Exercício Prático 06

Bruno Braga Guimarães Alves
João Victor Filardi
Laura Caetano Costa
Suzane Lemos de Lima
Vitor Dias De Britto Militão

Parte 1:

Questão 01:

A) um arquivo de texto que contém instruções de linguagem de programação.

Questão 02:

B) uma parte do processador que possui um padrão de bits.

Questão 03:

A) #

Questão 04:

C) 32

Questão 05:

D) parte do processador que contém o endereço da próxima instrução de máquina para ser obtida.

Questão 06:

C) 4

Questão 07:

D) uma declaração que diz ao montador algo sobre o que o programador quer, mas não corresponde diretamente a uma instrução de máquina.

Questão 08:

D) um nome usado no código-fonte em linguagem assembly para um local na memória.

Questão 09:

B) 0x00400000

Questão 10:

A) operando imediato

Questão 11:

B) operação bitwise

Questão 12:

D) Cada um dos registradores deve possuir 32 bit.

Questão 13:

B) Os dados são estendidos em zero à esquerda por 16 bits.

Questão 14:

C) ori $5, $0, 48

Questão 15:

A) Não.

Questão 16:

D)andi $8, $8, 0xFF

Questão 17:

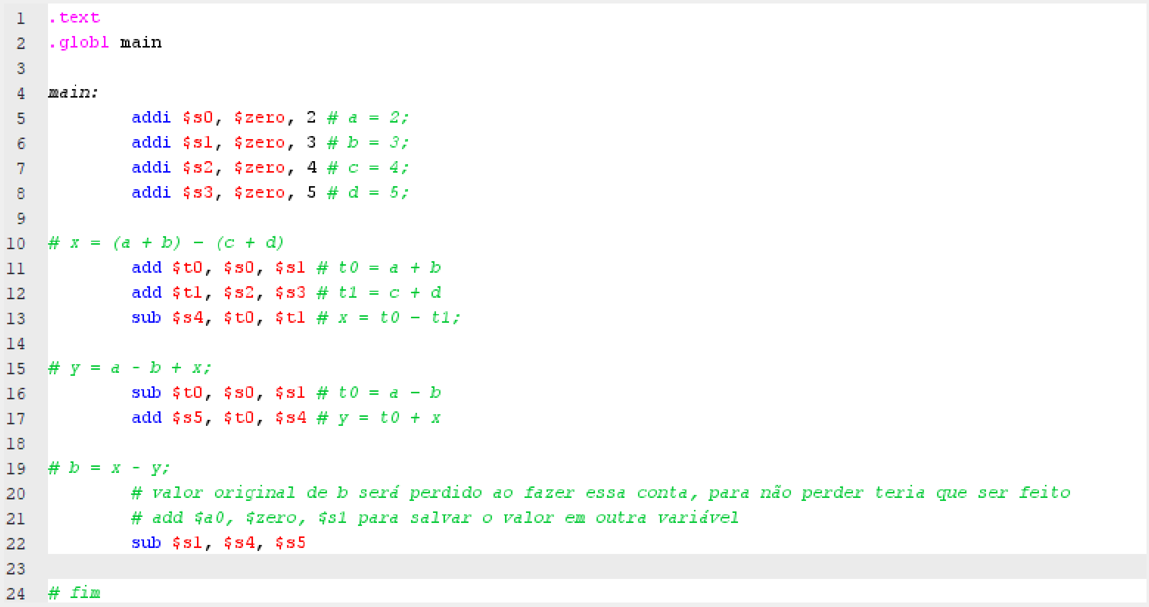
A) Todos os bits em zero.

Questão 18:

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

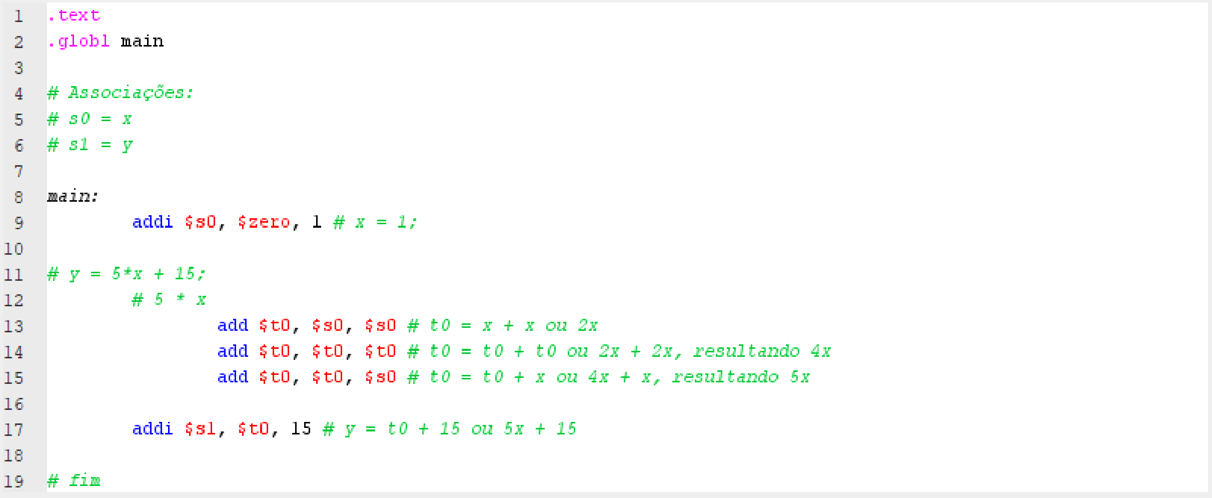
Parte 2: Implementar em MIPS/MARS os seguintes programas:

Programa 1:



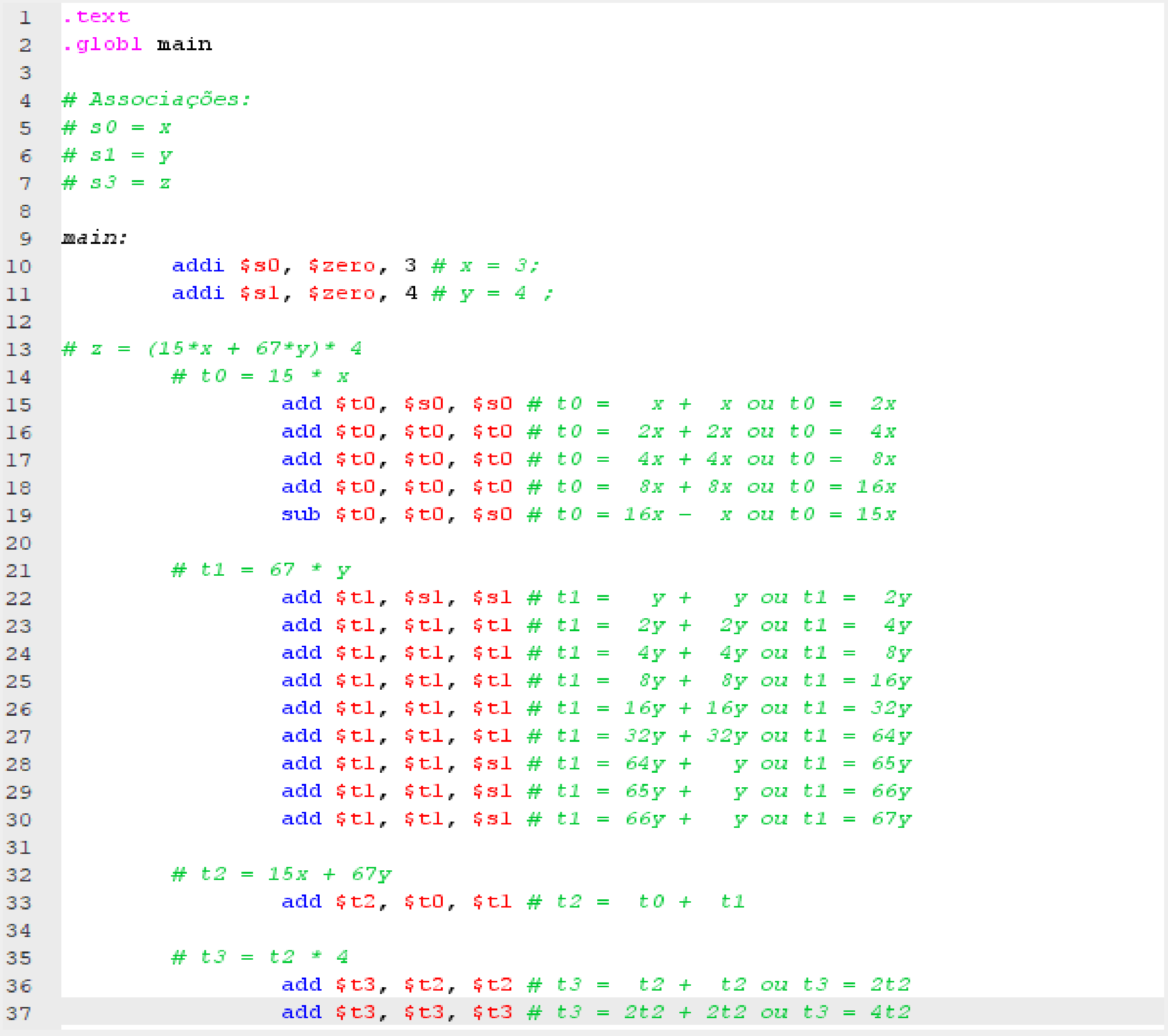
```
.text
.globl main

main:
        addi $s0, $zero, 2 # a = 2;
        addi $s1, $zero, 3 # b = 3;
        addi $s2, $zero, 4 # c = 4;
        addi $s3, $zero, 5 # d = 5;

# x = (a + b) - (c + d)
        add $t0, $s0, $s1 # t0 = a + b
        add $t1, $s2, $s3 # t1 = c + d
        sub $s4, $t0, $t1 # x = t0 - t1;

# y = a - b + x;
        sub $t0, $s0, $s1 # t0 = a - b
        add $s5, $t0, $s4 # y = t0 + x

# b = x - y;
        # valor original de b será perdido ao fazer essa conta, para não perder teria que ser feito
        # add $a0, $zero, $s1 para salvar o valor em outra variável
        sub $s1, $s4, $s5

# fim
```

Programa 2:



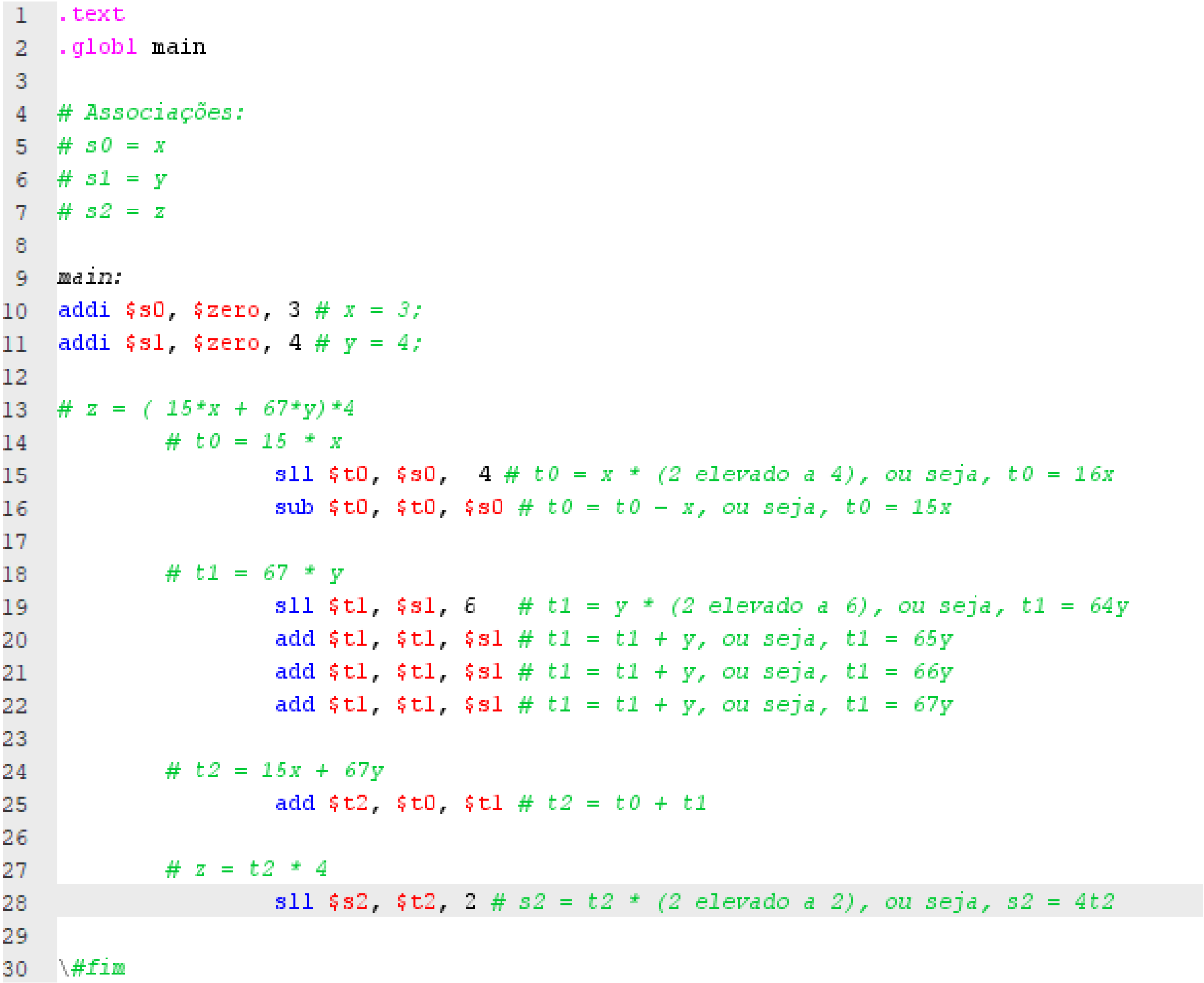
```
.text
.globl main

# Associações:
# s0 = x
# s1 = y

main:
        addi $s0, $zero, 1 # x = 1;

# y = 5*x + 15;
        # 5 * x
                add $t0, $s0, $s0 # t0 = x + x ou 2x
                add $t0, $t0, $t0 # t0 = t0 + t0 ou 2x + 2x, resultando 4x
                add $t0, $t0, $s0 # t0 = t0 + x ou 4x + x, resultando 5x

        addi $s1, $t0, 15 # y = t0 + 15 ou 5x + 15

# fim
```

Programa 3:



