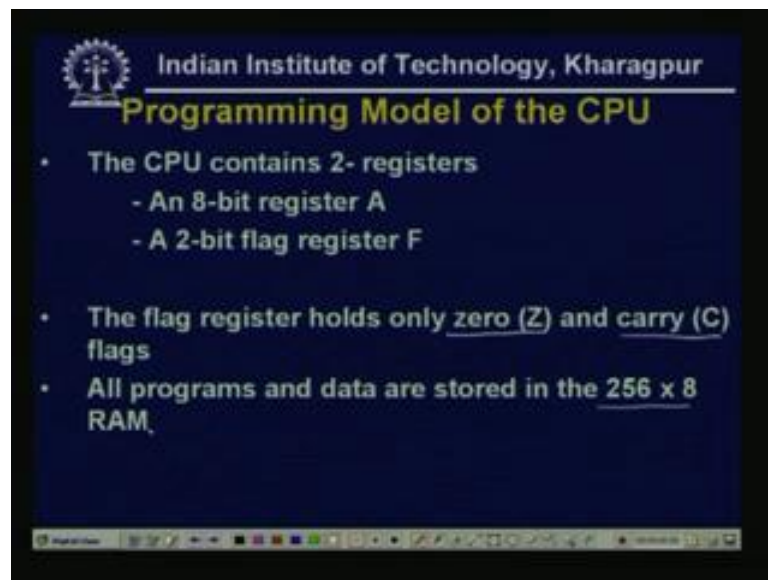


Digital Systems Design
Prof. D. Roychoudhury
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 39
Design of a Micro-Programmed CPU

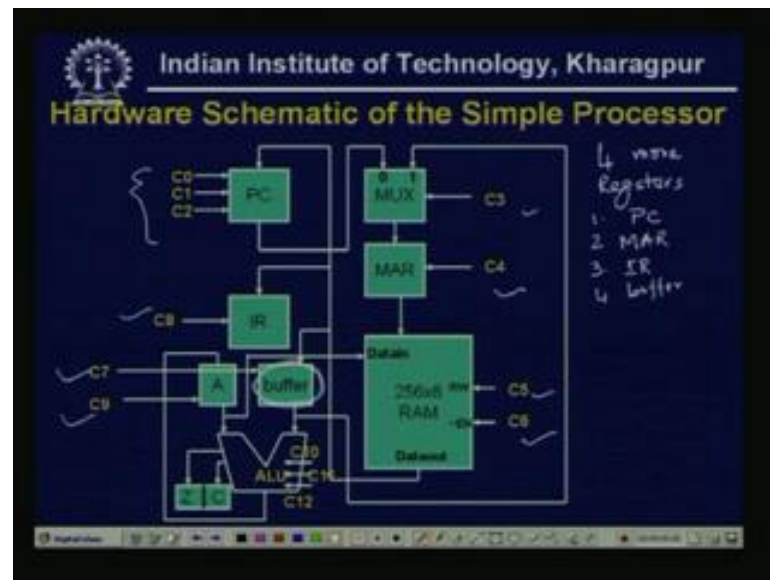
Already, we have read the design of a CPU mainly the two parts. We have read the design of ALU and the design of control unit, there are two approaches the hardware control unit and the micro programmed control unit. Now, this class we will read a Design of a Micro Programmed CPU.

(Refer Slide Time: 01:20)



So, now first we assume the programming model of the CPU, the CPU that we want to design contains two registers, an 8-bit registers A, a 2 bit flag resistor F, the flag register holds only 0 and carry C flags. All programs and data are stored in a memory and the memory size is a 256 by 8 bit RAM, now the first we see the hardware schematic of the data flow part of this processor.

(Refer Slide Time: 02:23)



So, it is the Program Counter or PC, MAR the Memory Address Register, IR the Instruction Register and buffer, so these four more registers are also there. So, in addition with A and F, the four more registers are PC Program Counter, Memory Address Registers MAR, Instruction Register IR and a buffer. These registers are transferring to a programmer that, 8 bits registers buffer is used to hold the data that is, retrieved from the memory.

So, this is the buffer, it will contain the data retrieved from the memory, only a restricted number of data pass are available in this system. And these pass are controlled by the control inputs C 0 through C N, so here this C 0, C 1, C 2 inputs, then C 3, C 4, C 5, C 6, C 7, C 8, C 9 and C 10, C 11 and C 12. These are the control lines for the ALU operations, so this 13 control lines are there in this for the simple processor.

(Refer Slide Time: 04:48)

Micro-operations	Comments
$C0: PC \leftarrow 0$	✓ Clear PC to zero
$C1: PC \leftarrow PC+1$	✓ Advance the PC
$C2C5\bar{C6}: PC \leftarrow M(MAR)$	✓ Read the data from memory and save it in PC
$\bar{C3}C4: MAR \leftarrow PC$	Transfer the contents of PC into MAR
$C5\bar{C6}C7: BUFFER \leftarrow M(MAR)$	Read the data from memory

Now, we define the control inputs first C 0 to C 9 the these are the micro operations, what are the micro operations for these control signals and what are the meaning of those micro operations. So, first one is C 0, the C 0 is micro operation is 0 assign to PC or PC is reset or PC is clear, so clear PC to 0, so C 0 is clear PC to 0. The control signals C 1 activates the operation, which is defined as PC is PC plus 1 means the program counter value is incremented by 1.

