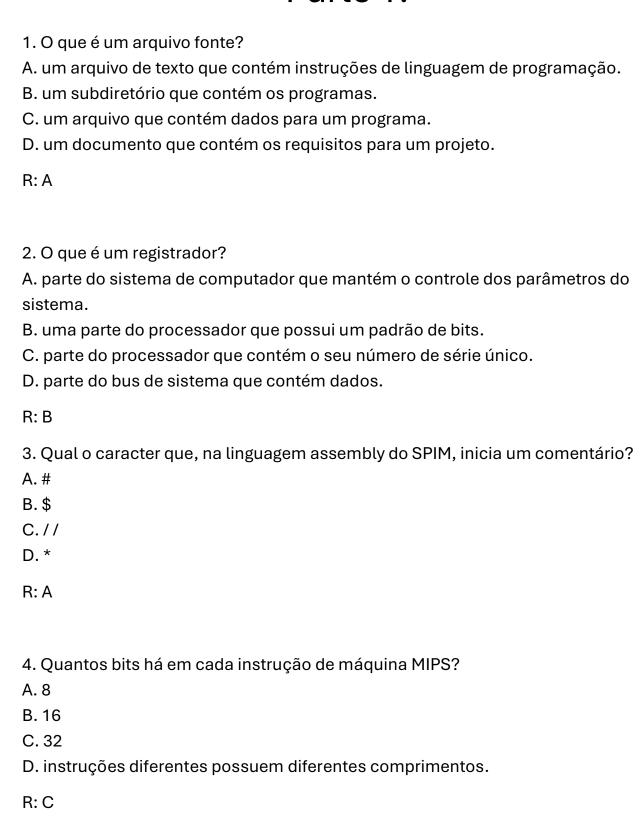
# Parte 1:



- 5. O que é o contador de programa?
- A. um registrador que mantém a conta do número de erros durante a execução de um programa.
- B. uma parte do processador que contém o endereço da primeira palavra de dados.
- C. uma variável na montadora que os números das linhas do arquivo de origem.
- D. parte do processador que contém o endereço da próxima instrução de máquina para ser obtida.

R: D

- 6. Ao executarmos uma instrução, quanto será adicionado ao contador de programa?
- A. 1
- B. 2
- C. 4
- D. 8
- R: C
- 7. O que é uma diretiva, tal como a diretiva .text?
- A. uma instrução em linguagem assembly que resulta em uma instrução em linguagem de máquina.
- B. uma das opções de menu do sistema SPIM.
- C. uma instrução em linguagem de máquina que faz com que uma operação sobre os dados ocorra.
- D. uma declaração que diz o montador algo sobre o que o programador quer, mas não corresponde diretamente a uma instrução de máquina.

R: D

- 8. O que é um endereço simbólico?
- A. um local de memória que contém dados simbólicos.
- B. um byte na memória que contém o endereço de dados.
- C. símbolo dado como argumento para uma directiva.

D. um nome usado no código-fonte em linguagem assembly para um local na memória.
R: D
9. Em qual endereço o simulador SPIM coloca a primeira instrução de máquina quando ele está sendo executado? A. 0x00000000 B. 0x00400000 C. 0x10000000 D. 0xFFFFFFF
R: B
10. Algumas instruções de máquina possuem uma constante como um dos operandos. Como é chamado tal operando? A. operando imediato B. operando embutido C. operando binário D. operando de máquina R: A
11. Como é chamada uma operação lógica executada entre bits de cada coluna dos operandos para produzir um bit de resultado para cada coluna?  A. operação lógica B. operação bitwise C. operação binária D. operação coluna R: B
<ul><li>12. Quando uma operação é de fato executada, como estão os operandos na ALU?</li><li>A. Pelo menos um operando deve ser de 32 bit.</li><li>B. Cada operando pode ser de qualquer tamanho.</li></ul>

C. Ambos operandos devem que vir de registros. D. Cada um dos registradores deve possuir 32 bit. R: D 13. Dezesseis bits de dados de uma instrução de ori são usados como um operando imediato. Durante execução, o que deve ser feito primeiro? A. Os dados são estendidos em zero à direita por 16 bits. B. Os dados são estendidos em zero à esquerda por 16 bits. C. Nada precisa ser feito. D. Apenas 16 bits são usados pelo outro operando. R: B 14. Qual das instruções seguintes armazenam no registrador \$5 um padrão de bits que representa positivo 48? A. ori \$5,\$0,0x48 B. ori \$5,\$5,0x48 C. ori \$5,\$0,48 D. ori \$0,\$5,0x48 R: C 15. A instrução de ori pode armazenar o complemento de dois de um número em um registrador? A. Não. B. Sim. R: A 16. Qual das instruções seguintes limpa todos os bits no registrador \$8 com exceção do byte de baixa ordem que fica inalterado?

A. ori \$8,\$8,0xFF B. ori \$8,\$0,0x00FF C. xori \$8,\$8,0xFF D. andi \$8,\$8,0xFF

R: D

- 17. Qual é o resultado de um ou exclusivo de padrão sobre ele mesmo?
- A. Todos os bits em zero.
- B. Todos os bits em um.
- C. O padrão original utilizado.
- D. O resultado é o contrário do original.

