Computer Architecture

Introduction to Instructions
Lecture 4

Dr. Anirban Sengupta, CSE, Prof
Indian Institute of Technology Indore
Web: http://www.anirban-sengupta.com/

Categories of Instructions

- **Arithmetic type**
  - ❖ Integer
  - ❖ Floating Point
- **Memory access instruction type**
  - ❖ Load & Store
- **Control flow type**
  - ❖ Jump
  - ❖ Conditional Branch
  - ❖ Call & Return

Ref. Computer Architecture: A Quantitative Approach 5th Edition, The Morgan Kaufmann Series, by John L. Hennessy , David A. Patterson

Registers in MIPS

Actual register numbers used in instruction formats

Register names

Name	Register Number	Usage
$zero	0	the constant 0
$v0 - $v1	2-3	returned values
$a0 - $a3	4-7	arguments
$t0 - $t7	8-15	temporaries
$s0 - $s7	16-23	saved values
$t8 - $t9	24-25	temporaries
$gp	28	global pointer
$sp	29	stack pointer
$fp	30	frame pointer
$ra	31	return address

Not frequently used

Ref. Computer Architecture: A Quantitative Approach 5th Edition, The Morgan Kaufmann Series, by John L. Hennessy , David A. Patterson

Arithmetic operations

- Most instructions have 3 operands
- Operand order is fixed (destination, source 1, source 2)

Example:

High level code: `Z = Y + X`

MIPS code: `add $s0, $s1, $s2`

($s0, $s1 and $s2 are associated with variables by compiler)

Arithmetic operations

High level code:
```
X = P + Q + R;
Z = Y - X;
```

MIPS code:
```
add $t0, $s1, $s2
add $s0, $t0, $s3
sub $s4, $s5, $s0
```

- Operands must be registers, MIPS provides only 32 registers

Ref. Computer Architecture: A Quantitative Approach 5th Edition, The Morgan Kaufmann Series, by John L. Hennessy , David A. Patterson

Instructions

MIPS instructions divided into 5 classes:

- Arithmetic/logical/shift/comparison
- Control instructions (branch and jump)
- Load/store
- Other (exception, register movement to/from GP registers, etc.)

Three instruction encoding formats:

- R-type (6-bit opcode, 5-bit rs, 5-bit rt, 5-bit rd, 5-bit shamt, 6-bit function code)
- I-type (6-bit opcode, 5-bit rs, 5-bit rt, 16-bit immediate)
- J-type (6-bit opcode, 26-bit pseudo-direct address)

Ref. Computer Architecture: A Quantitative Approach 5th Edition, The Morgan Kaufmann Series, by John L. Hennessy , David A. Patterson

MIPS Instruction format

Instruction encoding formats:

- R-type (6-bit opcode, 5-bit rs, 5-bit rt, 5-bit rd, 5-bit shamt, 6-bit function code)

31-26	25-21	20-16	15-11	10-6	5-0
opcode	*rs*	*rt*	*rd*	shamt	function

- I-type (6-bit opcode, 5-bit rs, 5-bit rt, 16-bit immediate)

31-26	25-21	20-16	15-0
opcode	*rs*	*rt*	*imm*

- J-type (6-bit opcode, 26-bit pseudo-direct address)

31-26	25-0
opcode	*pseudodirect jump address*

Ref. Computer Architecture: A Quantitative Approach 5th Edition, The Morgan Kaufmann Series, by John L. Hennessy , David A. Patterson

Instructions Format: Examples

Ex1

Example: `add $t0, $s1, $s2`
registers have numbers,
`$t0=9, $s1=17, $s2=18`

31-26	25-21	20-16	15-11	10-6	5-0
opcode	*rs*	*rt*	*rd*	shamt	function

rs = identifier of the first source register
rt = identifier of the second source register
rd = identifier of the destination register
shamt = shift amount
funct = differentiates amongst all R-type instructions

Ex2

I-type

lw $t0, 52($s3) lw $8, 52($19)

I-format

6 bits	5 bits	5 bits	16 bits
35	19	8	52
100011	10011	01000	0000 0000 0011 0100

Ref. Computer Architecture: A Quantitative Approach 5th Edition, The Morgan Kaufmann Series, by John L. Hennessy , David A. Patterson

