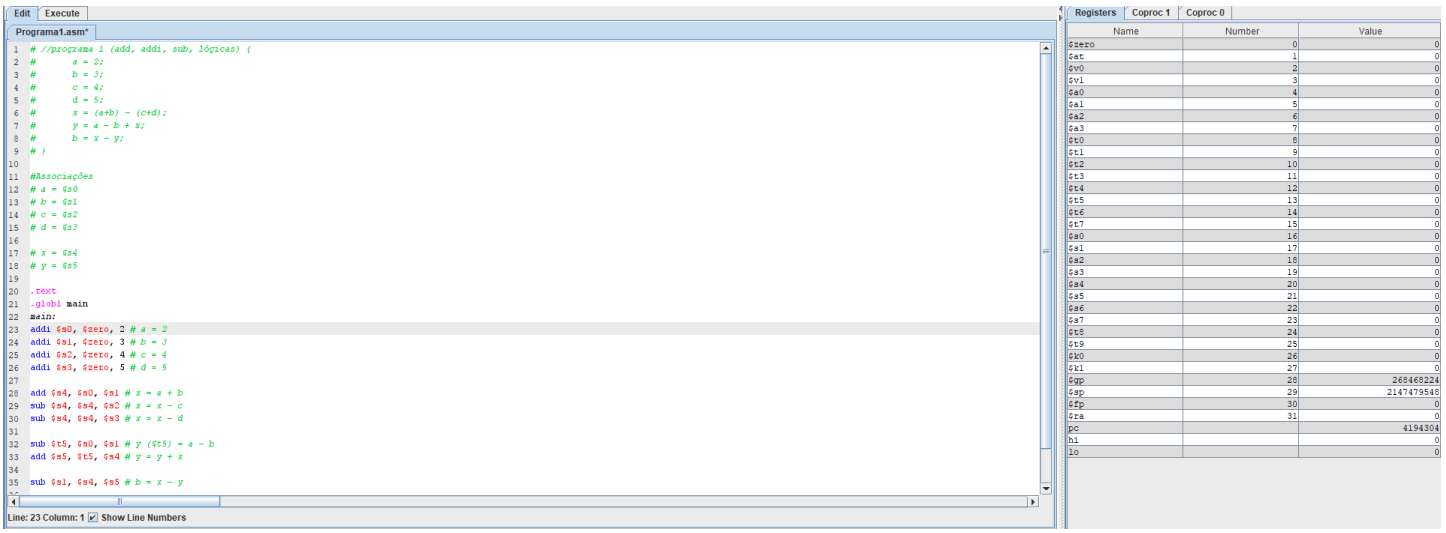
Nome: Gabriel Henrique Vieira de Oliveira
Data: 09/06/2024
Matéria: Arquitetura de Computadores II - Exercício prático 06

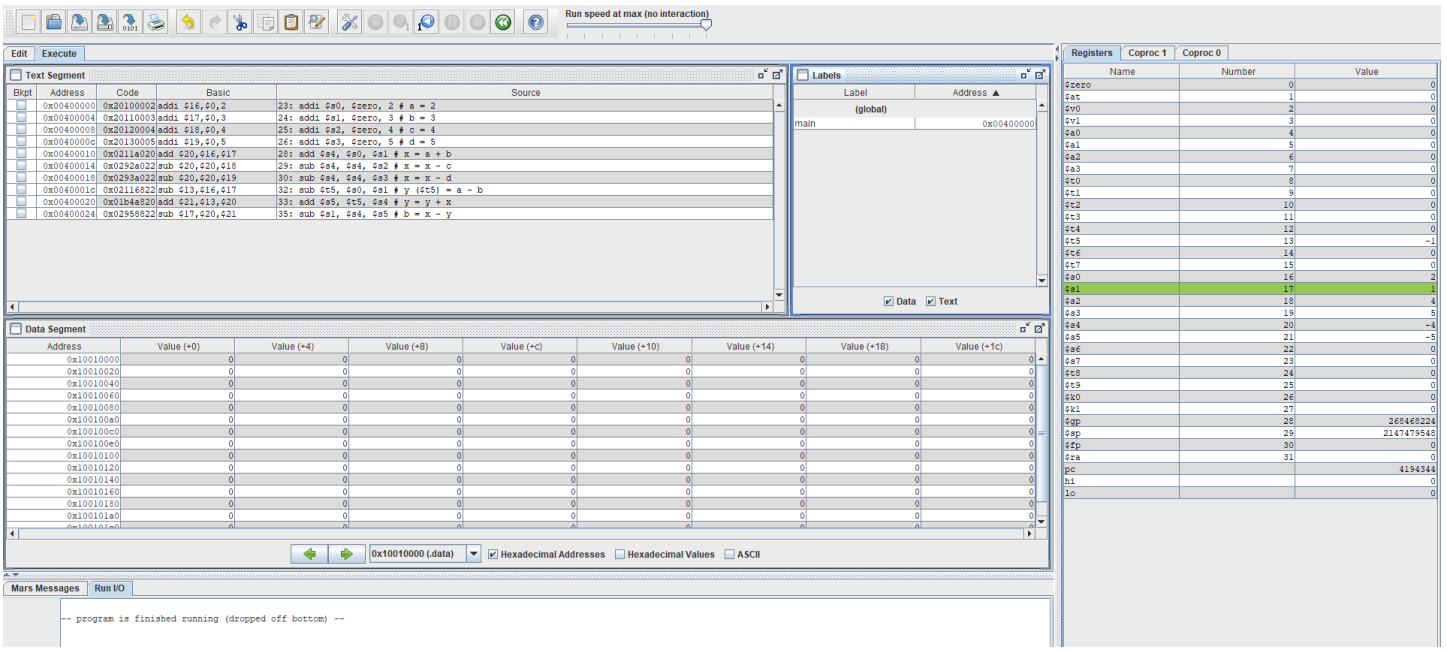
Parte 1 - Responda

1) A
2) B
3) A
4) C
5) D (É o indicador de endereço de memória externa de programa. Ele serve para endereçar a próxima instrução a ser lida pelo microprocessador a fim de que ele "saiba“ onde estava e para onde deve ir (ler nova instrução no próximo endereço).
6) C
7) D
8) D
9) B
10) A
11) B (OPERAÇÃO BITWISE)
12) D
13) A
14) C
15) B
16) D
17) A
18) A

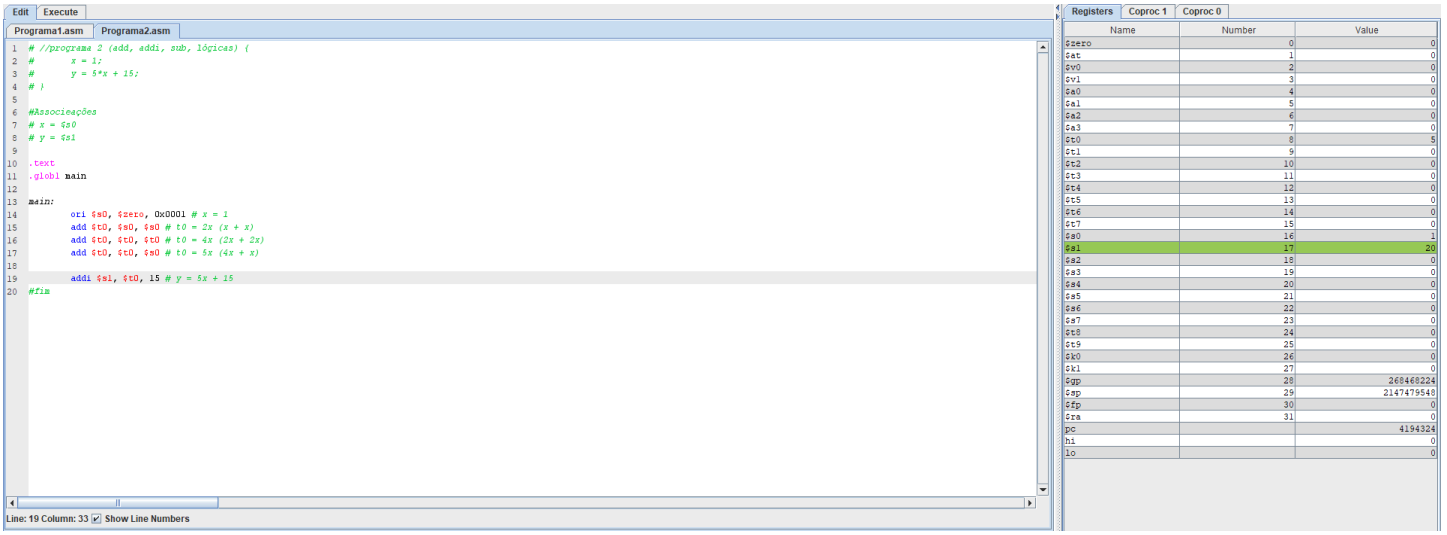
Parte 2 - Implementar em MIPS/MARS os seguintes programas

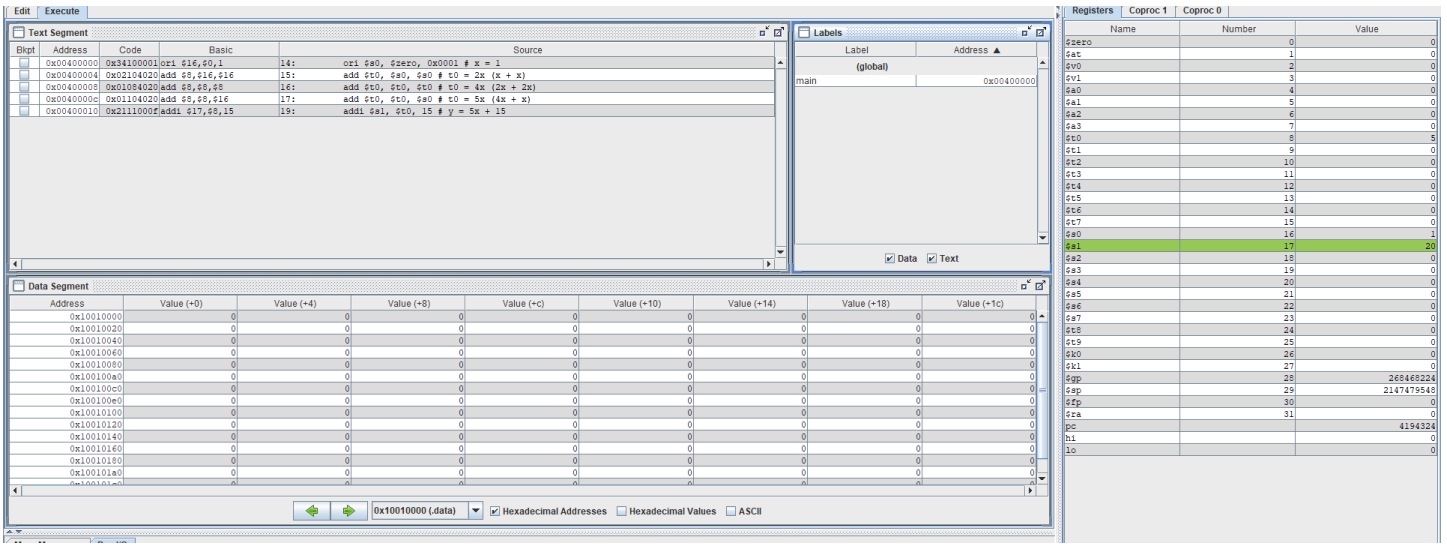
PROGRAMA 1:



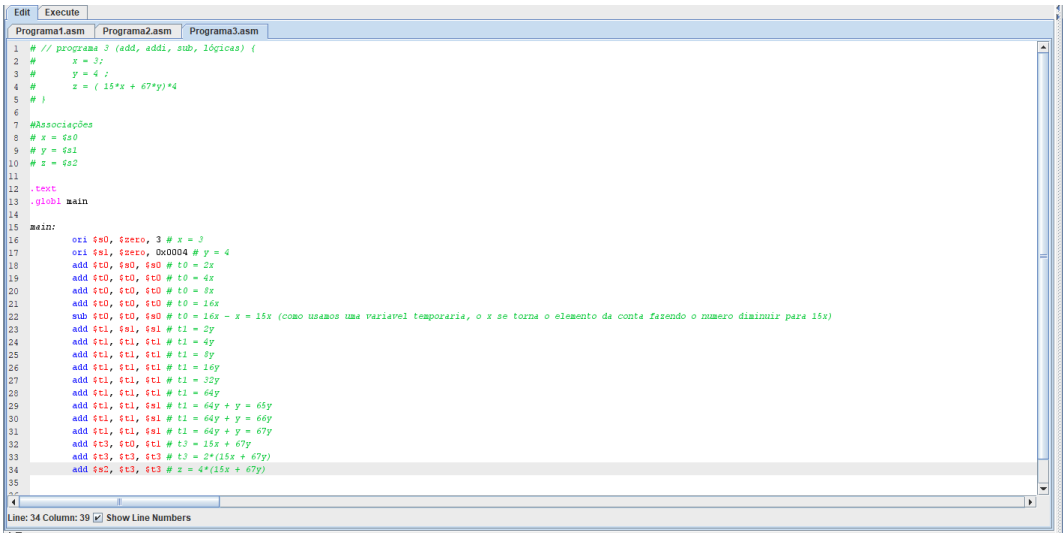

Run speed at max (no interaction)
Edit
Execute
Text Segment
Bkpt Address Code Basic Source
0x00400000 0x20100002 addi $16,$0,2 23: addi $s0, $zero, 2 # a = 2
0x00400004 0x20110003 addi $17,$0,3 24: addi $s1, $zero, 3 # b = 3
0x00400008 0x20120004 addi $18,$0,4 25: addi $s2, $zero, 4 # c = 4
0x0040000c 0x20130005 addi $19,$0,5 26: addi $s3, $zero, 5 # d = 5
0x00400010 0x0211a020 add $20,$16,$17 28: add $s4, $s0, $s1 # x = a + b
0x00400014 0x0292a022 sub $20,$20,$18 29: sub $s4, $s4, $s2 # x = x - c
0x00400018 0x0293a022 sub $20,$20,$19 30: sub $s4, $s4, $s3 # x = x - d
0x0040001c 0x02116822 sub $13,$16,$17 32: sub $t5, $s0, $s1 # y (t5) = a - b
0x00400020 0x01b4a820 add $21,$13,$20 33: add $s5, $t5, $s4 # y = y + x
0x00400024 0x02958822 sub $17,$20,$21 35: sub $s1, $s4, $s5 # b = x - y
Labels
Label Address
(global)
main 0x00400000
Data Text
Data Segment
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
Mars Messages
Run I/O
-- program is finished running (dropped off bottom) --
Registers
Coproc 1
Coproc 0
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 0
$t1 9 0
$t2 10 0
$t3 11 0
$t4 12 0
$t5 13 -1
$t6 14 0
$t7 15 0
$s0 16 2
$s1 17 1
$s2 18 4
$s3 19 5
$s4 20 -4
$s5 21 -5
$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

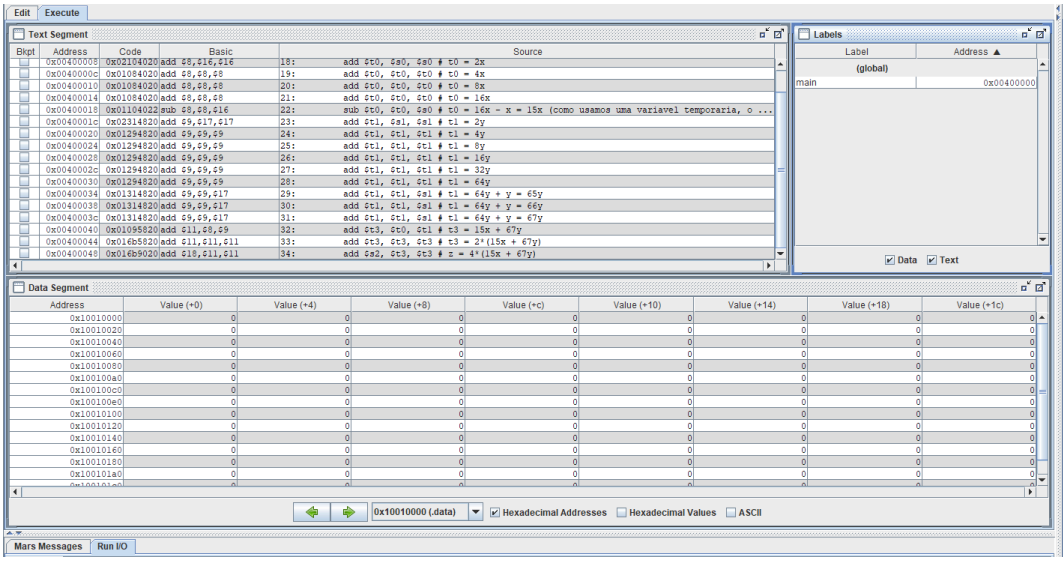
PROGRAMA 2:


Edit
Execute
Programa1.asm
Programa2.asm
1 # //programa 2 (add, addi, sub, lógicas) (
2 # x = 1;
3 # y = 5*x + 15;
4 #)
5
6 #Associeações
7 # x = $s0
8 # y = $s1
9
10 .text
11 .globl main
12
13 main:
14 ori $s0, $zero, 0x0001 # x = 1
15 add $t0, $s0, $s0 # t0 = 2x (x + x)
16 add $t0, $t0, $t0 # t0 = 4x (2x + 2x)
17 add $t0, $t0, $s0 # t0 = 5x (4x + x)
18
19 addi $s1, $t0, 15 # y = 5x + 15
20 #fim
Line: 19 Column: 33
Show Line Numbers
Registers
Coproc 1
Coproc 0
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

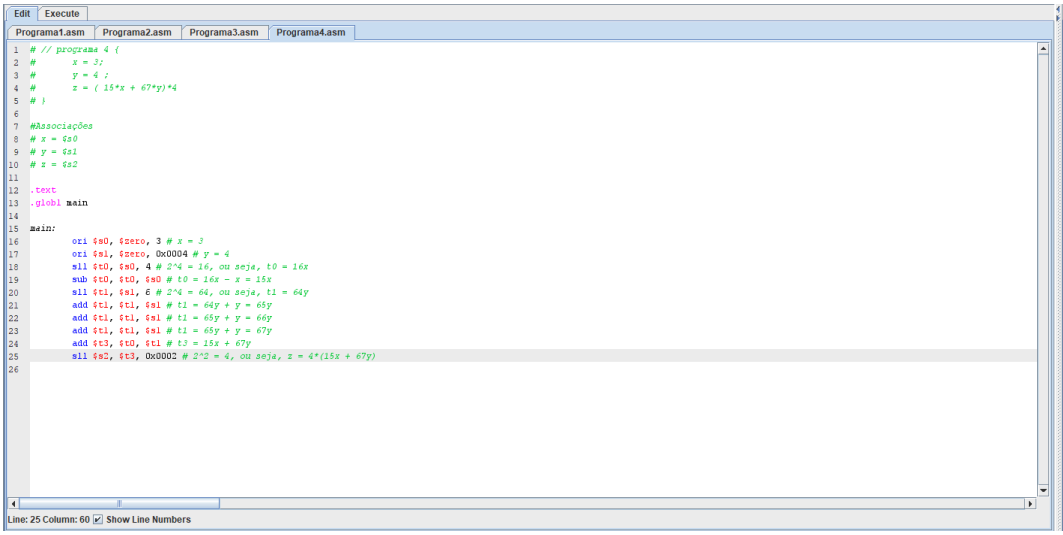

Edit
Execute
Text Segment
Bkpt Address Code Basic Source
0x00400000 0x34100001 ori $16,$0,1 14: ori $s0, $zero, 0x0001 # x = 1
0x00400004 0x02104020 add $8,$16,$16 15: add $t0, $s0, $s0 # t0 = 2x (x + x)
0x00400008 0x01084020 add $8,$8,$8 16: add $t0, $t0, $t0 # t0 = 4x (2x + 2x)
0x0040000c 0x01104020 add $8,$8,$16 17: add $t0, $t0, $s0 # t0 = 5x (4x + x)
0x00400010 0x2111000f addi $17,$8,15 19: addi $s1, $t0, 15 # y = 5x + 15
Labels
Label Address
(global)
main 0x00400000
Data Text
Data Segment
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
Registers
Coproc 1
Coproc 0
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

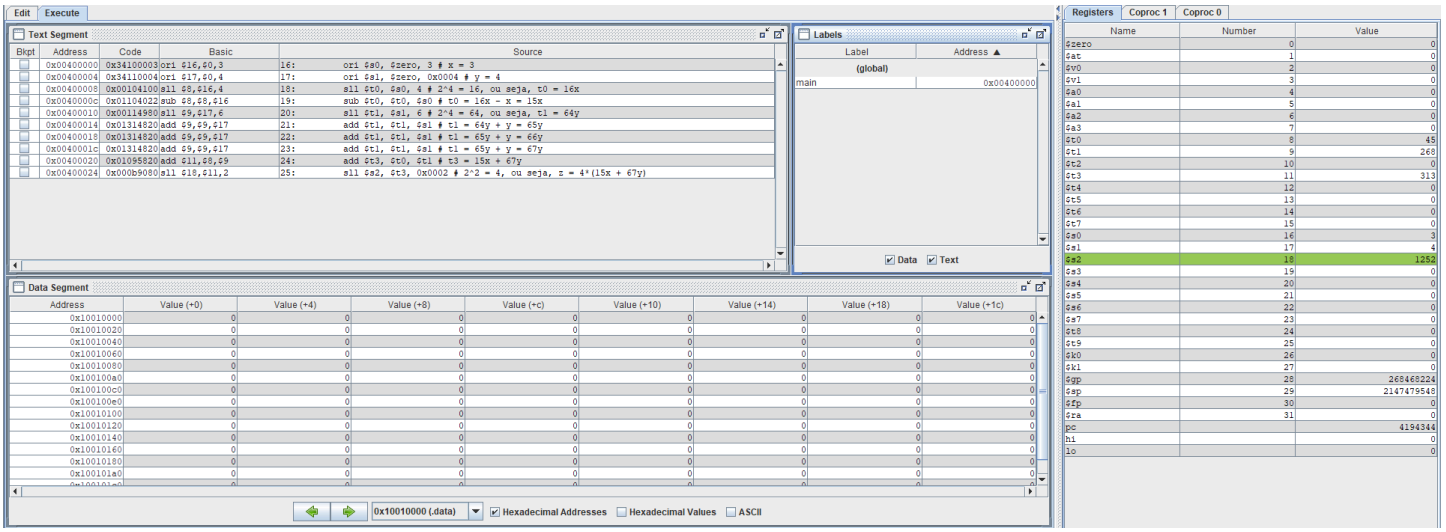
PROGRAMA 3:



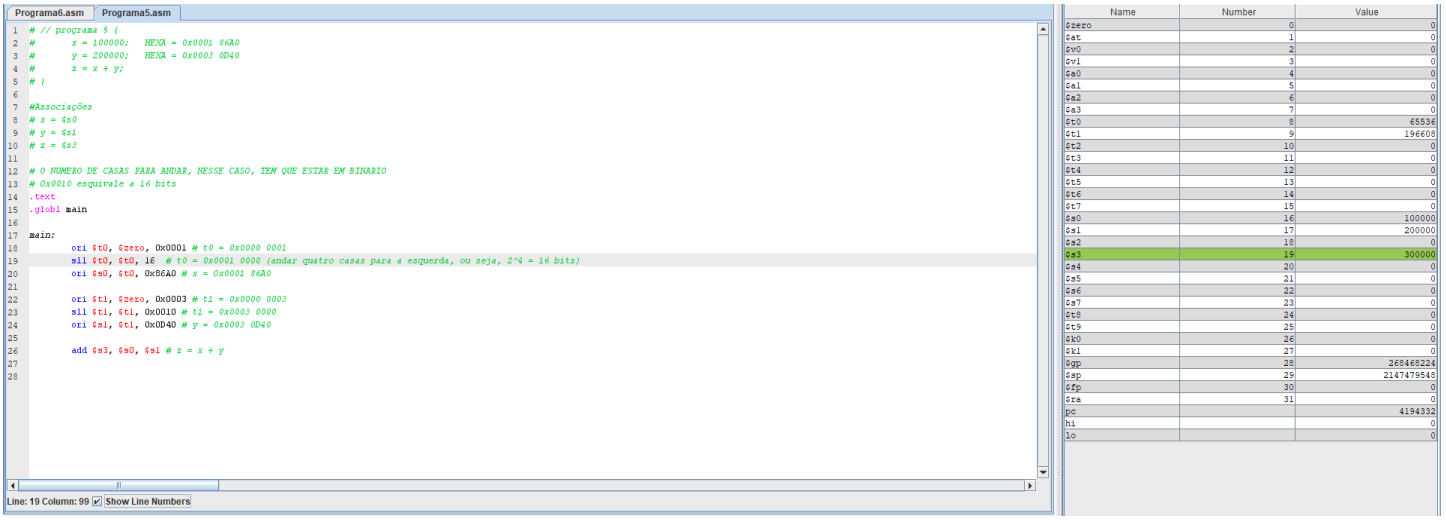


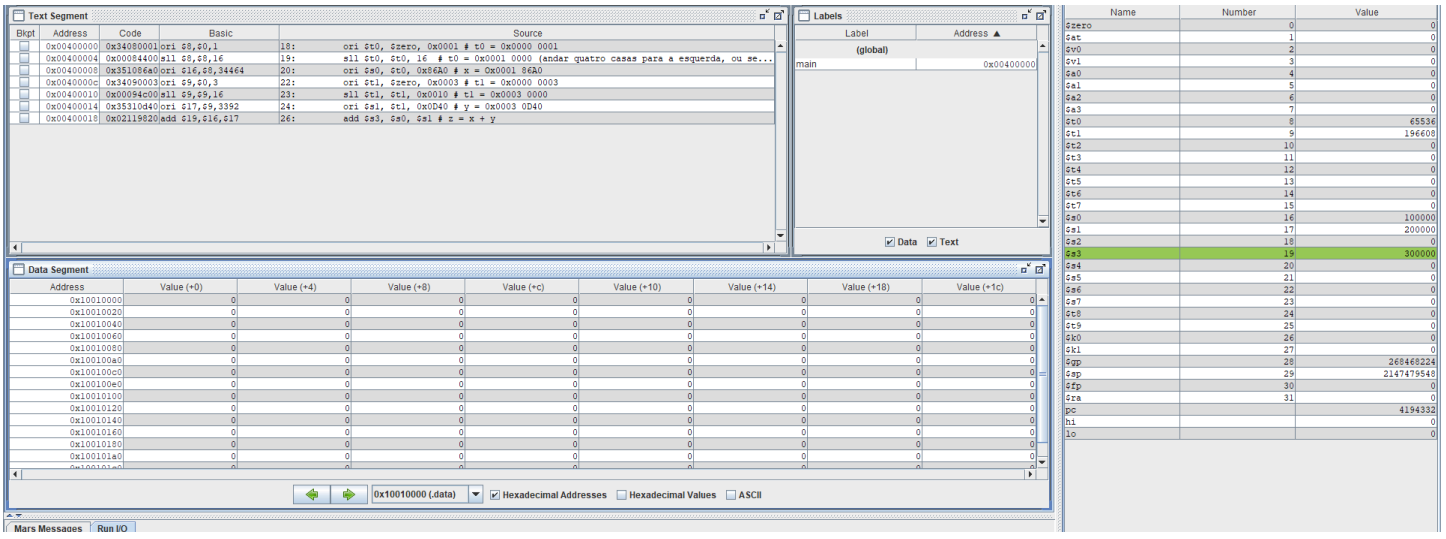
PROGRAMA 4:



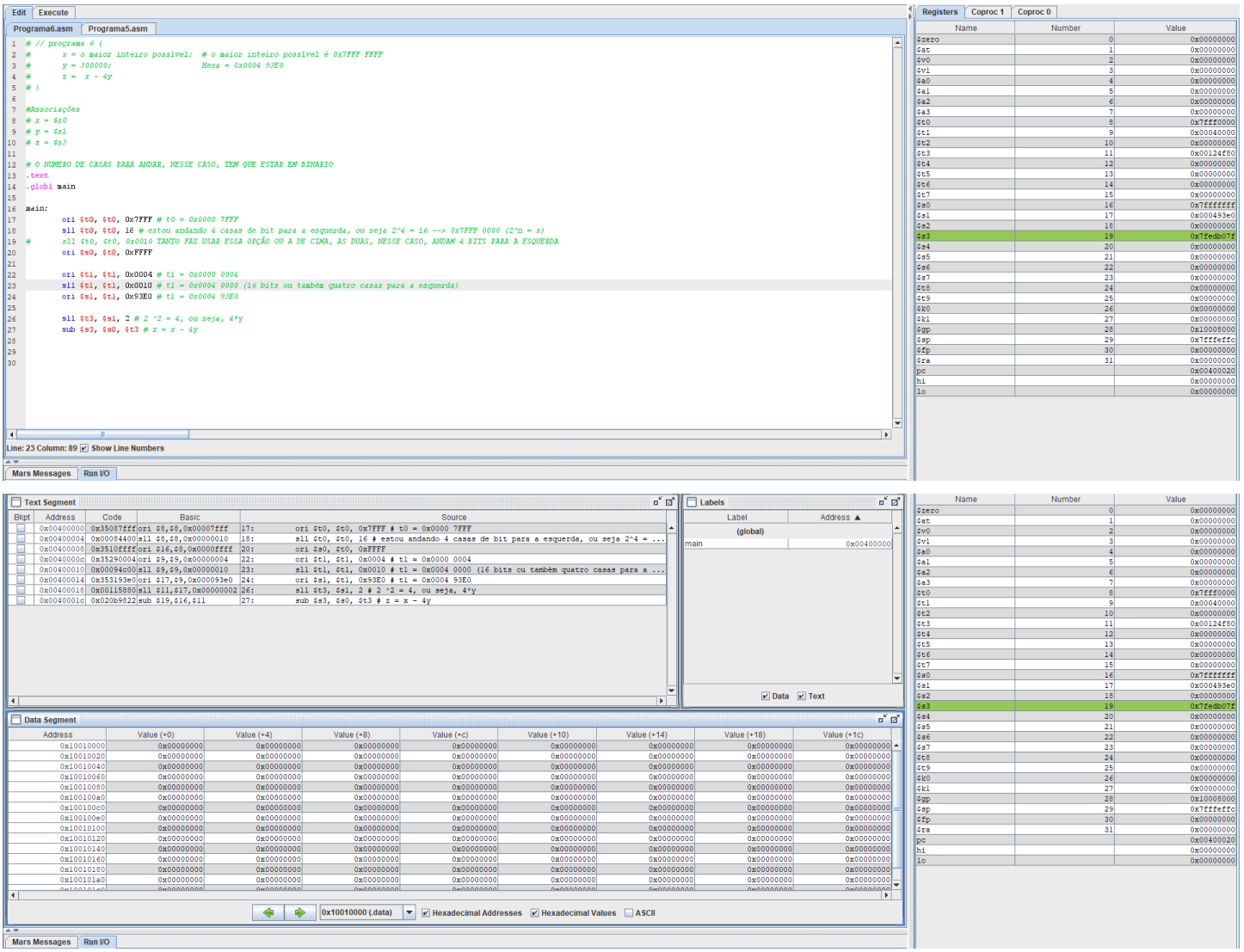

Edit
Execute
Text Segment
Bkpt Address Code Basic Source
0x00400000 0x34100003 ori $16,$0,3 16: ori $s0, $zero, 3 # x = 3
0x00400004 0x34110004 ori $17,$0,4 17: ori $s1, $zero, 0x0004 # y = 4
0x00400008 0x00104100 sll $8,$16,4 18: sll $t0, $s0, 4 # 2^4 = 16, ou seja, t0 = 16x
0x0040000c 0x01104022 sub $8,$8,$16 19: sub $t0, $t0, $s0 # t0 = 16x - x = 15x
0x00400010 0x00114980 sll $9,$17,6 20: sll $t1, $s1, 6 # 2^4 = 64, ou seja, t1 = 64y
0x00400014 0x01314820 add $9,$9,$17 21: add $t1, $t1, $s1 # t1 = 64y + y = 65y
0x00400018 0x01314820 add $9,$9,$17 22: add $t1, $t1, $s1 # t1 = 65y + y = 66y
0x0040001c 0x01314820 add $9,$9,$17 23: add $t1, $t1, $s1 # t1 = 65y + y = 67y
0x00400020 0x01095820 add $11,$8,$9 24: add $t3, $t0, $t1 # t3 = 15x + 67y
0x00400024 0x000b9080 sll $18,$11,2 25: sll $s2, $t3, 0x0002 # 2^2 = 4, ou seja, z = 4*(15x + 67y)
Labels
Label Address
(global)
main 0x00400000
Data Text
Registers Coproc 1 Coproc 0
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 45
$t1 9 268
$t2 10 0
$t3 11 313
$t4 12 0
$t5 13 0
$t6 14 0
$t7 15 0
$s0 16 3
$s1 17 4
$s2 18 1252
$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 4194344
hi 0
lo 0
Data Segment
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

