Relatório 09 - Laboratório de Arquitetura de Computadores

Luiz Junio Veloso Dos Santos - Matricula: 624037

1 de maio de 2019

1. Digite, compile e observe o programa a seguir:

```
# x mapeado em $s1
.text
.globl teste
teste:

    addi $s1,$zero,1 # x = 1
    lw $s1, 0($t0)

.data
x1: .word 15
x2: .word 25
x3: .word 13
x4: .word 17
```

programa9.asm

(a) Quando o programa é carregado, em quais posições de memória os dados foram armazenados?

R: 0x10010000 0x10010004 0x10010008 0x1001000c

(b) Complete o programa anterior de maneira a ler os valores armazenados em x1, x2, x3 e x4 em registradores ($s1, $s2, $s3, $s4)

R:

```
# x mapeado em $s1
.text
.globl teste
teste:
    addi $s1,$zero,1 # x = 1

    addi $t0,$zero,0x1001
    sll  $t0,$t0,16 # t0 = primeiro endereco de dados

    lw $s1,0($t0) # s1 = mem[0]
    lw $s2,4($t0) # s2 = mem[1]
    lw $s3,8($t0) # s3 = mem[2]
    lw $s4,12($t0) # s4 = mem[3]

.data
x1: .word 15
x2: .word 25
x3: .word 13
x4: .word 17
```

programa9b.asm

(c) Crie mais uma variável denominada soma e atribua um valor inicial de -1. A variável soma deverá estar na posição seguinte a x4. Escrever um programa que leia todos os números, calcule e substitua o valor da variável soma por este valor.

R:

```
# x - $s1
# y - $s2
# z - $s3
# w - $s4

.text
.globl teste
teste:
    addi $t0,$zero,0x1001
    sll  $t0,$t0,16 # t0 = primeiro endereco de dados

    lw $s1,0($t0) # x = mem[0]
    lw $s2,4($t0) # y = mem[1]
    lw $s3,8($t0) # z = mem[2]
    lw $s4,12($t0) # z = mem[3]

    add $t1,$s1,$s2 # t1 = x + y
    add $t1,$t1,$s3 # t1 = t1 + z
    add $t1,$t1,$s4 # t1 = t1 + w

    sw $t1,16($t0) # soma = t1

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

programa9c.asm

2. Considere o seguinte programa: y = 127x – 65z + 1
   Faça um programa que calcule o valor de y conhecendo os valores de x e z. Os valores de x e z estão armazenados na memória e, na posição imediatamente a seguir, o valor de y deverá ser escrito.

   R:

```
# x - $s1
# z - $s2

.text
.globl teste
teste:
    addi $t0,$zero,0x1001
    sll  $t0,$t0,16 # t0 = primeiro endereco de dados

    lw $s1,0($t0) # x = mem[0]
    lw $s2,4($t0) # z = mem[1]

    sub $t1,$s1,$s2 # t1 = x - y
    addi $t2,$zero,100000
    addi $t1,$t1,1 # t3 = t3 + 1

    sw $t3,8($t0) # y = t1

.data
x: .word 100000
z: .word 200000
y: .word 0
```

programa10.asm

3. Considere o seguinte programa: y = x − z + 300000
   Faça um programa que calcule o valor de y conhecendo os valores de x e z. Os valores de x e z estão armazenados na memória e, na posição imediatamente a seguir, o valor de y deverá ser escrito.

   R:

