Exercício Prático 06 - MIPS
806454 - Yago Almeida Melo

Parte 1 - Questões

1) A. um arquivo de texto que contém instruções de linguagem de programação.
2) B. uma parte do processador que possui um padrão de bits.
3) A. #
4) C. 32
5) D. parte do processador que contém o endereço da próxima instrução de máquina para ser obtida.
6) C. 4
7) D. uma declaração que diz o montador algo sobre o que o programador quer, mas não corresponde diretamente a uma instrução de máquina.
8) D. um nome usado no código-fonte em linguagem assembly para um local na memória.
9) B. 0x00400000
10) A. operando imediato.
11) B. operação bitwise.
12) D. Cada um dos registradores deve possuir 32 bits.
13) B. Os dados são estendidos em zero à esquerda por 16 bits.
14) C. ori $5, $0, 48
15) B. Sim
16) D. andi $8, $8, 0xFF
17) A. Todos os bits em zero
18) A. Não. Diferentes instruções de máquina possuem campos diferentes.

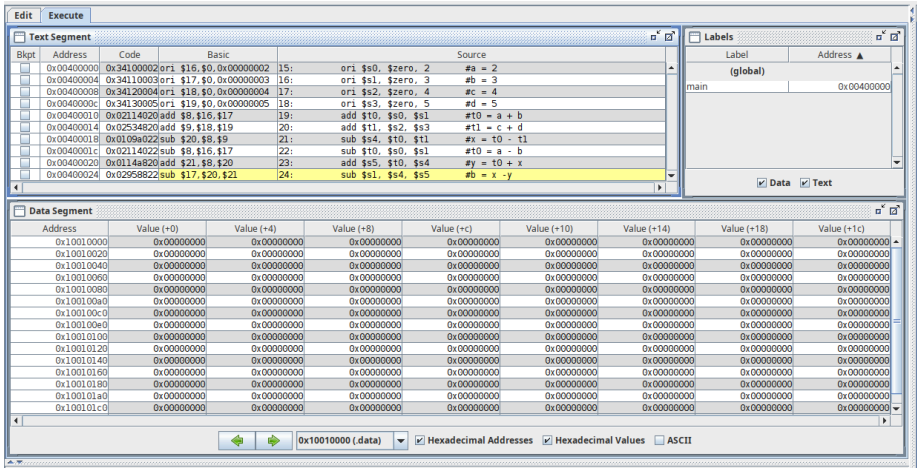
Parte 2 - Implementação no MARS

1)

Edit Execute

mips1.asm

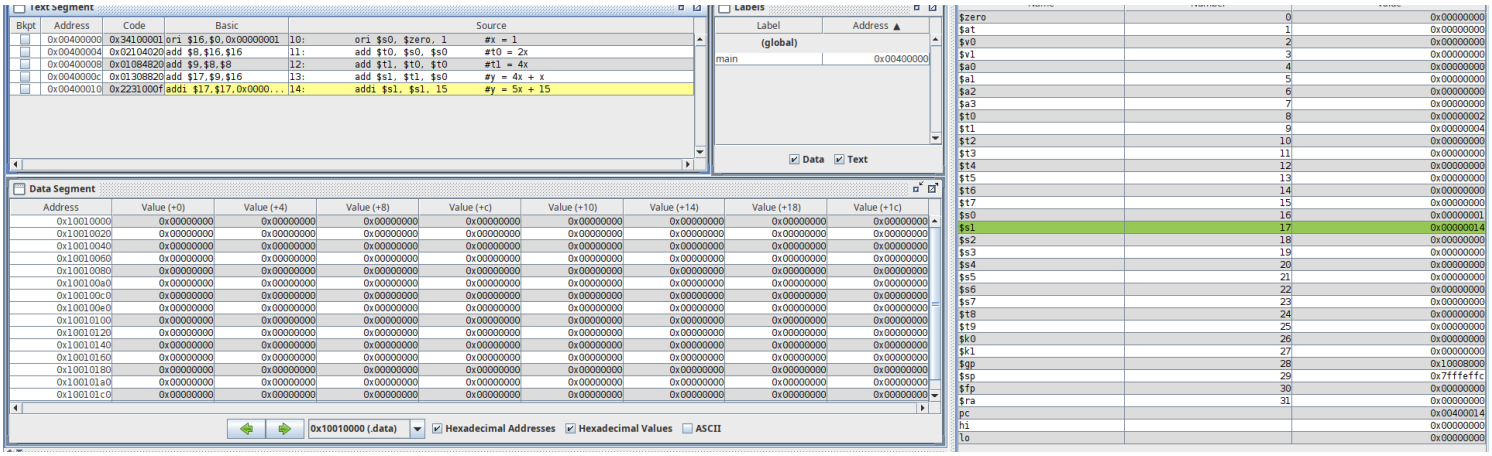
```
#       a=2;
#       b=3;
#       c=4;
#       d=5;
#       x=(a+b)-(c+d);
#       y=a-b+x;
#       b=x-y;

#INICIO

.text
.globl main
main:
        ori $s0, $zero, 2       #a = 2
        ori $s1, $zero, 3       #b = 3
        ori $s2, $zero, 4       #c = 4
        ori $s3, $zero, 5       #d = 5
        add $t0, $s0, $s1       #t0 = a + b
        add $t1, $s2, $s3       #t1 = c + d
        sub $s4, $t0, $t1       #x = t0 - t1
        sub $t0, $s0, $s1       #t0 = a - b
        add $s5, $t0, $s4       #y = t0 + x
        sub $s1, $s4, $s5       #b = x -y

#FIM
```



2)

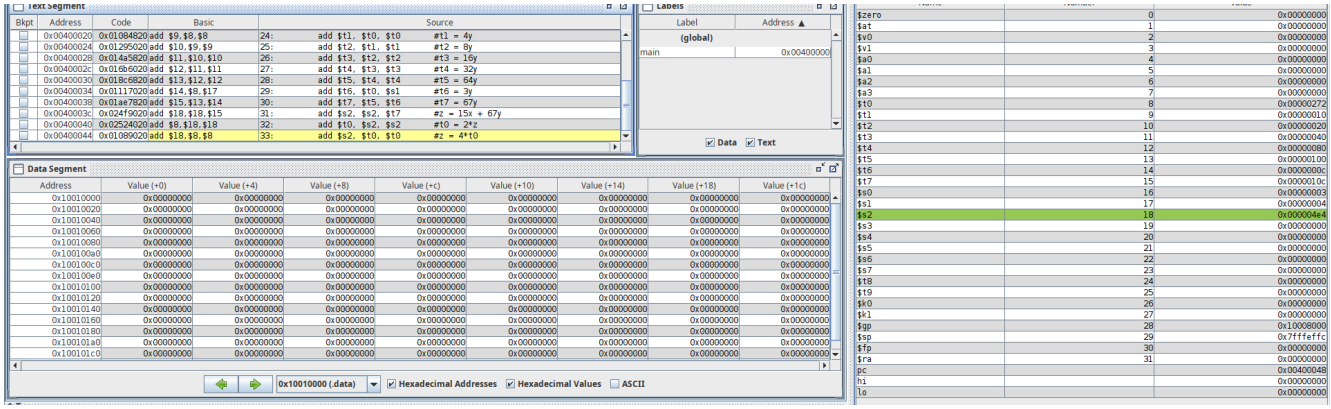
```
1  #       Programa 2
2  #       x=1;;
3  #       y = 5*x + 15;;
4
5  #INICIO
6
7  .text
8  .globl main
9  main:
10         ori $s0, $zero, 1       #x = 1
11         add $t0, $s0, $s0       #t0 = 2x
12         add $t1, $t0, $t0       #t1 = 4x
13         add $s1, $t1, $s0       #y = 4x + x
14         addi $s1, $s1, 15       #y = 5x + 15
15 #FIM
16
```

Line: 16 Column: 1 Show Line Numbers



3)

```
2  #INICIO
3
4  #       Programa 3
5  #       x=3;
6  #       y=4;
7  #       z=(15*x + 67*y)*4
8
9  .text
10 .globl main
11 main:
12         ori $s0, $zero, 3       #x = 3
13         ori $s1, $zero, 4       #y = 4
14         add $t0, $s0, $s0       #t0 = 2x
15         add $t1, $t0, $t0       #t1 = 4x
16         add $t2, $t1, $t1       #t2 = 8x
17         add $t3, $t2, $t2       #t3 = 16x
18         sub $s2, $t3, $s0       #z = 15x
19         add $t0, $s1, $s1       #t0 = 2y
20         add $t1, $t0, $t0       #t1 = 4y
21         add $t2, $t1, $t1       #t2 = 8y
22         add $t3, $t2, $t2       #t3 = 16y
23         add $t4, $t3, $t3       #t4 = 32y
24         add $t5, $t4, $t4       #t5 = 64y
25         add $t6, $t0, $s1       #t6 = 3y
26         add $t7, $t5, $t6       #t7 = 67y
27         add $s2, $s2, $t7       #z = 15x + 67y
28         add $t0, $s2, $s2       #t0 = 2*z
29         add $s2, $t0, $t0       #z = 4*t0
30
31 #FIM
32
33
34
35
Line: 35 Column: 5  Show Line Numbers
```