R: A

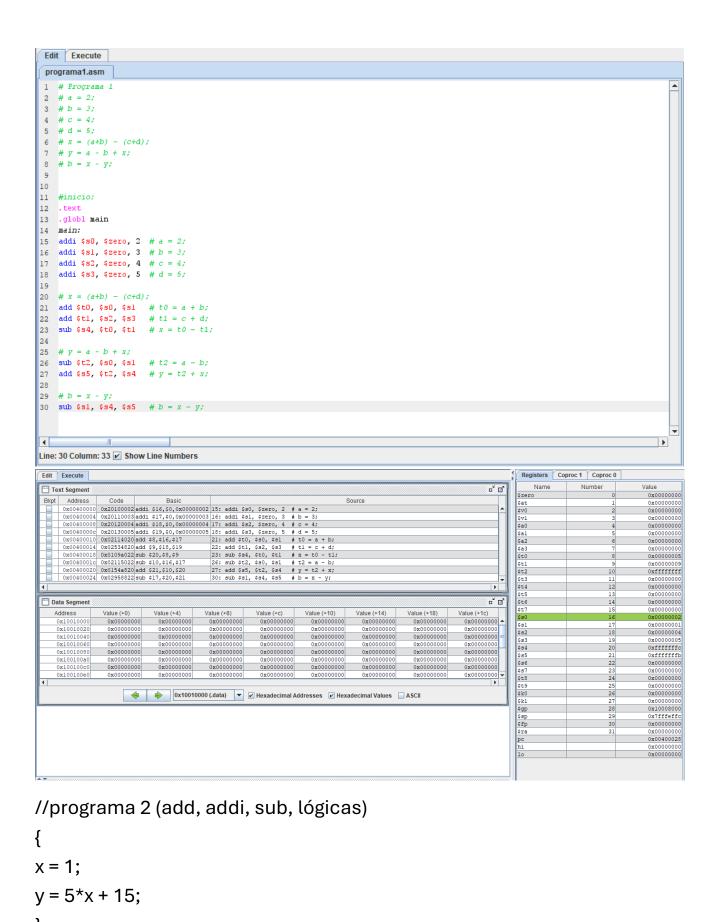
- 18. Todas as instruções de máquina têm os mesmos campos?
- A. Não. Diferentes de instruções de máquina possuem campos diferentes.
- B. Não. Cada instrução de máquina é completamente diferente de qualquer outra.
- C. Sim. Todas as instruções de máquina têm os mesmos campos na mesma ordem.
- D. Sim. Todas as instruções de máquina têm os mesmos campos, mas eles podem estar em ordens diferentes.

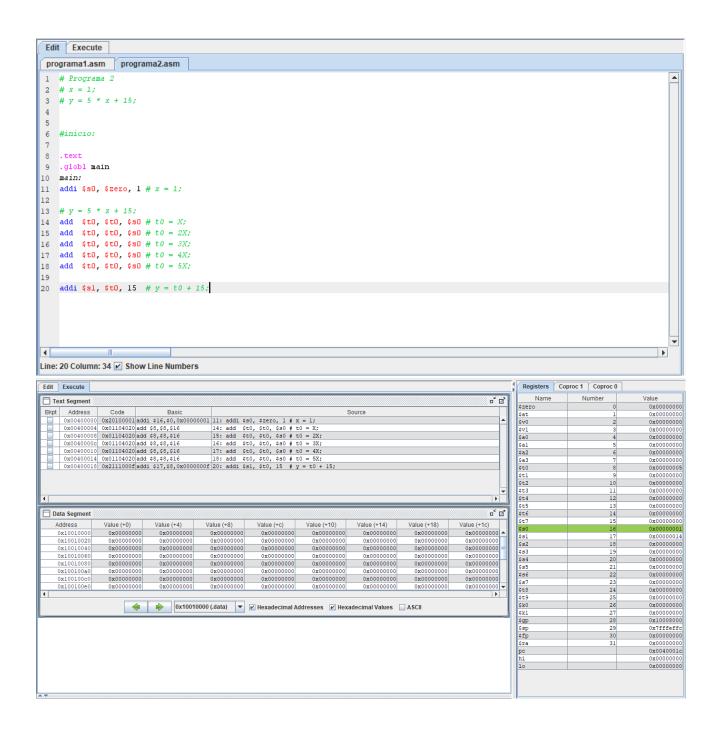
R: A

## Parte 2:

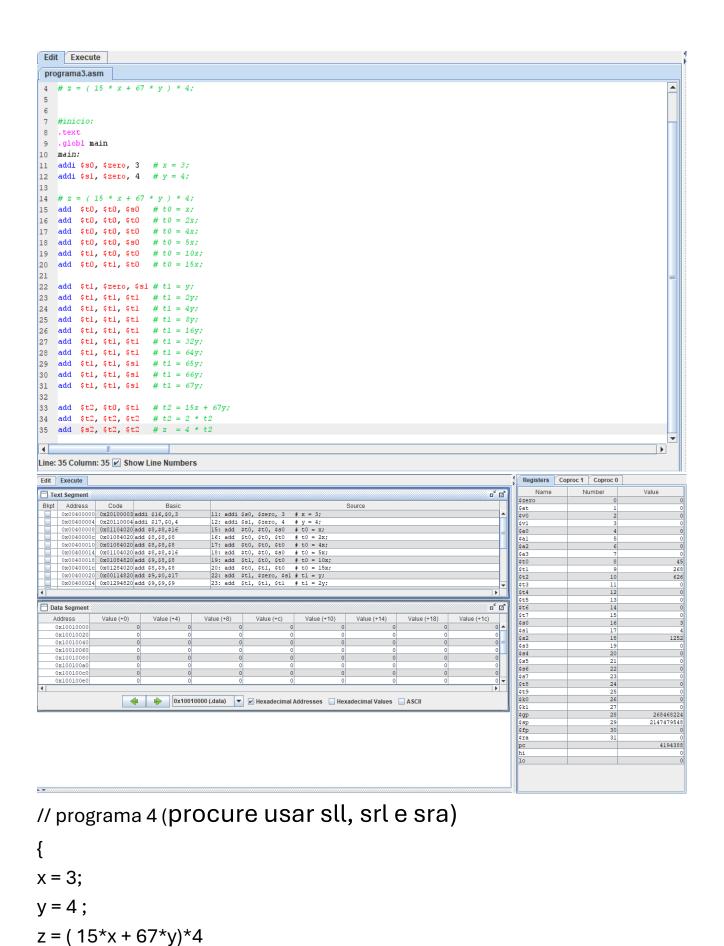
```
//programa 1 (add, addi, sub, lógicas) {
a =2;
```

