- Davi Falcao - 12121BSI233
- Guilherme Castilho Machado - 12021BSI225
- Henrique Costa Fernandes de Sousa - 11821BSI263
- Lucas Gabriel Dutra de Souza - 12121BSI226
- Thiago Flauzino Oliveira Dutra - 12011BSI287

1) Considere que as variáveis a, b, c, d, e, f são associadas aos registradores $s0, $s1, $s2, $s3, $s4, $s5.

Escreva os seguintes códigos em assembly do MIPS.

> a=$s0 b=$s1 c=$s2 d=$s3 e=$s4 f=$s5

a) `f = (a – b) + (c + d)`

```
$s5 = ($s0 - $s1) + ($s2 + $s3)
add $t0,$s2,$s3 # soma $s2 com $s3 e guarda em $t0
sub $t1,$s0,$s1 # subtrai $s1 de $s0 e guarda em $t1
add $s5,$t0,$t1 # soma $t0 com $t1 e guarda em $s5
```

b) `f = (a + b) – (c + d - e)`

```
# $s5 = ($s0 + $s1) – ($s2 + $s3 - $s4)
add $t0,$s0,$s1 # soma $s0 com $s1 e guarda em $t0
add $t1,$s2,$s3 # soma $s2 com $s3 e guarda em $t1
sub $t2,$t1,$s4 # subtrai $s4 de $t1 e guarda em $t2
sub $s5,$t0,$t2 # subtrai $t2 de $t0 e guarda em $s5
```

2) Considere que as variáveis a, b, c, d, e, f não estejam associadas a nenhum dos registradores. Utilizando as operações load e store para realizar a movimentação de dados. Considere também que as variáveis são do tipo inteiro e estão armazenadas sequencialmente na memória a partir do endereço 32.

Escreva os seguintes códigos em assembly do MIPS.

a) `f = (a - b) + (c + d)`

```
lw $s0,32($sp) # guarda no registrador $s0 o valor registrado em memória no
endereço 32, correspondente a letra a
lw $s1,36($sp) # guarda no registrador $s1 o valor registrado em memória no
endereço 36, correspondente a letra b
lw $s2,40($sp) # guarda no registrador $s2 o valor registrado em memória no
endereço 40, correspondente a letra c
lw $s3,44($sp) # guarda no registrador $s3 o valor registrado em memória no
endereço 44, correspondente a letra d
lw $s4,48($sp) # guarda no registrador $s4 o valor registrado em memória no
endereço 48, correspondente a letra e
lw $s5,52($sp) # guarda no registrador $s5 o valor registrado em memória no
endereço 52, correspondente a letra f


add $t0,$s2,$s3 # soma $s2 com $s3 e guarda em $t0
sub $t1,$s0,$s1 # subtrai $s1 de $s0 e guarda em $t1
add $s5,$t0,$t1 # soma $t0 com $t1 e guarda em $s5

sw $s5,52($sp) # guarda na memória resultado da operacao
```

b) `f = (a + b) - (c + d - e)`

```
lw $s0,32($sp) # guarda no registrador $s0 o valor registrado em memória no
endereço 32, correspondente a letra a
lw $s1,36($sp) # guarda no registrador $s1 o valor registrado em memória no
endereço 36, correspondente a letra b
lw $s2,40($sp) # guarda no registrador $s2 o valor registrado em memória no
endereço 40, correspondente a letra c
lw $s3,44($sp) # guarda no registrador $s3 o valor registrado em memória no
endereço 44, correspondente a letra d
lw $s4,48($sp) # guarda no registrador $s4 o valor registrado em memória no
endereço 48, correspondente a letra e
lw $s5,52($sp) # guarda no registrador $s5 o valor registrado em memória no
endereço 52, correspondente a letra f

# $s5 = ($s0 + $s1) - ($s2 + $s3 - $s4)

add $t0,$s0,$s1 # soma $s0 com $s1 e guarda em $t0
add $t1,$s2,$s3 # soma $s2 com $s3 e guarda em $t1
```

```
sub $t2,$t1,$s4 # subtrai $s4 de $t1 e guarda em $t2
sub $s5,$t0,$t2 # subtrai $t2 de $t0 e guarda em $s5

sw $s5,52($sp) # guarda na memória resultado da operacao
```

3) Considere dois vetores A e B de inteiros, com endereço de base 32 e 128 respectivamente. A variável f está na posição 256.