```
# x - $s1
# z - $s2

.text
.globl teste
teste:
    addi $t0,$zero,0x1001
    sll  $t0,$t0,16 # t0 = primeiro endereco de dados

    lw $s1,0($t0) # x = mem[0]
    lw $s2,4($t0) # z = mem[1]

    sub $t1,$s1,$s2 # t1 = x - y

    addi $t2,$zero,18750 # t2 = 18750
    sll $t2,$t2,4 # t2 = 18750 * 2^4 = 300000

    add $t1,$t1,$t2 # t1 = t1 + t2

    sw $t1,8($t0) # y = t1

.data
x: .word 100000
z: .word 200000
y: .word 0
```

programa11.asm

4. Considere a seguinte situação:
   int ***x;
   onde x contem um ponteiro para um ponteiro para um ponteiro para um inteiro.
   Nessa situação, considere que a posição inicial de memória contenha o inteiro em questão. Coloque todos os outros valores em registradores, use os endereços de memória que quiser dentro do espaço de endereçamento do Mips.

   Resumo do problema:
   k = MEM [MEM [MEM [x]]].

   Crie um programa que crie a estrutura de dados acima, leia o valor de K, dobre e o reescreva conhecendo-se apenas o valor de x.

   R:

```
1  .text
2  .globl teste
3  teste:
4      addi $t0,$zero,0x1001
5      sll  $t0,$t0,16 # t0 = primeiro endereco de dados
6
7      lw $t1,12($t0) # t1 = mem[3]
8      lw $t2,0($t1)  # t2 = mem(t1)
9      lw $t3,0($t2)  # t3 = mem(t2)
10     lw $t4,0($t3)  # t4 = 42 =  mem(t3)
11
12     sll $t4,$t4,1 # t4 = t4 * 2
13     sw $t4, 0($t3) # mem[t3] = t4
14
15 .data
16 k: .word 42
17 x: .word k
18 y: .word x
19 z: .word y
```

programa12.asm

Figura 1: Programa 9

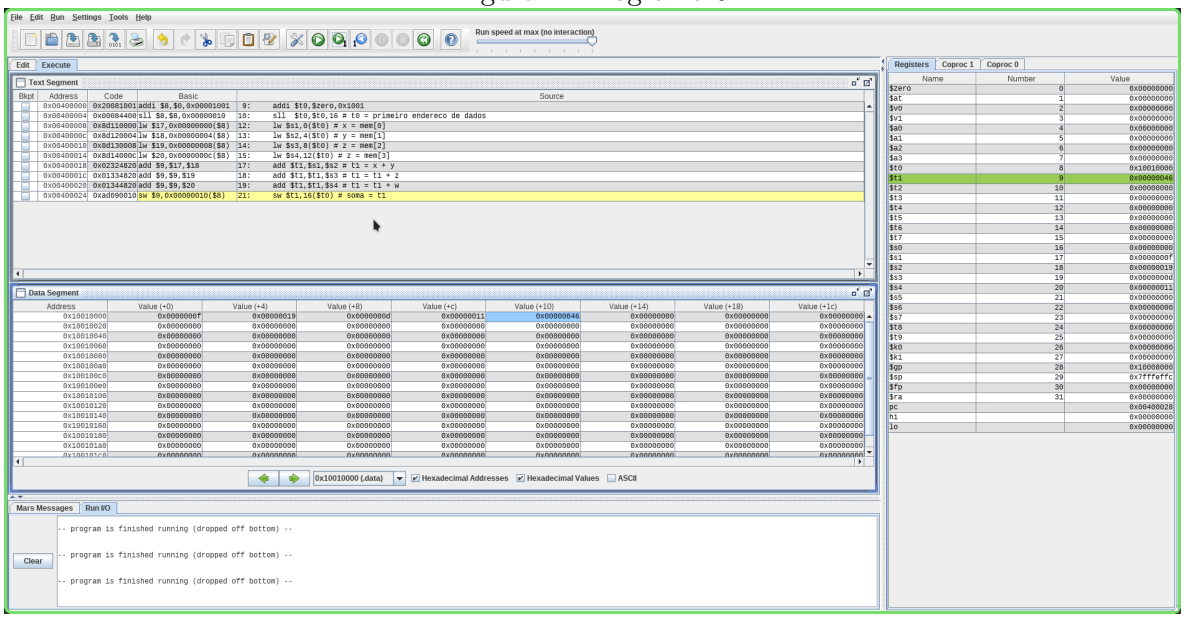

Figura 2: Programa 10

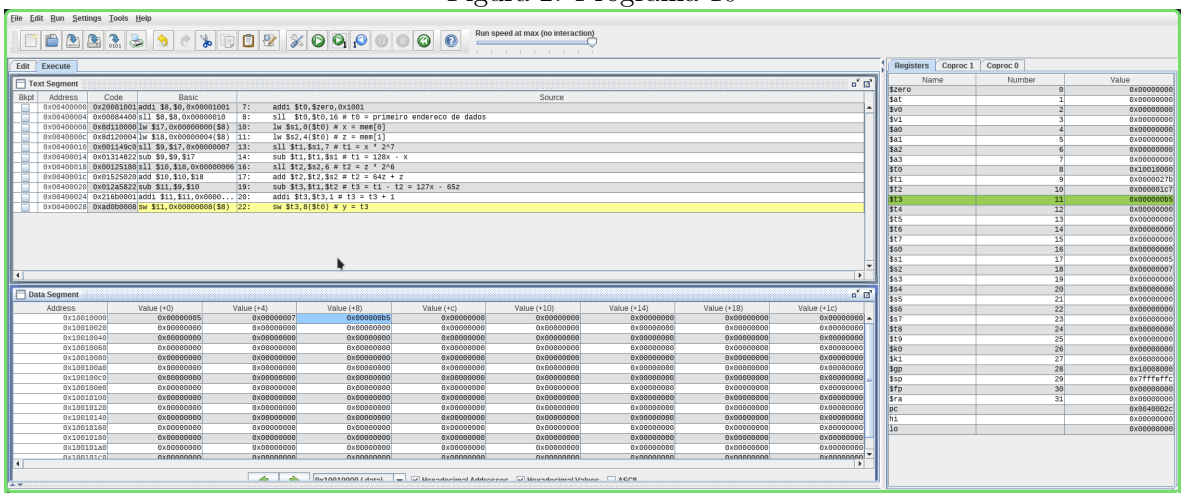

Figura 3: Programa 11

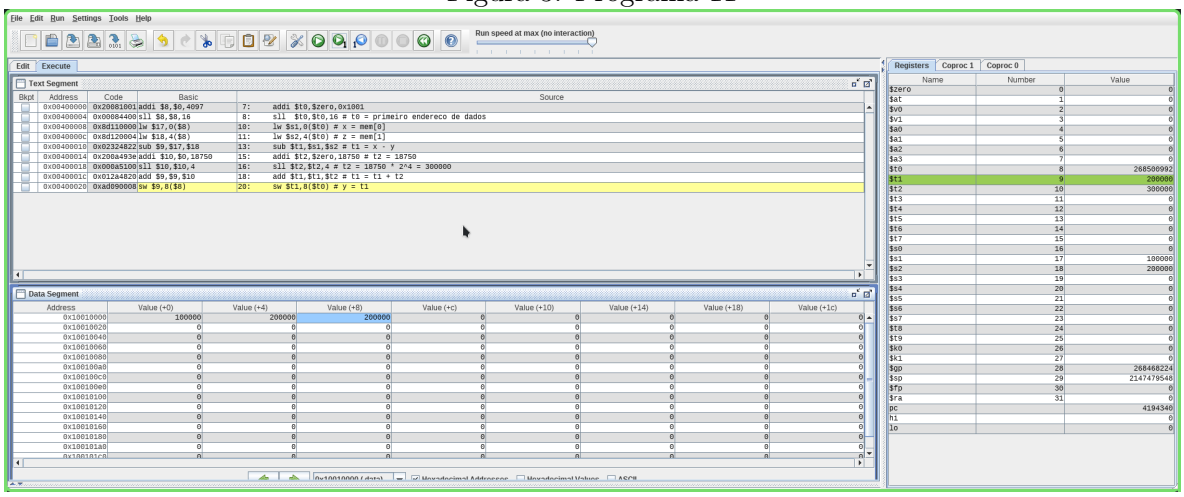

Figura 4: Programa 12