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

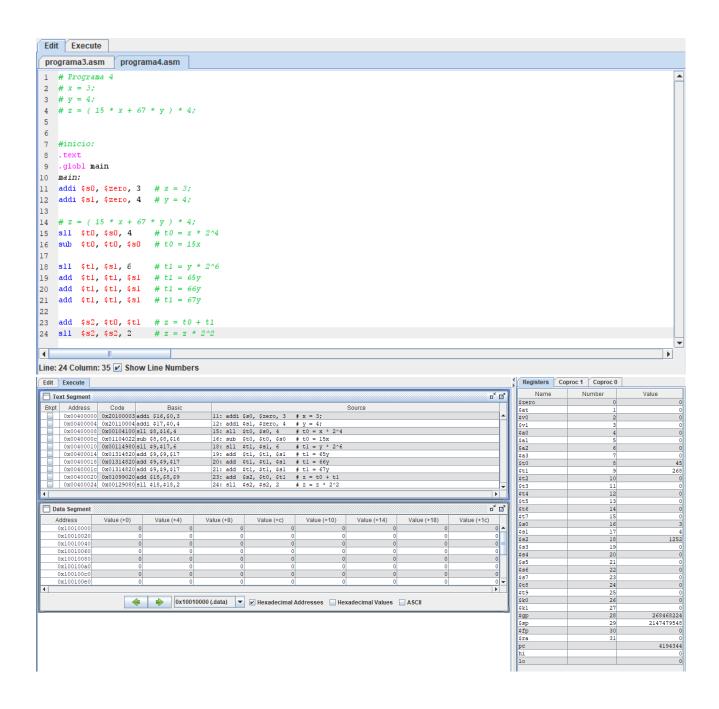




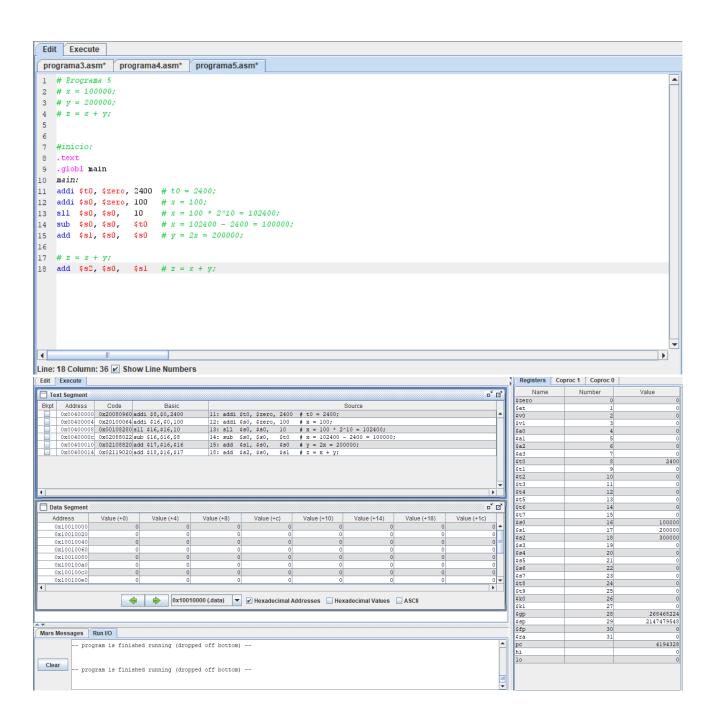
```
// programa 3 (add, addi, sub, lógicas) {
    x = 3;
    y = 4;
    z = (15*x + 67*y)*4
}
```



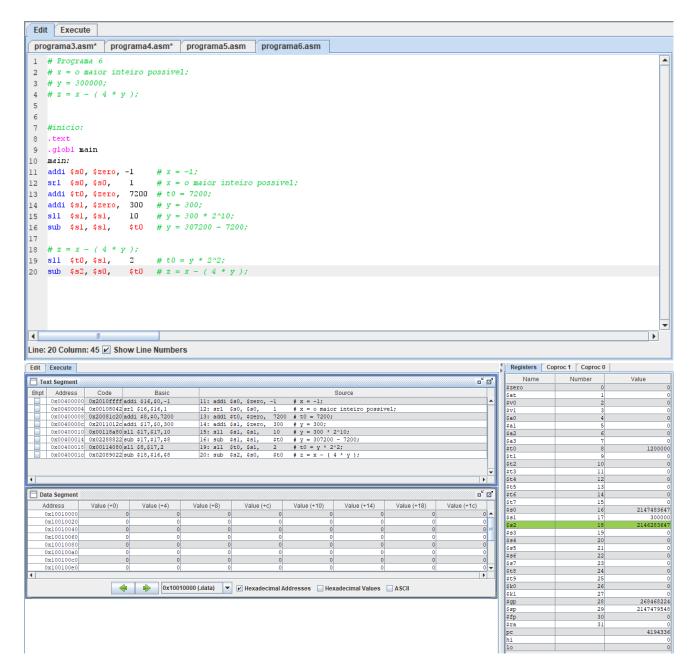
}



```
// programa 5
{
    x = 100000;
    y = 200000;
    z = x + y;
}
```



```
/ programa 6
{
  x = o maior inteiro possível;
  y = 300000;
  z = x - 4y
}
```