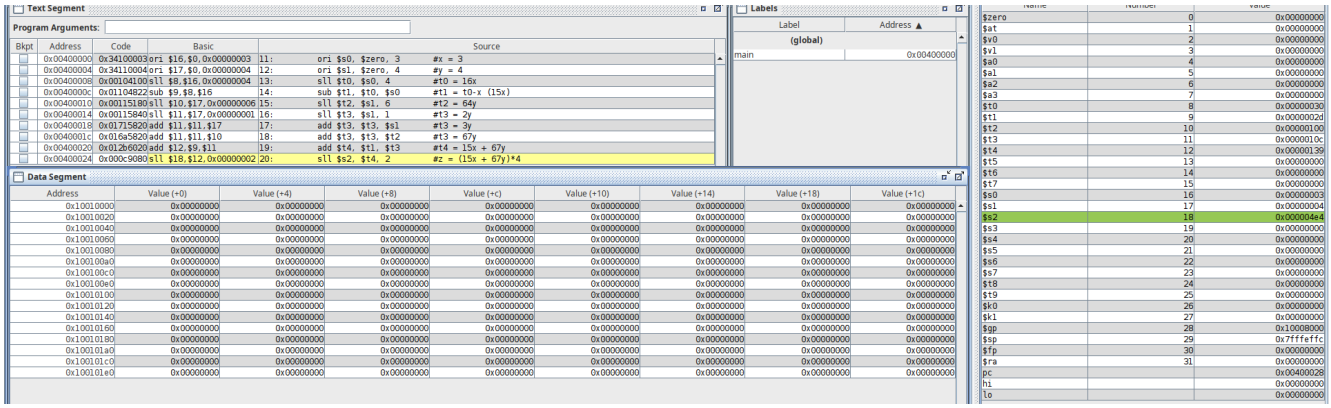


4)

mips6.asm | mips4.asm

```
1  #       Programa 4
2  #       x=3;
3  #       y=4;
4  #       z=(15*x + 67*y)*4
5
6  #INICIO
7
8  .text
9  .globl main
10 main:
11         ori $s0, $zero, 3       #x = 3
12         ori $s1, $zero, 4       #y = 4
13         sll $t0, $s0, 4         #t0 = 16x
14         sub $t1, $t0, $s0       #t1 = t0-x (15x)
15         sll $t2, $s1, 6         #t2 = 64y
16         sll $t3, $s1, 1         #t3 = 2y
17         add $t3, $t3, $s1       #t3 = 3y
18         add $t3, $t3, $t2       #t3 = 67y
19         add $t4, $t1, $t3       #t4 = 15x + 67y
20         sll $s2, $t4, 2         #z = (15x + 67y)*4
21
22 #FIM
23
```

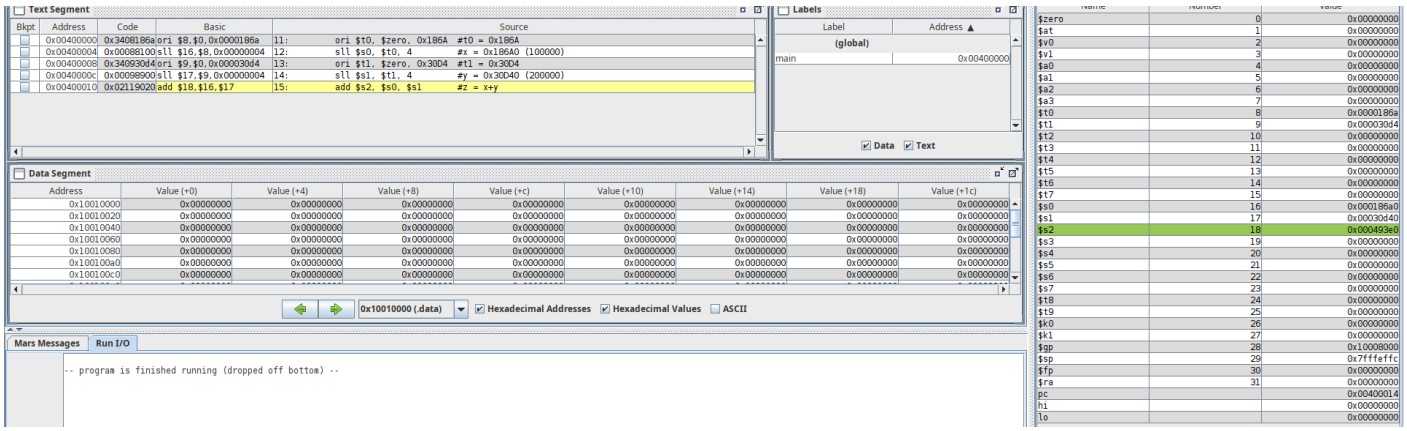
Line: 20 Column: 37 ☑ Show Line Numbers



5)

```
1  #       Programa 5
2  #       x=100000;
3  #       y=200000;
4  #       z=x+y
5
6  #INICIO
7
8  .text
9  .globl main
10 main:
11         ori $t0, $zero, 0x186A  #t0 = 0x186A
12         sll $s0, $t0, 4         #x = 0x186A0 (100000)
13         ori $t1, $zero, 0x30D4  #t1 = 0x30D4
14         sll $s1, $t1, 4         #y = 0x30D40 (200000)
15         add $s2, $s0, $s1       #z = x+y
16
17 #FIM
18
```

Line: 18 Column: 1 Show Line Numbers

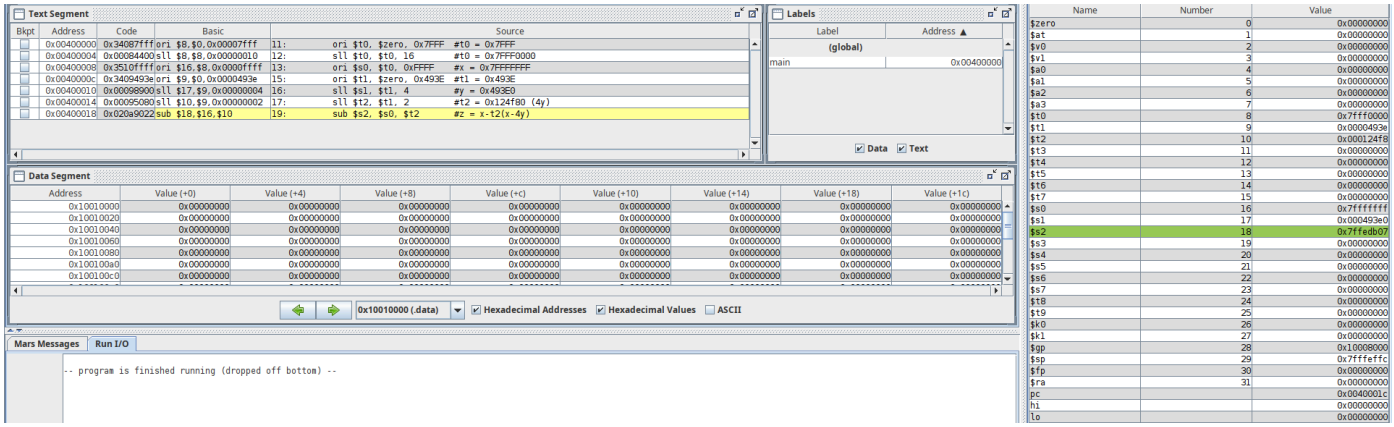


6)

mips6.asm

```
#       Programa 6
#       x=maior inteiro possivel (0x7FFFFFFF);
#       y=300000;
#       z=x-4y

#INICIO

.text
.globl main
main:
        ori $t0, $zero, 0x7FFF  #t0 = 0x7FFF
        sll $t0, $t0, 16        #t0 = 0x7FFF0000
        ori $s0, $t0, 0xFFFF    #x = 0x7FFFFFFF

        ori $t1, $zero, 0x493E  #t1 = 0x493E
        sll $s1, $t1, 4         #y = 0x493E0
        sll $t2, $t1, 2         #t2 = 0x124f80 (4y)

        sub $s2, $s0, $t2       #z = x-t2(x-4y)

#FIM
```

