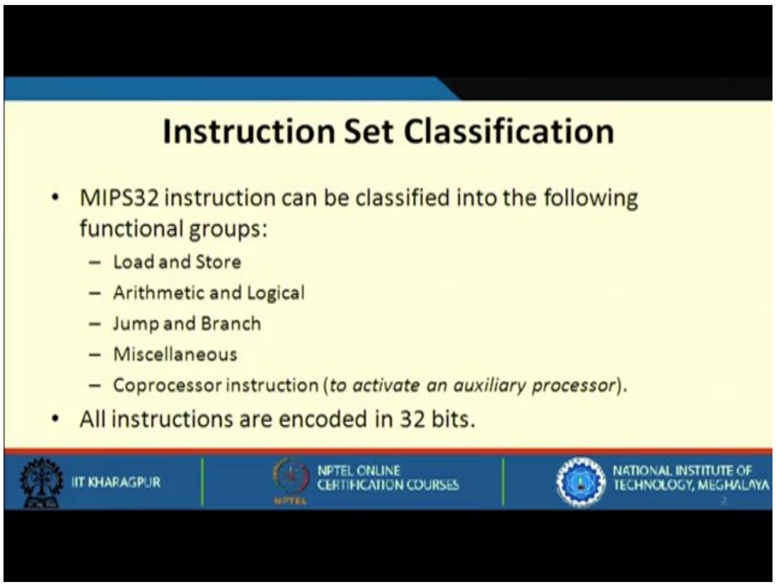
Computer Architecture and Organization

Prof. Kamalika Datta
Department of Computer Science and Engineering
National Institute of Technology, Meghalaya

Lecture - 09
MIPS32 Instruction Set

Welcome to the 9th lecture on MIPS32 instruction set. In this lecture, we will be seeing the instruction set of MIPS32 --- which means the various kinds of instructions that are possible in MIPS32.

(Refer Slide Time: 00:39)



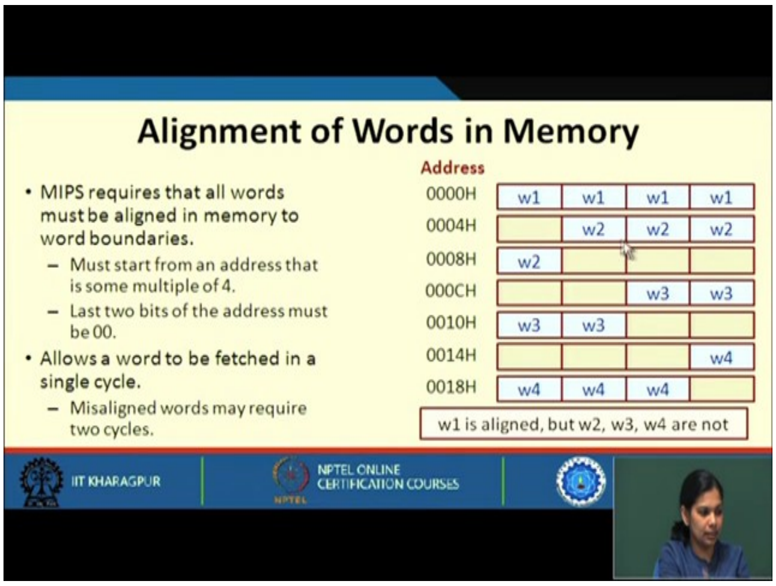
Instruction Set Classification

- MIPS32 instruction can be classified into the following functional groups:
  - Load and Store
  - Arithmetic and Logical
  - Jump and Branch
  - Miscellaneous
  - Coprocessor instruction (*to activate an auxiliary processor*).
- All instructions are encoded in 32 bits.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

Broadly instruction in MIPS can be classified into load store instructions, arithmetic and logic instructions, jump and branch, we have some miscellaneous instructions, and some coprocessor instructions. And all instructions can be encoded in 32 bits.

(Refer Slide Time: 01:11)



Alignment of Words in Memory

- MIPS requires that all words must be aligned in memory to word boundaries.
  - Must start from an address that is some multiple of 4.
  - Last two bits of the address must be 00.
- Allows a word to be fetched in a single cycle.
  - Misaligned words may require two cycles.

Address				
0000H	w1	w1	w1	w1
0004H		w2	w2	w2
0008H	w2			
000CH			w3	w3
0010H	w3	w3		
0014H				w4
0018H	w4	w4	w4	

w1 is aligned, but w2, w3, w4 are not

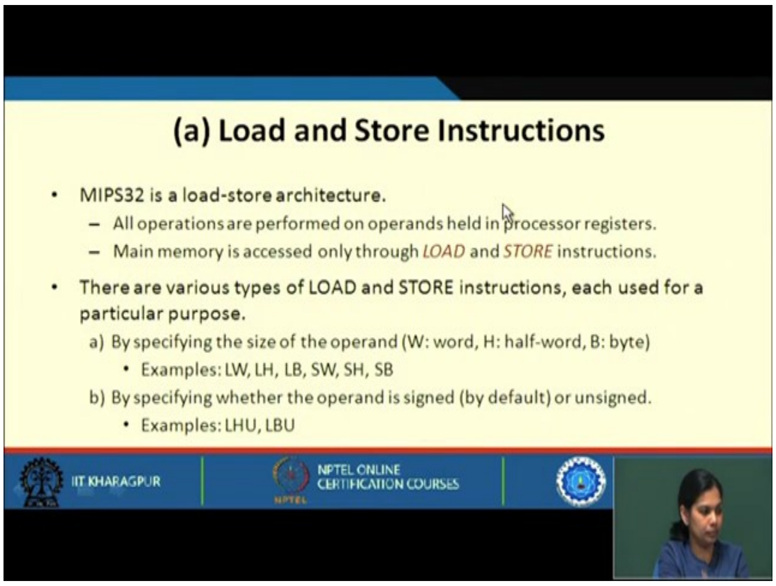
IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

Now, let us see what is this word alignment. It is alignment of the words in memory. MIPS always requires that all the words that we store in memory must be aligned to word boundaries; that means, must start from an address that is some multiple of four. So, you see the first address which is 0000, it starts from here then 0004 then 0008, and so on. So, must start from an address that is some multiple of 4.

So, the last two bits of the address must be 00, 0 – 0000, 4 – 0100, 8 – 1000, as well as all others. This allows a word to be fetched in a single cycle. So, in a single cycle, we access this word and we get this entire word in one cycle. This is why we say that MIPS requires that the words to be aligned in the memory.

Now, here the first word is aligned, but see the next word it is not aligned because it is not starting with this address it is starting with the fifth one. Similar way this word - word three and word four both are not aligned only first word w1 is aligned.

