEAR register means it is reset, so this register or the content of this register becomes 0 at reg 1, reg 2, now always this is conventional that after adding the content of 2 registers reg 1 and reg 2 the sum or the result of this addition goes to register 2.

Similarly, SUBTRACT, so the result of the subtraction of the content of register 1 and register 2 move to the register 2; that means, the content of register 2 now becomes the result of the subtraction. Similarly, AND, OR these are two operations and this is the reg 1 and reg 2 and reg 1 or reg 2 and the result goes to reg 2, INCREMENT reg normally increment means, the content of the register is incremented by 1.

So, reg plus 1 means the content of register is incremented by 1, 1 is added with the content and then the value is shifted to reg again. So, this is nothing but, our ADD again only here instead of 1 register 1 fixed value 1 is always added, now JUMP address, so this is PC means the Programmer Counter, this is the program counter. That means, always the current it contains the current addresses, so address goes to program counter; that means, this is a unconditional jump to address.

So, the address where I want to jump now, I give that value to the program counter and we know that always the program counter contains the current address. So, it will be jump to that particular address, so these are the meaning of these instructions. So, if we give these instructions, then what operations to be performed that are being specified by this right hand side expressions.

(Refer Slide Time: 13:04)

Indian Institute of Technology, Kharagpur
Op-code Encoding Using Block Code

Example : A hypothetical instruction set

Instructions	i_2 i_1 i_0 3-bit Op-code
MOVE	→ 0 0 0
CLR ✓	→ 0 0 1
ADD	→ 0 1 0
SUB	→ 0 1 1
AND ✓	→ 1 0 0
OR ✓	→ 1 0 1
INCR ✓	→ 1 1 0
JMP ✓	→ 1 1 1

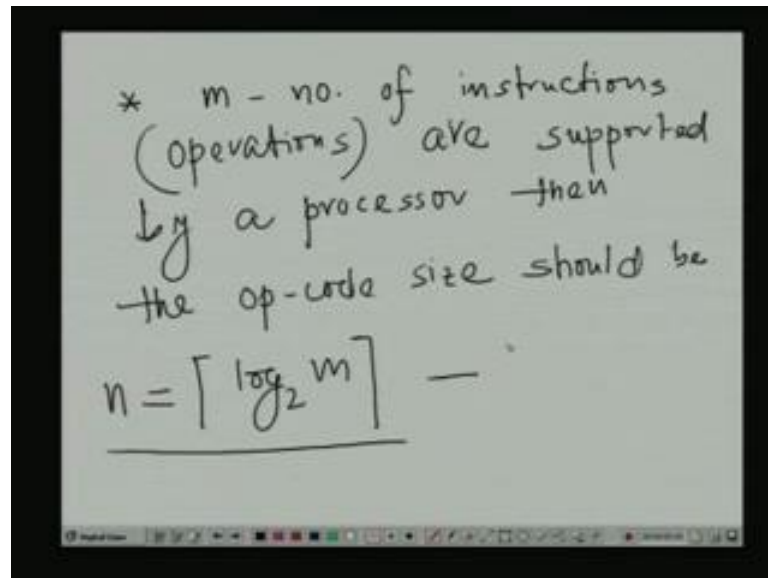
$2^3 = 8$
3-bit binary numbers

Now, if I want to design the instructions set; that means, I need some binary value to be assigned for each instructions. So, means say these binary value means this is the instructions. So, I have 8 instructions say, MOVE, CLEAR, ADD, SUB, AND, OR, INCREMENT, and JUMP and as there are 8. So, as 2 to the power 3, 2 to the power 3 is 8, so I need a 3 bit binary numbers to represent 8 instructions.

Now, this is the 3 bit op code, say I telling these are my i_0 , i_1 and i_2 , say this is my i_0 , i_1 , i_2 . Now, MOVE means 0 0 0; that means when the op code field contains a 0 0 0; that means, the computer understands that this is a nothing but, a MOVE instruction. Similarly, if the op code field contains a 0 0 1 then it is nothing but, a CLEAR means reset to 0, the register value is reset to 0, similarly 0 1 0 is ADD, 0 1 1 is SUB, 1 0 0 is AND, 1 0 1 is OR, 1 1 0 is INCREMENT and 1 1 1 is JUMP.

