Chapter 2 (Continued)

Chapter2_3.ppt

Dr. Ning Chen

Translating a MIPS Assembly Instruction into a Machine Instruction

Task: Given a MIPS instruction, translate it to a machine instruction

- Add $t0, $s1, $s2

Ans:

Before we try, let's answer some interesting questions:

1. How many bit(s) do we need to represent one object out of two objects?
2. How many bits do we need to represent one object out of four objects?
3. How many bits do we need to represent one MIPS register?

Ans: 5 bits (since we have 32 registers).

MIPS field (R-type)

op	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

- op: Opcode
- rs: Register source
- rt: Register source
- rd: Register destination
- shamt: shift amount
- funct: Function

MIPS Instruction

- Add $t0, $s1,$s2

Machine code in binary

- 000000 10001 10010 01000 00000 100000

Machine code in Decimal

- 0 17 18 8 0 32

What about lw $t0, 32($S3)?

Q. Can R-type MIPS field work for the lw instruction?

A. No.

Q. What should be the field format for lw?

A.

op	rs	rt	constant or address
6 bits	5 bits	5 bits	16 bits

Example:

- C code

A[300]=h + A[300];

- MIPS assembly code

```
lw $t0, 1200($t1)
add $t0,$s2,$t0
sw $t0, 1200($t1)
```

- Machine code in decimal

op	rs	rt	rd	Address /shamt	function
35	9	8	1200		
0	18	8	8	0	32
43	9	8	1200		

I-Format (Immediate)

- Addi $s3,$s3,4 #$s3=$s3+4
- I-Format

Opcode	rs	rt	Immediate (16 bits)

Logical Operations

- Sll $t2,$s0,4 #reg
  $t2=reg$s0 << 4 bits

op	rs	rt	rd	shamt (5 bits)	funct
0	0	16	10	4	0

Instructions for making decisions

- C code: if –then-else

```
if (j==j) f=g+h;
else
f=g-h;
```

- Assembly code

```
bne $s3,$s4, Else #go to Else if i≠ j
add $s0,$s1,$s2 #f=g+h
J Exit #go to Exit
Else: sub $s0,$s1,$s2
Exit:
```

J-format

- j Exit

opcode	Address (26 bits)

Branch instruction

- Bne $s3, $s4, Else

Machine code format:

opcode	rs	rt	address (16 bits)

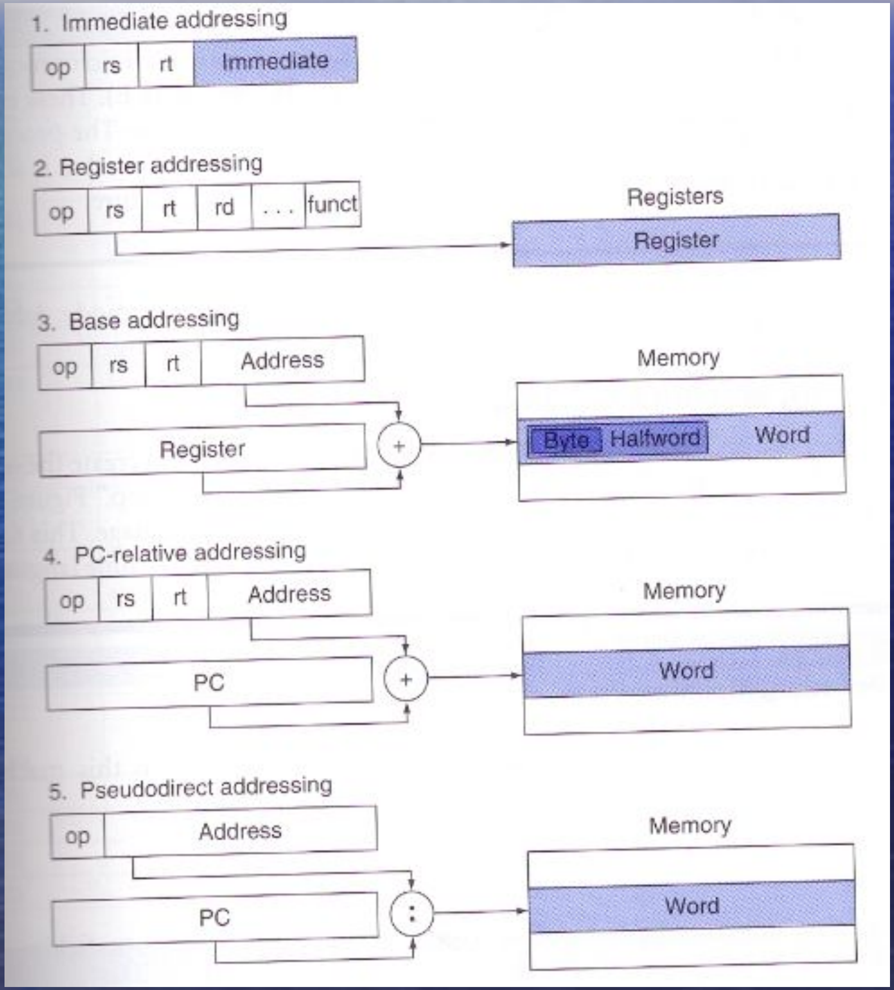
Any observations?

- Branch instructions actually use I-format. Nevertheless, the least significant 16 bits (lower 16 bits) have different meaning.

Five MIPS addressing modes

- Immediate addressing
- Register addressing
- Base addressing
- PC-relative addressing
- Pseudodirect addressing

Five MIPS addressing modes



Architecture design decisions made so far

- 32 registers with 32-bit width
- Byte memory arrangement
- 2^{32} bytes memory space
- 5 addressing modes
- 3 basic instruction formats (R, I and J)

Some architecture design decisions need more careful considerations (will explore those in Chapter 4)

- A branch instruction uses only 16 bits as address. (It does not take you to anywhere in the memory you want to go. Is this a problem?)
- A jump label (e.g., j Exit) instruction uses only 26 bits as address. (It does not take you to anywhere in the memory you want to go. Is this a problem?)

Concluding remarks: Less is more.

- Simplicity favors regularity
- Smaller is faster
- Make the common case fast
- Good design demands good compromise