#### **Chapter 6 :: Architecture**

Quiz 47 opcode 2 niz 47 opcode 7 (6 bits

## 6.9-6.10 MIPS instructions

Digital Design and Computer Architecture

David Money Harris and Sarah L. Harris

## **High-Level Code Constructs**

- if statements
- if/else statements
- while loops
- for loops

### If Statement

#### C Code

#### MIPS assembly code

bee/bre by itself

## If Statement

#### C Code

#### MIPS assembly code

```
# $s0 = f, $s1 = g, $s2 = h
# $s3 = i, $s4 = j
bne $s3, $s4, L1
add $s0, $s1, $s2
L1: sub $s0, $s0, $s3
```



Assembly tests opposite case (i != j) of high-level code (i == j)

## If/Else Statement

#### C Code

MIPS assembly code

Sed by

Sollowed by

That jumps

Forward

## If/Else Statement

#### C Code

#### MIPS assembly code

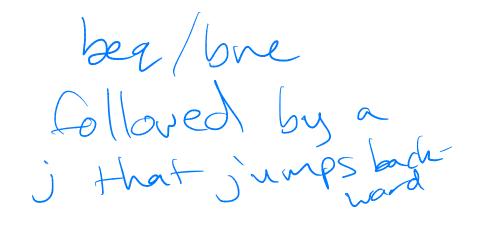
# While Loops

#### C Code

```
// determines the power
// of x such that 2x = 128
int pow = 1;
int x = 0;

while (pow != 128) {
  pow = pow * 2;
  x = x + 1;
```

#### MIPS assembly code





Assembly tests for the opposite case (pow == 128) of the C code (pow != 128).



## While Loops

#### C Code

#### MIPS assembly code

```
// determines the power # $s0 = pow, $s1 = x

// of x such that 2^x = 128

int pow = 1;

int x = 0;

while (pow)!= 128) {

pow = pow * 2;

x = x + 1;

}

while the power # $s0 = pow, $s1 = x

addi $s0, $0, 1

addi $s1, $0, $0

addi $t0, $0, 128

while: beq $s0, $t0, done

sll $s0, $s0, 1

addi $s1, $s1, 1

j while repeat the cop

done:
```

Assembly tests for the opposite case (pow == 128) of the C code (pow != 128).

## For Loops

```
for (initialization; condition; loop operation)
  statement
```

- initialization: executes before the loop begins
- condition: is tested at the beginning of each iteration
- loop operation: executes at the end of each iteration
- statement: executes each time the condition is met

## For Loops

#### C Code

```
// add the numbers from 0 to 9
int sum = 0;
int i;

for (i=0; i!=10; i = i+1) {
   sum = sum + i;
}
```

#### MIPS assembly code

braven followed by jump back

## For Loops

```
C Code
                                    MIPS assembly code
                                    \# \$s0 = i, \$s1 = sum
// add the numbers from 0 to 9
int sum = 0;
                                           addi $s1, $0, 0
                                          add $s0, $0, $0
int i;
                                           addi $t0, $0, 10 (oustant
                                           beq $s0, $t0, done
for (i=0; (i!=10; / i = i+1)) {
                                   for:
                                           add $s1, $s1, $s0
  sum = sum + i;
                                           addi $s0, $s0, 1
                                                for
                                    done:
```

stop lese a/22/25

(or refer to)

### How do we address the operands?

- Register Only
- Immediate
- Base Addressing
- PC-Relative
- Pseudo Direct

# Register Only 7

- Operands found in registers
  - **Example:** add \$s0, \$t2, \$t3
  - Example: sub \$t8, \$s1, \$0

### Immediate 7 most i-type

- 16-bit immediate used as an operand

  - Example: addi \$s4, \$t5, -73
     Example: ori \$t3, \$t7, 0xFF

Base Addressing (base + offset)

• Address of operand is: weword instructions

base address + sign-extended immediate

- Example: lw \$s4, 72(\$0) • address = \$0 + 72
- Example: sw \$t2, −25(\$t1)
  - address = \$t1 25

**PC-Relative Addressing** 

```
\rightarrow beq $t0, $0, else

  addi $v0, $0, 1 #else=3: (0x20-0x14)/4

                         $sp, $sp, i
0x18
0x1C
                                 $ra
                    addi $a0, $a0, -1
0x20
0x24
                           factorial
```

#### **Assembly Code**

#### Field Values

imm

rt

			OP	10				
beq \$t0,	\$O,	else	4	8	0		3	
(beq \$t0,	\$O,	3)	6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

rs

On

Pseudo-direct Addr	
0x0040005C jal	sum address directly
0x004000A0 sum: add	\$v0, \$a0, \$a1 add (ess
JTA 0000 0000 010	00 0000 0000 0000 1010 0000 (0x004000A0)
	0 0 0 0 2 8 (0x0100028)
Field Values  op imm  3 0x0100028	Machine Code op addr 000011 00 0001 0000 0000 0000 0010 1000 (0x0C100028)
6 bits 26 bits	6 bits 26 bits

# How to Compile & Run a Program High Level Code

