

---

# High Performance Computing

## Lecture 5

Matthew Jacob

Indian Institute of Science

---

# Addressing Modes: Operand in Memory

3. **Register Indirect** Addressing Mode
4. **Base-Displacement** Addressing Mode
5. **Absolute** Addressing Mode
6. **Indexed** Addressing Mode

Memory address of operand is calculated as sum of contents of 2 registers

ADD R1, R2, (R3+R4)

---

# Addressing Modes: Operand in Memory

3. **Register Indirect** Addressing Mode
  4. **Base-Displacement** Addressing Mode
  5. **Absolute** Addressing Mode
  6. **Indexed** Addressing Mode
- **Others**
    - Auto-increment/decrement (pre/post)
    - PC relative
      - The operand address is specified as a displacement from the PC value (i.e., from the address of the instruction itself)

---

# Example: A RISC Instruction Set

- We must know a specific instruction set to understand program code examples in the remainder of this course
- Next: Details of a typical RISC instruction set

---

# Example: A RISC Instruction Set (MIPS 1)

- Registers
  - 32 32b general purpose registers, R0..R31
    - R0 hardwired to value 0
    - R31 implicitly used by some instructions
    - “implicitly”: R31 will not be explicitly indicated as an operand of the instruction
    - Example: JALR R3
  - HI, LO: 2 other 32b registers
    - Used implicitly by multiply and divide instructions
    - Example: MULT R1, R2

---

# Example: A RISC Instruction Set (MIPS 1)

- Registers
  - 32 32b general purpose registers, R0..R31
- Addressing Modes
  - Immediate, Register direct (arithmetic, logical)
  - Absolute (jumps)
  - Base-displacement (loads, stores)
  - PC relative (conditional branches)

---

# MIPS I ISA: General Comments

- All instructions, registers are 32b in size
- **Load-store architecture**: the only instructions that have memory operands are loads&stores
- Terminology
  - Word: 32b
  - Halfword: 16b
  - Byte: 8b
- Displacements (for base-displacement mode) and immediate values (for immediate mode) are signed 16 bit quantities

# Data Transfer Instructions

	Mnemonics	Example	Meaning
Load	LB, LBU, LH, LHU, LW, LUI	LW R2, 4(R3)	$R2 \leftarrow \text{Mem}[R3+4]$
Store	SB, SH, SW	SB R2, -8(R4)	$\text{Mem}[R4-8] \leftarrow R2$
Move	MFHI, MFLO, MTHI, MTLO	MFHI R1	$R1 \leftarrow HI$

L: Load (data transfer from memory to a register)

S: Store (data transfer from a register to memory)

M: Move (between GPRs and HI/LO)

B: Byte (8b), H: Half (16b), W: Word (32b)

U: Unsigned; F: From; T: To, UI: Upper Immediate



# Data Transfer Instructions.

	Mnemonics	Example	Meaning
Load	<b>LB, LBU</b> , LH, LHU, LW, LUI	LW R2, 4(R3)	$R2 \leftarrow \text{Mem}[R3+4]$
Store	SB, SH, SW	SB R2, -8(R4)	$\text{Mem}[R4-8] \leftarrow R2$
Move	MFHI, MFLO, MTHI, MTLO	MFHI R1	$R1 \leftarrow HI$

What is the difference between LB (load byte) and LBU (load byte unsigned)?

---

# LB and LBU

- Both load a byte from memory into the least significant 8b of the destination register
- They differ in how they effect the rest of the destination register

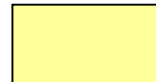
LB R1, 0(R2)



LBU R1, 0(R2)



The byte from main memory



# LB and LBU.

- Both load a byte from memory into the least significant 8b of the destination register
- They differ in how they effect the rest of the destination register

LB R1, 0(R2)

LBU R1, 0(R2)

R1 

11111...111111	1....
----------------	-------

R1 

00000...000000	
----------------	--

The byte from main memory 

--

# LB and LBU..

- Both load a byte from memory into the least significant 8b of the destination register
- They differ in how they effect the rest of the destination register

LB R1, 0(R2)

LBU R1, 0(R2)

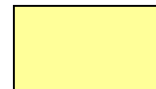
R1 00000...000000 0....

R1 00000...000000

Sign Extension

Zero Extension

The byte from main memory



# Integer Arithmetic/Logical Instructions

	Mnemonics	Example	Meaning
Add, Subtract	ADD, ADDU, ADDI, ADDIU, SUB, SUBU	ADD R1, R2, R3 ADDI R1, R2, 6	$R1 \leftarrow R2 + R3$ $R1 \leftarrow R2 + 6$
Multiply, Divide	MULT, DIV, MULTU, DIVU	MULT R1, R2	$LO \leftarrow \text{lsw}(R1 * R2)$ $HI \leftarrow \text{msw}(R1 * R2)$
Logical	AND, ANDI, OR, ORI, XOR, XORI, NOR	ORI R1, R2, 0xF	$R1 \leftarrow R2   \text{SE}(0xF)$

Shift and Comparison instructions have been left out of this table

I: Immediate

LSW: Least Significant Word

SE: Sign Extension

# Integer Arithmetic/Logical Instructions

	Mnemonics	Example	Meaning
Add, Subtract	ADD, ADDU, ADDI, ADDIU, SUB, SUBU	ADD R1, R2, R3 ADDI R1, R2, 6	$R1 \leftarrow R2 + R3$ $R1 \leftarrow R2 + 6$
Multiply, Divide	MULT, DIV, MULTU, DIVU	MULT R1, R2	$LO \leftarrow \text{lsw}(R1 * R2)$ $HI \leftarrow \text{msw}(R1 * R2)$
Logical	AND, ANDI, OR, ORI, XOR, XORI, NOR	ORI R1, R2, 0xF	$R1 \leftarrow R2   SE(0xF)$

1. How do you get a constant value into a register?
2. How does the MULT instruction work?

---

# Getting a Constant into a Register

- Example 1: We want to get the 16b value 0x01AB into register R3

```
ADDI R3, R0, 0x01AB
```

- Example 2: We want to get the 32b value 0x01ABCDEF into register R4

```
LUI R4, 0x01AB / puts 0s into LSBs
```

```
ADDI R4, R4, 0xCDEF
```

# Integer Arithmetic/Logical Instructions

	Mnemonics	Example	Meaning
Add, Subtract	ADD, ADDU, ADDI, ADDIU, SUB, SUBU	ADD R1, R2, R3 ADDI R1, R2, 6	$R1 \leftarrow R2 + R3$ $R1 \leftarrow R2 + 6$
Multiply, Divide	MULT, DIV, MULTU, DIVU	MULT R1, R2	$LO \leftarrow \text{lsw}(R1 * R2)$ $HI \leftarrow \text{msw}(R1 * R2)$
Logical	AND, ANDI, OR, ORI, XOR, XORI, NOR	ORI R1, R2, 0xF	$R1 \leftarrow R2   SE(0xF)$

1. How do you get a constant value into a register?
2. How does the MULT instruction work?