Exercício Prático 6:

Felipe Rivetti Mizher – 821811

Parte 1:

1. O que é um arquivo fonte?

A. um arquivo de texto que contém instruções de linguagem de programação.
B. um subdiretório que contém os programas.
C. um arquivo que contém dados para um programa.
D. um documento que contém os requisitos para um projeto.

2. O que é um registrador?

A. parte do sistema de computador que mantém o controle dos parâmetros do sistema.
B. uma parte do processador que possui um padrão de bits.
C. parte do processador que contém o seu número de série único.
D. parte do bus de sistema que contém dados.

3. Qual o caracter que, na linguagem assembly do SPIM, inicia um comentário?

A. #
B. $
C. / /
D. *

4. Quantos bits há em cada instrução de máquina MIPS?
A. 8
B. 16
C. 32
D. instruções diferentes possuem diferentes comprimentos.

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.

6. Ao executarmos uma instrução, quanto será adicionado ao contador de programa?
A. 1
B. 2
C. 4
D. 8

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.

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.

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. 0xFFFFFFFF

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

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

12. Quando uma operação é de fato executada, como estão os operandos na ALU?

A. Pelo menos um operando deve ser de 32 bit.
B. Cada operando pode ser de qualquer tamanho.
C. Ambos operandos devem que vir de registros.
D. Cada um dos registradores deve possuir 32 bit.

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.

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

15. A instrução de ori pode armazenar o complemento de dois de um número em um registrador?

A. Não.
B. Sim.

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

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.

18. Todas as instruções de máquina têm os mesmos campos?

A. Não. Diferentes de instruções de máquina possuem campos diferentes.

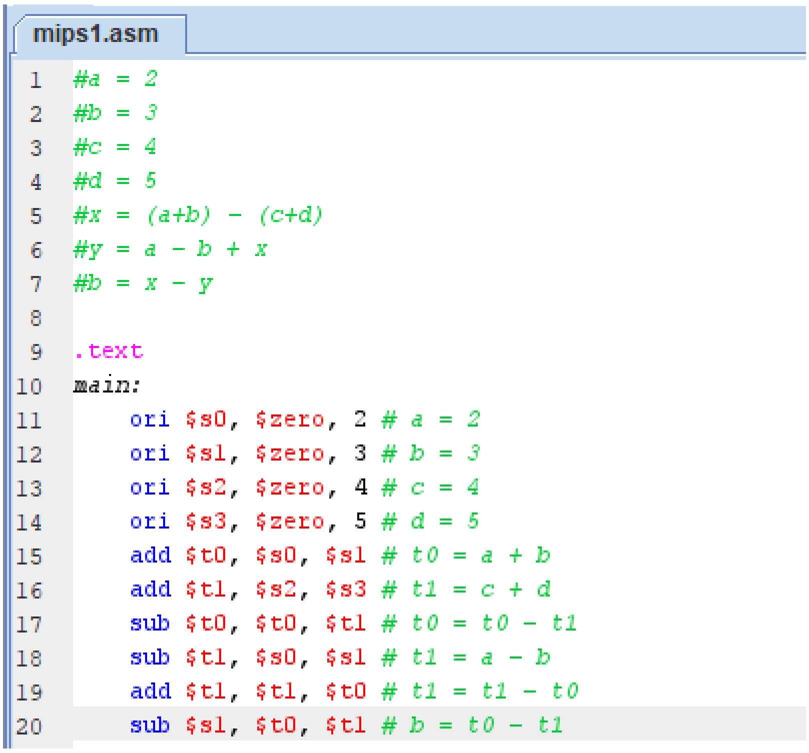
B. Não. Cada instrução de máquina é completamente diferente de qualquer outra.

C. Sim. Todas as instruções de máquina têm os mesmos campos na mesma ordem.

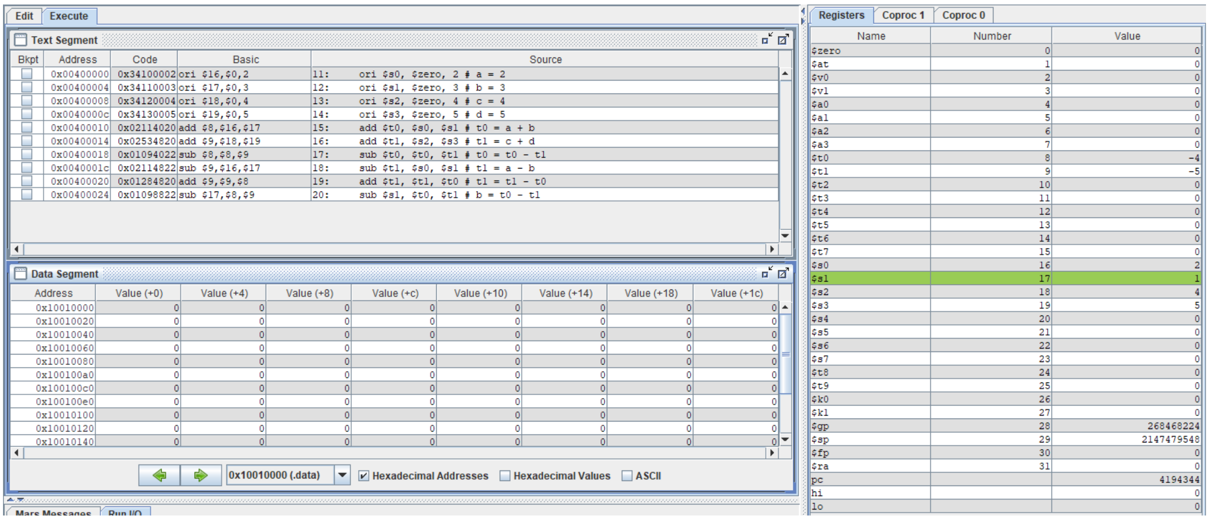
D. Sim. Todas as instruções de máquina têm os mesmos campos, mas eles podem estar em ordens diferentes

Parte 2:

Programa 1:



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

.text
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 $t0, $t0, $t1 # t0 = t0 - t1
    sub $t1, $s0, $s1 # t1 = a - b
    add $t1, $t1, $t0 # t1 = t1 - t0
    sub $s1, $t0, $t1 # b = t0 - t1
```



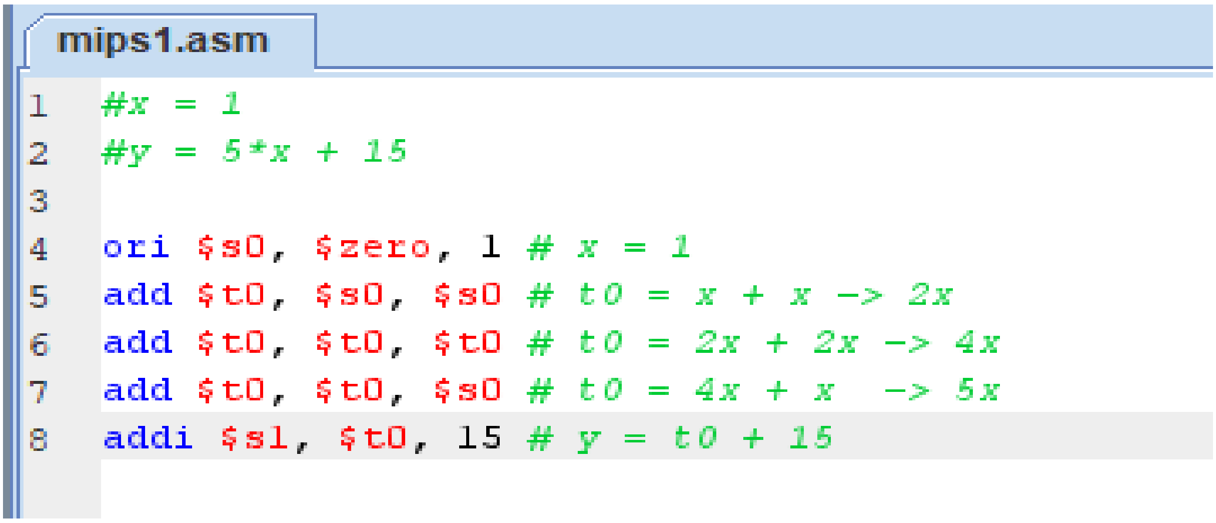
Bkpt	Address	Code	Basic	Source
☐	0x00400000	0x34100002	ori $16,$0,2	11: ori $s0, $zero, 2 # a = 2
☐	0x00400004	0x34110003	ori $17,$0,3	12: ori $s1, $zero, 3 # b = 3
☐	0x00400008	0x34120004	ori $18,$0,4	13: ori $s2, $zero, 4 # c = 4
☐	0x0040000c	0x34130005	ori $19,$0,5	14: ori $s3, $zero, 5 # d = 5
☐	0x00400010	0x02114020	add $8,$16,$17	15: add $t0, $s0, $s1 # t0 = a + b
☐	0x00400014	0x02534820	add $9,$18,$19	16: add $t1, $s2, $s3 # t1 = c + d
☐	0x00400018	0x01094022	sub $8,$8,$9	17: sub $t0, $t0, $t1 # t0 = t0 - t1
☐	0x0040001c	0x02114822	sub $9,$16,$17	18: sub $t1, $s0, $s1 # t1 = a - b
☐	0x00400020	0x01284820	add $9,$9,$8	19: add $t1, $t1, $t0 # t1 = t1 - t0
☐	0x00400024	0x01098822	sub $17,$8,$9	20: sub $s1, $t0, $t1 # b = t0 - t1

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0	0	0	0	0	0	0	0
0x10010020	0	0	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0
0x10010140	0	0	0	0	0	0	0	0

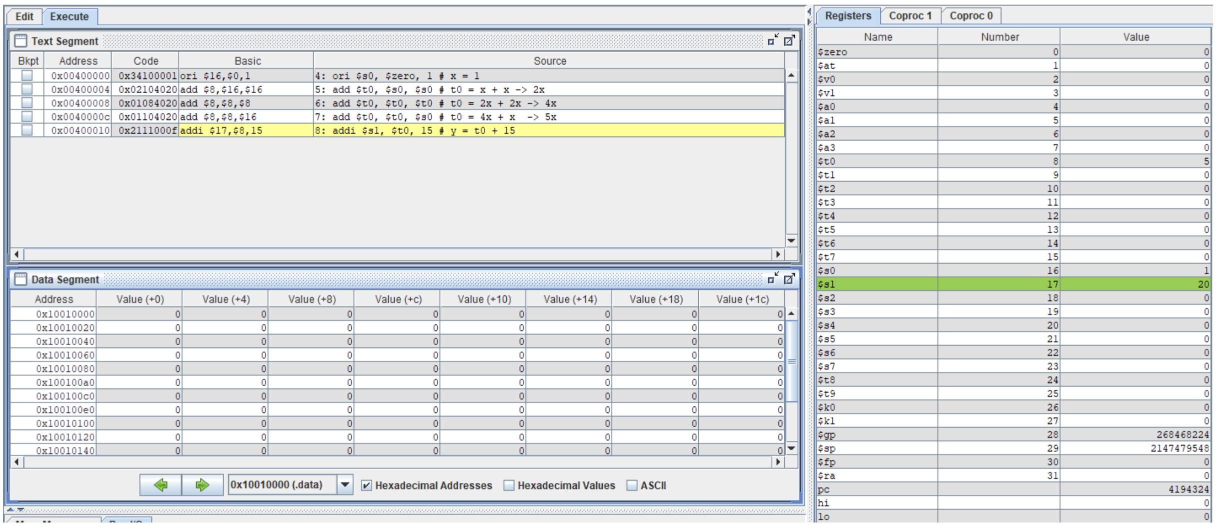
0x10010000 (.data) ☑ Hexadecimal Addresses ☐ Hexadecimal Values ☐ ASCII

Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	0
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	-4
$t1	9	-5
$t2	10	0
$t3	11	0
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	2
$s1	17	1
$s2	18	4
$s3	19	5
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0
pc		4194344
hi		0
lo		0

Programa 2:



```
#x = 1
#y = 5*x + 15

ori $s0, $zero, 1 # x = 1
add $t0, $s0, $s0 # t0 = x + x -> 2x
add $t0, $t0, $t0 # t0 = 2x + 2x -> 4x
add $t0, $t0, $s0 # t0 = 4x + x  -> 5x
addi $s1, $t0, 15 # y = t0 + 15
```



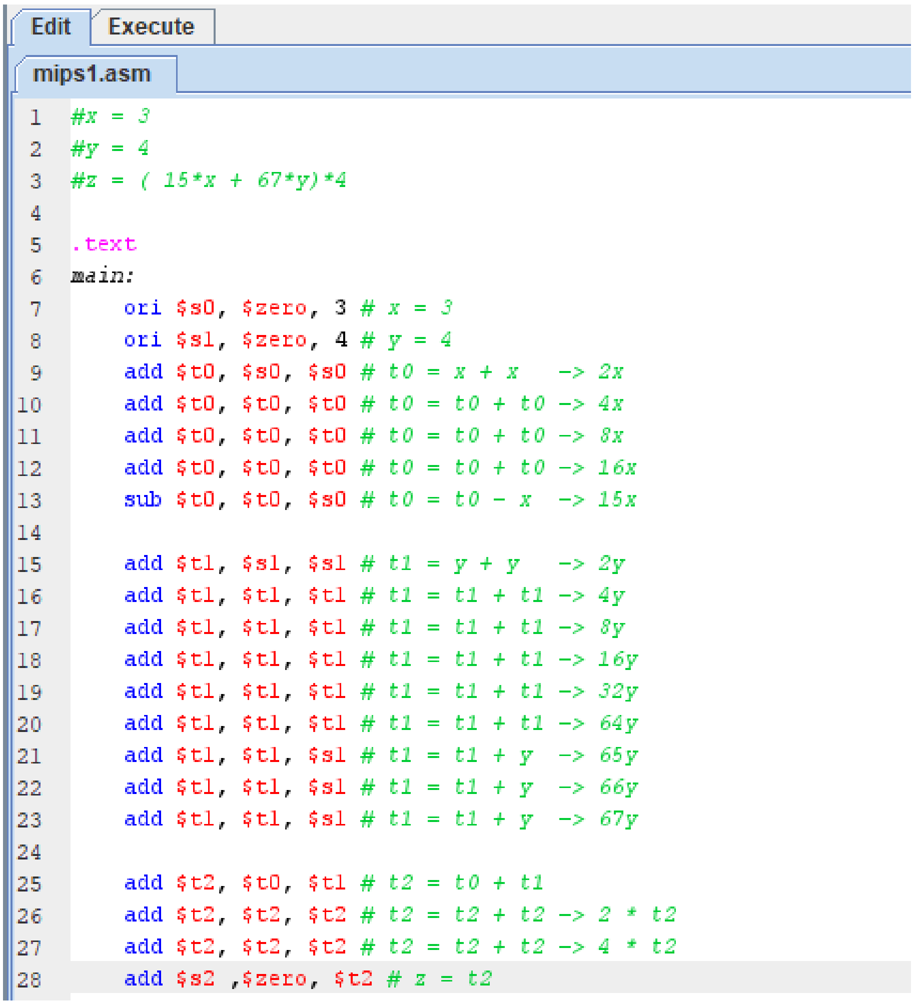
Bkpt	Address	Code	Basic	Source
☐	0x00400000	0x34100001	ori $16,$0,1	4: ori $s0, $zero, 1 # x = 1
☐	0x00400004	0x02104020	add $8,$16,$16	5: add $t0, $s0, $s0 # t0 = x + x -> 2x
☐	0x00400008	0x01084020	add $8,$8,$8	6: add $t0, $t0, $t0 # t0 = 2x + 2x -> 4x
☐	0x0040000c	0x01104020	add $8,$8,$16	7: add $t0, $t0, $s0 # t0 = 4x + x -> 5x
☐	0x00400010	0x2111000f	addi $17,$8,15	8: addi $s1, $t0, 15 # y = t0 + 15

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0	0	0	0	0	0	0	0
0x10010020	0	0	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0
0x10010140	0	0	0	0	0	0	0	0

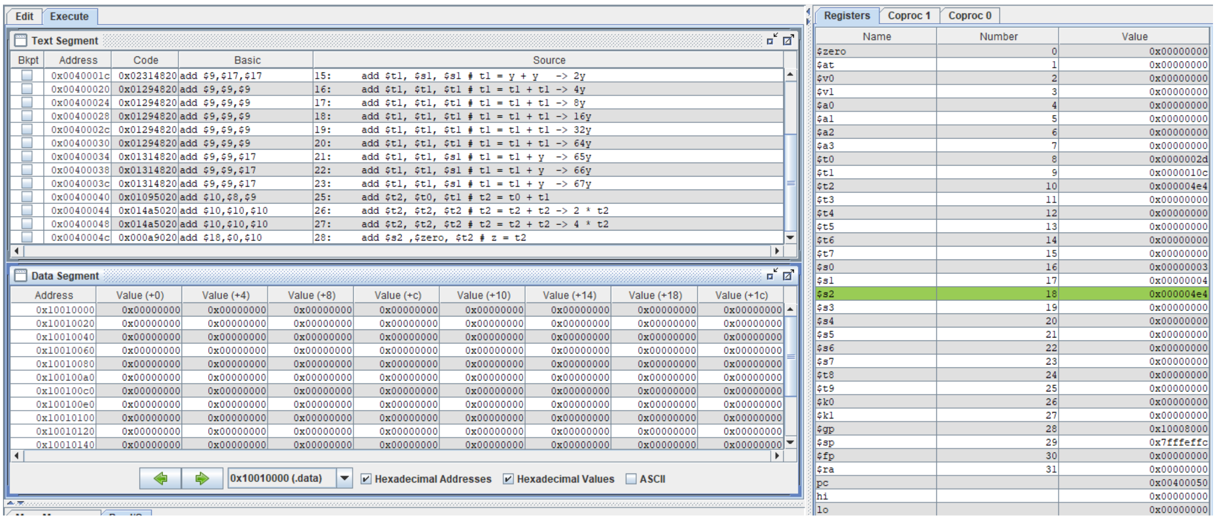
0x10010000 (.data) ☑ Hexadecimal Addresses ☐ Hexadecimal Values ☐ ASCII

Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	0
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	5
$t1	9	0
$t2	10	0
$t3	11	0
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	1
$s1	17	20
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0
pc		4194324
hi		0
lo		0

Programa 3:



```
#x = 3
#y = 4
#z = ( 15*x + 67*y)*4

.text
main:
    ori $s0, $zero, 3 # x = 3
    ori $s1, $zero, 4 # y = 4
    add $t0, $s0, $s0 # t0 = x + x   -> 2x
    add $t0, $t0, $t0 # t0 = t0 + t0 -> 4x
    add $t0, $t0, $t0 # t0 = t0 + t0 -> 8x
    add $t0, $t0, $t0 # t0 = t0 + t0 -> 16x
    sub $t0, $t0, $s0 # t0 = t0 - x  -> 15x

    add $t1, $s1, $s1 # t1 = y + y   -> 2y
    add $t1, $t1, $t1 # t1 = t1 + t1 -> 4y
    add $t1, $t1, $t1 # t1 = t1 + t1 -> 8y
    add $t1, $t1, $t1 # t1 = t1 + t1 -> 16y
    add $t1, $t1, $t1 # t1 = t1 + t1 -> 32y
    add $t1, $t1, $t1 # t1 = t1 + t1 -> 64y
    add $t1, $t1, $s1 # t1 = t1 + y  -> 65y
    add $t1, $t1, $s1 # t1 = t1 + y  -> 66y
    add $t1, $t1, $s1 # t1 = t1 + y  -> 67y

    add $t2, $t0, $t1 # t2 = t0 + t1
    add $t2, $t2, $t2 # t2 = t2 + t2 -> 2 * t2
    add $t2, $t2, $t2 # t2 = t2 + t2 -> 4 * t2
    add $s2 ,$zero, $t2 # z = t2
```



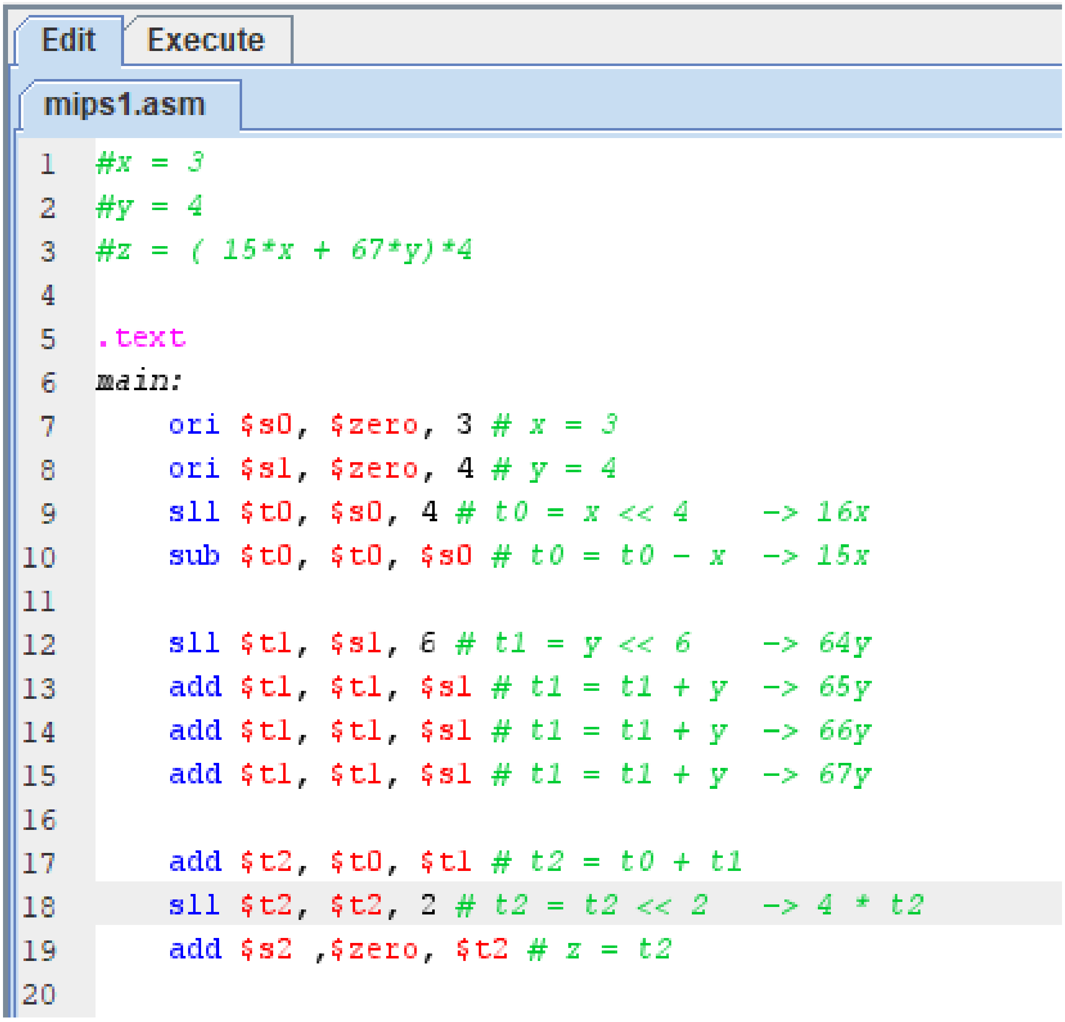
Bkpt	Address	Code	Basic	Source
□	0x0040001c	0x02314820	add $9,$17,$17	15: add $t1, $s1, $s1 # t1 = y + y -> 2y
□	0x00400020	0x01294820	add $9,$9,$9	16: add $t1, $t1, $t1 # t1 = t1 + t1 -> 4y
□	0x00400024	0x01294820	add $9,$9,$9	17: add $t1, $t1, $t1 # t1 = t1 + t1 -> 8y
□	0x00400028	0x01294820	add $9,$9,$9	18: add $t1, $t1, $t1 # t1 = t1 + t1 -> 16y
□	0x0040002c	0x01294820	add $9,$9,$9	19: add $t1, $t1, $t1 # t1 = t1 + t1 -> 32y
□	0x00400030	0x01294820	add $9,$9,$9	20: add $t1, $t1, $t1 # t1 = t1 + t1 -> 64y
□	0x00400034	0x01314820	add $9,$9,$17	21: add $t1, $t1, $s1 # t1 = t1 + y -> 65y
□	0x00400038	0x01314820	add $9,$9,$17	22: add $t1, $t1, $s1 # t1 = t1 + y -> 66y
□	0x0040003c	0x01314820	add $9,$9,$17	23: add $t1, $t1, $s1 # t1 = t1 + y -> 67y
□	0x00400040	0x01095020	add $10,$8,$9	25: add $t2, $t0, $t1 # t2 = t0 + t1
□	0x00400044	0x014a5020	add $10,$10,$10	26: add $t2, $t2, $t2 # t2 = t2 + t2 -> 2 * t2
□	0x00400048	0x014a5020	add $10,$10,$10	27: add $t2, $t2, $t2 # t2 = t2 + t2 -> 4 * t2
□	0x0040004c	0x000a9020	add $18,$0,$10	28: add $s2 ,$zero, $t2 # z = t2

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	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

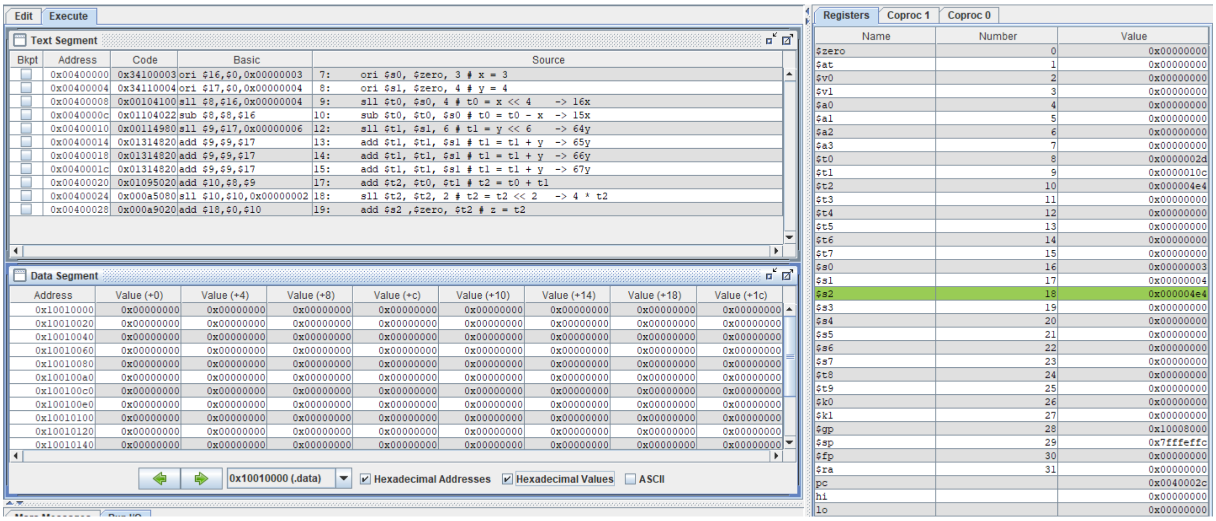
0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values □ ASCII

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x0000002d
$t1	9	0x0000010c
$t2	10	0x000004e4
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000003
$s1	17	0x00000004
$s2	18	0x000004e4
$s3	19	0x00000000
$s4	20	0x00000000
$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	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400050
hi		0x00000000
lo		0x00000000

Programa 4:



```
#x = 3
#y = 4
#z = ( 15*x + 67*y)*4

.text
main:
	ori $s0, $zero, 3 # x = 3
	ori $s1, $zero, 4 # y = 4
	sll $t0, $s0, 4 # t0 = x << 4    -> 16x
	sub $t0, $t0, $s0 # t0 = t0 - x  -> 15x

	sll $t1, $s1, 6 # t1 = y << 6    -> 64y
	add $t1, $t1, $s1 # t1 = t1 + y  -> 65y
	add $t1, $t1, $s1 # t1 = t1 + y  -> 66y
	add $t1, $t1, $s1 # t1 = t1 + y  -> 67y

	add $t2, $t0, $t1 # t2 = t0 + t1
	sll $t2, $t2, 2 # t2 = t2 << 2   -> 4 * t2
	add $s2 ,$zero, $t2 # z = t2

```



Text Segment

Bkpt	Address	Code	Basic	Source
□	0x00400000	0x34100003	ori $16,$0,0x00000003	7: ori $s0, $zero, 3 # x = 3
□	0x00400004	0x34110004	ori $17,$0,0x00000004	8: ori $s1, $zero, 4 # y = 4
□	0x00400008	0x00104100	sll $8,$16,0x00000004	9: sll $t0, $s0, 4 # t0 = x << 4 -> 16x
□	0x0040000c	0x01104022	sub $8,$8,$16	10: sub $t0, $t0, $s0 # t0 = t0 - x -> 15x
□	0x00400010	0x00114980	sll $9,$17,0x00000006	12: sll $t1, $s1, 6 # t1 = y << 6 -> 64y
□	0x00400014	0x01314820	add $9,$9,$17	13: add $t1, $t1, $s1 # t1 = t1 + y -> 65y
□	0x00400018	0x01314820	add $9,$9,$17	14: add $t1, $t1, $s1 # t1 = t1 + y -> 66y
□	0x0040001c	0x01314820	add $9,$9,$17	15: add $t1, $t1, $s1 # t1 = t1 + y -> 67y
□	0x00400020	0x01095020	add $10,$8,$9	17: add $t2, $t0, $t1 # t2 = t0 + t1
□	0x00400024	0x000a5080	sll $10,$10,0x00000002	18: sll $t2, $t2, 2 # t2 = t2 << 2 -> 4 * t2
□	0x00400028	0x000a9020	add $18,$0,$10	19: add $s2 ,$zero, $t2 # z = t2

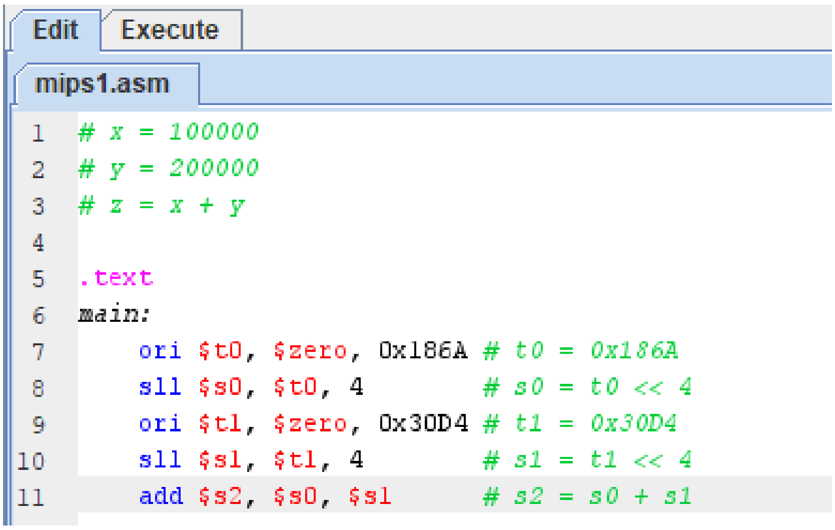
Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	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

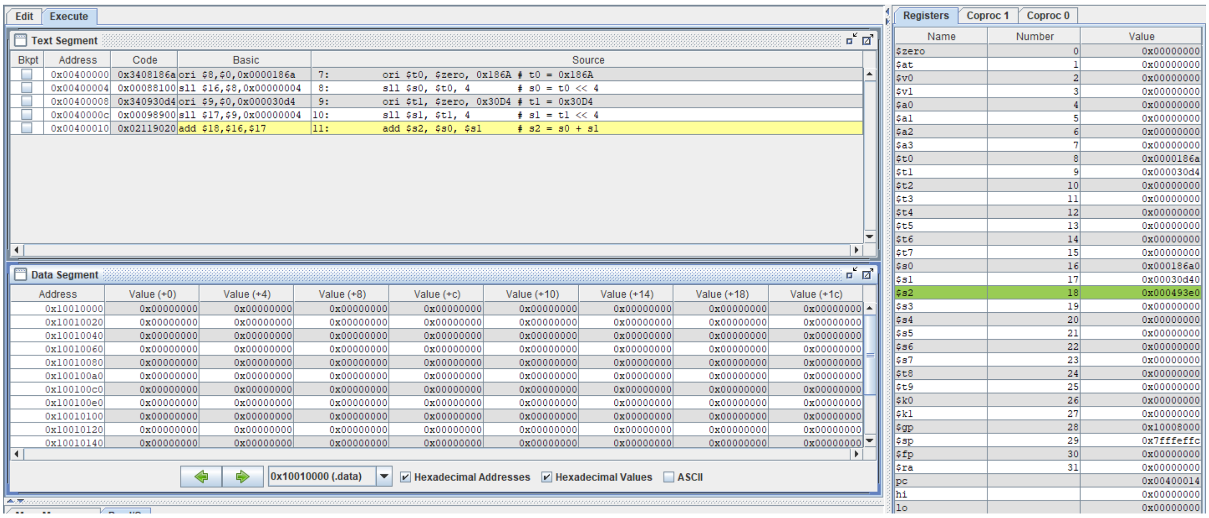
0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values □ ASCII

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x0000002d
$t1	9	0x0000010c
$t2	10	0x000004e4
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000003
$s1	17	0x00000004
$s2	18	0x000004e4
$s3	19	0x00000000
$s4	20	0x00000000
$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	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x0040002c
hi		0x00000000
lo		0x00000000

Programa 5:



```
# x = 100000
# y = 200000
# z = x + y

.text
main:
        ori $t0, $zero, 0x186A # t0 = 0x186A
        sll $s0, $t0, 4        # s0 = t0 << 4
        ori $t1, $zero, 0x30D4 # t1 = 0x30D4
        sll $s1, $t1, 4        # s1 = t1 << 4
        add $s2, $s0, $s1      # s2 = s0 + s1
```



Text Segment

Bkpt	Address	Code	Basic	Source
□	0x00400000	0x3408186a	ori $8,$0,0x0000186a	7: ori $t0, $zero, 0x186A # t0 = 0x186A
□	0x00400004	0x00088100	sll $16,$8,0x00000004	8: sll $s0, $t0, 4 # s0 = t0 << 4
□	0x00400008	0x340930d4	ori $9,$0,0x000030d4	9: ori $t1, $zero, 0x30D4 # t1 = 0x30D4
□	0x0040000c	0x00098900	sll $17,$9,0x00000004	10: sll $s1, $t1, 4 # s1 = t1 << 4
□	0x00400010	0x02119020	add $18,$16,$17	11: add $s2, $s0, $s1 # s2 = s0 + s1

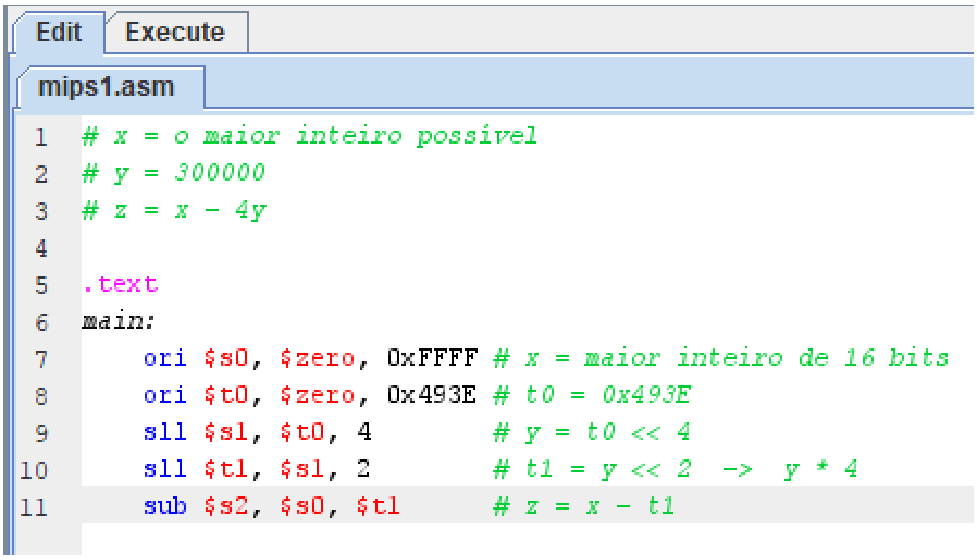
Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	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

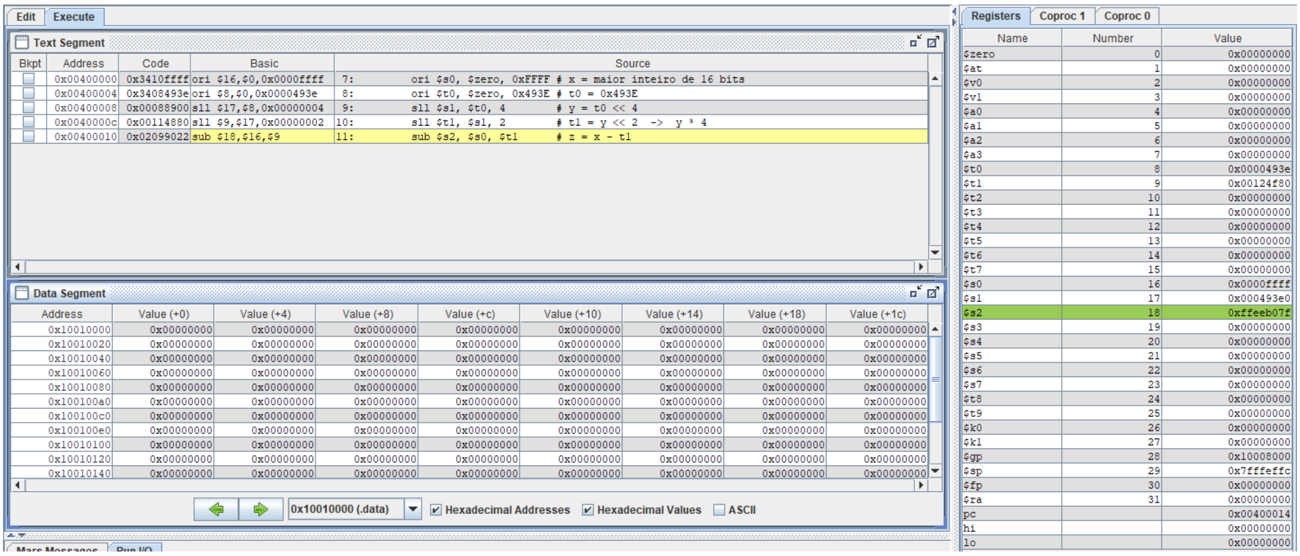
0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values □ ASCII

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x0000186a
$t1	9	0x000030d4
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x000186a0
$s1	17	0x00030d40
$s2	18	0x000493e0
$s3	19	0x00000000
$s4	20	0x00000000
$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	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400014
hi		0x00000000
lo		0x00000000

Programa 6:



```
# x = o maior inteiro possivel
# y = 300000
# z = x - 4y

.text
main:
    ori $s0, $zero, 0xFFFF # x = maior inteiro de 16 bits
    ori $t0, $zero, 0x493E # t0 = 0x493E
    sll $s1, $t0, 4        # y = t0 << 4
    sll $t1, $s1, 2        # t1 = y << 2  ->  y * 4
    sub $s2, $s0, $t1      # z = x - t1
```



Text Segment

Bkpt	Address	Code	Basic	Source
□	0x00400000	0x3410ffff	ori $16,$0,0x0000ffff	7: ori $s0, $zero, 0xFFFF # x = maior inteiro de 16 bits
□	0x00400004	0x3408493e	ori $8,$0,0x0000493e	8: ori $t0, $zero, 0x493E # t0 = 0x493E
□	0x00400008	0x00088900	sll $17,$8,0x00000004	9: sll $s1, $t0, 4 # y = t0 << 4
□	0x0040000c	0x00114880	sll $9,$17,0x00000002	10: sll $t1, $s1, 2 # t1 = y << 2 -> y * 4
□	0x00400010	0x02099022	sub $18,$16,$9	11: sub $s2, $s0, $t1 # z = x - t1

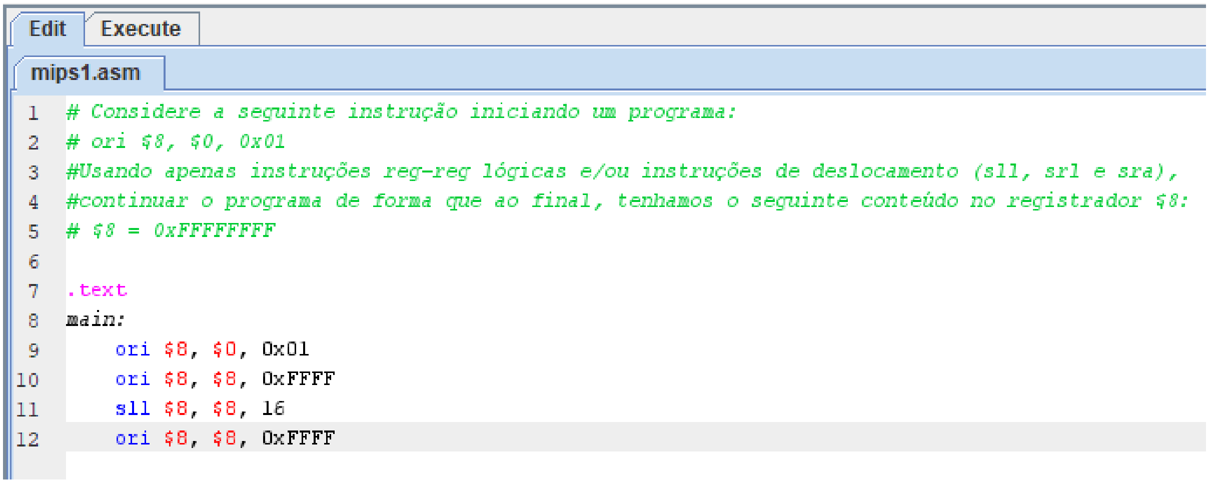
Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	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

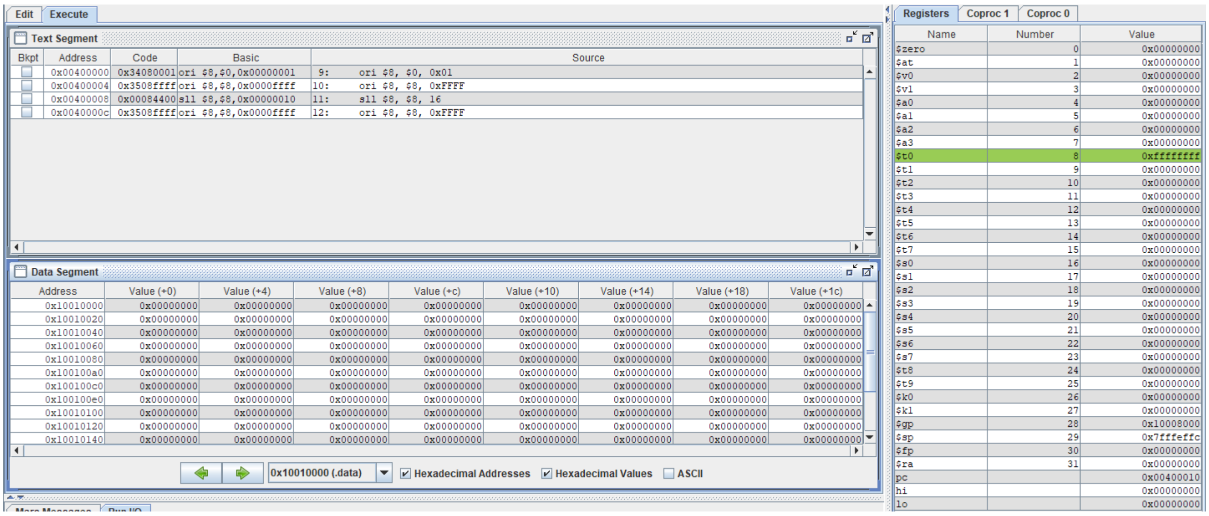
0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values □ ASCII

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x0000493e
$t1	9	0x00124f80
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x0000ffff
$s1	17	0x000493e0
$s2	18	0xffeeb07f
$s3	19	0x00000000
$s4	20	0x00000000
$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	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400014
hi		0x00000000
lo		0x00000000

Programa 7:



```
# 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 = 0xFFFFFFFF

.text
main:
	ori $8, $0, 0x01
	ori $8, $8, 0xFFFF
	sll $8, $8, 16
	ori $8, $8, 0xFFFF
```



Text Segment

Bkpt	Address	Code	Basic	Source
□	0x00400000	0x34080001	ori $8,$0,0x00000001	9: ori $8, $0, 0x01
□	0x00400004	0x3508ffff	ori $8,$8,0x0000ffff	10: ori $8, $8, 0xFFFF
□	0x00400008	0x00084400	sll $8,$8,0x00000010	11: sll $8, $8, 16
□	0x0040000c	0x3508ffff	ori $8,$8,0x0000ffff	12: ori $8, $8, 0xFFFF

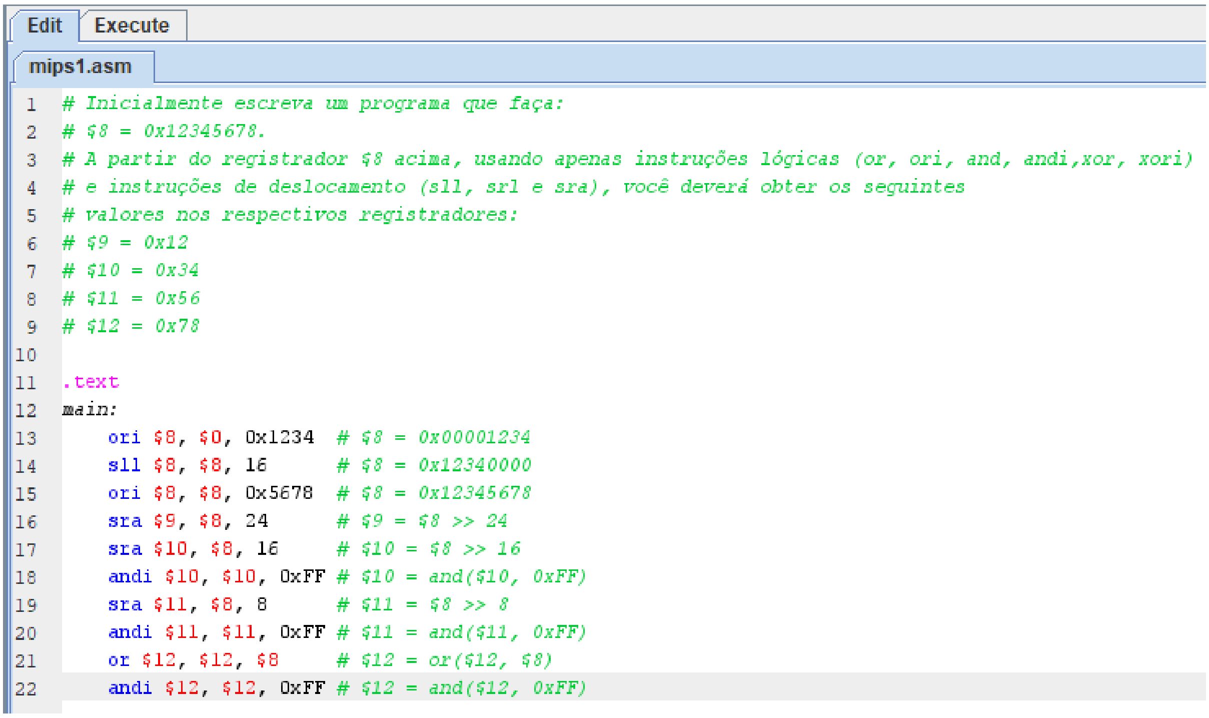
Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	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

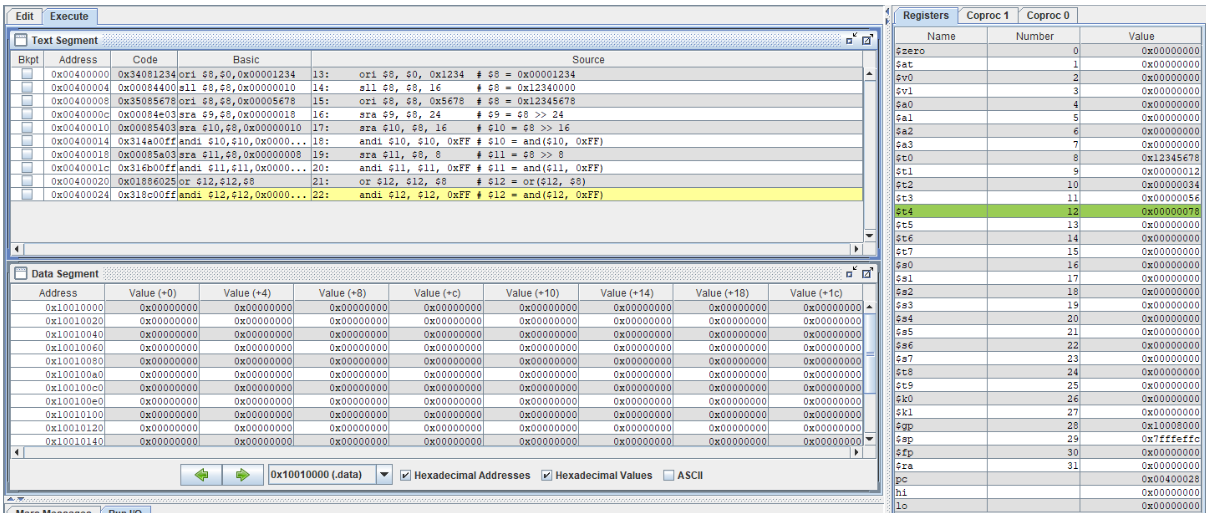
0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values □ ASCII

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0xffffffff
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$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	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400010
hi		0x00000000
lo		0x00000000

Programa 8:



```
# 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

.text
main:
    ori $8, $0, 0x1234  # $8 = 0x00001234
    sll $8, $8, 16      # $8 = 0x12340000
    ori $8, $8, 0x5678  # $8 = 0x12345678
    sra $9, $8, 24      # $9 = $8 >> 24
    sra $10, $8, 16     # $10 = $8 >> 16
    andi $10, $10, 0xFF # $10 = and($10, 0xFF)
    sra $11, $8, 8      # $11 = $8 >> 8
    andi $11, $11, 0xFF # $11 = and($11, 0xFF)
    or $12, $12, $8     # $12 = or($12, $8)
    andi $12, $12, 0xFF # $12 = and($12, 0xFF)
```



Text Segment

Bkpt	Address	Code	Basic	Source
☐	0x00400000	0x34081234	ori $8,$0,0x00001234	13: ori $8, $0, 0x1234 # $8 = 0x00001234
☐	0x00400004	0x00084400	sll $8,$8,0x00000010	14: sll $8, $8, 16 # $8 = 0x12340000
☐	0x00400008	0x35085678	ori $8,$8,0x00005678	15: ori $8, $8, 0x5678 # $8 = 0x12345678
☐	0x0040000c	0x00084e03	sra $9,$8,0x00000018	16: sra $9, $8, 24 # $9 = $8 >> 24
☐	0x00400010	0x00085403	sra $10,$8,0x00000010	17: sra $10, $8, 16 # $10 = $8 >> 16
☐	0x00400014	0x314a00ff	andi $10,$10,0x0000...	18: andi $10, $10, 0xFF # $10 = and($10, 0xFF)
☐	0x00400018	0x00085a03	sra $11,$8,0x00000008	19: sra $11, $8, 8 # $11 = $8 >> 8
☐	0x0040001c	0x316b00ff	andi $11,$11,0x0000...	20: andi $11, $11, 0xFF # $11 = and($11, 0xFF)
☐	0x00400020	0x01886025	or $12,$12,$8	21: or $12, $12, $8 # $12 = or($12, $8)
☐	0x00400024	0x318c00ff	andi $12,$12,0x0000...	22: andi $12, $12, 0xFF # $12 = and($12, 0xFF)

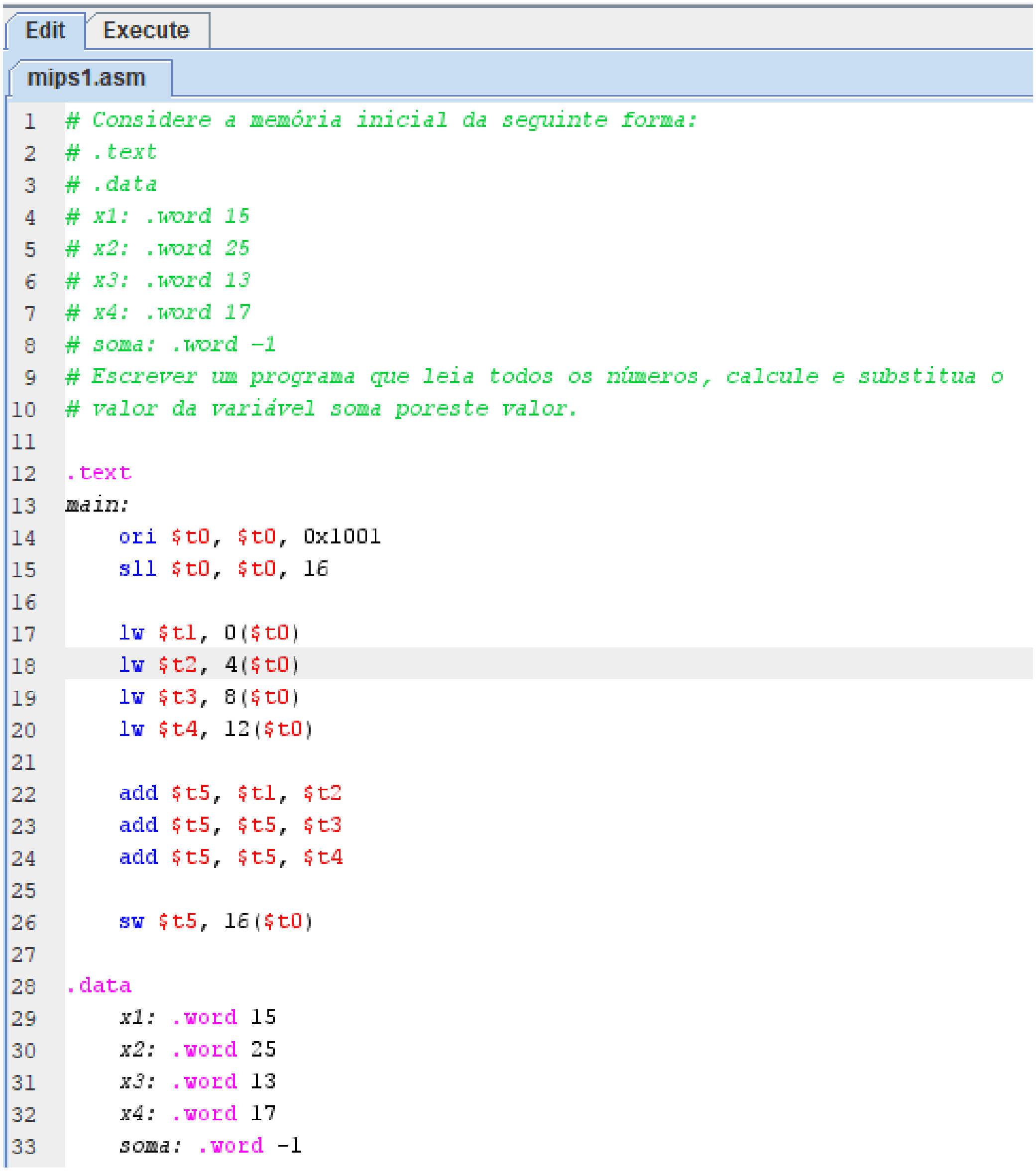
Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	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

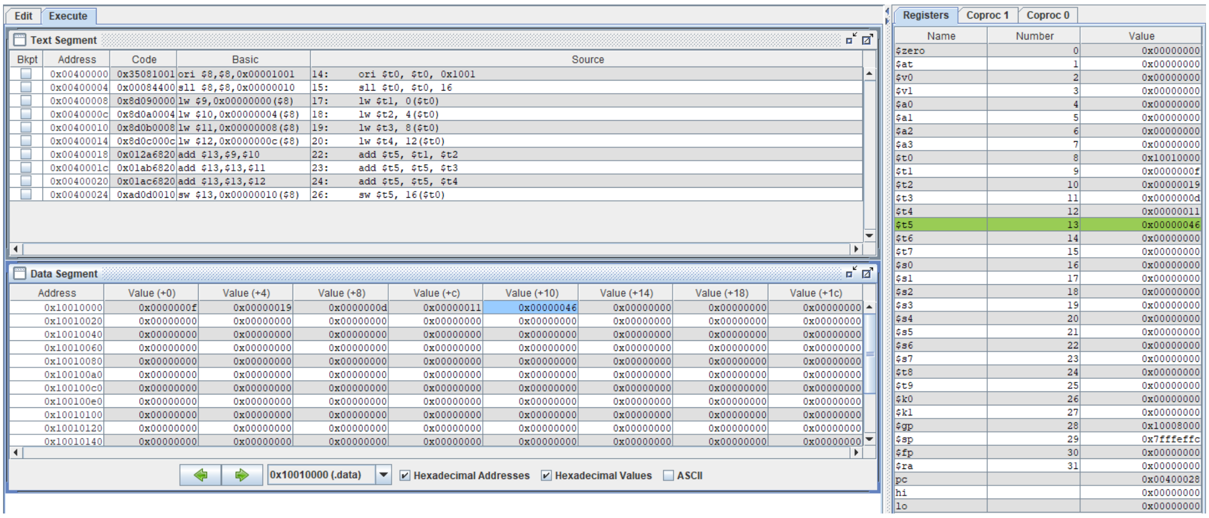
0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values ☐ ASCII

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x12345678
$t1	9	0x00000012
$t2	10	0x00000034
$t3	11	0x00000056
$t4	12	0x00000078
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$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	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400028
hi		0x00000000
lo		0x00000000

Programa 9:



```
# 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 poreste valor.