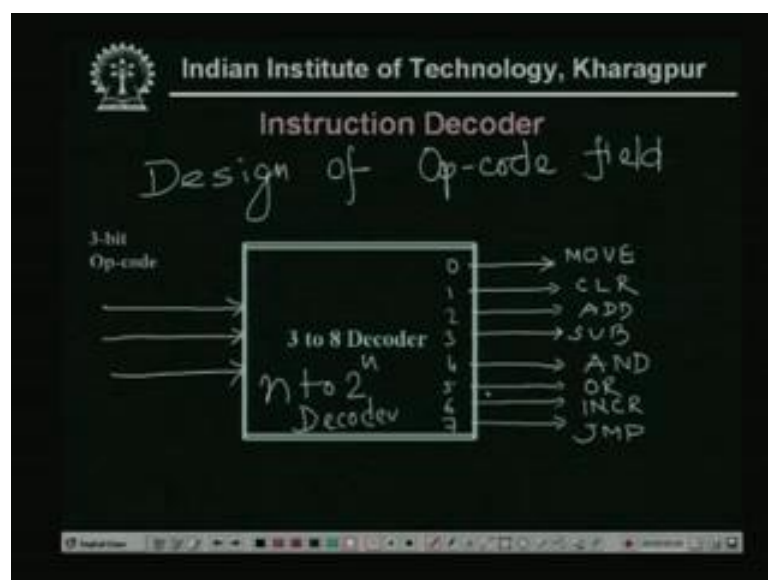
So, if more number of instructions allowed by a processor, say here it is a only 8 instructions. So, we know that 2 to the power 3 is 8, so log of 8 base 2 that many number of op code fields are needed.

(Refer Slide Time: 15:35)



So, if there are m number of instructions are allowed m number of instructions or the operations to be performed the operations are allowed or supported by a processor. Then, the op code size should be of log 2 base m, so this is the number of bits necessary for a op code. So, these are the in this way we can decide what would be the size of the op code field of the instruction set.

(Refer Slide Time: 17:09)



Now, we want to design this thing, so; that means, we need a hardware for this op code field. So, just the example we have shown the hypothetical example, we have 8 bit

instructions and we have seen we need 3 bit op code for that, so we have a 3 to 8 decoder is necessary, which has this 3 bit op code and there will be 8 lines for 8 instructions.

So, these are my 0, 1, 2, 3, 4, 5, 6, 7 that 2 to the power 3 values and 0 0 0 means, it was a MOVE instructions, then 0 0 1 means this is a CLEAR, 0 1 0 mean 2 that is a ADD, 3 is a SUBTRACT 1 0 0 is 4 that was an AND 1 0 1 is OR, this is a 6 means 1 1 0 is a INCREMENT and this was a unconditional JUMP 1 1 1. So, this is my design of op code field of instruction set, so that we need only a n to 2 to the power n decoder for supporting 2 to the power n number of operations or a n bit op code.

(Refer Slide Time: 20:20)

Indian Institute of Technology, Kharagpur
Op-code Encoding contd.

16-bit instructions

- 5-bit op-code field
- 11-bit address field

Op-code | Address

- 32 (2^5) operations can be specified which allow access to 2048 (2^{11}) memory locations
- If op-code field is 4-bit and address field increases to 12-bit

Handwritten notes on the slide:

- $2^4 \rightarrow 16$ operations with access to 4096 locations
- \rightarrow number of operations reduced to 50%, ($32 \rightarrow 16$)
- number of memory locations increased to 100% ($2048 \rightarrow 4096$)

So, op code encoding say I have a 16 bit instructions and I have designed like that, that 5 bit op-code field and so 11 bit address field. So, as 5 bit op code, so the operation supported are 2 to power 5 means 32 operations are specified and the operate on the data, which are stored on this 11 bit address field. So, by 11 bit addresses I can specify 2 to the power 11 means 2048 memory locations, I can specify 2048 locations. So, on these 2048 memory locations data can be stored and the operations to be performed on these data.

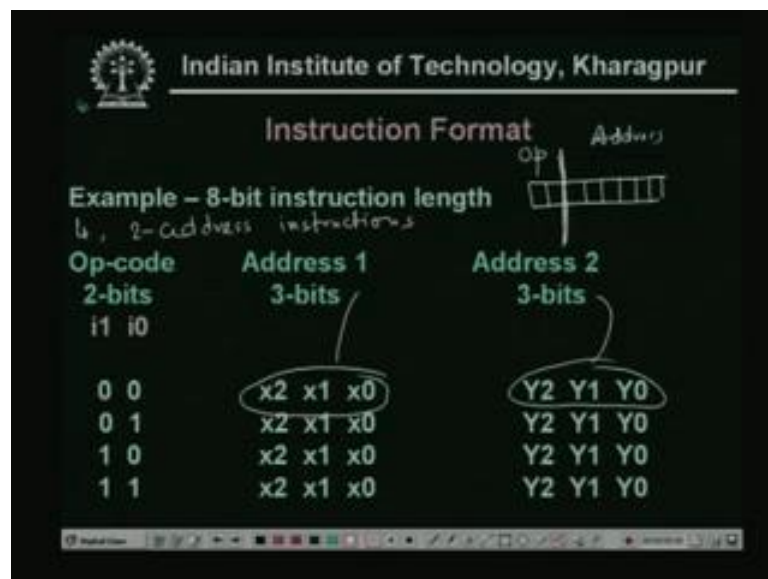
That means, the operations that are specified by these 5 bit op code, they access the data from these memory locations. Now, if op code field is 4 bit, initially it was 5 bit, now it is reduced to 4 bit, so my address field becomes 12s bit as the instruction set is a 16 bit. So, this is a 4 plus 12 16 bit because, I have the instruction set has to parts, one is op

code, one is address, so for 4 bit operations 2 to the power 4 is 16, 16 operations with access 2 to the power 12 means 4096 locations.