Considere a seguinte instrução iniciando um programa:

ori \$8, \$0, 0x01

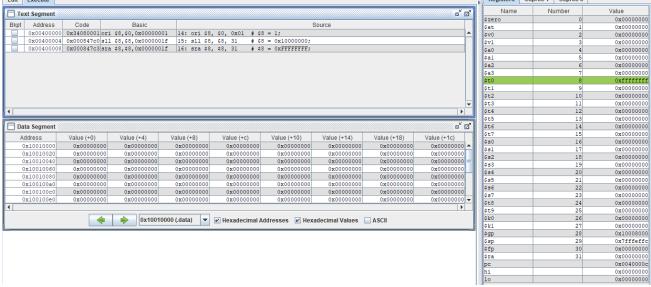
Usando apenas instruções reg-reg lógicas e/ou instruções de deslocamento (sll, srl e

sra), continuar o programa de forma que ao final, tenhamos o seguinte conteúdo no

registrador \$8:

\$8 = 0xFFFFFFF





Inicialmente escreva um programa que faça:

\$8 = 0x12345678.

A partir do registrador \$8 acima, usando apenas instruções lógicas (or, ori, and, andi,

xor, xori) e instruções de deslocamento (sll, srl e sra), você deverá obter os seguintes

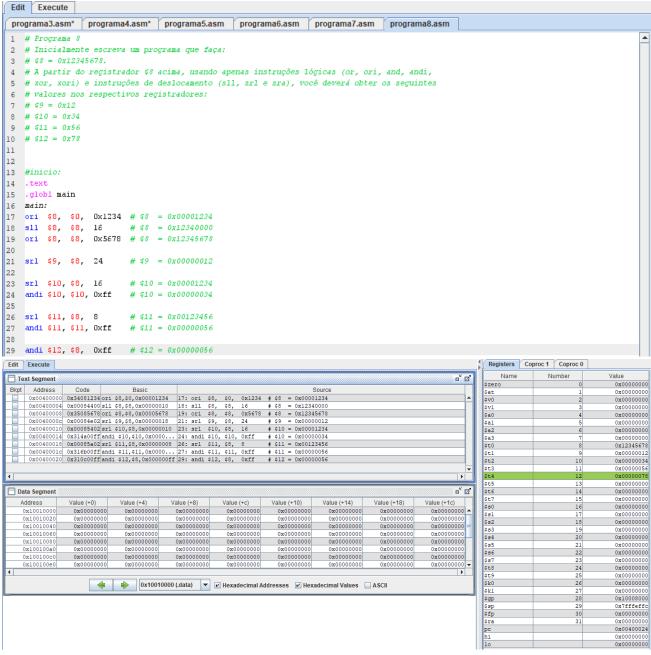
valores nos respectivos registradores:

\$9 = 0x12

\$10 = 0x34

\$11 = 0x56

\$12 = 0x78



Considere a memória inicial da seguinte forma:

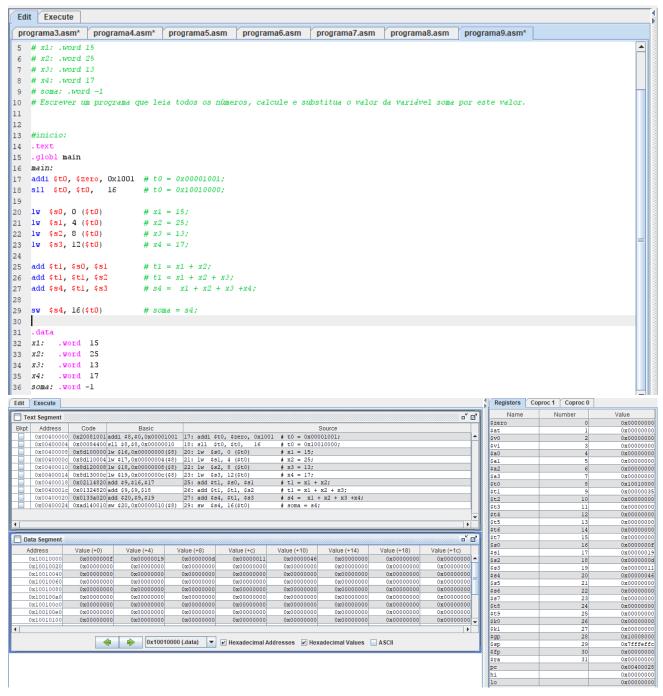
.text

.data

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

soma: .word -1

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



#### // programa 10

Considere o seguinte programa: y = 127x - 65z + 1

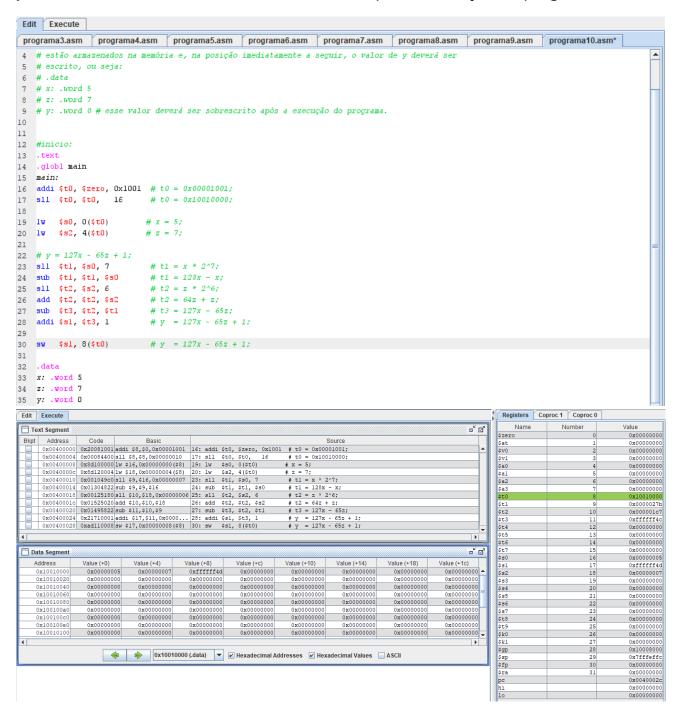
Faça um programa que calcule o valor de y conhecendo os valores de x e z. Os valores de x e z estão armazenados na memória e, na posição imediatamente a seguir, o valor de y deverá ser escrito, ou seja:

.data

x: .word 5

#### z: .word 7

y: .word 0 # esse valor deverá ser sobrescrito após a execução do programa.