.text
main:
    ori $t0, $t0, 0x1001
    sll $t0, $t0, 16

    lw $t1, 0($t0)
    lw $t2, 4($t0)
    lw $t3, 8($t0)
    lw $t4, 12($t0)

    add $t5, $t1, $t2
    add $t5, $t5, $t3
    add $t5, $t5, $t4

    sw $t5, 16($t0)

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



Text Segment

Bkpt	Address	Code	Basic	Source
□	0x00400000	0x35081001	ori $8,$8,0x00001001	14: ori $t0, $t0, 0x1001
□	0x00400004	0x00084400	sll $8,$8,0x00000010	15: sll $t0, $t0, 16
□	0x00400008	0x8d090000	lw $9,0x00000000($8)	17: lw $t1, 0($t0)
□	0x0040000c	0x8d0a0004	lw $10,0x00000004($8)	18: lw $t2, 4($t0)
□	0x00400010	0x8d0b0008	lw $11,0x00000008($8)	19: lw $t3, 8($t0)
□	0x00400014	0x8d0c000c	lw $12,0x0000000c($8)	20: lw $t4, 12($t0)
□	0x00400018	0x012a6820	add $13,$9,$10	22: add $t5, $t1, $t2
□	0x0040001c	0x01ab6820	add $13,$13,$11	23: add $t5, $t5, $t3
□	0x00400020	0x01ac6820	add $13,$13,$12	24: add $t5, $t5, $t4
□	0x00400024	0xad0d0010	sw $13,0x00000010($8)	26: sw $t5, 16($t0)

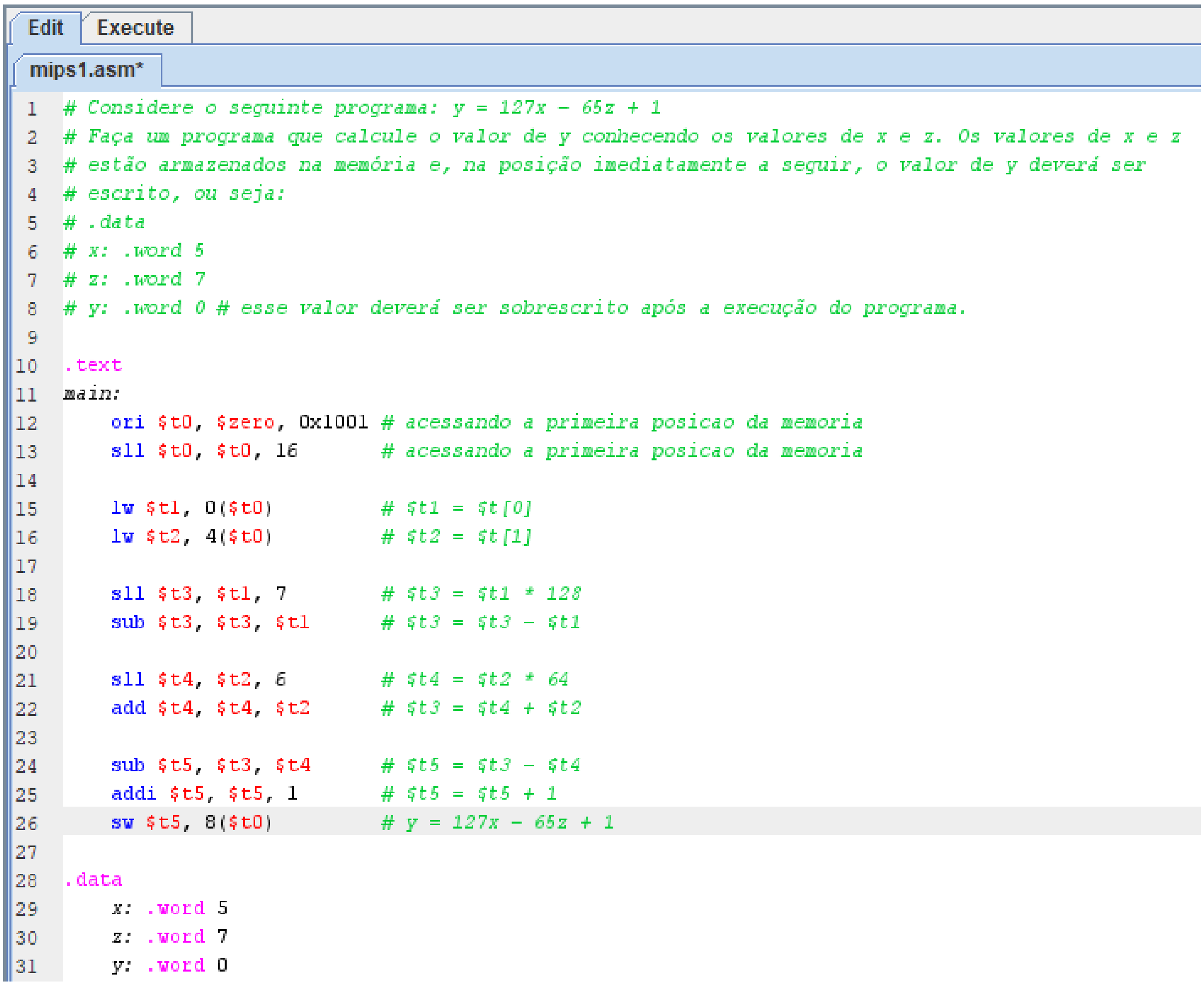
Data Segment

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

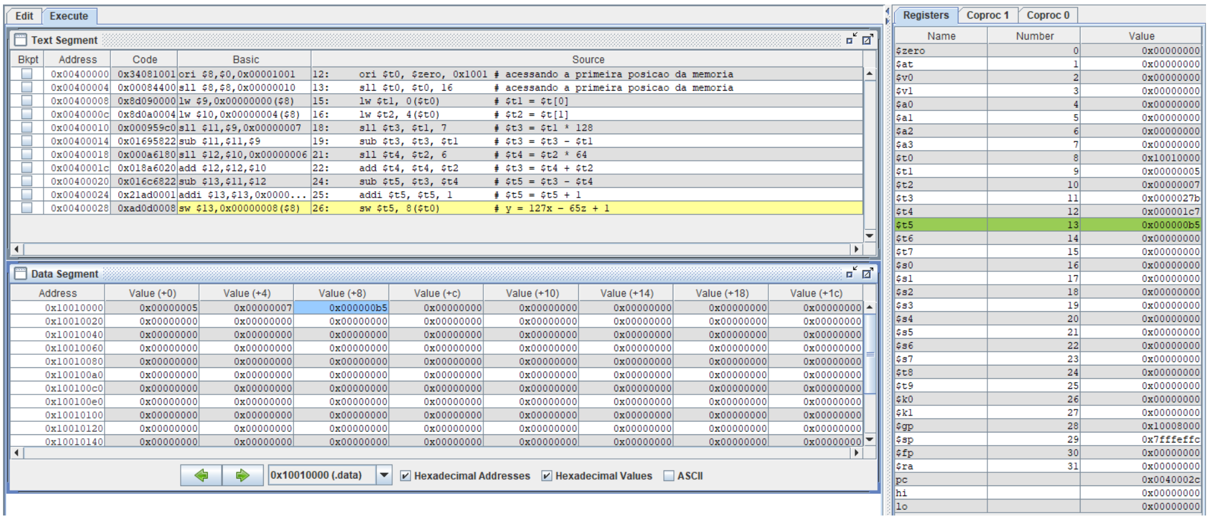
0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values □ ASCII

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x10010000
$t1	9	0x0000000f
$t2	10	0x00000019
$t3	11	0x0000000d
$t4	12	0x00000011
$t5	13	0x00000046
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$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	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400028
hi		0x00000000
lo		0x00000000

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.

.text
main:
    ori $t0, $zero, 0x1001 # acessando a primeira posicao da memoria
    sll $t0, $t0, 16       # acessando a primeira posicao da memoria

    lw $t1, 0($t0)         # $t1 = $t[0]
    lw $t2, 4($t0)         # $t2 = $t[1]

    sll $t3, $t1, 7        # $t3 = $t1 * 128
    sub $t3, $t3, $t1      # $t3 = $t3 - $t1

    sll $t4, $t2, 6        # $t4 = $t2 * 64
    add $t4, $t4, $t2      # $t3 = $t4 + $t2

    sub $t5, $t3, $t4      # $t5 = $t3 - $t4
    addi $t5, $t5, 1       # $t5 = $t5 + 1
    sw $t5, 8($t0)         # y = 127x - 65z + 1

.data
    x: .word 5
    z: .word 7
    y: .word 0
```



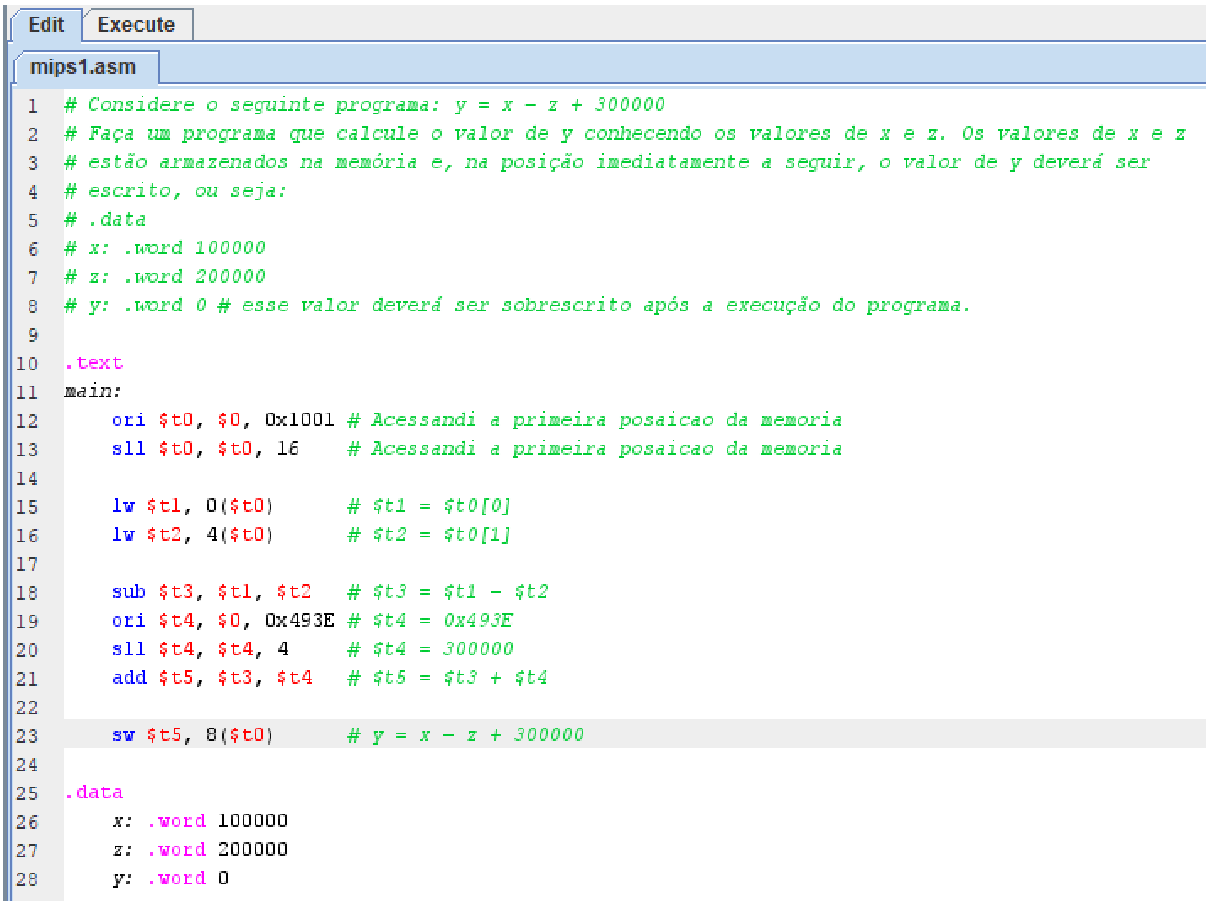
Bkpt	Address	Code	Basic	Source
□	0x00400000	0x34081001	ori $8,$0,0x00001001	12: ori $t0, $zero, 0x1001 # acessando a primeira posicao da memoria
□	0x00400004	0x00084400	sll $8,$8,0x00000010	13: sll $t0, $t0, 16 # acessando a primeira posicao da memoria
□	0x00400008	0x8d090000	lw $9,0x00000000($8)	15: lw $t1, 0($t0) # $t1 = $t[0]
□	0x0040000c	0x8d0a0004	lw $10,0x00000004($8)	16: lw $t2, 4($t0) # $t2 = $t[1]
□	0x00400010	0x000959c0	sll $11,$9,0x00000007	18: sll $t3, $t1, 7 # $t3 = $t1 * 128
□	0x00400014	0x01695822	sub $11,$11,$9	19: sub $t3, $t3, $t1 # $t3 = $t3 - $t1
□	0x00400018	0x000a6180	sll $12,$10,0x00000006	21: sll $t4, $t2, 6 # $t4 = $t2 * 64
□	0x0040001c	0x018a6020	add $12,$12,$10	22: add $t4, $t4, $t2 # $t3 = $t4 + $t2
□	0x00400020	0x016c6822	sub $13,$11,$12	24: sub $t5, $t3, $t4 # $t5 = $t3 - $t4
□	0x00400024	0x21ad0001	addi $13,$13,0x0000...	25: addi $t5, $t5, 1 # $t5 = $t5 + 1
□	0x00400028	0xad0d0008	sw $13,0x00000008($8)	26: sw $t5, 8($t0) # y = 127x - 65z + 1

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000005	0x00000007	0x000000b5	0x00000000	0x00000000	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

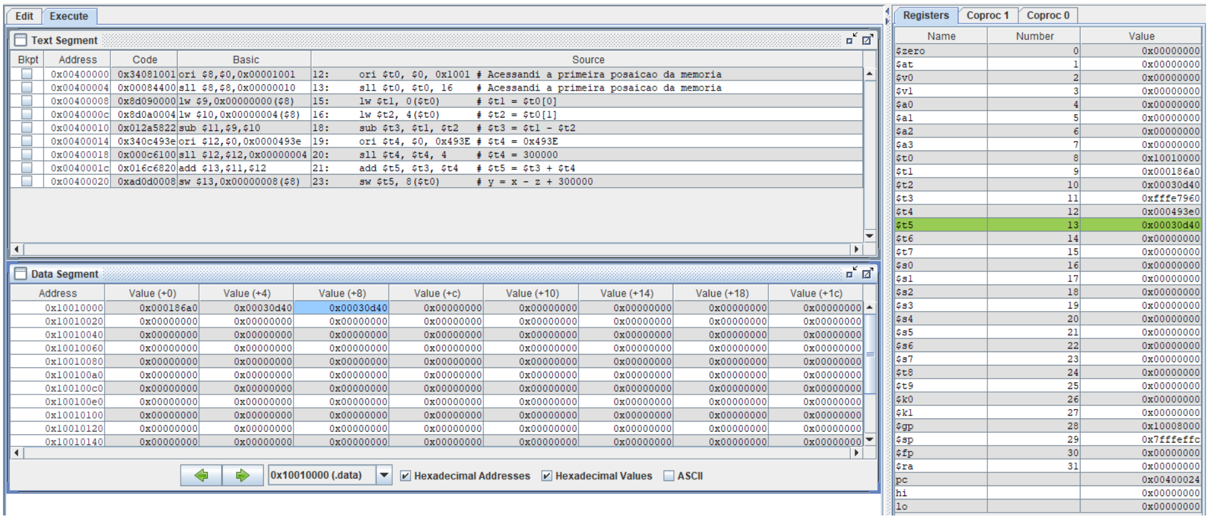
0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values □ ASCII

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x10010000
$t1	9	0x00000005
$t2	10	0x00000007
$t3	11	0x0000027b
$t4	12	0x000001c7
$t5	13	0x000000b5
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$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	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x0040002c
hi		0x00000000
lo		0x00000000

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.

.text
main:
    ori $t0, $0, 0x1001 # Acessandi a primeira posaicao da memoria
    sll $t0, $t0, 16    # Acessandi a primeira posaicao da memoria

    lw $t1, 0($t0)      # $t1 = $t0[0]
    lw $t2, 4($t0)      # $t2 = $t0[1]

    sub $t3, $t1, $t2   # $t3 = $t1 - $t2
    ori $t4, $0, 0x493E # $t4 = 0x493E
    sll $t4, $t4, 4     # $t4 = 300000
    add $t5, $t3, $t4   # $t5 = $t3 + $t4

    sw $t5, 8($t0)      # y = x - z + 300000

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



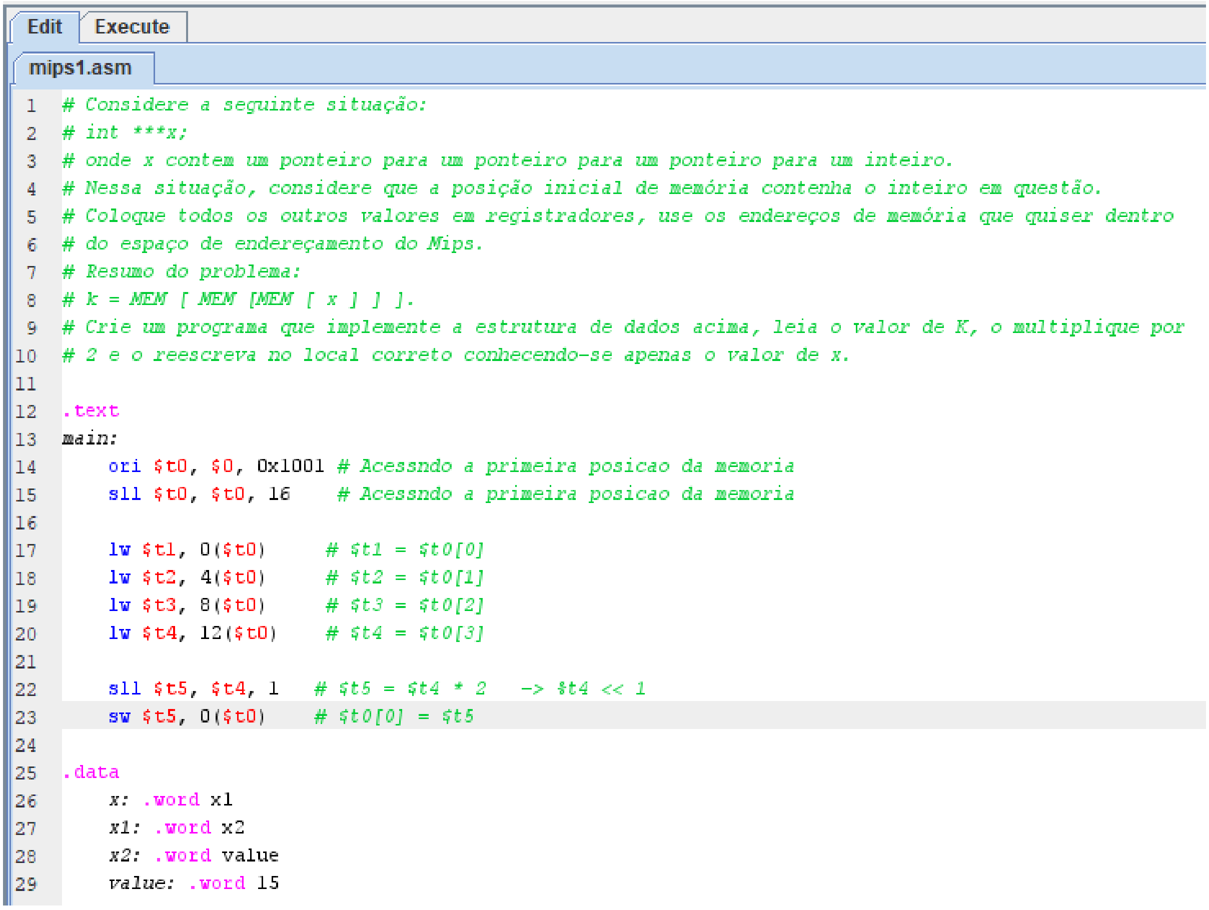
Bkpt	Address	Code	Basic	Source
□	0x00400000	0x34081001	ori $8,$0,0x00001001	12: ori $t0, $0, 0x1001 # Acessandi a primeira posaicao da memoria
□	0x00400004	0x00084400	sll $8,$8,0x00000010	13: sll $t0, $t0, 16 # Acessandi a primeira posaicao da memoria
□	0x00400008	0x8d090000	lw $9,0x00000000($8)	15: lw $t1, 0($t0) # $t1 = $t0[0]
□	0x0040000c	0x8d0a0004	lw $10,0x00000004($8)	16: lw $t2, 4($t0) # $t2 = $t0[1]
□	0x00400010	0x012a5822	sub $11,$9,$10	18: sub $t3, $t1, $t2 # $t3 = $t1 - $t2
□	0x00400014	0x340c493e	ori $12,$0,0x0000493e	19: ori $t4, $0, 0x493E # $t4 = 0x493E
□	0x00400018	0x000c6100	sll $12,$12,0x00000004	20: sll $t4, $t4, 4 # $t4 = 300000
□	0x0040001c	0x016c6820	add $13,$11,$12	21: add $t5, $t3, $t4 # $t5 = $t3 + $t4
□	0x00400020	0xad0d0008	sw $13,0x00000008($8)	23: sw $t5, 8($t0) # y = x - z + 300000

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x000186a0	0x00030d40	0x00030d40	0x00000000	0x00000000	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

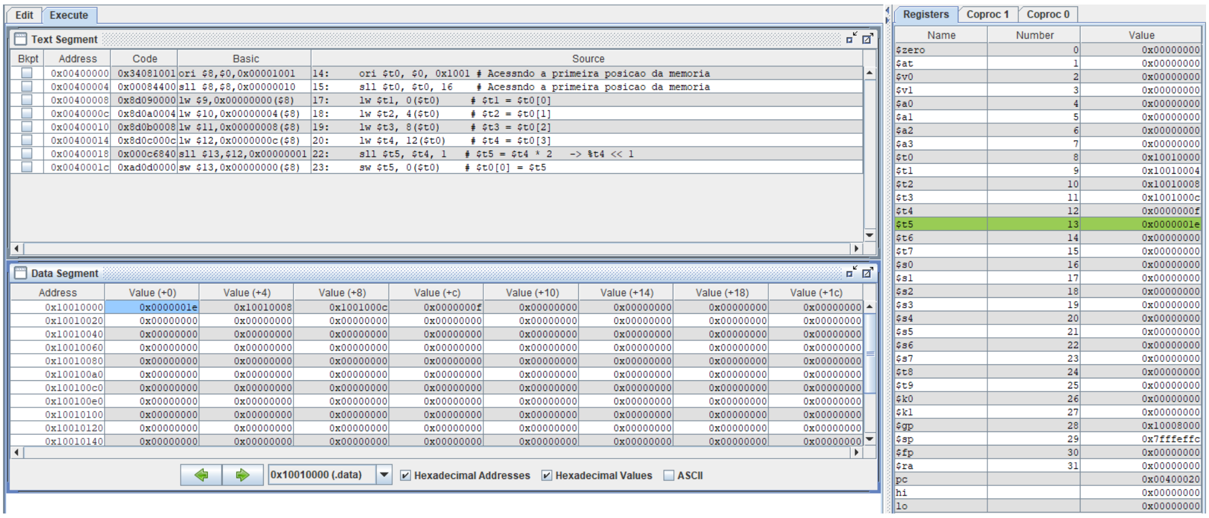
0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values □ ASCII

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x10010000
$t1	9	0x000186a0
$t2	10	0x00030d40
$t3	11	0xfffe7960
$t4	12	0x000493e0
$t5	13	0x00030d40
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$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	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400024
hi		0x00000000
lo		0x00000000

Programa 12:



```
# 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 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.

.text
main:
    ori $t0, $0, 0x1001 # Acessndo a primeira posicao da memoria
    sll $t0, $t0, 16    # Acessndo a primeira posicao da memoria

    lw $t1, 0($t0)      # $t1 = $t0[0]
    lw $t2, 4($t0)      # $t2 = $t0[1]
    lw $t3, 8($t0)      # $t3 = $t0[2]
    lw $t4, 12($t0)     # $t4 = $t0[3]

    sll $t5, $t4, 1   # $t5 = $t4 * 2   -> $t4 << 1
    sw $t5, 0($t0)    # $t0[0] = $t5

.data
    x: .word x1
    x1: .word x2
    x2: .word value
    value: .word 15
```



Text Segment

Bkpt	Address	Code	Basic	Source
□	0x00400000	0x34081001	ori $8,$0,0x00001001	14: ori $t0, $0, 0x1001 # Acessndo a primeira posicao da memoria
□	0x00400004	0x00084400	sll $8,$8,0x00000010	15: sll $t0, $t0, 16 # Acessndo a primeira posicao da memoria
□	0x00400008	0x8d090000	lw $9,0x00000000($8)	17: lw $t1, 0($t0) # $t1 = $t0[0]
□	0x0040000c	0x8d0a0004	lw $10,0x00000004($8)	18: lw $t2, 4($t0) # $t2 = $t0[1]
□	0x00400010	0x8d0b0008	lw $11,0x00000008($8)	19: lw $t3, 8($t0) # $t3 = $t0[2]
□	0x00400014	0x8d0c000c	lw $12,0x0000000c($8)	20: lw $t4, 12($t0) # $t4 = $t0[3]
□	0x00400018	0x000c6840	sll $13,$12,0x00000001	22: sll $t5, $t4, 1 # $t5 = $t4 * 2 -> %t4 << 1
□	0x0040001c	0xad0d0000	sw $13,0x00000000($8)	23: sw $t5, 0($t0) # $t0[0] = $t5

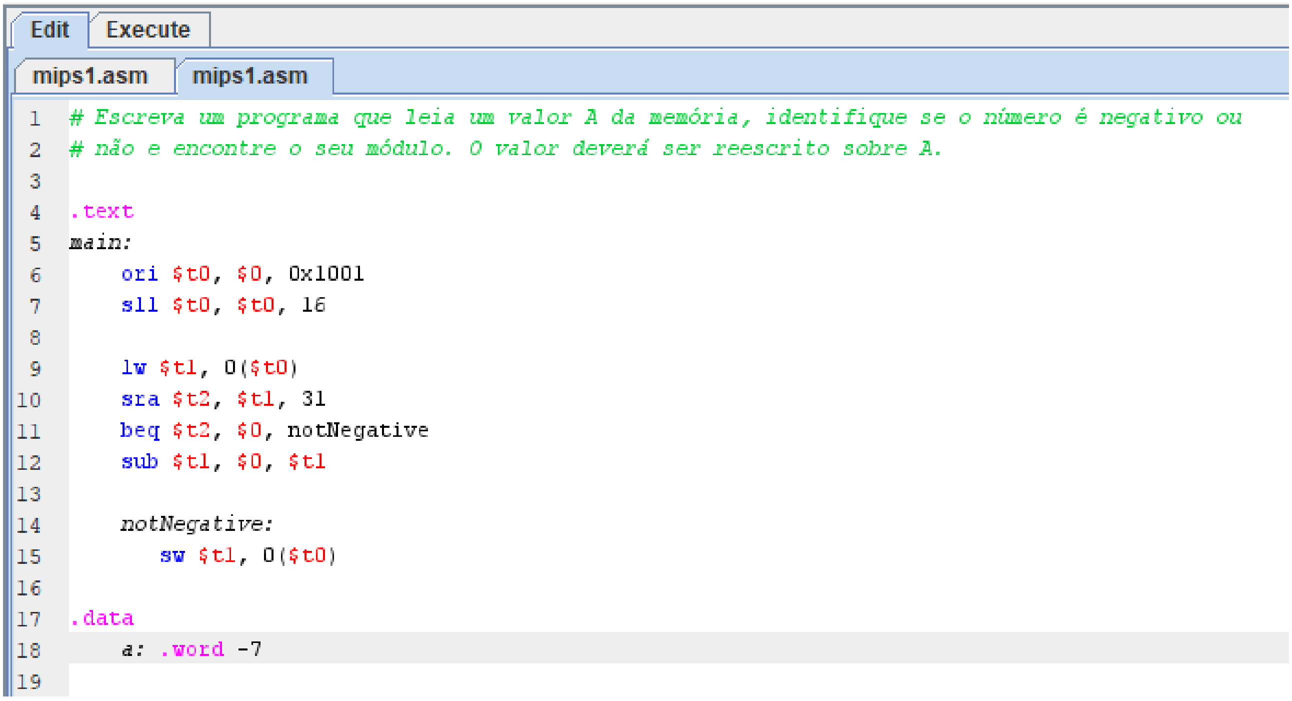
Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x0000001e	0x10010008	0x1001000c	0x0000000f	0x00000000	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

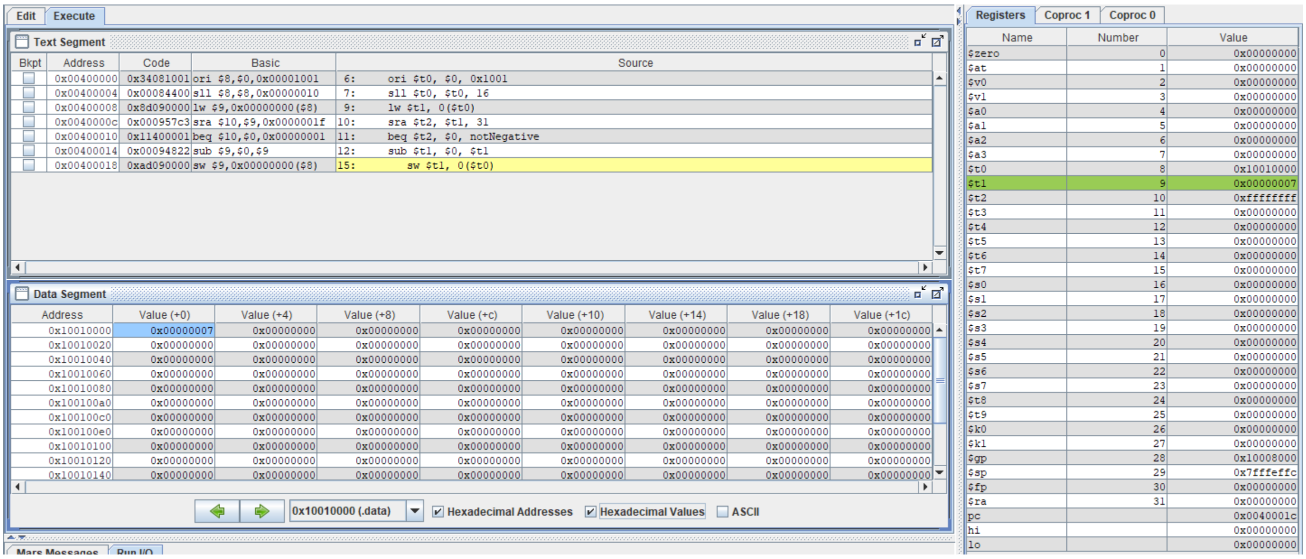
0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values □ ASCII

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x10010000
$t1	9	0x10010004
$t2	10	0x10010008
$t3	11	0x1001000c
$t4	12	0x0000000f
$t5	13	0x0000001e
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$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	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400020
hi		0x00000000
lo		0x00000000

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.

.text
main:
    ori $t0, $0, 0x1001
    sll $t0, $t0, 16

    lw $t1, 0($t0)
    sra $t2, $t1, 31
    beq $t2, $0, notNegative
    sub $t1, $0, $t1

    notNegative:
       sw $t1, 0($t0)

.data
    a: .word -7
```



Text Segment

Bkpt	Address	Code	Basic	Source
□	0x00400000	0x34081001	ori $8,$0,0x00001001	6: ori $t0, $0, 0x1001
□	0x00400004	0x00084400	sll $8,$8,0x00000010	7: sll $t0, $t0, 16
□	0x00400008	0x8d090000	lw $9,0x00000000($8)	9: lw $t1, 0($t0)
□	0x0040000c	0x000957c3	sra $10,$9,0x0000001f	10: sra $t2, $t1, 31
□	0x00400010	0x11400001	beq $10,$0,0x00000001	11: beq $t2, $0, notNegative
□	0x00400014	0x00094822	sub $9,$0,$9	12: sub $t1, $0, $t1
□	0x00400018	0xad090000	sw $9,0x00000000($8)	15: sw $t1, 0($t0)

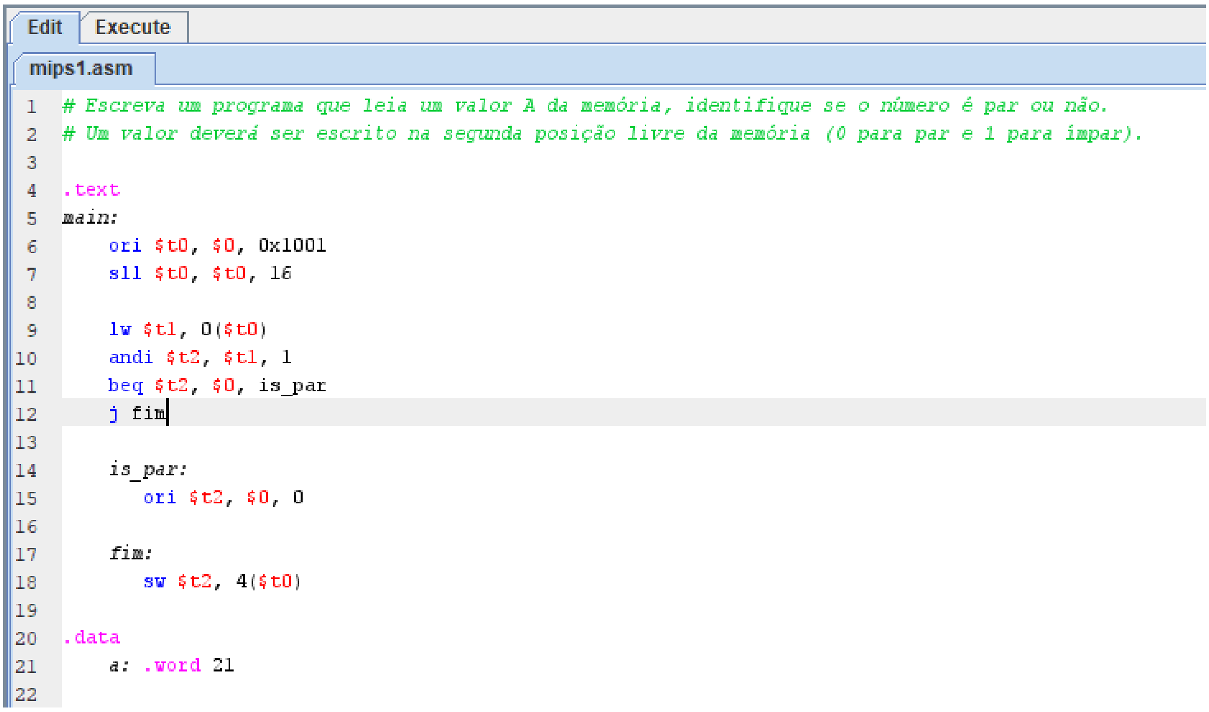
Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000007	0x00000000	0x00000000	0x00000000	0x00000000	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

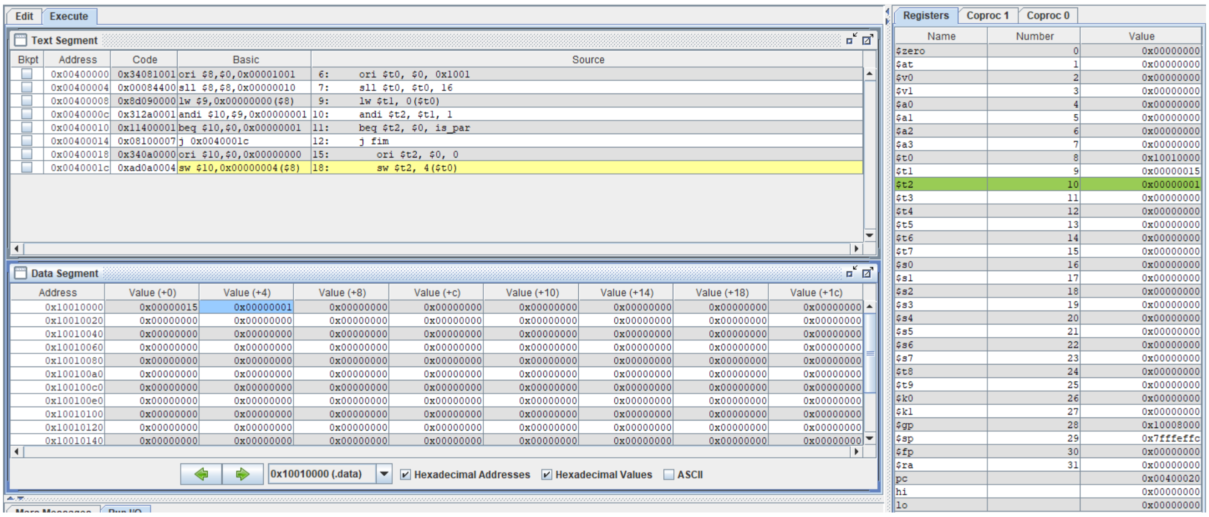
0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values □ ASCII

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x10010000
$t1	9	0x00000007
$t2	10	0xffffffff
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$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	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x0040001c
hi		0x00000000
lo		0x00000000

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 impar).