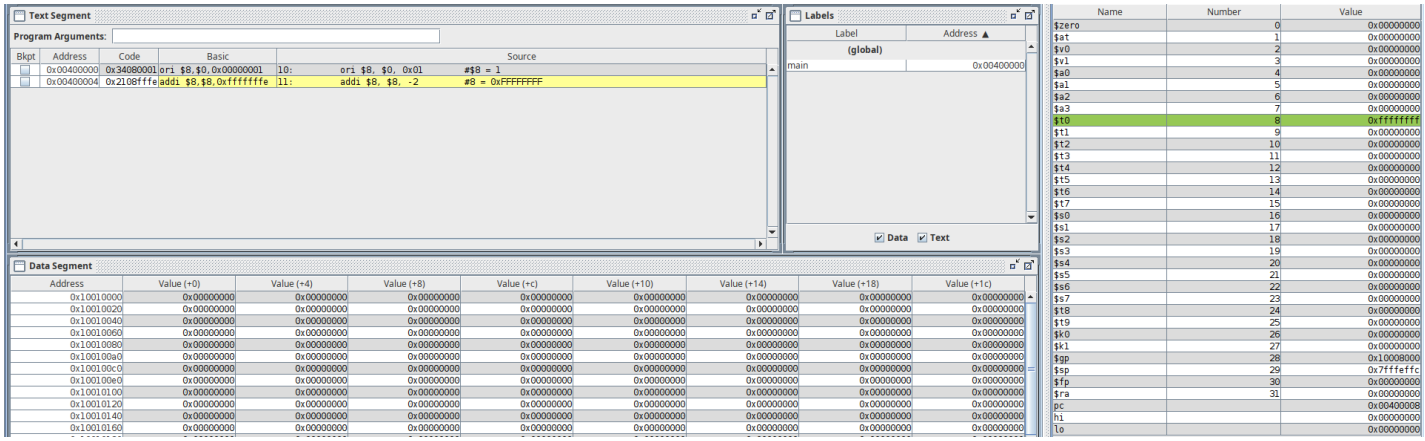
Line: 22 Column: 1 Show Line Numbers



7)

mips7.asm

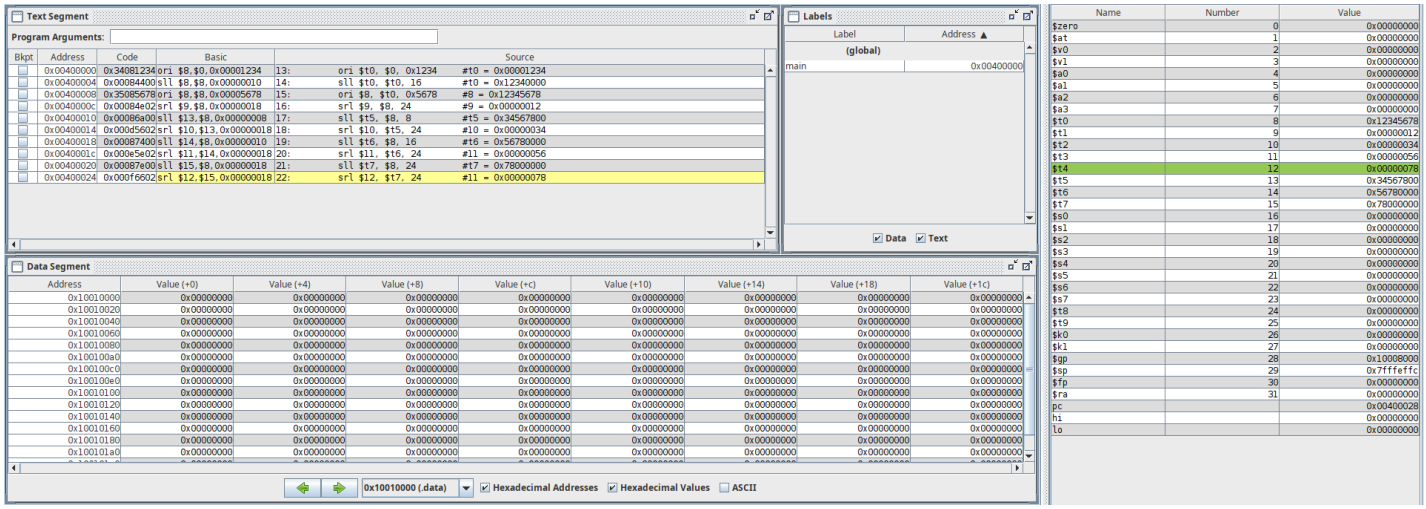
```
1  #       Programa 7
2  #       ori $8, $0, 0x01
3  #       $8 = 0xFFFFFFFF
4
5  #INICIO
6
7  .text
8  .globl main
9  main:
10         ori $8, $0, 0x01        #$8 = 1
11         addi $8, $8, -2         #8 = 0xFFFFFFFF
12
13 #FIM
14
```



8)

mips8.asm

```
1  #       Programa 8
2  #       ori $8, $0, 0x12345678
3  #       $9 = 0x12
4  #       $10 = 0x34
5  #       $11 = 0x56
6  #       $12 = 0x78
7
8  #INICIO
9
10 .text
11 .globl main
12 main:
13         ori $t0, $0, 0x1234     #t0 = 0x00001234
14         sll $t0, $t0, 16        #t0 = 0x12340000
15         ori $8, $t0, 0x5678     #8 = 0x12345678
16         srl $9, $8, 24          #9 = 0x00000012
17         sll $t5, $8, 8          #t5 = 0x34567800
18         srl $10, $t5, 24        #10 = 0x00000034
19         sll $t6, $8, 16         #t6 = 0x56780000
20         srl $11, $t6, 24        #11 = 0x00000056
21         sll $t7, $8, 24         #t7 = 0x78000000
22         srl $12, $t7, 24        #11 = 0x00000078
23
24 #FIM
25
```

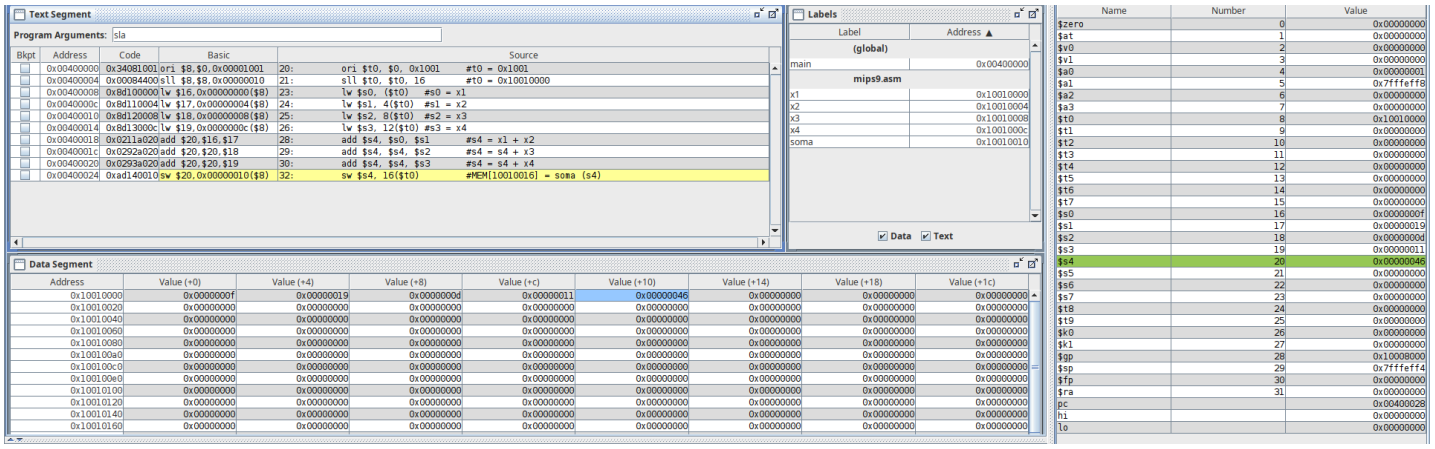


9)

mips9.asm

```
#       Programa 9
#       Escrever um programa que leia todos os números,
#       calcule e substitua o valor da variável soma por este valor.

#INICIO

.data
x1: .word 15
x2: .word 25
x3: .word 13
x4: .word 17
soma: .word -1

.text
.globl main
main:
        #t0 -> first address
        #t1 -> offset

        ori $t0, $0, 0x1001     #t0 = 0x1001
        sll $t0, $t0, 16        #t0 = 0x10010000

        lw $s0, ($t0)   #s0 = x1
        lw $s1, 4($t0)  #s1 = x2
        lw $s2, 8($t0)  #s2 = x3
        lw $s3, 12($t0) #s3 = x4

        add $s4, $s0, $s1       #s4 = x1 + x2
        add $s4, $s4, $s2       #s4 = s4 + x3
        add $s4, $s4, $s3       #s4 = s4 + x4

        sw $s4, 16($t0)         #MEM[10010016] = soma (s4)

#FIM
```