Or in other way we can tell that advance the program counter, this is C 1, now C 2 C 5 C 6 bar C 2 C 5 C 6 complement, it is defined as memory address register. The content of M of MAR means, the content of Memory Address Register is assign to PC what does it mean, that read the data from memory and save it in the PC. See here, the control inputs are C 2 C 5 C 6 bar, ((Refer Time: 06:37)) so if we see that C 2 C 5 and C 6 complement, so C 2 will be this is 1, C 5 is 1 and C 6 is 0 that means, C 6 complement is 1. Then the M of MAR Memory Address Register, that will go to the program counter, that is assign to program counter.

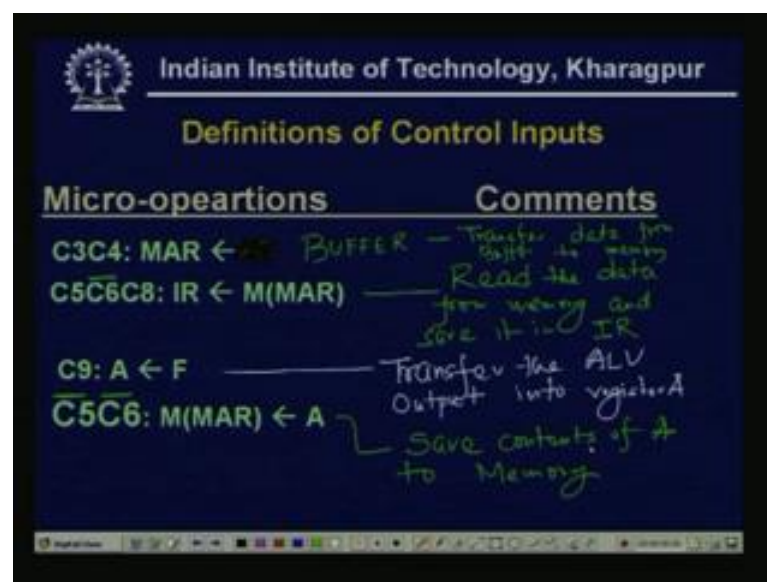
So, it will this line will be activated and that PC value will go, now next one is C 3 complement, C 4 bar. So, this is C 3 complement control line is C 3 complement C 4 and operation is the program counter, value of program counter will go to memory address register. That means, transfer the here, transfer the content of the PC that means, the Program Counter contents of PC into memory address register.

If we see that $\overline{C3} C4$ means, this is the $C3$ complement one input of the MUX and $C4$ is the memory address register, so this two will be high. Then, PC value will go to MAR, see this is 0, this will be activated if $\overline{C3}$ bar, if it is $\overline{C3}$ bar, if it $\overline{C3}$ complement then, this value will come again. $C4$ is 1 that means, this value is registered to MAR. So, that operation is define by, PC to MAR the control signal is $\overline{C3}$ complement $C4$.

The next one is $C5 \overline{C6}$ complement $C7$ and here the content of memory address register goes to buffer. So, read the data from the memory operation is read, the data from memory and save the result, save it in buffer, control signal is $C5 \overline{C6}$ complement $C7$, we see that thing in the picture, $C5$ is it is a, $C5$ it is 1. So, this line is 1 read then, $\overline{C6}$ complement, $\overline{C6}$ complement is enable line if it is 0 then, it will be enable, so $\overline{C6}$ complement will be enable line.

Then, it will be a read, read enable then $C7$, $C7$ is this buffer is activated, $C7$ is 1 means this buffer is activated. So, the value the, this value the memory, the data from the memory is stored in the buffer. This is this line will be activated, so M of MAR is assign to buffer.

(Refer Slide Time: 11:27)



Micro-operations	Comments
$C3C4: MAR \leftarrow PC$	BUFFER - Transfer data from PC to MAR
$C5C6C7: IR \leftarrow M(MAR)$	Read the data from memory and save it in IR
$C9: A \leftarrow F$	Transfer the ALU output into register A
$C5C6: M(MAR) \leftarrow A$	Save contents of A to Memory

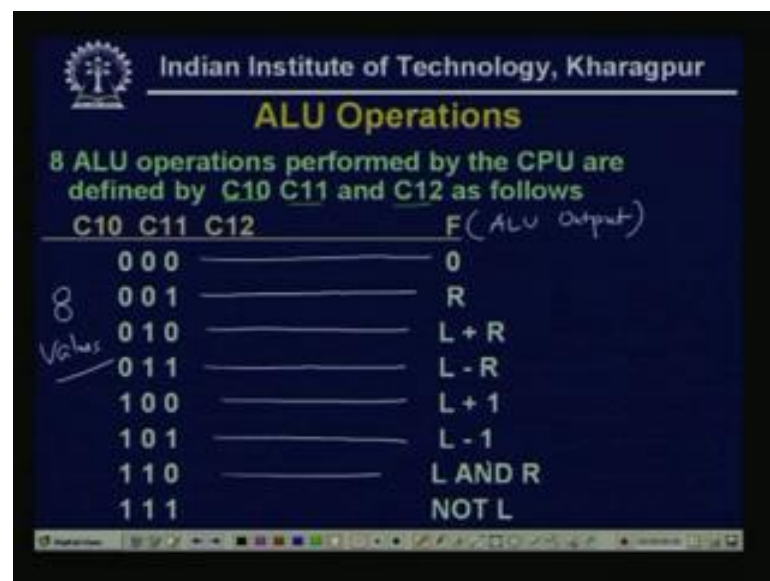
Then, $\overline{C3} C4$ if we see $\overline{C3} C4$, $\overline{C3} C4$ is buffer or $\overline{C3} \overline{C3}$ complement $C4$, first we see $\overline{C3}$ complement $C4$ then, the PC goes to MAR. Then, transfer the contents of the PC into MAR, if it is $\overline{C3} C4$ if we see, if it is $\overline{C3} C4$ then, it will be buffer to MAR, this

will not be a PC. This is, this will be a buffer, buffer to MAR and the control will be C 3 C 4 then, C 5 C 6 complement C 8.