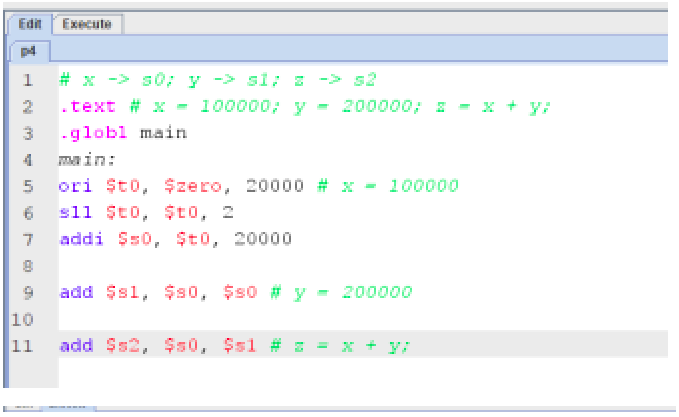
```
.text
.globl main

# Associações:
# s0 = x
# s1 = y
# s3 = z

main:
        addi $s0, $zero, 3 # x = 3;
        addi $s1, $zero, 4 # y = 4 ;

# z = (15*x + 67*y)* 4
        # t0 = 15 * x
                add $t0, $s0, $s0 # t0 =   x +   x ou t0 =  2x
                add $t0, $t0, $t0 # t0 =  2x +  2x ou t0 =  4x
                add $t0, $t0, $t0 # t0 =  4x +  4x ou t0 =  8x
                add $t0, $t0, $t0 # t0 =  8x +  8x ou t0 = 16x
                sub $t0, $t0, $s0 # t0 = 16x -   x ou t0 = 15x

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

        # t2 = 15x + 67y
                add $t2, $t0, $t1 # t2 =  t0 +  t1

        # t3 = t2 * 4
                add $t3, $t2, $t2 # t3 =  t2 +  t2 ou t3 = 2t2
                add $t3, $t3, $t3 # t3 = 2t2 + 2t2 ou t3 = 4t2
```

Programa 4:

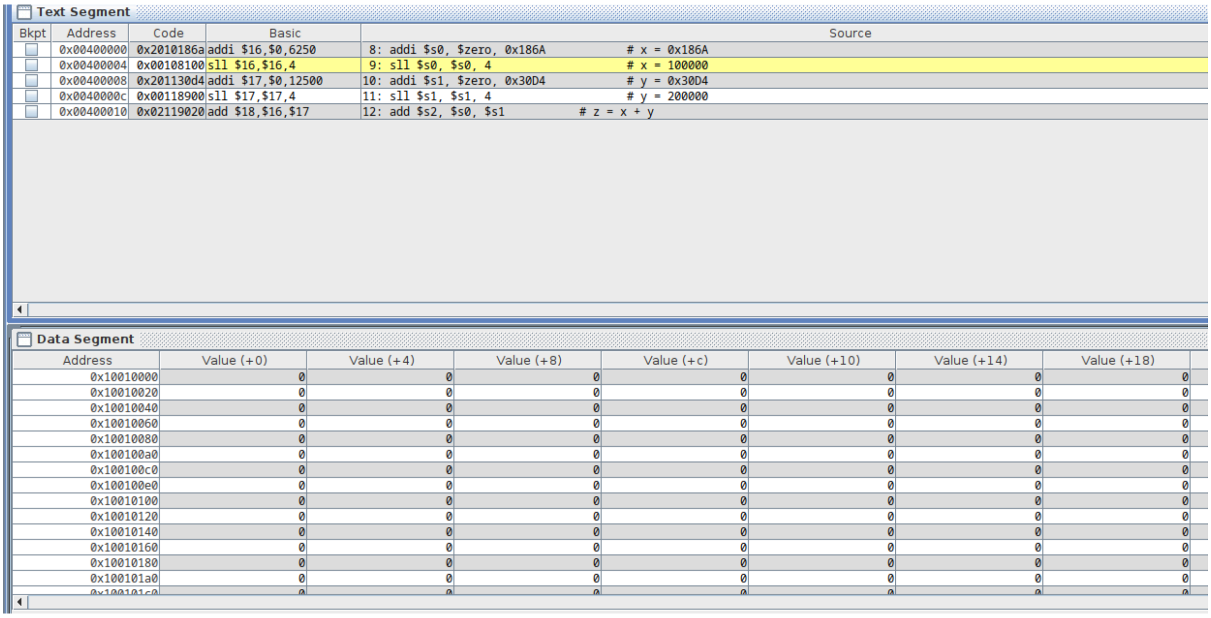


```
.text
.globl main

# Associações:
# s0 = x
# s1 = y
# s2 = z

main:
addi $s0, $zero, 3 # x = 3;
addi $s1, $zero, 4 # y = 4;

# z = ( 15*x + 67*y)*4
        # t0 = 15 * x
                sll $t0, $s0,  4 # t0 = x * (2 elevado a 4), ou seja, t0 = 16x
                sub $t0, $t0, $s0 # t0 = t0 - x, ou seja, t0 = 15x

        # t1 = 67 * y
                sll $t1, $s1, 6   # t1 = y * (2 elevado a 6), ou seja, t1 = 64y
                add $t1, $t1, $s1 # t1 = t1 + y, ou seja, t1 = 65y
                add $t1, $t1, $s1 # t1 = t1 + y, ou seja, t1 = 66y
                add $t1, $t1, $s1 # t1 = t1 + y, ou seja, t1 = 67y

        # t2 = 15x + 67y
                add $t2, $t0, $t1 # t2 = t0 + t1

        # z = t2 * 4
                sll $s2, $t2, 2 # s2 = t2 * (2 elevado a 2), ou seja, s2 = 4t2

\#fim
```

Programa 5:



```
# x -> s0; y -> s1; z -> s2
.text # x = 100000; y = 200000; z = x + y;
.globl main
main:
ori $t0, $zero, 20000 # x = 100000
sll $t0, $t0, 2
addi $s0, $t0, 20000

add $s1, $s0, $s0 # y = 200000

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



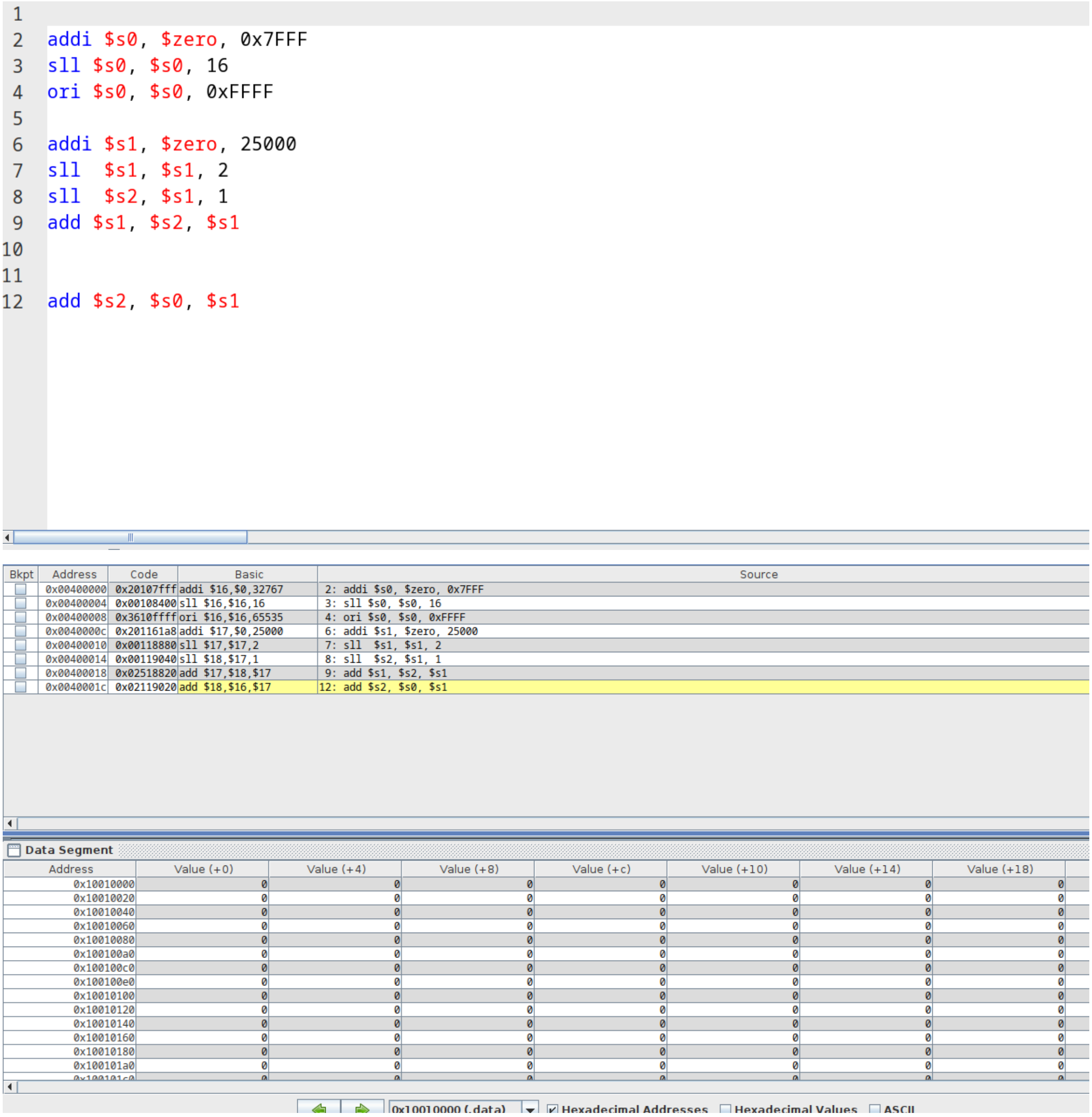
Text Segment

Bkpt	Address	Code	Basic	Source
□	0x00400000	0x2010186a	addi $16,$0,6250	8: addi $s0, $zero, 0x186A # x = 0x186A
□	0x00400004	0x00108100	sll $16,$16,4	9: sll $s0, $s0, 4 # x = 100000
□	0x00400008	0x201130d4	addi $17,$0,12500	10: addi $s1, $zero, 0x30D4 # y = 0x30D4
□	0x0040000c	0x00118900	sll $17,$17,4	11: sll $s1, $s1, 4 # y = 200000
□	0x00400010	0x02119020	add $18,$16,$17	12: add $s2, $s0, $s1 # z = x + y

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)
0x10010000	0	0	0	0	0	0	0
0x10010020	0	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0
0x10010140	0	0	0	0	0	0	0
0x10010160	0	0	0	0	0	0	0
0x10010180	0	0	0	0	0	0	0
0x100101a0	0	0	0	0	0	0	0
0x100101c0	0	0	0	0	0	0	0

Programa 6:



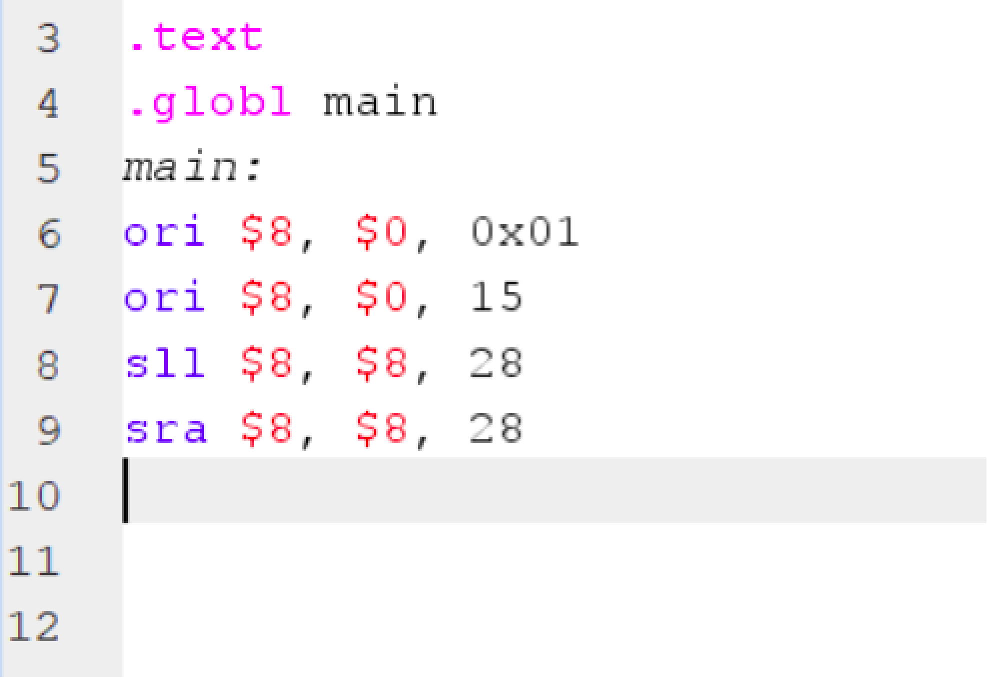
```

addi $s0, $zero, 0x7FFF
sll $s0, $s0, 16
ori $s0, $s0, 0xFFFF

addi $s1, $zero, 25000
sll  $s1, $s1, 2
sll  $s2, $s1, 1
add $s1, $s2, $s1


add $s2, $s0, $s1
```

Bkpt	Address	Code	Basic	Source
□	0x00400000	0x20107fff	addi $16,$0,32767	2: addi $s0, $zero, 0x7FFF
□	0x00400004	0x00108400	sll $16,$16,16	3: sll $s0, $s0, 16
□	0x00400008	0x3610ffff	ori $16,$16,65535	4: ori $s0, $s0, 0xFFFF
□	0x0040000c	0x201161a8	addi $17,$0,25000	6: addi $s1, $zero, 25000
□	0x00400010	0x00118880	sll $17,$17,2	7: sll $s1, $s1, 2
□	0x00400014	0x00119040	sll $18,$17,1	8: sll $s2, $s1, 1
□	0x00400018	0x02518820	add $17,$18,$17	9: add $s1, $s2, $s1
□	0x0040001c	0x02119020	add $18,$16,$17	12: add $s2, $s0, $s1

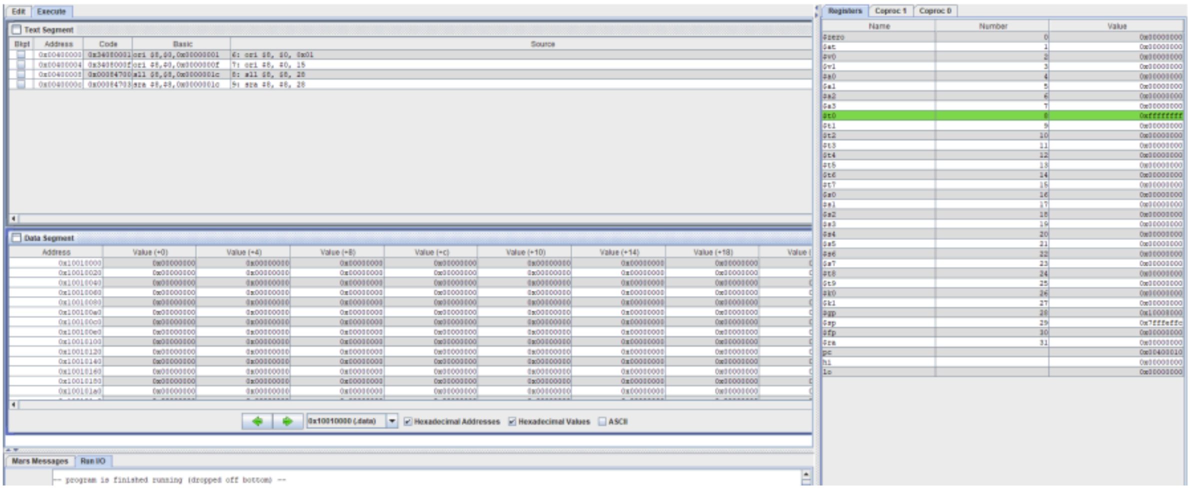
□ Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)
0x10010000	0	0	0	0	0	0	0
0x10010020	0	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0
0x10010140	0	0	0	0	0	0	0
0x10010160	0	0	0	0	0	0	0
0x10010180	0	0	0	0	0	0	0
0x100101a0	0	0	0	0	0	0	0
0x100101c0	0	0	0	0	0	0	0

Programa 7:



```
.text
.globl main
main:
ori $8, $0, 0x01
ori $8, $0, 15
sll $8, $8, 28
sra $8, $8, 28
```



Text Segment

Bkpt	Address	Code	Basic	Source
□	0x00400000	0x34080001	ori $8,$0,0x00000001	6: ori $8, $0, 0x01
□	0x00400004	0x3408000f	ori $8,$0,0x0000000f	7: ori $8, $0, 15
□	0x00400008	0x00084700	sll $8,$8,0x0000001c	8: sll $8, $8, 28
□	0x0040000c	0x00084703	sra $8,$8,0x0000001c	9: sra $8, $8, 28

Data Segment

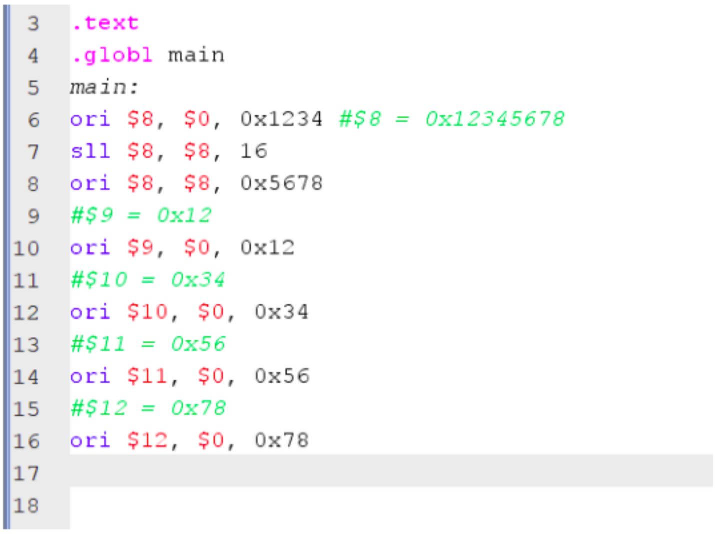
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (
0x10010000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0
0x10010140	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0
0x10010160	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0
0x10010180	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0
0x100101a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0

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

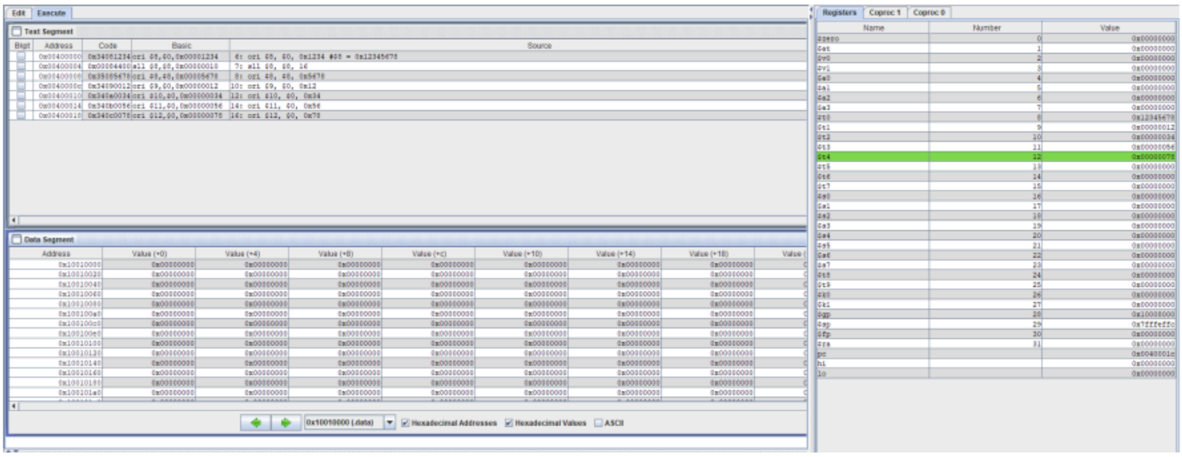
```
-- program is finished running (dropped off bottom) --
```

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:



```
.text
.globl main
main:
ori $8, $0, 0x1234 #$8 = 0x12345678
sll $8, $8, 16
ori $8, $8, 0x5678
#$9 = 0x12
ori $9, $0, 0x12
#$10 = 0x34
ori $10, $0, 0x34
#$11 = 0x56
ori $11, $0, 0x56
#$12 = 0x78
ori $12, $0, 0x78
```



□ Text Segment

Bkpt	Address	Code	Basic	Source
□	0x00400000	0x34081234	ori $8,$0,0x00001234	6: ori $8, $0, 0x1234 #$8 = 0x12345678
□	0x00400004	0x00084400	sll $8,$8,0x00000010	7: sll $8, $8, 16
□	0x00400008	0x35085678	ori $8,$8,0x00005678	8: ori $8, $8, 0x5678
□	0x0040000c	0x34090012	ori $9,$0,0x00000012	10: ori $9, $0, 0x12
□	0x00400010	0x340a0034	ori $10,$0,0x00000034	12: ori $10, $0, 0x34
□	0x00400014	0x340b0056	ori $11,$0,0x00000056	14: ori $11, $0, 0x56
□	0x00400018	0x340c0078	ori $12,$0,0x00000078	16: ori $12, $0, 0x78

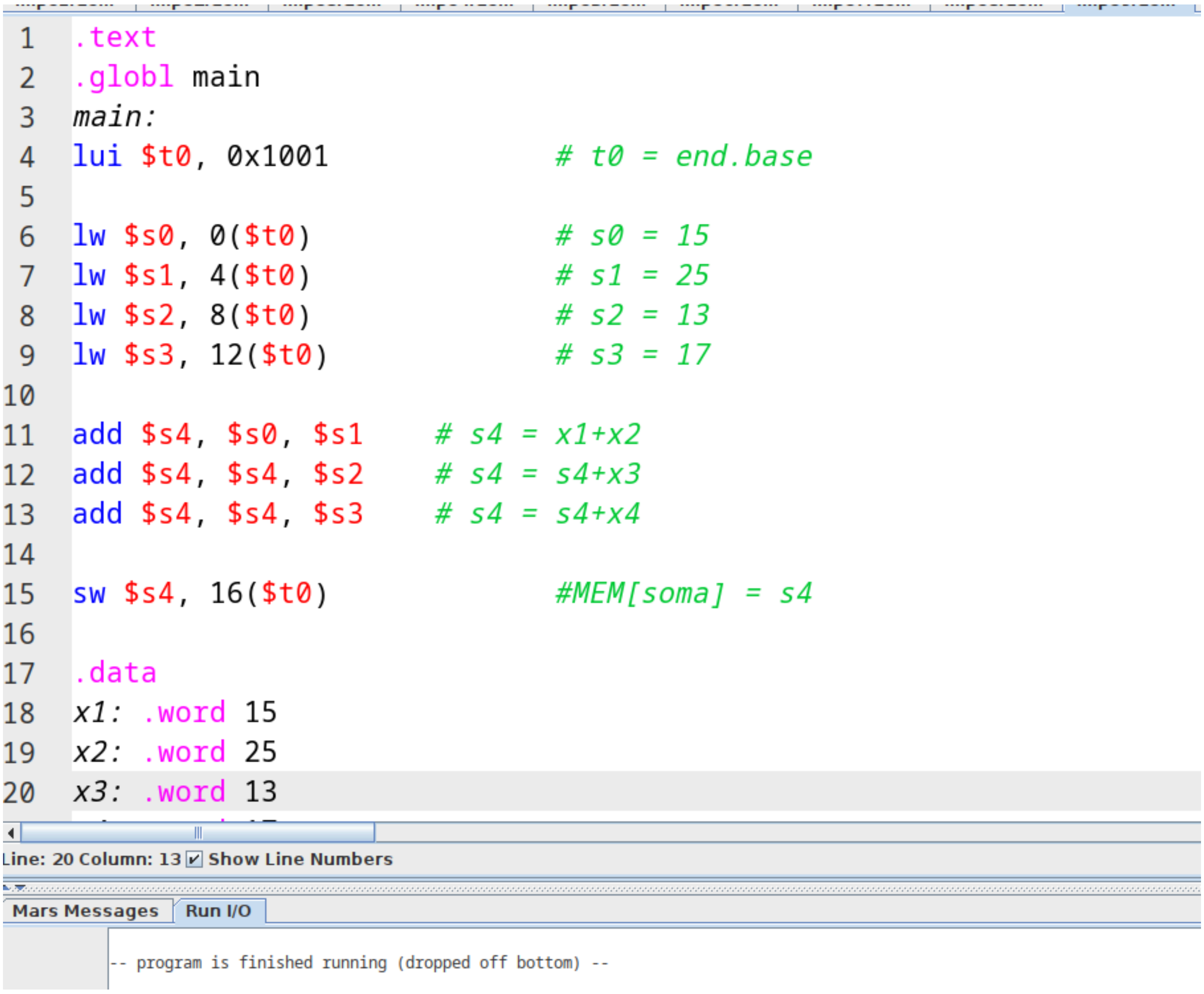
□ Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (
0x10010000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0
0x10010140	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0
0x10010160	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0
0x10010180	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0
0x100101a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0

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		0x0040001c
hi		0x00000000
lo		0x00000000

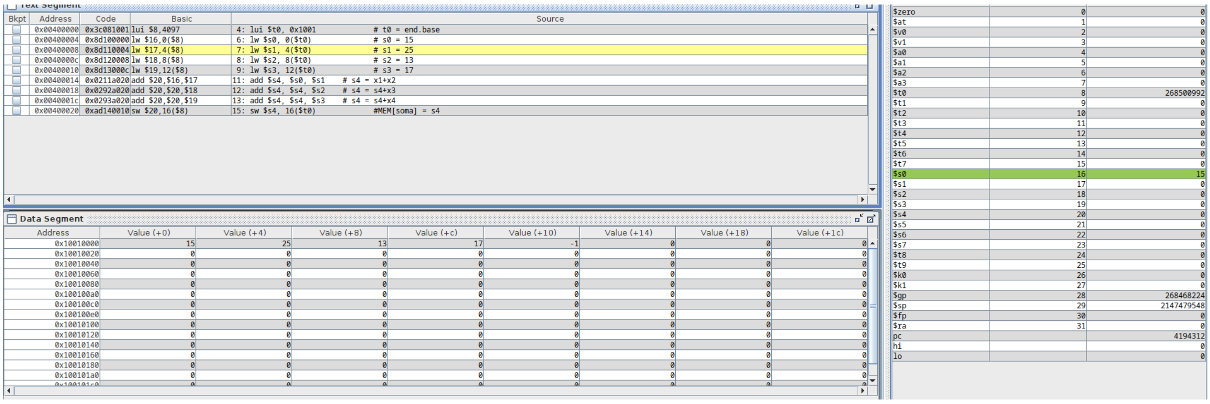
Programa 9:



```
.text
.globl main
main:
lui $t0, 0x1001              # t0 = end.base

lw $s0, 0($t0)               # s0 = 15
lw $s1, 4($t0)               # s1 = 25
lw $s2, 8($t0)               # s2 = 13
lw $s3, 12($t0)              # s3 = 17

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

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

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

Line: 20 Column: 13 ☑ Show Line Numbers

```
-- program is finished running (dropped off bottom) --
```



Text Segment

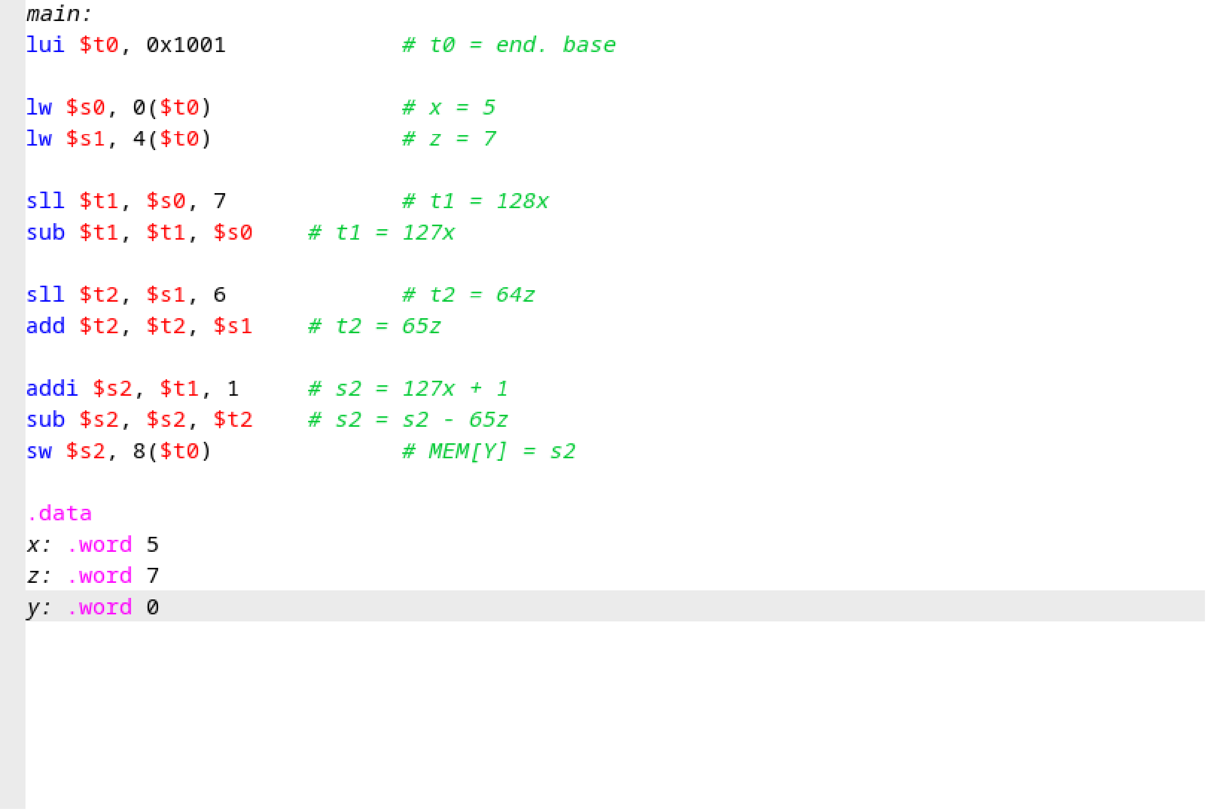
Bkpt	Address	Code	Basic	Source
□	0x00400000	0x3c081001	lui $8,4097	4: lui $t0, 0x1001 # t0 = end.base
□	0x00400004	0x8d100000	lw $16,0($8)	6: lw $s0, 0($t0) # s0 = 15
□	0x00400008	0x8d110004	lw $17,4($8)	7: lw $s1, 4($t0) # s1 = 25
□	0x0040000c	0x8d120008	lw $18,8($8)	8: lw $s2, 8($t0) # s2 = 13
□	0x00400010	0x8d13000c	lw $19,12($8)	9: lw $s3, 12($t0) # s3 = 17
□	0x00400014	0x0211a020	add $20,$16,$17	11: add $s4, $s0, $s1 # s4 = x1+x2
□	0x00400018	0x0292a020	add $20,$20,$18	12: add $s4, $s4, $s2 # s4 = s4+x3
□	0x0040001c	0x0293a020	add $20,$20,$19	13: add $s4, $s4, $s3 # s4 = s4+x4
□	0x00400020	0xad140010	sw $20,16($8)	15: sw $s4, 16($t0) #MEM[soma] = s4