(Refer Slide Time: 03:24)

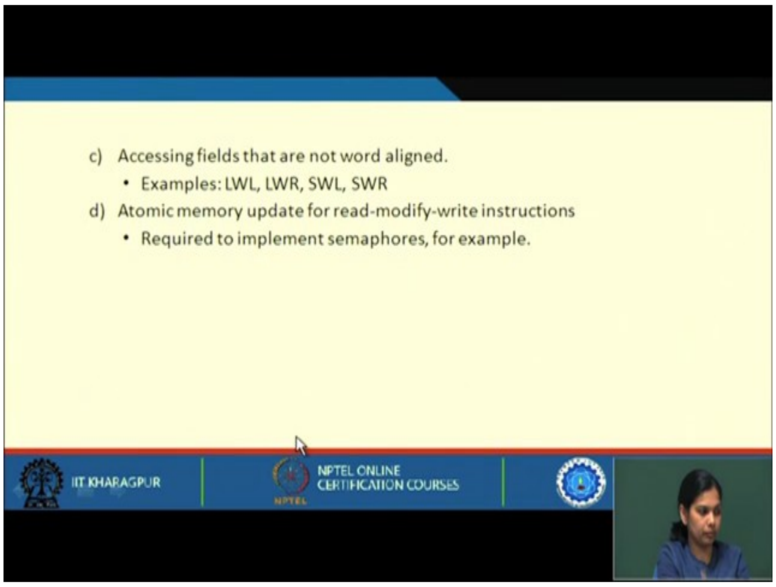


(a) Load and Store Instructions

- MIPS32 is a load-store architecture.
  - All operations are performed on operands held in processor registers.
  - Main memory is accessed only through *LOAD* and *STORE* instructions.
- There are various types of LOAD and STORE instructions, each used for a particular purpose.
  a) By specifying the size of the operand (W: word, H: half-word, B: byte)
    - Examples: LW, LH, LB, SW, SH, SB
  b) By specifying whether the operand is signed (by default) or unsigned.
    - Examples: LHU, LBU

We have been discussing about load-store architecture; what it means is that all operations are performed on operands held in the processor registers. Only two instructions can load the data from memory or it can store the data into the memory; no other instruction can use a memory location. There are various types of load-store instructions that can be used for a particular purpose; like we can load a word, we can load a byte, or we can load a half word. In the same way, we can also store a word, we can store a half word, or we can store a byte. By specifying whether the operand is signed or unsigned, we can also load a half word unsigned, load a byte unsigned.

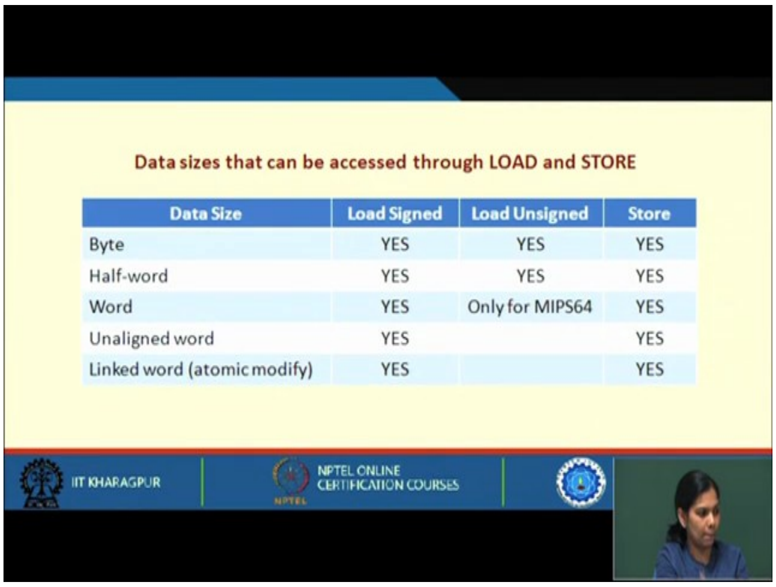
(Refer Slide Time: 04:34)



c) Accessing fields that are not word aligned.
  - Examples: LWL, LWR, SWL, SWR
d) Atomic memory update for read-modify-write instructions
  - Required to implement semaphores, for example.

Here we have another set of instructions, which are used for accessing fields that are not word aligned. The instruction that are used are load word left, and load word right, store word left, and store word right. And there are some other instructions as well like atomic memory updates for read-modify-write instruction. So, just think of an instruction that requires to be completed fully like when we read it, we modify it and we write back at the same go. So, we cannot have it that we read it, we update it, we do not update it. If we read it, we have to update it and then we can store that, so those are atomic operations.

(Refer Slide Time: 05:32)



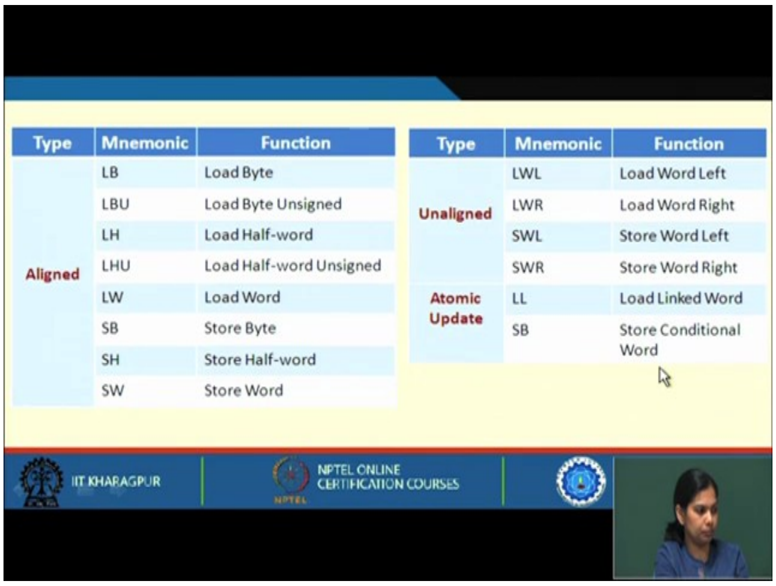
Data sizes that can be accessed through LOAD and STORE

Data Size	Load Signed	Load Unsigned	Store
Byte	YES	YES	YES
Half-word	YES	YES	YES
Word	YES	Only for MIPS64	YES
Unaligned word	YES		YES
Linked word (atomic modify)	YES		YES

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

These are the data sizes that can be accessed through load and store, but in load unsigned we can only use byte and half word. And for word this can be done only for MIPS64, and for store it can be done for all.

