

In the name of Allah

## Overview Presentation

Zohre Soorani  
Mahdi Haghverdi  
Hussein Hussein  
Hosna Rajaei



Isfahan University

November 14, 2023

# Content

## Computer Abstraction and Technology

### Instructions: Language of the computer

#### Operations of the Computer Hardware

#### Operands of the Computer Hardware

##### Memory Operands

##### Constant or Immediate Operands

#### Representing Instructions

# Computer Abstraction and Technology

# Computer Abstraction and Technology

# Instructions: Language of the computer

# Operations of the Computer Hardware

Figure: Arithmetic Instructions in MIPS

Category	Instruction	Example	Meaning	Comments
Arithmetic	add	add \$s1,\$s2,\$s3	$\$s1 = \$s2 + \$s3$	Three register operands
	subtract	sub \$s1,\$s2,\$s3	$\$s1 = \$s2 - \$s3$	Three register operands
	add immediate	addi \$s1,\$s2,20	$\$s1 = \$s2 + 20$	Used to add constants

# Operations of the Computer Hardware (Cont'd)

load word	lw \$s1, 20(\$s2)	\$s1 = Memory[\$s2 + 20]	Word from memory to register
store word	sw \$s1, 20(\$s2)	Memory[\$s2 + 20] = \$s1	Word from register to memory
load byte	lb \$s1, 20(\$s2)	\$s1 = Memory[\$s2 + 20]	Byte from memory to register
load byte unsigned	lbu \$s1, 20(\$s2)	\$s1 = Memory[\$s2 + 20]	Byte from memory to register
store byte	sb \$s1, 20(\$s2)	Memory[\$s2 + 20] = \$s1	Byte from register to memory
load upper immed	lui \$s1, 20	$\$s1 = 20 * 2^{16}$	Loads constant in upper 16 bits

Table: Data Transfer Instructions in MIPS

# Operations of the Computer Hardware (Cont'd)

Figure: Logical Instructions in MIPS

Logical	and	and \$s1,\$s2,\$s3	$\$s1 = \$s2 \& \$s3$	Three reg. operands; bit-by-bit AND
	or	or \$s1,\$s2,\$s3	$\$s1 = \$s2 \mid \$s3$	Three reg. operands; bit-by-bit OR
	nor	nor \$s1,\$s2,\$s3	$\$s1 = \sim (\$s2 \mid \$s3)$	Three reg. operands; bit-by-bit NOR
	and immediate	andi \$s1,\$s2,20	$\$s1 = \$s2 \& 20$	Bit-by-bit AND reg with constant
	or immediate	ori \$s1,\$s2,20	$\$s1 = \$s2 \mid 20$	Bit-by-bit OR reg with constant
	shift left logical	sll \$s1,\$s2,10	$\$s1 = \$s2 \ll 10$	Shift left by constant
	shift right logical	srl \$s1,\$s2,10	$\$s1 = \$s2 \gg 10$	Shift right by constant



# Operations of the Computer Hardware (Cont'd)

Figure: Conditional Branch Instructions in MIPS

Conditional branch	branch on equal	beq \$s1,\$s2,25	if (\$s1 == \$s2) go to PC + 4 + 100	Equal test; PC-relative branch
	branch on not equal	bne \$s1,\$s2,25	if (\$s1 != \$s2) go to PC + 4 + 100	Not equal test; PC-relative
	set on less than	slt \$s1,\$s2,\$s3	if (\$s2 < \$s3) \$s1 = 1; else \$s1 = 0	Compare less than; for beq, bne
	set on less than unsigned	sltu \$s1,\$s2,\$s3	if (\$s2 < \$s3) \$s1 = 1; else \$s1 = 0	Compare less than unsigned
	set less than immediate	slti \$s1,\$s2,20	if (\$s2 < 20) \$s1 = 1; else \$s1 = 0	Compare less than constant
	set less than immediate unsigned	sltiu \$s1,\$s2,20	if (\$s2 < 20) \$s1 = 1; else \$s1 = 0	Compare less than constant unsigned

# Operations of the Computer Hardware (Cont'd)

Figure: Unconditional Jump Instructions in MIPS

Unconditional jump	jump	j 2500	go to 10000	Jump to target address
	jump register	jr \$ra	go to \$ra	For switch, procedure return
	jump and link	j al 2500	\$ra = PC + 4; go to 10000	For procedure call

## Example - Compiling a Complex C Assignment into MIPS

A somewhat complex statement contains the five variables **f**, **g**, **h**, **i**, and **j**:

**f** = (**g** + **h**) - (**i** + **j**);

What might a C compiler produce?

# Answer

- add t0, g, h # temporary variable t0 contains  $g + h$
- add t1, i, j # temporary variable t1 contains  $i + j$
- sub f, t0, t1 # f gets  $t0 - t1$ , which is  $(g + h) - (i + j)$

## Example - Compiling a C Assignment Using Registers

It is the compiler's job to associate program variables with registers.

Take, for instance, the assignment statement from our earlier example:

```
f = (g + h) - (i + j);
```

The variables `f`, `g`, `h`, `i`, and `j` are assigned to the registers `$s0`, `$s1`, `$s2`, `$s3`, and `$s4`, respectively.

What is the compiled MIPS code?

# Answer

- add \$t0, \$s1, \$s2 # register \$t0 contains  $g + h$
- add \$t1, \$s3, \$s4 # register \$t1 contains  $i + j$
- sub \$s0, \$t0, \$t1 # f gets  $t0 - t1$ , which is  $(g + h) - (i + j)$

## Example - Compiling Using Load and Store

Assume variable `h` is associated with register `$s2` and the base address of the array `A` is in `$s3`.

What is the MIPS assembly code for the C assignment statement below?

```
A[12] = h + A[8];
```

# Answer

- `lw $t0, 32($s3)` # Temporary reg \$t0 gets A[8]
- `add $t0, $s2, $t0` # Temporary reg \$t0 gets h + A[8]
- `sw $t0, 48($s3)` # Stores h + A[8] back into A[12]



## Constant or Immediate Operands

- `addi $s3, $s3, 4`    `# $s3 = $s3 + 4`

# Instructions Big Picture

## MIPS machine language

Name	Format	Example						Comments
add	R	0	18	19	17	0	32	add \$s1,\$s2,\$s3
sub	R	0	18	19	17	0	34	sub \$s1,\$s2,\$s3
addi	I	8	18	17	100			addi \$s1,\$s2,100
lw	I	35	18	17	100			lw \$s1,100(\$s2)
sw	I	43	18	17	100			sw \$s1,100(\$s2)
Field size		6 bits	5 bits	5 bits	5 bits	5 bits	6 bits	All MIPS instructions are 32 bits long
R-format	R	op	rs	rt	rd	shamt	funct	Arithmetic instruction format
I-format	I	op	rs	rt	address			Data transfer format