PROGRAMA 5:


Programa6.asm
Programa5.asm
// programa 5 {
x = 100000; HEXA = 0x0001 86A0
y = 200000; HEXA = 0x0003 0D40
z = x + y;
}
#Associações
x = $s0
y = $s1
z = $s3
O NUMERO DE CASAS PARA ANDAR, NESSE CASO, TEM QUE ESTAR EM BINARIO
0x0010 esquivale a 16 bits
.text
.globl main
main:
ori $t0, $zero, 0x0001 # t0 = 0x0000 0001
sll $t0, $t0, 16 # t0 = 0x0001 0000 (andar quatro casas para a esquerda, ou seja, 2^4 = 16 bits)
ori $s0, $t0, 0x86A0 # x = 0x0001 86A0
ori $t1, $zero, 0x0003 # t1 = 0x0000 0003
sll $t1, $t1, 0x0010 # t1 = 0x0003 0000
ori $s1, $t1, 0x0D40 # y = 0x0003 0D40
add $s3, $s0, $s1 # z = x + y
Line: 19 Column: 99
Show Line Numbers
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 65536
$t1 9 196608
$t2 10 0
$t3 11 0
$t4 12 0
$t5 13 0
$t6 14 0
$t7 15 0
$s0 16 100000
$s1 17 200000
$s2 18 0
$s3 19 300000
$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 4194332
hi 0
lo 0


Text Segment
Bkpt Address Code Basic Source
0x00400000 0x34080001 ori $8,$0,1 18: ori $t0, $zero, 0x0001 # t0 = 0x0000 0001
0x00400004 0x00084400 sll $8,$8,16 19: sll $t0, $t0, 16 # t0 = 0x0001 0000 (andar quatro casas para a esquerda, ou se...
0x00400008 0x351086a0 ori $16,$8,34464 20: ori $s0, $t0, 0x86A0 # x = 0x0001 86A0
0x0040000c 0x34090003 ori $9,$0,3 22: ori $t1, $zero, 0x0003 # t1 = 0x0000 0003
0x00400010 0x00094c00 sll $9,$9,16 23: sll $t1, $t1, 0x0010 # t1 = 0x0003 0000
0x00400014 0x35310d40 ori $17,$9,3392 24: ori $s1, $t1, 0x0D40 # y = 0x0003 0D40
0x00400018 0x02119820 add $19,$16,$17 26: add $s3, $s0, $s1 # z = x + y
Labels
Label Address
(global)
main 0x00400000
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 65536
$t1 9 196608
$t2 10 0
$t3 11 0
$t4 12 0
$t5 13 0
$t6 14 0
$t7 15 0
$s0 16 100000
$s1 17 200000
$s2 18 0
$s3 19 300000
$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 4194332
hi 0
lo 0
Data Segment
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
Mars Messages
Run I/O

PROGRAMA 6:



```
# // programa 6 {
#       x = o maior inteiro possivel;  # o maior inteiro possivel é 0x7FFF FFFF
#       y = 300000;                    Hexa = 0x0004 93E0
#       z =  x - 4y
# }

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

# O NUMERO DE CASAS PARA ANDAR, NESSE CASO, TEM QUE ESTAR EM BINARIO
.text
.globl main

main:
        ori $t0, $t0, 0x7FFF # t0 = 0x0000 7FFF
        sll $t0, $t0, 16 # estou andando 4 casas de bit para a esquerda, ou seja 2^4 = 16 --> 0x7FFF 0000 (2^n = x)
#       sll $t0, $t0, 0x0010 TANTO FAZ USAR ESSA OPÇÃO OU A DE CIMA, AS DUAS, NESSE CASO, ANDAM 4 BITS PARA A ESQUERDA
        ori $s0, $t0, 0xFFFF

        ori $t1, $t1, 0x0004 # t1 = 0x0000 0004
        sll $t1, $t1, 0x0010 # t1 = 0x0004 0000 (16 bits ou também quatro casas para a esquerda)
        ori $s1, $t1, 0x93E0 # t1 = 0x0004 93E0

        sll $t3, $s1, 2 # 2 ^2 = 4, ou seja, 4*y
        sub $s3, $s0, $t3 # z = x - 4y
```

Line: 23 Column: 89 Show Line Numbers

Mars Messages | Run I/O

Registers | Coproc 1 | Coproc 0

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	0x7fff0000
$t1	9	0x00040000
$t2	10	0x00000000
$t3	11	0x00124f80
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x7fffffff
$s1	17	0x000493e0
$s2	18	0x00000000
$s3	19	0x7fedb07f
$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

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x35087fff	ori $8,$8,0x00007fff	17: ori $t0, $t0, 0x7FFF # t0 = 0x0000 7FFF
	0x00400004	0x00084400	sll $8,$8,0x00000010	18: sll $t0, $t0, 16 # estou andando 4 casas de bit para a esquerda, ou seja 2^4 = ...
	0x00400008	0x3510ffff	ori $16,$8,0x0000ffff	20: ori $s0, $t0, 0xFFFF
	0x0040000c	0x35290004	ori $9,$9,0x00000004	22: ori $t1, $t1, 0x0004 # t1 = 0x0000 0004
	0x00400010	0x00094c00	sll $9,$9,0x00000010	23: sll $t1, $t1, 0x0010 # t1 = 0x0004 0000 (16 bits ou também quatro casas para a ...
	0x00400014	0x353193e0	ori $17,$9,0x000093e0	24: ori $s1, $t1, 0x93E0 # t1 = 0x0004 93E0
	0x00400018	0x00115880	sll $11,$17,0x00000002	26: sll $t3, $s1, 2 # 2 ^2 = 4, ou seja, 4*y
	0x0040001c	0x020b9822	sub $19,$16,$11	27: sub $s3, $s0, $t3 # z = x - 4y

Labels

Label	Address
(global)	
main	0x00400000

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
0x10010160	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010180	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

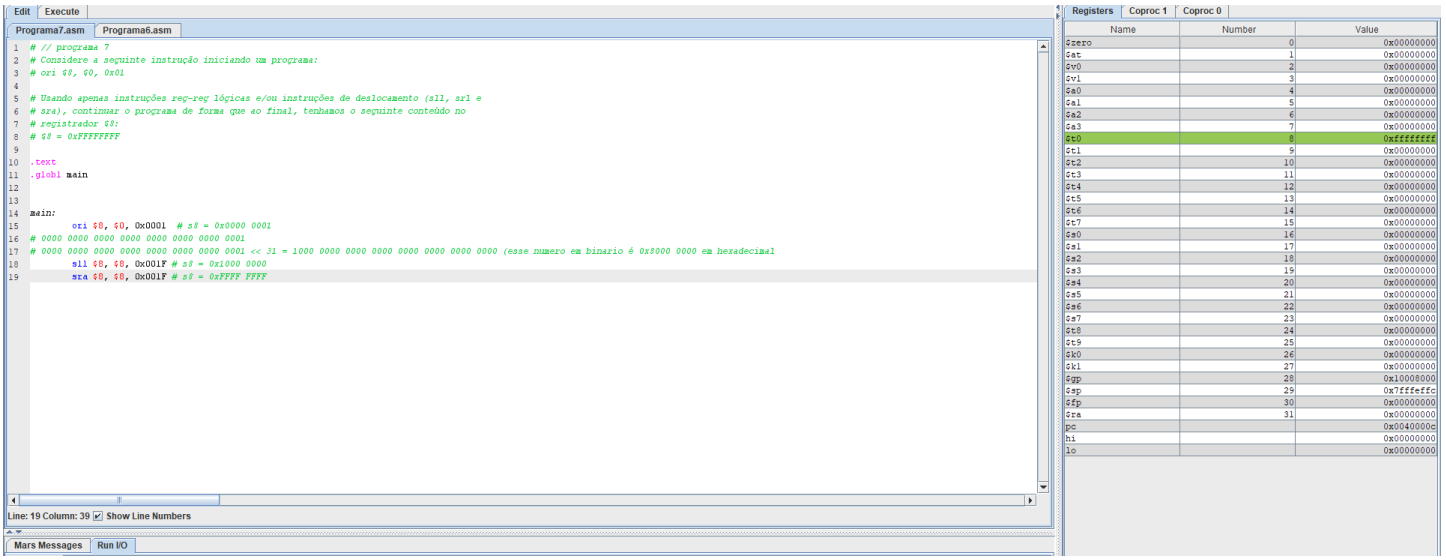
0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII

Mars Messages | Run I/O

Registers | Coproc 1 | Coproc 0

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	0x7fff0000
$t1	9	0x00040000
$t2	10	0x00000000
$t3	11	0x00124f80
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x7fffffff
$s1	17	0x000493e0
$s2	18	0x00000000
$s3	19	0x7fedb07f
$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 7:



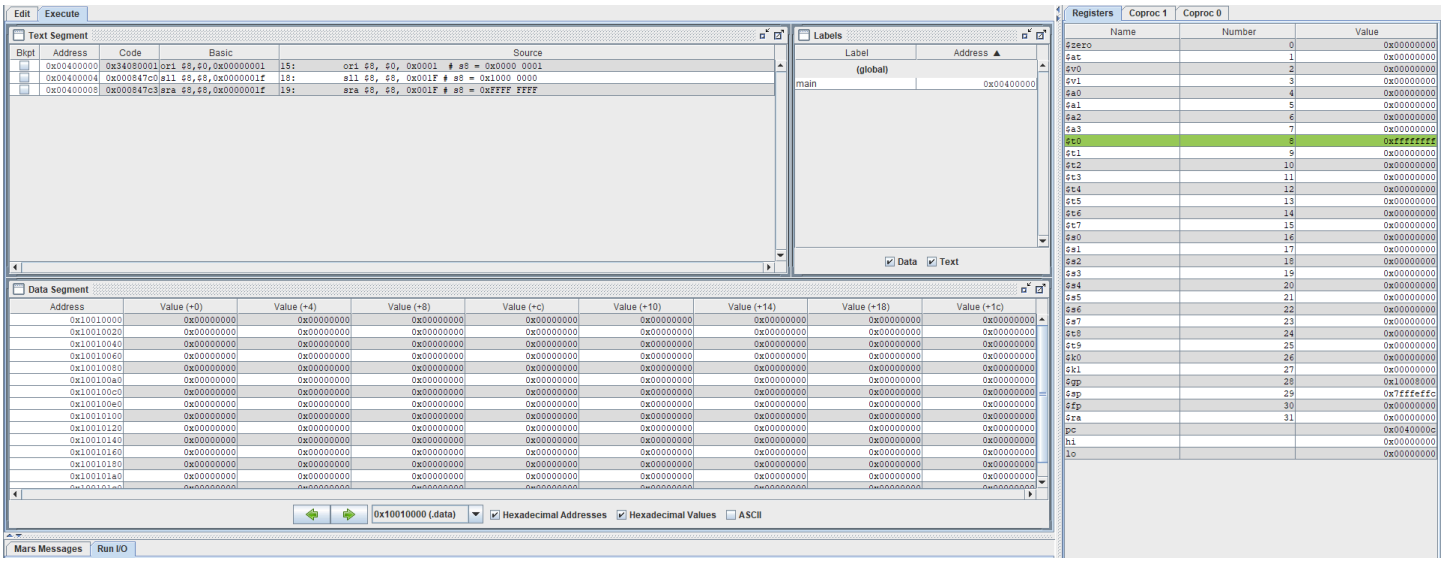
```
# // programa 7
# Considere a seguinte instrução iniciando um programa:
# ori $8, $0, 0x01

# 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
.globl main


main:
        ori $8, $0, 0x0001  # s8 = 0x0000 0001
# 0000 0000 0000 0000 0000 0000 0000 0001
# 0000 0000 0000 0000 0000 0000 0000 0001 << 31 = 1000 0000 0000 0000 0000 0000 0000 0000 (esse numero em binario é 0x8000 0000 em hexadecimal
        sll $8, $8, 0x001F # s8 = 0x1000 0000
        sra $8, $8, 0x001F # s8 = 0xFFFF FFFF
```

Line: 19 Column: 39 Show Line Numbers

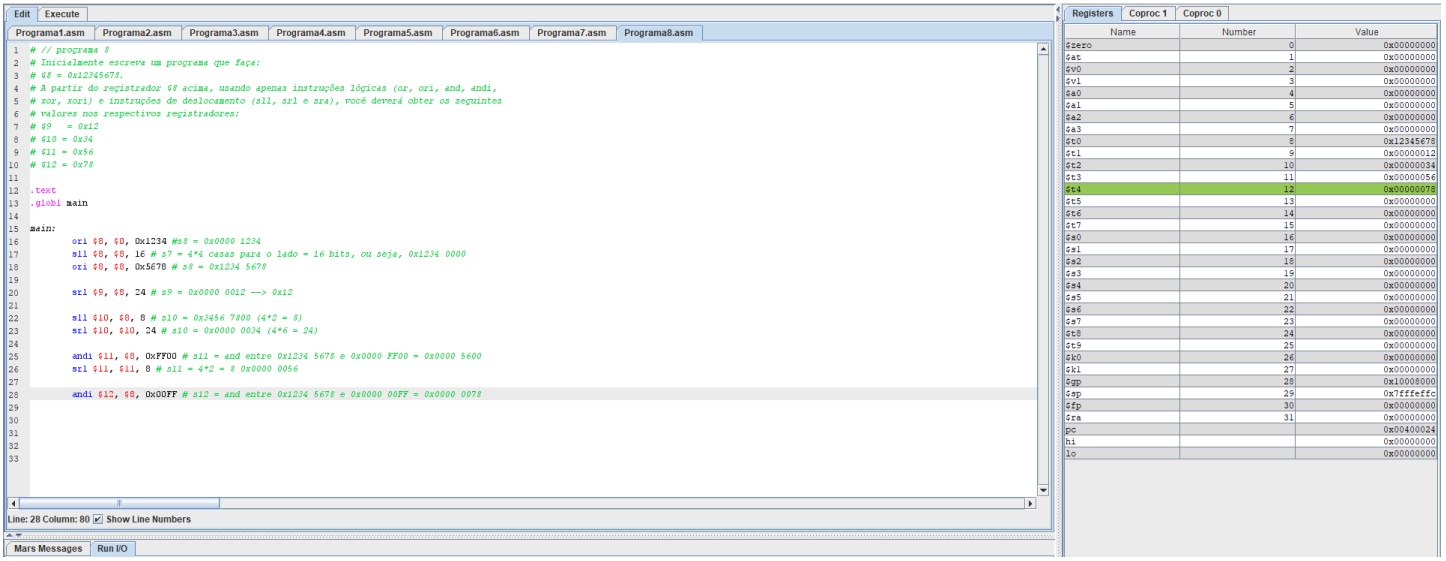
Mars Messages | Run I/O

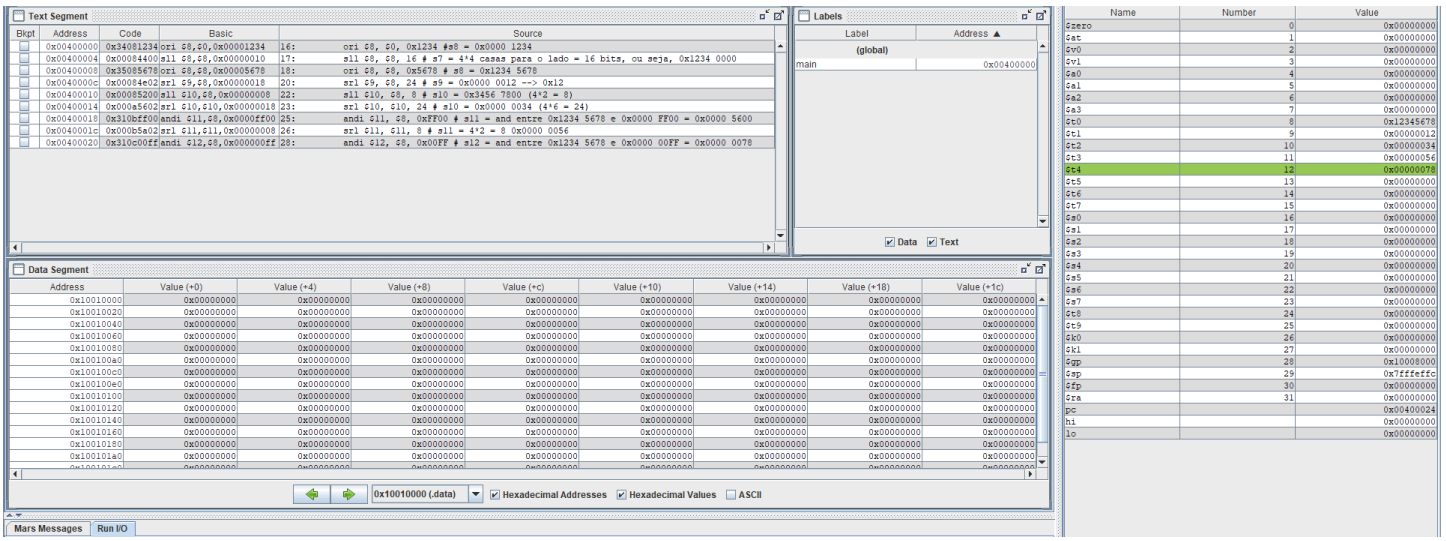
Registers | Coproc 1 | Coproc 0

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		0x0040000c
hi		0x00000000
lo		0x00000000