Data Segment

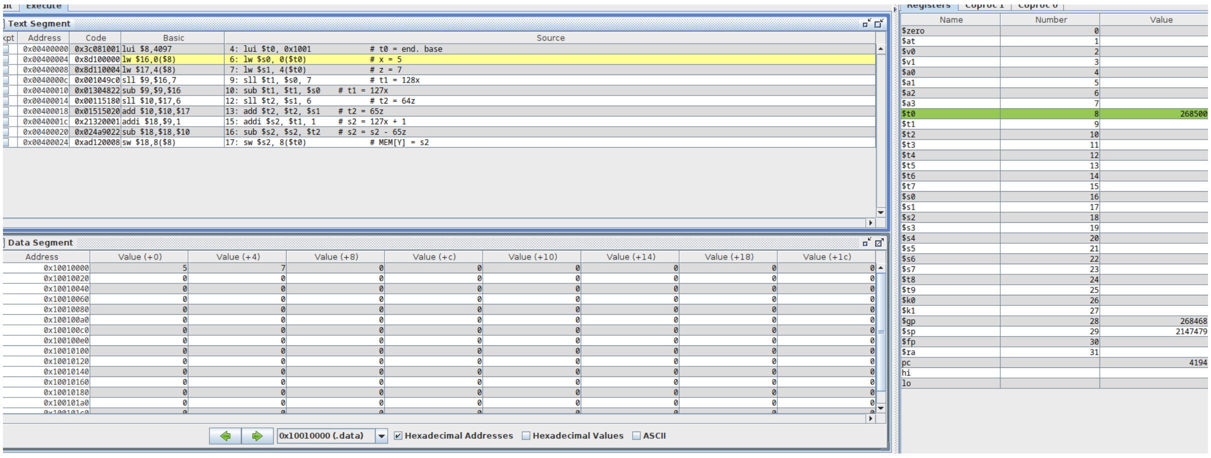
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	15	25	13	17	-1	0	0	0
0x10010020	0	0	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	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
0x10010160	0	0	0	0	0	0	0	0
0x10010180	0	0	0	0	0	0	0	0
0x100101a0	0	0	0	0	0	0	0	0
0x100101c0	0	0	0	0	0	0	0	0

$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	0
$t2	10	0
$t3	11	0
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	15
$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		4194312
hi		0
lo		0

Programa 10:



```
main:
lui $t0, 0x1001              # t0 = end. base

lw $s0, 0($t0)               # x = 5
lw $s1, 4($t0)               # z = 7

sll $t1, $s0, 7              # t1 = 128x
sub $t1, $t1, $s0     # t1 = 127x

sll $t2, $s1, 6              # t2 = 64z
add $t2, $t2, $s1     # t2 = 65z

addi $s2, $t1, 1      # s2 = 127x + 1
sub $s2, $s2, $t2     # s2 = s2 - 65z
sw $s2, 8($t0)               # MEM[Y] = s2

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



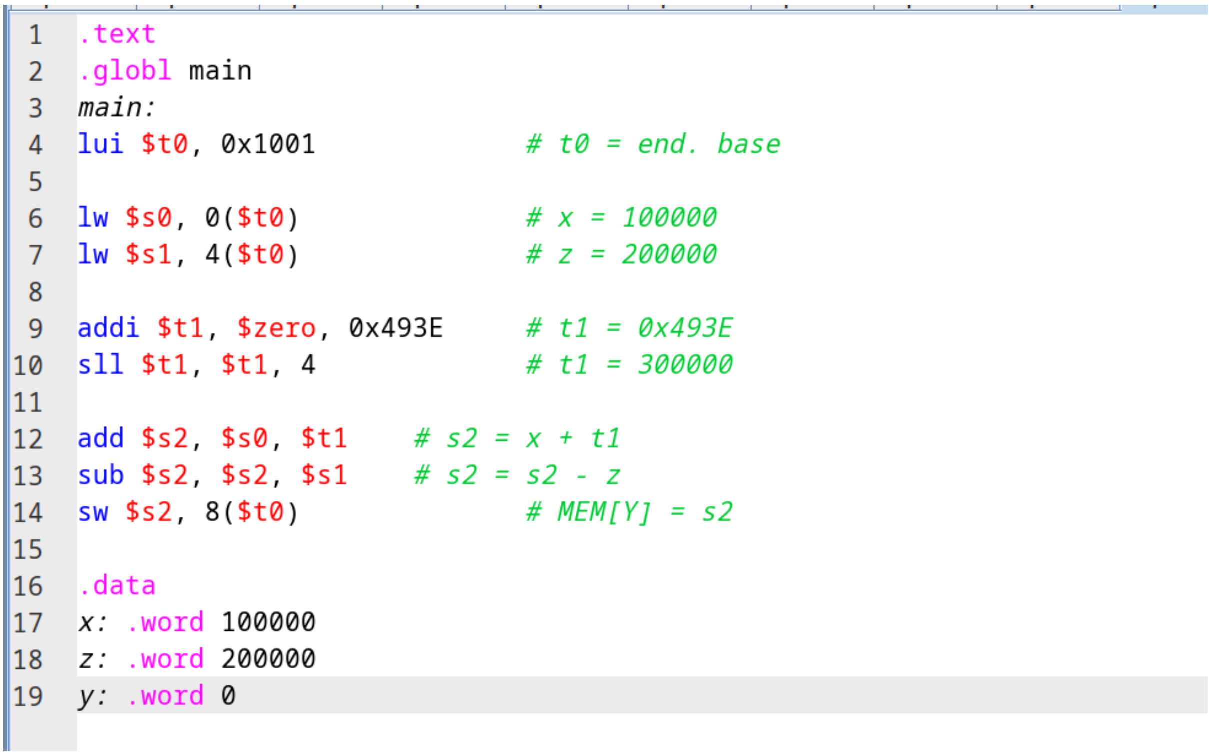
Bkpt	Address	Code	Basic	Source
□	0x00400000	0x3c081001	lui $8,4097	4: lui $t0, 0x1001 # t0 = end. base
□	0x00400004	0x8d100000	lw $16,0($8)	6: lw $s0, 0($t0) # x = 5
□	0x00400008	0x8d110004	lw $17,4($8)	7: lw $s1, 4($t0) # z = 7
□	0x0040000c	0x001049c0	sll $9,$16,7	9: sll $t1, $s0, 7 # t1 = 128x
□	0x00400010	0x01304822	sub $9,$9,$16	10: sub $t1, $t1, $s0 # t1 = 127x
□	0x00400014	0x00115180	sll $10,$17,6	12: sll $t2, $s1, 6 # t2 = 64z
□	0x00400018	0x01515020	add $10,$10,$17	13: add $t2, $t2, $s1 # t2 = 65z
□	0x0040001c	0x21320001	addi $18,$9,1	15: addi $s2, $t1, 1 # s2 = 127x + 1
□	0x00400020	0x024a9022	sub $18,$18,$10	16: sub $s2, $s2, $t2 # s2 = s2 - 65z
□	0x00400024	0xad120008	sw $18,8($8)	17: sw $s2, 8($t0) # MEM[Y] = s2

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	5	7	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
0x10010160	0	0	0	0	0	0	0	0
0x10010180	0	0	0	0	0	0	0	0
0x100101a0	0	0	0	0	0	0	0	0

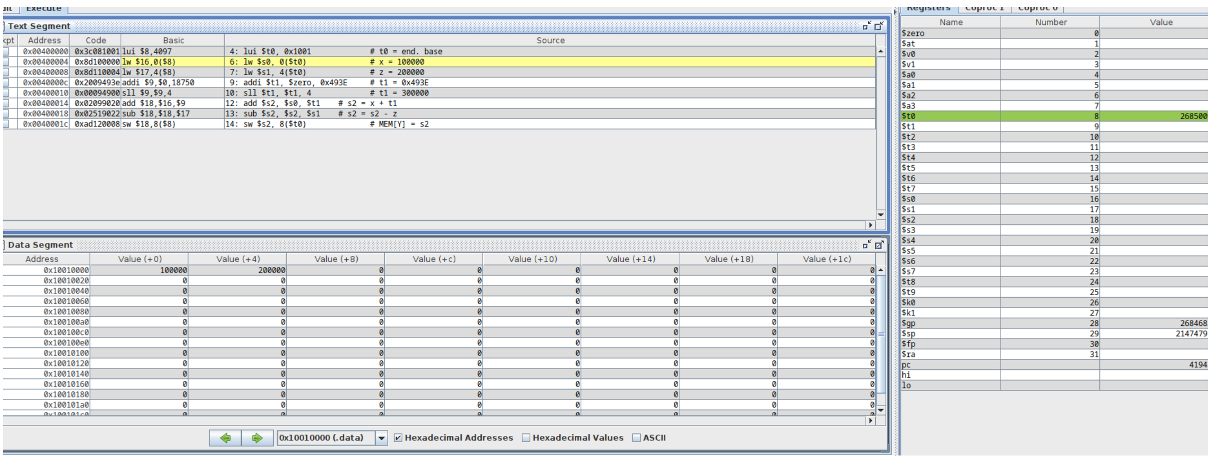
0x10010000 (.data) ☑ Hexadecimal Addresses □ Hexadecimal Values □ ASCII

Name	Number	Value
$zero	0	
$at	1	
$v0	2	
$v1	3	
$a0	4	
$a1	5	
$a2	6	
$a3	7	
$t0	8	268500
$t1	9	
$t2	10	
$t3	11	
$t4	12	
$t5	13	
$t6	14	
$t7	15	
$s0	16	
$s1	17	
$s2	18	
$s3	19	
$s4	20	
$s5	21	
$s6	22	
$s7	23	
$t8	24	
$t9	25	
$k0	26	
$k1	27	
$gp	28	268468
$sp	29	2147479
$fp	30	
$ra	31	
pc		4194
hi		
lo		

Programa 11:



```
.text
.globl main
main:
lui $t0, 0x1001              # t0 = end. base

lw $s0, 0($t0)               # x = 100000
lw $s1, 4($t0)               # z = 200000

addi $t1, $zero, 0x493E      # t1 = 0x493E
sll $t1, $t1, 4              # t1 = 300000

add $s2, $s0, $t1      # s2 = x + t1
sub $s2, $s2, $s1      # s2 = s2 - z
sw $s2, 8($t0)               # MEM[Y] = s2

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



Text Segment

Bkpt	Address	Code	Basic	Source
☐	0x00400000	0x3c081001	lui $8,4097	4: lui $t0, 0x1001 # t0 = end. base
☐	0x00400004	0x8d100000	lw $16,0($8)	6: lw $s0, 0($t0) # x = 100000
☐	0x00400008	0x8d110004	lw $17,4($8)	7: lw $s1, 4($t0) # z = 200000
☐	0x0040000c	0x2009493e	addi $9,$0,18750	9: addi $t1, $zero, 0x493E # t1 = 0x493E
☐	0x00400010	0x00094900	sll $9,$9,4	10: sll $t1, $t1, 4 # t1 = 300000
☐	0x00400014	0x02099020	add $18,$16,$9	12: add $s2, $s0, $t1 # s2 = x + t1
☐	0x00400018	0x02519022	sub $18,$18,$17	13: sub $s2, $s2, $s1 # s2 = s2 - z
☐	0x0040001c	0xad120008	sw $18,8($8)	14: sw $s2, 8($t0) # MEM[Y] = s2

Data Segment

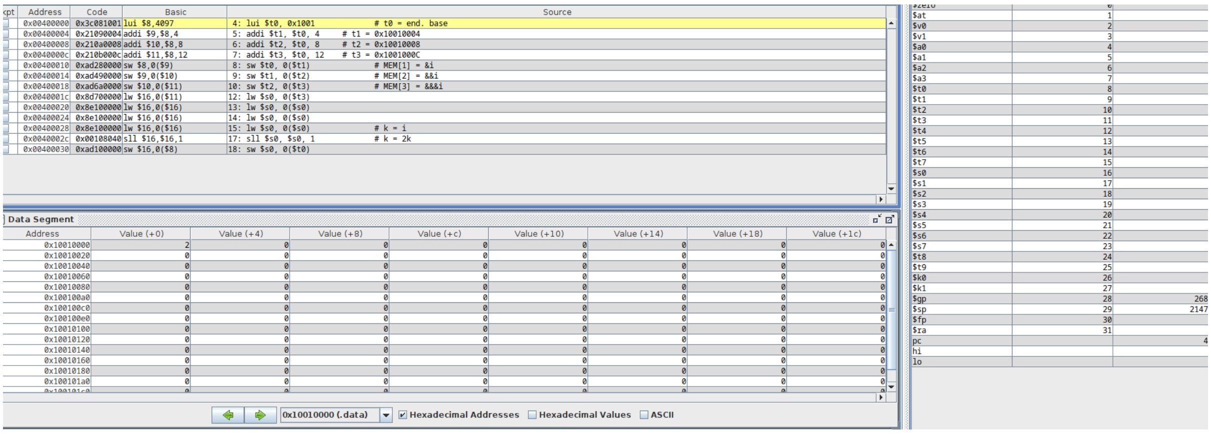
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	100000	200000	0	0	0	0	0	0
0x10010020	0	0	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	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
0x10010160	0	0	0	0	0	0	0	0
0x10010180	0	0	0	0	0	0	0	0
0x100101a0	0	0	0	0	0	0	0	0

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

Name	Number	Value
$zero	0	
$at	1	
$v0	2	
$v1	3	
$a0	4	
$a1	5	
$a2	6	
$a3	7	
$t0	8	268500
$t1	9	
$t2	10	
$t3	11	
$t4	12	
$t5	13	
$t6	14	
$t7	15	
$s0	16	
$s1	17	
$s2	18	
$s3	19	
$s4	20	
$s5	21	
$s6	22	
$s7	23	
$t8	24	
$t9	25	
$k0	26	
$k1	27	
$gp	28	268468
$sp	29	2147479
$fp	30	
$ra	31	
pc		4194
hi		
lo		

