Computer Organization

605.204

Module Three

Part Five

MIPS Machine Code

Module Three

- Part Five
- In this presentation, we are going to talk about :
- Hardware Instruction Formats

Previously

- Previously we talked about:

- A People Language - the MIPS Assembly Language

Now: The MIPS Machine Code

A First Set of Instructions

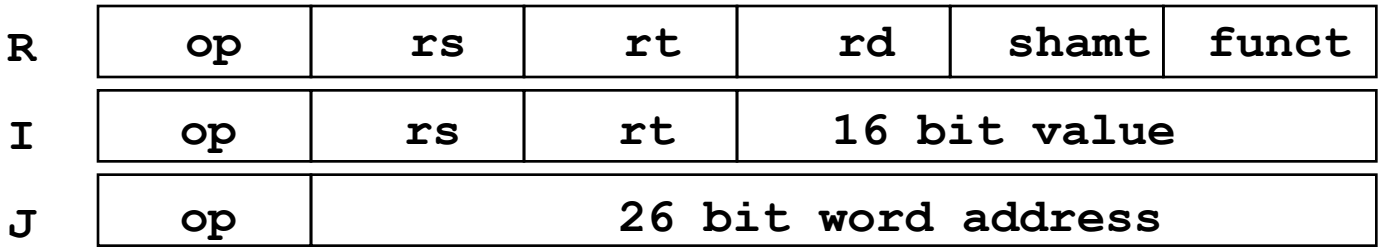
Instruction	Meaning
`add $s1,$s2,$s3`	$s1 = $s2 + $s3
`sub $s1,$s2,$s3`	$s1 = $s2 – $s3
`lw $s1,100($s2)`	$s1 = Memory[$s2+100]
`sw $s1,100($s2)`	Memory[$s2+100] = $s1
`bne $s4,$s5,Label`	Next instr. is at Label if $s4 != $s5
`beq $s4,$s5,Label`	Next instr. is at Label if $s4 = $s5
`j Label`	Next instr. is at Label

MIPS Register Use Convention

Name	Register number	Usage
$zero	0	the constant value 0
$at	1	reserved for Assembler
$v0-$v1	2-3	values for results and expression evaluation
$a0-$a3	4-7	arguments
$t0-$t7	8-15	temporaries
$s0-$s7	16-23	saved
$t8-$t9	24-25	more temporaries
$k0-$k1	26-27	reserved for Operating System
$gp	28	global pointer
$sp	29	stack pointer
$fp	30	frame pointer
$ra	31	return address

MIPS Hardware Instruction Formats

- Only three instruction formats:
  - Register
  - Immediate
  - Jump



Arithmetic Instructions

- Instructions, like registers and words of data, are also 32 bits long
  - Example: **`add $t0, $s1, $s2`**
  - registers have numbers, $t0 = 8, $s1 = 17, $s2 = 18

- Instruction Format: Register

	000000	10001	10010	01000	00000	100000
# bits	6	5	5	5	5	6
	opcode	rs	rt	rd	shamt	funct

- What do the field names stand for?

  Operation Code; Register Source; Register source Two; Register Destination; Shift Amount; Function Code

Load and store Instructions

- Example:

C code: A[8] = h + A[8];

MIPS code:

```
lw  $t0, 32($s3) ; content of  A[8]
add $t0, $s2, $t0; add h
sw  $t0, 32($s3) ; save to A[8]
```

- Instruction Format: Immediate
- Machine code: Memory address = rs + 16 bit value

op	rs	rt	16 bit value

Branch Instructions

- Instructions:
  - **`beq  $t4, $t5, Labelb`**
  - Instruction Format: Immediate

op	rs	rt	16 bit word address

- Addresses are 32 bits

  How did we handle this with load and store instructions?

- Specify a register and add its value to the 16 bit word address.
  - use the Program Counter Register
  - most branches are local - +/- 32K instructions
  - (principle of locality)
- By the way, how do you implement the Compare process?

Instructions for Constants

- Small constants are used quite frequently (50% of operands)

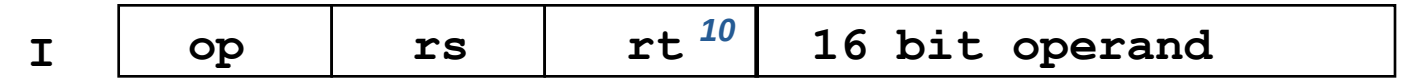
  for example: AB = A + 5;
  CQ = C - 18;

- Solutions ? ?
  - put 'typical constants' in memory and load them, or …
  - create hard-wired registers (like $zero) for constants, or ...

- MIPS Instructions:

```
addi $29, $29, 4
slti $8,  $18, 10
andi $29, $29, 6
ori  $29, $29, 4
```

- How do we make this work?



Jump Instruction

- MIPS unconditional branch instructions:

```
j  label
```

- Example:

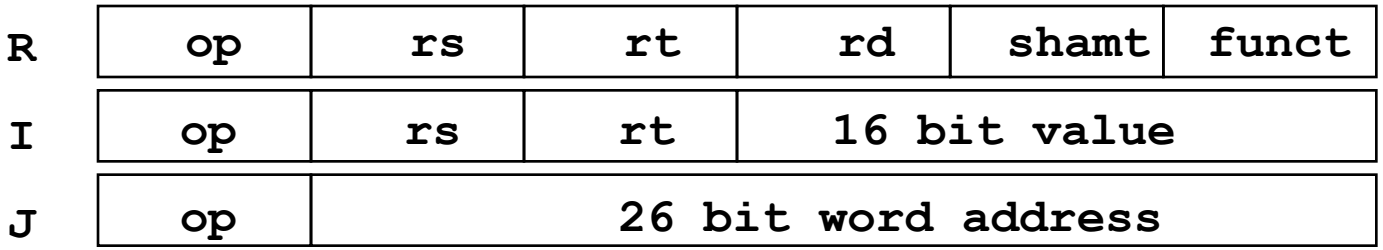
```
if (i!=j)                    beq $s4, $s5, Lab1
   h=i+j;                    add $s3, $s4, $s5
else                         j Lab2
   h=i-j;              Lab1: sub $s3, $s4, $s5
                       Lab2: ...
```

Format:

- Jump instructions use the high order 4 bits of Program Counter
  - address boundaries of 256 MB

MIPS Hardware Instruction Formats

- Only three instruction formats:
  - Register
  - Immediate
  - Jump



Summary

- MIPS Machine Language
- All processing operands are in registers
- Load and Store access to memory
- Three instruction formats
  - Three Register
  - Immediate
  - Jump
- Five addressing modes
  - Immediate, Register, Base, Program Counter, Pseudo-Direct