Line: 34 Column: 5 Show Line Numbers



Text Segment

Program Arguments: sla

Bkpt	Address	Code	Basic		Source
	0x00400000	0x34081001	ori $8,$0,0x00001001	20:	ori $t0, $0, 0x1001 #t0 = 0x1001
	0x00400004	0x00084400	sll $8,$8,0x00000010	21:	sll $t0, $t0, 16 #t0 = 0x10010000
	0x00400008	0x8d100000	lw $16,0x00000000($8)	23:	lw $s0, ($t0) #s0 = x1
	0x0040000c	0x8d110004	lw $17,0x00000004($8)	24:	lw $s1, 4($t0) #s1 = x2
	0x00400010	0x8d120008	lw $18,0x00000008($8)	25:	lw $s2, 8($t0) #s2 = x3
	0x00400014	0x8d13000c	lw $19,0x0000000c($8)	26:	lw $s3, 12($t0) #s3 = x4
	0x00400018	0x0211a020	add $20,$16,$17	28:	add $s4, $s0, $s1 #s4 = x1 + x2
	0x0040001c	0x0292a020	add $20,$20,$18	29:	add $s4, $s4, $s2 #s4 = s4 + x3
	0x00400020	0x0293a020	add $20,$20,$19	30:	add $s4, $s4, $s3 #s4 = s4 + x4
	0x00400024	0xad140010	sw $20,0x00000010($8)	32:	sw $s4, 16($t0) #MEM[10010016] = soma (s4)

Labels

Label	Address
(global)	
main	0x00400000
mips9.asm	
x1	0x10010000
x2	0x10010004
x3	0x10010008
x4	0x1001000c
soma	0x10010010

Data Text

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x0000000f	0x00000019	0x0000000d	0x00000011	0x00000046	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010140	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010160	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000001
$a1	5	0x7fffeff8
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x10010000
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x0000000f
$s1	17	0x00000019
$s2	18	0x0000000d
$s3	19	0x00000011
$s4	20	0x00000046
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeff4
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400028
hi		0x00000000
lo		0x00000000

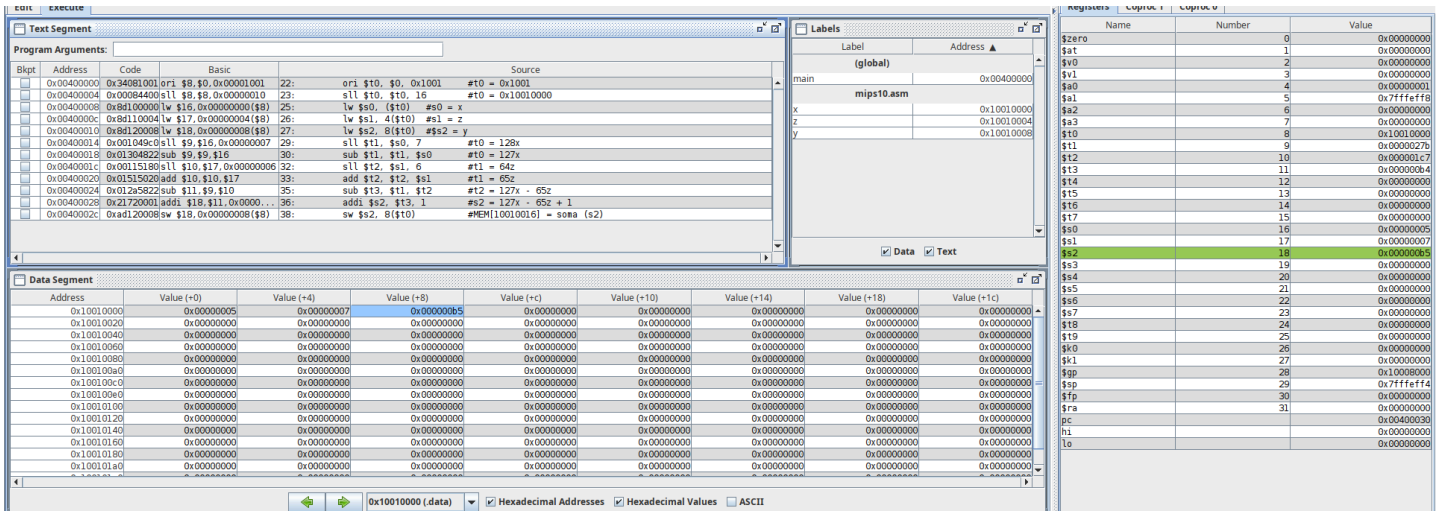
10)

Edit Execute

mips10.asm

```
1  #       Programa 10
2  #       Calcule o valor de y conhecendo os valores de x e z, estes armazenados na memória
3  #       Substitua o valor de y pelo seguinte programa y = 127x - 65z +1.
4
5  #INICIO
6
7  .data
8  x: .word 5
9  z: .word 7
10 y: .word 0      #será sobrescrito após execução
11
12 .text
13 .globl main
14 main:
15         #t0 -> first address
16         #t1 -> offset
17
18         #x -> $s0
19         #z -> $s1
20         #y -> $s2
21
22         ori $t0, $0, 0x1001     #t0 = 0x1001
23         sll $t0, $t0, 16        #t0 = 0x10010000
24
25         lw $s0, ($t0)   #s0 = x
26         lw $s1, 4($t0)  #s1 = z
27         lw $s2, 8($t0)  #$s2 = y
28
29         sll $t1, $s0, 7         #t0 = 128x
30         sub $t1, $t1, $s0       #t0 = 127x
31
32         sll $t2, $s1, 6         #t1 = 64z
33         add $t2, $t2, $s1       #t1 = 65z
34
35         sub $t3, $t1, $t2       #t2 = 127x - 65z
36         addi $s2, $t3, 1        #s2 = 127x - 65z + 1
37
38         sw $s2, 8($t0)          #MEM[10010016] = soma (s2)
39 #FIM
```

Line: 26 Column: 24 Show Line Numbers



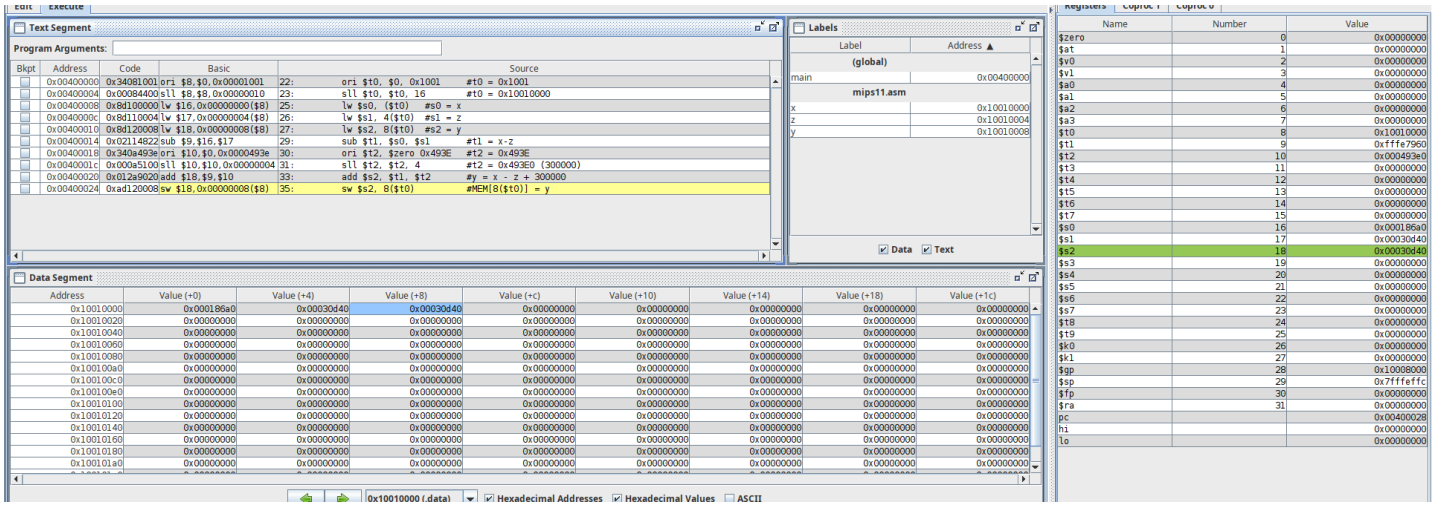
11)