This is memory, the data of memory address register goes to instruction register, so this one is read the data from the memory M of MAR means, read the data from memory and save the, save it in Instruction Registers IR. Here, it is transfer, transfer the buffer, transfer data from buffer to memory, Memory Address Register MAR. Now C 9, C 9 is F to A means, this is a transfer the ALU, the ALU output into register A.

Now, C 5 complement C 6 complement, if the control signal is C 5 complement and C 6 complement then, save contents of register A to memory that is, M of MAR to memory. So, these are the nine micro operations that are, or now by, define by nine control signals, these are the nine micro operations defined, for this example CPU.

(Refer Slide Time: 15:54)



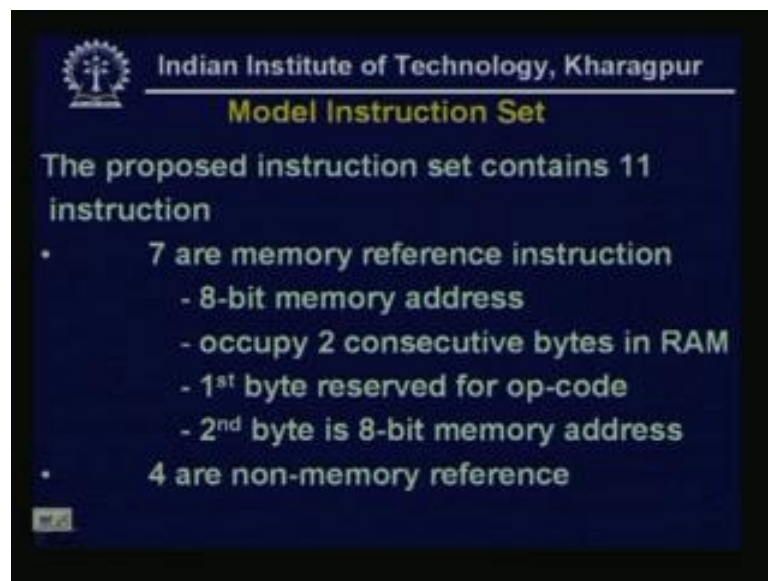
C10	C11	C12	F (ALU output)
0	0	0	0
0	0	1	R
0	1	0	L + R
0	1	1	L - R
1	0	0	L + 1
1	0	1	L - 1
1	1	0	L AND R
1	1	1	NOT L


Now, we see the 8 ALU operations perform by the example CPU, that are define by three control signals C 10 C 11 and C 12. So, C 0 to C 9 already we have defined and now, we see the eight control 8 ALU operations, by this three control signals. So, the C 10 C 11 C 12 we can take 0 0 0 to 1 1 1, this eight possible values these are some, these are the eight values.

Now, for each three bits, that one ALU operations are defined, see 0 0 0 means ALU output is 0, this is F is the ALU output, so F is, F is ALU output. Now, if it is 0 0 1 then,

ALU output is simply the, the value of register R, if it is 0 1 0 then, it is defined as L plus R, if it is 0 1 1, it is L minus R, 1 0 0 L plus 1, 1 0 1 is L minus 1, 1 1 0 is L AND R. This is a logical operation and this is a logical AND operation and 1 1 1 means not L that means, complement, again this is again logical. So, these eight operations are ALU operations are define for this example CPU. Now, what are the instruction set implemented by this model.

(Refer Slide Time: 18:03)



 Indian Institute of Technology, Kharagpur

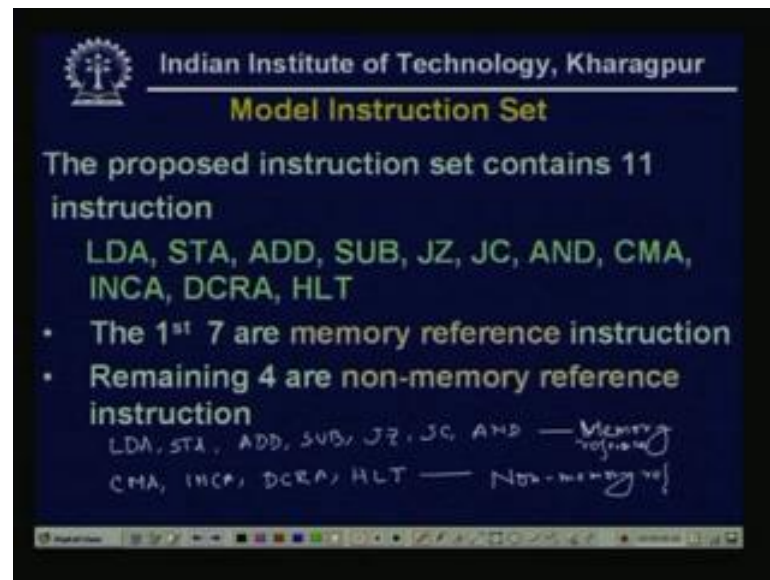
Model Instruction Set

The proposed instruction set contains 11 instruction

- 7 are memory reference instruction
 - 8-bit memory address
 - occupy 2 consecutive bytes in RAM
 - 1st byte reserved for op-code
 - 2nd byte is 8-bit memory address
- 4 are non-memory reference

So, the proposed instruction set contains eleven instructions. These are this eleven instructions.

