Lista 2 de Arquitetura de Computadores II

Gustavo Lopes Rodrigues

12 de Junho de 2020

1 Parte 1

1.1 Programa1

```
1  .data
2    a: .word 10 #armazenando o valor de "a". a = 10
3    b: .word -1 #armazenando o valor de "b". b = -1
4    c: .word 0 #armazenando o valor de "c". c = 0
5  .text
6    lw $t1, a #carregar o valor de "a" para "t1"
7    lw $t2, b #carregar o valor de "b" para "t2"
8
9    add $t1,$t1,1 #t1 += 1
10   add $t3,$t1,$t2 # t3 = t1 + t2
11
12   sw $t3,c # salvando o valor de "t3" em "c"
```

1.2 Programa2

```
1 .data
2   x: .word 3 #armazenando o valor de "x". x = 3
3   y: .word 0 #armazenando o valor de "y". y = 0
4
5 .text
6   lw $t1, x #carregar o valor de "x" para "t1"
7
8   sll $t1,$t1,2 #t1 *= 2^2
9   sw $t1, y #armazenando o valor de "t1" em "y"
```

1.3 Programa3

```
1 .data
2   x: .word 3 #armazenando o valor de "x". x = 3
3   y: .word 0 #armazenando o valor de "y". y = 0
4
5 .text
6   lw $t1, x #carregar o valor de "x" para "t1"
7
8   mul $t1,$t1,1025 #t1 *= 1025
9   sw $t1, y #armazenando o valor de "t1" em "y"
```

1.4 Programa4

```
1  .data
2    x: .word 3 #armazenando o valor de "x" no formato . x = 3
3    y: .word 0 #armazenando o valor de "y" no formato . y = 0
4
5  .text
6
7    lw $t0,x       # carregar o valor de "x" para "t0"
8    add $t1,$t1,4  # t1 = 4
9
10   div $t0,$t1    # 3 / 4 | Mandar o quociente e o restante para LO
       e HI
11
12   mflo $t0   # pegar o restante e armazenar em t0
13
14   sw $t0,y   # Armazenar t0 em y
```

1.5 Programa5

```
1 .data
2   aux: .word 157109 # aramzenando o valor 157109 em "aux" | Estou
      fazendo isso pois este   um n mero primo
3   x: .word 0 #armazenando o valor de "x".
4
5 # 305419896 = 1944 x 157109
6 .text
7   lw $t0, aux #carregar o valor de "x" para "t0"
8   add $s0,$zero,1944 # s0 = 0 + 1944
9   mul $s0,$s0,$t0 # s0 *= t0 | 1944 x 157109
10  sw $s0,x # salvando o valor de "t0" em x
```

1.6 Programa6

```
.data
  x: .word -1 #armazenando o valor de "x" no formato . x = -1
  y: .word 0 #armazenando o valor de "y" no formato . y = 0

.text
  lw $t0, x #carregar o valor de "x" para "f2"
  add $t1,$zero,32 # Armazenar 32 em t1

  div $t0,$t1  # Dividir t0 por t1

  mflo $t0   # pegar o restante e armazenar em t0

  sw $t0,y   # armazenar o valor de "t0" em "y"
```

1.7 Programa7

```
1  .data
2    A: .space 48 # Declarando um array com 48 bits de espao (ou seja
       , cabe 12 nmeros inteiros)
3    h: .word 0 # Declarando um inteiro chamado "h", com o valor
       inicial igual a 0
4  .text
5    lw $s0,h # Carregando o valor de "h" em "s0"
6
7    #Pegar a posicao "8" do array
8    add $t0,$zero,8 # t0 = 0 + 8
9    sll $t0,$t0,2 # t0 *= 2^2
10
11   lw $t1,A($t0) # Carregando o conteudo do array em "t0" para a
       variavel "t1"
12
13   add $s1,$s0,$t1 # s1 = s0 + t1 |  h + A [8];
14
15   #Pegar a posicao "12" de um array
16   add $t0,$zero,12 # t0 = 0 + 12
17   sll $t0,$t0,2 # t0 *= 2^2
18
19   sw $s1,A($t0) # A[12] = h + A[8]
```

1.8 Programa8

```
1  .data
2
3    A:   .space 12 # Criando um array "A" com espao de 12 bits ( ou
       seja, o array tem um espao de 3 inteiros)
4    h:   .word  0 # Criando um inteiro "h"
5    k:   .word  0 # Criando um inteiro "k"
6    i:   .word  0 # Criando um inteiro "i"
7
8  .text
9
10   lw $t0,h # Carregar o conteudo em "h" para "t0"
11   lw $s0,k # Carregar o conteudo em "k" para "s0"
12   lw $s1,i # Carregar o conteudo em "i" para "i"
13
14   lw $s2,A($s1) # Carregar o conteudo de "A($s1)" |   A [ i ]
15
16   add $t0,$s0,$s2 # t0 = s0 + s2 | h = k + A [ i ]
```

1.9 Programa9

```
1 .data
2   A:   .space 12 # Declarando um array com 12 bits de espao (ou
      seja, cabe 3 nmeros inteiros)
3   h:   .word  0 # Declarando um inteiro "h"
4   j:   .word  0 # Declarando um inteiro "j"
5   i:   .word  0 # Declarando um inteiro "i"
6 .text
7
8   lw $t0,h # Carregando o contedo em "h" para "t0"
9   lw $s0,j # Carregando o contedo em "j" para "s0"
10  lw $s1,i # Carregando o contedo em "i" para "s1"
11
12  lw $s2,A($s1) # Carregando o conteudo em "A($s1)" para a variavel
       "s2"
13  add $t0,$t0,$s2 # t0 = t0 + s2 | h + A [ i ]
14
15  sw $t0,A($s0) # Salvando o conteudo de "t0" no endereo "A($s0)"
      |  A [ j ] = h + A [ i ]
```

1.10 Programa10

```
.data
  A:   .space 12 # Declarando um array com 12 bits de espao (ou
    seja, cabe 3 nmeros inteiros)
  h:   .word  0 # Declarando um inteiro "h"
  i:   .word  0 # Declarando um inteiro "i"
.text

  # Carregando o conteudo de "h" e "i"
  lw $t0,h # Carregando o contedo em "h" para "t0"
  lw $s0,i # Carregando o contedo em "i" para "s1"

  # h = A[i]
  lw $t0,A($s0) # h = A($s0) | h = A[i]

  #A[i] = A[i+1]
  add $t1,$s0,1 # t1 = s0 + 1 | t1 = i + 1
  lw $s1,A($t1) # s1 = A[t1] | s1 = A[i+1]
  sw $s1,A($s0) # A[s0] = s1 | A[i] = A[i+1]

  #A[i+1] = h
  sw $t0,A($t1) # A [i+1] = h
```

1.11 Programa11

```
.data
  j: .word 0 #armazenando o valor de "j". j = 0
  i: .word 10 #armazenando o valor de "i". i = 10

.text
  lw $t0, j #carregar o valor de "j" para "t0"
  lw $t1, i #carregar o valor de "i" para "t1"
  while: #comeo do loop while
    beq $t0,$t1, exit # fazer o loop at  "t0" no for igual a "t1"
    add $t0,$t0,1 # t0 += 1 | j = j + 1
    j while # ir para "while"
  exit: # saida
```