Edit | Execute

mips11.asm

```
1  #       Programa 11
2  #       Calcule o valor de y conhecendo os valores de x e z, estes armazenados na memória
3  #       Substitua o valor de y pelo seguinte programa y = x - z + 300000
4
5  #INICIO
6
7  .data
8  x: .word 100000
9  z: .word 200000
10 y: .word 0      #será sobrescrito após execução
11
12 .text
13 .globl main
14 main:
15         #t0 -> first address
16         #t1 -> offset
17
18         #x -> $s0
19         #z -> $s1
20         #y -> $s2
21
22         ori $t0, $0, 0x1001     #t0 = 0x1001
23         sll $t0, $t0, 16        #t0 = 0x10010000
24
25         lw $s0, ($t0)   #s0 = x
26         lw $s1, 4($t0)  #s1 = z
27         lw $s2, 8($t0)  #s2 = y
28
29         sub $t1, $s0, $s1       #t1 = x-z
30         ori $t2, $zero 0x493E   #t2 = 0x493E
31         sll $t2, $t2, 4         #t2 = 0x493E0 (300000)
32
33         add $s2, $t1, $t2       #y = x - z + 300000
34
35         sw $s2, 8($t0)          #MEM[8($t0)] = y
36
37 #FIM
38
```

Line: 38 Column: 1 ☑ Show Line Numbers



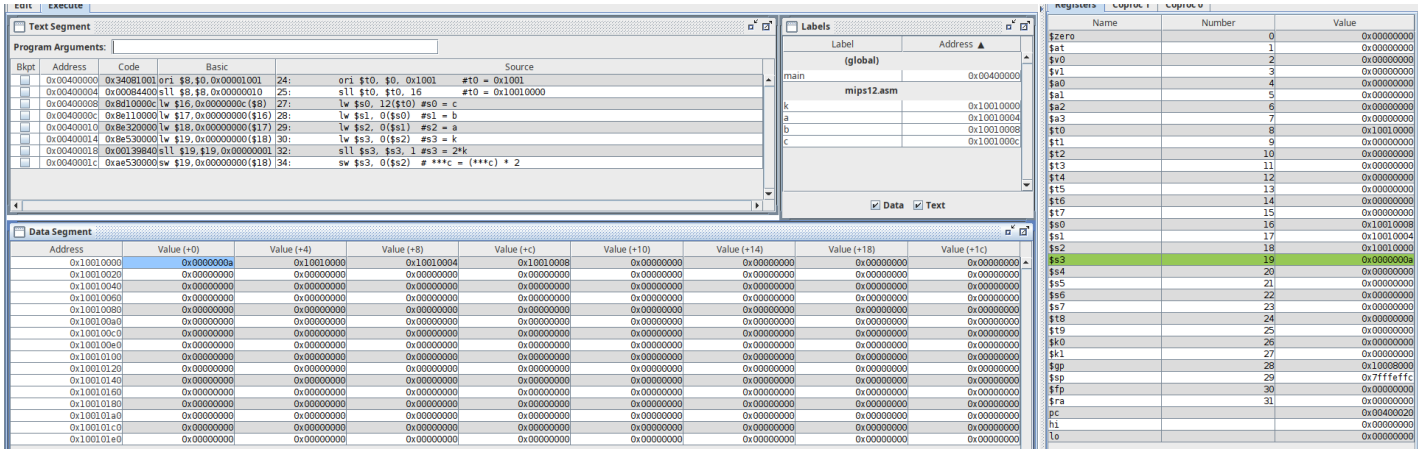
12)

Edit | Execute

mips12.asm

```
1  #       Programa 12
2  #       Considere int ***x;
3  #       A primeira posição da memória é do int, coloque os outros valores em reg's e use endereços de memória
4  #       O programa deve ler o valor k, multiplicar por 2 e reescrever no local da memoria
5
6  #INICIO
7
8  .data
9  k: .word 5
10 a: .word 0x10010000
11 b: .word 0x10010004
12 c: .word 0x10010008
13
14 .text
15 .globl main
16 main:
17         #t0 -> first address
18
19         #k -> $s0
20         #a -> $s1
21         #b -> $s2
22         #c -> $s3
23
24         ori $t0, $0, 0x1001     #t0 = 0x1001
25         sll $t0, $t0, 16        #t0 = 0x10010000
26
27         lw $s0, 12($t0) #s0 = c
28         lw $s1, 0($s0)  #s1 = b
29         lw $s2, 0($s1)  #s2 = a
30         lw $s3, 0($s2)  #s3 = k
31
32         sll $s3, $s3, 1 #s3 = 2*k
33
34         sw $s3, 0($s2)  # ***c = (***c) * 2
35
36 #FIM
37
```

Line: 37 Column: 1 ☑ Show Line Numbers

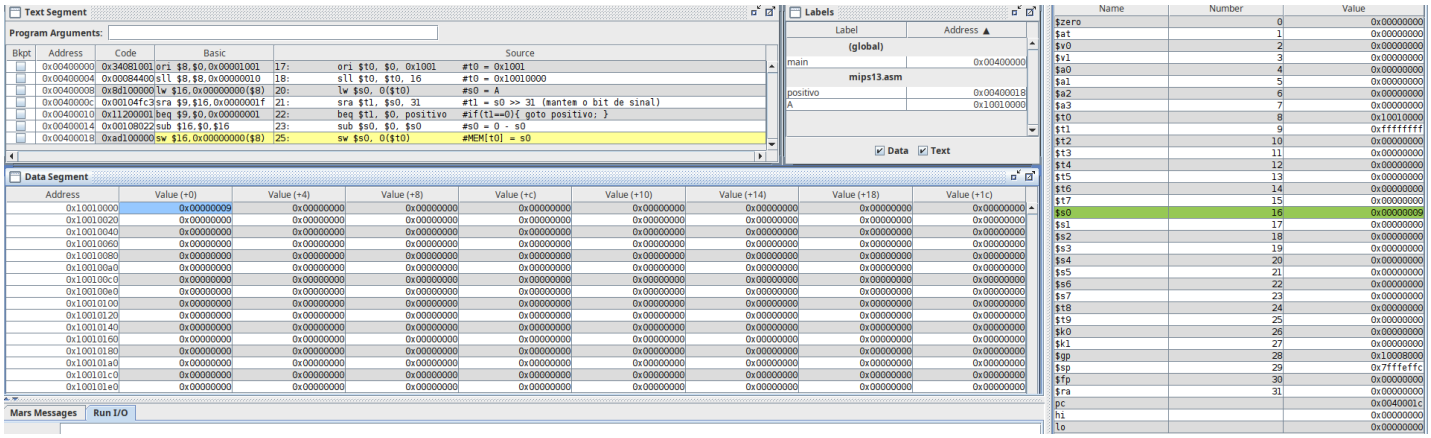


13)

mips13.asm

```
#       Programa 13
#       Escreva um programa que leia um valor A da memória, identifique se é negativo ou não e encontre seu modulo
#       O valor deve ser reescrito sobre A

#INICIO

.data
A: .word -9

.text
.globl main
main:
        #t0 -> first address

        #A -> $s0

        ori $t0, $0, 0x1001     #t0 = 0x1001
        sll $t0, $t0, 16        #t0 = 0x10010000

        lw $s0, 0($t0)          #s0 = A
        sra $t1, $s0, 31        #t1 = s0 >> 31 (mantem o bit de sinal)
        beq $t1, $0, positivo   #if(t1==0){ goto positivo; }
        sub $s0, $0, $s0        #s0 = 0 - s0
positivo:
        sw $s0, 0($t0)          #MEM[t0] = s0


#FIM