(Refer Slide Time: 05:54)



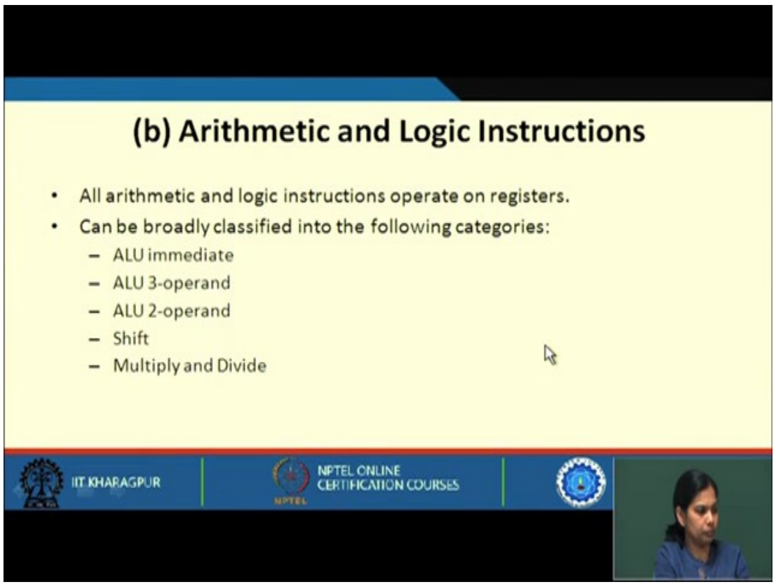
Type	Mnemonic	Function
Aligned	LB	Load Byte
	LBU	Load Byte Unsigned
	LH	Load Half-word
	LHU	Load Half-word Unsigned
	LW	Load Word
	SB	Store Byte
	SH	Store Half-word
	SW	Store Word

Type	Mnemonic	Function
Unaligned	LWL	Load Word Left
	LWR	Load Word Right
	SWL	Store Word Left
	SWR	Store Word Right
Atomic Update	LL	Load Linked Word
	SB	Store Conditional Word

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

Now, we can see some more instructions; LB is load byte, this is load byte unsigned load half word and so on. And here for an unaligned one, we have load word left, load word right and similar way store word left and store word like right. And also for atomic update, we have load linked word and store conditional word.

(Refer Slide Time: 06:28)



(b) Arithmetic and Logic Instructions

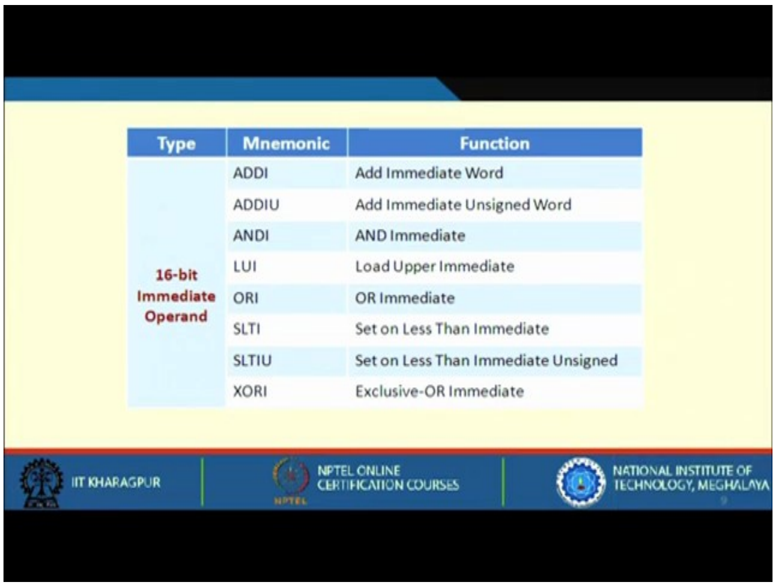
- All arithmetic and logic instructions operate on registers.
- Can be broadly classified into the following categories:
  - ALU immediate
  - ALU 3-operand
  - ALU 2-operand
  - Shift
  - Multiply and Divide

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

Let us move on with arithmetic and logic instructions. MIPS32 has a wide variety of arithmetic and logic instructions that can broadly classified into the following categories.

The categories include ALU immediate, ALU with 3-operand is possible, ALU with 2-operand, shift, multiply and divide.

(Refer Slide Time: 06:59)

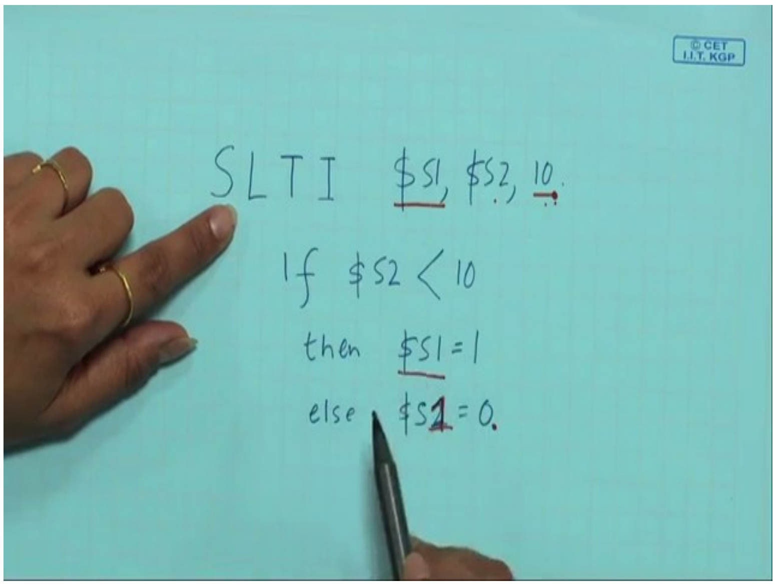


Type	Mnemonic	Function
16-bit Immediate Operand	ADDI	Add Immediate Word
	ADDIU	Add Immediate Unsigned Word
	ANDI	AND Immediate
	LUI	Load Upper Immediate
	ORI	OR Immediate
	SLTI	Set on Less Than Immediate
	SLTIU	Set on Less Than Immediate Unsigned
	XORI	Exclusive-OR Immediate

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

So, let us see the this set of arithmetic operation this is ADDI - add immediate word. This is ADDIU - add immediate unsigned word. This is LUI - load upper immediate. This is ORI - or immediate. This is SLTI - set on less than immediate.

(Refer Slide Time: 07:33)



SLTI $s1, $s2, 10.