That means, the data stored on these 4096 locations are being operated by these 16 operations, the number of operations reduced to 50 percent. So, when it was 5 bit there are 32 operations can be supported, now the op code field is reduce to 4 bit, so there are 16 operations. So, the number of operations reduced from 32 to 16 means 50 percent reduction, now number of memory locations because, as the op code field decreases; obviously, the address field increases.

So, the number of memory locations increased to 100 percent because, earlier number of locations were only 2048 2 to power 11, now it becomes 4096 means 2 to the power 12. So, now it is increased to 100 percent double, now this concept is used in designing instructions with expanding op code technique, this is called the expanding op code technique.

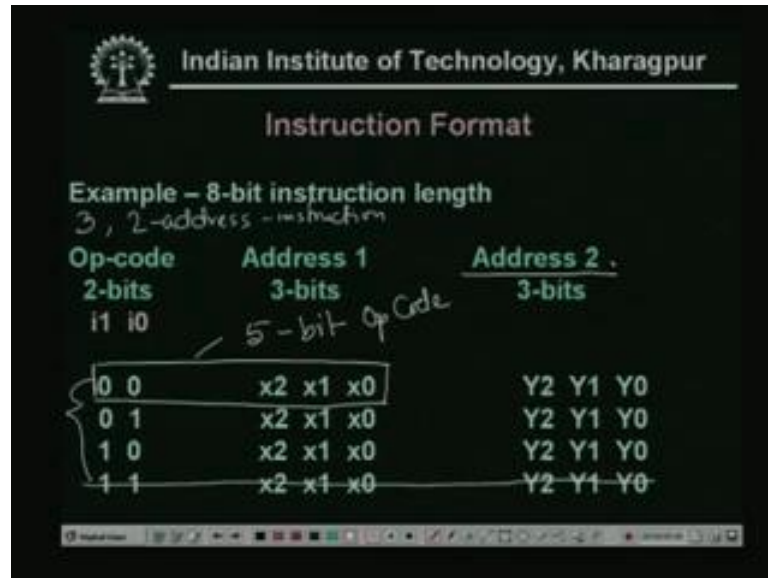
(Refer Slide Time: 24:52)



Now, we consider an instruction format with 8 bit instruction length and a 2 bit op code field. So, address field is 6 bits. That means, the full length is instruction set the full length is 8 bit, this is op code, this is address, so this is the thing, so op code 2 bits can be 0 0, 0 1, 1 0, 1 1. Now, the 6 bits your are dividing into 2 address field bits 3 bits 3 bits each, say we are representing this thing by x 0 x 1 x 2, Y 0 Y 1 Y 2. So, this is one address, address 1 Y 0 Y 1 Y 2 this is address 2.

So, this is 4, 2 address instructions, we call this is a 4, 2 address instructions, now how many different representations can be there with these 8 bit instruction length. So, this is remember this is one simple thing is a 4, 2 address instructions.

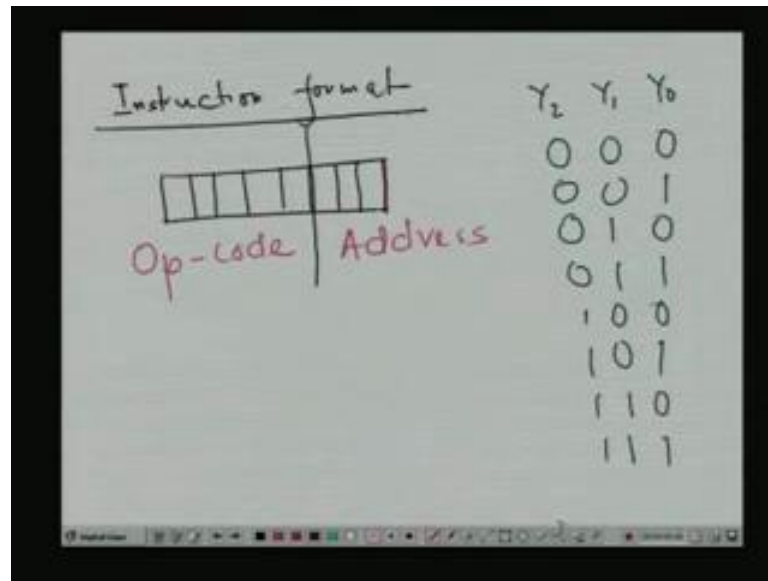
(Refer Slide Time: 27:03)



Now, what we do that whether it can be represented by a 3, 2 address instructions, so instead of 4, see we are considering 3 addresses. Say I am taking 0 0, as if the last one I am deleting, say I have only this 3, 2 address, instructions, now say I am representing this thing in a different way, as if say these up to op code bits and the first 3 address bits it is a 2 address. So, the first address we are taking and as if this becomes a 5 bit op code.