## // programa 11

Considere o seguinte programa: y = x - z + 300000

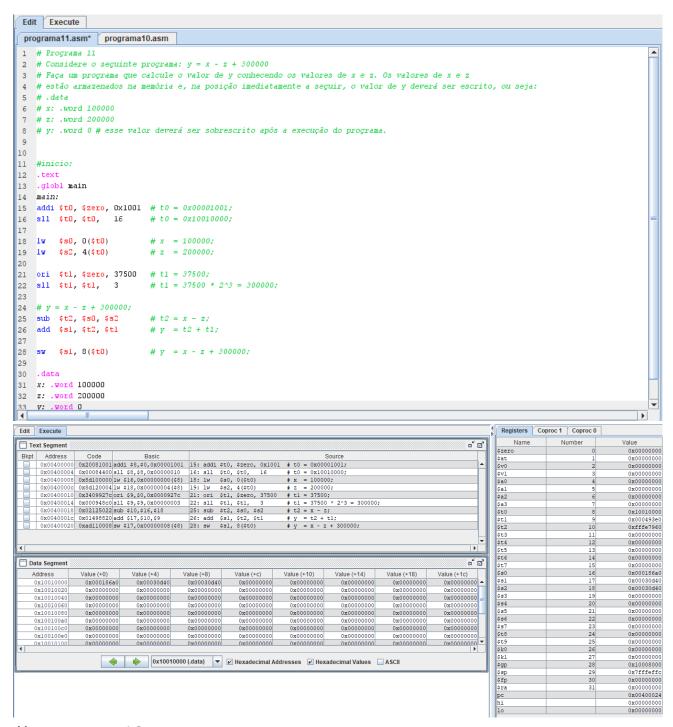
Faça um programa que calcule o valor de y conhecendo os valores de x e z. Os valores de x e z estão armazenados na memória e, na posição imediatamente a seguir, o valor de y deverá ser escrito, ou seja:

#### .data

x: .word 100000

#### z: .word 200000

y: .word 0 # esse valor deverá ser sobrescrito após a execução do programa.



#### // programa 12

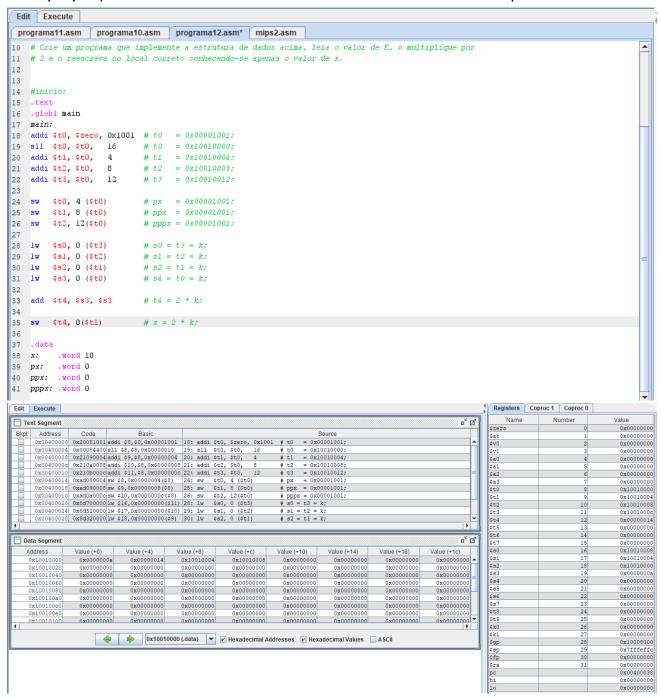
Considere a seguinte situação:

int \*\*\*x;

considere que a posição inicial de memória contenha o inteiro em questão.

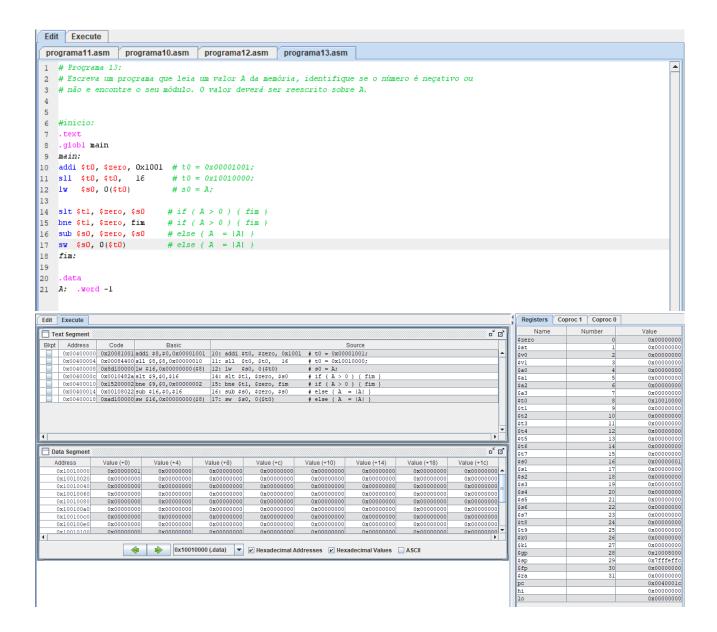
k = MEM [MEM [MEM [x]]].