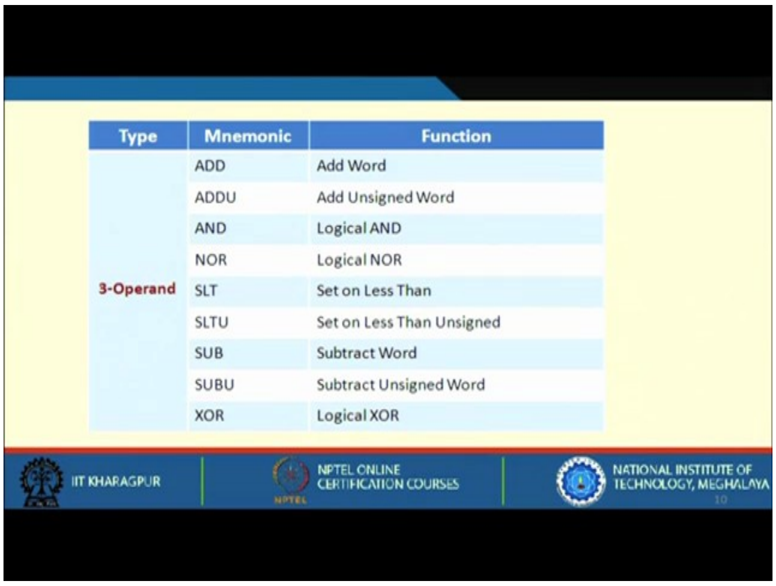
If $s2 < 10

then $s1 = 1

else $s1 = 0.

Let us take an example of SLTI - set on less than immediate, the meaning of which is if $s2 is less than this immediate value 10, then you set $s1 to 1, else you set $s1 to 0.

(Refer Slide Time: 08:15)

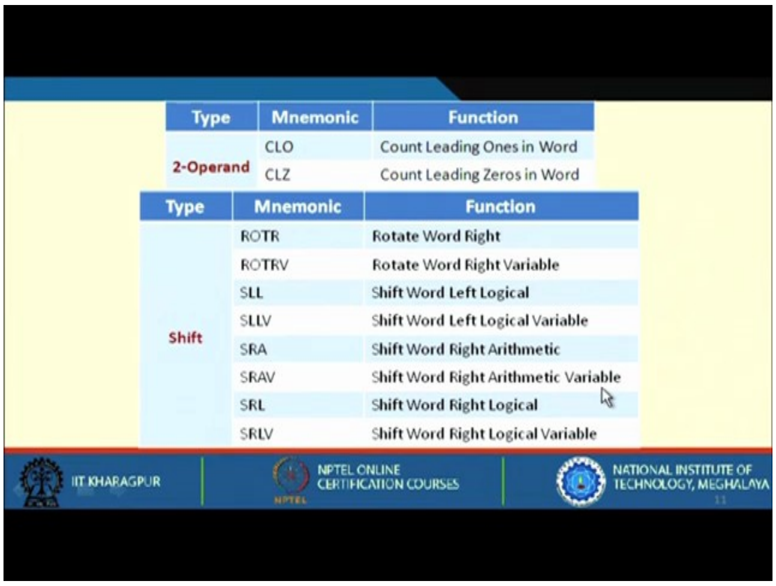


Type	Mnemonic	Function
3-Operand	ADD	Add Word
	ADDU	Add Unsigned Word
	AND	Logical AND
	NOR	Logical NOR
	SLT	Set on Less Than
	SLTU	Set on Less Than Unsigned
	SUB	Subtract Word
	SUBU	Subtract Unsigned Word
	XOR	Logical XOR

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

We have three operand instructions, where these are add, add unsigned, and, nor, set less than, set less than unsigned, sub, sub unsigned, xor.

(Refer Slide Time: 08:34)



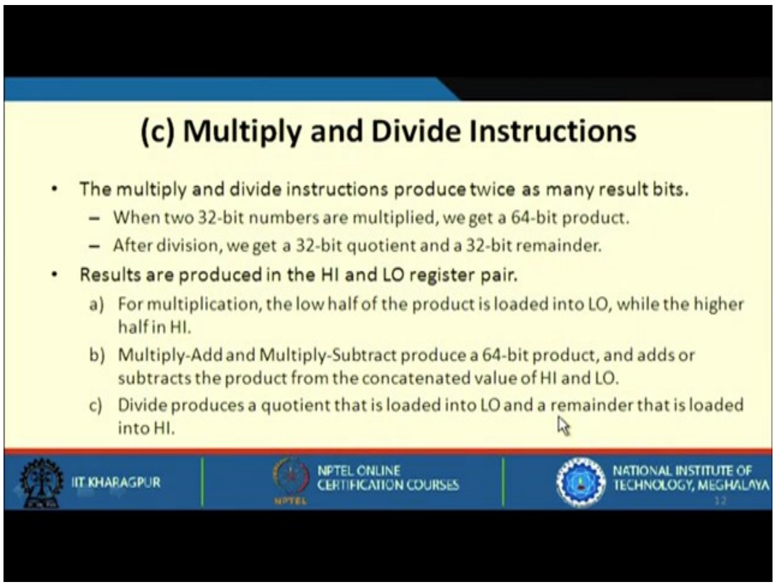
Type	Mnemonic	Function
2-Operand	CLO	Count Leading Ones in Word
	CLZ	Count Leading Zeros in Word

Type	Mnemonic	Function
Shift	ROTR	Rotate Word Right
	ROTRV	Rotate Word Right Variable
	SLL	Shift Word Left Logical
	SLLV	Shift Word Left Logical Variable
	SRA	Shift Word Right Arithmetic
	SRAV	Shift Word Right Arithmetic Variable
	SRL	Shift Word Right Logical
	SRLV	Shift Word Right Logical Variable

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

For two-operand, we have these instructions CLO that is count leading ones in a word, or CLZ that is count leading zeros in a word. So, these instructions are also sometimes required for various programming. We have another set of instructions for rotating a word. So, we can rotate a word right, we can rotate a word with a variable, and we can specify that how many bits we need to rotate. We can shift a word left logical, that is, we can do a logical shift; we can shift a word left logical, and we can specify the variable. And similarly we can do it for shifting a word right arithmetic - shifting a word right with arithmetic right shift; and with arithmetic right shift with some variable, and we can also do logical right shift.

