

Microprocessadores

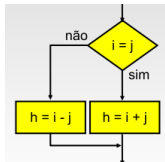
Hugo Marcondes
hugo.marcondes@ifsc.edu.br

Aula 05

Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina

Instruções de controle

- Branch on not equal (bne):
 - Desvia o programa para <label1> se \$t0 != \$t1
 - Ex: bne \$t0, \$t1, label1 #if (\$t0 != \$t1) goto label1



```
bne $s8, $s9, else
add $t0, $s8, $s9
j sai
else: sub $t0, $s8, $s9
sai: nop
```

- Branch on equal (beq):
 - Desvia o programa para <label2> se \$t0 == \$t1
 - Ex: beq \$t0, \$t1, label2 #if (\$t0 == \$t1) goto label2

² IFSC - Departamento Acadêmico de Eletrônica

Instruções de controle

- Instruções BNE e BEQ:
 - Campo "immediate" possui **quantidade de palavras** (words) que devem ser **saltadas** para chegar à **instrução** marcada pelo **label** (rótulo);
 - O Número pode ser positivo (**desvio** para frente) e negativo (**desvio** para trás);
- Instrução J (jump):
 - Definição do **endereço da memória** correspondente à **instrução** marcada pelo **label**;
 - Novo endereço: **4 MSB do PC atual + 26 Bits da instrução deslocado à esquerda de 2 Bits** (como o **endereço da memória** é um **múltiplo de 4 bytes**);

³ IFSC - Departamento Acadêmico de Eletrônica

Estruturas de Controle

Como implementar:

- Inserir rótulos no segmento de texto;
- Testes e desvios (branches e jumps) para os rótulos;

Jump instructions (unconditional branches)
Jump j label # goto label
Jump register jr \$t1 # goto the address in \$t1

```
# Basic instructions
beq $t1, $t2, label # if ($t1 == $t2) goto label
bne $t1, $t2, label # if ($t1 != $t2) goto label
bgez $t1, label # if ($t1 >= 0) goto label
bgtz $t1, label # if ($t1 > 0) goto label
blez $t1, label # if ($t1 <= 0) goto label
bltz $t1, label # if ($t1 < 0) goto label

# Macro instructions
beqz $t1, label # if ($t1 == 0) goto label
bneqz $t1, label # if ($t1 != 0) goto label
beq $t1, 123, label # if ($t1 == 123) goto label
bne $t1, 123, label # if ($t1 != 123) goto label
bge $t1, $t2, label # if ($t1 >= $t2) goto label
bgt $t1, $t2, label # if ($t1 > $t2) goto label
bge $t1, 123, label # if ($t1 >= 123) goto label
bgt $t1, 123, label # if ($t1 > 123) goto label
and similarly for ble and blt
```

J = Jump → label
JR = Jump Reg. → address reg.

⁴ IFSC - Departamento Acadêmico de Eletrônica

LMEBRANDO QUE:

1 = VERDADE
0 = MENTIRA

Declaração IF-THEN-ELSE



Structure of an if-then-else statement

```
if (condition) {
    then-block (execute if condition is true)
} else {
    else-block (execute if condition is false)
}
```

Sketch of translation to assembly

```
(translation of condition, ending in branch to thenLabel)
(translation of else-block)
j endLabel
thenLabel:
(translation of then-block)
endLabel:
(rest of program)
```

5 IFSC - Departamento Acadêmico de Eletrônica

Declaração IF-THEN-ELSE



Example

```
# Pseudocode:
# if (a < b + 3)
#   a = a + 1
# else
#   a = a + 2
# b = b + a
# Register mappings:
# a: $t0, b: $t1

    addi $t2, $t1, 3    # tmp = b + 3
    blt  $t0, $t2, then # if (a < tmp)
    addi $t0, $t0, 2    # (else case) a = a + 2
    j    end
then: addi $t0, $t0, 1    # (then case) a = a + 1
end:  add  $t1, $t1, $t0  # b = b + a
```

Handwritten notes: "Se a < tmp" with an arrow pointing to the `blt` instruction. "a = a + 2" with an arrow pointing to the `addi $t0, $t0, 2` instruction. "Se a < tmp" with an arrow pointing to the `addi $t0, $t0, 1` instruction.

