5.13 beq, bne, j: Branch and jump instructions

Branch instructions: beq, bne

A ***branch*** instruction specifies the location of the next instruction to execute, depending on the branch instruction's conditic
equal (***beq***) instruction branches to an instruction at a specified location if the values held in two registers are equal. If the \
the branch is taken, and the instruction at the specified location is executed. Otherwise, the branch is not taken, and the ins
immediately following the branch instruction is executed.

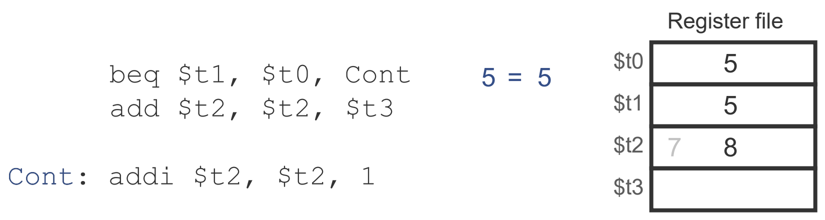
A branch instruction typically uses a label to specify the next instruction's location. A ***label*** is a named position in a progran
an instruction's memory address. The MIPS beq instruction format below branches to the instruction at location Label if th
in regA and regB are equal.

```
beq regA, regB, Label
```

A label is a sequence of letters (a-z, A-Z, _) and digits (0-9) starting with a letter and followed by a colon (:).

PARTICIPATION ACTIVITY 5.13.1: Branch on equal (beq) instruction.

Start 2x speed



```
      beq $t1, $t0, Cont
      add $t2, $t2, $t3

Cont: addi $t2, $t2, 1
```

PARTICIPATION ACTIVITY 5.13.2: beq instruction.

Which instruction is executed immediately after the branch instruction. Assume initial register values of:

- $t0: 5
- $t1: 10
- $t2: 0
- $t3: 10

1)
```
      beq $t1, $t3, Cont
      sub $t1, $t1, $t5
Cont: sw $t4, 0($t6)
```

2)
```
      beq $t0, $t1, Cont
      sw $t1, 0($t5)
Cont:  addi $t1, $t1, -2
```

3)
```
      beq $t2, $zero, Cont
      addi $t4, $t4, 11
      sw $t4, 0($t6)
Cont:  lw $t2, 0($t6)
```

4)
```
      beq $t3, $t1, Cont
      addi $t3, $t3, 2
Cont:
      sub $t3, $t3, $t5
```

PARTICIPATION ACTIVITY 5.13.3: Labels.

Which are valid labels for the addi instruction?

1)
```
Cont: addi $t2, $t2, 1
```
- ○ Valid
- ○ Invalid

2)
```
After_Adjust: addi $t2, $t2, 1
```
- ○ Valid
- ○ Invalid

3)
```
userValEq3: addi $t2, $t2, 1
```
- ○ Valid
- ○ Invalid

4)
```
IsGood?: addi $t2, $t2, 1
```
- ○ Valid
- ○ Invalid

5)
```
Grade equals 100: addi $t2,
$t2, 1
```
- ○ Valid
- ○ Invalid

6)
```
CheckResult:
   addi $t2, $t2, 1
```
- ○ Valid
- ○

Invalid

7)

```
2ndTask: addi $t2, $t2, 1
```

○ Valid

○ Invalid

A ***branch on not equal*** (***bne***) instruction branches to an instruction at a specified location if the values held in two registers
The MIPS bne instruction format below branches to the instruction at Label if the values held in regA and regB are not equa

```
bne regA, regB, Label
```

PARTICIPATION ACTIVITY 5.13.4: Branch instructions: bne and beq.

For each question, assume initial register values of:

- $t0: 20
- $t1: 15
- $t2: 15
- $t3: 21

1) After the following, what is $t3?

```
    bne $t0, $t1, Cont
    addi $t3, $t3, 5
Cont: addi $t2, $t2, 2
```

Check Show answer

2) After the following, what is $t3?

```
   bne $t1, $t2, Cont
   addi $t3, $t3, 7
Cont: addi $t2, $t2, 3
```

Check Show answer

3) After the following, what is $t2?

```
   bne $t1, $t2, Cont
   addi $t3, $t3, 7
Cont: addi $t2, $t2, 3
```

Check Show answer

4) After the following, what is $t3?

```
   bne $t2, $t1, Cont
   addi $t3, $t3, 8
Cont: addi $t3, $t3, 4
```

Check Show answer

5) How many instructions execute in the following?

```
   bne $t2, $t0, Cont1
   addi $t3, $t3, 8
Cont1:
   bne $t2, $t1, Cont2
```

```
    addi $t3, $t3, 5
Cont2:
    addi $t3, $t3, 7
```

Check Show answer

Jump instruction

A ***jump*** (***j***) instruction specifies the location of the next execution to execute. A jump instruction is also know as an uncondi
The MIPS j instruction format below jumps to the instruction at Label.

```
j Label
```

PARTICIPATION ACTIVITY 5.13.5: Jump (j) instruction.