(Refer Slide Time: 09:50)



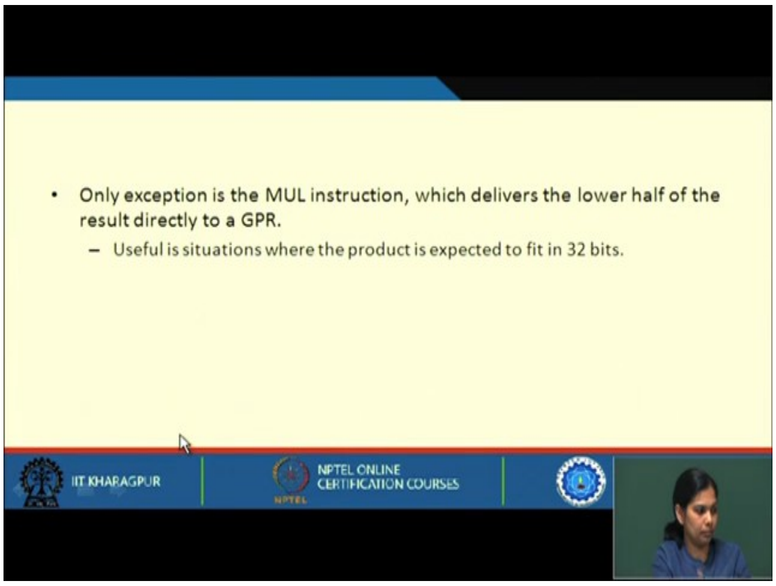
(c) Multiply and Divide Instructions

- The multiply and divide instructions produce twice as many result bits.
  - When two 32-bit numbers are multiplied, we get a 64-bit product.
  - After division, we get a 32-bit quotient and a 32-bit remainder.
- Results are produced in the HI and LO register pair.
  a) For multiplication, the low half of the product is loaded into LO, while the higher half in HI.
  b) Multiply-Add and Multiply-Subtract produce a 64-bit product, and adds or subtracts the product from the concatenated value of HI and LO.
  c) Divide produces a quotient that is loaded into LO and a remainder that is loaded into HI.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

Let us move on and see multiply and divide instructions. So, the next set of instructions that MIPS32 instruction set is having is multiply and divide. So, when two 32-bit numbers are multiplied we can get a 64-bit product; and after division also you may have to store a 32-bit quotient and a 32-bit remainder. So, where we will store this 64-bit result? We have a register called HI-LO; this is a register pair. So, for multiplication the low half of the product is loaded in LO while high half is loaded in HI. Multiply and multiply subtract produces a 64-bit product and adds or subtract the product from the concatenated value of HI and LO. And also divide produces a quotient that is loaded into LO and a remainder that is loaded into HI.

(Refer Slide Time: 11:03)

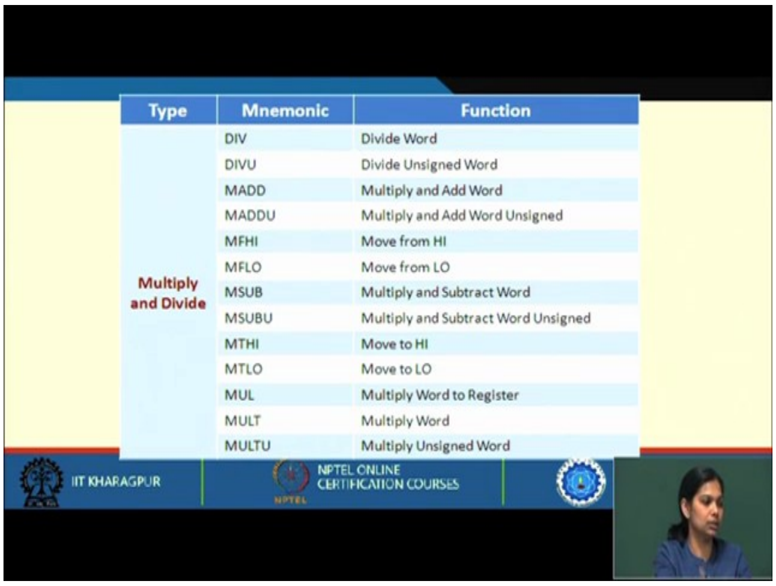


- Only exception is the MUL instruction, which delivers the lower half of the result directly to a GPR.
  - Useful is situations where the product is expected to fit in 32 bits.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

There is only one exception that for the multiplication instruction which delivers the lower half of the result directly to GPR, because it is useful for the situation when the product is expected because when we multiply two numbers always the result may not be 64-bit, the result can fit into 32-bit as well. So, in that case is an exception with MUL instruction.

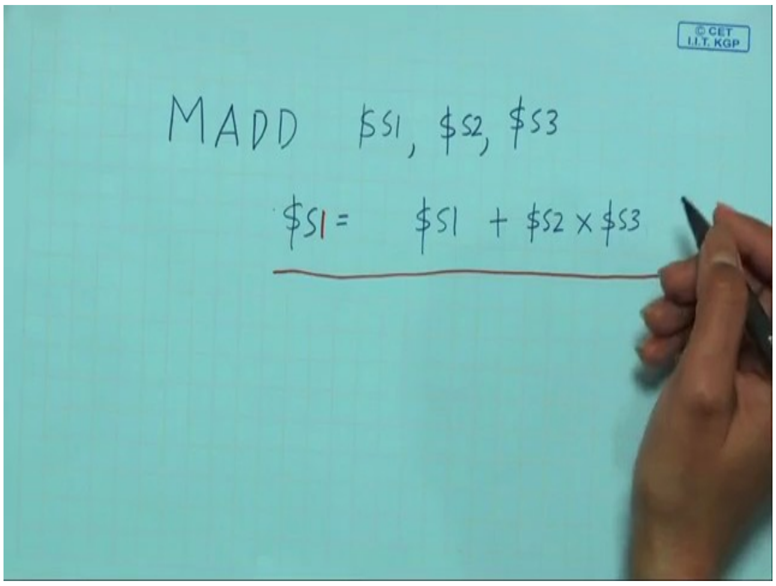
(Refer Slide Time: 11:34)



Type	Mnemonic	Function
Multiply and Divide	DIV	Divide Word
	DIVU	Divide Unsigned Word
	MADD	Multiply and Add Word
	MADDU	Multiply and Add Word Unsigned
	MFHI	Move from HI
	MFLO	Move from LO
	MSUB	Multiply and Subtract Word
	MSUBU	Multiply and Subtract Word Unsigned
	MTHI	Move to HI
	MTLO	Move to LO
	MUL	Multiply Word to Register
	MULT	Multiply Word
	MULTU	Multiply Unsigned Word

So, these are the various instructions that I have talking about: div, divide with unsigned word multiply and add word. So, what is this multiply and add word.

