

# Outline

- Addressing in Branches and Jumps
- Addressing modes summary

## Section 3.8

1

# Addressing in Branches and Jumps<sub>1/7</sub>

- MIPS Jump instruction j (J-type)

Opcode (6 bits) | Address (26 bits)      j opcode is 2

- MIPS bne, and beq (I-type)

Opcode (6 bits) | rs (5 bits) | rt (5 bits) | branch address (16 bits)

- What are the implications of having 16 bit branch address in bne, beq?

Program addresses would have to fit in 16-bits, which means no program could be bigger than  $2^{16}$

- What is the solution?

Add a register (Program Counter) to the branch address

Address of instruction to branch to = PC + branch address

Which allows program sizes to be as large as  $2^{32}$

2

# Addressing in Branches and Jumps<sub>2/7</sub>

- Why PC?

- PC holds the address of the current instruction

- Measurements have shown that conditional branches tend to branch to a nearby instruction

- If we use one bit for sign in branch address, we have 15 bits left, which means we can branch to  $\pm 2^{15}$  words of the current instruction (Measurements show that almost all loops are smaller than  $2^{16}$  words)

- At the end of the fetch phase, PC is incremented to point to next instruction (PC+4)

- Actually, MIPS branch address is relative to the address of following instruction (PC+4) as opposed to the current instruction (PC)

- This form of addressing is termed PC-relative addressing

3

# Addressing in Branches and Jumps<sub>3/7</sub>

- How to interpret the branch address?

- Number of bytes after the next instruction? OR

- Number of words after the next instruction?

- A word is 4 bytes

- We can branch 4 times as far (16-bit field) by interpreting the branch address as a relative word address instead of a relative byte address

- Relative byte address = relative word address \* 4

4

## Addressing in Branches and Jumps<sub>4/7</sub>

### Example

beq \$s3, \$s4, L1

add \$s0,\$s1,\$s2

L1: sub \$s0,\$s0,\$s3

What is the machine code of beq instruction?

•beq is I-format(opcode, rs, rt, branch address)

Opcode is 4, rs (\$s3) is 19, rt (\$s4) is 20, branch address is 1

•Why is the branch address 1?

➤When we are executing beq instruction, after fetch phase PC is incremented to PC+4 to point to add instruction

➤If \$s3 = \$s4, in the execution phase for beq instruction, we need to skip 1 \* 4 bytes relative to the address of add instruction (the current value of the PC)

5

## Addressing in Branches and Jumps<sub>5/7</sub>

### How about the 26 bit address field in j instruction?

•The 26-bit address field is also a word address (28-bit byte address)

•How to get the 28-bit byte address from the 26-bit byte address?

➤Add two zeros as low order bits to 26 bits to obtain 28 bits (Why?)

•The instruction address is 32 bits, we need 4 more bits

•What is the solution?

➤The 28-bit byte address replaces the lower 28 bits of the PC leaving the upper 4 bits unchanged

➤This form of addressing is called **pseudodirect addressing**

•How far can the jump be?

$2^{28} = 2^8 * 2^{20} = 256 \text{ MB}$  or  $256 \text{ MB}/4 = 64 \text{ Mwords}$  (64 million instructions)

6

## Addressing in Branches and Jumps<sub>6/7</sub>

### Example page 149

Loop: add \$t1, \$s3, \$s3 # \$t1 ← 2 \* i

add \$t1, \$t1, \$t1 # \$t1 ← 4 \* i

add \$t1, \$t1, \$s6 # \$t1 ← address of save[i]

lw \$t0, 0(\$t1) # \$t0 ← save[i]

bne \$t0, \$s5, Exit # go to Exit if save [i] != k

add \$s3, \$s3, \$s4 # i = i + j

j Loop # go to Loop

Exit:

Loop started at location 80000 in memory , what is the MIPS machine code for loop

80000	0	19	19	9	0	32
80004	0	9	9	9	0	32
80008	0	9	22	9	0	32
80012	35	9	8			0
80016	5	8	21			2
80020	0	19	20	19	0	32
80024	2					20000
80028						

Machine code with addresses

•Note that textbook uses 8 for branch address in bne instruction, then later states it should be 2 on page 150. PC-relative addressing mode refers to NUMBER OF WORDS

7

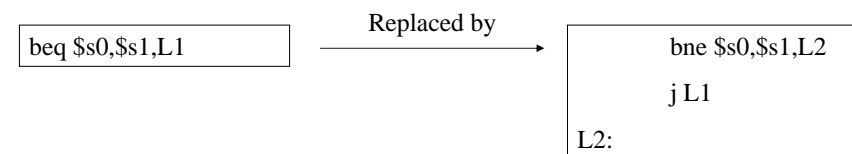
## Addressing in Branches and Jumps<sub>7/7</sub>

### Branching Far Away

beq \$s0, \$s1, L1

•If this conditional branch is a branch to a far away location, the assembler transforms the code to replace with a conditional branch to a nearby location and a j instruction

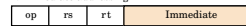
•Why this replacement makes it more feasible to branch to a far away location?



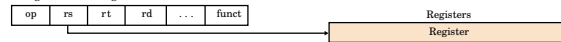
8

# Addressing Modes Summary

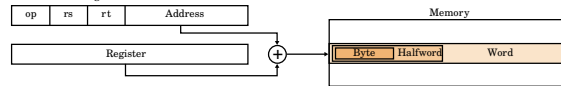
## 1. Immediate addressing



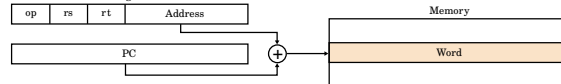
## 2. Register addressing



## 3. Base addressing



## 4. PC-relative addressing



## 5. Pseudodirect addressing