2 Parte 2

2.1 Programa1

```
1  .text
2
3  addi  $s0,$zero,2 # s0 = a = 2
4  addi  $s1,$zero,3 # s1 = b = 3
5  addi  $s2,$zero,4 # s2 = c = 4
6  addi  $s3,$zero,5 # s3 = d = 5
7
8  add $t0,$s0,$s1 # t0 = s0 + s1 | a+b
9  add $t1,$s2,$s3 # t1 = s2 + s3 | c+d
10 sub $s4,$t0,$t1 # s4 = t0 + t1 | x
11
12 sub $t0,$s0,$s1 # t0 = s0 + s1 | a-b
13 add $s5,$t0,$s4 # s5 = t0 + s4 | y
14
15 sub $s1,$s4,$s5 # s1 = s4 + s5 | b
```

2.2 Programa2

```
.text

addi  $s0,$zero,1 # Declarando x | x = 1

#Somando "x" cinco vezes, para fazer o mesmo "que 5 * x"
add $t0,$t0,$s0
add $t0,$t0,$s0
add $t0,$t0,$s0
add $t0,$t0,$s0
add $t0,$t0,$s0

add $s1,$t0,15  # s1 = y = 5 * x + 15
```

2.3 Programa3

```
1  .text
2
3    addi $s0,$zero,3 # s0 = x, x = 3
4    addi $s1,$zero,4 # s1 = y, y = 4
5
6    # x * 15
7    add $t0,$zero,15
8    add $t0,$t0,15
9    add $t0,$t0,15
10
11   # y * 67
12   add $t1,$zero,67
13   add $t1,$t1,$t1
14   add $t1,$t1,$t1
15
16   # (15 * x + 67 * y ) * 4
17   add $t2,$t1,$t0 # t2 = t1 + t0 | ( 15 * 3 + 67  )
18   add $t2,$t2,$t2
19   add $t2,$t2,$t2
```

2.4 Programa4

```
1  .text
2
3    addi $s0,$zero,3 # s0 = x, x = 3
4    addi $s1,$zero,4 # s1 = y, y = 4
5
6    mul $t0,$s0,15 # t0 = tmp1 , tmp1 = 15 * 3
7
8    mul $t1,$s1,67 # t1 = tmp2 , tmp2 = 67 * 4
9
10   add $t2,$t1,$t0 # t2 = t1 + t0 | ( 15 * 3 + 67  )
11   sll $t2,$t2,2 # t2 *= 2^2 | ( 15 * 3 + 67  ) * 4
```

2.5 Programa5

```
.text

addi $t0,$zero,25000  # t0 = 0 + 25000

sll $s0,$t0,2           # s0 = t0 * 2^2, x = s0, x = 100000
sll $s1,$t0,3           # s1 = t0 * 2^3, y = s1, y = 200000

add $s2,$s0,$s1         # s2 = s0 + s1 | z = x + y
```

2.6 Programa6

```
1  .text
2
3 addi $t0,$zero,-1 # t0 = 0 +(-1)
4 srl $s0,$t0,1 # s0 = t0 / 2 | s0 = x, x = maior inteiro possivel
5 add $t1,$zero,  9375 # s1 = 0 + 9375
6 sll $s1,$t1,5   # s1 = t1 * 2^5 | s1 = y , y = 300000;
7
8 add $t2,$zero,$s1 # t2 = 0 + s1
9 sll $t2,$t2,2   # t2 *= 2^2 | 300000 * 4;
10
11 sub $s2,$s0,$t2  # s2 = s0 + t2 | z = x - 4y
```

2.7 Programa7

```
1 .text
2
3 ori $8, $0, 0x01 # Comando necessrio para a questo
4 sll $8, $8, 31 # $8 *= 2^31
5 sra $8, $8, 31 # $8 /= 2^31 | $8 = 0xFFFFFFFF
```

2.8 Programa8

```
.text

#$8 = 0x12345678.
addi $8,$zero,0x1234
sll $8,$8,16
add $8,$8,0x5678

#$9 = 0x12
ori $9, $0, 0x1234
srl $9, $9, 8

#$10 = 0x34
addi $7,$zero,0x1234
andi $10, $7, 0x34

#$11 = 0x56
andi $11, $8, 0x5678
srl $11, $11, 8

#$12 = 0x78
andi $12,$8,0x78
```

2.9 Programa9

```
.data
x1: .word 15 # configurando o valor do inteiro "x1" como 15
x2: .word 25  # configurando o valor do inteiro "x2" como 25
x3: .word 13  # configurando o valor do inteiro "x3" como 13
x4: .word 17  # configurando o valor do inteiro "x1" como 17
soma: .word -1  # configurando o valor do inteiro "soma" como -1
.text

# Referencia : lw = load word ; sw = save word.

# Armazendo os valores em ".data" em variaveis temporarias

ori $t0,$zero,0x1000
sll $t0,$t0,16

lw $s0, x1 # t0 = x1 = 15
lw $s1, x2 # t1 = x2 = 25
lw $s2 ,x3 # t2 = x3 = 13
lw $s3 ,x4 # t3 = x4 = 17

#Somando os elementos
add $t1 ,$s0,$s1 # t4/temp4 = t0 + t1 ou 15 + 25
add $t2 ,$s2,$s3 # t5/temp5 = t2 + t3 ou 13 + 17
add $s4 ,$t1,$t2 # t6/temp6 = t5 + t4 ou 40 + 30

#salvando o valor das somas em "soma"
sw $s4,16($t0)
```

2.10 Programa10

```
1 .data
2 x: .word 5    # Declarando "X"
3 z: .word 7    # Declarando "Z"
4 y: .word 0    # Declarando "Y" ,o valor devera ser sobrescrito apos
      a execucao do programa.
5 .text
6
7 # Armazendo os valores do ".data" em vari veis tempor rias
8 lw $t0, x    # t0/temp0 = x = 5
9 lw $t1, z    # t1/temp1 = x = 7
10
11 # Armazenando os valores que seram usados para a multiplicacao
12 addi $t4,$t4,127
13 addi $t5,$t5,65
14
15 #127x5
16 sll $t2,$t4,2     # $t2 = 127 x 4
17 add $t2,$t2,$t4   # $t2 += 127
18
19 #65x7
20 sll $t3,$t5,3     # $t3 = 65 x 8
21 sub $t3,$t3,$t5   # $t3 -= 65
22
23 #Conta final
24 sub $t3,$t2,$t3   # 127x5 - 65x7 | $t2 - $t3
25 add $t3,$t3,1     # $t3 += 1
26
27 #salvando o valor em "y"
28 sw $t3, y
```

2.11 Programa11

```
.data
x: .word 100000
z: .word 200000
y: .word 0
.text
# Referencia : lw = load word ; sw = save word.

# Armazendo os valores do ".data" em vari veis tempor rias
lw $t0, x    # t0/temp0 = x = 100000
lw $t1, z    # t1/temp1 = z = 200000

# Pegando o valor 300000
addi $t2,$zero,18750
sll $t2,$t2,4     # 18750 * (2^4) = 300000

#x - z + 300000
sub $t1,$t0,$t1
add $t2,$t2,$t1

#salvando o valor em "y"
sw $t2, y
```

2.12 Programa12

```
.data
    k: .word 5
    x: .word

.text
    main:
    la $t0, k # pointeiro 1
    la $t1, 4($t0) # pointeiro 2
    la $t2, 4($t1) # pointeiro , t2 = x

    sw $t2, x # armazenando em x o ponteiro

    la $s0, -8($t2) # pegando o ponteiro original
    lw $s1,($s0) # pegar o conteudo do ponteiro

    sll $s1,$s1,1 # multiplicar por 2 ( 1 shift logical left) | K
    *= 2
    sw $s1,($s0) # salvar o resultado da multiplicacao no conteudo
    do ponteiro
```