(Refer Slide Time: 11:49)

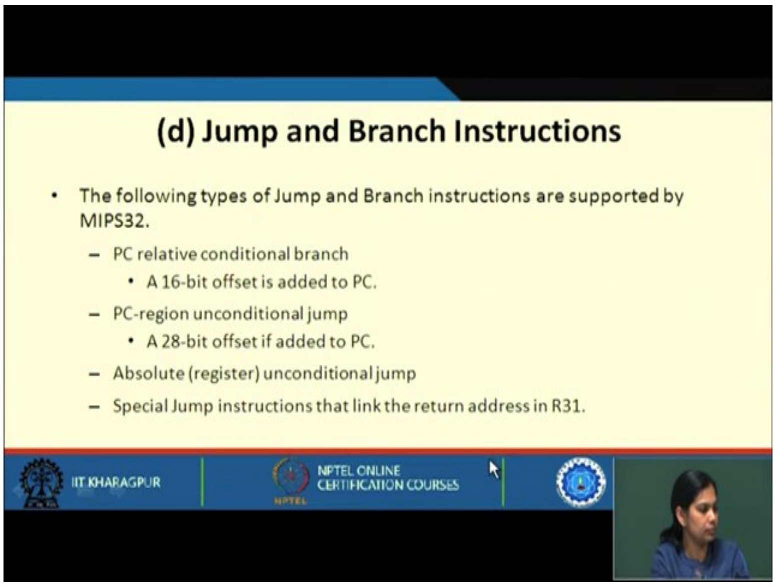


MADD \$S1, \$S2, \$S3

$\$S1 = \$S1 + \$S2 \times \$S3$

Let us see this instruction. MADD means we are multiplying with three operand here. So, we are multiplying \$s2 and \$s3, and we are adding with \$s1, and also we are storing the result in \$s1. Such kind of instructions are required in digital signal processing, and is supported in MIPS32 architecture. So, we have various move from high, move from low, you have various multiply word to a register, this is generally used multiply a word, multiply unsigned word.

(Refer Slide Time: 12:43)

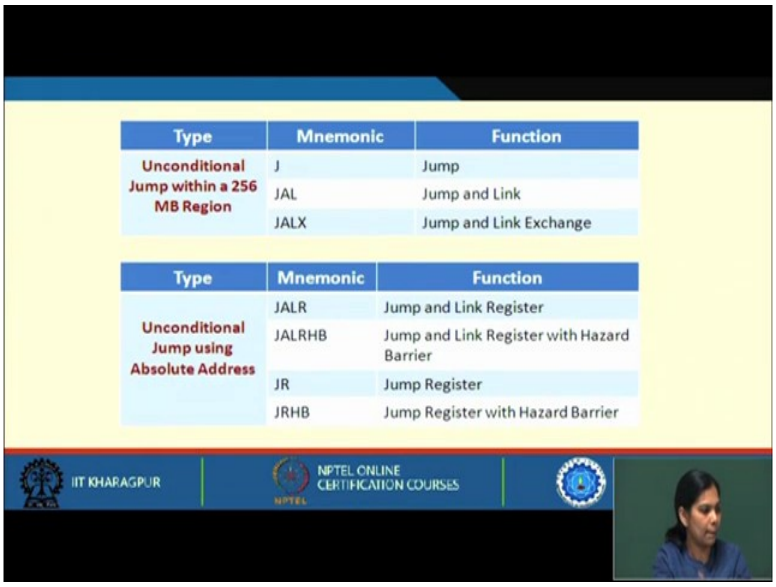

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

(d) Jump and Branch Instructions

- The following types of Jump and Branch instructions are supported by MIPS32.
  - PC relative conditional branch
    - A 16-bit offset is added to PC.
  - PC-region unconditional jump
    - A 28-bit offset if added to PC.
  - Absolute (register) unconditional jump
  - Special Jump instructions that link the return address in R31.

Next set of instructions is jump and branch. The following types of jump and branch instructions are supported in MIPS32. We know that whenever we wanted to perform some kind of branching like in a program with loops, for those we require jump and branch instructions. We have PC relative conditional branch where a 16-bit offset is added to PC, or in a conditional unconditional branch a 28-bit offset is added to PC. There can be absolute unconditional branch whether absolute address can be provided in the register, and special jump functions that link the return address in R31. So, we are jumping from one location to another and after executing that particular location we have to come back to the previous one. So, the value of the PC must be loaded with that return address value.

(Refer Slide Time: 13:53)

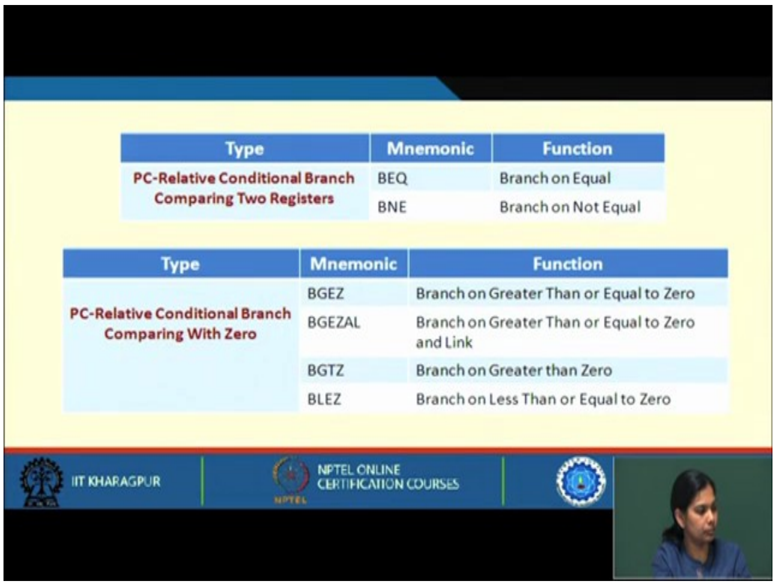


Type	Mnemonic	Function
Unconditional Jump within a 256 MB Region	J	Jump
	JAL	Jump and Link
	JALX	Jump and Link Exchange

Type	Mnemonic	Function
Unconditional Jump using Absolute Address	JALR	Jump and Link Register
	JALRHB	Jump and Link Register with Hazard Barrier
	JR	Jump Register
	JRHB	Jump Register with Hazard Barrier

In this context we have some instruction, this is unconditional branch jump and link. Jump and link exchange, and these are some unconditional jump using absolute address. So, jump and link register, jump and link register with hazard barrier. So, you will be seeing some of these instructions when we will be studying pipelining in course of time.