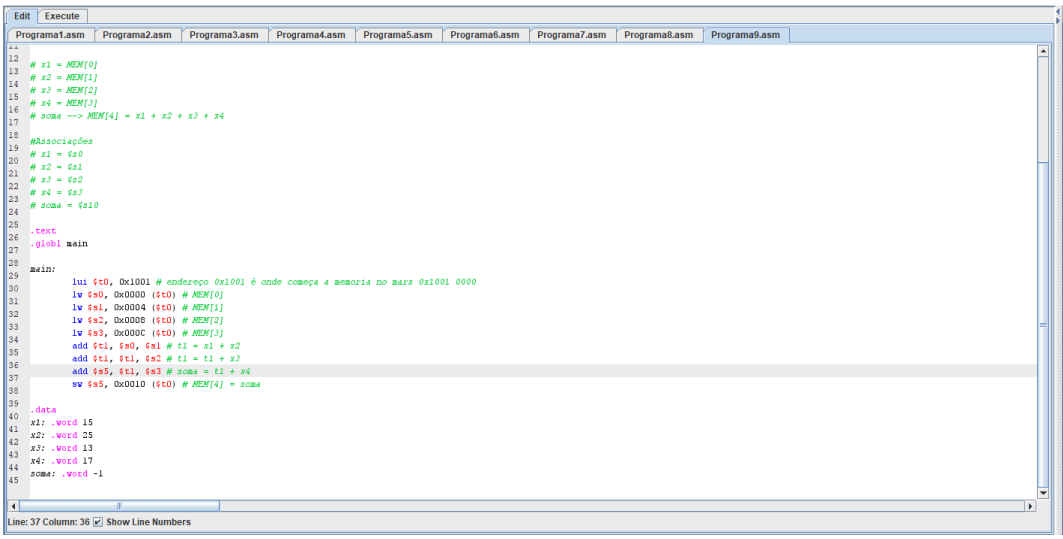

Edit
Execute
Text Segment
Bkpt Address Code Basic Source
0x00400000 0x340B0001 ori $8,$0,0x00000001 15: ori $8, $0, 0x0001 # s8 = 0x0000 0001
0x00400004 0x000847c0 sll $8,$8,0x0000001f 18: sll $8, $8, 0x001F # s8 = 0x1000 0000
0x00400008 0x000847c3 sra $8,$8,0x0000001f 19: sra $8, $8, 0x001F # s8 = 0xFFFF FFFF
Labels
Label Address
(global)
main 0x00400000
Data Text
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
0x10010160 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010180 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100101a0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010000 (.data)
Hexadecimal Addresses
Hexadecimal Values
ASCII
Mars Messages
Run I/O
Registers
Coproc 1
Coproc 0
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 0x0040000c
hi 0x00000000
lo 0x00000000

PROGRAMA 8:

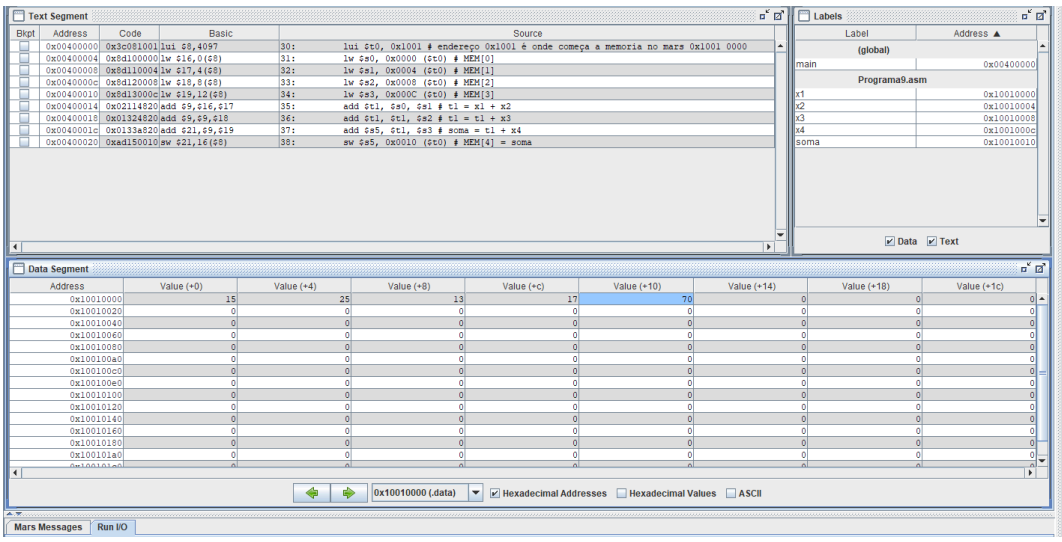

Edit
Execute
Programa1.asm Programa2.asm Programa3.asm Programa4.asm Programa5.asm Programa6.asm Programa7.asm Programa8.asm
1 # // programa 8
2 # Inicialmente escreva um programa que faça:
3 # $8 = 0x12345678.
4 # A partir do registrador $8 acima, usando apenas instruções lógicas (or, ori, and, andi,
5 # xor, xori) e instruções de deslocamento (sll, srl e sra), você deverá obter os seguintes
6 # valores nos respectivos registradores:
7 # $9 = 0x12
8 # $10 = 0x34
9 # $11 = 0x56
10 # $12 = 0x78
11
12 .text
13 .globl main
14
15 main:
16 ori $8, $0, 0x1234 #s8 = 0x0000 1234
17 sll $8, $8, 16 # s7 = 4*4 casas para o lado = 16 bits, ou seja, 0x1234 0000
18 ori $8, $8, 0x5678 # s8 = 0x1234 5678
19
20 srl $9, $8, 24 # s9 = 0x0000 0012 --> 0x12
21
22 sll $10, $8, 8 # s10 = 0x3456 7800 (4*2 = 8)
23 srl $10, $10, 24 # s10 = 0x0000 0034 (4*6 = 24)
24
25 andi $11, $8, 0xFF00 # s11 = and entre 0x1234 5678 e 0x0000 FF00 = 0x0000 5600
26 srl $11, $11, 8 # s11 = 4*2 = 8 0x0000 0056
27
28 andi $12, $8, 0x00FF # s12 = and entre 0x1234 5678 e 0x0000 00FF = 0x0000 0078
29
30
31
32
33
Line: 28 Column: 80
Show Line Numbers
Mars Messages
Run I/O
Registers
Coproc 1
Coproc 0
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 0x00400024
pc 0x00400024
hi 0x00000000
lo 0x00000000

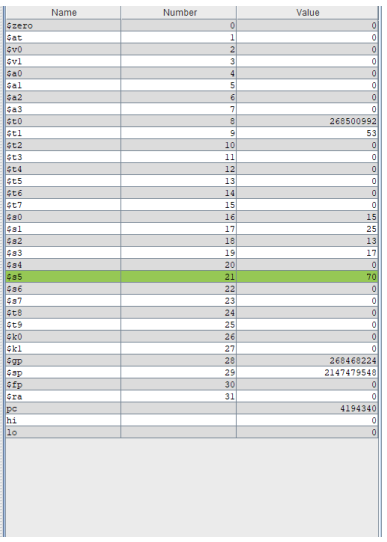

Text Segment
Bkpt Address Code Basic Source
0x00400000 0x34081234 ori $8,$0,0x00001234 16: ori $8, $0, 0x1234 #s8 = 0x0000 1234
0x00400004 0x00084400 sll $8,$8,0x00000010 17: sll $8, $8, 16 # s7 = 4*4 casas para o lado = 16 bits, ou seja, 0x1234 0000
0x00400008 0x35085678 ori $8,$8,0x00005678 18: ori $8, $8, 0x5678 # s8 = 0x1234 5678
0x0040000c 0x00084e02 srl $9,$8,0x00000018 20: srl $9, $8, 24 # s9 = 0x0000 0012 --> 0x12
0x00400010 0x00085200 sll $10,$8,0x00000008 22: sll $10, $8, 8 # s10 = 0x3456 7800 (4*2 = 8)
0x00400014 0x000a5602 srl $10,$10,0x00000018 23: srl $10, $10, 24 # s10 = 0x0000 0034 (4*6 = 24)
0x00400018 0x310bff00 andi $11,$8,0x0000ff00 25: andi $11, $8, 0xFF00 # s11 = and entre 0x1234 5678 e 0x0000 FF00 = 0x0000 5600
0x0040001c 0x000b5a02 srl $11,$11,0x00000008 26: srl $11, $11, 8 # s11 = 4*2 = 8 0x0000 0056
0x00400020 0x310c00ff andi $12,$8,0x000000ff 28: andi $12, $8, 0x00FF # s12 = and entre 0x1234 5678 e 0x0000 00FF = 0x0000 0078
Labels
Label Address
(global)
main 0x00400000
Data Text
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
0x10010160 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010180 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100101a0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010000 (.data)
Hexadecimal Addresses
Hexadecimal Values
ASCII
Mars Messages
Run I/O
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 0x00400024
pc 0x00400024
hi 0x00000000
lo 0x00000000

PROGRAMA 9:

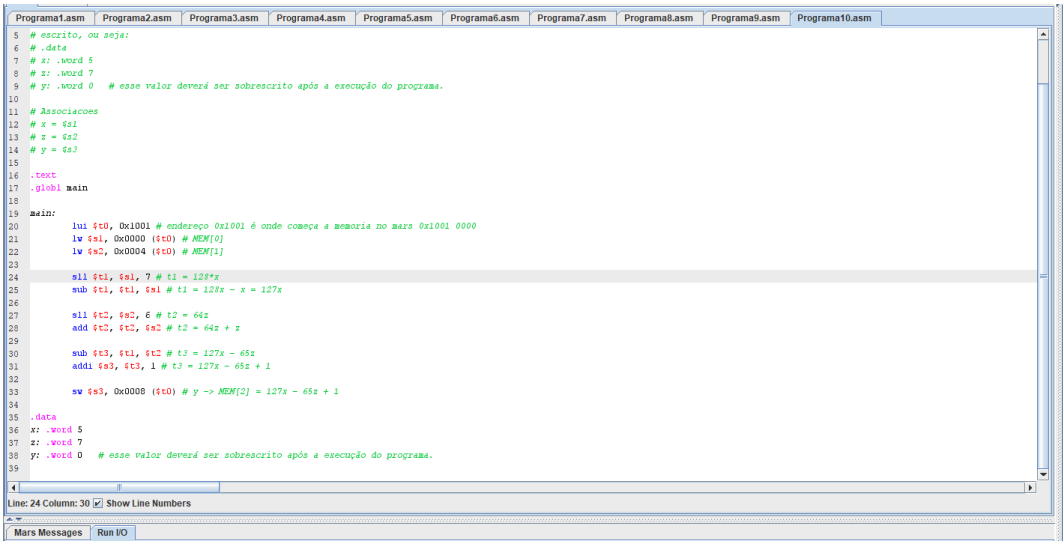


```
# x1 = MEM[0]
# x2 = MEM[1]
# x3 = MEM[2]
# x4 = MEM[3]
# soma --> MEM[4] = x1 + x2 + x3 + x4

#Associações
# x1 = $s0
# x2 = $s1
# x3 = $s2
# x4 = $s3
# soma = $s10

.text
.globl main

main:
        lui $t0, 0x1001 # endereço 0x1001 é onde começa a memoria no mars 0x1001 0000
        lw $s0, 0x0000 ($t0) # MEM[0]
        lw $s1, 0x0004 ($t0) # MEM[1]
        lw $s2, 0x0008 ($t0) # MEM[2]
        lw $s3, 0x000C ($t0) # MEM[3]
        add $t1, $s0, $s1 # t1 = x1 + x2
        add $t1, $t1, $s2 # t1 = t1 + x3
        add $s5, $t1, $s3 # soma = t1 + x4
        sw $s5, 0x0010 ($t0) # MEM[4] = soma

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

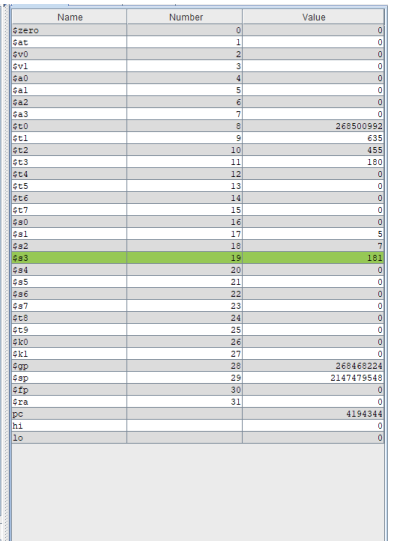


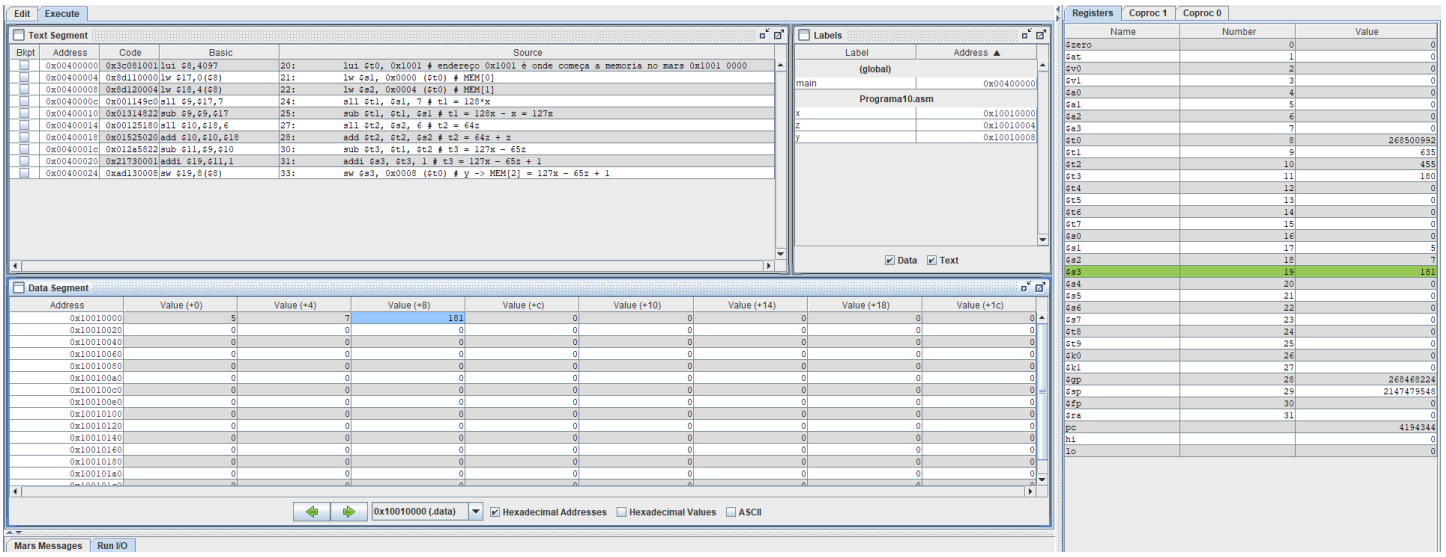


PROGRAMA 10:

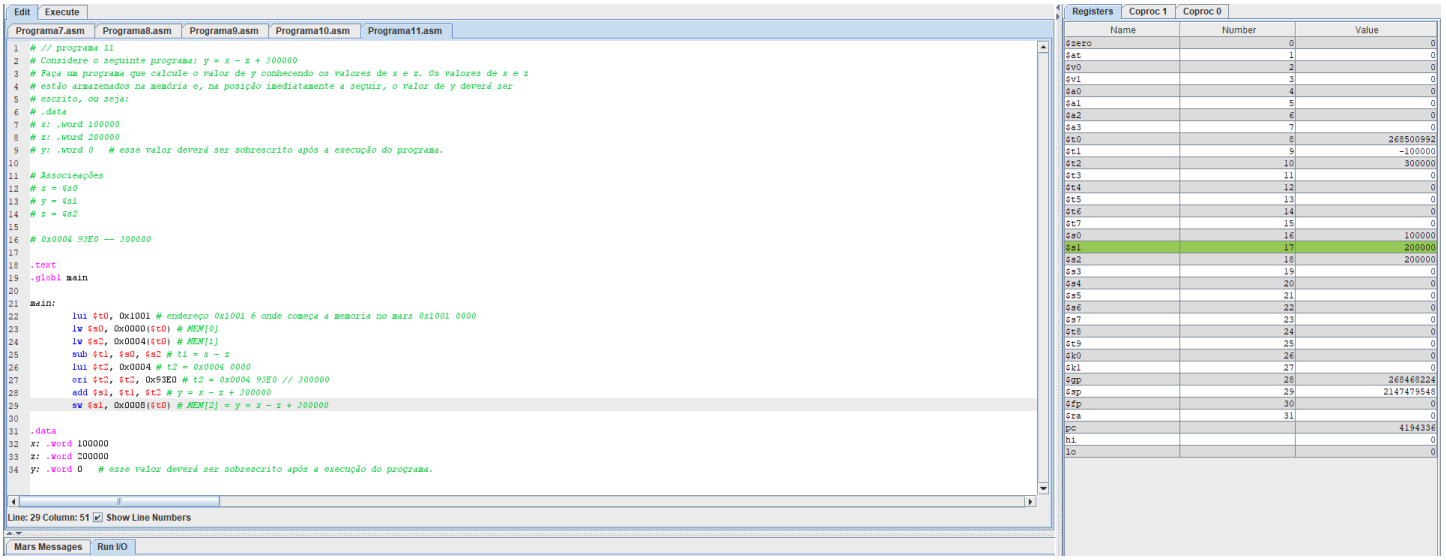


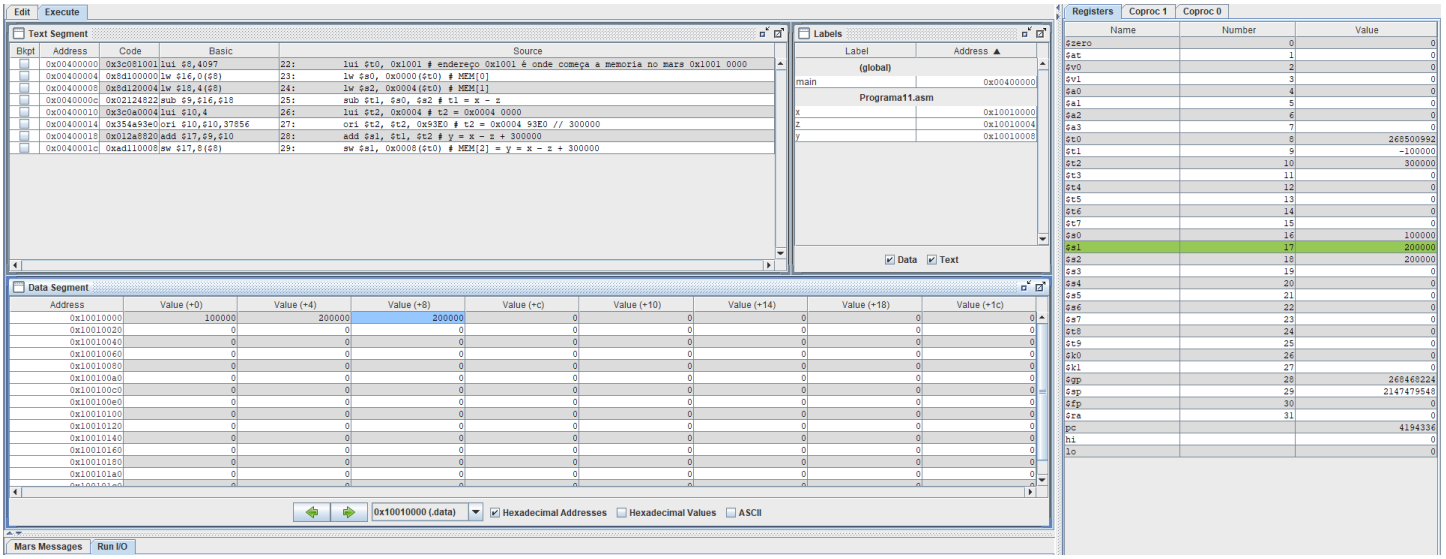
```
# escrito, ou seja:
# .data
# x: .word 5
# z: .word 7
# y: .word 0   # esse valor deverá ser sobrescrito após a execução do programa.

