

LAB 04: Flow Control

Saleh AlSaleh

salehs@kfupm.edu.sa

King Fahd University of Petroleum and Minerals
College of Computing and Mathematics
Computer Engineering Department

COE301: Computer Architecture
Term 222

Agenda

① Unconditional Jump

② Conditional Jump

③ Pseudo Instructions

④ Examples

⑤ Tasks

Unconditional Jump

- Code Labels are used to define important locations in code.

Unconditional Jump

- Code Labels are used to define important locations in code.
- jump instruction is used to jump to another location in code unconditionally.

Unconditional Jump

- Code Labels are used to define important locations in code.
- jump instruction is used to jump to another location in code unconditionally.
- Syntax: `j label`

Conditional Jump

- Branch instructions is used to jump to another location in code if a condition is satisfied.

Conditional Jump

- Branch instructions is used to jump to another location in code if a condition is satisfied.
- Basic Branch Instructions: `beq`, `bne`, `blez`, `bgtz`, `bltz`, `bgez`

Conditional Jump

- Branch instructions is used to jump to another location in code if a condition is satisfied.
- Basic Branch Instructions: `beq`, `bne`, `blez`, `bgtz`, `bltz`, `bgez`
- Syntax: `beq $op1, $op2, label2`
if value in `$op1` is equal to the value in `$op2`, go to label2.
- Used in loops and if statements

Branch Pseudo Instructions

- blt, bltu
- ble, bleu
- bgt, bgtu
- bge, bgeu
- e.g. `blt $s1, $s2, label` ⇒ `slt $at, $s1, $s2`
`bne $at, $zero, label`

Example #1: if statement

```
if (a==b)
{
    c = d + e ;
}
else
{
    c = d - e ;
}
```

Assume **a, b, c, d, e**
are stored in
\$s0, \$s1, \$s2, \$s3, \$s4
respectively.

Example #1: if statement

```

if (a==b)
{
    c = d + e ;
}
else
{
    c = d - e ;
}

```

beq \$s0, \$s1, true
 # false cond here
 sub \$s2, \$s3, \$s4
 j exit
 true:
 add \$s2, \$s3, \$s4
 exit:

...

Assume a, b, c, d, e
 are stored in
 \$s0, \$s1, \$s2, \$s3, \$s4
 respectively.

Example #1: if statement

```

if (a==b)
{
    c = d + e ;
}
else
{
    c = d - e ;
}

```

```

beq $s0, $s1, true
# false cond here
sub $s2, $s3, $s4
j exit
true:
add $s2, $s3, $s4
exit:
...

```

```

bne $s0, $s1, false
# true cond here
add $s2, $s3, $s4
j exit
false:
sub $s2, $s3, $s4
exit:
...

```

Assume **a, b, c, d, e**
are stored in
\$s0, \$s1, \$s2, \$s3, \$s4
respectively.

Example #2: for loop

```
for (int i=0; i<n; i++)  
{  
    //loop body  
}
```

Assume `i` is stored in
`$s0` and `n` is stored
in `$s1`.

Example #2: for loop

```
for (int i=0; i<n; i++)  
{  
    //loop body  
}
```

Assume **i** is stored in
\$s0 and **n** is stored
in **\$s1**.

```
li $s0, 0  
loop:  
bge $s0, $s1, endLoop  
# loop body  
addi $s0, $s0, 1  
j loop  
endLoop:  
...
```

Example #2: for loop

```
for (int i=0; i<n; i++)
{
    //loop body
}
```

Assume **i** is stored in **\$s0** and **n** is stored in **\$s1**.

```
li $s0, 0
loop:
bge $s0, $s1, endLoop
# loop body
addi $s0, $s0, 1
j loop
endLoop:
...
```

```
li $s0, 0
loopCheck:
blt $s0, $s1, loop
...
loop:
# loop body
addi $s0, $s0, 1
j loopCheck
```

Live Examples

Task #1

Write a MIPS program where you ask the user to enter a character. Then, print one of the following messages based on the user's input.

- Uppercase
- Lowercase
- Digit
- Special Character

Sample Run 1

Enter a character: a
Lowercase

Sample Run 2

Enter a character: \$
Special Character

Task #2

Write a MIPS assembly program that reads 6 integers and correctly report back their sum.

NOTE: Reading the 6 integers should be done in a loop not by repeating the reading instructions six times.

Sample Run

```
Enter integer 0: 5
Enter integer 1: 12
Enter integer 2: 76
Enter integer 3: 43
Enter integer 4: 37
Enter integer 5: 58
Sum = 231
```