2.13 Programa13

```
.data
A: .word 0
.text

#Para o teste $t1 mostra se o numero eh positivo ou nao
# Numero positivo: t1 = 1     Numero negativo: t1 = 2

main:
ori $t0,$zero,0x1001  # t0 = 1001
sll $t0,$t0,16    # t0 = 10010000
add $t1, $zero, $zero # t1 = 0
lw $s0,0($t0)   # s0 = Valor da memoria de A

#Funcao modulo para numero positivo
modulo:
slt $t3, $s0, $t1 # t3 = if ( s0 < 0 )
bne $t3,$zero, negativo # ir para a funcao "negativo" se "t3" nao
      for igual a zero
addi $t1, $t1, 1 # t1 = 1
sw $s0,0($t0) # Mem [ t0 ] = s0
j exit
#Funcao modulo para numero negativo
negativo:
sub $s1, $s0, $s0 # s1 = - A - ( - A ) = 0
sub $s0, $s1, $s0 # s0 = 0 - ( - A ) = A
addi $t1, $t1, 2 # t1 = 2
sw $s0,0($t0) # Mem [ t0 ] = s0

exit:
```

2.14 Programa14

```
.data
TEMP: .word 51
FLAG: .word -2
.text
#Pegando o endereco base da data

main:
ori $t0,$zero,0x1001  # t0 = 1001 # Observacao: t0 eh o vetor
sll $t0,$t0,16    # t0 = t0 *= 2^16 = 10010000
addi $t1,$zero, 30  # t1 = 0 + 30
addi $t2,$zero, 50  # t2 = 0 + 50
lw $s0,0($t0)   # Resgatar o valor em "0(t0)"

#Primeiro Teste do AND
test1:
slt $t3, $s0, $t1 # se "s0" for menor do que "t1", t3 sera igual a
    1 , caso contrario, sera igual a 0
beq $t3,$zero, test2  # va para a funcao "test2" se t3 for igual a
    zero
sw $zero,4($t0)   # salve 0 na posicao "4" em "t0"
j fim

#Segundo Teste do AND
test2:
slt $t3, $t2, $s0 # se "t2" for menor do que "s0", t3 sera igual a
    1 , caso contrario, sera igual a 0
beq $t3,$zero, verdade  #  va para a funcao "verdade" se "t3" for
    igual a zero
sw $zero,4($t0)         # salve 0 na posicao "4" em "t0"
j fim
## if resp == true

verdade:
addi $s0, $zero, 1  # s0 = 1
sw $s0, 4($t0)    # salve s0 na posicao "4" em "t0"

fim:
```

2.15 Programa15

```
1  .text
2
3    ori $t0,$zero,0x1001  # t0 = 1001 | t0 eh o vetor
4    sll $t0,$t0,16    # t0 = 10010000
5
6    add $s0, $zero, $zero # Configurando s0 ; s0 = soma
7    add $t1,$s0, $zero  # t1 = s0 + 0 | primeira pos = 0
8    addi $t2,$zero,100  # t2 = 0 + 100 | ultima pos = 100
9
10 vetor:
11   sll $t3, $t1, 1   # t3 = t1 * 2^1
12   addi $t3, $t3, 1  # t3 += 1
13   sw $t3,0($t0)   # salvar "t3" no endereco 0 em t0
14   add $s0, $s0, $t3 # s0 += t3      | Acumulador da soma de todos
       os elementos do vetor
15
16 conserta:
17   addi $t0, $t0, 4  # t0 += 4
18   add $t1, $t1, 1   # t1 += 1
19   bne $t1, $t2, vetor   # ir para "vetor" se "t1" for diferente de
       "t2"
```

2.16 Programa16

```
.data
    array: .space 400 #Declarando o array de 400 bits( com tamanho
    100 )

.text
    main:
    addi $t0, $zero, 0 # t0 = i ; i = 0
    addi $t1, $zero, 1300

    while: # while ( i < 400 ) {
        bgt $t0,400, exit
        sub $t1,$t1,13
        sw $t1,array($t0) # array[i] = t1
        addi $t0,$t0,4
        j while

    exit: # }
    add $t0,$zero, 0 # int i
    add $t1, $zero, -4

    blt $t0, 400, bubblesort1 # for (int i = 0; i < n; i++)

    bubblesort1:
    add $t1,$t1,4 # j++
    blt $t1, 396, bubblesort2 # for (int j = 0; j < n-1; j++)

    bubblesort2:
    add $t3,$t1,4 # t3 = j + 1
    lw $s1,array($t1) # resgatando o conteudo de "array(j)"
    lw $s2,array($t3) # resgatando o conteudo de "array(j+1)"
    bgt $s1,$s2,swap # if (arr[j] > arr[j+1])

    bubblesort3:
    blt $t1, 396, bubblesort1 # conferindo se t1 ja percorreu o
    array in
    add $t0,$t0,4 # i++
    add $t1,$zero,-4 # resetando j
```

```
36      blt $t0,400, bubblesort1 # conferindo se t0 ja percorreu o
        array inteiro
37      j fim
38
39      swap:
40      #trocando o conteudo em array[j] e array[j+1]
41      sw $s2,array($t1)
42      sw $s1,array($t3)
43      j bubblesort3
44
45      fim:
46      #fim do programa
```

2.17 Programa17

```
.data
  x: .word 3
.text

  main:
  ori $t0,$zero,0x1001  # t0 = 1001
  sll $t0,$t0,16    # t0 *= 2^16 | t0 = 10010000

  addi $t1,$zero,1  # t1 = 0 + 1 |t1 = count = 1

  addi $t2, $zero,2   # t2 = 0 + 2 | t2 = expoente1 = 2
  addi $t3, $zero,3 # t3 = 0 + 3 | t3 = expoente2 = 3
  addi $t4, $zero,4 # t4 = 0 + 4 | t4 = expoente3 = 4
  addi $t5, $zero,5 # t5 = 0 + 5 | t5 = expoente4 = 5

  lw $s0, 0($t0)    # Carregar o conteudo em 0 para s0 | s0 = x =
    10
  lw $s1, 0($t0)    # Carregar o conteudo em 0 para s1 | s1 = x =
    10
  lw $s2, 0($t0)    # Carregar o conteudo em 0 para s2 | s2 = x =
    10
  lw $s3, 0($t0)    # Carregar o conteudo em 0 para s3 | s3 = x =
    10

  div $s0,$t2             # Dividir s0 por t2
  mfhi $t6                # Mover o conteudo do HI para t6
  bne $t6, $zero, impar   # Ir para "impar" se t6 for igual a 0

  par:                    # funcao "par"
  addi $t1,$t1,1    # t1 += 1
  mult $s1,$s0    # Multiplicar s1 por s0
  mflo $s1    # Mover o conteudo de LO para s1 | s2 = k = (x  y)
  bne $t1, $t4, par # Ir para "par" se t1 for igual a t4

  addi $t1,$zero,1  # t1 += 1

  par2:
```

```
34  addi $t1,$t1,1    # t1 += 1
35  mult $s2,$s0    # Multiplicar s2 por s0
36  mflo $s2    # Mover o conteudo de LO para s2
37  bne $t1, $t3, par2  # Ir para "par2" se t1 for igual a t3
38
39  addi $t1,$zero,1  # t1 += 1
40
41  par3:
42  addi $t1,$t1,1    # t1 += 1
43  mult $s3,$s0    # Multiplicar s3 por s0 |
44  mflo $s3    # Mover o conteudo de LO para s3
45  sll $s3, $s3, 1   # s3 *= 2^1
46  add $s1,$s1,$s2   # s1 += s2
47  sub $s1,$s1,$s3   # s1 += s3
48  j fim     # ir para "fim"
49
50  impar:
51  addi $t1,$t1,1    # t1 += 1
52  mult $s1,$s0    # (s1 * s0)
53  mflo $s1
54  bne $t1, $t5, impar
55
56  addi $t1,$zero,1  # t1 = 1
57
58  impar2:
59  addi $t1,$t1,1    # t1 = t1 + 1
60  mult $s2,$s0    # Multiplicar s1 por s0
61  mflo $s2    # Mover o conteudo de LO para s2
62  bne $t1, $t3, impar2  # Ir para "impar2" se t1 for igual a t3
63  sub $s1,$s1,$s2   # s1 -= s2
64  addi $s1, $s1, 1  # s1 += 1
65
66  fim:
67  sw $s1,4($t0) #Salvando o valor de k na memoria
```