# Associacoes
# x = $s1
# z = $s2
# y = $s3

.text
.globl main

main:
        lui $t0, 0x1001 # endereço 0x1001 é onde começa a memoria no mars 0x1001 0000
        lw $s1, 0x0000 ($t0) # MEM[0]
        lw $s2, 0x0004 ($t0) # MEM[1]

        sll $t1, $s1, 7 # t1 = 128*x
        sub $t1, $t1, $s1 # t1 = 128x - x = 127x

        sll $t2, $s2, 6 # t2 = 64z
        add $t2, $t2, $s2 # t2 = 64z + z

        sub $t3, $t1, $t2 # t3 = 127x - 65z
        addi $s3, $t3, 1 # t3 = 127x - 65z + 1

        sw $s3, 0x0008 ($t0) # y -> MEM[2] = 127x - 65z + 1

.data
x: .word 5
z: .word 7
y: .word 0   # esse valor deverá ser sobrescrito após a execução do programa.
```



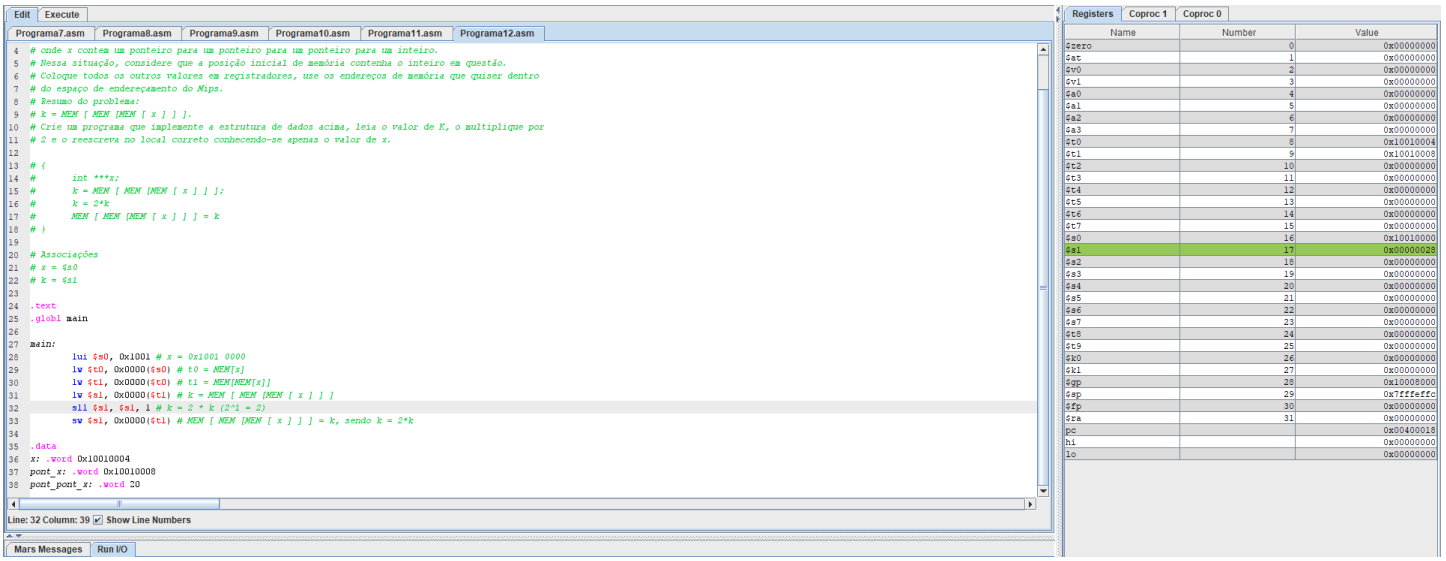

Edit
Execute
Text Segment
Bkpt Address Code Basic Source
0x00400000 0x3c081001 lui $8,4097 20: lui $t0, 0x1001 # endereço 0x1001 é onde começa a memoria no mars 0x1001 0000
0x00400004 0x8d110000 lw $17,0($8) 21: lw $s1, 0x0000 ($t0) # MEM[0]
0x00400008 0x8d120004 lw $18,4($8) 22: lw $s2, 0x0004 ($t0) # MEM[1]
0x0040000c 0x001149c0 sll $9,$17,7 24: sll $t1, $s1, 7 # t1 = 128*x
0x00400010 0x01314822 sub $9,$9,$17 25: sub $t1, $t1, $s1 # t1 = 128x - x = 127x
0x00400014 0x00125180 sll $10,$18,6 27: sll $t2, $s2, 6 # t2 = 64z
0x00400018 0x01525020 add $10,$10,$18 28: add $t2, $t2, $s2 # t2 = 64z + z
0x0040001c 0x012a5822 sub $11,$9,$10 30: sub $t3, $t1, $t2 # t3 = 127x - 65z
0x00400020 0x21730001 addi $19,$11,1 31: addi $s3, $t3, 1 # t3 = 127x - 65z + 1
0x00400024 0xaad30008 sw $19,8($8) 33: sw $s3, 0x0008 ($t0) # y -> MEM[2] = 127x - 65z + 1
Labels
Label Address
(global)
main 0x00400000
Programa10.asm
x 0x10010000
z 0x10010004
y 0x10010008
Data Segment
Address Value (+0) Value (+4) Value (+8) Value (+c) Value (+10) Value (+14) Value (+18) Value (+1c)
0x10010000 5 7 181 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
Mars Messages
Run I/O
Registers
Coproc 1
Coproc 0
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 9 635
$t2 10 455
$t3 11 180
$t4 12 0
$t5 13 0
$t6 14 0
$t7 15 0
$s0 16 0
$s1 17 5
$s2 18 7
$s3 19 181
$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 11:


Edit
Execute
Programa7.asm
Programa8.asm
Programa9.asm
Programa10.asm
Programa11.asm
1 # // programa 11
2 # Considere o seguinte programa: y = x - z + 300000
3 # Faça um programa que calcule o valor de y conhecendo os valores de x e z. Os valores de x e z
4 # estão armazenados na memória e, na posição imediatamente a seguir, o valor de y deverá ser
5 # escrito, ou seja:
6 # .data
7 # x: .word 100000
8 # z: .word 200000
9 # y: .word 0 # esse valor deverá ser sobrescrito após a execução do programa.
10
11 # Associeações
12 # x = $s0
13 # y = $s1
14 # z = $s2
15
16 # 0x0004 93E0 -- 300000
17
18 .text
19 .globl main
20
21 main:
22 lui $t0, 0x1001 # endereço 0x1001 é onde começa a memoria no mars 0x1001 0000
23 lw $s0, 0x0000($t0) # MEM[0]
24 lw $s2, 0x0004($t0) # MEM[1]
25 sub $t1, $s0, $s2 # t1 = x - z
26 lui $t2, 0x0004 # t2 = 0x0004 0000
27 ori $t2, $t2, 0x93E0 # t2 = 0x0004 93E0 // 300000
28 add $s1, $t1, $t2 # y = x - z + 300000
29 sw $s1, 0x0008($t0) # MEM[2] = y = x - z + 300000
30
31 .data
32 x: .word 100000
33 z: .word 200000
34 y: .word 0 # esse valor deverá ser sobrescrito após a execução do programa.
Line: 29 Column: 51
Show Line Numbers
Mars Messages
Run I/O
Registers
Coproc 1
Coproc 0
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 -100000
$t2 10 300000
$t3 11 0
$t4 12 0
$t5 13 0
$t6 14 0
$t7 15 0
$s0 16 100000
$s1 17 200000
$s2 18 200000
$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


Edit
Execute
Text Segment
Bkpt Address Code Basic Source
0x00400000 0x3c081001 lui $8,4097 22: lui $t0, 0x1001 # endereço 0x1001 é onde começa a memoria no mars 0x1001 0000
0x00400004 0x8d100000 lw $16,0($8) 23: lw $s0, 0x0000($t0) # MEM[0]
0x00400008 0x8d120004 lw $18,4($8) 24: lw $s2, 0x0004($t0) # MEM[1]
0x0040000c 0x02124822 sub $9,$16,$18 25: sub $t1, $s0, $s2 # t1 = x - z
0x00400010 0x3c0a0004 lui $10,4 26: lui $t2, 0x0004 # t2 = 0x0004 0000
0x00400014 0x354a93e0 ori $10,$10,37856 27: ori $t2, $t2, 0x93E0 # t2 = 0x0004 93E0 // 300000
0x00400018 0x012a8820 add $17,$9,$10 28: add $s1, $t1, $t2 # y = x - z + 300000
0x0040001c 0xad110008 sw $17,8($8) 29: sw $s1, 0x0008($t0) # MEM[2] = y = x - z + 300000
Labels
Label Address
(global)
main 0x00400000
Programa11.asm
x 0x10010000
z 0x10010004
y 0x10010008
Data Segment
Address Value (+0) Value (+4) Value (+8) Value (+c) Value (+10) Value (+14) Value (+18) Value (+1c)
0x10010000 100000 200000 200000 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
Mars Messages
Run I/O
Registers
Coproc 1
Coproc 0
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 -100000
$t2 10 300000
$t3 11 0
$t4 12 0
$t5 13 0
$t6 14 0
$t7 15 0
$s0 16 100000
$s1 17 200000
$s2 18 200000
$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 12:



```
# 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.

# {
#       int ***x;
#       k = MEM [ MEM [MEM [ x ] ] ];
#       k = 2*k
#       MEM [ MEM [MEM [ x ] ] ] = k
# }

# Associações
# x = $s0
# k = $s1

.text
.globl main

main:
        lui $s0, 0x1001 # x = 0x1001 0000
        lw $t0, 0x0000($s0) # t0 = MEM[x]
        lw $t1, 0x0000($t0) # t1 = MEM[MEM[x]]
        lw $s1, 0x0000($t1) # k = MEM [ MEM [MEM [ x ] ] ]
        sll $s1, $s1, 1 # k = 2 * k (2^1 = 2)
        sw $s1, 0x0000($t1) # MEM [ MEM [MEM [ x ] ] ] = k, sendo k = 2*k