.text
main:
    ori $t0, $0, 0x1001
    sll $t0, $t0, 16

    lw $t1, 0($t0)
    andi $t2, $t1, 1
    beq $t2, $0, is_par
    j fim

    is_par:
       ori $t2, $0, 0

    fim:
       sw $t2, 4($t0)

.data
    a: .word 21

```



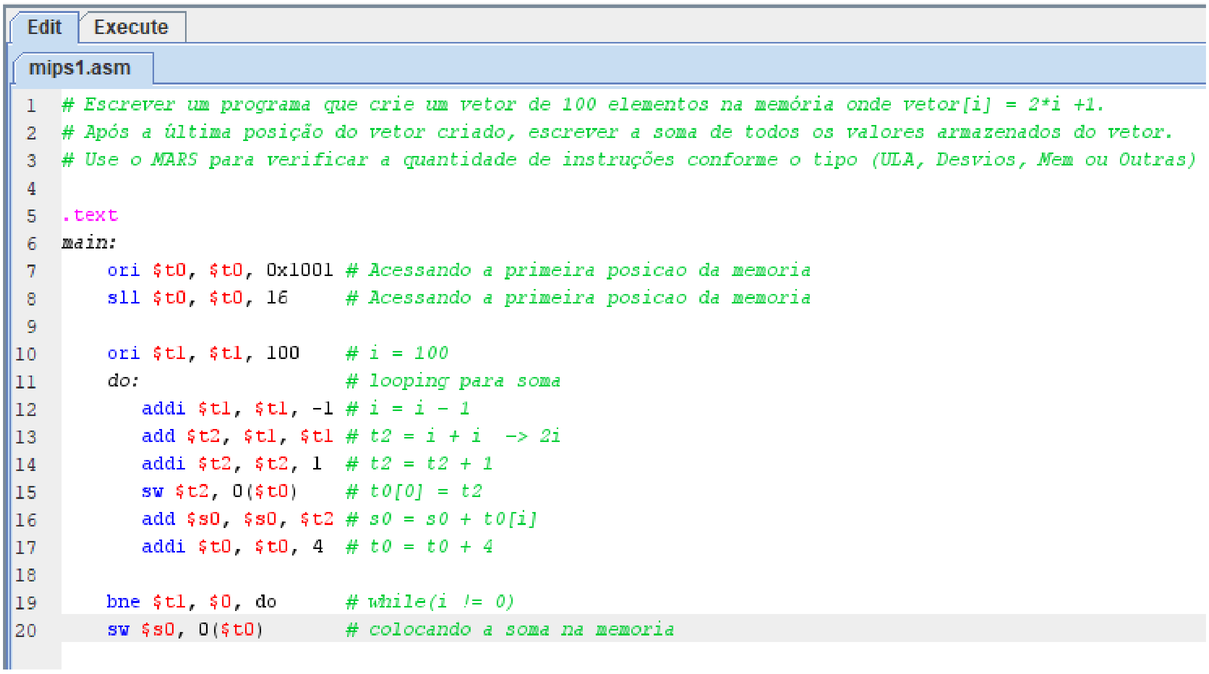
Bkpt	Address	Code	Basic	Source
□	0x00400000	0x34081001	ori $8,$0,0x00001001	6: ori $t0, $0, 0x1001
□	0x00400004	0x00084400	sll $8,$8,0x00000010	7: sll $t0, $t0, 16
□	0x00400008	0x8d090000	lw $9,0x00000000($8)	9: lw $t1, 0($t0)
□	0x0040000c	0x312a0001	andi $10,$9,0x00000001	10: andi $t2, $t1, 1
□	0x00400010	0x11400001	beq $10,$0,0x00000001	11: beq $t2, $0, is_par
□	0x00400014	0x08100007	j 0x0040001c	12: j fim
□	0x00400018	0x340a0000	ori $10,$0,0x00000000	15: ori $t2, $0, 0
□	0x0040001c	0xad0a0004	sw $10,0x00000004($8)	18: sw $t2, 4($t0)

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000015	0x00000001	0x00000000	0x00000000	0x00000000	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

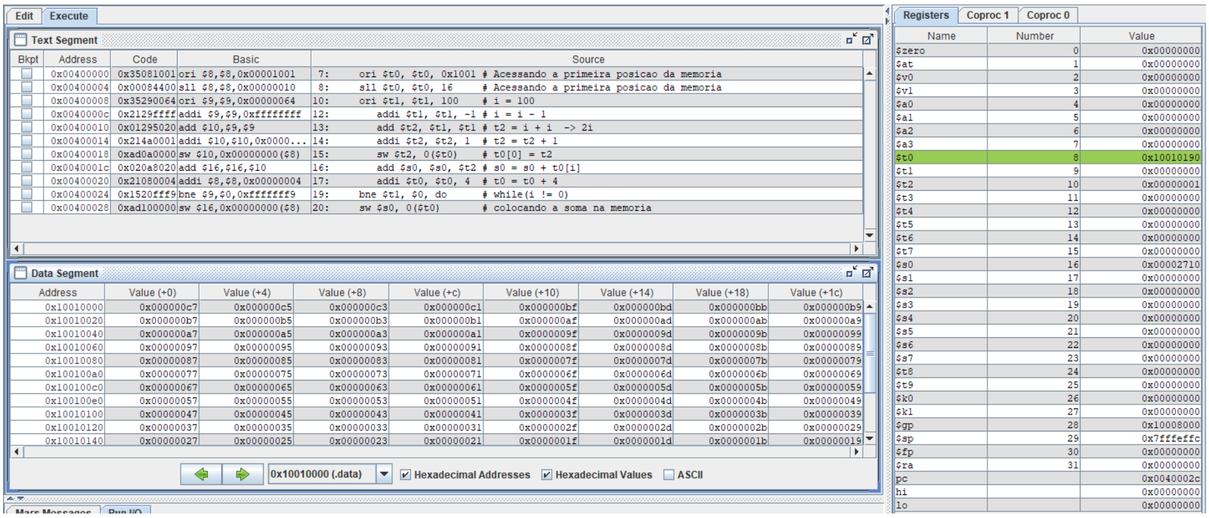
0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values □ ASCII

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x10010000
$t1	9	0x00000015
$t2	10	0x00000001
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$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	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400020
hi		0x00000000
lo		0x00000000

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)

.text
main:
    ori $t0, $t0, 0x1001 # Acessando a primeira posicao da memoria
    sll $t0, $t0, 16     # Acessando a primeira posicao da memoria

    ori $t1, $t1, 100    # i = 100
    do:                  # looping para soma
       addi $t1, $t1, -1 # i = i - 1
       add $t2, $t1, $t1 # t2 = i + i  -> 2i
       addi $t2, $t2, 1  # t2 = t2 + 1
       sw $t2, 0($t0)    # t0[0] = t2
       add $s0, $s0, $t2 # s0 = s0 + t0[i]
       addi $t0, $t0, 4  # t0 = t0 + 4

    bne $t1, $0, do      # while(i != 0)
    sw $s0, 0($t0)       # colocando a soma na memoria
```



Text Segment

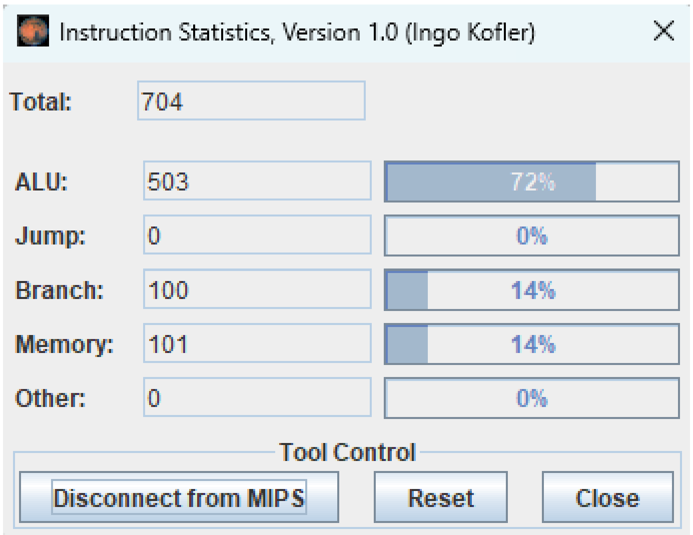
Bkpt	Address	Code	Basic	Source
☐	0x00400000	0x35081001	ori $8,$8,0x00001001	7: ori $t0, $t0, 0x1001 # Acessando a primeira posicao da memoria
☐	0x00400004	0x00084400	sll $8,$8,0x00000010	8: sll $t0, $t0, 16 # Acessando a primeira posicao da memoria
☐	0x00400008	0x35290064	ori $9,$9,0x00000064	10: ori $t1, $t1, 100 # i = 100
☐	0x0040000c	0x2129ffff	addi $9,$9,0xffffffff	12: addi $t1, $t1, -1 # i = i - 1
☐	0x00400010	0x01295020	add $10,$9,$9	13: add $t2, $t1, $t1 # t2 = i + i -> 2i
☐	0x00400014	0x214a0001	addi $10,$10,0x0000...	14: addi $t2, $t2, 1 # t2 = t2 + 1
☐	0x00400018	0xad0a0000	sw $10,0x00000000($8)	15: sw $t2, 0($t0) # t0[0] = t2
☐	0x0040001c	0x020a8020	add $16,$16,$10	16: add $s0, $s0, $t2 # s0 = s0 + t0[i]
☐	0x00400020	0x21080004	addi $8,$8,0x00000004	17: addi $t0, $t0, 4 # t0 = t0 + 4
☐	0x00400024	0x1520fff9	bne $9,$0,0xfffffff9	19: bne $t1, $0, do # while(i != 0)
☐	0x00400028	0xad100000	sw $16,0x00000000($8)	20: sw $s0, 0($t0) # colocando a soma na memoria

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x000000c7	0x000000c5	0x000000c3	0x000000c1	0x000000bf	0x000000bd	0x000000bb	0x000000b9
0x10010020	0x000000b7	0x000000b5	0x000000b3	0x000000b1	0x000000af	0x000000ad	0x000000ab	0x000000a9
0x10010040	0x000000a7	0x000000a5	0x000000a3	0x000000a1	0x0000009f	0x0000009d	0x0000009b	0x00000099
0x10010060	0x00000097	0x00000095	0x00000093	0x00000091	0x0000008f	0x0000008d	0x0000008b	0x00000089
0x10010080	0x00000087	0x00000085	0x00000083	0x00000081	0x0000007f	0x0000007d	0x0000007b	0x00000079
0x100100a0	0x00000077	0x00000075	0x00000073	0x00000071	0x0000006f	0x0000006d	0x0000006b	0x00000069
0x100100c0	0x00000067	0x00000065	0x00000063	0x00000061	0x0000005f	0x0000005d	0x0000005b	0x00000059
0x100100e0	0x00000057	0x00000055	0x00000053	0x00000051	0x0000004f	0x0000004d	0x0000004b	0x00000049
0x10010100	0x00000047	0x00000045	0x00000043	0x00000041	0x0000003f	0x0000003d	0x0000003b	0x00000039
0x10010120	0x00000037	0x00000035	0x00000033	0x00000031	0x0000002f	0x0000002d	0x0000002b	0x00000029
0x10010140	0x00000027	0x00000025	0x00000023	0x00000021	0x0000001f	0x0000001d	0x0000001b	0x00000019

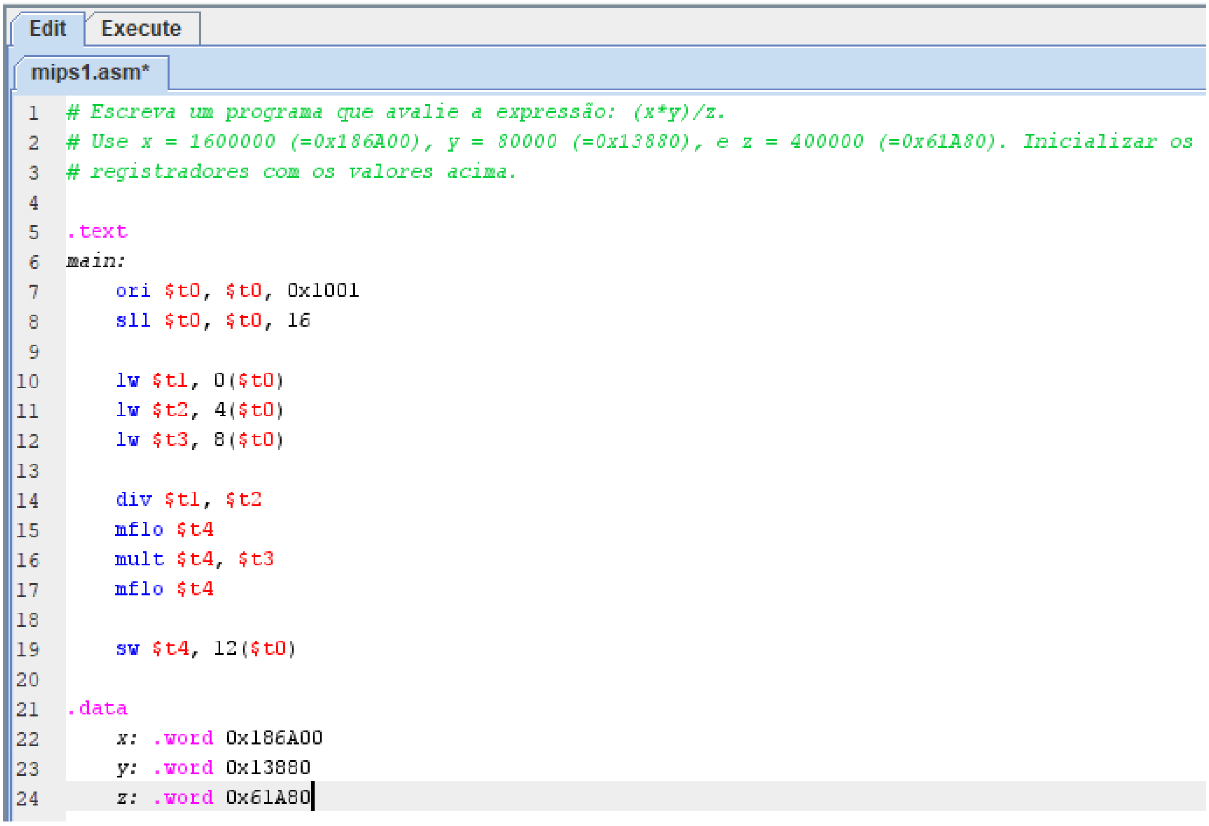
0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values ☐ ASCII

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x10010190
$t1	9	0x00000000
$t2	10	0x00000001
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00002710
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$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	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x0040002c
hi		0x00000000
lo		0x00000000

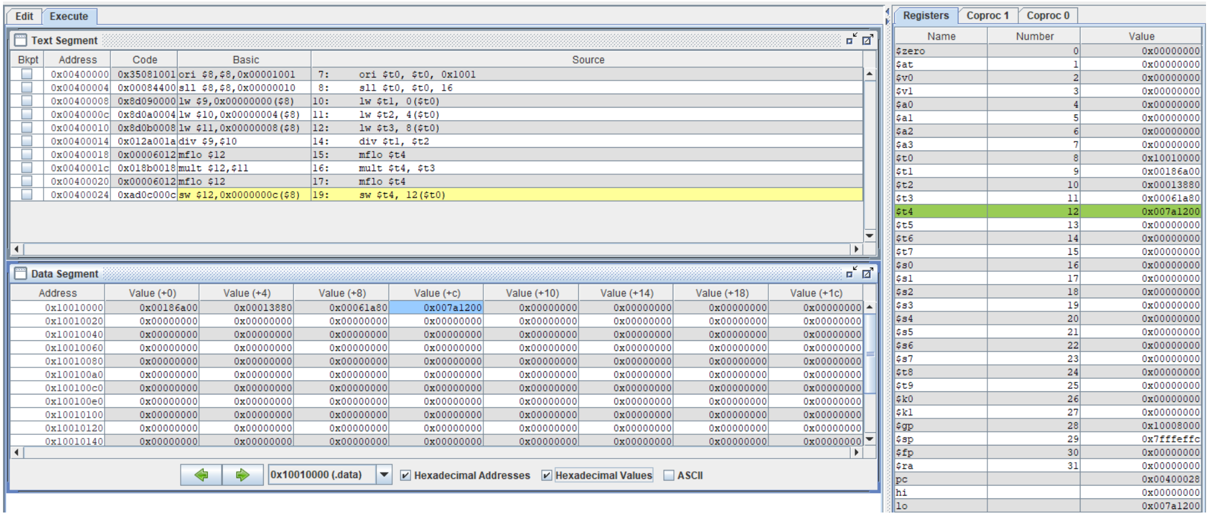


Total:	704	
ALU:	503	72%
Jump:	0	0%
Branch:	100	14%
Memory:	101	14%
Other:	0	0%

Programa 16:



```
# 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.

.text
main:
    ori $t0, $t0, 0x1001
    sll $t0, $t0, 16

    lw $t1, 0($t0)
    lw $t2, 4($t0)
    lw $t3, 8($t0)

    div $t1, $t2
    mflo $t4
    mult $t4, $t3
    mflo $t4

    sw $t4, 12($t0)

.data
    x: .word 0x186A00
    y: .word 0x13880
    z: .word 0x61A80
```



Text Segment

Bkpt	Address	Code	Basic	Source
□	0x00400000	0x35081001	ori $8,$8,0x00001001	7: ori $t0, $t0, 0x1001
□	0x00400004	0x00084400	sll $8,$8,0x00000010	8: sll $t0, $t0, 16
□	0x00400008	0x8d090000	lw $9,0x00000000($8)	10: lw $t1, 0($t0)
□	0x0040000c	0x8d0a0004	lw $10,0x00000004($8)	11: lw $t2, 4($t0)
□	0x00400010	0x8d0b0008	lw $11,0x00000008($8)	12: lw $t3, 8($t0)
□	0x00400014	0x012a001a	div $9,$10	14: div $t1, $t2
□	0x00400018	0x00006012	mflo $12	15: mflo $t4
□	0x0040001c	0x018b0018	mult $12,$11	16: mult $t4, $t3
□	0x00400020	0x00006012	mflo $12	17: mflo $t4
□	0x00400024	0xad0c000c	sw $12,0x0000000c($8)	19: sw $t4, 12($t0)

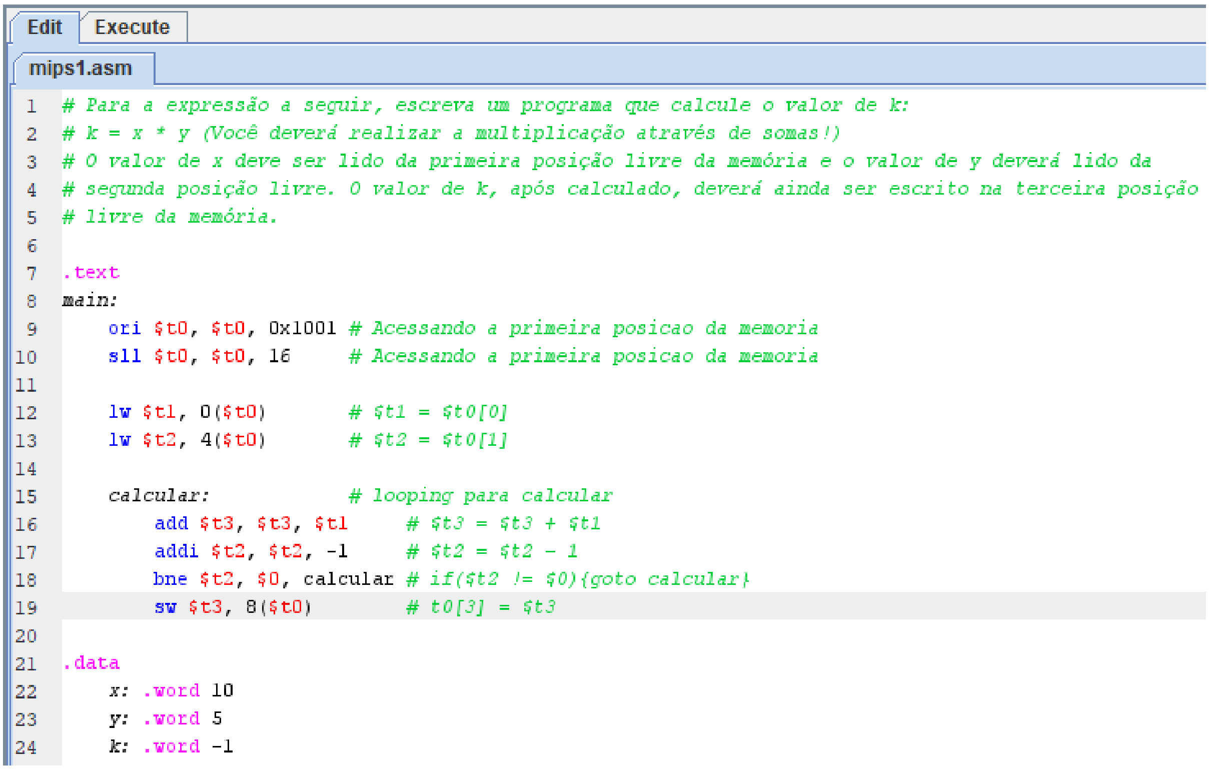
Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00186a00	0x00013880	0x00061a80	0x007a1200	0x00000000	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