Programa 12:

```
.text
.globl main
main:
lui $t0, 0x1001              # t0 = end. base
addi $t1, $t0, 4      # t1 = 0x10010004
addi $t2, $t0, 8      # t2 = 0x10010008
addi $t3, $t0, 12     # t3 = 0x1001000C
sw $t0, 0($t1)               # MEM[1] = &i
sw $t1, 0($t2)               # MEM[2] = &&i
sw $t2, 0($t3)               # MEM[3] = &&&i

lw $s0, 0($t3)
lw $s0, 0($s0)
lw $s0, 0($s0)
lw $s0, 0($s0)               # k = i

sll $s0, $s0, 1              # k = 2k
sw $s0, 0($t0)

.data
i: .word 2
```



kpt	Address	Code	Basic	Source
□	0x00400000	0x3c081001	lui $8,4097	4: lui $t0, 0x1001 # t0 = end. base
□	0x00400004	0x21090004	addi $9,$8,4	5: addi $t1, $t0, 4 # t1 = 0x10010004
□	0x00400008	0x210a0008	addi $10,$8,8	6: addi $t2, $t0, 8 # t2 = 0x10010008
□	0x0040000c	0x210b000c	addi $11,$8,12	7: addi $t3, $t0, 12 # t3 = 0x1001000C
□	0x00400010	0xad280000	sw $8,0($9)	8: sw $t0, 0($t1) # MEM[1] = &i
□	0x00400014	0xad490000	sw $9,0($10)	9: sw $t1, 0($t2) # MEM[2] = &&i
□	0x00400018	0xad6a0000	sw $10,0($11)	10: sw $t2, 0($t3) # MEM[3] = &&&i
□	0x0040001c	0x8d700000	lw $16,0($11)	12: lw $s0, 0($t3)
□	0x00400020	0x8e100000	lw $16,0($16)	13: lw $s0, 0($s0)
□	0x00400024	0x8e100000	lw $16,0($16)	14: lw $s0, 0($s0)
□	0x00400028	0x8e100000	lw $16,0($16)	15: lw $s0, 0($s0) # k = i
□	0x0040002c	0x00108040	sll $16,$16,1	17: sll $s0, $s0, 1 # k = 2k
□	0x00400030	0xad100000	sw $16,0($8)	18: sw $s0, 0($t0)

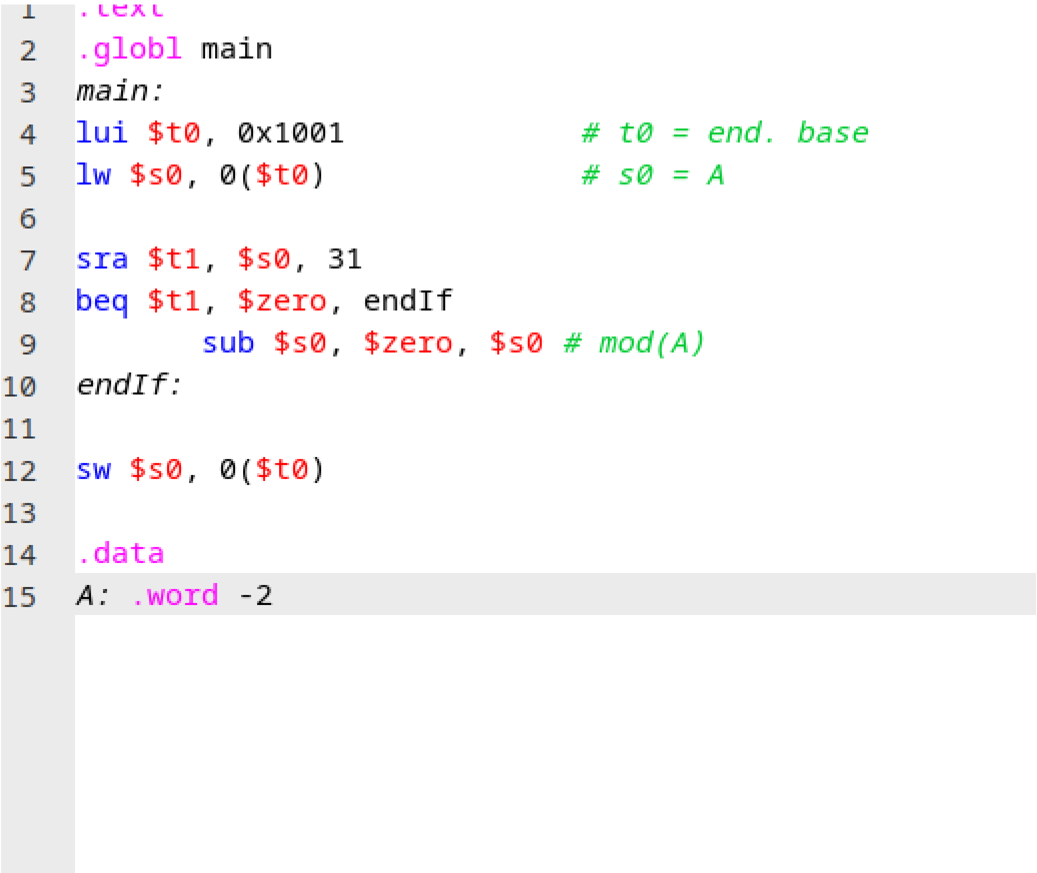
Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	2	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
0x10010160	0	0	0	0	0	0	0	0
0x10010180	0	0	0	0	0	0	0	0
0x100101a0	0	0	0	0	0	0	0	0

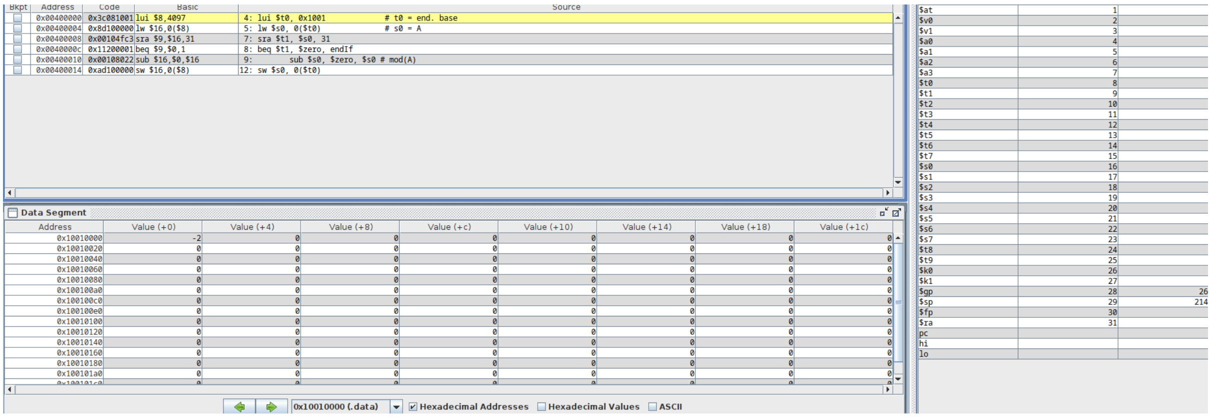
0x10010000 (.data) ☑ Hexadecimal Addresses □ Hexadecimal Values □ ASCII

$zero	0	
$at	1	
$v0	2	
$v1	3	
$a0	4	
$a1	5	
$a2	6	
$a3	7	
$t0	8	
$t1	9	
$t2	10	
$t3	11	
$t4	12	
$t5	13	
$t6	14	
$t7	15	
$s0	16	
$s1	17	
$s2	18	
$s3	19	
$s4	20	
$s5	21	
$s6	22	
$s7	23	
$t8	24	
$t9	25	
$k0	26	
$k1	27	
$gp	28	268
$sp	29	2147
$fp	30	
$ra	31	
pc		4
hi		
lo		

Programa 13:



```
.text
.globl main
main:
lui $t0, 0x1001             # t0 = end. base
lw $s0, 0($t0)              # s0 = A

sra $t1, $s0, 31
beq $t1, $zero, endIf
        sub $s0, $zero, $s0 # mod(A)
endIf:

sw $s0, 0($t0)

.data
A: .word -2
```



Bkpt	Address	Code	Basic	Source
□	0x00400000	0x3c081001	lui $8,4097	4: lui $t0, 0x1001 # t0 = end. base
□	0x00400004	0x8d100000	lw $16,0($8)	5: lw $s0, 0($t0) # s0 = A
□	0x00400008	0x00104fc3	sra $9,$16,31	7: sra $t1, $s0, 31
□	0x0040000c	0x11200001	beq $9,$0,1	8: beq $t1, $zero, endIf
□	0x00400010	0x00108022	sub $16,$0,$16	9: sub $s0, $zero, $s0 # mod(A)
□	0x00400014	0xad100000	sw $16,0($8)	12: sw $s0, 0($t0)

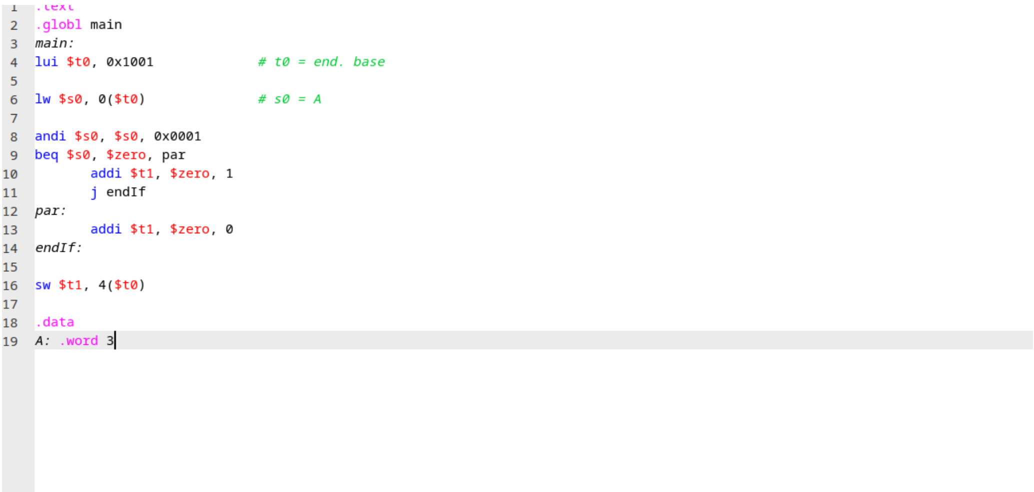
Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	-2	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
0x10010160	0	0	0	0	0	0	0	0
0x10010180	0	0	0	0	0	0	0	0
0x100101a0	0	0	0	0	0	0	0	0
0x100101c0	0	0	0	0	0	0	0	0

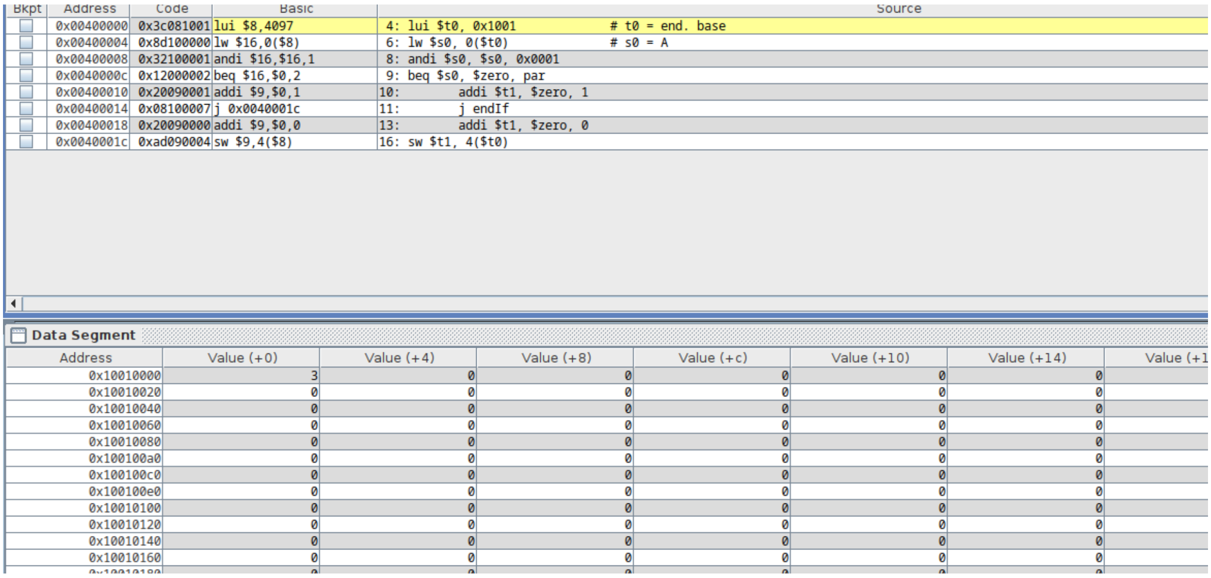
0x10010000 (.data) ☑ Hexadecimal Addresses □ Hexadecimal Values □ ASCII

$at	1	
$v0	2	
$v1	3	
$a0	4	
$a1	5	
$a2	6	
$a3	7	
$t0	8	
$t1	9	
$t2	10	
$t3	11	
$t4	12	
$t5	13	
$t6	14	
$t7	15	
$s0	16	
$s1	17	
$s2	18	
$s3	19	
$s4	20	
$s5	21	
$s6	22	
$s7	23	
$t8	24	
$t9	25	
$k0	26	
$k1	27	
$gp	28	26
$sp	29	214
$fp	30	
$ra	31	
pc		
hi		
lo		

Programa 14:



```
.text
.globl main
main:
lui $t0, 0x1001              # t0 = end. base

lw $s0, 0($t0)               # s0 = A

andi $s0, $s0, 0x0001
beq $s0, $zero, par
       addi $t1, $zero, 1
       j endIf
par:
       addi $t1, $zero, 0
endIf:

sw $t1, 4($t0)

.data
A: .word 3
```