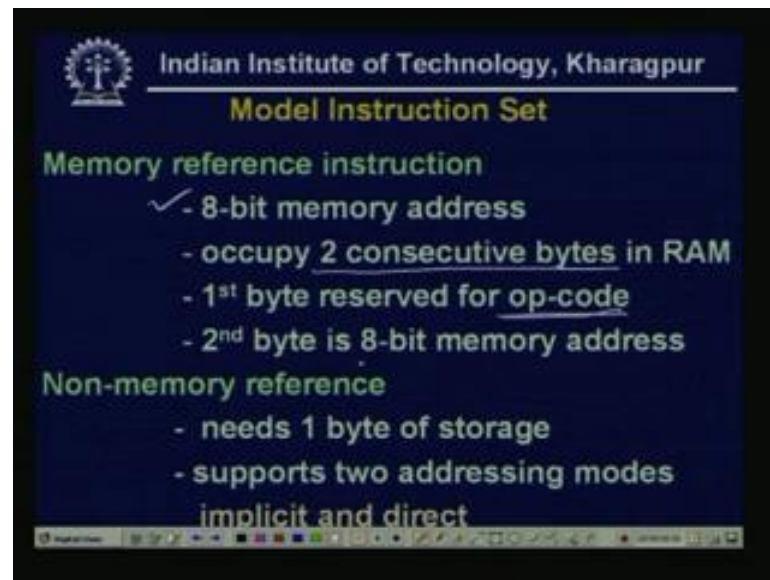
(Refer Slide Time: 18:24)



The LDA, then STA, ADD, subtract, JZ means Jump on 0, JC Jump on Carry flag set and logical CMA, this is Complement of Register A increment, decrement and halt, later again, we will see this thing in details, so this eleven instructions are implemented. Now, here the first seven are memory reference instructions and remaining four are non-memory reference instructions. That means, that LDA, STA, store ADD, subtract jump on zero, jump on carry and the logical AND, these are memory reference instructions.

Memory reference instruction means, they all require a memory address, so they are memory reference instructions. But, the remaining four, the complement, the increment, the decrement and the halt they do not need any memory address, so they are non-memory reference. Normally, all instructions can be divided into these two categories, these are non-memory reference. Now the memory reference instructions are those instructions.

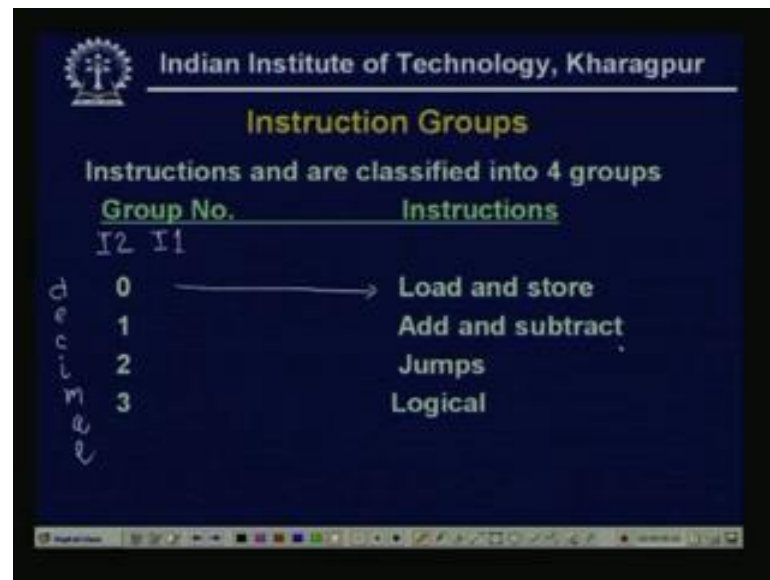
(Refer Slide Time: 20:52)



As I defined, that they need a memory address in this example, we have assumed that then they need an 8 bit they occupy 8 bit memory address. They occupy, the memory reference instruction occupy two consecutive bytes in RAM, the first byte reserved for op code and the second byte is 8 bit memory address. So, op code and address, these two are, are the two content of the two bytes occupied by memory reference instruction.

Now, what about the non-memory reference instruction, they need only 1 byte of storage because, they do not need any memory address. So, this part is off, the second byte is not needed, only the first byte that is the op code, the op code is needed, no memory address is involved here. So, as this is a non-memory reference. Now, this supports two addressing modes, one is implicit and another is direct.

