

# Linguagem Assembly

## Condicionais



# COMANDOS CONDICIONAIS

Comando	Significado	Pronúncia
beq \$t1, \$t2, label	Se \$t1 for igual a \$t2, execute a partir do rótulo label	branch if equal
bne \$t1, \$t2, label	Se \$t1 for diferente de \$t2, execute a partir do rótulo label	branch if not equal
blt \$t1, \$t2, label	Se \$t1 for menor que \$t2, execute a partir do rótulo label	branch if less than
bgt \$t1, \$t2, label	Se \$t1 for maior que \$t2, execute a partir do rótulo label	branch if greater than
ble \$t1, \$t2, label	Se \$t1 for menor ou igual a \$t2, execute a partir do rótulo label	branch if less or equal
bge \$t1, \$t2, label	Se \$t1 for maior ou igual a \$t2, execute a partir do rótulo label	branch if greater or equal

## **EXERCÍCIO**

Escreva um programa em Assembly MIPS que lê um número inteiro e imprime se ele é par ou ímpar.

```
.data
    msg:.ascii "Digite um número:"
    par:.ascii "O número digitado é par!"
    impar:.ascii "O número digitado é ímpar!"

.text
    li $v0, 4
    la $a0, msg
    syscall

    li $v0, 5
    syscall

    li $t0, 2

    div $v0, $t0

    mfhi $t1 #o registrador hi possui o resto da divisão que é movido para $t1

    beq $t1, $zero, print_par
    li $v0, 4
    la $a0, impar
    syscall

    li $v0,10
    syscall

    print_par:
        li $v0, 4
        la $a0, par
        syscall

    li $v0,10
    syscall
```

## **EXERCÍCIO 2**

Escreva um programa em Assembly MIPS que lê um número inteiro e imprime se ele é maior, menor ou igual a zero.

```
.data
    msg:.asciiz "Digite um número:"
    maior:.asciiz "O número digitado é maior que zero!"
    menor:.asciiz "O número digitado é menor que zero!"
    igual:.asciiz "O número digitado é igual a zero!"

.text
    li $v0, 4
    la $a0, msg
    syscall

    li $v0, 5
    syscall

    move $t0, $v0

    beq $t0, $zero, print_igual
    bgt $t0, $zero, print_maior
    blt $t0, $zero, print_menor

    print_igual:
        li $v0, 4
        la $a0, igual
        syscall
    li $v0,10 #finalização obrigatória, senão o programa executa as linhas abaixo
    syscall

    print_maior:
        li $v0, 4
        la $a0, maior
        syscall
    li $v0,10
    syscall

    print_menor:
        li $v0, 4
        la $a0, menor
        syscall
```