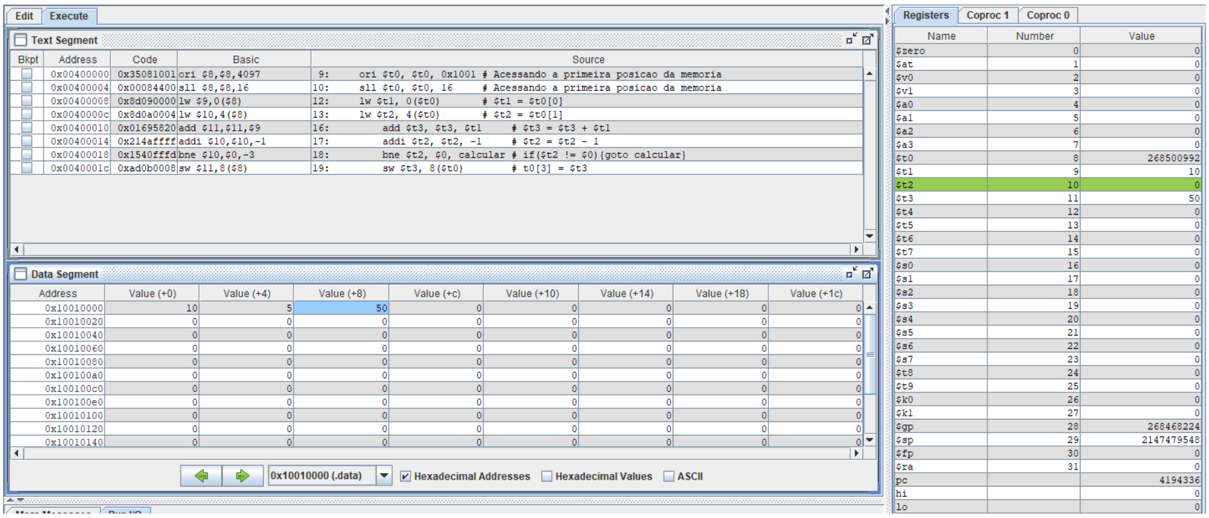
0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values □ ASCII

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x10010000
$t1	9	0x00186a00
$t2	10	0x00013880
$t3	11	0x00061a80
$t4	12	0x007a1200
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$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	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400028
hi		0x00000000
lo		0x007a1200

Programa 17:



```
# 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.

.text
main:
    ori $t0, $t0, 0x1001 # Acessando a primeira posicao da memoria
    sll $t0, $t0, 16     # Acessando a primeira posicao da memoria

    lw $t1, 0($t0)       # $t1 = $t0[0]
    lw $t2, 4($t0)       # $t2 = $t0[1]

    calcular:            # looping para calcular
        add $t3, $t3, $t1     # $t3 = $t3 + $t1
        addi $t2, $t2, -1     # $t2 = $t2 - 1
        bne $t2, $0, calcular # if($t2 != $0){goto calcular}
        sw $t3, 8($t0)        # t0[3] = $t3

.data
    x: .word 10
    y: .word 5
    k: .word -1
```



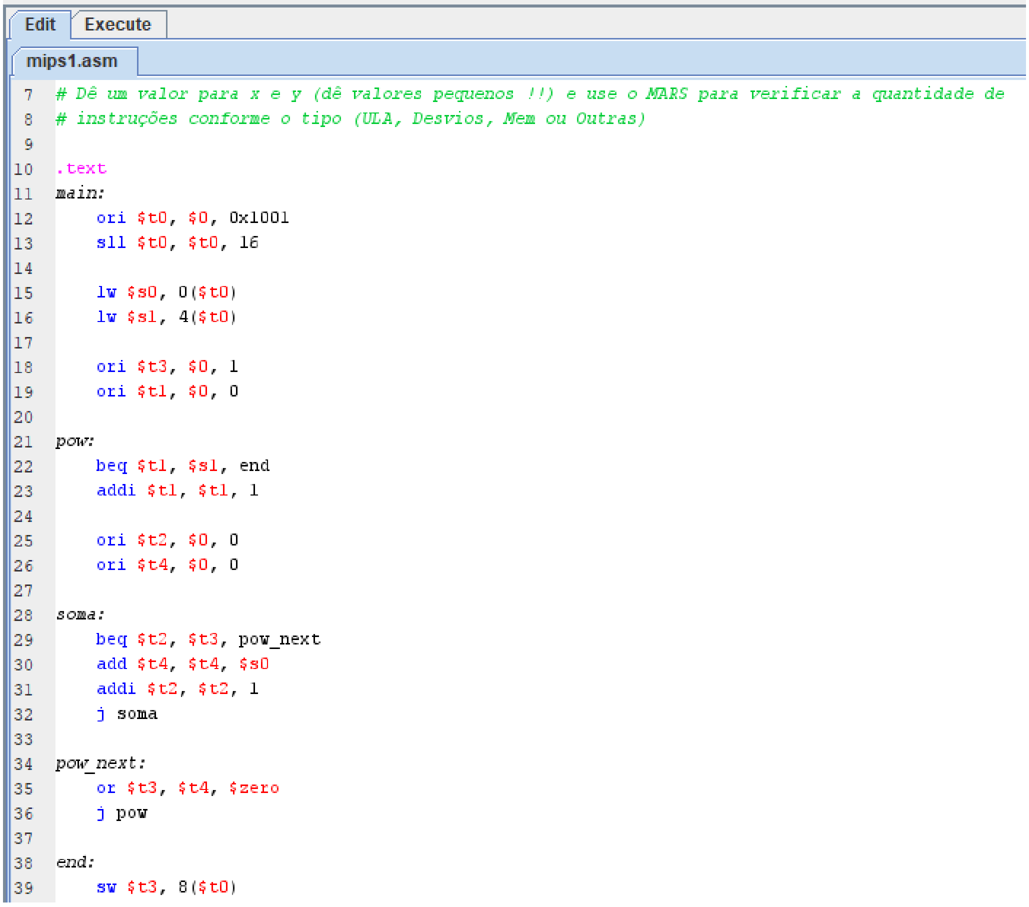
Bkpt	Address	Code	Basic	Source
☐	0x00400000	0x35081001	ori $8,$8,4097	9: ori $t0, $t0, 0x1001 # Acessando a primeira posicao da memoria
☐	0x00400004	0x00084400	sll $8,$8,16	10: sll $t0, $t0, 16 # Acessando a primeira posicao da memoria
☐	0x00400008	0x8d090000	lw $9,0($8)	12: lw $t1, 0($t0) # $t1 = $t0[0]
☐	0x0040000c	0x8d0a0004	lw $10,4($8)	13: lw $t2, 4($t0) # $t2 = $t0[1]
☐	0x00400010	0x01695820	add $11,$11,$9	16: add $t3, $t3, $t1 # $t3 = $t3 + $t1
☐	0x00400014	0x214affff	addi $10,$10,-1	17: addi $t2, $t2, -1 # $t2 = $t2 - 1
☐	0x00400018	0x1540fffd	bne $10,$0,-3	18: bne $t2, $0, calcular # if($t2 != $0){goto calcular}
☐	0x0040001c	0xad0b0008	sw $11,8($8)	19: sw $t3, 8($t0) # t0[3] = $t3

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	10	5	50	0	0	0	0	0
0x10010020	0	0	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0
0x10010140	0	0	0	0	0	0	0	0

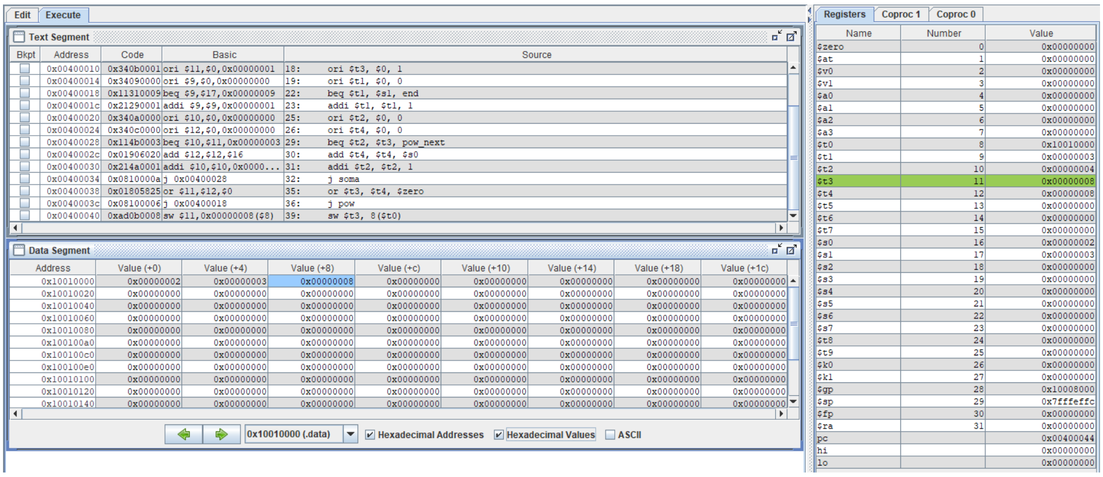
0x10010000 (.data) ☑ Hexadecimal Addresses ☐ Hexadecimal Values ☐ ASCII

Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	0
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	268500992
$t1	9	10
$t2	10	0
$t3	11	50
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	0
$s1	17	0
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0
pc		4194336
hi		0
lo		0

Programa 18:



```
# 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)

.text
main:
    ori $t0, $0, 0x1001
    sll $t0, $t0, 16

    lw $s0, 0($t0)
    lw $s1, 4($t0)

    ori $t3, $0, 1
    ori $t1, $0, 0

pow:
    beq $t1, $s1, end
    addi $t1, $t1, 1

    ori $t2, $0, 0
    ori $t4, $0, 0

soma:
    beq $t2, $t3, pow_next
    add $t4, $t4, $s0
    addi $t2, $t2, 1
    j soma

pow_next:
    or $t3, $t4, $zero
    j pow

end:
    sw $t3, 8($t0)
```

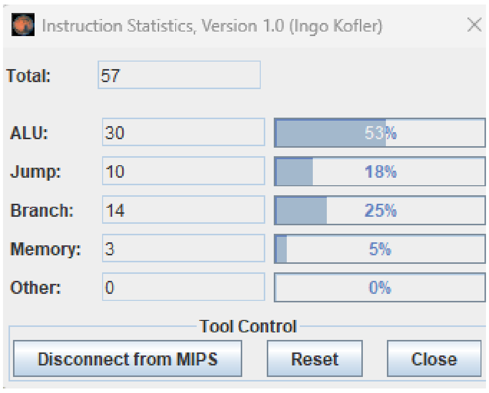


Bkpt	Address	Code	Basic	Source
☐	0x00400010	0x340b0001	ori $11,$0,0x00000001	18: ori $t3, $0, 1
☐	0x00400014	0x34090000	ori $9,$0,0x00000000	19: ori $t1, $0, 0
☐	0x00400018	0x11310009	beq $9,$17,0x00000009	22: beq $t1, $s1, end
☐	0x0040001c	0x21290001	addi $9,$9,0x00000001	23: addi $t1, $t1, 1
☐	0x00400020	0x340a0000	ori $10,$0,0x00000000	25: ori $t2, $0, 0
☐	0x00400024	0x340c0000	ori $12,$0,0x00000000	26: ori $t4, $0, 0
☐	0x00400028	0x114b0003	beq $10,$11,0x00000003	29: beq $t2, $t3, pow_next
☐	0x0040002c	0x01906020	add $12,$12,$16	30: add $t4, $t4, $s0
☐	0x00400030	0x214a0001	addi $10,$10,0x0000...	31: addi $t2, $t2, 1
☐	0x00400034	0x0810000a	j 0x00400028	32: j soma
☐	0x00400038	0x01805825	or $11,$12,$0	35: or $t3, $t4, $zero
☐	0x0040003c	0x08100006	j 0x00400018	36: j pow
☐	0x00400040	0xad0b0008	sw $11,0x00000008($8)	39: sw $t3, 8($t0)

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000002	0x00000003	0x00000008	0x00000000	0x00000000	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

0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values ☐ ASCII

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x10010000
$t1	9	0x00000003
$t2	10	0x00000004
$t3	11	0x00000008
$t4	12	0x00000008
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000002
$s1	17	0x00000003
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$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	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400044
hi		0x00000000
lo		0x00000000



Total:	57	
ALU:	30	53%
Jump:	10	18%
Branch:	14	25%
Memory:	3	5%
Other:	0	0%

1. Se tivermos 2 inteiros, cada um com 32 bits, quantos bits podemos esperar para o produto?

A. 16
B. 32
C. 64
D. 128

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

3. Qual a operação usada para multiplicar inteiros em comp. de dois?

A. mult
B. multu
C. multi
D. Mutt

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

5. Se tivermos dois inteiros, cada um com 32 bits, quantos bits deveremos estar preparados para receber no quociente?

A. 16
B. 32

C. 64
D. 128

6. Após a instrução div, qual registrador possui o quociente?

A. lo
B. hi
C. high
D. $2

7. Qual a inst. Usada para dividir dois inteiros em comp. de dois?

A. dv
B. divide
C. divu
D. div

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

9. Qual o efeito de um arithmetic shift right de uma posição?

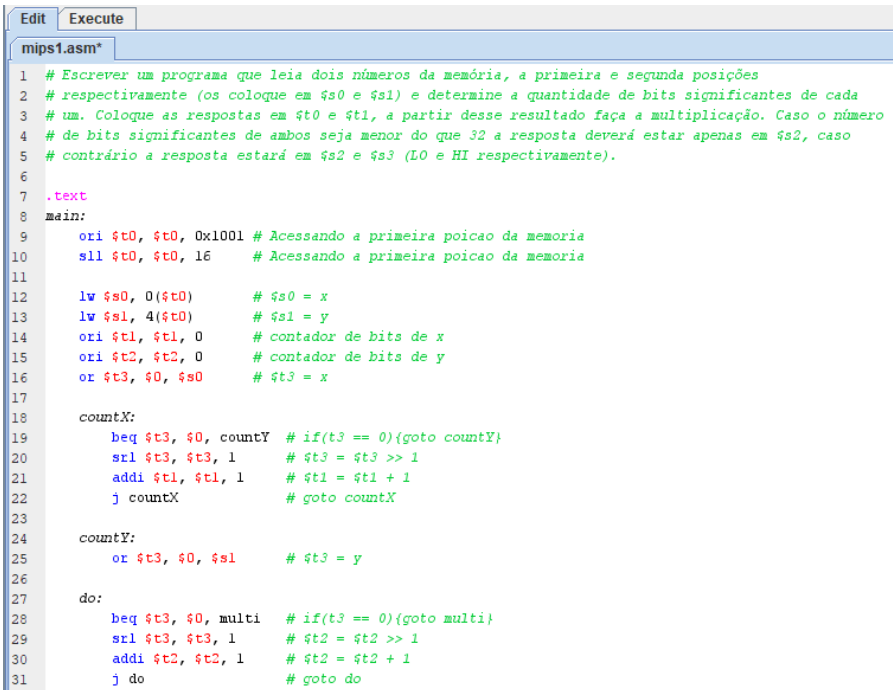
A. Se o inteiro for unsigned, o shift o divide por 2. Se o inteiro for signed, o shift o divide por 2.
B. Se o inteiro for unsigned, o shift o divide por 2. Se o inteiro for signed, o shift pode resultar em um valor errado.
C. Se o inteiro for unsigned, o shift pode ocasionar um valor errado. Se o inteiro for signed, o shift o divide por 2.
D. O shift multiplica o número por dois.

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

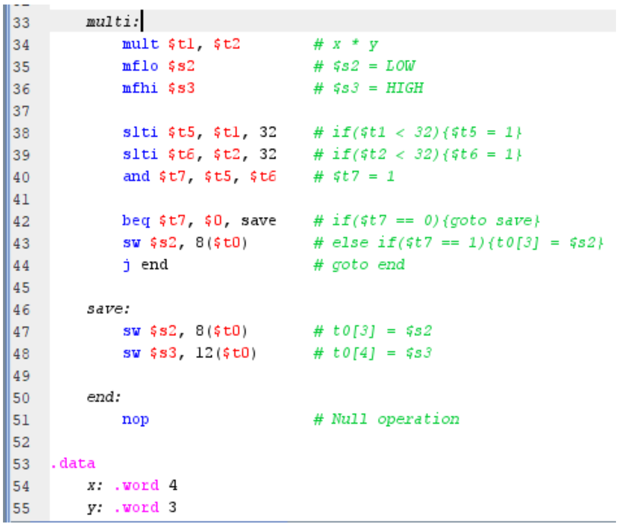
A.
ori $3,$0,3
mult $8,$3
mflo $9
addi $9,$9,7

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

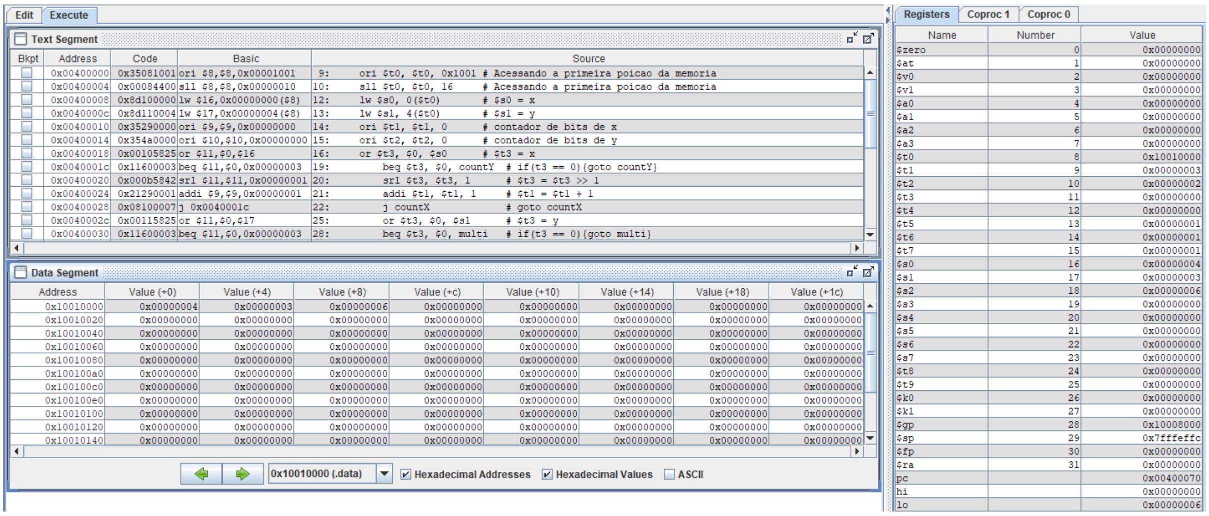
Programa 19:



```
# 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).

.text
main:
    ori $t0, $t0, 0x1001 # Acessando a primeira poicao da memoria
    sll $t0, $t0, 16     # Acessando a primeira poicao da memoria

    lw $s0, 0($t0)       # $s0 = x
    lw $s1, 4($t0)       # $s1 = y
    ori $t1, $t1, 0      # contador de bits de x
    ori $t2, $t2, 0      # contador de bits de y
    or $t3, $0, $s0      # $t3 = x

    countX:
        beq $t3, $0, countY  # if(t3 == 0){goto countY}
        srl $t3, $t3, 1      # $t3 = $t3 >> 1
        addi $t1, $t1, 1     # $t1 = $t1 + 1
        j countX             # goto countX

    countY:
        or $t3, $0, $s1      # $t3 = y

    do:
        beq $t3, $0, multi   # if(t3 == 0){goto multi}
        srl $t3, $t3, 1      # $t2 = $t2 >> 1
        addi $t2, $t2, 1     # $t2 = $t2 + 1
        j do                 # goto do
```



```
    multi:
        mult $t1, $t2           # x * y
        mflo $s2                # $s2 = LOW
        mfhi $s3                # $s3 = HIGH

        slti $t5, $t1, 32       # if($t1 < 32){$t5 = 1}
        slti $t6, $t2, 32       # if($t2 < 32){$t6 = 1}
        and $t7, $t5, $t6       # $t7 = 1

        beq $t7, $0, save       # if($t7 == 0){goto save}
        sw $s2, 8($t0)          # else if($t7 == 1){t0[3] = $s2}
        j end                   # goto end

    save:
        sw $s2, 8($t0)          # t0[3] = $s2
        sw $s3, 12($t0)         # t0[4] = $s3

    end:
        nop                     # Null operation

.data
    x: .word 4
    y: .word 3
```