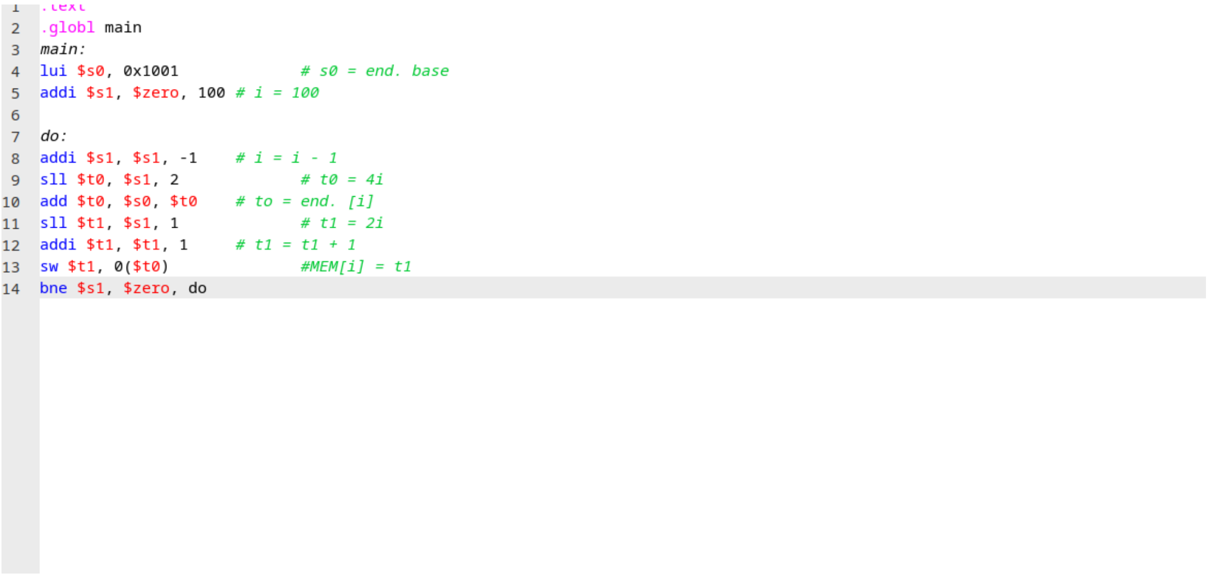


Bkpt	Address	Code	Basic	Source
□	0x00400000	0x3c081001	lui $8,4097	4: lui $t0, 0x1001 # t0 = end. base
□	0x00400004	0x8d100000	lw $16,0($8)	6: lw $s0, 0($t0) # s0 = A
□	0x00400008	0x32100001	andi $16,$16,1	8: andi $s0, $s0, 0x0001
□	0x0040000c	0x12000002	beq $16,$0,2	9: beq $s0, $zero, par
□	0x00400010	0x20090001	addi $9,$0,1	10: addi $t1, $zero, 1
□	0x00400014	0x08100007	j 0x0040001c	11: j endIf
□	0x00400018	0x20090000	addi $9,$0,0	13: addi $t1, $zero, 0
□	0x0040001c	0xad090004	sw $9,4($8)	16: sw $t1, 4($t0)

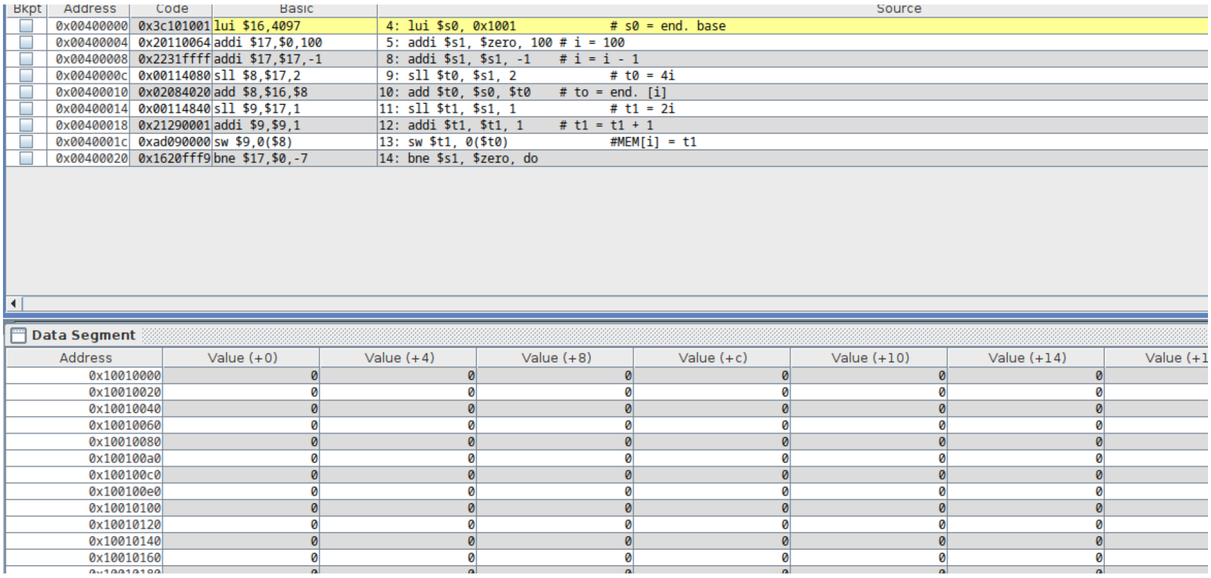
Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+1
0x10010000	3	0	0	0	0	0	
0x10010020	0	0	0	0	0	0	
0x10010040	0	0	0	0	0	0	
0x10010060	0	0	0	0	0	0	
0x10010080	0	0	0	0	0	0	
0x100100a0	0	0	0	0	0	0	
0x100100c0	0	0	0	0	0	0	
0x100100e0	0	0	0	0	0	0	
0x10010100	0	0	0	0	0	0	
0x10010120	0	0	0	0	0	0	
0x10010140	0	0	0	0	0	0	
0x10010160	0	0	0	0	0	0	
0x10010180	0	0	0	0	0	0	

Programa 15:



```
.text
.globl main
main:
lui $s0, 0x1001              # s0 = end. base
addi $s1, $zero, 100 # i = 100

do:
addi $s1, $s1, -1      # i = i - 1
sll $t0, $s1, 2              # t0 = 4i
add $t0, $s0, $t0      # to = end. [i]
sll $t1, $s1, 1              # t1 = 2i
addi $t1, $t1, 1       # t1 = t1 + 1
sw $t1, 0($t0)               #MEM[i] = t1
bne $s1, $zero, do
```

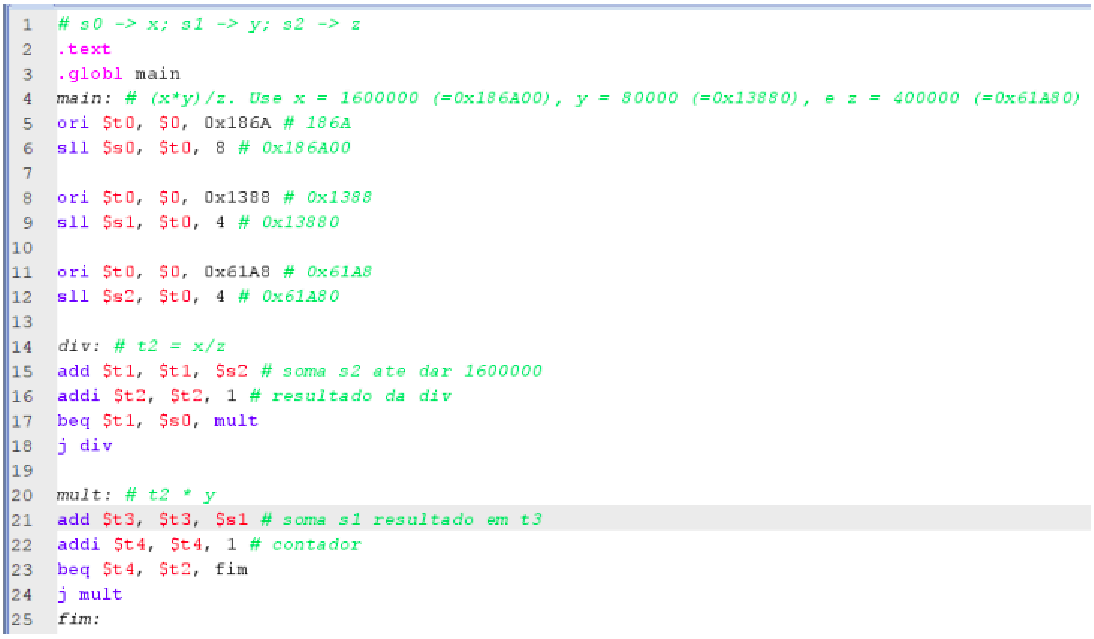


```
Bkpt  Address     Code        Basic                     Source
      0x00400000  0x3c101001  lui $16,4097              4:  lui $s0, 0x1001          # s0 = end. base
      0x00400004  0x20110064  addi $17,$0,100           5:  addi $s1, $zero, 100 # i = 100
      0x00400008  0x2231ffff  addi $17,$17,-1           8:  addi $s1, $s1, -1     # i = i - 1
      0x0040000c  0x00114080  sll $8,$17,2              9:  sll $t0, $s1, 2          # t0 = 4i
      0x00400010  0x02084020  add $8,$16,$8             10: add $t0, $s0, $t0     # to = end. [i]
      0x00400014  0x00114840  sll $9,$17,1              11: sll $t1, $s1, 1          # t1 = 2i
      0x00400018  0x21290001  addi $9,$9,1              12: addi $t1, $t1, 1      # t1 = t1 + 1
      0x0040001c  0xad090000  sw $9,0($8)               13: sw $t1, 0($t0)           #MEM[i] = t1
      0x00400020  0x1620fff9  bne $17,$0,-7             14: bne $s1, $zero, do
```

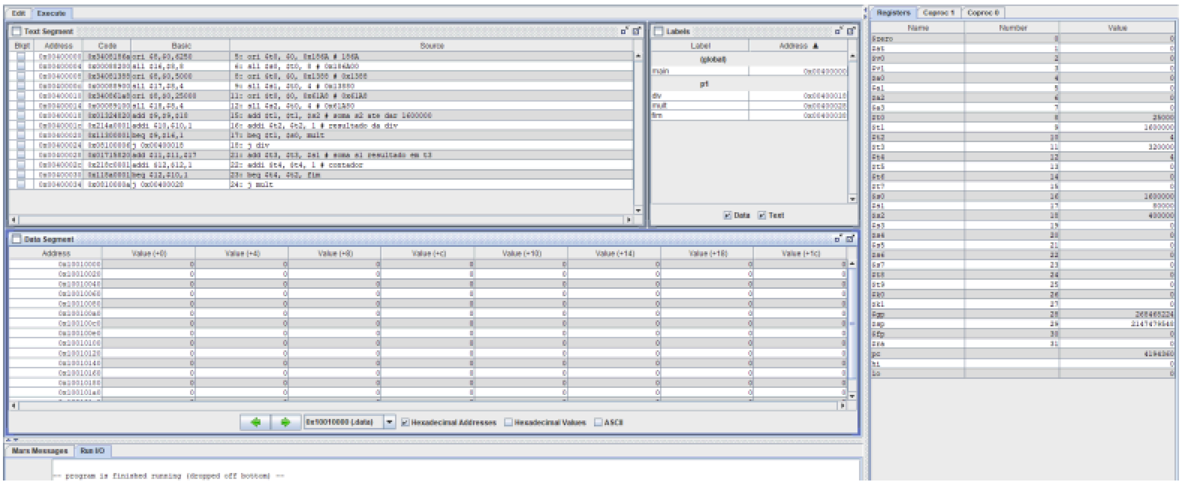
Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+1
0x10010000	0	0	0	0	0	0	
0x10010020	0	0	0	0	0	0	
0x10010040	0	0	0	0	0	0	
0x10010060	0	0	0	0	0	0	
0x10010080	0	0	0	0	0	0	
0x100100a0	0	0	0	0	0	0	
0x100100c0	0	0	0	0	0	0	
0x100100e0	0	0	0	0	0	0	
0x10010100	0	0	0	0	0	0	
0x10010120	0	0	0	0	0	0	
0x10010140	0	0	0	0	0	0	
0x10010160	0	0	0	0	0	0	

Programa 16:



```
# s0 -> x; s1 -> y; s2 -> z
.text
.globl main
main: # (x*y)/z. Use x = 1600000 (=0x186A00), y = 80000 (=0x13880), e z = 400000 (=0x61A80)
ori $t0, $0, 0x186A # 186A
sll $s0, $t0, 8 # 0x186A00

ori $t0, $0, 0x1388 # 0x1388
sll $s1, $t0, 4 # 0x13880

ori $t0, $0, 0x61A8 # 0x61A8
sll $s2, $t0, 4 # 0x61A80

div: # t2 = x/z
add $t1, $t1, $s2 # soma s2 ate dar 1600000
addi $t2, $t2, 1 # resultado da div
beq $t1, $s0, mult
j div

mult: # t2 * y
add $t3, $t3, $s1 # soma s1 resultado em t3
addi $t4, $t4, 1 # contador
beq $t4, $t2, fim
j mult
fim:
```



```
Bkpt  Address     Code        Basic                 Source
      0x00400000  0x3408186a  ori $8,$0,6250        5:  ori $t0, $0, 0x186A # 186A
      0x00400004  0x00088200  sll $16,$8,8          6:  sll $s0, $t0, 8 # 0x186A00
      0x00400008  0x34081388  ori $8,$0,5000        8:  ori $t0, $0, 0x1388 # 0x1388
      0x0040000c  0x00088900  sll $17,$8,4          9:  sll $s1, $t0, 4 # 0x13880
      0x00400010  0x340861a8  ori $8,$0,25000       11: ori $t0, $0, 0x61A8 # 0x61A8
      0x00400014  0x00089100  sll $18,$8,4          12: sll $s2, $t0, 4 # 0x61A80
      0x00400018  0x01324820  add $9,$9,$18         15: add $t1, $t1, $s2 # soma s2 ate dar 1600000
      0x0040001c  0x214a0001  addi $10,$10,1        16: addi $t2, $t2, 1 # resultado da div
      0x00400020  0x11300001  beq $9,$16,1          17: beq $t1, $s0, mult
      0x00400024  0x08100006  j 0x00400018          18: j div
      0x00400028  0x01715820  add $11,$11,$17       21: add $t3, $t3, $s1 # soma s1 resultado em t3
      0x0040002c  0x218c0001  addi $12,$12,1        22: addi $t4, $t4, 1 # contador
      0x00400030  0x118a0001  beq $12,$10,1         23: beq $t4, $t2, fim
      0x00400034  0x0810000a  j 0x00400028          24: j mult
```

Label	Address ▲
(global)	
main	0x00400000
p1	
div	0x00400018
mult	0x00400028
fim	0x00400038

☑ Data ☑ Text

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	25000
$t1	9	1600000
$t2	10	4
$t3	11	320000
$t4	12	4
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	1600000
$s1	17	80000
$s2	18	400000
$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		4194360
hi		0
lo		0

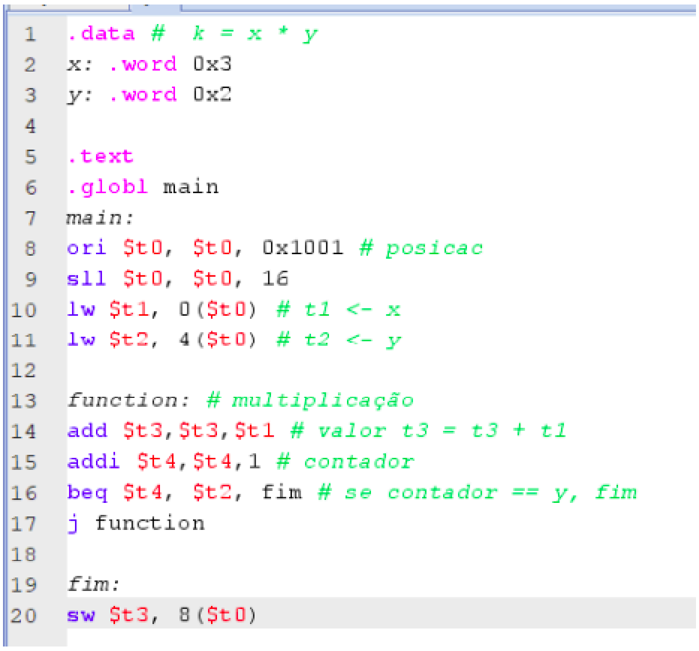
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0	0	0	0	0	0	0	0
0x10010020	0	0	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	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
0x10010160	0	0	0	0	0	0	0	0
0x10010180	0	0	0	0	0	0	0	0
0x100101a0	0	0	0	0	0	0	0	0