2.18 Programa18

```
.data
  x: .word 2
.text
  ori $t0,$zero,0x1001  # t0 = 1000
  sll $t0,$t0,16    # to *= 2^16 |t0 = 10000000

  addi $t1,$zero,1  # t1 += 1 | t1 = count = 1
  addi $t5,$zero,1  # t5 += 1

  addi $t2, $zero,3   # t2 = expoente1 = 3
  addi $t3, $zero,4 # t3 = expoente2 = 4

  lw $s0, 0($t0)    # carregar o conteudo em "0(t0)" em "s0"
  lw $s1, 0($t0)    # carregar o conteudo em "0(t00" em "s1"

  slt $t4, $zero, $s0 #Se "s0" for menor que zero, t4 sera igual a
    1 , se no , ser igual a 0
  bne $t4, $zero, maior #Se "t4" nao for igual a zero , va para
    maior

  menor:
  addi $t1,$t1,1    # t1 += 1
  mult $s1,$s0    # multiplique o valor de "s1" com "s0"
  mflo $s1    # pegue o valor de LO e coloque em "s1"
  bne $t1, $t3, menor   # Se "t1" for menor que "t3", va para menor
  sub $s1, $s1, $t5 # s1 -= t5
  j fim     # va para fim

  maior:
  addi $t1,$t1,1    # ti += 1
  mult $s1,$s0    # Multiplique "s1" com "s0"
  mflo $s1    # Pegue o conteudo em LO para "s1"
  bne $t1, $t2, maior # Se "t1" for menor que "t2" , va para maior
  addi $s1, $s1, 1  # s1 += 1

  fim:
  sw $s1,4($t0) #Salvando o valor de k na memoria
```

2.19 Programa19

```
1  .data
2  x: .word 1600000
3  y: .word 80000
4  z: .word 400000
5
6  .text
7
8  ori $t0,$zero,0x1001  # t0 = 1001
9  sll $t0,$t0,16    # t0 *= 2^16 | t0 = 10010000
10
11 add $t1,$zero,10
12 add $s3,$zero,1
13
14 lw $s0,0($t0)   # Carregar o conteudo em 0(t0) | s0 = Valor da
       memoria de x
15 lw $s1,4($t0)   # Carregar o conteudo em 4(t0) | s1 = Valor da
       memoria de y
16 lw $s2,8($t0)   # Carregar o conteudo em 8(t0) | s2 = Valor da
       memoria de z
17
18 #As funcoes a seguir servem para guardar a quantidade de zeros para
        multiplicar no final
19
20 #Quantidade de Zeros em S0
21 divS0:
22 div $s0,$t1 # Dividir S0 por 10
23 mfhi $t2    # Pegar o restante da divisao em HI
24 beq $t2,0,addZeroS0 # Se o restante da divisao for 0, adicione um
       zero ao contador
25 j divS1 # Se nao, va para calcular a quantidade de zeros em S1
26
27 #Quantidade de Zeros em S1
28 divS1:
29 div $s1,$t1 # Dividir S1 por 10
30 mfhi $t2    # Pegar o restante da divisao em HI
31 beq $t2,0,addZeroS1 # Se o restante da divisao for 0, adicione um
       zero ao contador
```

```
32 j divS2 # Se nao, va para calcular a quantidade de zeros em S2
33
34 #Quantidade de Zeros em S2
35 divS2:
36 div $s2,$t1 # Dividir s2 por 10
37 mfhi $t2    # Pegar o restante da divisao em HI
38 beq $t2,0,subZeroS2 # Se o restante da divisao for 0, diminua um
       zero ao contador(ja que s2 sera dividido na operacao)
39 j calcular # Se nao, va para calcular a quantidade de zeros em s2
40
41 #Funcao para adicionar 0 ao contador de S0
42 addZeroS0:
43 mflo $s0       #Pegar o quociente em LO
44 mul $s3,$s3,10 #s3 *= 10
45 j divS0
46
47 #Funcao para adicionar 0 ao contador em S1
48 addZeroS1:
49 mflo $s1       #Pegar o quociente em LO
50 mul $s3,$s3,10 #s3 *= 10
51 j divS1
52
53 #Funcao para remover 0 ao contador em S2
54 subZeroS2:
55 mflo $s2       #Pegar o quociente em LO
56 div $s3,$s3,10 #s3 *= 10
57 j divS2
58
59 #Depois de todos os zeros contabilizados, hora de calcular
60 calcular:
61 mul $s0,$s0,$s1  # x * y
62 div $s0,$s0,$s2  # (x * y) / z
63 mul $s0,$s0,$s3  # adicionar os zeros # Respotas em s0
```

2.20 Programa20

```
.data
x: .word 10   # Declarando "x" como 10
y: .word 15   # Declarando "y" como 15
k: .word -1   # Declarando "k" como -1
.text

ori $t0,$zero,0x1001  # t0 = 1000
sll $t0,$t0,16    # t0 = 10000000

lw $s0, 0($t0)    # s0 = x = 10
lw $s1, 4($t0)    # s1 = y = 15

mult $s0,$s1    # Multiplicar o conteudo em s0 com s1
mflo $s2    # Carregar o conteudo em LO em s2

#Salvando o valor de k na m moria
sw $s2,8($t0)
```

2.21 Programa21

```
1  .data
2  x: .word 10     # Declarando "x" como 10
3  y: .word  3   # Declarando "y" como 3
4  .text
5
6  ori $t0,$zero,0x1001  # t0 = 1001
7  sll $t0,$t0,16    # t0 = 10010000
8
9  addi $t1,$t1,1    # t1 += 1
10
11 lw $s0, 0($t0)    # Carregando o conteudo em "0(t0)" s0 = x = 10
12 lw $s2, 0($t0)    # Carregando o conteudo em "0(t0)" s2 = copia de
       x
13 lw $s1, 4($t0)    # Carregando o conteudo em "4(t0)" s1 = y = 3
14
15 beq $t1, $s1, fim # va para fim ,se s1 for igual a t1
16
17 potencia:
18 addi $t1,$t1,1    # t1 += 1
19 mult $s2,$s0    # multiplique s2 com s0
20 mflo $s2    # Pegue o valor de LO e armazene em s2
21 bne $t1, $s1, potencia  # Va para potencia , se t1 for diferente de
       s1
22
23 fim:
24 #Salvando o valor de k na memoria
25 sw $s2,8($t0)
```

2.22 Programa22

```
1 .data
2   print: .ascii "Hello World!" # Declarando a String "Hello World!"
3 .text
4   li $v0, 4     #Carregando a v0, o comando de printar string , que
       eh v0
5   la $a0, print #Carregando em a0, o conteudo de print, (Obs: a0 ,
        o argumento)
6   syscall # chamar o sistema para realizar o comando em v0
```