```

Line: 29 Column: 1 Show Line Numbers

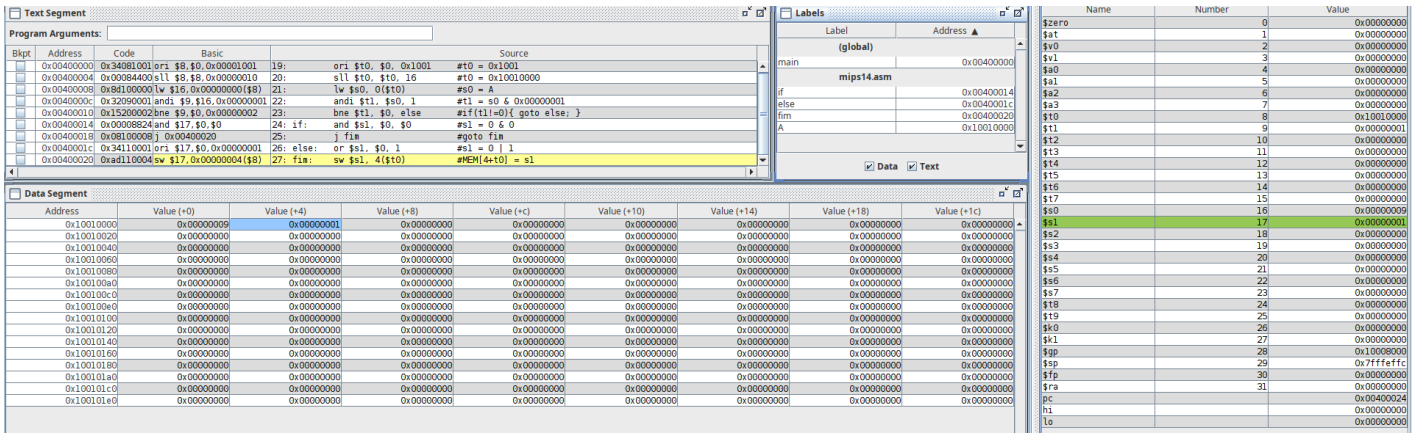


14)

mips13.asm | mips14.asm

```
1  #       Programa 14
2  #       Escreva um programa que leia um valor da memoria e identifique se é par ou não
3  #       Devera ser escrito na segunda posição da memoria:
4  #       0 para par ou 1 para impar
5
6  #INICIO
7
8  .data
9  A: .word 9
10
11 .text
12 .globl main
13 main:
14         #t0 -> first address
15
16         #A -> $s0
17         #isEven -> $s1
18
19         ori $t0, $0, 0x1001      #t0 = 0x1001
20         sll $t0, $t0, 16         #t0 = 0x10010000
21         lw $s0, 0($t0)           #s0 = A
22         andi $t1, $s0, 1         #t1 = s0 & 0x00000001
23         bne $t1, $0, else        #if(t1!=0){ goto else; }
24 if:     and $s1, $0, $0          #s1 = 0 & 0
25         j fim                    #goto fim
26 else:   or $s1, $0, 1            #s1 = 0 | 1
27 fim:    sw $s1, 4($t0)           #MEM[4+t0] = s1
28
29 #FIM
30
```

Line: 30 Column: 1 ☑ Show Line Numbers

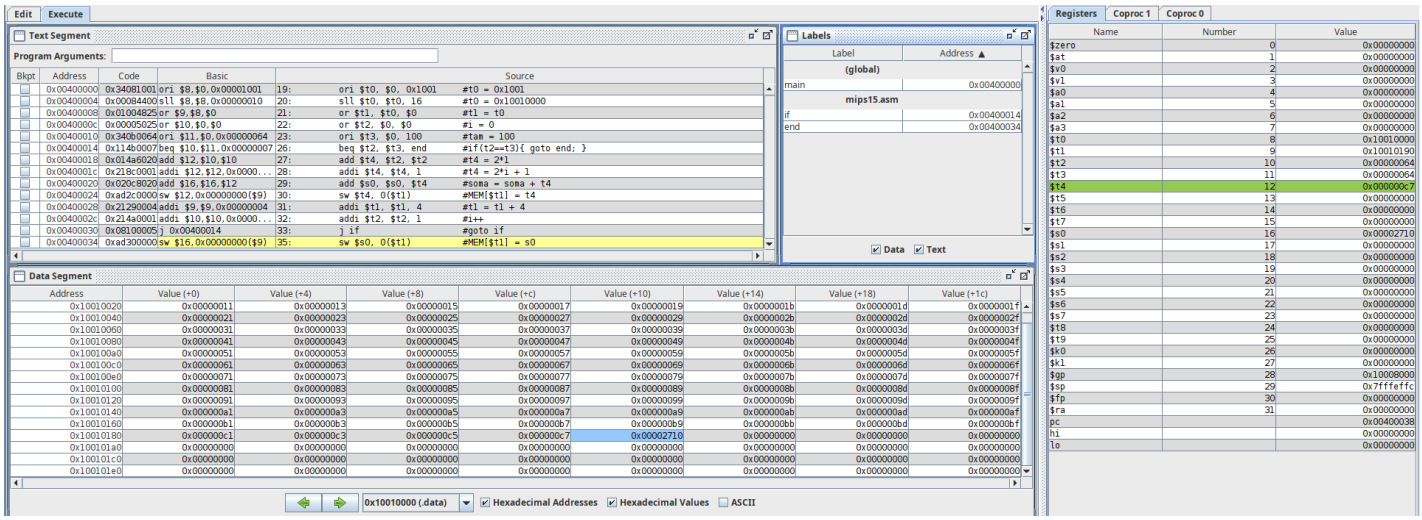


15)

```
#       Programa 15
#       Crie um vetor de 100 elementos, onde v[i] = 2*1+1
#       Apos a ultima posicao, escrever a soma de todos os valores armazenados no vetor

#INICIO

.data
.text
.globl main
main:
        #t0 -> first address
        #t1 -> offset

        #v[i] -> $t4
        #soma -> s1
        #i -> $t2   (0)
        #tam -> $t3 (100)

        ori $t0, $0, 0x1001     #t0 = 0x1001
        sll $t0, $t0, 16        #t0 = 0x10010000
        or $t1, $t0, $0         #t1 = t0
        or $t2, $0, $0          #i = 0
        ori $t3, $0, 100        #tam = 100

if:
        beq $t2, $t3, end       #if(t2==t3){ goto end; }
        add $t4, $t2, $t2       #t4 = 2*1
        addi $t4, $t4, 1        #t4 = 2*i + 1
        add $s0, $s0, $t4       #soma = soma + t4
        sw $t4, 0($t1)          #MEM[$t1] = t4
        addi $t1, $t1, 4        #t1 = t1 + 4
        addi $t2, $t2, 1        #i++
        j if                    #goto if
end:
        sw $s0, 0($t1)          #MEM[$t1] = s0

#FIM
```

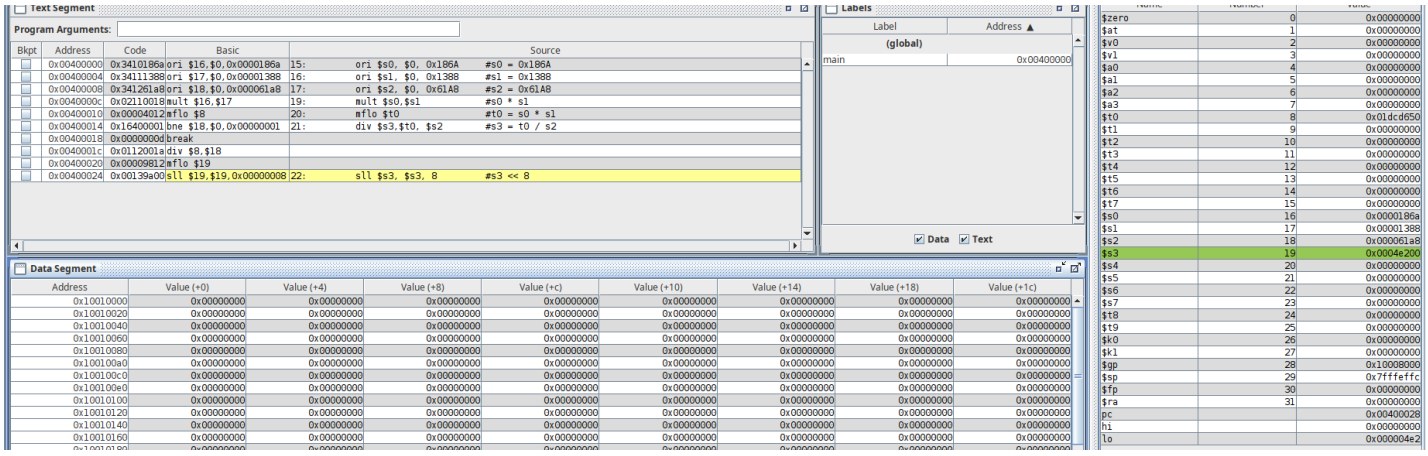
Line: 35 Column: 32 Show Line Numbers

Mars Messages | Run I/O



16)

```
1  #       Programa 16
2  #       Avalie a expressao (x*y)/z
3  #       Use x=1600000(0x186A00), y=80000(0x13880) e z = 400000(0x61A80), inicialize nos regs
4
5  #INICIO
6
7  .data
8  .text
9  .globl main
10 main:
11         #x -> s0
12         #y -> s1
13         #z -> s2
14
15         ori $s0, $0, 0x186A      #s0 = 0x186A
16         ori $s1, $0, 0x1388      #s1 = 0x1388
17         ori $s2, $0, 0x61A8      #s2 = 0x61A8
18
19         mult $s0,$s1             #s0 * s1
20         mflo $t0                 #t0 = s0 * s1
21         div $s3,$t0, $s2         #s3 = t0 / s2
22         sll $s3, $s3, 8          #s3 << 8
23
24 #FIM
25
```