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

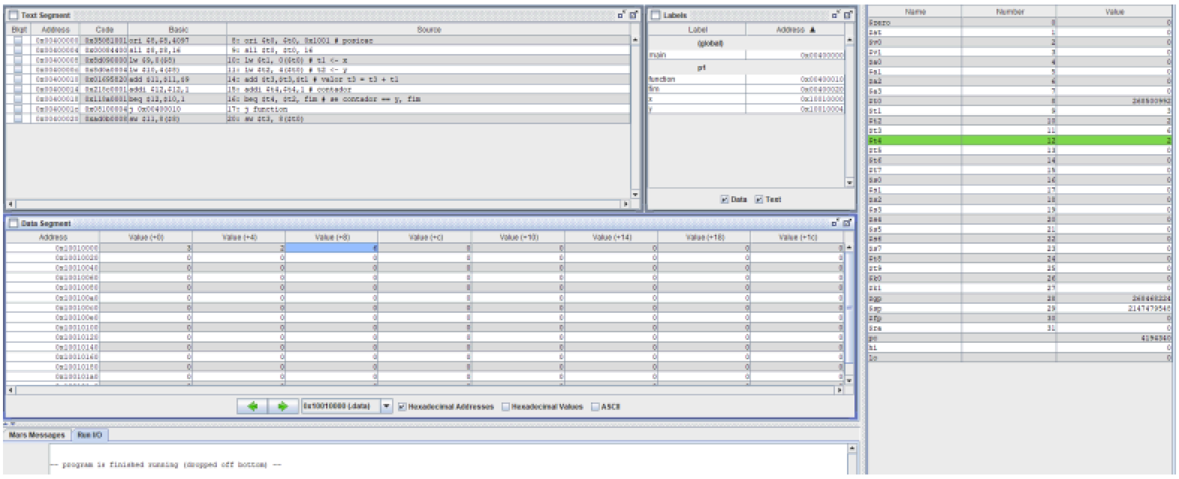
```
-- program is finished running (dropped off bottom) --
```

Programa 17:

valor de k, após calculado, deverá ainda ser escrito na terceira posição livre da memória.



```
.data #  k = x * y
x: .word 0x3
y: .word 0x2

.text
.globl main
main:
ori $t0, $t0, 0x1001 # posicac
sll $t0, $t0, 16
lw $t1, 0($t0) # t1 <- x
lw $t2, 4($t0) # t2 <- y

function: # multiplicação
add $t3,$t3,$t1 # valor t3 = t3 + t1
addi $t4,$t4,1 # contador
beq $t4, $t2, fim # se contador == y, fim
j function

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



Text Segment

Bkpt	Address	Code	Basic	Source
□	0x00400000	0x35081001	ori $8,$8,4097	8: ori $t0, $t0, 0x1001 # posicac
□	0x00400004	0x00084400	sll $8,$8,16	9: sll $t0, $t0, 16
□	0x00400008	0x8d090000	lw $9,0($8)	10: lw $t1, 0($t0) # t1 <- x
□	0x0040000c	0x8d0a0004	lw $10,4($8)	11: lw $t2, 4($t0) # t2 <- y
□	0x00400010	0x01695820	add $11,$11,$9	14: add $t3,$t3,$t1 # valor t3 = t3 + t1
□	0x00400014	0x218c0001	addi $12,$12,1	15: addi $t4,$t4,1 # contador
□	0x00400018	0x118a0001	beq $12,$10,1	16: beq $t4, $t2, fim # se contador == y, fim
□	0x0040001c	0x08100004	j 0x00400010	17: j function
□	0x00400020	0xad0b0008	sw $11,8($8)	20: sw $t3, 8($t0)

Labels

Label	Address ▲
(global)	
main	0x00400000
p1	
function	0x00400010
fim	0x00400020
x	0x10010000
y	0x10010004

☑ Data ☑ Text

Data Segment

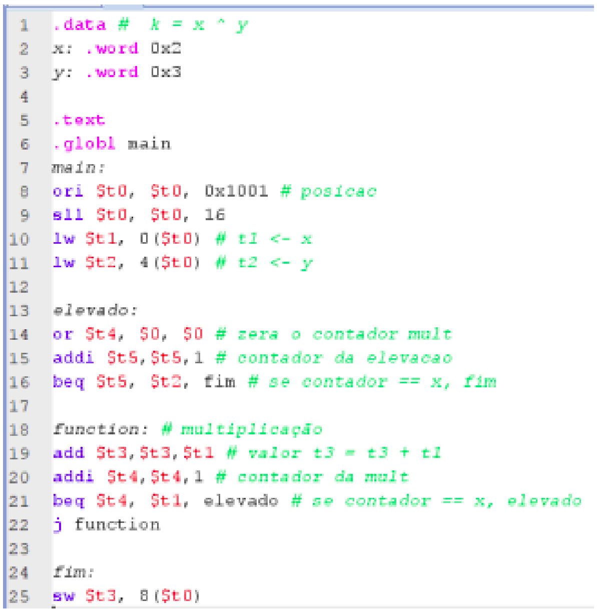
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	3	2	6	0	0	0	0	0

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

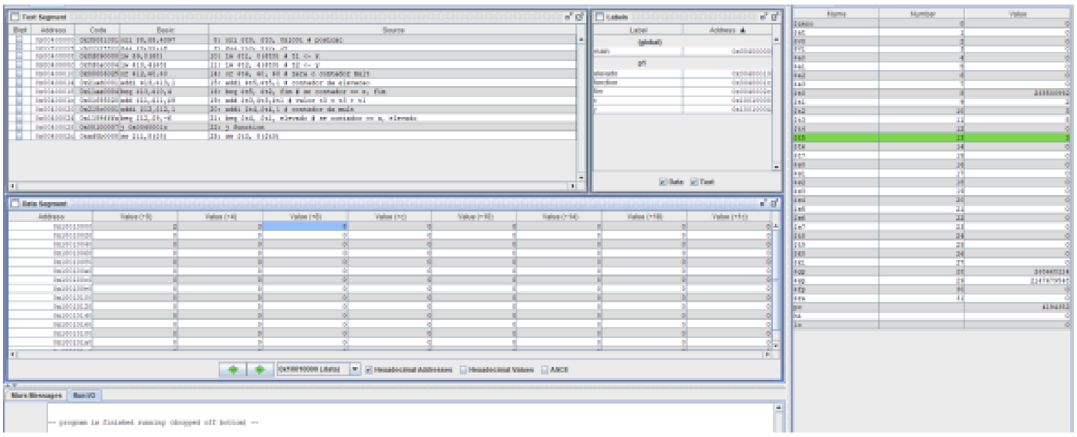
Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	0
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	268500992
$t1	9	3
$t2	10	2
$t3	11	6
$t4	12	2
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	0
$s1	17	0
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0
pc		4194340
hi		0
lo		0

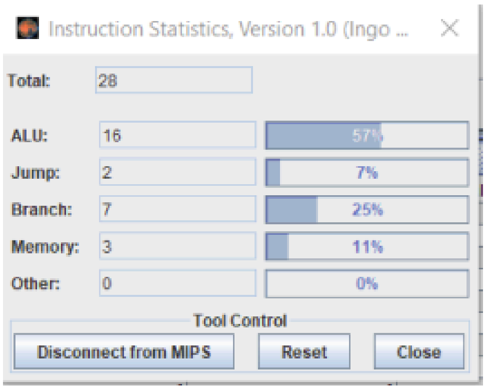
```
-- program is finished running (dropped off bottom) --
```

Programa 18:



```
.data #  k = x ^ y
x: .word 0x2
y: .word 0x3

.text
.globl main
main:
ori $t0, $t0, 0x1001 # posicao
sll $t0, $t0, 16
lw $t1, 0($t0) # t1 <- x
lw $t2, 4($t0) # t2 <- y

elevado:
or $t4, $0, $0 # zera o contador mult
addi $t5,$t5,1 # contador da elevacao
beq $t5, $t2, fim # se contador == x, fim

function: # multiplicação
add $t3,$t3,$t1 # valor t3 = t3 + t1
addi $t4,$t4,1 # contador da mult
beq $t4, $t1, elevado # se contador == x, elevado
j function

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





Instruction Statistics, Version 1.0 (Ingo ...

Total: 28

ALU: 16 57%

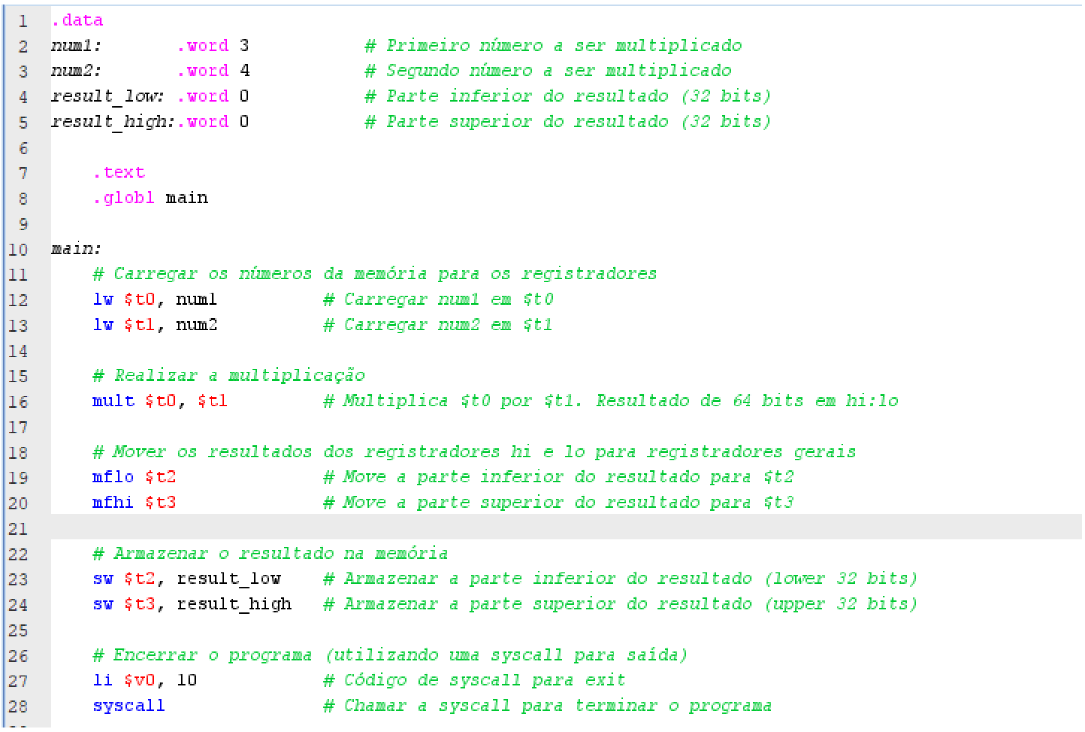
Jump: 2 7%

Branch: 7 25%

Memory: 3 11%

Other: 0 0%

Desafio:



```
.data
num1:       .word 3          # Primeiro número a ser multiplicado
num2:       .word 4          # Segundo número a ser multiplicado
result_low: .word 0          # Parte inferior do resultado (32 bits)
result_high:.word 0          # Parte superior do resultado (32 bits)

    .text
    .globl main

main:
    # Carregar os números da memória para os registradores
    lw $t0, num1          # Carregar num1 em $t0
    lw $t1, num2          # Carregar num2 em $t1

    # Realizar a multiplicação
    mult $t0, $t1         # Multiplica $t0 por $t1. Resultado de 64 bits em hi:lo

    # Mover os resultados dos registradores hi e lo para registradores gerais
    mflo $t2              # Move a parte inferior do resultado para $t2
    mfhi $t3              # Move a parte superior do resultado para $t3

    # Armazenar o resultado na memória
    sw $t2, result_low    # Armazenar a parte inferior do resultado (lower 32 bits)
    sw $t3, result_high   # Armazenar a parte superior do resultado (upper 32 bits)

    # Encerrar o programa (utilizando uma syscall para saída)
    li $v0, 10            # Código de syscall para exit
    syscall               # Chamar a syscall para terminar o programa
```

Perguntas finais:

Questão 01:

C) 64

Questão 02:

B) hi e lo

Questão 03:

A) mult

Questão 04:

C) mflo $8

Questão 05:

B) 32

Questão 06:

A) lo

Questão 07:

D) div

Questão 08:

A) 1110 0110

Questão 09:

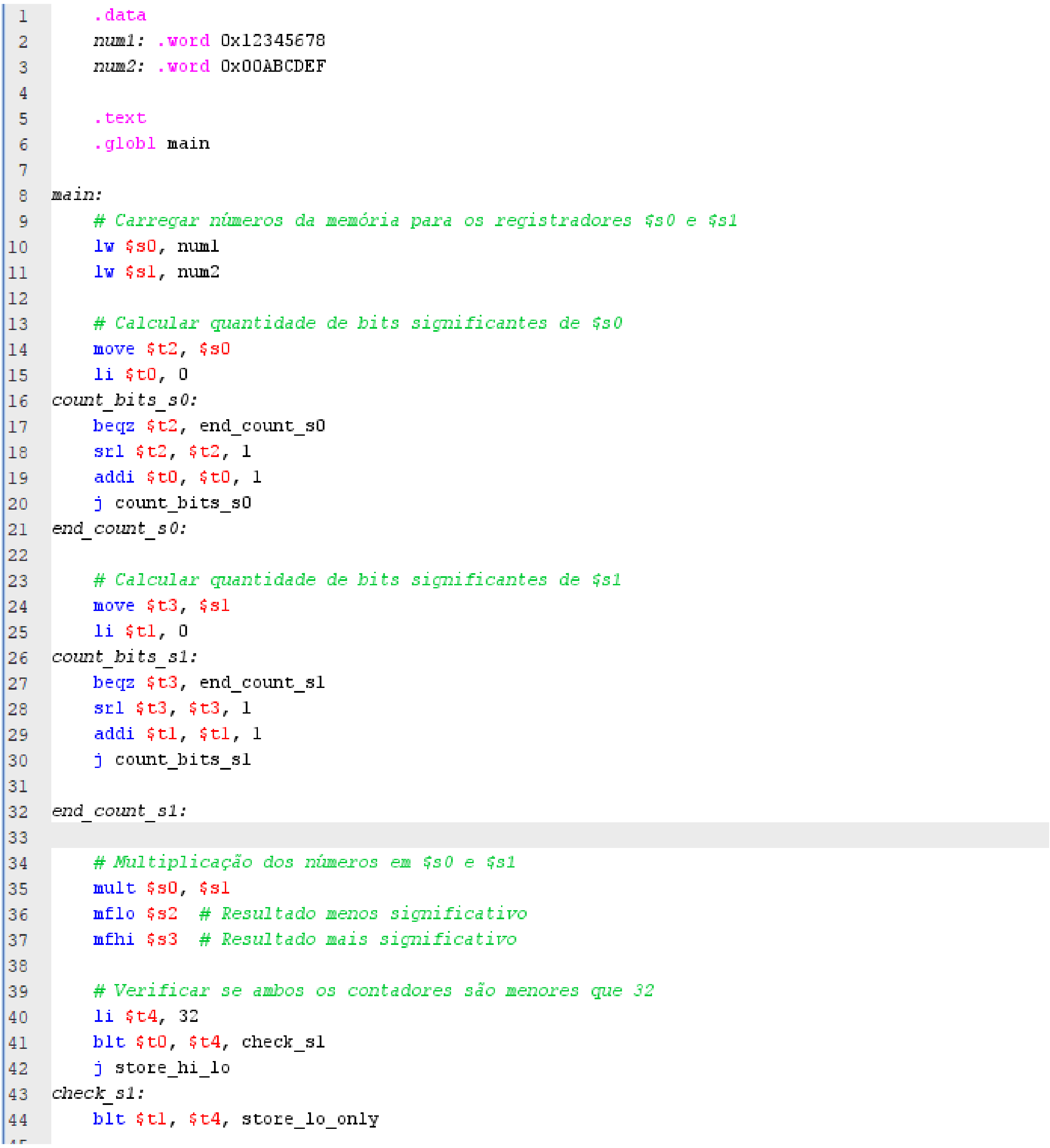
A) Se o inteiro for unsigned, o shift o divide por 2. Se o inteiro for signed, o shift o divide por 2.