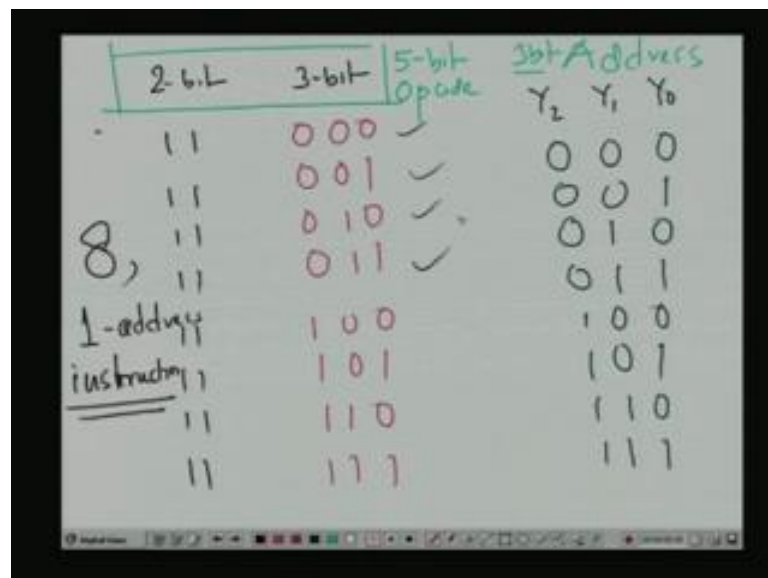
Then, only these 3 bits of address 2 that actually represents the address field of the instruction set. So, these 3, 2 address instructions I have modified that as if my op code field extends it is length up to the 3 bits of the address 1 and it becomes a 5 bit op code.

(Refer Slide Time: 29:38)



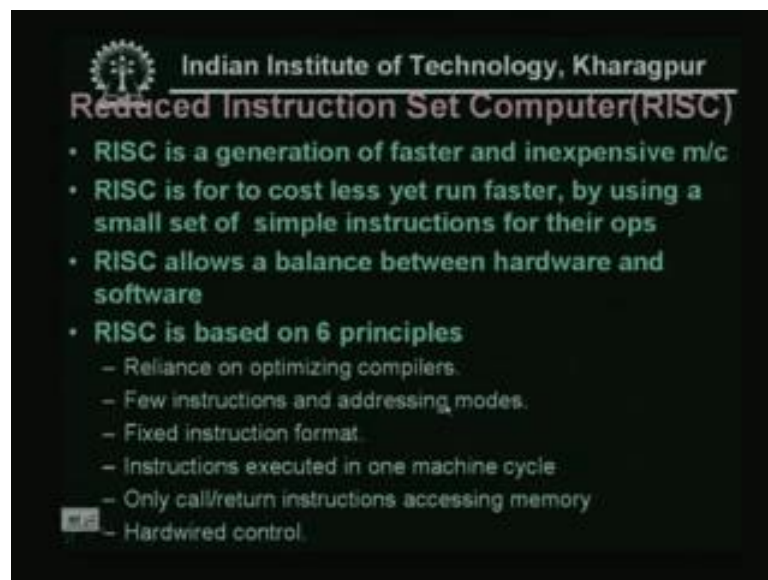
So, what it becomes now that actually my instruction format is like that, my instruction format will be say 1, 2, 3, 4, 5 up to these 5 this is my op code and say this 3 fields actually the address. So, this is the actually the address 2 of the previous one, so the address 2 of the previous one was the Y₀ Y₁ Y₂ fields. So, it has 000, 001 these are the 8 different values it can take, so for these 8 different values this 5 bit will can be of this of type.

(Refer Slide Time: 31:41)



Say I am giving that was the first 2 bit op-code and these were the actually the address one, the next 3 bit these becomes see as if this is always one 1 1, 1 1. And for these the next that this 3 bit of address one they are changing, so 0 0 1, 0 1 0, 0 1 1, 1 0 0, 1 0 1, 1 1 0, 1 1 1. So, this are my 5 bits op-code, these are my 3 bit address, 5 bit op-code. So, this is one instruction format, so 8, 1 address instructions we can tell this is 8, 1 address instructions because, there are 8 bit and this is a only one address 1 address 2 become only the 1 address. So, this is a 8 number of instructions and this is a 1 address instructions. So, earlier we have shown that this is a 3, 2 address instructions and here that can be modified or that can be represented as a different way that 8, 1 address instructions.

(Refer Slide Time: 34:07)



Now, we give one the most important instruction set or the microprocessor developed on these concept of the reduced instruction set, how they can be designed or what is the concept of that we do RISC a processor we discuss that thing. So, reduced instruction set computer, now RISC stands for Reduce Instruction Set Computer, this we can define as if this RISC is a generation of faster and inexpensive machine.