(Refer Slide Time: 22:25)



Indian Institute of Technology, Kharagpur

Instruction Groups

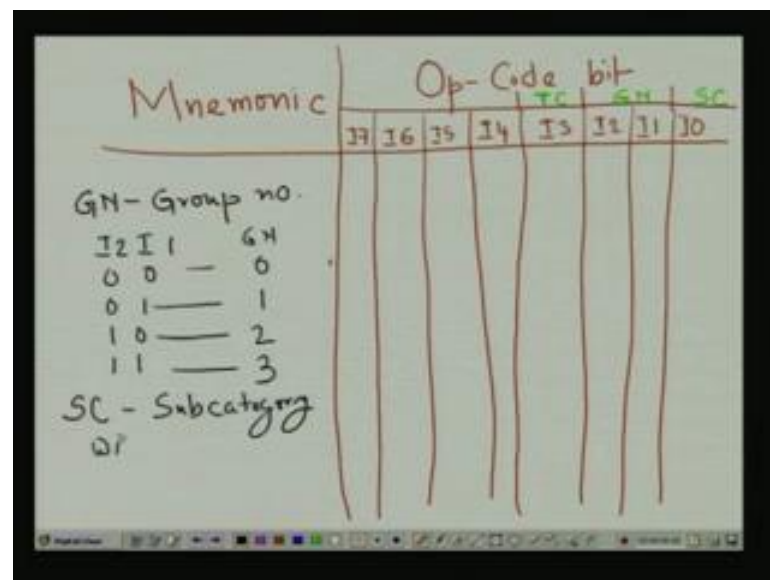
Instructions are classified into 4 groups

Group No.	Instructions
0	Load and store
1	Add and subtract
2	Jumps
3	Logical

reversed

Now, the op code encoding for this instruction set is carried out in a logical manner. Now, we see that how this op coding, op code instruction, encoding can be done.

(Refer Slide Time: 22:47)

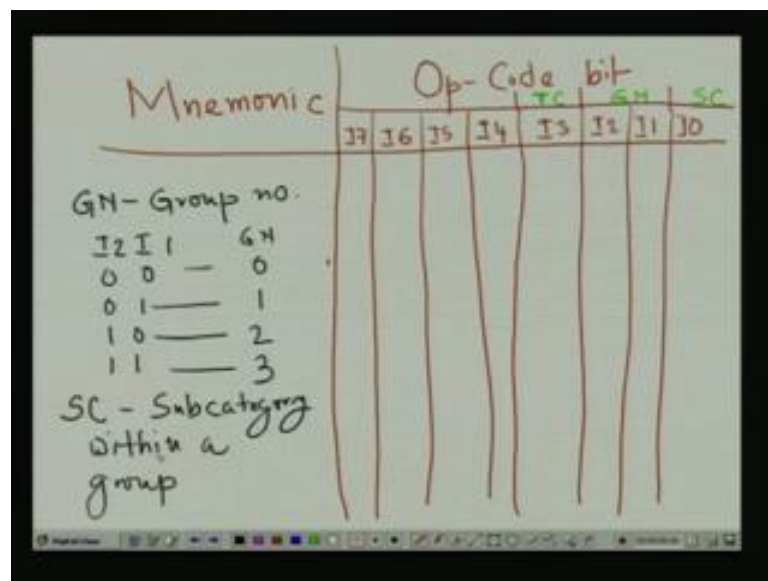


Mnemonic	Op-Code bit
	17 16 15 14 13 12 11 10
GN - Group no.	
I2 I1 GN	
0 0 - 0	
0 1 - 1	
1 0 - 2	
1 1 - 3	
SC - Subcategory	
I0	

Say we have eleven instructions defined for this example CPU, and these instructions in short form, we called mnemonic, so these are the, we write the mnemonics, then the op code bit. Now, we know op code bit are of 8 bit. So, I 7, then I 6, I 5 and I 4, I 3, I 2, I 1 and I 0, these are the 8 bits, again this I 3 we call this is the TC part, I 2 I 1 is GN and I 0 is SC.

TC is the type classifier, so actually I 3, I 3 will give, whether it is a memory reference instruction or it is a non-memory reference instruction. Just now, we have seen there are two type of instruction, Memory Reference MRI or NMRI. So, if I 3 equal to 1, then it is see, if I 3 equal to 1, then it is a MRI otherwise; that means, I 3 equal to 0, it is NMRI and that is defined as a, the TC the Type Classifier, TC is Type Classifier. So, this is, this is TC Type Classifier now, GN is the Group Number within a type.

(Refer Slide Time: 26:44)



Now, GN, GN is Group Number, GN is Group Number within a type and it has two field two bits I 2 I 1, so I 2 I 1 can take four value 0 0 0 1 1 0 1 1. Now, I 2 I 1, then group number is 0, 0 1 means 1, 1 0 is 2 and 1 1 means 3. So, there are four groups for each type of instruction, four groups of each type of instructions; that means, if I 3 equal to 0, I 2 I 1 can take four values, 0 1 to 3.

If I 3 equal to 1 MRI, then it is group, group can be of four groups, again 0 1 to 3 and SC is Sub Category within a group. So, SC is a Sub Category within a group, see in this way the I 0 I 1 I 2 I 3, this four values can be set in the, for a op code bit this can be design in this way. Now, the I 7 I 6 I 5 I 4 values are can be, so we take some of the examples, say first I take the load instructions mnemonic is LDA.