Escreva os seguintes códigos em assembly do MIPS.

a) `A[16] = B[4] - f`

```
la $s0,32($sp) #guarda no registrador endereço de A
la $s1,128($sp) #guarda no registrador endereço de B
la $s2,256($sp) #guarda no registrador endereço de f

lw $t0, 64($s0) #guarda no registrador valor de A[16]
lw $t1, 16($s1) #guarda no registrador valor de B[4]
lw $t2, 0($s2) #guarda no registrador valor de f

sub $t0,$t1,$t2 #guarda em $t0 o resultado de B[4] - f

sw $t0, 64($s0)# armazena resultado em A[16]
```

b) `B[8] = A[14] + B[12] + h`

```
la $s0,32($sp) #guarda no registrador endereço de A
la $s1,128($sp) #guarda no registrador endereço de B
la $s2,256($sp) #guarda no registrador endereço de f

lw $t0, 32($s0) #guarda no registrador valor de B[8]
lw $t1, 56($s1) #guarda no registrador valor de A[14]
lw $t2, 48($s1) #guarda no registrador valor de B[12]
lw $t3, 0($s2) #guarda no registrador valor de f, como não foi informado endereço
de h, foi utilizado f no lugar

add $t4,$t1,$t2  #guarda em $t0 o resultado de A[14] + B[12]
add $t0,$t3,$t4 #guarda em $t0 o resultado de $t4+ f

sw $t0, 32($s0)# armazena resultado em B[8]
```

4) Considere que vetor V esteja associado ao registrador base $s6, e as variáveis a, b, c, d estejam associadas aos registradores $s0, $s1, $s2, $s3.

Converta a instrução em linguagem C para MIPS.

> V=$s6 a=$s0 b=$s1 c=$s2 d=$s3

a)

```
if( a == b)
    a = b + c;
else
    a = b - c;
```

```
# a)
# if( $s0 == $s1)
# $s0 = $s1 + $s2;
# else
# $s0 = $s1 - $s2;


bne $s0,$s1, Else
add $s0,$s1,$s2
j Exit
Else: sub $s0,$s1,$s2
Exit:
```

b)

```
if( a != b)
    a = b + c;
else
    a = b - c;
```

```
beq $s0,$s1, Else
add $s0,$s1,$s2
j Exit
Else: sub $s0,$s1,$s2
Exit:
```

c)

```
if( a < b)
    a = b + c;
else
    a = b - c;
```

```
bge $s0,$s1, Else #é necessário usar o comando 'bge' de maior ou igual, 'bgt' não considera casos de igualdade
add $s0,$s1,$s2
j Exit
Else: sub $s0,$s1,$s2
Exit:
```

d)

```
if( V[8] <= 12) {
    a = a + b + c;
    V[8] = V[8] + a;
} else {
    a = a - b - c;
    V[8] = V[8] + a;
}
```

```
lw $t1,32($s6) # V[8]
li $t2, 12 # armazena constante 12 em registrador
bgt $t1,$t2 Else #compara se $t1 é maior 12
add $s0,$s0,$s1
add $s0,$s0,$s2
move $t1,$s0
sw $t1,32($s6) #retorna v[8] para $s6
j Exit
Else: sub $s0,$s0,$s1
sub $s0,$s0,$s2
move $t1,$s0
sw $t1,32($s6) #retorna v[8] para $s6
Exit:
```

e)

```
if( V[8] > b) {
    a = a + b + c;
    V[8] = V[8] + a;
} else {
    a = a - b - c;
```

```
    V[8] = V[8] + a;
}
```

```
lw $t1,32($s6) # V[8]
li $t2 12 # armazena constante 12 em registrador
ble $t1,$t2 Else #compara se $t1 é menor ou igual 12
add $s0,$s0,$s1
add $s0,$s0,$s2
move $t1,$s0
sw $t1,32($s6) #retorna v[8] para $s6
j Exit
Else: sub $s0,$s0,$s1
sub $s0,$s0,$s2
move $t1,$s0
sw $t1,32($s6) #retorna v[8] para $s6
Exit:
```

f)

```
if( V[8] >= 12) {
    a = a + b + c;
    V[8] = V[8] + a;
} else {
    a = a - b - c;
    V[8] = V[8] + a;
}
```

```
lw $t1,32($s6) # V[8]
li $t2 12 # armazena constante 12 em registrador
blt $t1,$t2 Else #compara se $t1 é menor que 12
add $s0,$s0,$s1
add $s0,$s0,$s2
move $t1,$s0
sw $t1,32($s6) #retorna v[8] para $s6
j Exit
Else: sub $s0,$s0,$s1
sub $s0,$s0,$s2
move $t1,$s0
sw $t1,32($s6) #retorna v[8] para $s6
Exit:
```

5) Considere que vetor V esteja associado ao registrador base $s6.

