Nome: Eduardo Lucas Lemes Januário Curso: ADS - 1ºSemestre
Pesquisa Individual - Assembly

1- Descubra como funcionam as seguintes instruções de Assembly MIPS. Descreva seu funcionamento, e detalhe quais registradores e endereços de memória são usados:

LW:

Transfere os dados (word) da memória para um registrador

SW:

Transfere os dados do registrador para a memória

LI:

Transfere os valores imediatamente para o registrador

LA:

Carrega um endereço da memória no registrador

MOVE:

Copia o valor do operando fonte para o operando destino (de registrador para registrador).
Exemplo:

mov reg, r/m
mov reg, imm
mov r/m, reg
mov r/m, imm

```
destiny = source;
```

ADD:

Soma o valor do operando destino com o valor do operando fonte, armazenando o resultado no próprio operando destino.
Exemplo:

add reg, r/m
add reg, imm
add r/m, reg
add r/m, imm

```
opdestino = opdestino + opfonte;
```

ADDI:

add immediate → adicionar constante
"Immediate" significa um número constante

SUB:

Subtrai dois valores e armazena em dois valores
(Não existe SUBI, como fazer a operação equivalente?)

MUL:

Multiply (without overflow) → Multiplicar sem Overflow
O resultado é apenas em 32 bits

MULT:

Multiply → multiplicar
Superiores a 32 bits serão armazenados em um registo
Inferiores a 32 bits serão armazenados em outro regist

DIV:

Dividir
Resto armazenado em um registro
Quociente armazenado em outro registo
Exemplo:

```
div $2,$3
$hi,$low=$2/$3
```

J:

J L1
Quando executado faz com que o programa seja desviado para L1
Exemplo:

Compilando um comando if-then-else
Seja o comando abaixo:
if (i == j) f = g + h; else f = g – h;
Solução

```
bne $s3,$s4 Else # vá para Else se i != j
add $s0,$s1,$s2  # f = g + h, se i != j
j Exit           # vá para Exit
Else: sub $s0,$s1,$s2 # f = g – h, se i = j
Exit:
```

JR:

jump register → Salto (pulo) de registro
Para o interruptor, procedimento de retorno
Exemplo:

```
r jr $1
```

vai para o endereço armazenado em $1

JAL:

jump and link → Saltar (pular) e linkar
Utilizar quando se efetua uma chamada de procedimento.
Isto guarda o endereço de retorno em $ra
Exemplo:

```
jal 1000
```

$ra=PC+4; vai para o endereço 1000

BEQZ:

Quando BEQZ é executado, se o valor no registrador especificado for zero, o programa saltará para o rótulo especificado. Caso contrário, a execução continuará com a próxima instrução da sequência.

BEQ:

beq = branch if equal → ramo (caminho, desvio) é igual
beq registrador1, registrador2, L1
se o valor do registrador1 for igual ao do registrador2 o programa será desviado para o label L1

BNE:

beq = branch if not equal → ramo (caminho, desvio) não é igual
bne registrador1, registrador2, L1
se o valor do registrador1 não for igual ao do registrador2 o programa será desviado para o label L1

SLT:

set on less than → Fixado (configurado) em menos de…
Testa se é menor que.
Exemplo:

if($2<$3)$1=1
else $1=0

Se for verdadeiro, define $1 como 1. Caso contrário, define $1 para 0.

BLT:

A instrução blt compara 2 registradores, tratando-os como inteiros com sinal, e faz uma ramificação se um registrador for menor que outro.

```
blt $ 8, $ 9, etiqueta
```

traduz para

```
vale $ 1, $ 8, $ 9
bne $ 1, $ 0, rótulo
```

2- Implementem códigos em Assembly MIPS que utilizem as instruções anteriores. Ou seja, testem as instruções e pratiquem seu uso. Por exemplo, podem criar um código simples em que definem um rótulo e efetua o jump incondicional para este rótulo.

```
.data
    num1: .word 5
    num2: .word 2
    resultado: .word 0
    msgMM: .asciiz "A soma é superior a 10."
    msgMN: .asciiz "A soma é inferior a 10."

.text

.globl main

main:
    lw $t0, num1 # Carrega o valor de num1 em $t0
    lw $t1, num2 # Carrega o valor de num2 em $t1
    add $t2, $t0, $t1 # Soma $t0 e $t1 e armazena o resultado em $t2
    sw $t2, resultado

    li $t3, 10

    slt $t4, $t2, $t3
    bne $t4, $zero, resultMenor

resultMaior:
    la $a0, msgMM # Carrega o endereço da string em $a0
    li $v0, 4 # Carrega o código do serviço de impressão de string em $v0
    syscall # Chama o serviço de impressão de string
    j fim

resultMenor:
    la $a0, msgMN # Carrega o endereço da string em $a0
    li $v0, 4 # Carrega o código do serviço de impressão de string em $v0
    syscall # Chama o serviço de impressão de string
```

```
fim:
        li $v0, 10
        syscall
```

Referências

https://embarcados.com.br/lw-e-sw-com-array-no-mips/
https://www.inf.ufpr.br/wagner/ci243/GuiaMIPS.pdf
https://www.inf.ufpr.br/roberto/ci210/assembly.pdf
https://www.ic.unicamp.br/~pannain/mc542/aulas/ch3_arq.pdf
https://mentebinaria.gitbook.io/assembly/a-base/instrucoes
https://www.dsi.unive.it/~gasparetto/materials/MIPS_Instruction_Set.pdf