Load/Store Instructions

- ✓ **Two components for load/Store instructions using memory address**
  - ✓ A register whose content are known
  - ✓ An offset stored in 16 bits
- ✓ **Offset**
  - ✓ It is written in terms of number of bytes
  - ✓ It is but in instruction in terms of number of words
  - ✓ 32 byte offset is written as 32 but stored as 8
- ✓ **Computation of Address is**:

  Adding content of register with offset
- ✓ **All address has both these components**
- ✓ Register 0 is used when no register is needed to be used
  - ✓ Register 0 always stores value 0

Ref. Computer Architecture: A Quantitative Approach 5th Edition, The Morgan Kaufmann Series, by John L. Hennessy , David A. Patterson

Instructions

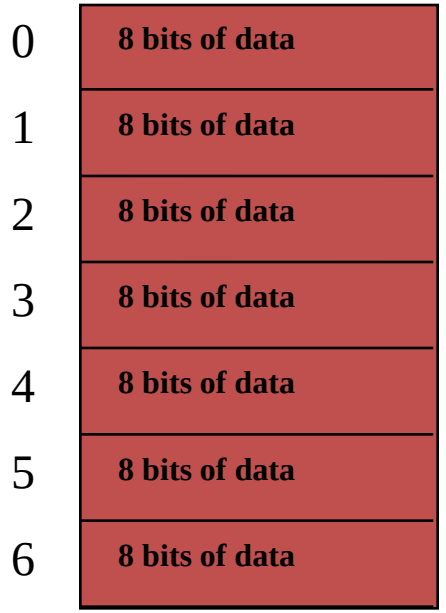
Three operand instruction

Category	Example Instruction		Meaning
Arithmetic	add	$t0, $t1, $t2	$t0 = $t1 + $t2
	sub	$t0, $t1, $t2	$t0 = $t1 – $t2
	addi	$t0, $t1, 100	$t0 = $t1 + 100
	mul	$t0, $t1, $t2	$t0 = $t1 x $t2
	div	$t0, $t1, $t2	$t0 = $t1 / $t2
Logical	and	$t0, $t1, $t2	$t0 = $t1 & $t2 (Logical AND)
	or	$t0, $t1, $t2	$t0 = $t1 | $t2 (Logical OR)
	sll	$t0, $t1, $t2	$t0 = $t1 << $t2 (Shift Left Logical)
	srl	$t0, $t1, $t2	$t0 = $t1 >> $t2 (Shift Right Logical)
Register Setting	move	$t0, $t1	$t0 = $t1
	li	$t0, 100	$t0 = 100
Data Transfer	lw	$t0, 100($t1)	$t0 = Mem[100 + $t1] 4 bytes
	lb	$t0, 100($t1)	$t0 = Mem[100 + $t1] 1 byte
	sw	$t0, 100($t1)	Mem[100 + $t1] = $t0 4 bytes
	sb	$t0, 100($t1)	Mem[100 + $t1] = $t0 1 byte
Branch	beq	$t0, $t1, Label	if ($t0 = $t1) go to Label
	bne	$t0, $t1, Label	if ($t0 ≠ $t1) go to Label
	bge	$t0, $t1, Label	if ($t0 ≥ $t1) go to Label
	bgt	$t0, $t1, Label	if ($t0 > $t1) go to Label
	ble	$t0, $t1, Label	if ($t0 ≤ $t1) go to Label
	blt	$t0, $t1, Label	if ($t0 < $t1) go to Label
Set	slt	$t0, $t1, $t2	if ($t1 < $t2) then $t0 = 1 else $t0 = 0
	slti	$t0, $t1, 100	if ($t1 < 100) then $t0 = 1 else $t0 = 0
Jump	j	Label	go to Label
	jr	$ra	go to address in $ra
	jal	Label	$ra = PC + 4; go to Label

Ref. Computer Architecture: A Quantitative Approach 5th Edition, The Morgan Kaufmann Series, by John L. Hennessy , David A. Patterson

Memory Organization

- Viewed as a large, single-dimension array, with an address
- A memory address is an index into the array
- "Byte addressing" means that successive addresses are one byte apart



Ref. Computer Architecture: A Quantitative Approach 5th Edition, The Morgan Kaufmann Series, by John L. Hennessy , David A. Patterson