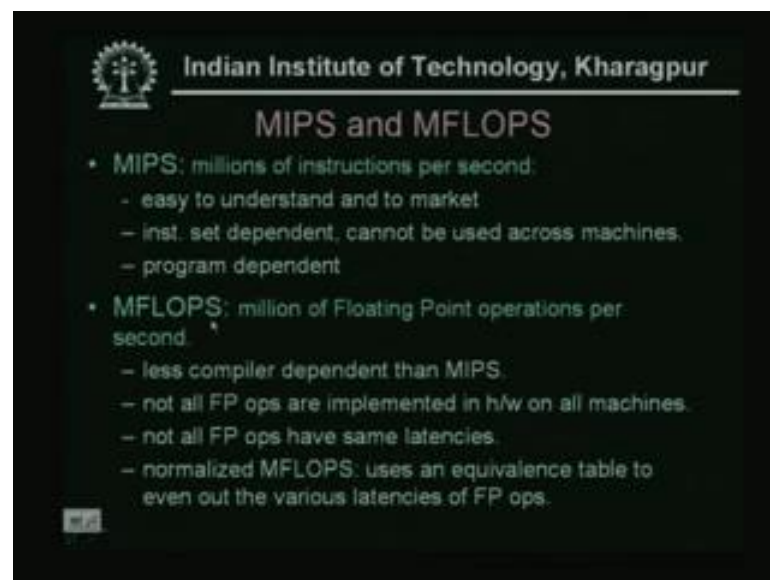
So, the main concept is or that is the basic principle for the evolution of RISC is that RISC is for to cost less yet run faster. Now, at the cost of what by using a small set of simple instructions for their operations, so mainly we are discussing the design of the

instruction set. And just now we have seen, that mainly that op code or the size op code that gives the number of operations or number operations to be performed.

So, by using a small set of simple instructions for their operations, that is the basic principle of this machine, which makes it faster as well as the less costly. Now, RISC allows a balance between hardware and software we will see, so mainly RISC is based on 6 principles, what are the principles, that reliance and optimizing compilers, few instructions and addressing modes RISC as its name implies it is few instructions. We will see later that what do we mean by addressing modes, that are also very less.

Fixed instruction format, it is not of a variable format like the, complex or the earlier old machines, complex instructions at machines or the old machines it has a fixed instruction format. Now, instructions executed in one machine cycle, only call or return instructions access, can access memory and this is totally hardwired control, as it is a hardwired control. So, this is the point that why it is faster because, as it is a hardwired control, this becomes much faster, so these are the six basic principles on which the this machine is developed.

(Refer Slide Time: 37:44)



The slide features the IIT Kharagpur logo and title at the top. It lists two performance metrics: MIPS and MFLOPS, each with a definition and several characteristics. MIPS is defined as millions of instructions per second and is noted as being easy to understand, but dependent on instruction sets and programs. MFLOPS is defined as million of floating point operations per second and is noted as being less compiler dependent, but not all FP ops are implemented or have the same latencies, leading to the use of normalized MFLOPS with an equivalence table.

Indian Institute of Technology, Kharagpur

MIPS and MFLOPS

- MIPS: millions of instructions per second.
 - easy to understand and to market
 - inst. set dependent, cannot be used across machines.
 - program dependent
- MFLOPS: million of Floating Point operations per second.
 - less compiler dependent than MIPS
 - not all FP ops are implemented in h/w on all machines.
 - not all FP ops have same latencies.
 - normalized MFLOPS: uses an equivalence table to even out the various latencies of FP ops.

Now, before we go in details the RISC instructions sets, we define the MIPS and MFLOPS, MIPS is Millions of Instructions Per Second, as we are discussing the computer design of computer instruction sets. So, how many instructions can be

executed per second or per unit time that is one very important thing because, mainly the speed of the processor it depends on that.

So, what we are telling that this machine is very is faster, so it depends this is some units that how many instructions can be executed per second. So, easy to understand and to market and instruction set dependent and this cannot be used across machines and this is only the program dependent. Another important thing nowadays is the MFLOPS, means the how many floating point operations, it can supports it is possible to execute the number floating point operations per second that we are defining as MFLOPS.

So, this is less compiler dependent that a MIPS because, it is a some floating point operations unit. So, whenever floating point operations will be there it only it will be executed by that particular unit not the compiler, not all floating point operations are implemented in hardware on all machines. But, if it is implemented in hardware then; obviously, that becomes or that makes the computer faster not all floating point operations have same latencies. Obviously, different operations have different complexities, so it will take different time. So, normalized MFLOPS uses an equivalence table to even out the various latencies of floating point operations.

(Refer Slide Time: 40:15)

Indian Institute of Technology, Kharagpur

RISC Vs CISC

- CISC (complex instruction set computer)
 - VAX, Intel X86, etc.
- RISC (reduced instruction set computer)
 - MIPS, DEC Alpha, SUN Sparc, IBM 801

Data — Register / main memory
Addition ← op1 op2

Now, always the trend was the complex instruction set computer, now, because of their complexity more hardware would have to be used for CISC the Complex Instruction Set Computer. However, because of the more instructions, the more hardware logic is

needed to implement and support them, it has more instructions, so hardware will be more. Say for an example, an RISC machine and ADD instruction takes the data from register only, as I mentioned that the data can be stored either in registers or the main memory.