(Refer Slide Time: 14:24)

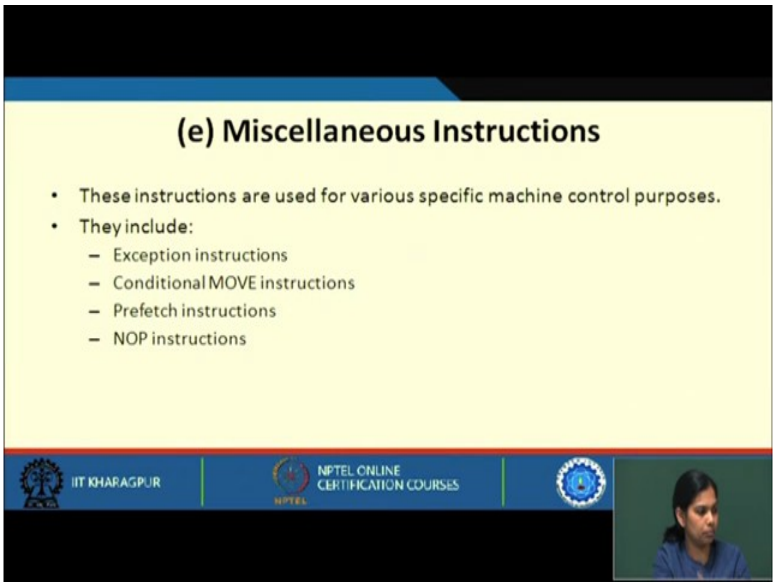


Type	Mnemonic	Function
PC-Relative Conditional Branch Comparing Two Registers	BEQ	Branch on Equal
	BNE	Branch on Not Equal

Type	Mnemonic	Function
PC-Relative Conditional Branch Comparing With Zero	BGEZ	Branch on Greater Than or Equal to Zero
	BGEZAL	Branch on Greater Than or Equal to Zero and Link
	BGTZ	Branch on Greater than Zero
	BLEZ	Branch on Less Than or Equal to Zero

So, related to PC there are some instructions: BEQ and BNE. We shall come across these kind of branch instruction very frequently when we do some programming. And there are some PC relative conditional branch comparing with zero. So, branch on greater than or equal to zero, or branch on greater than or equal to zero and link. We have few branch on greater than zero, branch on less than equal to zero. So, we have various instructions that we can we may use for our programming constructs.

(Refer Slide Time: 15:11)



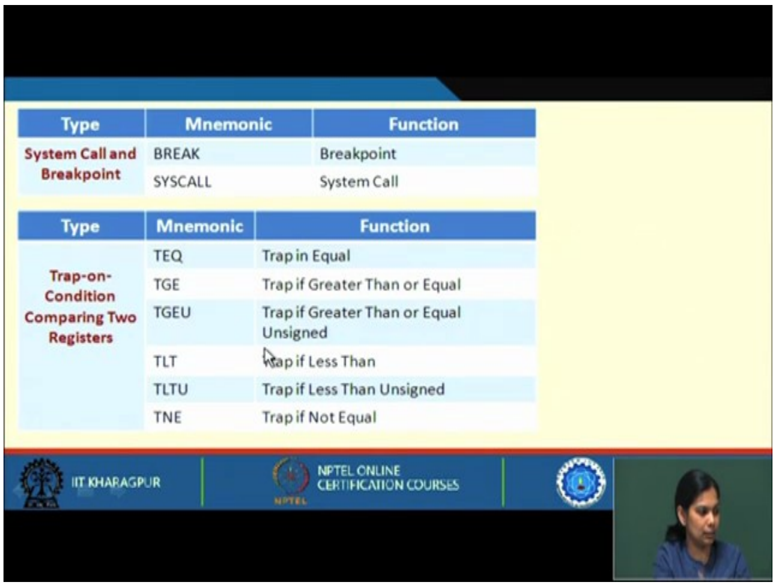
(e) Miscellaneous Instructions

- These instructions are used for various specific machine control purposes.
- They include:
  - Exception instructions
  - Conditional MOVE instructions
  - Prefetch instructions
  - NOP instructions

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

We also have some miscellaneous instructions used for various specific machine control purposes, and they include some exceptional instructions, conditional move instruction, some prefetch instructions are also there, no operation instructions. So, we will again see that some of these instructions like NOP may be used in pipeline for some purposes.

(Refer Slide Time: 15:40)

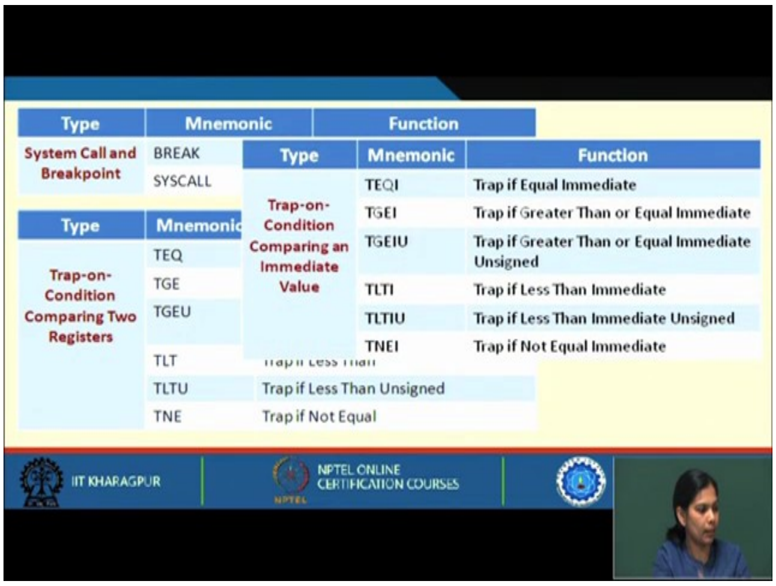


Type	Mnemonic	Function
System Call and Breakpoint	BREAK	Breakpoint
	SYSCALL	System Call

Type	Mnemonic	Function
Trap-on-Condition Comparing Two Registers	TEQ	Trap in Equal
	TGE	Trap if Greater Than or Equal
	TGEU	Trap if Greater Than or Equal Unsigned
	TLT	Trap if Less Than
	TLTU	Trap if Less Than Unsigned
	TNE	Trap if Not Equal

Next is system call. SYSCALL is something which we will be using often in programming using QtSPIM where we use system call which is syscall. And there can be some instructions like trap; trap if equal, trap if greater than or equal, trap if greater than or equal unsigned, and so on. So, these are some OS related instructions that are used if you want to request OS for some requirement, and then the OS takes care of it.