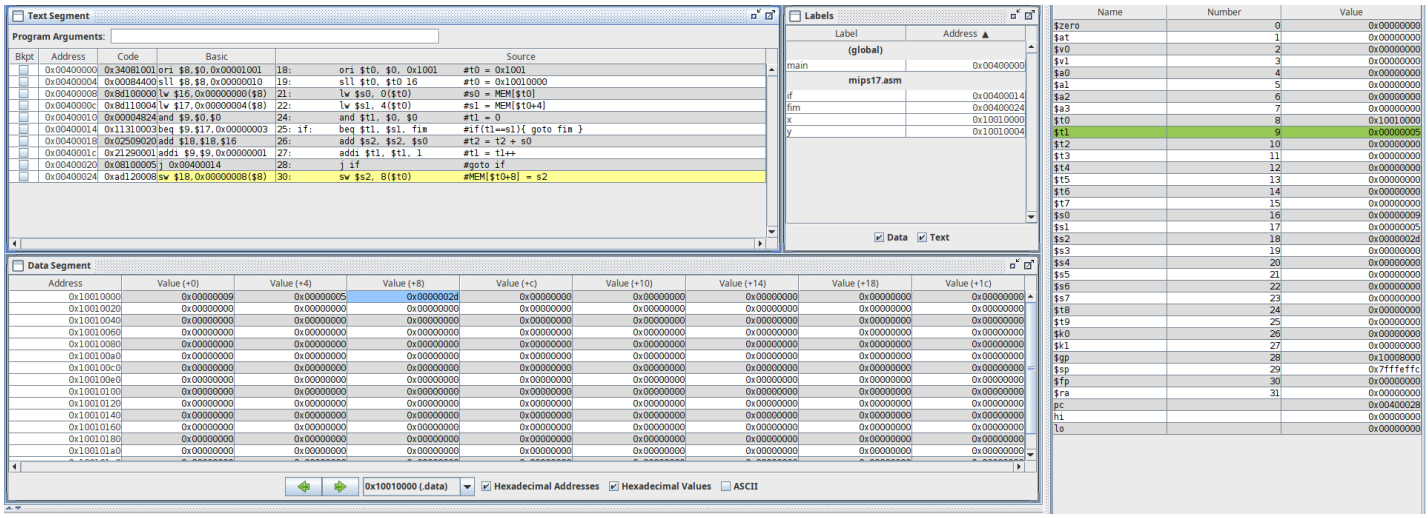


17)

mips17.asm

```
1   #       Programa 17
2   #       k = x * y
3   #       x sera lido da primeira posicao, o y da segunda, o k sera resscrito na terceira
4
5   #INICIO
6
7   .data
8   x: .word 9
9   y: .word 5
10  .text
11  .globl main
12  main:
13          #t0 -> 0x10010000 (first position)
14          #x -> s0
15          #y -> s1
16          #k -> s2
17
18          ori $t0, $0, 0x1001     #t0 = 0x1001
19          sll $t0, $t0 16         #t0 = 0x10010000
20
21          lw $s0, 0($t0)          #s0 = MEM[$t0]
22          lw $s1, 4($t0)          #s1 = MEM[$t0+4]
23
24          and $t1, $0, $0         #t1 = 0
25  if:     beq $t1, $s1, fim       #if(t1==s1){ goto fim }
26          add $s2, $s2, $s0       #t2 = t2 + s0
27          addi $t1, $t1, 1        #t1 = t1++
28          j if                    #goto if
29  fim:
30          sw $s2, 8($t0)          #MEM[$t0+8] = s2
31
32  #FIM
33
```

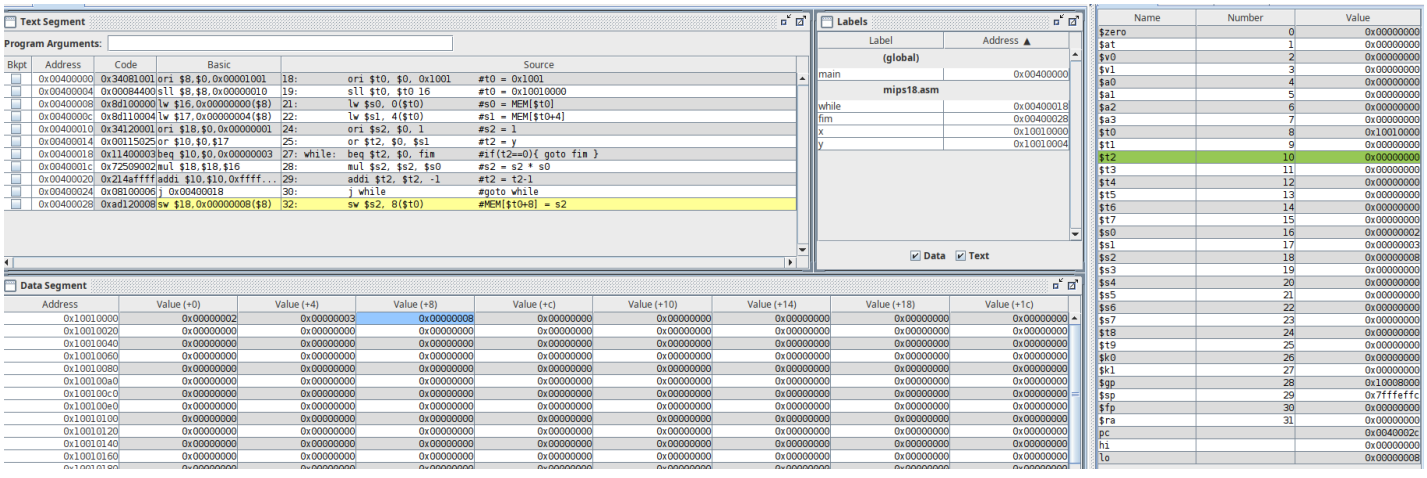
Line: 33 Column: 1 ☑ Show Line Numbers



18)

mips18.asm

```
#       Programa 18
#       k = x^y
#       x sera lido da primeira posicao, o y da segunda, o k sera resscrito na terceira

#INICIO

.data
x: .word 2
y: .word 3
.text
.globl main
main:
        #t0 -> 0x10010000 (first position)
        #x -> s0
        #y -> s1
        #k -> s2

        ori $t0, $0, 0x1001     #t0 = 0x1001
        sll $t0, $t0 16         #t0 = 0x10010000

        lw $s0, 0($t0)          #s0 = MEM[$t0]
        lw $s1, 4($t0)          #s1 = MEM[$t0+4]

        ori $s2, $0, 1          #s2 = 1
        or $t2, $0, $s1         #t2 = y

while:  beq $t2, $0, fim        #if(t2==0){ goto fim }
        mul $s2, $s2, $s0       #s2 = s2 * s0
        addi $t2, $t2, -1       #t2 = t2-1
        j while                 #goto while
fim:
        sw $s2, 8($t0)          #MEM[$t0+8] = s2

#FIM
```