Start 2x speed

```
        j After
CalcE:  # other instructions
        # ...
After:  addi $t2, $t2, 1
```

PARTICIPATION ACTIVITY 5.13.6: Jump instructions.

1) A jump instruction will always jump to

the labeled instruction.

○ True

○ False

2) A jump instruction can only jump to a labeled instruction located after the jump instruction.

○ True

○ False

3) Which instruction is executed after the jump instruction?

```
j Comp2
Comp1: addi $t2, $zero, -5
Comp2: sw $t3, 0($t5)
```

○ addi

○ sw

Branch and jump instructions are commonly used together to direct a program to conditionally execute either one group of another group, but not both. A branch instruction is used to decide which group of statements to execute. If the branch is t instruction group at the label specified in the branch is executed. If the branch is not taken, the instruction group after the b executed. That instruction group ends with a jump instruction to the first instruction after the other instruction group, so th instruction group is not executed.

PARTICIPATION ACTIVITY 5.13.7: Using branch and jump instructions to execute one of two instruction groups.

Start ☐ 2x speed

```
beq $t1, $t2, Equal
```

```
beq $t1, $t2, Equal
```

```
        # instructions executed
        # if not equal
        add $t3, $t3, $t0
        j After
Equal:  # instructions executed
        # if equal
        addi $t3, $t3, 25
After:  # instructions executed
        # afterward
```

Branch taken
(Highlighted instructions executed)

```
        # instructions executed
        # if not equal
        add $t3, $t3, $t0
        j After
Equal:  # instructions executed
        # if equal
        addi $t3, $t3, 25
After:  # instructions executed
        # afterward
```

Branch not taken
(Highlighted instructions executed)

PARTICIPATION ACTIVITY 5.13.8: Branch and jumps instructions.

Refer to the animation above.

1) Assume initial register values of $t1: 4, $t2: 7.

How many instructions are executed?

Check Show answer

2) Assume initial register values of $t1: 10, $t2: 10.

How many instructions are executed?

Check Show answer

3) Assume initial register values of $t0: 5, $t1: 10, $t2: 10, $t3: 20.

What is $t3 when execution reaches the label After?

Check Show answer

4) Assume initial register values of $t0: 5, $t1: 4, $t2: 18, $t3: 20.

What is $t3 when execution reaches the label After?

Check Show answer

PARTICIPATION ACTIVITY 5.13.9: Branch and jump instruction example.

The assembly program below adds 5 to DM[5004] if DM[5000] is 100. Otherwise, the program adds 10 to DM[5000]. The sum is stored in DM[5008].

1. Run the simulation step-by-step, observing memory values.
2. Change DM[5000]'s value to 95, then run again.
3. Modify the program to add 5 to DM[5004] if DM[5000] is 100, add 10 if DM[5000] is 95, and add 20 otherwise.

Assembly

```
    addi $t5, $zero, 5000
    lw $t0, 0($t5)     # Load DM[5000]
    addi $t5, $zero, 5004
    lw $t1, 0($t5)     # Load DM[5004]
    addi $t2, $zero, 100
    bne $t0, $t2, Add10
    addi $t1, $t1, 5  # Add 5
    j After
Add10:
    addi $t1, $t1, 10 # Add 10
After:
    addi $t5, $zero, 5008
    sw $t1, 0($t5)     # Store sum to DM[5008]
```

Registers

$zero	0
$t0	0
$t1	0
$t2	0
$t3	0
$t4	0
$t5	0

+

D

5000	
5004	
5008	

+

ENTER SIMULATION STEP RUN

More options

Table 5.13.1: Instruction summary: beq, bne, j.

Instruction	Format	Description	Example
beq	`beq $a, $b, BLabel`	Branch on equal: Branches to the instruction at BLabel if the values held in $a and $b are equal. Otherwise, instruction	`beq $t3, $t2, SumEq5`

		immediately after beq is executed.	
bne	`bne $a, $b, BLabel`	Branch on not equal: Branches to the instruction at BLabel if the values held in $a and $b are not equal. Otherwise, instruction immediately after bne is executed.	`bne $t4, $t5, GuessNeqCorrect`
j	`j JLabel`	Jump: Causes execution to continue with the instruction at JLabel.	`j CalcTip`

CHALLENGE ACTIVITY 5.13.1: Branch and jump instructions.

1
2
3
4
5

Start

Convert pseudocode to MIPS:

```
if ($t0 != $t1)
   $t2 = $t2 + $t2;
```

```
add ▼  $t2 ▼ ,  $t2 ▼ ,  $t2 ▼
add ▼  $t2 ▼ ,  $t2 ▼ ,  $t2 ▼
L1:
```

1	2	3	4	5

Check Next

Provide feedback on this section