(Refer Slide Time: 16:39)



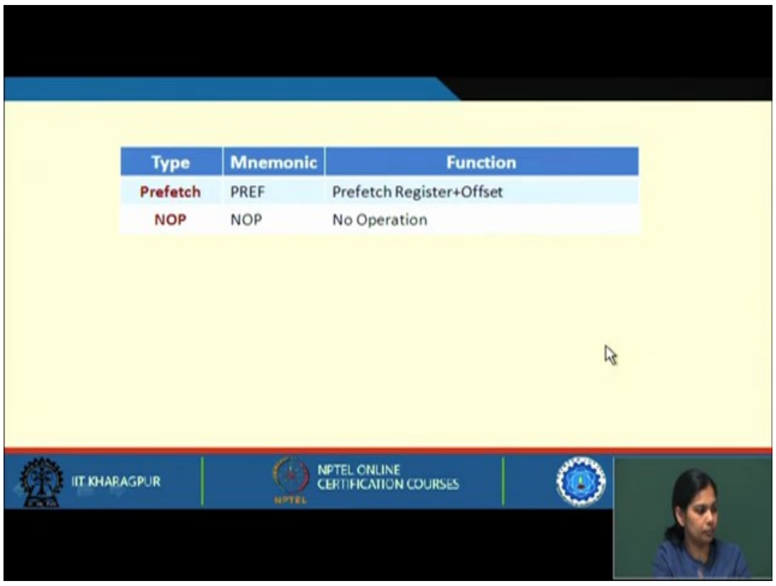
Type	Mnemonic	Function
System Call and Breakpoint	BREAK	
	SYSCALL	

Type	Mnemonic	
Trap-on-Condition Comparing Two Registers	TEQ	
	TGE	
	TGEU	
	TLT	Trap if Less Than
	TLTU	Trap if Less Than Unsigned
	TNE	Trap if Not Equal

Type	Mnemonic	Function
Trap-on-Condition Comparing an Immediate Value	TEQI	Trap if Equal Immediate
	TGEI	Trap if Greater Than or Equal Immediate
	TGEIU	Trap if Greater Than or Equal Immediate Unsigned
	TLTI	Trap if Less Than Immediate
	TLTIU	Trap if Less Than Immediate Unsigned
	TNEI	Trap if Not Equal Immediate

So, these are also various kind of trap-on-condition comparing with an immediate value; these are some of those instructions.

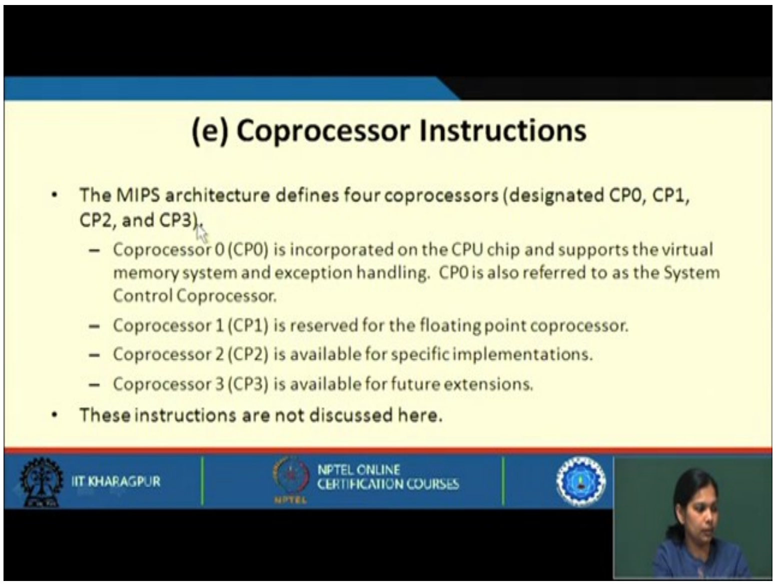
(Refer Slide Time: 16:51)



Type	Mnemonic	Function
Prefetch	PREF	Prefetch Register+Offset
NOP	NOP	No Operation

And this is prefetch and no operation.

(Refer Slide Time: 16:57)



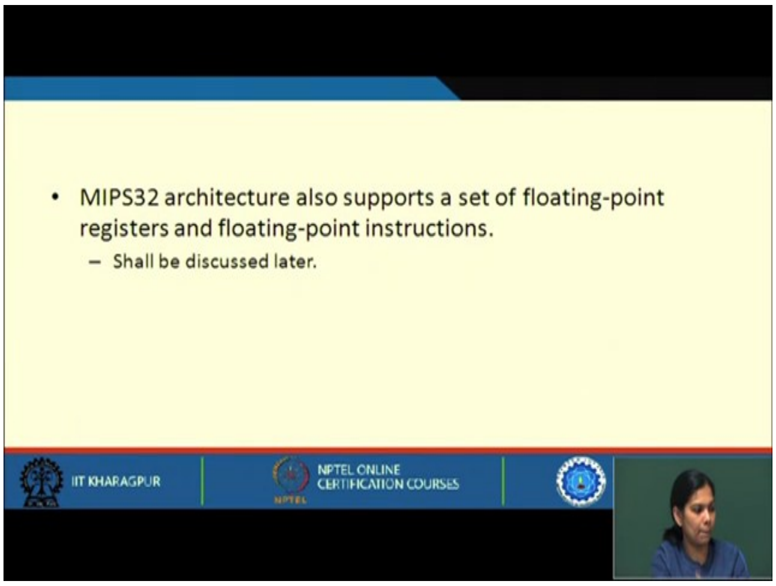
(e) Coprocessor Instructions

- The MIPS architecture defines four coprocessors (designated CP0, CP1, CP2, and CP3).
  - Coprocessor 0 (CP0) is incorporated on the CPU chip and supports the virtual memory system and exception handling. CP0 is also referred to as the System Control Coprocessor.
  - Coprocessor 1 (CP1) is reserved for the floating point coprocessor.
  - Coprocessor 2 (CP2) is available for specific implementations.
  - Coprocessor 3 (CP3) is available for future extensions.
- These instructions are not discussed here.

Next set of instruction is coprocessor instructions. The MIPS architecture defines four coprocessors, designated as coprocessor 0, 1, 2, and 3. CP0 is incorporated within the CPU chip and this also supports the virtual memory system and exceptional handling. CP0 is also referred to as system control coprocessor.

Now, these four coprocessors may not be used for all cases, but like CP1 is reserved for floating point coprocessor. CP2 is available for specific implementations and CP3 can be used for future extension. So, we really do not know that which kind of coprocessor we will be using in near feature. So, for that reason we need some coprocessor that can be available in future, and these instruction are not discussed here. So, we are just saying that we will have some kind of instructions like coprocessor instructions.

(Refer Slide Time: 18:15)



- MIPS32 architecture also supports a set of floating-point registers and floating-point instructions.
  - Shall be discussed later.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL

MIPS architecture also supports a set of floating point registers and floating point instructions that shall be discussed later.

So, we came through end of lecture 9, where we discussed the various kind of instructions that are there in MIPS32 architecture.

Thank you.