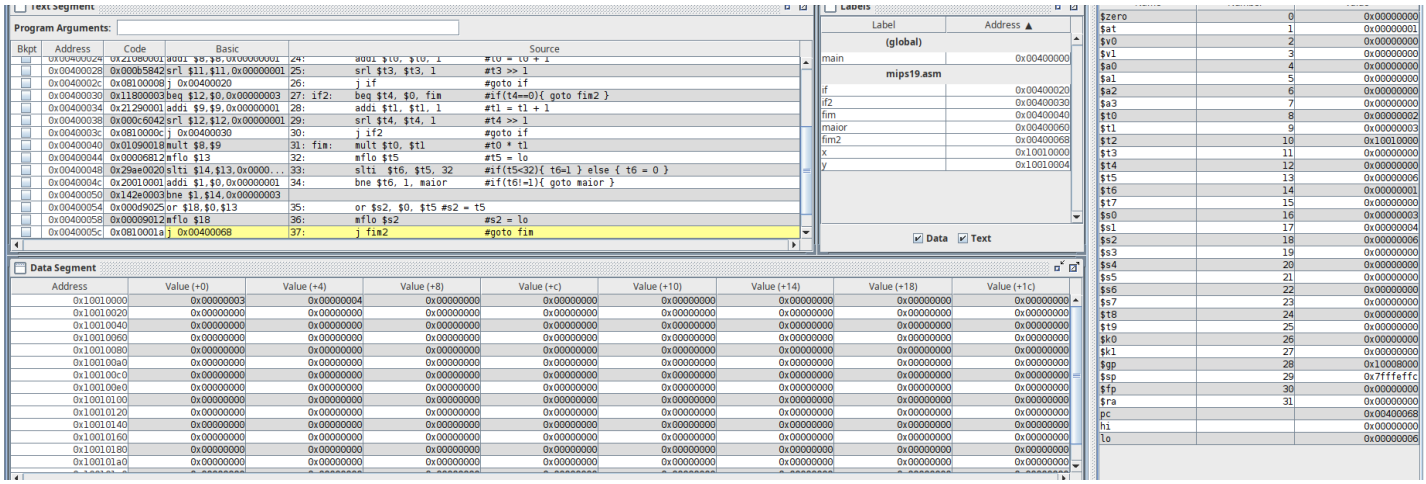


Responda

1) C. 64
2) B. hi e lo
3) A. mult
4) C. mflo $8
5) B. 32
6) A. lo
7) D. div
8) B. 0010 0110
9) A. Se o inteiro for unsigned, o shift o divide por 2. Se o inteiro for signed, o shift o divide por 2.
10) A.
    ori $3, $0, 3
    mult $8, $3
    mflo $9
    addi $9, $9, 7

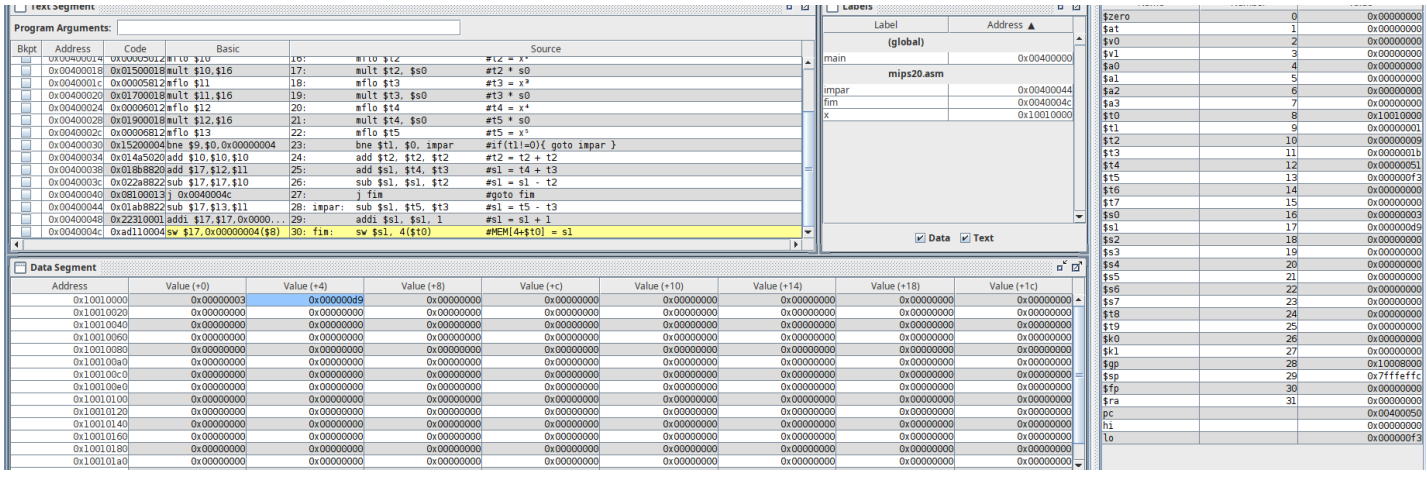
19)

```
1  #       Programa 19
2  #       Ler dois numeros da memoria, determinar qtd de bits
3  #       significantes de cada um, multiplicar ambos,
4  #INICIO
5  .data
6  x: .word 2
7  y: .word 3
8  .text
9  .globl main
10 main:
11         #t0 -> 0x10010000 (first position)
12         #x -> s0
13         #y -> s1
14         #k -> s2
15         ori $t2, $0, 0x1001     #t2 = 0x1001
16         sll $t2, $t2 16         #t2 = 0x10010000
17         lw $s0, 0($t2)          #s0 = MEM[$t0]
18         lw $s1, 4($t2)          #s1 = MEM[$t0+4]
19         or $t3, $s0, $0         #t3 = x
20         or $t4, $s1, $0         #t4 = y
21         or $t0, $0, $0          #t0 = 0 (contador de x)
22         or $t1, $0, $0          #t1 = 0 (contador de y)
23 if:     beq $t3, $0, if2        #if(t3==0){ goto fim1}
24         addi $t0, $t0, 1        #t0 = t0 + 1
25         srl $t3, $t3, 1         #t3 >> 1
26         j if                    #goto if
27 if2:    beq $t4, $0, fim        #if(t4==0){ goto fim2 }
28         addi $t1, $t1, 1        #t1 = t1 + 1
29         srl $t4, $t4, 1         #t4 >> 1
30         j if2                   #goto if
31 fim:    mult $t0, $t1           #t0 * t1
32         mflo $t5                #t5 = lo
33         slti  $t6, $t5, 32      #if(t5<32){ t6=1 } else { t6 = 0 }
34         bne $t6, 1, maior       #if(t6!=1){ goto maior }
35         ori $s2, $0, $t5        #s2 = t5
36         mflo $s2                #s2 = lo
37         j fim                   #goto fim
38 maior:  mfhi $s2                #s2 = hi
39         mflo $s3                #s3 = lo
40 fim:
41
42 #FIM
```



20)

```
1  #       Programa 20
2  #       Ler x da primeira pos da memoria, se x par y = x⁴ + x³ - 2x²
3  #       se x impar y = x⁵ - x³ + 1, escrever y na segunda pos da memoria
4  #INICIO
5  .data
6  x: .word 3
7  .text
8  .globl main
9  main:
10         ori $t0, $0, 0x1001     #t0 -> 0x10010000 (first position)
11         sll $t0, $t0, 16        #t0 << 16
12
13         lw $s0, 0($t0)          #s0 = MEM[$t0]
14         andi $t1, $s0, 1        #t1 = s0 & 1
15         mult $s0, $s0           #s0 * s0
16         mflo $t2                #t2 = x²
17         mult $t2, $s0           #t2 * s0
18         mflo $t3                #t3 = x³
19         mult $t3, $s0           #t3 * s0
20         mflo $t4                #t4 = x⁴
21         mult $t4, $s0           #t5 * s0
22         mflo $t5                #t5 = x⁵
23         bne $t1, $0, impar      #if(t1!=0){ goto impar }
24         add $t2, $t2, $t2       #t2 = t2 + t2
25         add $s1, $t4, $t3       #s1 = t4 + t3
26         sub $s1, $s1, $t2       #s1 = s1 - t2
27         j fim                   #goto fim
28 impar:  sub $s1, $t5, $t3       #s1 = t5 - t3
29         addi $s1, $s1, 1        #s1 = s1 + 1
30 fim:    sw $s1, 4($t0)          #MEM[4+$t0] = s1
31 #FIM
32
```



21)

```
1  #       Programa 21
2  #       Ler x da primeira pos da memoria, se x>0 y = x⁴ + x³ - 2x²
3  #       se x<=0 y = x⁵ - x³ + 1, escrever y na segunda pos da memoria
4  #INICIO
5  .data
6  x: .word -3
7  .text
8  .globl main
9  main:
10         ori $t0, $0, 0x1001      #t0 -> 0x10010000 (first position)
11         sll $t0, $t0, 16         #t0 << 16
12
13         lw $s0, 0($t0)           #s0 = MEM[$t0]
14         mult $s0, $s0            #s0 * s0
15         mflo $t2                 #t2 = x²
16         mult $t2, $s0            #t2 * s0
17         mflo $t3                 #t3 = x³
18         mult $t3, $s0            #t3 * s0
19         mflo $t4                 #t4 = x⁴
20         sle $t1, $s0, $0         #ifs0 <=0){ t1 = 1 } else { t1 = 0 }
21         bne $t1, $0, maior       #if(t1!=0){ goto impar }
22         addi $s1, $t3, 1         #s1 = t3 + 1
23         j fim                    #goto fim
24 maior:  addi $s1, $t4, -1        #s1 = t4 - 1
25 fim:    sw $s1, 4($t0)           #MEM[4+$t0] = s1
26
27 #FIM
28
```