Converta a instrução em linguagem C para MIPS.

a)

```
while (b < 32) {
    a = a + a;
    V[4] = a + b;
    b = b + 1;
}
```

```
loop:
    slt $t0, $s1, 32      # if (b < 32)
    beq $t0, $zero, exit  # exit loop
    add $s0, $s0, $s0     # a = a + a
    add $t1, $s0, $s1     # t1 = a + b
    add $t2, $s6, 16      # t2 = &V[4]
    sw $t1, ($t2)         # V[4] = a + b
    addi $s1, $s1, 1      # b = b + 1
    j loop                # repete o loop
exit:
```

b)

```
while( V[0] <= a )
    V[0] = V[0] + b;
```

```
loop:
    lw $t0, ($s6)         # t0 = V[0]
    slt $t1, $t0, $s0     # if (V[0] <= a)
    beq $t1, $zero, exit  # exit loop
    add $t2, $t0, $s1     # t2 = V[0] + b
    sw $t2, ($s6)         # V[0] = V[0] + b
    j loop                # repete o loop
exit:
```

6) Qual o código em assembly do MIPS para cada uma das seguintes funções C?

a)

```c
int busca(int x, int n, int v[]) {
    int k;
    k = n-1;
    while(k>=0 && v[k] != x)
        k -= 1;
    return k;
}
```

```assembly
busca_r:
    addi $sp, $sp, -12      # ajusta o ponteiro de pilha
    sw $ra, 8($sp)          # salva o endereço de retorno
    sw $s0, 4($sp)          # salva o valor de s0

    lw $s0, 8($a1)          # $s0 = n
    beq $s0, $zero, retorno # se n == 0, retorna -1

    addi $s0, $s0, -1       # $s0 = n - 1
    lw $t0, ($a2)           # $t0 = v[n-1]
    beq $a0, $t0, encontrado # se x == v[n-1], retorna n-1

    addi $a1, $a1, -1        # n = n - 1
    addi $a2, $a2, 4        # v = v + 1
    j busca_r                # chama a função recursivamente

encontrado:
    addi $s0, $s0, -1       # $s0 = n - 1
    j retorno                # retorna n-1

retorno:
    lw $s0, 4($sp)          # retorna o valor de s0
    lw $ra, 8($sp)          # retorna o endereço de retorno
    addi $sp, $sp, 12       # ajusta o ponteiro da pilha
    jr $ra                  # retorna o chamador
```

b)

```c
int busca_r(int x, int n, int v[]) {
    if(n == 0)
        return -1;
    if(x == v[n-1])
        return n-1;
```

```
    return busca_r(x, n-1, v);
}
```

```
busca_r:
    addi $sp, $sp, -12      # ajusta o ponteiro de pilha
    sw $ra, 8($sp)          # salva o endereço de retorno
    sw $s0, 4($sp)          # salva o valor de s0

    lw $s0, 8($a1)          # $s0 = n
    beq $s0, $zero, retorno # se n == 0, retorna -1

    addi $s0, $s0, -1       # $s0 = n - 1
    lw $t0, ($a2)           # $t0 = v[n-1]
    beq $a0, $t0, encontrado # se x == v[n-1], retorna n-1

    addi $a1, $a1, -1        # n = n - 1
    addi $a2, $a2, 4        # v = v + 1
    j busca_r                # chama a função recursivamente

encontrado:
    addi $s0, $s0, -1       # $s0 = n - 1
    j retorno                # retorna n-1

retorno:
    lw $s0, 4($sp)          # retorna o valor de s0
    lw $ra, 8($sp)          # retorna o endereço de retorno
    addi $sp, $sp, 12       # ajusta o ponteiro da pilha
    jr $ra                  # retorna o chamador
```

7) Complete a tabela para as seguintes instruções MIPS. Quando um campo não existir no formato de instrução utilize na (não se aplica). Para coluna categoria use A – Aritmética, T – Transferência de dados, L – Lógica, DC – Desvio condicional e D – Desvio.