2.23 Programa23

```
1  .data
2    text1: .asciiz "O menor numero  : \n"
3    menor: .word 0
4    text2: .ascii "O maior numero  : \n"
5    maior: .word 0
6    n: .ascii "\n"
7  .text
8
9    #Leitura dos tres numeros
10   li $v0, 5
11   syscall
12   move $t0, $v0
13
14   li $v0, 5
15   syscall
16   move $t1, $v0
17
18   li $v0, 5
19   syscall
20   move $t2, $v0
21
22   #Inicio da funcao para achar o menor
23   acharMenor:
24
25   #Imprimindo espacamento de linha
26   li $v0, 4
27   la $a0 ,n
28   syscall
29
30   blt $t0,$t1, compareMenorT0 #ir para compareMenorT0 se T0 for
       menor que T1
31   blt $t2,$t1, setMenorT2     #ir para setMenorT2, se T2 for menor
       que T1
32   j setMenorT1        #Se os ultimos dois testes derem errado,
       entao T1 e o menor numero
33
34   compareMenorT0:
```

```
35    blt $t0,$t2, setMenorT0 # Se t0 for menor do que t2 ir para
      setMenorT0
36    blt $t2,$t1, setMenorT2 # Se t2 for menor do que t1 ir para
      setMenorT2
37
38  # Funcao para configurar t0 como o menor valor
39  setMenorT0:
40    sw $t0, menor # salvando o menor valor em "menor"
41    j imprimirMenor # imprimir o menor elemento
42
43  # Funcao para configurar t1 como o menor valor
44  setMenorT1:
45    sw $t1,menor  # salvando o menor valor em "menor"
46    j imprimirMenor # imprimir o menor elemento
47
48  # Funcao para configurar t2 como o menor valor
49  setMenorT2:
50    sw $t2,menor  # salvando o menor valor em "menor"
51    j imprimirMenor # imprimir o menor elemento
52
53  imprimirMenor:
54
55    #Imprimindo text1 na tela
56                li $v0, 4
57    la $a0 ,text1
58    syscall
59
60    #Imprimindo na tela o menor numero
61    li $v0, 1
62    lw $a0, menor
63    syscall
64
65    j acharMaior #Ir para funcao para achar o menor elemento
66
67  acharMaior:
68
69  #Imprimindo o espacamento de tela
70  li $v0, 4
```

```
71  la $a0 ,n
72  syscall
73
74  bgt $t0,$t1, compareMaiorT0   #ir para compareMaiorT0, se t0 for
      maior que t1
75  bgt $t2,$t1, setMaiorT2   #ir para setMaiorT2, se t2 for maior
      que t1
76  j setMaiorT1      #ultimo caso, T1 e o maior elemento
77
78  compareMaiorT0:
79    bgt $t0,$t2, setMaiorT0 #se t0 for maior que T2, va para
      setMaiorT0
80    bgt $t2,$t1, setMaiorT2 #se t2 for maior que t1, va para
      setMaiorT2
81
82  # Funcao para configurar t0 como o maior valor
83    setMaiorT0:
84    sw $t0, maior   # salvando o maior valor em "maior"
85    j imprimirMaior   # imprimir o maior elemento
86
87  # Funcao para configurar t1 como o menor valor
88  setMaiorT1:
89    sw $t1,maior    # salvando o maior valor em "maior"
90    j imprimirMaior   # imprimir o maior elemento
91
92  # Funcao para configurar t2 como o menor valor
93  setMaiorT2:
94    sw $t2,maior    # salvando o maior valor em "maior"
95    j imprimirMaior   # imprimir o maior elemento
96
97   imprimirMaior:
98
99    #Imprimindo text2 na tela
100               li $v0, 4
101   la $a0 ,text2
102   syscall
103
104   #Imprimindo na tela o maior numero
```

```
105     li $v0, 1
106     lw $a0, maior
107     syscall
108
109   #FIM DO PROGRAMA
```

2.24 Programa24

```
.data
  text: .ascii "A soma dos elementos  :\n" # Texto para imprimir a
    soma de todos os elementos

.text

  #Funcao para configurar o array
  startArray:
  ori $s0,$zero,0x1001
  sll $s0,$s0,16
  add $s0,$s0,28

  #Ler o numero inteiro que sera o tamanho do array
  li $v0, 5
  syscall
  move $t0, $v0

  #Como e preciso declarar a quantidade de bits , multiplicamos o
    valor do usuario por 4
  sll $t0,$t0,2
  sub $t0,$t0,4

  add $t1, $zero,0 # Iniciando o contador para armazenar a posicao
    do array
  add $t2, $zero,1 # Iniciando o contador2 que sera os numeros
    adicionados ao array

  #Iniciando a insercao dos elementos no array
  while:
    bgt $t1,$t0, exit # Sair do loop se t1 for maior que t0
    sw $t2,0($s0) # salvar o conteudo de t2 na posicao t1 do array
    add $t3,$t3,$t2   # t3 += t2
    add $s0,$s0,4
    add $t2,$t2,2   # t2 += 2
    add $t1,$t1,4   # t1 += 4
    j while     # ir para "while"

```

```
34  exit:
35
36  #Printar na tela o conteudo em text
37  li $v0, 4
38  la $a0 ,text
39  syscall
40
41  #Printar na tela a soma dos elementos do array
42  li $v0, 1
43  add $a0, $zero,$t3
44  syscall
45
```

2.25 Programa25

```
1  .data
2    text1: .asciiz "Qual a temp. em Fahrenheit?\n" # Texto para
       perguntar o usuario a temperatura em Fahrenheit
3    text2: .asciiz "A temp. em Celsius e:\n"       # Texto para sair
       depois do calculo
4    ponto: .asciiz "."             # String para ter o "."
5  .text
6
7    #Armazenar em t1
8    add $t1,$zero,10
9
10   # Imprimir o conteudo de text1
11   li $v0,4
12   la $a0,text1
13   syscall
14
15   #Pegar o usuario o input da inteiro
16   li $v0,5
17   syscall
18   move $t0, $v0
19
20   #Fazer o calculo da conversao de fahrenheit para celsius
21   sub $t0,$t0,32  # t0 -= 32
22   mul $t0,$t0,50  # t0 *= 50
23   div $t0,$t0,9 # t0 /= 9
24
25   div $t0,$t1    # Separar o numero inteiro do decimal em LO e HI
26
27   mflo $t1       # Guardar o inteiro em t1
28   mfhi $t0       # Guardar o numero depois da virgula em t0
29
30   #Imprimir text2
31   li $v0,4
32   la $a0,text2
33   syscall
34
35   #Imprimir o numero inteiro
```

```
36  li $v0, 1
37  add $a0,$zero,$t1
38  syscall
39
40  #Imprimir o ponto
41  li $v0,4
42  la $a0,ponto
43  syscall
44
45  #Imprimir o valor depois da virgula
46  li $v0, 1
47  add $a0,$zero,$t0
48  syscall
```

3 Parte 3

3.1 Programa1

```
.data
  array:  .space 180       # Declaracao do array com 180 bits(
    espaco para 45 inteiros)
.text
  #Iniciando contadores

  addi $s0, $zero,0      # Auxiliar no fibonnaci
  addi $s1, $zero,1      # Primeiro termo da serie fibonnaci
  li $t0 , 0       # Contador da posicao do array

  sw $s1, array($t0)     # Salvando o primeiro termo da serie
    fibonnnaci no array

  while:
    beq $t0,184,exit   # Sair do loop , se t0 for igual a 184

    add $s2,$s1,$s0    # soma dos termos
    add $s1,$zero,$s0  # Auxiliar agora com o valor do primeiro
    termo
    add $s0,$zero,$s2  # salvando o valor da soma fibonnaci
    sw $s2,array($t0)  # salvando a soma no array

    add $t0,$t0,4    # aumentando o contador
    j while      # voltando ao while
  exit:
```