(Refer Slide Time: 28:52)

Op-Code Mnemonic		Logic Encoding Op-Code bit							
		I7	I6	I5	I4	I3	I2	I1	I0
1.	LDA	0	0	0	0	1	0	0	0
2.	STA	0	0	0	0	1	0	0	1
3.	ADD	0	0	0	0	1	0	1	0
4.	SUB	0	0	0	0	1	1	0	0
5.	JZ	0	0	0	0	1	1	1	0
6.	JC	0	0	0	0	1	1	1	1
7.	AND	0	0	0	0	0	0	0	0
8.	CMA	0	0	0	0	0	0	1	0
9.	INCA	0	0	0	0	0	0	1	1
10.	DCRA	0	0	0	0	0	1	1	0
11.	HLT	0	0	0	0	0	1	1	1

For LDA, we can say these are all zero values, as this is a MRI, so I 3 equal to 1, this is a Memory Reference Instructions and the remaining are 0 0; that means, this is a group 0, subcategory also 0. Similarly, for STA, STA again four 0, this is store again this is a MRI and now, I can give 0 0 1, this is subcategory 1. Then ADD, ADD also a MRI, so I put 1, again these are all I 4 to I 7 all zero values and this is a 0 1 0.

So, this is a different group, load and store are in group 0, ADD is in group 1 because, I 2 I 1 is a group number. So, 0 1 means this is a group 1, SUB is, SUB is also subtract the same way, this is 0 0 0 0 0, this is MRI. Obviously, this will be in the same group and this is a subcategory is different. Similarly, the Jump on Zero JZ, we can define these are all MRI, as the first seven instructions are MRI.

So, this is now I 2, I 2 is 1, so this is a group 3, now jump on carry, again the same way I can tell, that this is a 10 group, subcategory is 1. Now, this is a AND, the logical AND, again this is a MRI, this is a group 3 and subcategory 0, so up to this, they are MRI type of instructions, the remaining four instructions were. So, these remaining four instructions were complement of register A increment, decrement and halt, these are non- memory reference.

So, first what we, that non memory reference means, I 3 this becomes 0, so these four we can put these are 0 and similarly I can give the other values the I 2 I 1 I 0. These are 0 0 0, these are 0 1 0, then decrement is 1 0 0 and halt is 1 1 0 and the remaining all values

are 0. So, in this way we can encode the op code logic. So, this is my op code logic encoding, this table gives me the op code logic encoding.

In this way, for these eleven instructions, for these eleven instructions I can implement I can design my 8 bit op code, these are the meanings of the bits. So, the bit I 3 of the op code decides the instruction type, if I 3 equal to 1 this is MRI, I 3 equal to 0 it is NMRI. Now, within the memory reference category, instructions are classified into four groups.

Just now we have seen, the I 2 I 1 value, so these are my I 2 I 1 group number, that is in the op code, these are my I 2 I 1. And these are the decimal value, these are my decimal values, so group number 0 1 2 3, already we have seen. And if it is 0 that means, this is a load and store, that is why we have put 0 there, 1 means ADD subtract. Just now we have seen, 2 means JZ Jump on Zero and jump on carry, 3 is AND, we have seen the logical operations.

So, in this way we have define the group number of the instructions, now another thing is, there are two instructions in the first groups. So, bit I 0 is use to determine the desired instruction of a particular group, now the, and the in the same groups say STA. That means, LDA is 0 STA is 1. In this way, we can similarly ADD subtract they are in the same group, ADD subtract they are in same group. Group 1 and subcategory I 0 is different 0 and 1, for JZ and JC same group, group 2 and their subcategory 1 is 0, 1 is 1. So, in this way we can, we have defined this thing.

(Refer Slide Time: 36:25)

Indian Institute of Technology, Kharagpur

Instruction Execution

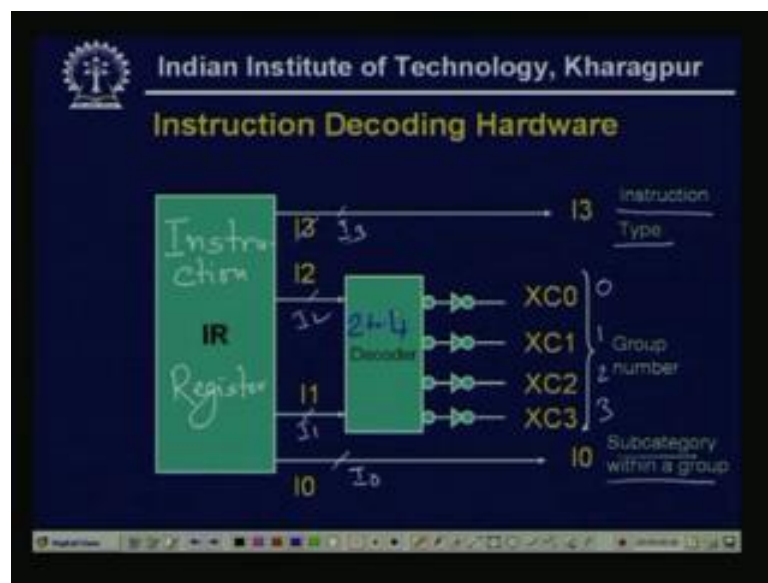
- The instruction execution involves the following steps:
 - step 1: Fetch the instruction — Fetch cycle
 - step 2: Decode the instruction to find out the required operation
 - step 3: if it is halt, go to step 6 else continue
 - Step 4: retrieve operands & perform operation
 - Step 5: go to step 1
 - Step 6: execute an infinite loop

EXECUTION

Now, how they can be these instructions can be executed, so the instruction execution involves some steps, we can define the, what is the algorithm of this instruction or what are the, what are the steps of instruction executions. So, step one is to fetch the instructions, it will read the instruction is read, step two after reading it, it should be decoded what instructions it is.