Now, register means it has some fixed say binary values, it is some binary numbers that is stored in a register. Now, the RISC machine can take only the data say I am using the addition, so it has two operands op 1 and op 2 and this means 2 value 2 data it is taking from the register. Whereas, the CISC supports because, it is a complex instruction set say the data can be of in different form, so when the data is accessed first it has to be decided that in which form the data is...

So, it has to take some decision and some if it is by hardware, then some complex hardware is needed just to decode the form of the data. So, complexity will increase, hardware will increase, the speed will less because it is and it becomes expensive, so that is the main difference from the complex instructions set computer and why that reduced instruction set computers have evolved. Earlier, this where the some of the examples of CISC machines, the VAX, Intel X86 and nowadays all the machines are of RISC.

There are MIPS machine, DEC Alpha, the SUN Sparc, the IBM machines these are normally the all RISC machines that are nowadays the machines available that are all RISC. Initially, now they are only the power PC's or the desktop work stations are where the RISC machines, but nowadays that all computers are that of RISC type.

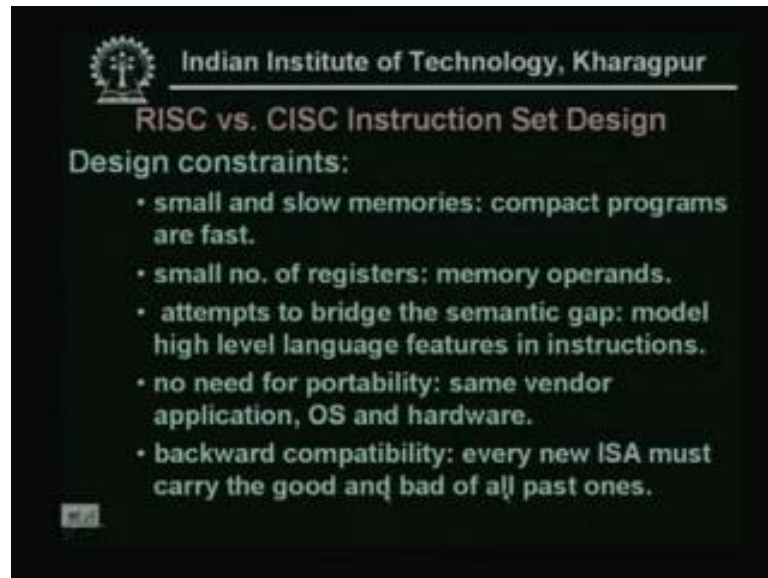
(Refer Slide Time: 43:41)

CISC	RISC
Variable length instruction	Single word instruction <i>Fixed</i>
Variable format	Fixed-field decoding
Memory operands	Load/store architecture *
Complex operations	Simple operations

So, if we summarize the comparison between the CISC and RISC, so it has some the main differences lies, the CISC has variable length instructions. Whereas, this is fixed instructions we call the single word instructions or we can tell this is a it has a fixed instruction format. Similarly, the variable format and the fixed field fixed field decoding the format is fixed, now the CISC is memory operands, means the operands are data just now I told that stored in the memory.

So, the operations to be performed on the data, the data are accessed from the memory and these are load store architecture. So, this is the main thing or this is the RISC is a load store architecture, so here the all the data are stored on the register, so the operations to be performed on the data are accessed from the register. It has some complex operations and these are some simple operations, so this is the main differences lies between the design of CISC and the RISC.

(Refer Slide Time: 45:35)



Indian Institute of Technology, Kharagpur

RISC vs. CISC Instruction Set Design

Design constraints:

- small and slow memories: compact programs are fast.
- small no. of registers: memory operands.
- attempts to bridge the semantic gap: model high level language features in instructions.
- no need for portability: same vendor application, OS and hardware.
- backward compatibility: every new ISA must carry the good and bad of all past ones.

Now, we consider the instruction set design because, we are mainly discussing on the instruction set. So, what are the small the design constants, so small and slow memories compact programs and fast and there are fast because, if it is a small memory then the accessing will be the maximum access is restricted by the size of the memory, as it is a very small. So, only less number of accesses are needed, so it will be fast.

Small number of registers because, as all ready we mentioned the RISC operands that is the one concept the data are stored on the registers. Now, attempts to bridge the semantic gap the model high level language features in instructions, no need for portability the same vendor application OS and hardware. And the backward compatibility every new ISA the Instruction Set Architecture must carry good and bad of all past ones.

(Refer Slide Time: 47:02)

Indian Institute of Technology, Kharagpur

RISC vs. CISC Instruction Set Design

Design constraints:

- powerful and complex instructions that are rarely used.
- IC technology and microprocessors in 1970s: lower costs, low power consumption, higher clock rates, cheaper and larger memories.