Crie um programa que implemente a estrutura de dados acima, leia o valor de K, o multiplique por 2 e o reescreva no local correto conhecendo-se apenas o valor de x.



## // programa 13:

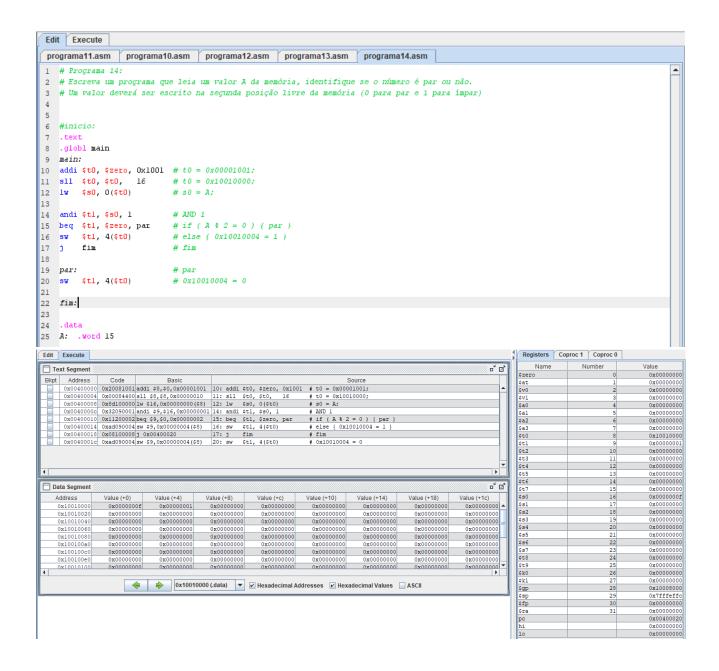
Escreva um programa que leia um valor A da memória, identifique se o número é negativo ou não e encontre o seu módulo. O valor deverá ser reescrito sobre A.



## // programa 14:

Escreva um programa que leia um valor A da memória, identifique se o número é par ou não.

Um valor deverá ser escrito na segunda posição livre da memória (0 para par e 1 para ímpar)

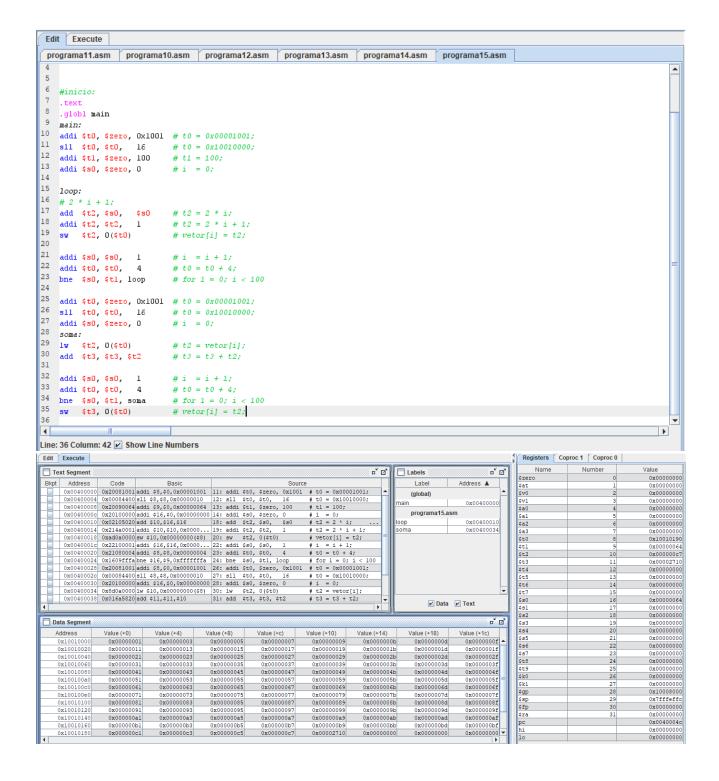


## // programa 15:

Escrever um programa que crie um vetor de 100 elementos na memória onde vetor[i] = 2\*i +1.

Após a última posição do vetor criado, escrever a soma de todos os valores armazenados do vetor.

Use o MARS para verificar a quantidade de instruções conforme o tipo (ULA, Desvios, Mem ou Outras)

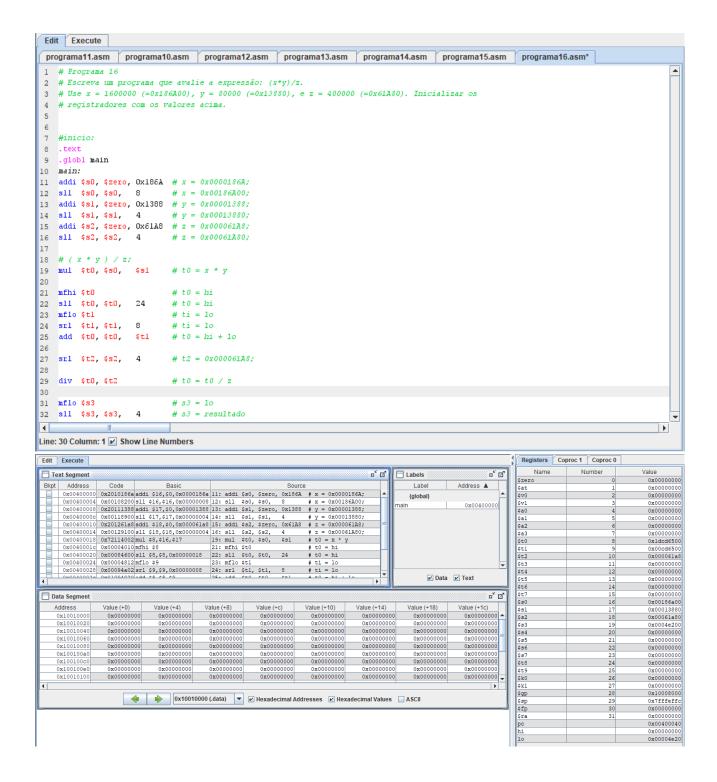


Escreva um programa que avalie a expressão: (x\*y)/z.