3.2 Programa2

```
.data
x: .word 121    # Iniciando com alguma palavra


.text

#Pegando a primeira posicao da memoria
ori $t0,$zero,0x1001
sll $t0,$t0,16

lw $s0,0($t0)   # Carregando o conteudo da primeira posicao da
    memoria

sub $s1,$zero,2   # Configurando para que s1 seja a "flag" do meu
    programa

#Verificando se o conteudo em s0 e menor ou igual a 100 e maior ou
    igual a 50
sle $s1,$s0,100
sge $s2,$s0,50
seq $s1,$s1,$s2
beq $s1,1,exit

#Verificando se o conteudo em s0 e menor ou igual a 200 e maior ou
    igual a 150
sle $s1,$s0,200
sge $s2,$s0,150
seq $s1,$s1,$s2
beq $s1,1,exit

#Saida do programa
exit:

#Imprimindo o valor da flag
li $v0, 1
add $a0,$zero,$s1
syscall
```

3.3 Programa3

```
1  .data
2  #Declarando os trs inteiros do programa, para que o mesmo descubra
       qual e a mediana
3  A: .word 23
4  B: .word 98
5  C: .word 17
6  .text
7
8  #Reservendo os primeiros espacos de memoria
9  ori $t0,$zero,0x1001
10 sll $t0,$t0,16
11
12 lw $s0,0($t0)   # Carregar o conteudo em 0(t0) | s0 = Valor da
       memoria de A
13 lw $s1,4($t0)         # Carregar o conteudo em 4(t0) | s1 = Valor
       da memoria de B
14 lw $s2,8($t0)   # Carregar o conteudo em 8(t0) | s1 = Valor da
       memoria de C
15
16 #Funcao para achar o valor da Mediana
17 AcharMediana:
18 ble $s1,$s0,verificarS1   #Se S1 for menor que S0, va para
       verificarS1
19 bge $s0,$s2,setMedianaS0  #Se S0 for maior que S2, sabendo que este
       e menor que S1, S0 e a mediana entao, va para setMedianaS0
20 j setMedianaS2      #Se no, va para setMedianaS2
21
22 #Verificando se S1 e a mediana ou nao
23 verificarS1:
24 bge $s1,$s2,setMedianaS1  # Se S1 for maior que S2, configure S1
       como a mediana em setMedianaS1
25 ble $s2,$s0 setMedianaS2      # Se no, a Mediana e S2, va para
       setMedianaS2
26 j setMedianaS1
27
28 #Configurar S0 como a mediana
29 setMedianaS0:
```

```
30 add $a0,$zero,$s0   #Adicionando a mediana como argumento a ser
       imprimido
31 j exit
32
33 #Configurar S1 como a mediana
34 setMedianaS1:
35 add $a0,$zero,$s1   #Adicionando a mediana como argumento a ser
       imprimido
36 j exit
37
38 #Configurar S2 como a mediana
39 setMedianaS2:
40 add $a0,$zero,$s2   #Adicionando a mediana como argumento a ser
       imprimido
41 j exit
42
43 #Saida do programa
44 exit:
45
46 #Imprimindo a mediana
47 li $v0, 1
48 syscall
```

3.4 Programa4

```
.data
x: .word 4
y: .word 2
z: .word 84
.text
#Criando um endereco para a primeira posicao de memoria em s1
ori $s1 ,$zero, 0x1001
sll $s1,$s1,16

#Iniciando variaveis para acumular os numeros
add $t0,$zero,0
add $s2,$zero,0
add $s3,$zero,0

 #Somar os elementos do array em s1
 while:
  lw $s2,0($s1)   # carregar o conteudo do array em t0, para s2
  add $s3,$s3,$s2   # t3 += t2
  add $s1,$s1,4   # t1 += 4
  add $t0,$t0,4
  ble $t0,400,while  # voltar ao loop se t0 for diferente de 404
 exit:

#resetando a posicao do vetor
ori $s1 ,$zero, 0x1001
sll $s1,$s1,16

sw $s3,0($s1) # salvando a soma dos elementos do inicio do array
```

3.4.1 Programa4 Adicionando nops

```
1 .data
2 x: .word 4
3 y: .word 2
4 z: .word 84
5 .text
6 #Criando um endereco para a primeira posicao de memoria em s1
7 ori $s1 ,$zero, 0x1001
8 sll $s1,$s1,16
9
10 #Iniciando variaveis para acumular os numeros
11 add $t0,$zero,0
12 add $s2,$zero,0
13 add $s3,$zero,0
14
15  #Somar os elementos do array em s1
16  while:
17   lw $s2,0($s1)   # carregar o conteudo do array em t0, para s2
18   add $s3,$s3,$s2   # t3 += t2
19   add $s1,$s1,4   # t1 += 4
20   add $t0,$t0,4
21   nop
22   nop
23   ble $t0,400,while  # voltar ao loop se t0 for diferente de 404
24  exit:
25
26 #resetando a posicao do vetor
27 ori $s1 ,$zero, 0x1001
28 sll $s1,$s1,16
29
30 sw $s3,0($s1) # salvando a soma dos elementos do inicio do array
```

Fazendos os calculos(Programa4):

Instruções da ALU: 3 — 78%— 2,34

Instruções de desvio: 4 — 11%— 0,44

Instruções de MEM: 5 — 11%— 0,55

$$CPIMedio = \frac{3 * 78\% + 4 * 11\% + 5 * 11\%}{12} = 0,2775 \quad (1)$$

Tempo de execução = 0,2775*12*10us = 0,00000333ms

(Programa4 com nops):

Instruções da ALU: 3 — 72%— 2,16

Instruções de desvio: 4 — 14%— 0,56

Instruções de MEM: 5 — 14%— 0,7

$$CPIMedio = \frac{3 * 72\% + 4 * 14\% + 5 * 14\%}{12} = 0,285 \quad (2)$$

Tempo de execução = 0,285*12*10us = 0,00000342ms

Speedup = 0,00000333ms / 0,00000342ms = 1,027 = 2,7%

3.5 Programa5

```
1  .text
2
3  #Iniciando s2 com um endereamento de memoria
4  ori $s2 ,$zero, 0x1001
5  sll $s2, $s2,16
6
7  addi $s3, $s2, 396    # s3 = s2 + 396
8
9  LOOP:
10   lw $s1, 0($s2)    # carregar o conteudo em "0($s2) para $s1
11   addi $s1, $s1, 1  # s1 += 1
12   sw $s1, 0 ($s2)   # salvando em "0($s2)" s1
13   addi $s2, $s2, 4  # s2 += 4
14   sub $s4, $s3, $s2 # s4 = s3 - s2
15   bne $s4, $zero, LOOP  # ir para LOOP , se s4 for diferente de
       zero
```