So, decode the instruction, to find out the required operation what we have to do or what this instruction supposed to do. Step three, if it is halt go to step six, else continue halt means you go to step six completed, otherwise continue means step four, step four retrieve operands and perform operations. Now, we know the operation, so we need some operands, we need some operands, so retrieve this, those operands and perform operation. Step five, go to step one, again the next instruction means, instruction should be fetched next instruction should be read. In this way we can execute the instructions.

(Refer Slide Time: 38:16)



Normally, the first step is only called the fetch cycle, the fetch instruction is normally called the fetch cycle and remaining instructions, this is called the fetch cycle and remaining from step 2 to step 6, these are called execution cycle, is a execution cycle. Now, how to decode the instruction, so we see the hardware for instruction decoding, see this is the Instruction Register IR, this is my Instruction Register.

So, I₀ I₁, this is my I₀ I₁ I₂ and I₃, now I₃ gives a instruction type, so this is my instruction type, I₀ is the subcategory within a group, I₂ I₁ is the group. So, I₂ I₁ is

the decoder input of a decoder and as there are 2, 2 bit input already we have seen, there are four groups and these groups are 0 1 2 3, these are the four groups and these are the define by this four output of the 2 to 4 decoder.

So, these are my 2 to 4, 2 to 4 decoder, so in this way this is a simply hardware for instruction decoding. Now, with this hardware and the status flag Z and C, we defined initially a micro program to implement the instruction set can be written. So, we see some of the sample instruction sets, that can be implemented, how the eleven instructions can be implemented, we can see some of them.

(Refer Slide Time: 42:08)

Instruction	Instruction Length in bytes	I-Type	Operation
1. LDA Load register A direct	2	MRI	$A \leftarrow M(addr)$
2. STA Store register A direct	2	MRI	$M(addr) \leftarrow A$
3. ADD Add register A direct	2	MRI	$A \leftarrow A + M(addr)$

So, first we see the instruction load, these are my, these are my instructions or the mnemonic instructions are or we can tell mnemonic. Instruction length, so this is instruction length in bytes, instruction type then, what is the operation. So, first we see the, the LDA the load instructions, so first what do we mean by this LDA load, this is Load Register A direct this is, load register A direct.

This is the meaning of this instruction, the instruction type is we know, MRI memory reference, so instruction length in byte is 2, one byte for op code, one byte for the address and the operation will be that LDA. So, load register from memory, so M address, so from memory the data is loaded to register A. STA is actually the reverse one, so this is store, store register A direct, the meaning is store register A direct, again

that is a type MRI and length is 2, this is totally reverse, as it is a store, so memory is stored from register A.

Now, we see ADD, this is an addition operation. So, again this is a MRI, so ADD register A, this is meaning, is meaning is ADD register A direct. So, it will be 2 with which it will be added, it is the, it will be added with the memory; that means, A is replaced by A plus M 1 memory address register. And this is subtract is totally similar, subtract is also similar only this will be replaced by A is A minus N, A is a minus N that is my, subtract again this is a MRI.

(Refer Slide Time: 47:14)

Instruction	Length in bytes	Inst-Type	Operation
5 JZ Jump to zero flag	2	MRI	If Z = 1 then $PC \leftarrow (addr)$ else $PC \leftarrow PC + 1$
6 JC Jump on Carry Flag	2	MRI	If C = 1 then $PC \leftarrow (addr)$ else $PC \leftarrow PC + 1$
7 AND AND register A	2	MRI	$A \leftarrow A \wedge$

Now, in this way that we can write the other instructions, quickly we can write the other instructions, again length in bytes, instruction type then operation. So, jump on 0 and jump on carry, there are of jump type, jump on zero flag set, this is a, this is a jump on zero flag and this is a jump on carry flag set. That means, if there 1 if Z equal to 1, if C equal to 1, both are of MRI type.

The memory reference instruction, so length is 2 and the operation will be, here one condition is checked; that means, for jump on zero I have to check whether the carry flag is 1 or not. If Z equal to 1 then, the program counter is the address, else PC is PC plus 1; that means, if it is 1 then, jump to a new address and that value is given here. This is the, this is the value, so that address is loaded to program counter.

So, that program counter always contains the address of the instruction to be executed next, else; that means, if this condition is not satisfied; that means, zero flag is not set. Then, PC is PC plus 1 that means, it will be the, it is it will be continue that means, program counter will be incremented by 1. The next address similarly, that JC means Jump on Carry, so here also that if C equal to 1 then, PC is the new address where it will be jumped.

Else PC is PC plus 1, so these two are similar type of instruction only the condition checked is zero flag or carry flag, now we see the NMRI type of none memory reference instructions, another logical operation is there, which is MRI. So, if we see the AND, that is a logical operations, so this is a AND register A. And this is a MRI type length is 2, operation is A is replaced by A, and the memory, memory address AND.