Use x = 1600000 (=0x186A00), y = 80000 (=0x13880), e z = 400000 (=0x61A80).

Inicializar os

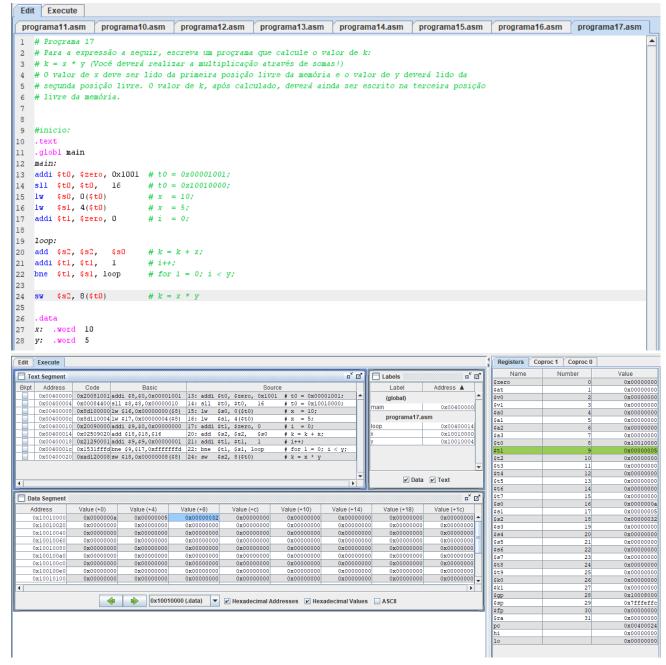
registradores com os valores acima.



Para a expressão a seguir, escreva um programa que calcule o valor de k:

k = x \* y (Você deverá realizar a multiplicação através de somas!)

O valor de x deve ser lido da primeira posição livre da memória e o valor de y deverá lido da segunda posição livre. O valor de k, após calculado, deverá ainda ser escrito na terceira posição livre da memória.



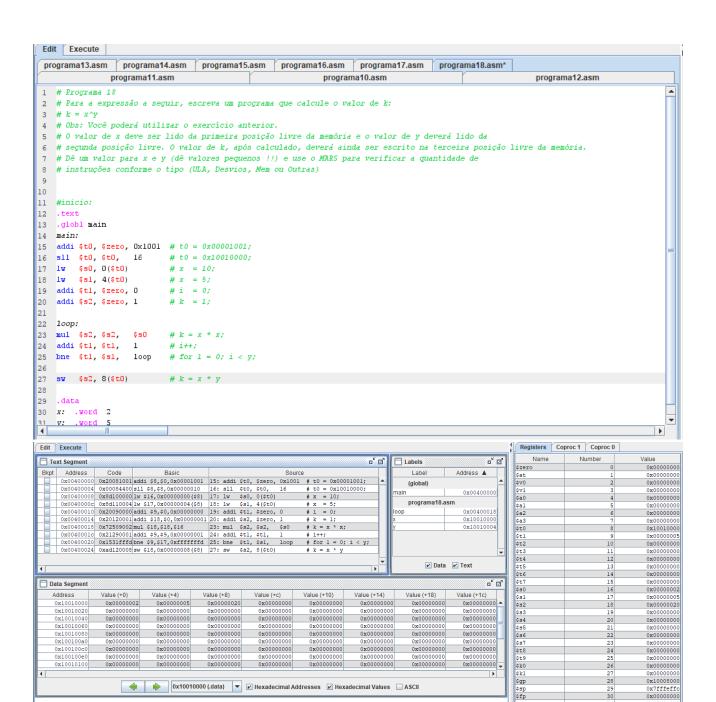
Para a expressão a seguir, escreva um programa que calcule o valor de k:

 $k = x_y$ 

Obs: Você poderá utilizar o exercício anterior.

O valor de x deve ser lido da primeira posição livre da memória e o valor de y deverá lido da segunda posição livre. O valor de k, após calculado, deverá ainda ser escrito na terceira posição livre da memória.

Dê um valor para x e y (dê valores pequenos !!) e use o MARS para verificar a quantidade de instruções conforme o tipo (ULA, Desvios, Mem ou Outras)



# Responda:

1. Se tivermos 2 inteiros, cada um com 32 bits, quantos bits podemos esperar para o produto?
o produto?
A. 16
B. 32
C. 64
D. 128
R: C
2. Quais os registradores que armazenam os resultados na multiplicação?
A. high e low
B. hi e lo
C. R0 e R1
D. \$0 e \$1
R: B
3. Qual a operação usada para multiplicar inteiros em comp. de dois?
A. mult
B. multu
C. multi
D. Mutt
R: A
4. Qual instrução move os bits menos significativos da multiplicação para o reg. 8?
A. move \$8,lo
B. mvlo \$8,lo
C. mflo \$8
D. addu \$8,\$0,lo