6 IFSC - Departamento Acadêmico de Eletrônica

Declaração IF-THEN



Example of first strategy

```
# Pseudocode:
# if (a < b + 3)
#   a = a + 1
# b = b + a
# Register mappings:
# a: $t0, b: $t1

    addi $t2, $t1, 3    # tmp = b + 3
    blt  $t0, $t2, then # if (a < tmp)
    j    end
then: addi $t0, $t0, 1    # (then case) a = a + 1
end:  add  $t1, $t1, $t0  # b = b + a
```

Handwritten notes: "Se a < tmp" with an arrow pointing to the `blt` instruction. "vai p/then" with an arrow pointing to the `then` label. "Jump to end." with an arrow pointing to the `end` label.

7 IFSC - Departamento Acadêmico de Eletrônica

```
beq $s0, $t1, SUBTN # if(input==2) {goto SUBTN}
beq $s0, $t0, ADDTN # if(input==1) {goto ADDTN}
beq $s0, $t2, MULT  # if(input==3) {goto MULT}
beq $s0, $t3, DIV   # if(input==4) {goto DIV}
```

```
ADDTN:
# Code for addition
j EXIT
```

```
SUBTN:
# Code for subtraction
j EXIT
```

```
MULT:
# Code for multiplication
j EXIT
```

```
DIV:
# Code for division
j EXIT
```

```
EXIT:
# Code to exit
j EXIT
```

Declaração IF-THEN



Example of second strategy

```
# Pseudocode:
# if (a < b + 3)
#   a + 1
# b = b + a
# Register mappings:
# a: $t0, b: $t1

    addi $t2, $t1, 3    # tmp = b + 3
    bge  $t0, $t2, end  # (if (a >= tmp) goto end)
    addi $t0, $t0, 1    # a + 1
    add  $t1, $t1, $t0  # b = b + a
end:
```

Handwritten note: "Se a >= tmp" with an arrow pointing to the `bge` instruction.

8 IFSC - Departamento Acadêmico de Eletrônica

Laço DO-WHILE



Structure of a do-while loop

```
do {  
  loop-body  
} while (condition);
```

Sketch of translation to assembly

```
loopLabel:  
  (translation of loop-body)  
  (translation of condition, ending in branch to loopLabel)  
  (rest of program)
```

Laço DO-WHILE



Example

```
# Pseudocode:  
# do {  
#   a = a + 3  
# } while (a < b*2)  
# Register mappings:  
# a: $t0, b: $t1  
  
loop:  addi $t0, $t0, 3    # (loop) a = a + 3  
       mul  $t2, $t1, 2   # tmp = b*2  
       blt  $t0, $t2, loop # if (a < tmp) goto loop
```

Optimization: Extract loop invariants

```
mul  $t2, $t1, 2    # tmp = b*2  
loop: addi $t0, $t0, 3 # (loop) a = a + 3  
      blt  $t0, $t2, loop # if (a >= tmp) goto loop
```

Laço WHILE



Duas estratégias:

- Traduza a condição a ser testada, desvio ou pula para o final;

Sketch of translation to assembly

```
loopLabel:  
  (translation of condition, ending in branch to bodyLabel)  
  j endLabel  
bodyLabel:  
  (translation of loop-body)  
  j loopLabel  
endLabel:  
  (rest of program)
```

Laço WHILE



Duas estratégias:

- Traduza a condição a ser testada: teste complementar;

Sketch of translation to assembly

```
loopLabel:  
  (complement of condition, ending in branch to endLabel)  
  (translation of loop-body)  
  j loopLabel  
endLabel:  
  (rest of program)
```

Laço WHILE



```
# Pseudocode: while (a <= c + 4) { a = a + 3 }
#       b = b + a
# Registers: a: $t0, b: $t1, c: $t2
```

Strategy 1: Condition branches over jump to end

```
      addi $t3, $t2, 4      # tmp = c + 4
loop: ble $t0, $t3, body    # while (a <= tmp) goto body
      j    end              # goto end
body: addi $t0, $t0, 3      # (in loop) a = a + 3
      j    loop             # end loop, repeat
end:  add  $t1, $t1, $t0     # b = b + a
```