So, powerful and complex instructions that are rarely used, IC technology and microprocessors in 1970's they are of lower cost, low power consumption, higher clock rates, cheaper and larger memories. So, mainly to utilize this advancement of technologies the RISC's are evolved.

(Refer Slide Time: 47:31)

Indian Institute of Technology, Kharagpur

Top 10 80x86 Instructions

Rank	Instruction	Integer Average Percent total executed
1	load	22%
2	conditional branch	20%
3	compare	16%
4	store	12%
5	add	8%
6	and	6%
7	sub	5%
8	move register-register	4%
9	call	1%
10	return	1%
	Total	96%

Simple instructions dominate instruction frequency

Now, another very important concept is there in the relation of the RISC, so one processor have some n number of instructions or the operations allowed. Now, the research have been done that what are the frequencies of usage of different type of

instructions. See, here one example is given, see some instruction these are the instructions, see this is a load and the average percent of the total execution it has been computed that 22 percent of the total execution is a load instructions.

Now, a conditional branch means if a condition is satisfied it goes to some address, this type of instructions are executed 20 percent of the total execution. Similarly, a compare has 16 percent execution, store is 12 percent, add is 8 percent, and is 6 percent, a subtract is 5 percent, see add is more addition is more than subtraction, move register to register it is only 4 percent. A call routine is only 1 percent and return is 1 percent, total say 96 percent there are many miscellaneous other instructions that the rest 4 percent are of that type.

Now, on these observations or these research we will see that actually the load is 22 percent; that means the maximum execution or the maximum percentage of execution is a load instructions. See store is of 12 percent it is of actually... so now people can be made all operations whether can be made or the design can be made of load store type, using load store. Only the whether the instructions can be developed, all other instructions can be developed based on these two.

So, this is a very important thing this is a some top 10, 80, 86 instruction are observed and the percent of execution has been computed and it has been seen that the load and store has a has more number has greater percentages used. So, this is another important point that why the it is based on the or why it is a load store architecture.

(Refer Slide Time: 51:27)

Indian Institute of Technology, Kharagpur

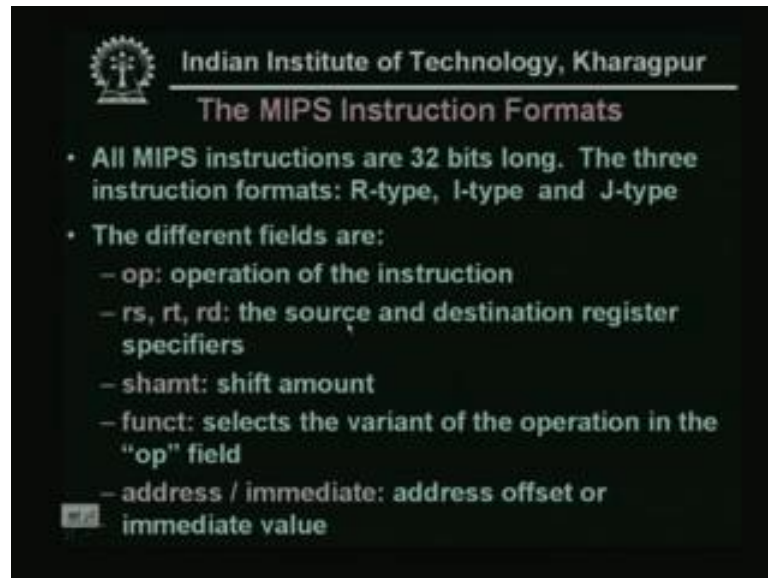
RISC vs. CISC Instruction Set Design

- Emergence of RISC
 - Very large scale integration (processor on a chip).
 - Micro-store occupies about 70% of chip area; replace micro-store with registers ==> load/store ISA.
 - Increased difference between CPU and memory speeds.
 - Complex instructions were not used by new compilers.
 - Software changes:
 - reduced reliance on assembly programming, new ISA can be introduced.
 - standardized vendor independent OS (Unix) academia and research – need for portability
 - Early RISC projects: IBM 801 (America), Berkeley SPUR, RISC I and RISC II and Stanford MIPS.

So, if we summarize that instructions set design, so the very large scale integration; that means, for the advancement of the VLSI that chip can be of small size. And then micro store occupies 70 percent of chip area, so replace micro store with registers the load store. So, more resistors can be implemented, increase difference between CPU and memory speed, complex instructions were not used by new compilers and software changes are reduced reliance on assembly programming, new instruction set architecture can be introduced, standardize vendor independent of operating system and the academics and research they need for portability.

So, early RISC projects the IBM 801 the Berkeley RISC 1 and RISC 2 and the Stanford MIPS they are developed or some research is based on that preliminary machines based on the RISC instruction sets.