.data
x: .word 0x10010004
pont_x: .word 0x10010008
pont_pont_x: .word 20
```

Line: 32 Column: 39 — Show Line Numbers

Mars Messages | Run I/O

Registers | Coproc 1 | Coproc 0

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	0x10010004
$t1	9	0x10010008
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x10010000
$s1	17	0x00000028
$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		0x00400018
hi		0x00000000
lo		0x00000000

Text Segment

Bkpt	Address	Code	Basic		Source
	0x00400000	0x3c101001	lui $16,0x00001001	28:	lui $s0, 0x1001 # x = 0x1001 0000
	0x00400004	0x8e080000	lw $8,0x00000000($16)	29:	lw $t0, 0x0000($s0) # t0 = MEM[x]
	0x00400008	0x8d090000	lw $9,0x00000000($8)	30:	lw $t1, 0x0000($t0) # t1 = MEM[MEM[x]]
	0x0040000c	0x8d310000	lw $17,0x00000000($9)	31:	lw $s1, 0x0000($t1) # k = MEM [MEM [MEM [x]]]
	0x00400010	0x00118840	sll $17,$17,0x00000001	32:	sll $s1, $s1, 1 # k = 2 * k (2^1 = 2)
	0x00400014	0xad310000	sw $17,0x00000000($9)	33:	sw $s1, 0x0000($t1) # MEM [MEM [MEM [x]]] = k, sendo k = 2*k

Labels

Label	Address
(global)	
main	0x00400000
Programa12.asm	
x	0x10010000
pont_x	0x10010004
pont_pont_x	0x10010008

Data | Text

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x10010004	0x10010008	0x00000028	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
0x10010160	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010180	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

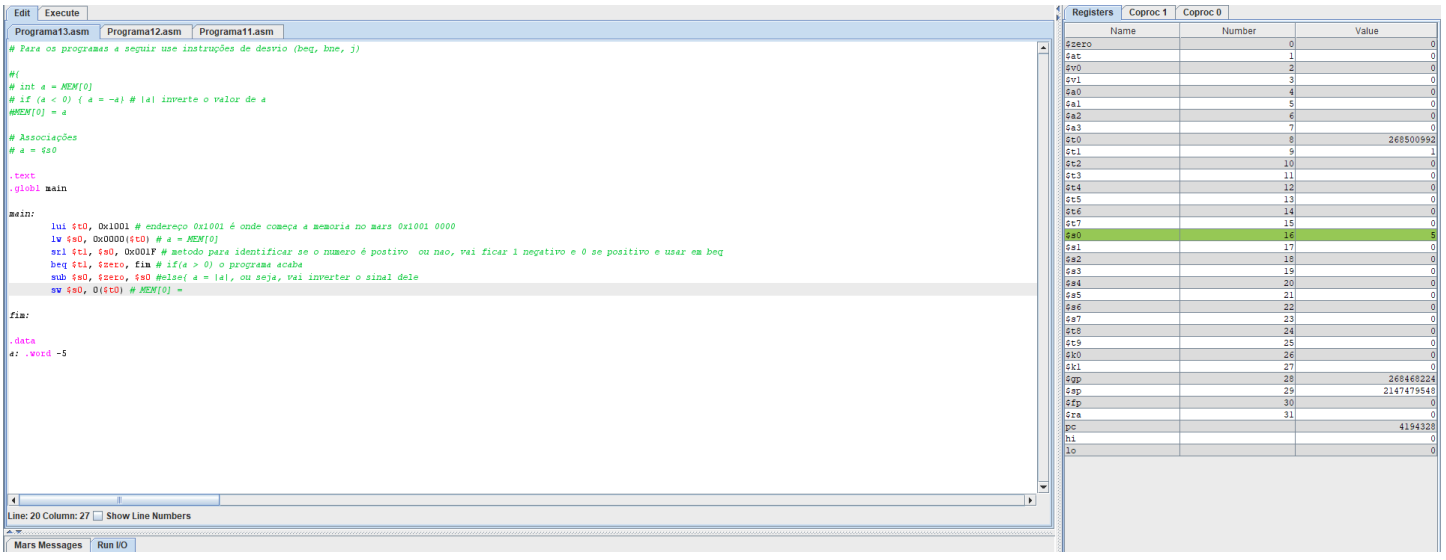
0x10010000 (.data) — Hexadecimal Addresses — Hexadecimal Values — ASCII

Mars Messages | Run I/O

Registers | Coproc 1 | Coproc 0

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	0x10010004
$t1	9	0x10010008
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x10010000
$s1	17	0x00000028
$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		0x00400018
hi		0x00000000
lo		0x00000000

PROGRAMA 13:



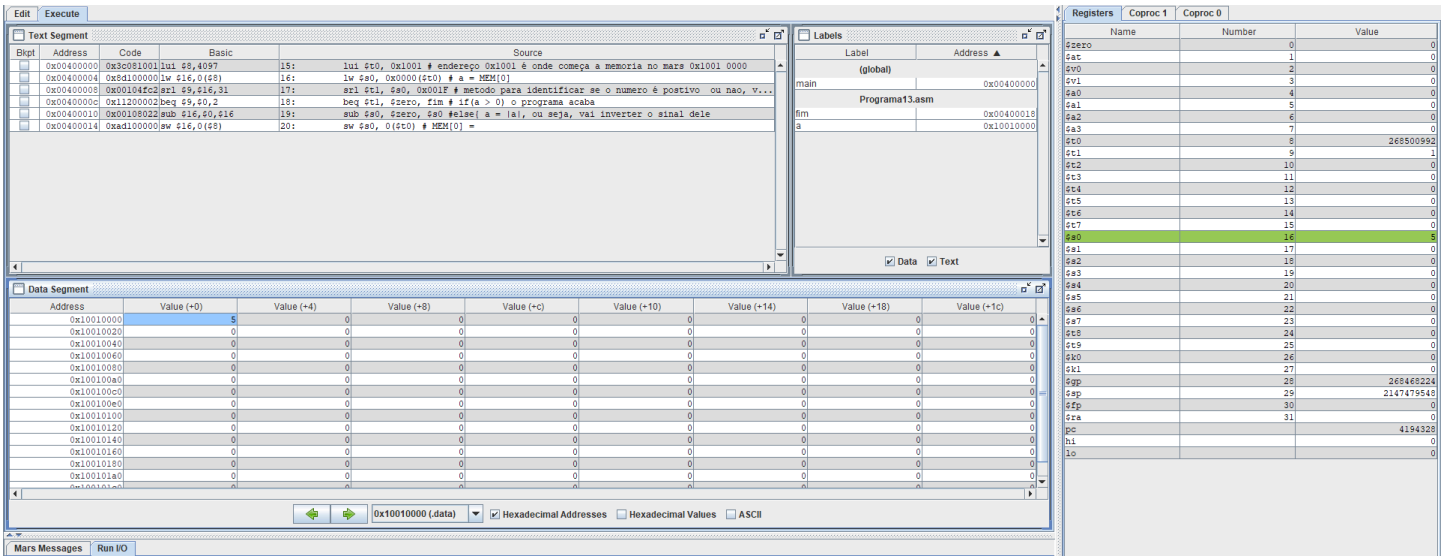
```
# Para os programas a seguir use instruções de desvio (beq, bne, j)

#{
# int a = MEM[0]
# if (a < 0) { a = -a} # |a| inverte o valor de a
#MEM[0] = a

# Associações
# a = $s0

.text
.globl main

main:
        lui $t0, 0x1001 # endereço 0x1001 é onde começa a memoria no mars 0x1001 0000
        lw $s0, 0x0000($t0) # a = MEM[0]
        srl $t1, $s0, 0x001F # metodo para identificar se o numero é postivo  ou nao, vai ficar 1 negativo e 0 se positivo e usar em beq
        beq $t1, $zero, fim # if(a > 0) o programa acaba
        sub $s0, $zero, $s0 #else( a = |a|, ou seja, vai inverter o sinal dele
        sw $s0, 0($t0) # MEM[0] =

fim:

.data
a: .word -5
```

Line: 20 Column: 27 — Show Line Numbers

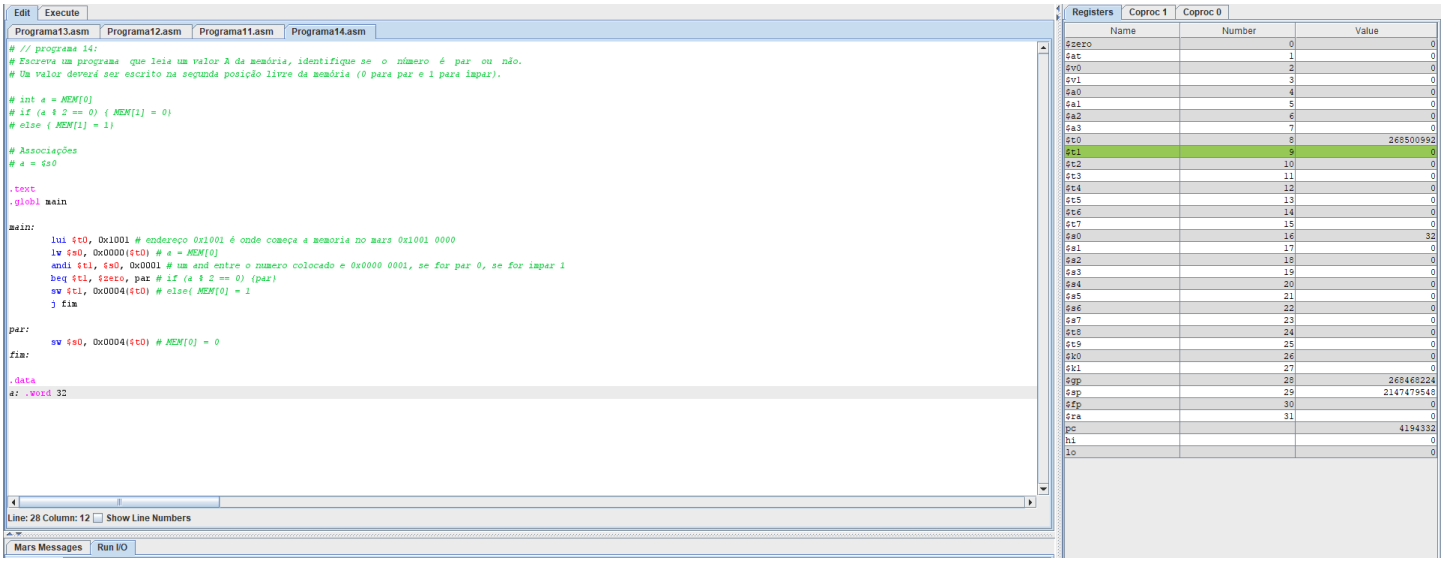
Mars Messages | Run I/O

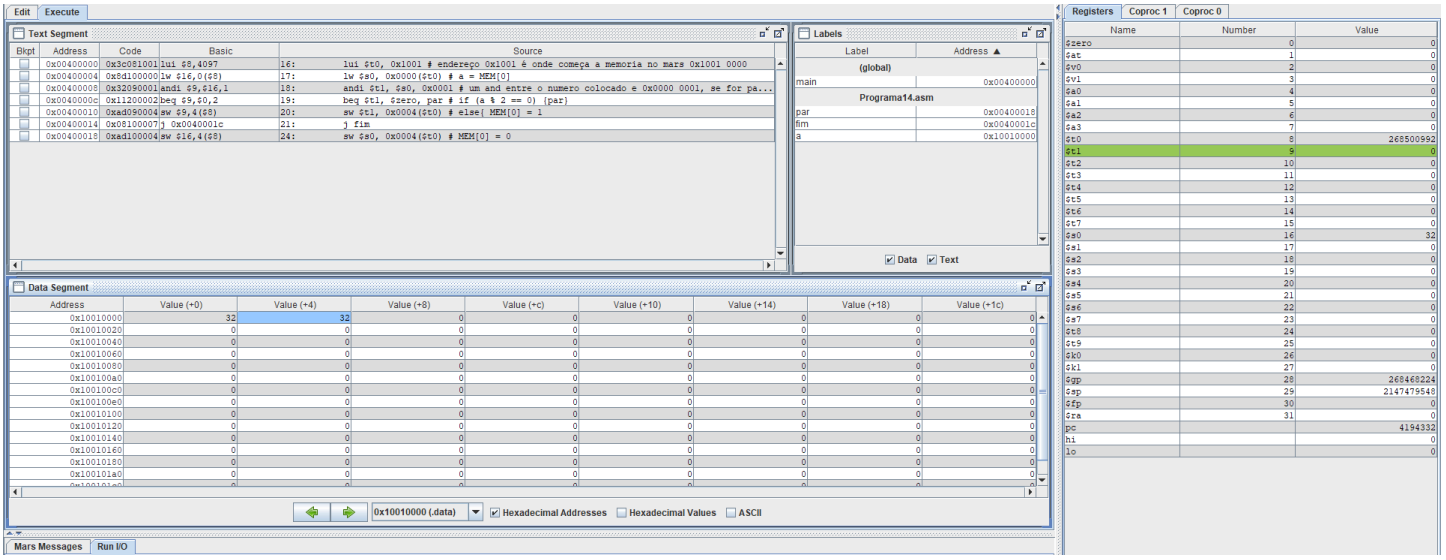
Registers | Coproc 1 | Coproc 0

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	1
$t2	10	0
$t3	11	0
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	5
$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		4194328
hi		0
lo		0

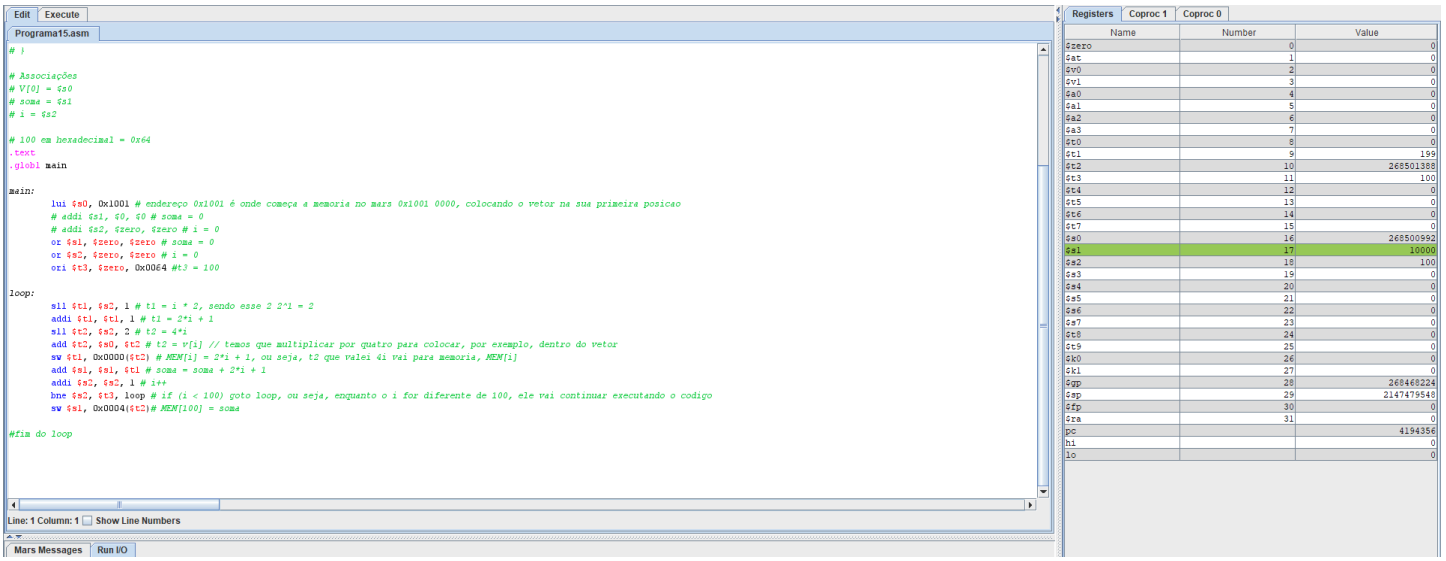

Edit
Execute
Text Segment
Bkpt Address Code Basic Source
0x00400000 0x3c081001 lui $8,4097 15: lui $t0, 0x1001 # endereço 0x1001 é onde começa a memoria no mars 0x1001 0000
0x00400004 0x8d100000 lw $16,0($8) 16: lw $s0, 0x0000($t0) # a = MEM[0]
0x00400008 0x00104fc2 srl $9,$16,31 17: srl $t1, $s0, 0x001F # metodo para identificar se o numero é postivo ou nao, v...
0x0040000c 0x11200002 beq $9,$0,2 18: beq $t1, $zero, fim # if(a > 0) o programa acaba
0x00400010 0x00108022 sub $16,$0,$16 19: sub $s0, $zero, $s0 #else{ a = |a|, ou seja, vai inverter o sinal dele
0x00400014 0xad100000 sw $16,0($8) 20: sw $s0, 0($t0) # MEM[0] =
Labels
Label Address
(global)
main 0x00400000
Programa13.asm
fim 0x00400018
a 0x10010000
Data Text
Data Segment
Address Value (+0) Value (+4) Value (+8) Value (+c) Value (+10) Value (+14) Value (+18) Value (+1c)
0x10010000 5 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
Mars Messages
Run I/O
Registers
Coproc 1
Coproc 0
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 1
$t2 10 0
$t3 11 0
$t4 12 0
$t5 13 0
$t6 14 0
$t7 15 0
$s0 16 5
$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 4194328
hi 0
lo 0

PROGRAMA 14:


Edit
Execute
Programa13.asm
Programa12.asm
Programa11.asm
Programa14.asm
// 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).
int a = MEM[0]
if (a % 2 == 0) { MEM[1] = 0}
else { MEM[1] = 1}
Associações
a = $s0
.text
.globl main
main:
lui $t0, 0x1001 # endereço 0x1001 é onde começa a memoria no mars 0x1001 0000
lw $s0, 0x0000($t0) # a = MEM[0]
andi $t1, $s0, 0x0001 # um and entre o numero colocado e 0x0000 0001, se for par 0, se for impar 1
beq $t1, $zero, par # if (a % 2 == 0) {par}
sw $t1, 0x0004($t0) # else{ MEM[0] = 1
j fim
par:
sw $s0, 0x0004($t0) # MEM[0] = 0
fim:
.data
a: .word 32
Line: 28 Column: 12
Show Line Numbers
Mars Messages
Run I/O
Registers
Coproc 1
Coproc 0
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 0
$t2 10 0
$t3 11 0
$t4 12 0
$t5 13 0
$t6 14 0
$t7 15 0
$s0 16 32
$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 4194332
hi 0
lo 0


Edit
Execute
Text Segment
Bkpt Address Code Basic Source
0x00400000 0x3c081001 lui $8,4097 16: lui $t0, 0x1001 # endereço 0x1001 é onde começa a memoria no mars 0x1001 0000
0x00400004 0x8d100000 lw $16,0($8) 17: lw $s0, 0x0000($t0) # a = MEM[0]
0x00400008 0x32090001 andi $9,$16,1 18: andi $t1, $s0, 0x0001 # um and entre o numero colocado e 0x0000 0001, se for pa...
0x0040000c 0x11200002 beq $9,$0,2 19: beq $t1, $zero, par # if (a % 2 == 0) {par}
0x00400010 0xad090004 sw $9,4($8) 20: sw $t1, 0x0004($t0) # else{ MEM[0] = 1
0x00400014 0x08100007 j 0x0040001c 21: j fim
0x00400018 0xad100004 sw $16,4($8) 24: sw $s0, 0x0004($t0) # MEM[0] = 0
Labels
Label Address
(global)
main 0x00400000
Programa14.asm
par 0x00400018
fim 0x0040001c
a 0x10010000
Data Text
Data Segment
Address Value (+0) Value (+4) Value (+8) Value (+c) Value (+10) Value (+14) Value (+18) Value (+1c)
0x10010000 32 32 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
Mars Messages
Run I/O
Registers
Coproc 1
Coproc 0
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 0
$t2 10 0
$t3 11 0
$t4 12 0
$t5 13 0
$t6 14 0
$t7 15 0
$s0 16 32
$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 4194332
hi 0
lo 0

PROGRAMA 15:



Edit Execute

Programa15.asm

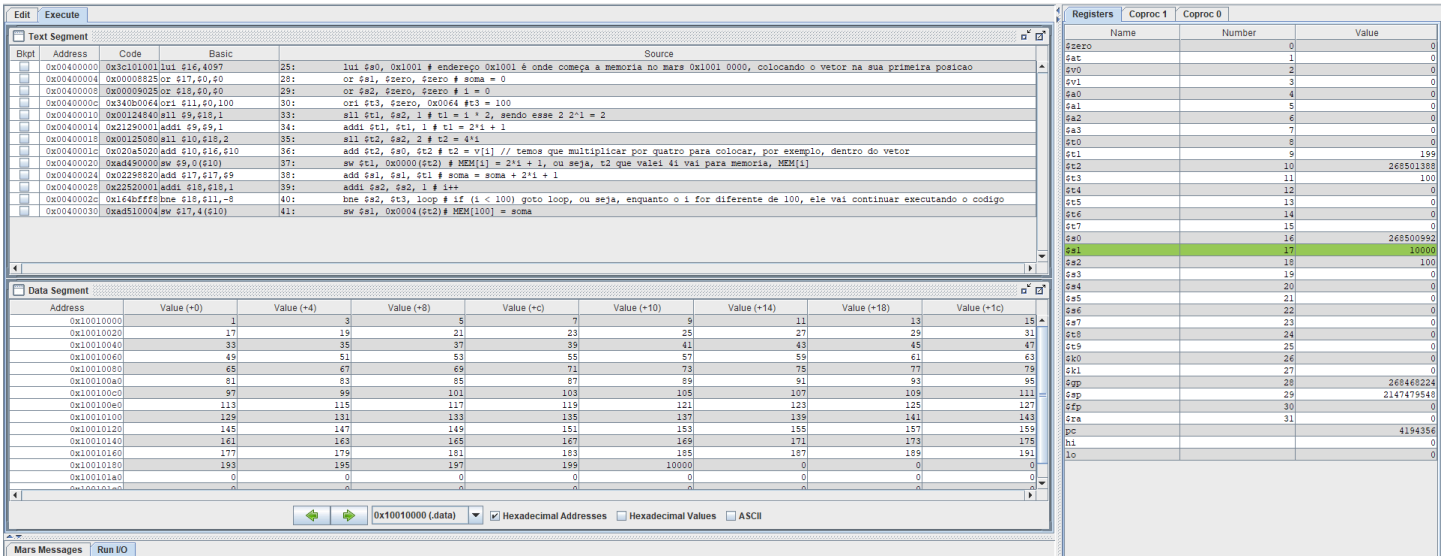
```
# }

