5.16 If-else expressions

Comparing variables for equal or not equal

An earlier section showed that x == y should use a bne instruction in assembly. Conversely, an x != y should use a beq instru assembly below, when x does not equal y, execution falls through beq to the If substatement, as desired. When x equals y, b After, skipping the If substatement.

Figure 5.16.1: != uses beq.

```
if (x != y) {
   w = w + 50;
}
```

```
#
      beq $t0, $t1, After
      addi $t3, $t3, 50 # If substatement
After:
```

PARTICIPATION ACTIVITY 5.16.1: bne for == and beq for !=.

For the given C expression that completes the shown C, choose the correct assembly instruction to complete the shown assembly.

```
if ( _____ ) {
   w = w + 50;
}
```

```
#
      ___ ___ ___ After
      addi $t3, $t3, 50
After:
```

1) x == y

- beq $t0, $t1
-

- bne $t0, $t1

2) x != y
   - beq $t0, $t1
   - bne $t0, $t1

3) y == x
   - beq $t0, $t1
   - bne $t0, $t1

4) x == 0
   - beq $t0, $zero
   - bne $t0, $zero

5) x != 0
   - beq $t0, $zero
   - bne $t0, $zero

6) y != 0
   - beq $zero, $t1
   - bne $zero, $t1

Other comparisons

Common if expressions are not just equal or not equal, but also <, ≤, >, and ≥. MIPS pseudoinstructions exist for the latter f later in this section).

Table 5.16.1: MIPS branch instructions for various comparisons.

MIPS instruction	Example	Meaning
beq: Branch on equal	`beq $t0, $t1, L`	Branch if $t0 equals $t1
bne: Branch on not equal	`bne $t0, $t1, L`	Branch if $t0 does not equal $t1
blt: Branch on less than	`blt $t0, $t1, L`	Branch if $t0 < $t1
ble: Branch on less than or equal	`ble $t0, $t1, L`	Branch if $t0 ≤ $t1
bgt: Branch on greater than	`bgt $t0, $t1, L`	Branch if $t0 > $t1
bge: Branch on greater than or equal	`bge $t0, $t1, L`	Branch if $t0 ≥ $t1

(Above, L means Label, and, "$t0 equals $t1" actually means "$t0's value equals $t1's value")

An earlier section showed that an efficient pattern in assembly for if statements involving == or != uses the opposite compa assembly: == uses bne, != uses beq. Similarly, opposites should be used for the other comparisons.

Table 5.16.2: Comparison opposites.

Comparison	Opposite comparison
equal	not equal
not equal	equal
<	≥
≤	>
>	≤
≥	<

PARTICIPATION ACTIVITY 5.16.2: Various comparisons.

For each question's C expression that completes the given C, choose the correct assembly instruction to complete the assembly.

```
if ( _____ ) {
   w = w + 50;
}
```

```
#
   __________
   addi $t3, $t3, 50
N:
```

1) x == y
   - beq $t0, $t1, N
   - bne $t0, $t1, N

2) x != y
   - beq $t0, $t1, N
   - bne $t0, $t1, N

3) x < y
   - blt $t0, $t1, N
   - bge $t0, $t1, N

4) x >= y
   - blt $t0, $t1, N
   - bge $t0, $t1, N

5) x > 0
   - blt $t0, $zero, N
   - ble $t0, $zero, N

6) x <= 0

- bgt $t0, $zero, N
- bge $t0, $zero, N

Comparing with an expression rather than a variable

Earlier examples have compared with variables, like x == y. Sometimes an if statement in C compares with an expression, li implement in assembly, one can first compute the expression and write the result to a register, and then compare with that

Figure 5.16.2: Comparing with an expression is done by first writing the expression's result to a register.

```
if ((x - y) == z) {
   w = w + 50;
}
```

```
#
      sub $t5, $t0, $t1  # $t5 = x - y
      bne $t5, $t2, After # Compares (x - y) with z
      addi $t3, $t3, 50  # If substatement
After:
```

PARTICIPATION ACTIVITY 5.16.3: Comparing with expressions.

Implement the C by completing the assembly. Assume $t0 has x's value, $t1 has y's, $t2 has z's.

1) C

```
if ((x - y) == z) {
   w = w + 50;
}
```

Assembly

```
#
      sub $t5, $t0, $t1
      bne ___, $t2, After
      addi $t3, $t3, 50
After:
```

Check Show answer

2) C

```
if ((x + y) == z) {
   w = w + 50;
}
```

Assembly

```
#
      ___ ___, $t0, $t1
      bne $t5, $t2, After
      addi $t3, $t3, 50 # If substatement
After:
```

Check Show answer

3) C

```
if ((x + y) > z) {
   w = w + 50;
}
```

Assembly

```
#
      add $t5, $t0, $t1
      ___ ___, $t2, After
      addi $t3, $t3, 50 # If substatement
After:
```

Check Show answer

4) C

```
if ((x + y + 9) != z) {
   w = w + 50;
}
```

Assembly

```
#
      add $t5, $t0, $t1
      ___ ___ ___ ___
      beq  $t5, $t2, After
      addi $t3, $t3, 50 # If substatement
After:
```

Check Show answer

5) C

```
if ((x + y) == (x * y) {
   w = w + 50;
}
```

Assembly

```
#
      add $t4, $t0, $t1
      mul $t5, $t0, $t1
      ___ ___, $t5, After
      addi $t3, $t3, 50 # If substatement
After:
```

Check Show answer

6) C

```
if ((x + 3) >= (x * (y - 1)) {
   w = w + 50;
}
```

Assembly

```
#
      addi $t4, $t0, 3  # x + 3
      sub $t5, $t1, 1   # y - 1
      mul $t5, $t0, $t5 # x*(y-1)

      ___ ___ ___ ____
      addi $t3, $t3, 50 # If substatement
After:
```

Check Show answer

CHALLENGE ACTIVITY 5.16.1: If-else expressions in assembly.

Start

Convert the C to assembly. Variables: x is in $t1, y is in $t2, and z is in $t3.

```
if (y != z) {
    x = x + 50;
}
```

```
bne  $t2, $t2, After
addi $t1, $t1, 50
After:
```

Registers	
$t1	11
$t2	3
$t3	2

1 2 3 4 5

Check Next

1 2 3 4 5

Provide feedback on this section