(Refer Slide Time: 52:46)



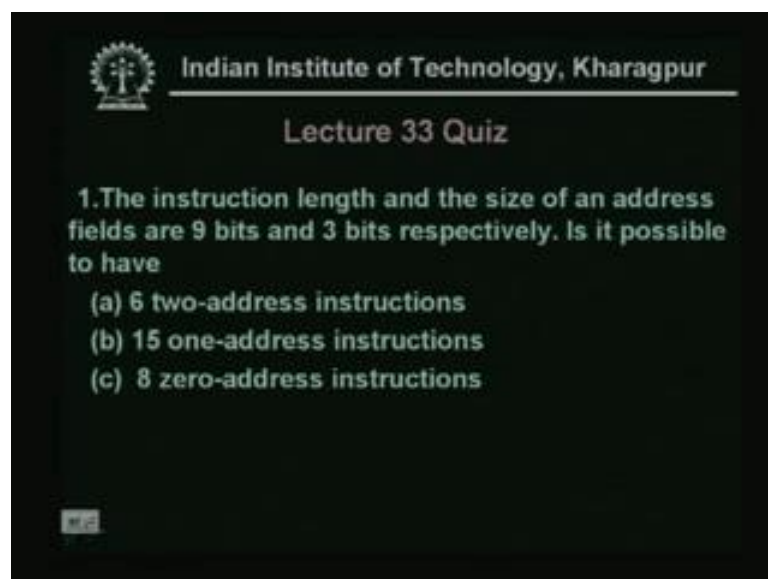
Indian Institute of Technology, Kharagpur

The MIPS Instruction Formats

- All MIPS instructions are 32 bits long. The three instruction formats: R-type, I-type and J-type
- The different fields are:
 - op: operation of the instruction
 - rs, rt, rd: the source and destination register specifiers
 - shamt: shift amount
 - funct: selects the variant of the operation in the "op" field
 - address / immediate: address offset or immediate value

Now, one we are taking one example that is a the MIPS machine, the MIPS instruction formats or what we can do that there is actually ((Refer Time: 53:13)) this if we take one example of any one of this RISC machine, say that Stanford MIPS or our Berkeley RISC machine, Berkeley RISC machine. We will discuss actually how the RISC machines are or the designed or the how the instruction set of this RISC machine is design, what is the formatting of the instruction set, etcetera.

(Refer Slide Time: 54:08)



Indian Institute of Technology, Kharagpur

Lecture 33 Quiz

1. The instruction length and the size of an address fields are 9 bits and 3 bits respectively. Is it possible to have

- (a) 6 two-address instructions
- (b) 15 one-address instructions
- (c) 8 zero-address instructions

Now, we will see the quiz of this class, so we have discussed the mainly the instruction set and the instruction format and the op code encoding particular the expanding op code technique. So, based on that our quiz question is, so the instruction length and the size of an address fields are 9 bits and 3 bits respectively, is it possible to have 6 two address instructions, 15 one address instructions and 8 zero address instructions.

So, one instructions length and the address bits are given, then whether we can construct the instructions such that it can specify these many this type of instruction sets. See this is the quiz questions of the lecture 33.

Digital Systems Design
Prof. D. Roychoudhury
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 34
Design of Computer Instruction Set and the CPU (Contd.)

This class we will continue the discussion on the Design of Computer Instruction Set.

(Refer Slide Time: 55:44)



So, we have seen the how what is the format of a computer instruction, what are the two parts or two fields of a instruction and how the op code about the instruction op code is encoded. And then we introduced the reduced set instruction computer and what are the different RISC machine, so far developed. Now, already I mentioned that now a day's almost all the high performance computers are the RISC machines.

So, just to see what are the basic architecture of RISC, some early machines developed based on RISC we will see that thing. So, some early RISC projects are IBM machine and the Berkeley RISC 1 and RISC 2 machines and Stanford MIPS, so this class we see the as an example the MPIS architecture.

(Refer Slide Time: 57:12)

Indian Institute of Technology, Kharagpur

The MIPS Instruction Formats

- All MIPS instructions are **32 bits long**. The three instruction formats: R-type, I-type and J-type
- The different fields are:
 - op: operation of the instruction
 - rs, rt, rd: the source and destination register specifiers
 - shamt: shift amount
 - funct: selects the variant of the operation in the "op" field
 - address / immediate: address offset or immediate value

So, the MIPS instruction formats that all MIPS instructions are 32 bit long, see this is a instructions are 32 bits. The 3 instructions formats are R-type, I-type and J-type. So, it supports 3 different instruction formats and the different fields are op means operation of the instructions. That means, which operation are supported or which operations are currently being executed by the instruction, r s, r t, r d the 3 source and destination register specifiers.

Then, shift amount shamt is a shift amount, that is a different field kept in MIPS, another is funct, it selects the variant of operation in the op field. Then, address and immediate; that means, address offset or immediate value; that means, the data or this is some kind of mode of operations, that whether it is immediate value the is being operated on or the address is given or address offset and from some addresses the data are to be accessed before it operates.

Thank you.