# Associações
# V[0] = $s0
# soma = $s1
# i = $s2

# 100 em hexadecimal = 0x64
.text
.globl main

main:
        lui $s0, 0x1001 # endereço 0x1001 é onde começa a memoria no mars 0x1001 0000, colocando o vetor na sua primeira posicao
        # addi $s1, $0, $0 # soma = 0
        # addi $s2, $zero, $zero # i = 0
        or $s1, $zero, $zero # soma = 0
        or $s2, $zero, $zero # i = 0
        ori $t3, $zero, 0x0064 #t3 = 100

loop:
        sll $t1, $s2, 1 # t1 = i * 2, sendo esse 2 2^1 = 2
        addi $t1, $t1, 1 # t1 = 2*i + 1
        sll $t2, $s2, 2 # t2 = 4*i
        add $t2, $s0, $t2 # t2 = v[i] // temos que multiplicar por quatro para colocar, por exemplo, dentro do vetor
        sw $t1, 0x0000($t2) # MEM[i] = 2*i + 1, ou seja, t2 que valei 4i vai para memoria, MEM[i]
        add $s1, $s1, $t1 # soma = soma + 2*i + 1
        addi $s2, $s2, 1 # i++
        bne $s2, $t3, loop # if (i < 100) goto loop, ou seja, enquanto o i for diferente de 100, ele vai continuar executando o codigo
        sw $s1, 0x0004($t2)# MEM[100] = soma

#fim do loop
```

Line: 1 Column: 1 Show Line Numbers

Mars Messages Run I/O

Registers Coproc 1 Coproc 0

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	0
$t1	9	199
$t2	10	268501388
$t3	11	100
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	268500992
$s1	17	10000
$s2	18	100
$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		4194356
hi		0
lo		0



Edit Execute

Text Segment

Bkpt	Address	Code	Basic		Source
	0x00400000	0x3c101001	lui $16,4097	25:	lui $s0, 0x1001 # endereço 0x1001 é onde começa a memoria no mars 0x1001 0000, colocando o vetor na sua primeira posicao
	0x00400004	0x00008825	or $17,$0,$0	28:	or $s1, $zero, $zero # soma = 0
	0x00400008	0x00009025	or $18,$0,$0	29:	or $s2, $zero, $zero # i = 0
	0x0040000c	0x340b0064	ori $11,$0,100	30:	ori $t3, $zero, 0x0064 #t3 = 100
	0x00400010	0x00124840	sll $9,$18,1	33:	sll $t1, $s2, 1 # t1 = i * 2, sendo esse 2 2^1 = 2
	0x00400014	0x21290001	addi $9,$9,1	34:	addi $t1, $t1, 1 # t1 = 2*i + 1
	0x00400018	0x00125080	sll $10,$18,2	35:	sll $t2, $s2, 2 # t2 = 4*i
	0x0040001c	0x020a5020	add $10,$16,$10	36:	add $t2, $s0, $t2 # t2 = v[i] // temos que multiplicar por quatro para colocar, por exemplo, dentro do vetor
	0x00400020	0xad490000	sw $9,0($10)	37:	sw $t1, 0x0000($t2) # MEM[i] = 2*i + 1, ou seja, t2 que valei 4i vai para memoria, MEM[i]
	0x00400024	0x02298820	add $17,$17,$9	38:	add $s1, $s1, $t1 # soma = soma + 2*i + 1
	0x00400028	0x22520001	addi $18,$18,1	39:	addi $s2, $s2, 1 # i++
	0x0040002c	0x164bfff8	bne $18,$11,-8	40:	bne $s2, $t3, loop # if (i < 100) goto loop, ou seja, enquanto o i for diferente de 100, ele vai continuar executando o codigo
	0x00400030	0xad510004	sw $17,4($10)	41:	sw $s1, 0x0004($t2)# MEM[100] = soma

Data Segment

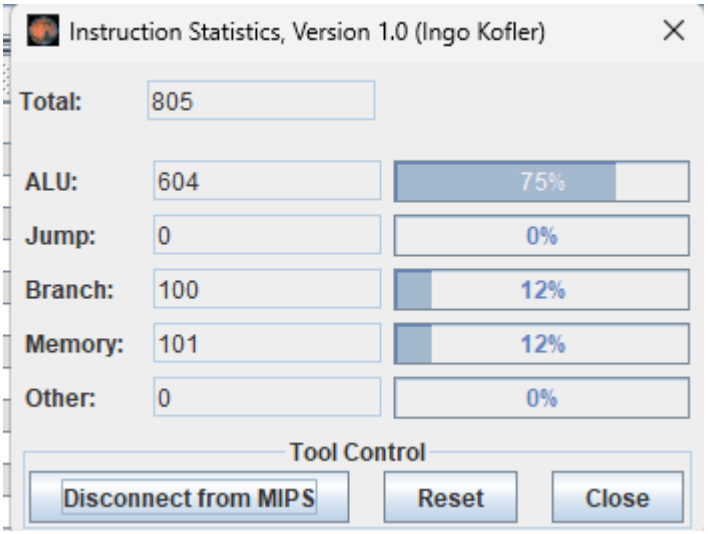
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	1	3	5	7	9	11	13	15
0x10010020	17	19	21	23	25	27	29	31
0x10010040	33	35	37	39	41	43	45	47
0x10010060	49	51	53	55	57	59	61	63
0x10010080	65	67	69	71	73	75	77	79
0x100100a0	81	83	85	87	89	91	93	95
0x100100c0	97	99	101	103	105	107	109	111
0x100100e0	113	115	117	119	121	123	125	127
0x10010100	129	131	133	135	137	139	141	143
0x10010120	145	147	149	151	153	155	157	159
0x10010140	161	163	165	167	169	171	173	175
0x10010160	177	179	181	183	185	187	189	191
0x10010180	193	195	197	199	10000	0	0	0
0x100101a0	0	0	0	0	0	0	0	0

0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII

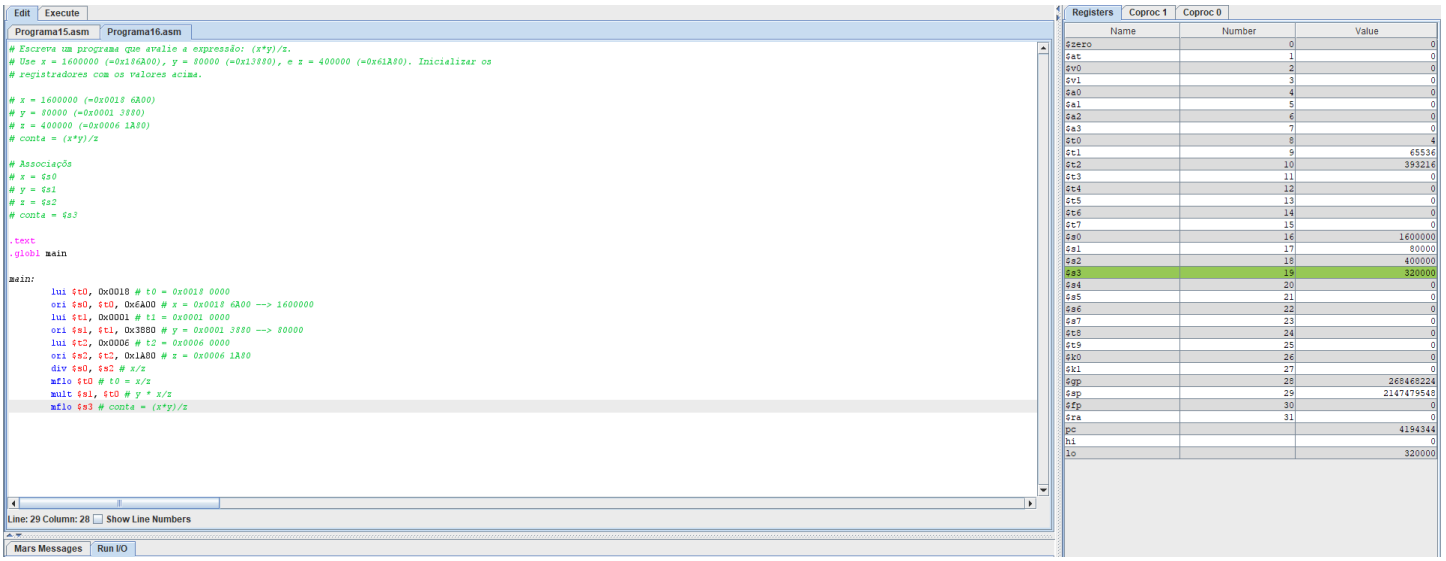
Mars Messages Run I/O

Registers Coproc 1 Coproc 0

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	0
$t1	9	199
$t2	10	268501388
$t3	11	100
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	268500992
$s1	17	10000
$s2	18	100
$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		4194356
hi		0
lo		0



PROGRAMA 16:

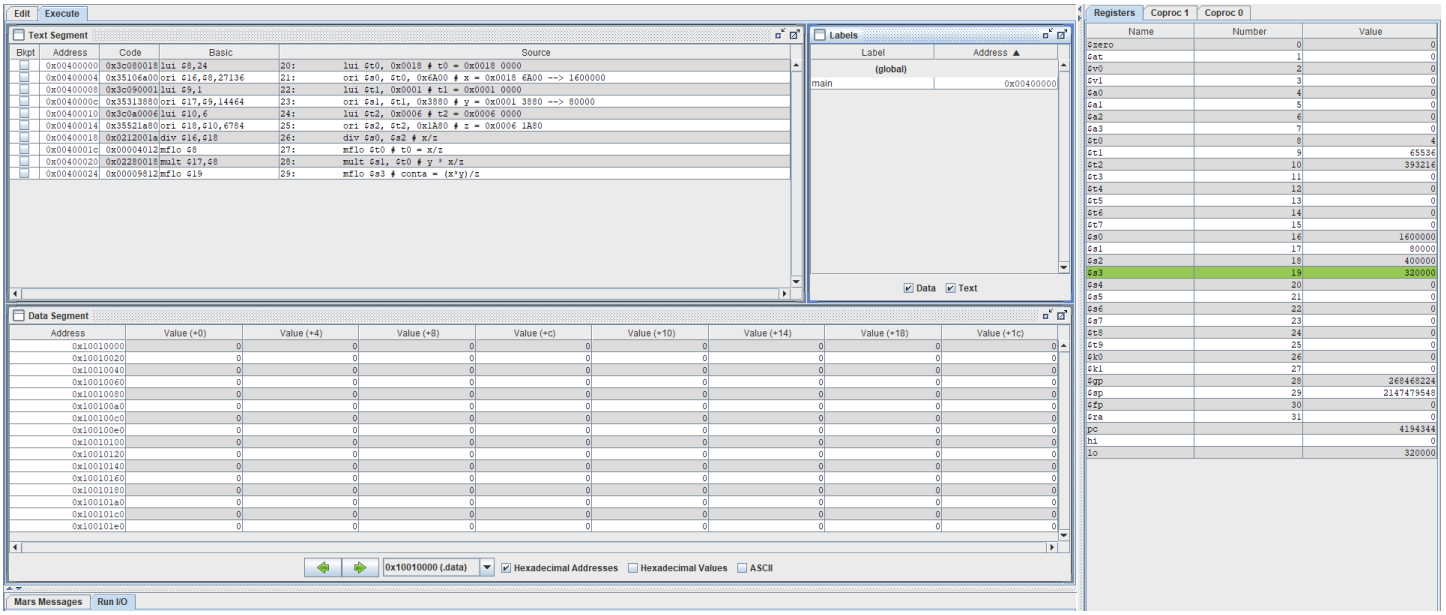


```
# Escreva um programa que avalie a expressão: (x*y)/z.
# Use x = 1600000 (=0x186A00), y = 80000 (=0x13880), e z = 400000 (=0x61A80). Inicializar os
# registradores com os valores acima.