Questão 10:

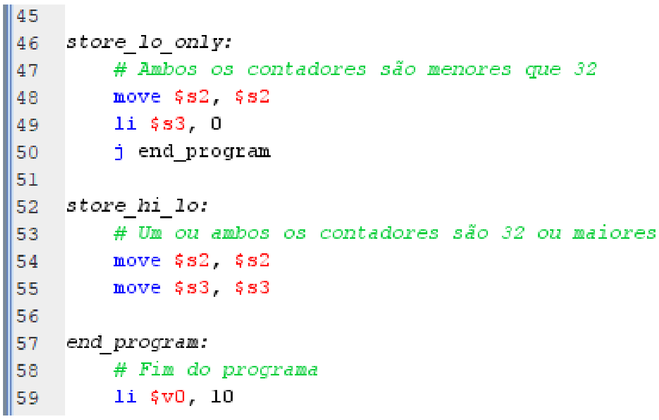
A)
```
ori $3,$0,3
mult $8,$3
mflo $9
addi $9,$9,7
```

Programas finais:

Programa 19:

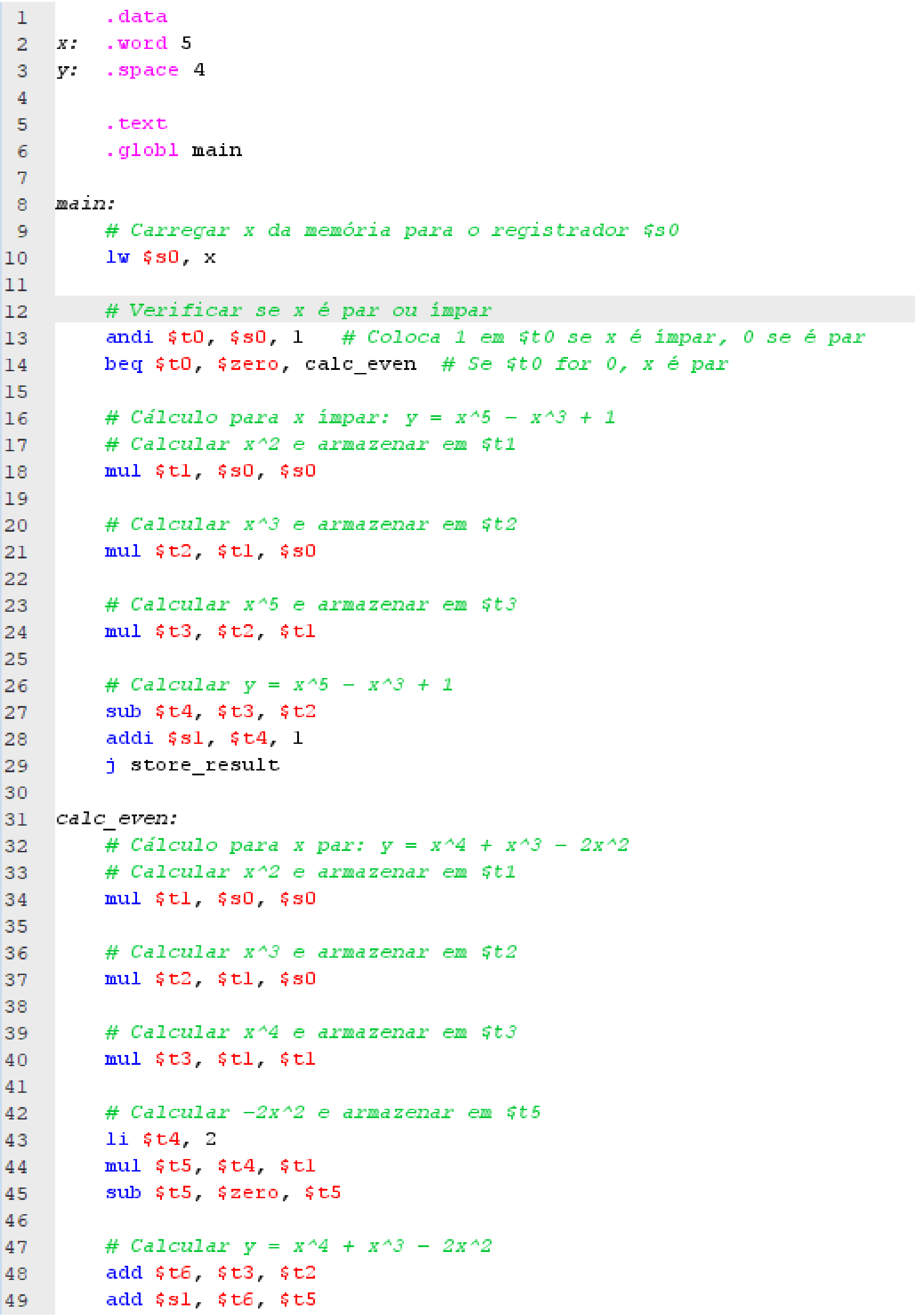


```
    .data
    num1: .word 0x12345678
    num2: .word 0x00ABCDEF

    .text
    .globl main

main:
    # Carregar números da memória para os registradores $s0 e $s1
    lw $s0, num1
    lw $s1, num2

    # Calcular quantidade de bits significantes de $s0
    move $t2, $s0
    li $t0, 0
count_bits_s0:
    beqz $t2, end_count_s0
    srl $t2, $t2, 1
    addi $t0, $t0, 1
    j count_bits_s0
end_count_s0:

    # Calcular quantidade de bits significantes de $s1
    move $t3, $s1
    li $t1, 0
count_bits_s1:
    beqz $t3, end_count_s1
    srl $t3, $t3, 1
    addi $t1, $t1, 1
    j count_bits_s1

end_count_s1:

    # Multiplicação dos números em $s0 e $s1
    mult $s0, $s1
    mflo $s2  # Resultado menos significativo
    mfhi $s3  # Resultado mais significativo

    # Verificar se ambos os contadores são menores que 32
    li $t4, 32
    blt $t0, $t4, check_s1
    j store_hi_lo
check_s1:
    blt $t1, $t4, store_lo_only
```

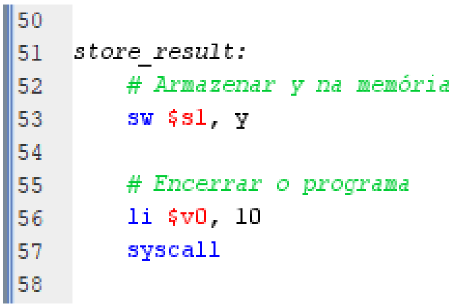


```

store_lo_only:
    # Ambos os contadores são menores que 32
    move $s2, $s2
    li $s3, 0
    j end_program

store_hi_lo:
    # Um ou ambos os contadores são 32 ou maiores
    move $s2, $s2
    move $s3, $s3

end_program:
    # Fim do programa
    li $v0, 10
```

Programa 20:

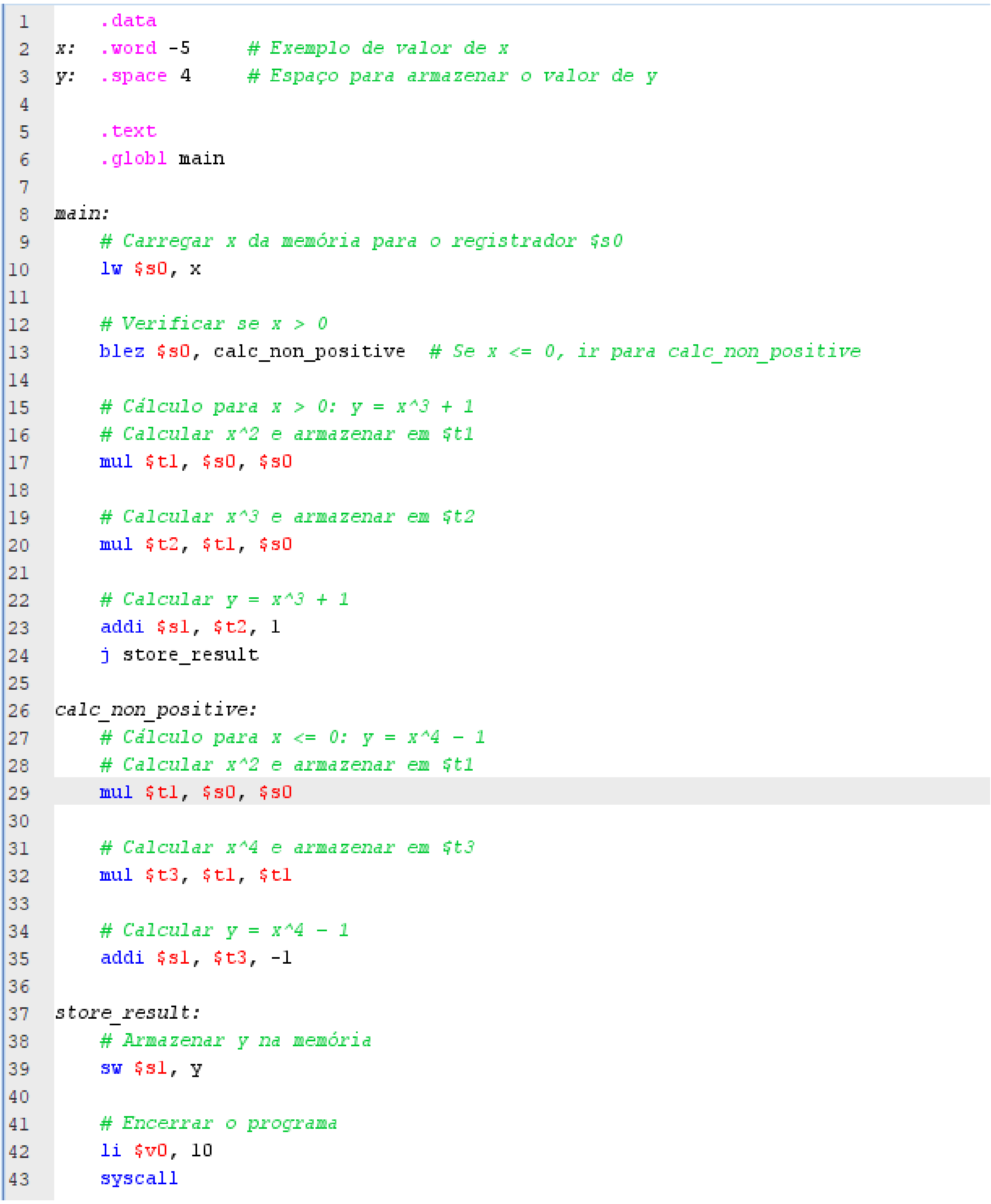


```
    .data
x:  .word 5
y:  .space 4

    .text
    .globl main

main:
    # Carregar x da memória para o registrador $s0
    lw $s0, x

    # Verificar se x é par ou impar
    andi $t0, $s0, 1   # Coloca 1 em $t0 se x é ímpar, 0 se é par
    beq $t0, $zero, calc_even  # Se $t0 for 0, x é par

    # Cálculo para x ímpar: y = x^5 - x^3 + 1
    # Calcular x^2 e armazenar em $t1
    mul $t1, $s0, $s0

    # Calcular x^3 e armazenar em $t2
    mul $t2, $t1, $s0

    # Calcular x^5 e armazenar em $t3
    mul $t3, $t2, $t1

    # Calcular y = x^5 - x^3 + 1
    sub $t4, $t3, $t2
    addi $s1, $t4, 1
    j store_result

calc_even:
    # Cálculo para x par: y = x^4 + x^3 - 2x^2
    # Calcular x^2 e armazenar em $t1
    mul $t1, $s0, $s0

    # Calcular x^3 e armazenar em $t2
    mul $t2, $t1, $s0

    # Calcular x^4 e armazenar em $t3
    mul $t3, $t1, $t1

    # Calcular -2x^2 e armazenar em $t5
    li $t4, 2
    mul $t5, $t4, $t1
    sub $t5, $zero, $t5

    # Calcular y = x^4 + x^3 - 2x^2
    add $t6, $t3, $t2
    add $s1, $t6, $t5
```



```

store_result:
    # Armazenar y na memória
    sw $s1, y

    # Encerrar o programa
    li $v0, 10
    syscall

```

Programa 21:



```
    .data
x:  .word -5     # Exemplo de valor de x
y:  .space 4     # Espaço para armazenar o valor de y

    .text
    .globl main

main:
    # Carregar x da memória para o registrador $s0
    lw $s0, x

    # Verificar se x > 0
    blez $s0, calc_non_positive  # Se x <= 0, ir para calc_non_positive

    # Cálculo para x > 0: y = x^3 + 1
    # Calcular x^2 e armazenar em $t1
    mul $t1, $s0, $s0

    # Calcular x^3 e armazenar em $t2
    mul $t2, $t1, $s0

    # Calcular y = x^3 + 1
    addi $s1, $t2, 1
    j store_result

calc_non_positive:
    # Cálculo para x <= 0: y = x^4 - 1
    # Calcular x^2 e armazenar em $t1
    mul $t1, $s0, $s0

    # Calcular x^4 e armazenar em $t3
    mul $t3, $t1, $t1

    # Calcular y = x^4 - 1
    addi $s1, $t3, -1

store_result:
    # Armazenar y na memória
    sw $s1, y

    # Encerrar o programa
    li $v0, 10
    syscall
```