3.5.1 Programa5 Adicionando nops

```
1  .text
2
3  #Iniciando s2 com um endereamento de memoria
4  ori $s2 ,$zero, 0x1001
5  sll $s2, $s2,16
6
7  addi $s3, $s2, 396    # s3 = s2 + 396
8
9  LOOP:
10   lw $s1, 0($s2)    # carregar o conteudo em "0($s2) para $s1
11   addi $s1, $s1, 1  # s1 += 1
12   sw $s1, 0 ($s2)   # salvando em "0($s2)" s1
13   addi $s2, $s2, 4  # s2 += 4
14   sub $s4, $s3, $s2 # s4 = s3 - s2
15   nop     # Comando nulo
16   nop     # Comando nulo
17   bne $s4, $zero, LOOP  # ir para LOOP , se s4 for diferente de
       zero
```

Fazendos os calculos(Programa4):

Instruções da ALU: 3 — 50%— 1,5

Instruções de desvio: 4 — 17%— 0,68

Instruções de MEM: 5 — 33%— 1,65

$$CPIMedio = \frac{3 * 50\% + 4 * 17\% + 5 * 33\%}{12} = 0,31916666666 \quad (3)$$

Tempo de execução = 0,31916666666*12*10us = 0,0000038299ms

(Programa4 com nops):

Instruções da ALU: 3 — 63%— 1,89

Instruções de desvio: 4 — 12%— 0,48

Instruções de MEM: 5 — 25%— 1,25

$$CPIMedio = \frac{3 * 63\% + 4 * 12\% + 5 * 25\%}{12} = 0,30166666666 \quad (4)$$

Tempo de execução = 0,30166666666*12*10us = 0,00000361999ms

Speedup = 0,0000038299ms / 0,00000361999ms = 1,057 = 5,7%

3.6 Programa6

```
1  .data
2
3  .text
4    main:
5      #Configurando uma posicao de memoria
6      ori $a1, $zero, 0x1001
7      sll $a1, $a1,16   #a1 , o primeiro argumento, a posicao de
       memoria
8
9      #Configurando o tamanho do array
10     add $a2, $zero, 30 # a2, o segundo argumento, o tamanho do
       array
11
12     #Chamando a funcao IniciarVetor
13     jal iniciarVetor
14
15     #Imprimindo na tela a soma dos elementos do vetor
16     li $v0,1
17     add $a0,$zero,$v1
18     syscall
19
20     #Chamada de sistema para encerrar o programa
21     li $v0,10
22     syscall
23
24   iniciarVetor:
25
26     #Configurando um vetor "a1" de tamanho "a2"
27     add $s3,$zero,$a2
28     sll $a2, $zero,2
29
30     #iniciando os contadores
31     li $t0, 0
32     li $s4, 0
33     #iniciando variavel para verificar se o numero eh par ou impar
34     li $s0,2
35
```

```
36    configurarVetor:
37
38      #Verificando se a posicao atual no vetor e par ou impar
39      div $t0,$s0
40      mfhi $s1
41
42      beq $s1,0,opPar # Se for par
43      beq $s1,1,opImpar # Se for Impar
44
45      saida:
46      sw $s2,0($a1)     # Salvando a operacao feita em "opPar" ou "
      opImpar" na posicao atual do array
47      add $s4,$s4,$s2     # Adicionando o resultado da operacao no
      acumulador
48      add $a1,$a1,4     # Movendo a posicao no array
49      add $t0,$t0,1     # Aumentando o contador
50      bne $t0,$s3,configurarVetor # Comando para sair do loop
      ConfigurarVetor
51
52  configurarVetor2:
53
54    #Adicionando o acumulador ao registrador de retorno
55    add $v1,$zero,$s4
56    jr $ra
57
58  opPar:
59    #v[i] = 2i - i
60    mul $s2,$t0,2
61    sub $s2,$s2,1
62    j saida
63
64  opImpar:
65    #v[i] = i
66    add $s2,$zero,$t0
67    j saida
68
69
```

3.7 Programa7

```
.data
input1:   .word 0
input2:   .word 0
resultado:  .word 0
n:    .ascii "\n"

.text

  #Pegar o usuario o input da inteiro
  main:
  li $v0,5
  syscall
  move $a1, $v0

  beqz $a1,terminarPrograma

  #Pegar o usuario o input da inteiro
  li $v0,5
  syscall
  move $a2, $v0

  #Conferir se a potencia lida e igual a 0 e 1
  beq $a2, 1, fim # Se a potencia for igual a 1, va para fim
  beq $a2, 0, fim2# Se a potencia for igual a 0, va para fim 2

  #ir para "potencia" e levar "a1" e "a2"
  jal potencia

  #imprimir o resultado de potencia na tela
  li $v0,1
  add $a0,$zero,$v1
  syscall

  #imprimir espacamento de tela
  li $v0,4
  la $a0,n
  syscall
```

```
38
39  #chamar funcao pra terminar o programa
40  j main
41
42  potencia:
43
44  add $s0,$zero,$a1   #Salvar uma copia de "a1" em "s0"
45  add $t1,$zero,1     #Iniciar "t1" com o valor 1
46  while:
47    addi $t1,$t1,1    # t1 += 1
48    mult $a1,$s0    # multiplique a1 com s0
49    mflo $a1    # Pegue o valor de LO e armazene em a1
50    bne $t1, $a2, while     # Volte ao loop, se t1 for diferente de
       a2
51
52  add $v1 ,$zero,$a1    # Adicionar o resultado ao argumento de
      volta
53  sw $v1,resultado    # Salvar o resultado encontrado em "resultado
      "
54  jr $ra        # voltar para onde potencia foi chamado, com o
      resultado em "v1"
55
56  #Fim do codigo se o numero esta sendo elevado a 1
57  fim:
58  sw $a1,resultado
59
60  #Imprima na tela o proprio numero
61  li $v0,1
62  add $a0,$zero,$a1
63  syscall
64
65  #imprimir espacamento de tela
66  li $v0,4
67  la $a0,n
68  syscall
69
70  #chamar funcao pra terminar o programa
71  j main
```

```
72
73  #Fim do programa se o numero esta sendo elevado a 0
74  fim2:
75  sw $0,resultado
76
77  #Imprima na tela o proprio numero
78  li $v0,1
79  add $a0,$zero,1
80  syscall
81
82  #imprimir espacamento de tela
83  li $v0,4
84  la $a0,n
85  syscall
86
87  #chamar funcao pra terminar o programa
88  j main
89
90  #Funcao pra terminar o programa
91  terminarPrograma:
92  #Pedir ao programa para terminar
93  li $v0,10
94  syscall
```

4 Parte 4

4.1 Questão 1

```
.data
  endereco: .word 0x10010020  #Endereco de memoria
  tam: .word 3              #Quantidade de elementos
.text
main:
  addi $t0, $zero, 30     #Criando variavel para armazenar o numero
     limite de elementos

  lw $a0, endereco      #Carregando da memoria o endereco
  lw $a1, tam     #Carregando da memoria o tamanho do array

verificar:
  bgt $a1, $t0, maiorDoQue30  #Se o tamanho do array for maior do
    que 30, va para maiorDoQue30
dentro:
  add $s0, $zero, $a0     #s0 = Endereco de memoria
  add $t0, $zero, $a1     #t0 = quantidade de elementos
  sll $t0, $t0, 2         #t0*= 2^2
  add $s1, $t0, $s0       #s1 = array[Quantidade-1]
  j somar       #ir para somar
maiorDoQue30:
  add $s0, $zero, $a0   #s0 = Endereco de memoria
  addi $t0, $zero, 30     #t0 = 0 + 30
  sll $t0, $t0, 2         #t0*= 2^2
  add $s1, $t0, $s0       #s1 = array[29]
  j somar

somar:
  addi $t0, $zero, 0    #resetar t0

parOuImpar:
  andi $t1, $t0, 1      #Decidi se t1 for par ou impar
  beqz $t1, par     #Se t1 for par
  j impar       #Caso contrario
```