Text Segment

Bkpt	Address	Code	Basic	Source
□	0x00400000	0x35081001	ori $8,$8,0x00001001	9: ori $t0, $t0, 0x1001 # Acessando a primeira poicao da memoria
□	0x00400004	0x00084400	sll $8,$8,0x00000010	10: sll $t0, $t0, 16 # Acessando a primeira poicao da memoria
□	0x00400008	0x8d100000	lw $16,0x00000000($8)	12: lw $s0, 0($t0) # $s0 = x
□	0x0040000c	0x8d110004	lw $17,0x00000004($8)	13: lw $s1, 4($t0) # $s1 = y
□	0x00400010	0x35290000	ori $9,$9,0x00000000	14: ori $t1, $t1, 0 # contador de bits de x
□	0x00400014	0x354a0000	ori $10,$10,0x00000000	15: ori $t2, $t2, 0 # contador de bits de y
□	0x00400018	0x00105825	or $11,$0,$16	16: or $t3, $0, $s0 # $t3 = x
□	0x0040001c	0x11600003	beq $11,$0,0x00000003	19: beq $t3, $0, countY # if(t3 == 0){goto countY}
□	0x00400020	0x000b5842	srl $11,$11,0x00000001	20: srl $t3, $t3, 1 # $t3 = $t3 >> 1
□	0x00400024	0x21290001	addi $9,$9,0x00000001	21: addi $t1, $t1, 1 # $t1 = $t1 + 1
□	0x00400028	0x08100007	j 0x0040001c	22: j countX # goto countX
□	0x0040002c	0x00115825	or $11,$0,$17	25: or $t3, $0, $s1 # $t3 = y
□	0x00400030	0x11600003	beq $11,$0,0x00000003	28: beq $t3, $0, multi # if(t3 == 0){goto multi}

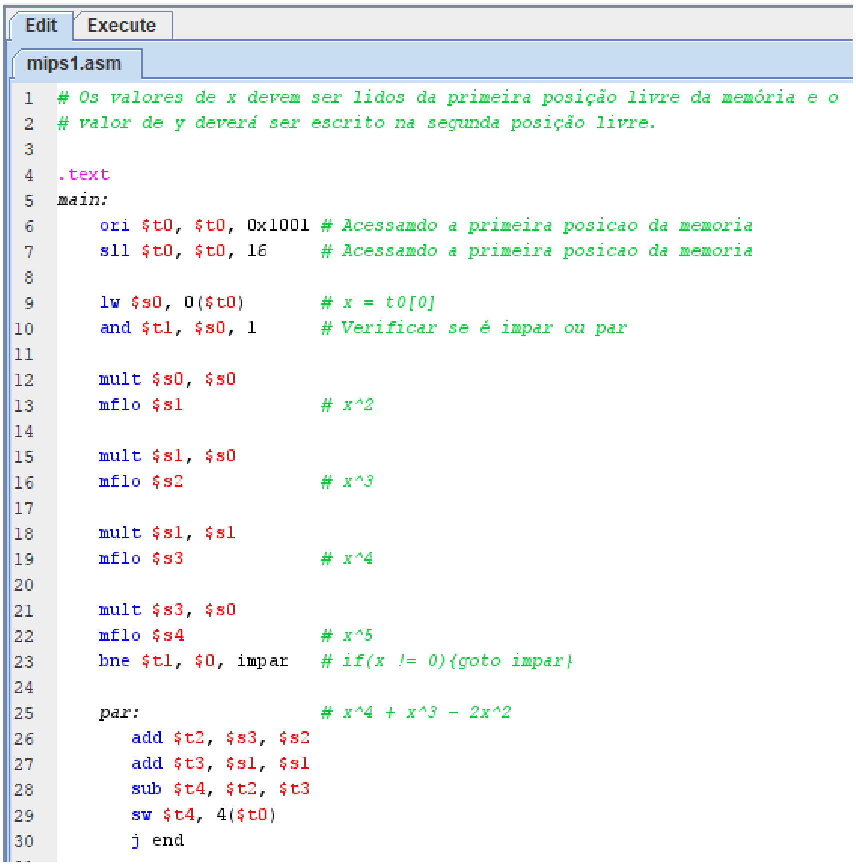
Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000004	0x00000003	0x00000006	0x00000000	0x00000000	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

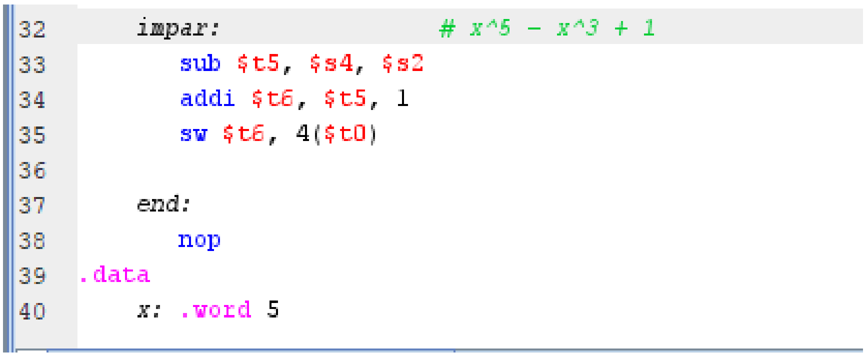
0x10010000 (.data)   ☑ Hexadecimal Addresses   ☑ Hexadecimal Values   □ ASCII

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x10010000
$t1	9	0x00000003
$t2	10	0x00000002
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000001
$t6	14	0x00000001
$t7	15	0x00000001
$s0	16	0x00000004
$s1	17	0x00000003
$s2	18	0x00000006
$s3	19	0x00000000
$s4	20	0x00000000
$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	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400070
hi		0x00000000
lo		0x00000006

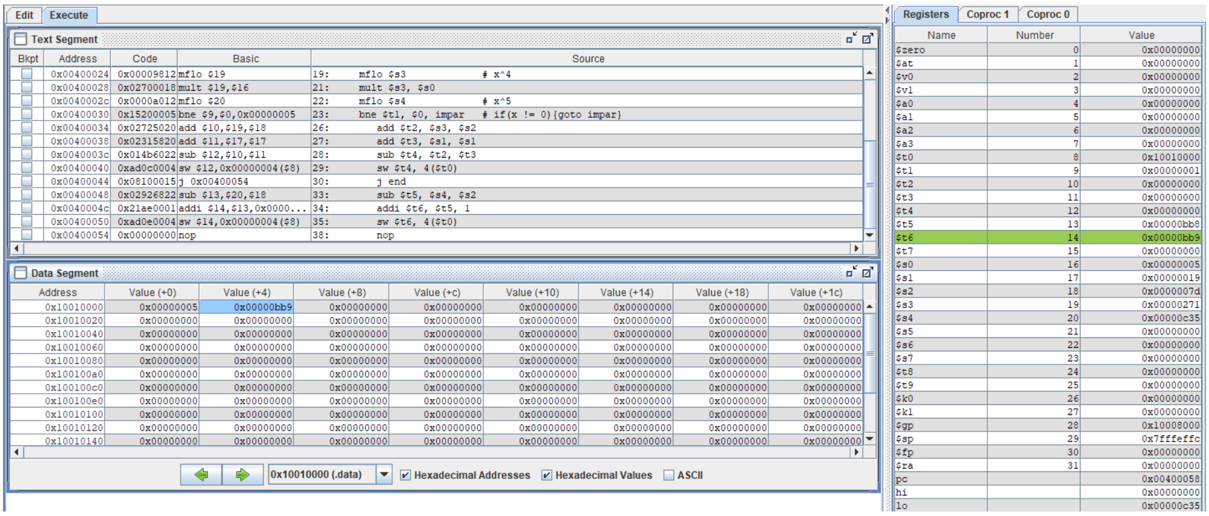
Programa 20:



```
# Os valores de x devem ser lidos da primeira posição livre da memória e o
# valor de y deverá ser escrito na segunda posição livre.

.text
main:
    ori $t0, $t0, 0x1001 # Acessamdo a primeira posicao da memoria
    sll $t0, $t0, 16     # Acessamdo a primeira posicao da memoria

    lw $s0, 0($t0)       # x = t0[0]
    and $t1, $s0, 1      # Verificar se é impar ou par

    mult $s0, $s0
    mflo $s1             # x^2

    mult $s1, $s0
    mflo $s2             # x^3

    mult $s1, $s1
    mflo $s3             # x^4

    mult $s3, $s0
    mflo $s4             # x^5
    bne $t1, $0, impar   # if(x != 0){goto impar}

    par:                 # x^4 + x^3 - 2x^2
        add $t2, $s3, $s2
        add $t3, $s1, $s1
        sub $t4, $t2, $t3
        sw $t4, 4($t0)
        j end
```



```
    impar:               # x^5 - x^3 + 1
        sub $t5, $s4, $s2
        addi $t6, $t5, 1
        sw $t6, 4($t0)

    end:
        nop
.data
    x: .word 5
```



Text Segment

Bkpt	Address	Code	Basic	Source
□	0x00400024	0x00009812	mflo $19	19: mflo $s3 # x^4
□	0x00400028	0x02700018	mult $19,$16	21: mult $s3, $s0
□	0x0040002c	0x0000a012	mflo $20	22: mflo $s4 # x^5
□	0x00400030	0x15200005	bne $9,$0,0x00000005	23: bne $t1, $0, impar # if(x != 0){goto impar}
□	0x00400034	0x02725020	add $10,$19,$18	26: add $t2, $s3, $s2
□	0x00400038	0x02315820	add $11,$17,$17	27: add $t3, $s1, $s1
□	0x0040003c	0x014b6022	sub $12,$10,$11	28: sub $t4, $t2, $t3
□	0x00400040	0xad0c0004	sw $12,0x00000004($8)	29: sw $t4, 4($t0)
□	0x00400044	0x08100015	j 0x00400054	30: j end
□	0x00400048	0x02926822	sub $13,$20,$18	33: sub $t5, $s4, $s2
□	0x0040004c	0x21ae0001	addi $14,$13,0x0000...	34: addi $t6, $t5, 1
□	0x00400050	0xad0e0004	sw $14,0x00000004($8)	35: sw $t6, 4($t0)
□	0x00400054	0x00000000	nop	38: nop

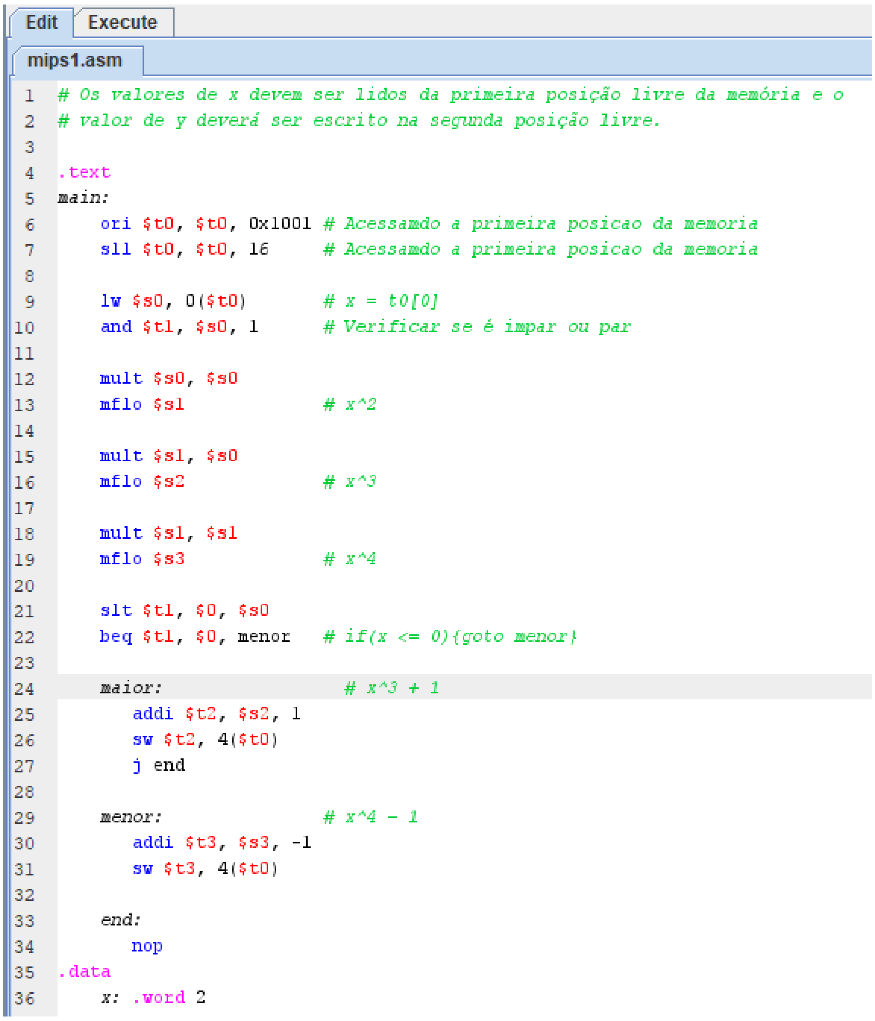
Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000005	0x00000bb9	0x00000000	0x00000000	0x00000000	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

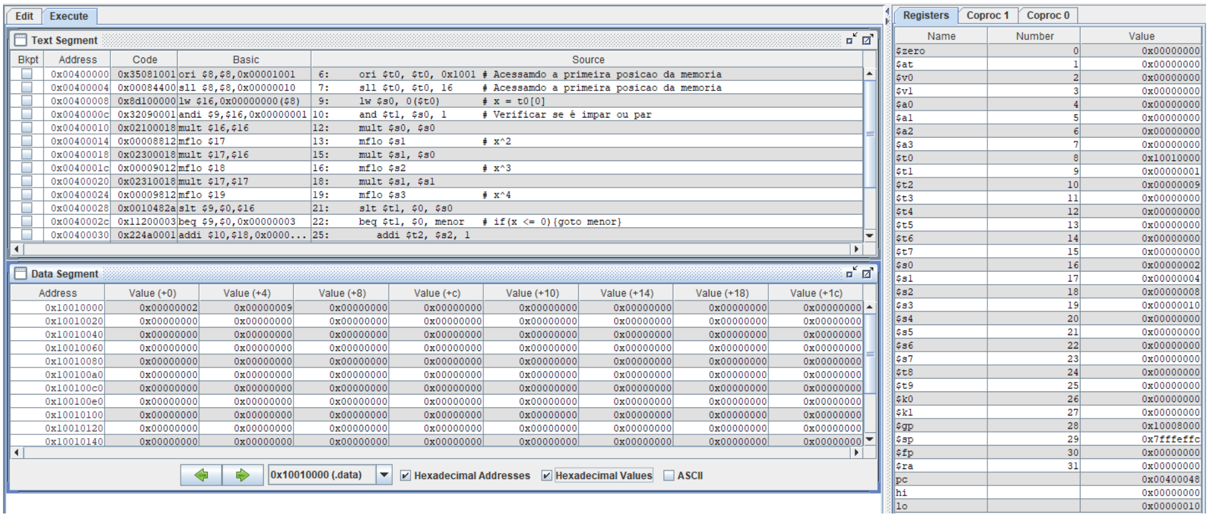
0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values □ ASCII

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x10010000
$t1	9	0x00000001
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000bb8
$t6	14	0x00000bb9
$t7	15	0x00000000
$s0	16	0x00000005
$s1	17	0x00000019
$s2	18	0x0000007d
$s3	19	0x00000271
$s4	20	0x00000c35
$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	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400058
hi		0x00000000
lo		0x00000c35

Programa 21:



```
# Os valores de x devem ser lidos da primeira posição livre da memória e o
# valor de y deverá ser escrito na segunda posição livre.

.text
main:
    ori $t0, $t0, 0x1001 # Acessamdo a primeira posicao da memoria
    sll $t0, $t0, 16     # Acessamdo a primeira posicao da memoria

    lw $s0, 0($t0)       # x = t0[0]
    and $t1, $s0, 1      # Verificar se é impar ou par

    mult $s0, $s0
    mflo $s1             # x^2

    mult $s1, $s0
    mflo $s2             # x^3

    mult $s1, $s1
    mflo $s3             # x^4

    slt $t1, $0, $s0
    beq $t1, $0, menor   # if(x <= 0){goto menor}

    maior:                 # x^3 + 1
       addi $t2, $s2, 1
       sw $t2, 4($t0)
       j end

    menor:               # x^4 - 1
       addi $t3, $s3, -1
       sw $t3, 4($t0)

    end:
       nop
.data
    x: .word 2
```



Bkpt	Address	Code	Basic	Source
□	0x00400000	0x35081001	ori $8,$8,0x00001001	6: ori $t0, $t0, 0x1001 # Acessamdo a primeira posicao da memoria
□	0x00400004	0x00084400	sll $8,$8,0x00000010	7: sll $t0, $t0, 16 # Acessamdo a primeira posicao da memoria
□	0x00400008	0x8d100000	lw $16,0x00000000($8)	9: lw $s0, 0($t0) # x = t0[0]
□	0x0040000c	0x32090001	andi $9,$16,0x00000001	10: and $t1, $s0, 1 # Verificar se é impar ou par
□	0x00400010	0x02100018	mult $16,$16	12: mult $s0, $s0
□	0x00400014	0x00008812	mflo $17	13: mflo $s1 # x^2
□	0x00400018	0x02300018	mult $17,$16	15: mult $s1, $s0
□	0x0040001c	0x00009012	mflo $18	16: mflo $s2 # x^3
□	0x00400020	0x02310018	mult $17,$17	18: mult $s1, $s1
□	0x00400024	0x00009812	mflo $19	19: mflo $s3 # x^4
□	0x00400028	0x0010482a	slt $9,$0,$16	21: slt $t1, $0, $s0
□	0x0040002c	0x11200003	beq $9,$0,0x00000003	22: beq $t1, $0, menor # if(x <= 0){goto menor}
□	0x00400030	0x224a0001	addi $10,$18,0x0000...	25: addi $t2, $s2, 1

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000002	0x00000009	0x00000000	0x00000000	0x00000000	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

0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values □ ASCII

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x10010000
$t1	9	0x00000001
$t2	10	0x00000009
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000002
$s1	17	0x00000004
$s2	18	0x00000008
$s3	19	0x00000010
$s4	20	0x00000000
$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	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400048
hi		0x00000000
lo		0x00000010