Strategy 2: Complement of condition branches to end

```
      addi $t3, $t2, 4      # tmp = c + 4
loop: bgt $t0, $t3, end     # if (a > tmp) goto end
      addi $t0, $t0, 3      # (in loop) a = a + 3
      j    loop             # end loop, repeat
end:  add  $t1, $t1, $t0     # b = b + a
```

13 IFSC - Departamento Acadêmico de Eletrônica

Laço FOR



Structure of a for loop

```
for (initialize; condition; update) {
    loop-body
}
```

Equivalent program using while loop

```
initialize
while (condition) {
    loop-body
    update
}
```

14 IFSC - Departamento Acadêmico de Eletrônica

Laço FOR



```
# Pseudocode:
# sum = 0
# for (i = 0; i < n; i++) {
#     sum = sum + i
# }
# Registers: n: $t0, i: $t1, sum: $t2
```

Translate to lower-level pseudocode:

```
# sum = 0
# i = 0
# while (i < n) {
#     sum = sum + i
#     i = i + 1
# }
li $t2, 0      # sum = 0
li $t1, 0      # i = 0
loop: bge $t1, $t0, end # (start loop) if i >= n goto end
      add $t2, $t2, $t1 # sum = sum + i
      addi $t1, $t1, 1  # i = i + 1
      j    loop         # (end loop)
end:  # ...
```

15 IFSC - Departamento Acadêmico de Eletrônica

BREAK e continue



Translation of break to assembly

```
j endLabel
```

Translation of continue to assembly

In while loop:

- j loopLabel

In for loop:

- Must execute **update** first ← **gotcha!** (next slide)

16 IFSC - Departamento Acadêmico de Eletrônica

BREAK e continue



Sketch of for-loop, translated to assembly

```
(translation of initialize)
loopLabel:
  (complement of condition, ending in branch to endLabel)
  (translation of loop-body)
updateLabel:      # new label added for continue
  (translation of update)
  j loopLabel
endLabel:
  (rest of program)
```

Translation of continue to assembly

```
j updateLabel
```

BREAK e continue



- E.g. if (condition) break

```
# Pseudocode:
# while (true) {
#   ...
#   if (a < b) break
#   ...
# }
# Register mappings: a = $t0, b = $t1
```

Naive: translate if-then and break separately

```
loop:  ...      # (begin loop)
      bge $t0, $t1, else  # if (a < b)
      j   end           # (then branch) break
else:  ...      # (rest of loop body)
      j   loop          # (end loop)
end:
```

BREAK e continue



Naive: translate if-then and break separately

```
loop:  ...      # (begin loop)
      bge $t0, $t1, else  # if (a < b)
      j   end           # (then branch) break
else:  ...      # (rest of loop body)
      j   loop          # (end loop)
end:
```

Better: implement if-break as one conditional branch

```
loop:  ...      # (begin loop)
      blt $t0, $t1, end  # if (a < b) break
      ...      # (rest of loop body)
      j   loop          # (end loop)
end:
```

Laços indefinidos



Structure of an indefinite loop

```
while (true) { loop-body }
```

Trivial to implement in assembly

```
loopLabel:
  (translation of loop-body)
  j loopLabel
endLabel: # needed for break
  (rest of program)
```

Break and continue

- break – jump or branch to endLabel
- continue – jump or branch to loopLabel

```
# Pseudocode:
# total = 0
# for (i = 0; i < n; i++) {
#   if (i % 5 > 2) continue
#   total += i
# }
# Registers: total = $t0, i = $t1, n = $t2
# Note: rem $t3, $t1, 5 ==> $t3 = $t1 % 5
```

```
li    $t1, 0          # (init) i = 0
loop: bge $t1, $t2, end # while (i < n)
      rem $t3, $t1, 5  # tmp = i % 5
      bgt $t3, 2, update # if (tmp > 2) continue
      add $t0, $t0, $t1 # total += i
update: addi $t1, $t1, 1 # (update) i++
      j    loop        # (end while)
end:    # ...
```