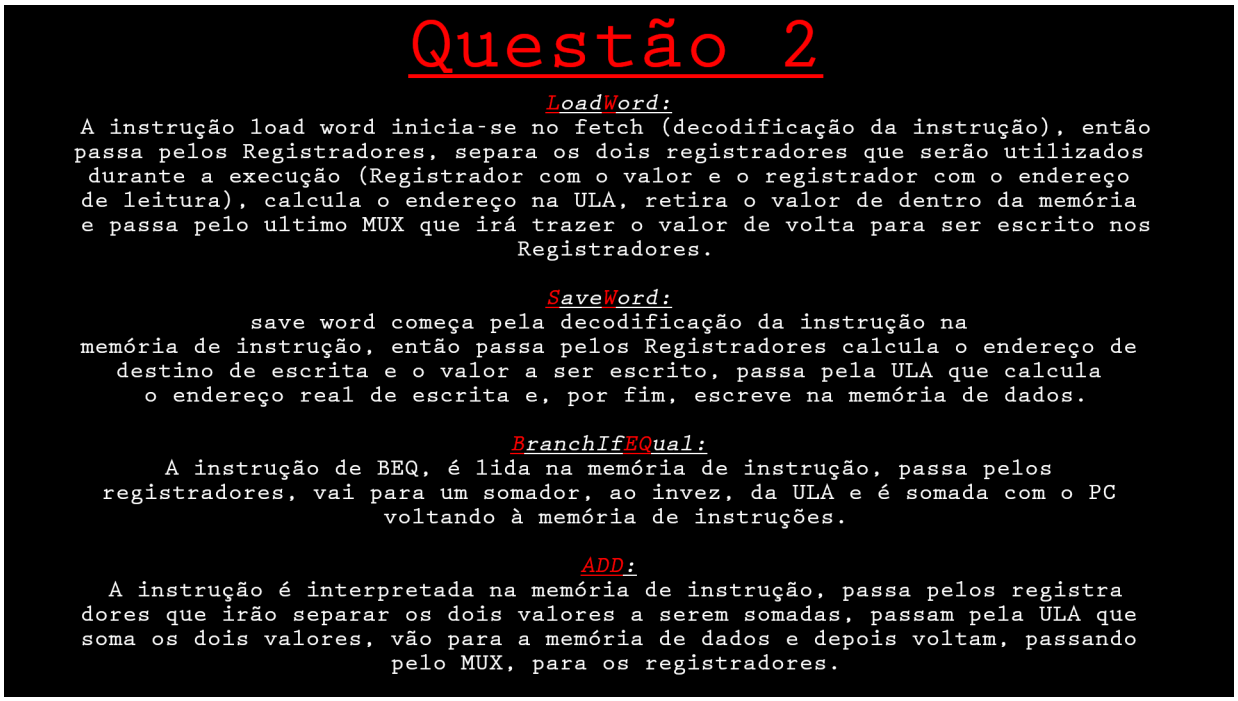
```
ret:
  addi $t0, $t0, 1      #t0++
  sll $t6, $t0, 2     #t6 = t0 * 2^2
  add $t6, $s0, $t6     #t6 = s0 + t6
  bne $t6, $s1, parOuImpar    #Va para parOuImpar, se t6 for
    diferente de s1
  j fim       #Ir para fim
impar:
  add $t2, $zero, $t0   # t2 = t0
  mult $t2, $t2     # t2 * t2 -> vai para LO e HI
  mflo $t2      # pegar o conteudo em LO e passar para t2
  sll $t5, $t0, 2     # t5 = t0 * 2^2
  add $t5, $s0, $t5     # t5 = s0 + t5
  sw $t2, ($t5)     # Salvar t2 na posicao de memoria de t5
  j ret       # ir para ret

par:
  addi $t2, $zero, 1    # t2 = 1
  addi $t3, $zero, 2    # t3 = 2
  mult $t3, $t0     # t3 * t0 -> vai para LO e HI
  mflo $t3      # armazene o conteudo de LO em t3
  mult $t0, $t0     # t0 * t0 -> vai para LO e HI
  mflo $t4      # armazene o conteudo de LO em t4
  sll $t4, $t4, 1     # t4 *= 2^1
  add $t2, $t2, $t3     # t2 += t3
  add $t2, $t2, $t4     # t2 += t4
  sll $t5, $t0, 2     # t5 = t0 * 2^2
  add $t5, $s0, $t5     # t5 += s0
  sw $t2, ($t5)     # salvar t2 na posicao de memoria em t5
  j ret       # ir para ret


retornar:
  add $t0, $zero, $a0     #t0 = a0
  mult $t0, $t0           #t0 * t0 -> vai para LO e HI
  mflo $t0,               #armazena o conteudo de LO em t0
  addu $v0, $zero, $t0    #v0 = t0
  jr $ra                #retornar
```

```
70 fim:
71   li $v0, 10
72   syscall
```

4.2 Questão 2


Questão 2
LoadWord:
A instrução load word inicia-se no fetch (decodificação da instrução), então passa pelos Registradores, separa os dois registradores que serão utilizados durante a execução (Registrador com o valor e o registrador com o endereço de leitura), calcula o endereço na ULA, retira o valor de dentro da memória e passa pelo ultimo MUX que irá trazer o valor de volta para ser escrito nos Registradores.
SaveWord:
save word começa pela decodificação da instrução na memória de instrução, então passa pelos Registradores calcula o endereço de destino de escrita e o valor a ser escrito, passa pela ULA que calcula o endereço real de escrita e, por fim, escreve na memória de dados.
BranchIfEQual:
A instrução de BEQ, é lida na memória de instrução, passa pelos registradores, vai para um somador, ao invez, da ULA e é somada com o PC voltando à memória de instruções.
ADD:
A instrução é interpretada na memória de instrução, passa pelos registra dores que irão separar os dois valores a serem somadas, passam pela ULA que soma os dois valores, vão para a memória de dados e depois voltam, passando pelo MUX, para os registradores.

4.3 Questão 3

Questão3

A)

LW - 10ns/11ns
SW - 10ns/11ns
BEQ - 6ns/7ns
ADD - 6ns/8ns
J - 4ns

B)

GCC:

10*0.22 + 10*0.11 + 6*0.49 + 6*0.16 + 4*0.02 = 2.2 + 1.1
+ 2.94 + 0.96 + 0.08 = 7.28
SpeedUp = 10 / 7.28 = 1.37

11*0.22 + 11*0.11 + 8*0.49 + 7*0.16 + 4*0.02 = 2.42 +
1.21 + 3.92 + 1.12 + 0.08 = 8.75
SpeedUp = 11 / 8.75 = 1.25

Questão3

ABC:

10*0.11 + 10*0.49 + 6*0.22 + 6*0.02 + 4*0.16 = 1.1 + 4.9 + 1.32 +
0.12 + 0.64 = 8.08
SpeedUp = 10 / 8.08 = 1.23

11*0.11 + 11*0.49 + 8*0.22 + 7*0.02 + 4*0.16 = 1.21 + 5.39 + 1.76 + 0.14 +
0.64 = 9.14
SpeedUp = 11/9.14 = 1.20

4.4 Questão 4

```
 Resolução da Questão 4
* Carregar uma instrução da memória e incrementa o PC;
* Traduz o "OpCode" para sinais de controle, e carrega o conteúdo
dos registradores;
* Executa a instrução carregada e/ou computa operações de desvio;
* Acessa a memória de dados se necessário;
* Escreve o resultado da instrução em um registrador;
```