Na sequencia explique como funciona cada uma das instruções:

Exemplo:

```
add $s1, $s2, $s3 #Soma o conteúdo dos registradores $s2 e $s3 e armazena no registrador $s1.
```

Instrução	Categoria	Formato	op	rs	rt	rd	shamt	funct	endereço
`add $s1, $s2, $s3`	A	R	0	$s2	$s3	$s1	0	32	n.a.
`sub $s1, $s2, $s3`	A	R	0	$s2	$s3	$s1	0	34	n.a.
`addi $s1, $s2, 10`	A	I	8	$s2	$s1	n.a	n.a	n.a	10
`lw $s1, 100($s2)`	T	I	35	$s2	$s1	n.a	n.a	n.a	100
`sw $s1, 100($s2)`	T	I	43	$s2	$s1	n.a	n.a	n.a	100
`and $s1, $s2, $s3`	L	R	0	$s2	$s3	$s1	0	36	n.a.
`or $s1, $s2, $s3`	L	R	0	$s2	$s3	$s1	0	37	n.a.
`nor $s1, $s2, $s3`	L	R	0	$s2	$s3	$s1	0	39	n.a.
`andi $s1, $s2, 10`	L	I	12	$s2	$s1	n.a	n.a	n.a	10
`ori $s1, $s2, 10`	L	I	13	$s2	$s1	n.a	n.a	n.a	10
`sll $s1, $s2, 10`	L	R	0	0	$s2	$s1	10	0	n.a.
`srl $s1, $s2, 10`	L	R	0	0	$s2	$s1	10	2	n.a.
`beq $s1, $s2, L`	DC	I	4	$s1	$s1	n.a	n.a	n.a	L
`bnq $s1, $s2, L`	DC	I	5	$s1	$s2	n.a	n.a	n.a	L
`slt $s1, $s2, $s3`	L	R	0	$s2	$s3	$s1	0	42	n.a.
`slti $s1, $s2, 10`	L	I	10	$s2	$s1	n.a	n.a	n.a	10
`j L`	D	J	2	n.a	n.a	n.a	n.a	n.a	L
`jr $ra`	D	R	0	$ra	0	0	0	8	n.a.
`jal L`	D	J	3	n.a	n.a	n.a	n.a	n.a	L

```
add $s1, $s2, $s3 #Soma o conteúdo dos registradores $s2 e $s3 e armazena no
registrador $s1.
sub $s1, $s2, $s3 # Subtrai o conteúdo do registrador $s3 do registrador $s2 e
armazena o resultado no registrador $s1.
addi $s1, $s2, 10 # Adiciona o valor imediato 10 ao conteúdo do registrador $s2 e
armazena em $s1.
lw $s1, 100($s2)  #Carrega o valor armazenado na memória no endereço representado
pela soma do conteúdo do registrador $s2 com o valor imediato 100 e armazena o
resultado no registrador $s1.
sw $s1, 100($s2) #Armazena o conteúdo do registrador $s1 na memória no endereço
representado pela soma do conteúdo do registrador $s2 com o valor imediato 100.
and $s1, $s2, $s3  #Realiza uma operação lógica AND entre conteúdos dos
registradores $s2 e $s3 e armazena o resultado no registrador $s1.
or $s1, $s2, $s3 #: Realiza uma operação lógica OR entre os conteúdos dos
registradores $s2 e $s3 e armazena o resultado no registrador $s1.
nor $s1, $s2, $s3 # Realiza uma operação lógica NOR entre os conteúdos dos
registradores $s2 e $s3 e armazena o resultado no registrador $s1.
andi $s1, $s2, 10 # Realiza uma operação lógica AND entre o conteúdo do
registrador $s2 e o valor imediato 10 e armazena o resultado no registrador $s1.
ori $s1, $s2, 10 # Realiza uma operação lógica OR entre o conteúdo do registrador
$s2 e o valor imediato 10 e armazena o resultado no registrador $s1.
sll $s1, $s2, 10 # Desloca o conteúdo do registrador $s2 em 10 bits para a
esquerda e armazena o resultado no registrador $s1.
srl $s1, $s2, 10 # Desloca o conteúdo do registrador $s2 em 10 bits para a direita
e armazena o resultado no registrador $s1.
beq $s1, $s2, L # Desvia para o endereço indicado por L se o conteúdo dos
registradores $s1 e $s2 forem iguais.
bnq $s1, $s2, L # Desvia para o endereço indicado por L se o conteúdo dos
registradores $s1 e $s2 forem diferentes.
slt $s1, $s2, $s3 # Compara o conteúdo dos registradores $s2 e $s3, e se $s2 for
menor que $s3, armazena 1 no registrador
slti $s1, $s2, 10 # Compara o conteúdo do registrador $s2 com o valor imediato 10
e, se $s2 for menor que 10, armazena 1 no registrador $s1. Caso contrário,
armazena 0.
```