(Refer Slide Time: 51:49)

Instruction	Length	I-Type	Operation
8 CMA Complement register A	1	NMRI	$\bar{A} \leftarrow \bar{A}$
9 INCA	1	NMRI	$A \leftarrow A+1$
10 DCRA	1	NMRI	$A \leftarrow A-1$
11 HLT Halt CPU	1	NMRI	Halt

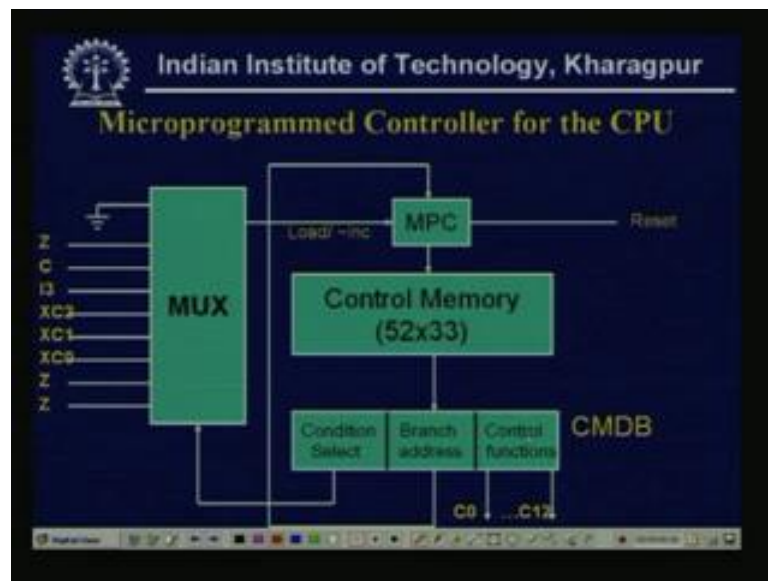
Now, we see that non-memory reference type of instructions, so the again this is a instruction or the length instruction, length in bytes then instruction type and the operation. Now, we have four instructions of type NMRI, non-memory reference instructions, one is CMA, CMA is Complement Register A, this is Complement Register A.

So, this is of length 1 byte because, NMRI again this is NMRI, operation is, here operation is A is A complement, A is replaced by A complement. Similarly, INCA the increment and decrement thereof, similar type and I can write, these are all 1 length

NMRI and this is A is replaced by this increment. So, A plus 1 A is replaced by A minus 1 and halt again this is a 1 NMRI.

And this is nothing, we will do actually this is a halt, so halt CPU, this is a halt CPU, so these are the how the instruction sets are implemented we have seen that portion. So, if we see the hardware organization of the micro programmed controlling unit.

(Refer Slide Time: 54:42)

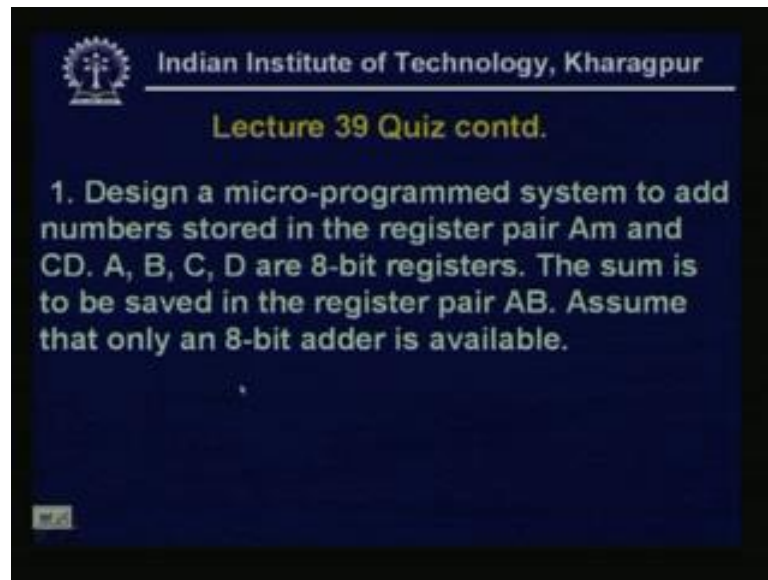


For this example CPU, will be seen that there is a multiplexer, this is a the MPC, the control memory and the CMDV that actually these are the register where three fields are they are condition select branch address and control functions that already we have seen. So, now what we have to do, that the symbolic micro program, that implements the instruction sets.

So, we have only implemented the instructions, now the, we can write the symbolic micro program for this eleven instructions. And then only we can design the, this micro program controller for the CPU. So, in this way the micro programmed CPU, we can design, so first the, we have to define the instructions. We have defined the control sets, control signals and control signals for ALU also and then, that how the hardware can be decoded or the instructions can be decoded.

That what are the, what is the hardware for that and then for all the instructions, how the instruction set can be implemented and what will be the hardware for that, that can be there. So, in this way we can design a micro program control CPU.

(Refer Slide Time: 56:33)



This is our today's quiz question that, design a micro programmed system to add numbers stored in the register pair A B and C D. This is A B, A B and C D, A B C D are 8 bit registers, the sum is to be saved in the register pair A B, assume that only an 8 adder is available. So, we have to write or we have to design a micro programmed system for this problem, this is the today's quiz question.

Thank you.

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

Lecture - 40
Digital System Design Current State of the Art

So, in the last class we have read, how one micro programmed CPU can be designed, already we have read that, how the combinational modules the sequential modules they can be designed. And the one CPU and the part of the CPU that arithmetic, logical unit, the control unit all these things can be designed. Now, with the advancement of technology, the digital systems has it is growing enormously and the complexity of the system is also large.

So, now all the design procedure is automated. So, today we will discuss that what so far we have read, how this can be automated using the CAD tools and what is the current state of the art of designing the digital systems.

(Refer Slide Time: 58:39)



Now, design complexity is increasing rapidly with the increased size and complexity. Now, so far in this class what we discussed the different design procedure, when the system is.