R: C
<ul> <li>5. Se tivermos dois inteiros, cada um com 32 bits, quantos bits deveremos estar preparados para receber no quociente?</li> <li>A. 16</li> <li>B. 32</li> <li>C. 64</li> <li>D. 128</li> </ul>
R: B
<ul> <li>6. Após a instrução div, qual registrador possui o quociente?</li> <li>A. lo</li> <li>B. hi</li> <li>C. high</li> <li>D. \$2</li> </ul>
R: A
<ul> <li>7. Qual a inst. Usada para dividir dois inteiros em comp. de dois?</li> <li>A. dv</li> <li>B. divide</li> <li>C. divu</li> <li>D. Div</li> </ul>
R: D
8. Faça um arithmetic shift right de dois no seguinte padrão de bits: 1001 1011 A. 1110 0110 B. 0010 0110 C. 1100 1101 D. 0011 0111
R: A

- 9. Qual o efeito de um arithmetic shift right de uma posição?
- A. Se o inteiro for unsigned, o shift o divide por 2. Se o inteiro for signed, o shift o divide por 2.
- B. Se o inteiro for unsigned, o shift o divide por 2. Se o inteiro for signed, o shift pode resultar em um valor errado.
- C. Se o inteiro for unsigned, o shift pode ocasionar um valor errado. Se o inteiro for signed, o shift o divide por 2.
- D. O shift multiplica o número por dois.

R: A

10. Qual sequencia de instruções avalia 3x+7, onde x é iniciado no reg. \$8 e o resultado armazenado em \$9?

```
A. ori $3,$0,3
mult $8,$3
mflo $9
addi $9,$9,7
B. ori $3,$0,3
mult $8,$3
addi $9,$8,7
C. ori $3,$0,3
mult $8,$3
mfhi $9
addi $9,$9,7
D. mult $8,3
mflo $9
addi $9,$9,7
```

R: A

## // programa 19

Escrever um programa que leia dois números da memória, a primeira e segunda posições respectivamente (os coloque em \$s0 e \$s1) e determine a quantidade de bits significantes de cada um. Coloque as respostas em \$t0 e \$t1, a partir desse resultado faça a multiplicação. Caso o número de bits significantes de ambos seja

menor do que 32 a resposta deverá estar apenas em \$s2, caso contrário a resposta estará em \$s2 e \$s3 (LO e HI respectivamente).

```
9 #inicio:
10 .text
11 .globl main
12 main:
13 addi $t0, $zero, 0x1001 # t0 = 0x00001001;
14 sll $t0, $t0, 16 # t0 = 0x10010000;
15 lw $s0, 0($t0)
                                     # s0 = x;
16 lw $sl, 4($t0)
                                     # s1 = y;
17 lw $t4, 0($t0)
18 lw $t5, 4($t0)
                                     # t5 = y;
19 addi $t0, $zero, 0
                                     #i = 0;
20
21 significativox:
22 beq $t4, $zero, endx # while x /= 0;
23 srl $t4, $t4, 1
                                      # conta os bits sgnificativos;
24 addi $t0, $t0, 1
                                      # i++;
25 j significativox
26
27 endx:
28 addi $tl, $zero, O
                                     # j = 0;
29 significativov:
30 beq $t5, $zero, endy # while y /= 0;
31 srl $t5, $t5, 1
                                     # conta os bits scnificativos;
32 addi $t1. $t1. 1
                                     # 1++:
33 j significativoy
34
35 endy:
36 mult $s0, $s1 # s0 * s1;
37
38 addi $t2. $zero. 32
                                      # t2 = 32
39 slt $t3, $t2, $t0
 40 beq $t2, $zero, menor # Se $t3 == 0, x <= 32
41 slt $t3, $t2, $t1
42 beq $t2, $zero, menor # Se $t3 == 0, x <= 32
                                       # $2 = 10
43 mflo $s2
44 mfhi $s3
                                       # s3 = hi
 45 menor:
 46 mflo $s2
                                        \# s2 = 10
 47
 48 .data
49 x: .word 100000
50 y: .word 100000
Edit Execute
                                                                                                                                            Name
Text Segment
                                                                                                                               교 [조]
                                                                                                                                                         Number
                                                                                                                                                                           Value
 Source
13: addi $t0, $zero, 0x1001 # t0 = 0x00001001;
                                                 0x00400008 0x8d1000001 bt 517,4(58)
0x00400001 0x8d100001 bt 517,4(58)
0x00400010 0x8d0c00001 bt 512,0(58)
0x00400014 0x8d0c00001 bt 512,0(58)
0x00400018 0x20080000 addi 58,50,0
0x00400010 0x105000001 bt 512,50,3
0x00400010 0x00060602 st 512,512,512,1
0x00400024 0x20080001 addi 58,69,1
0x00400020 0x0006001 dt 58,69,1
0x00400020 0x200800001 addi 58,50,0
0x00400030 0x101000001 bt 513,60,3
0x00400030 0x0000001 bt 513,60,3
0x00400030 0x0000001 bt 513,60,3
                                                 16: lw $s1, 4($t0)
17: lw $t4, 0($t0)
                                                                            # s1 = y;
# t4 = x;
                                                 18: lw $t5, 4($t0)
19: addi $t0, $zero, 0
                                                                         # t5 = y;
# i = 0;
                                                 22: beq $t4, $zero, endx
23: srl $t4, $t4, 1
                                                                          # while x != 0;
# conta os bits sgnificativos;
                                                 24: addi $t0, $t0, 1
25: j significativox
                                                 # conta os bits sgnificativos;
# i++*
  0x00400034 0x000d6842 srl $13,$13,1
                                                31: srl $t5, $t5, 1
32: addi $t1 $t1 1
                                                                                                                                ு் ⊠ீ
                                                                                          Value (+14)
                                                                                                                                        $t9
$k0
                                                                                                                                       $kl
$gp
                                                                                                                                                                             268468224
                                                                                                                                       $sp
$fp
                                                                                                                                                                            4194404
 Mars Messages Run I/O
                                                                                                                                                                            1410065408
```