# x = 1600000 (=0x0018 6A00)
# y = 80000 (=0x0001 3880)
# z = 400000 (=0x0006 1A80)
# conta = (x*y)/z

# Associações
# x = $s0
# y = $s1
# z = $s2
# conta = $s3

.text
.globl main

main:
        lui $t0, 0x0018 # t0 = 0x0018 0000
        ori $s0, $t0, 0x6A00 # x = 0x0018 6A00 --> 1600000
        lui $t1, 0x0001 # t1 = 0x0001 0000
        ori $s1, $t1, 0x3880 # y = 0x0001 3880 --> 80000
        lui $t2, 0x0006 # t2 = 0x0006 0000
        ori $s2, $t2, 0x1A80 # z = 0x0006 1A80
        div $s0, $s2 # x/z
        mflo $t0 # t0 = x/z
        mult $s1, $t0 # y * x/z
        mflo $s3 # conta = (x*y)/z
```

Line: 29 Column: 28 Show Line Numbers

Mars Messages | Run I/O

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	65536
$t2	10	393216
$t3	11	0
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	1600000
$s1	17	80000
$s2	18	400000
$s3	19	320000
$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		320000



Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x3c080018	lui $8,24	20: lui $t0, 0x0018 # t0 = 0x0018 0000
	0x00400004	0x35106a00	ori $16,$8,27136	21: ori $s0, $t0, 0x6A00 # x = 0x0018 6A00 --> 1600000
	0x00400008	0x3c090001	lui $9,1	22: lui $t1, 0x0001 # t1 = 0x0001 0000
	0x0040000c	0x35313880	ori $17,$9,14464	23: ori $s1, $t1, 0x3880 # y = 0x0001 3880 --> 80000
	0x00400010	0x3c0a0006	lui $10,6	24: lui $t2, 0x0006 # t2 = 0x0006 0000
	0x00400014	0x35521a80	ori $18,$10,6784	25: ori $s2, $t2, 0x1A80 # z = 0x0006 1A80
	0x00400018	0x0212001a	div $16,$18	26: div $s0, $s2 # x/z
	0x0040001c	0x00004012	mflo $8	27: mflo $t0 # t0 = x/z
	0x00400020	0x02280018	mult $17,$8	28: mult $s1, $t0 # y * x/z
	0x00400024	0x00009812	mflo $19	29: mflo $s3 # conta = (x*y)/z

Labels

Label	Address
(global)	
main	0x00400000

Data Segment

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
0x100101c0	0	0	0	0	0	0	0	0
0x100101e0	0	0	0	0	0	0	0	0

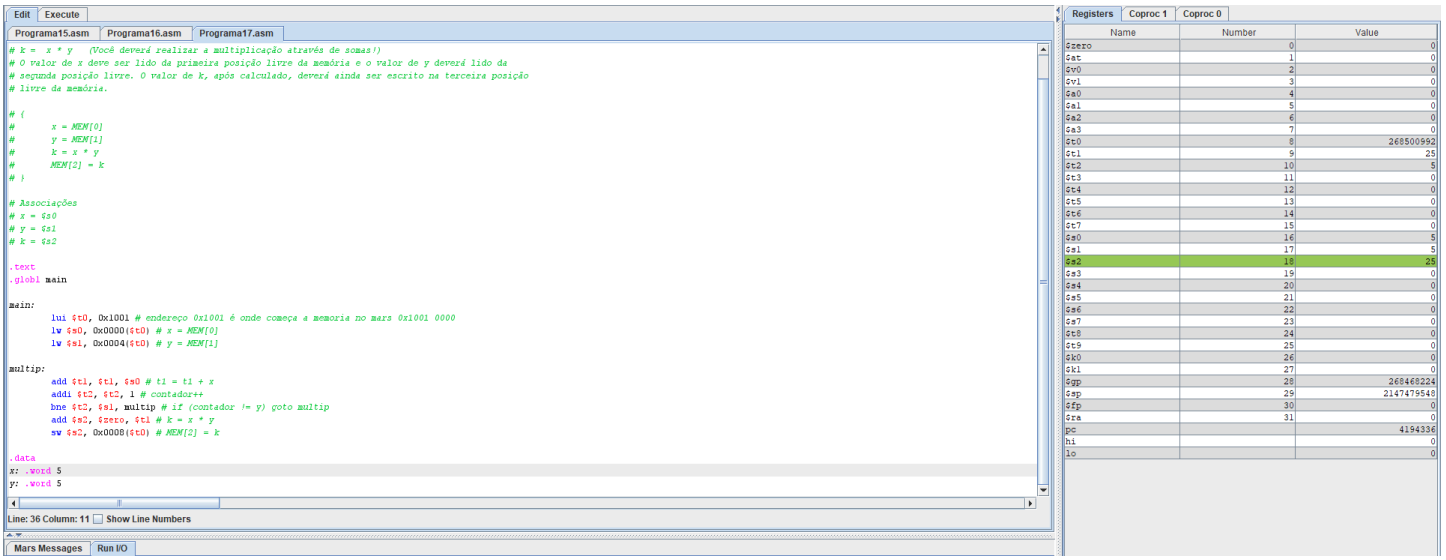
0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII

Data Text

Mars Messages | Run I/O

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	65536
$t2	10	393216
$t3	11	0
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	1600000
$s1	17	80000
$s2	18	400000
$s3	19	320000
$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		320000

PROGRAMA 17:

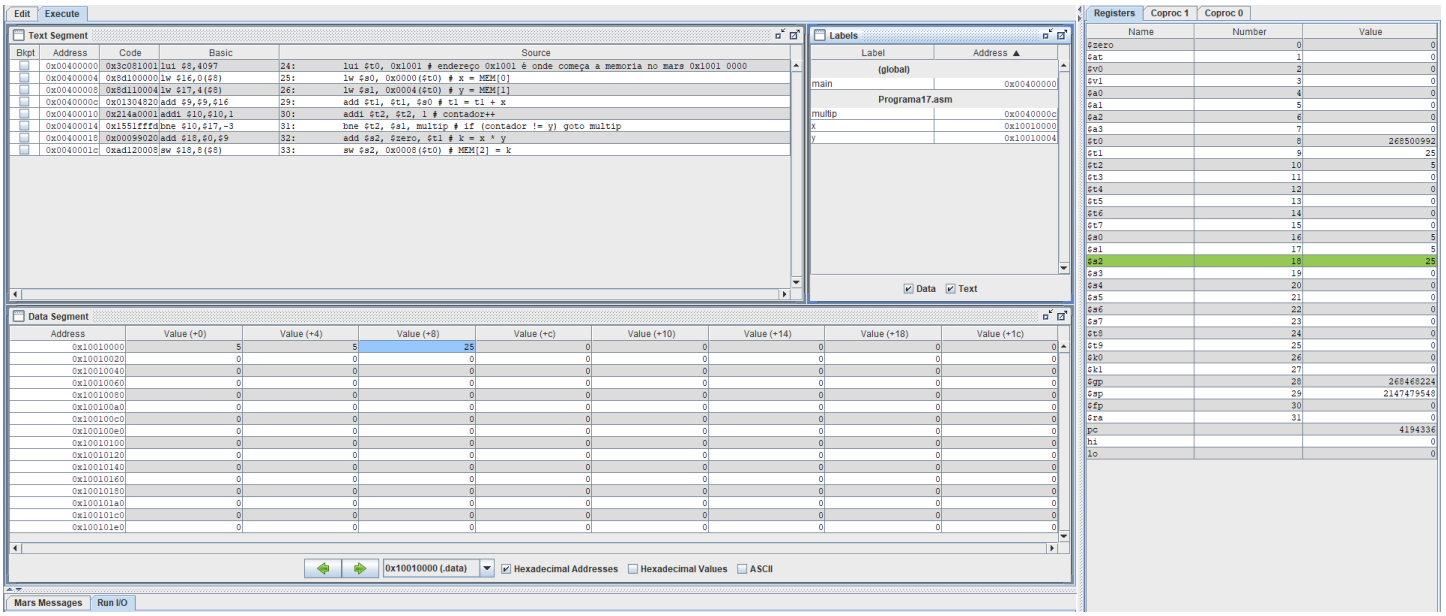


```
# k =  x * y   (Você deverá realizar a multiplicação através de somas!)
# O valor de x deve ser lido da primeira posição livre da memória e o valor de y deverá lido da
# segunda posição livre. O valor de k, após calculado, deverá ainda ser escrito na terceira posição
# livre da memória.

# (
#       x = MEM[0]
#       y = MEM[1]
#       k = x * y
#       MEM[2] = k
# )

# Associações
# x = $s0
# y = $s1
# k = $s2

.text
.globl main

main:
        lui $t0, 0x1001 # endereço 0x1001 é onde começa a memoria no mars 0x1001 0000
        lw $s0, 0x0000($t0) # x = MEM[0]
        lw $s1, 0x0004($t0) # y = MEM[1]

multip:
        add $t1, $t1, $s0 # t1 = t1 + x
        addi $t2, $t2, 1 # contador++
        bne $t2, $s1, multip # if (contador != y) goto multip
        add $s2, $zero, $t1 # k = x * y
        sw $s2, 0x0008($t0) # MEM[2] = k

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

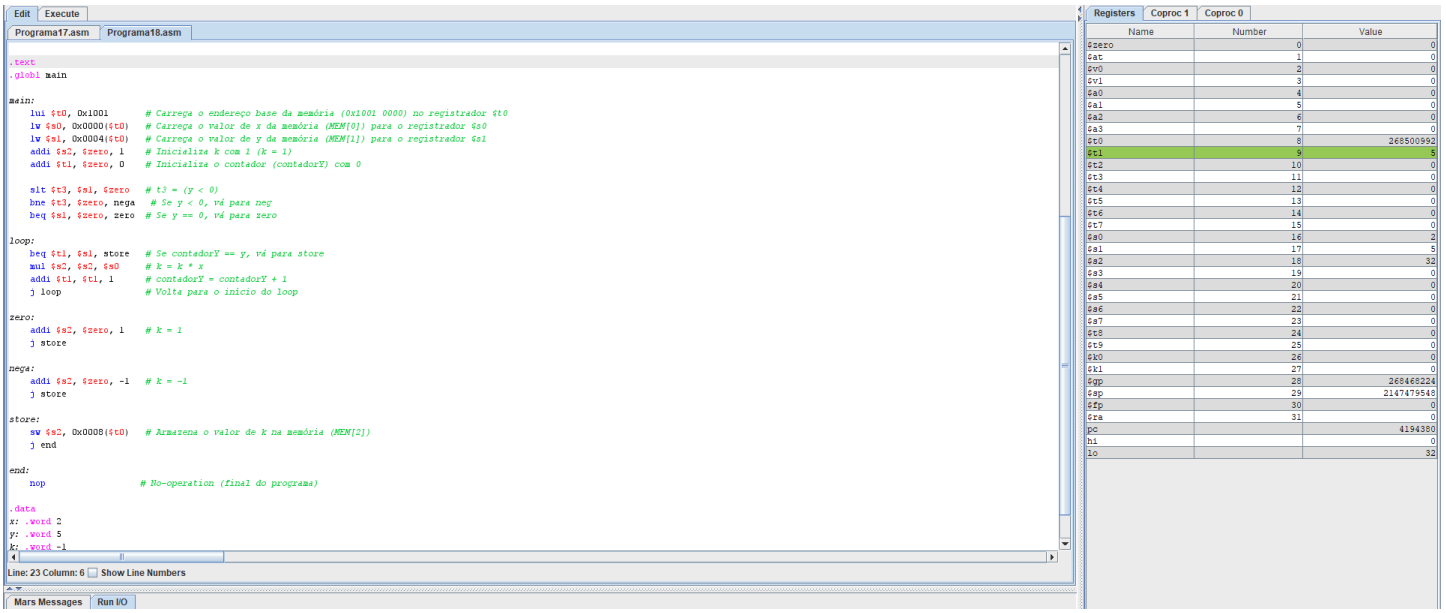
Line: 36 Column: 11 Show Line Numbers

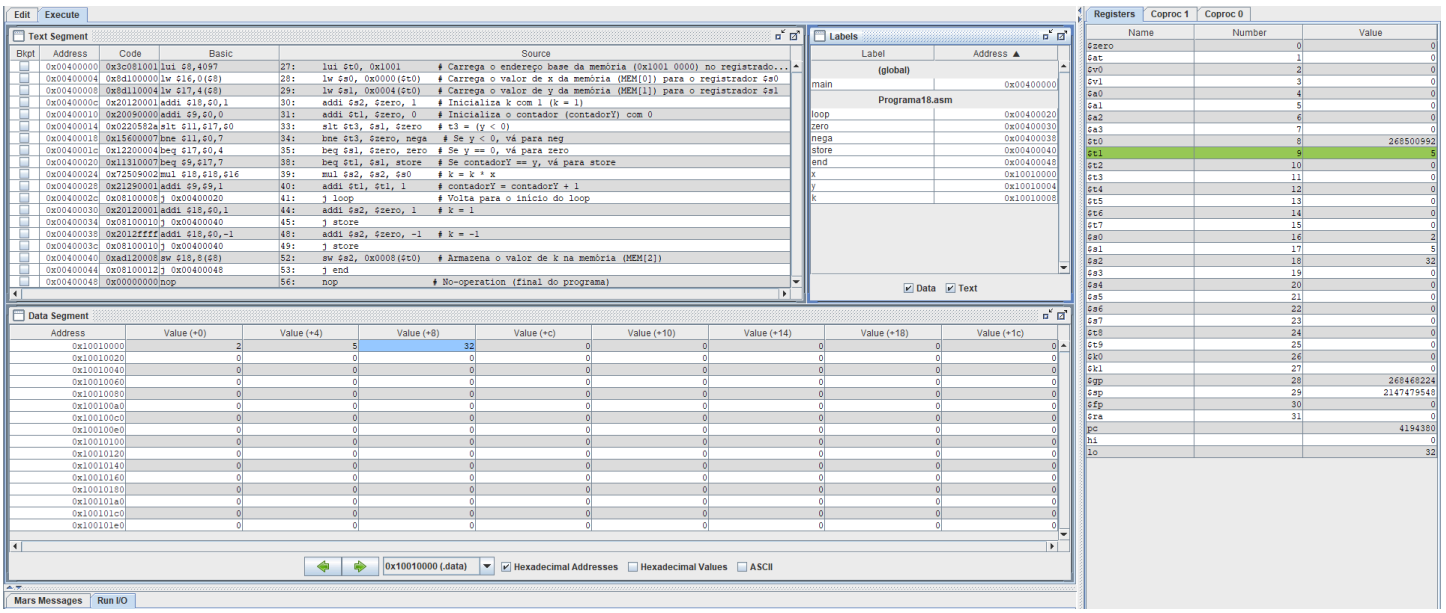
Mars Messages | Run I/O

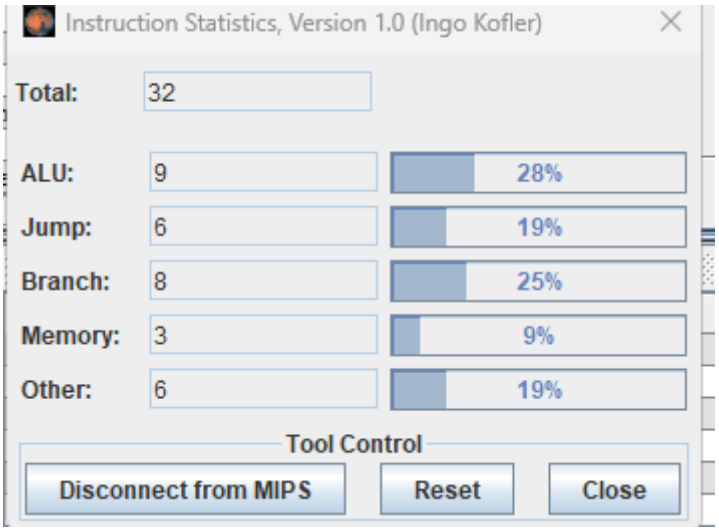
Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	0
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	268500992
$t1	9	25
$t2	10	5
$t3	11	0
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	5
$s1	17	5
$s2	18	25
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548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


Edit
Execute
Text Segment
Bkpt Address Code Basic Source
0x00400000 0x3c081001 lui $8,4097 24: lui $t0, 0x1001 # endereço 0x1001 é onde começa a memoria no mars 0x1001 0000
0x00400004 0x8d100000 lw $16,0($8) 25: lw $s0, 0x0000($t0) # x = MEM[0]
0x00400008 0x8d110004 lw $17,4($8) 26: lw $s1, 0x0004($t0) # y = MEM[1]
0x0040000c 0x01304820 add $9,$9,$16 29: add $t1, $t1, $s0 # t1 = t1 + x
0x00400010 0x214a0001 addi $10,$10,1 30: addi $t2, $t2, 1 # contador++
0x00400014 0x155ffffd bne $10,$17,-3 31: bne $t2, $s1, multip # if (contador != y) goto multip
0x00400018 0x00099020 add $18,$0,$9 32: add $s2, $zero, $t1 # k = x * y
0x0040001c 0xad120008 sw $18,8($8) 33: sw $s2, 0x0008($t0) # MEM[2] = k
Labels
Label Address
(global)
main 0x00400000
Programa17.asm
multip 0x0040000c
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 5 5 25 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
0x100101e0 0 0 0 0 0 0 0 0
0x10010000 (.data)
Hexadecimal Addresses
Hexadecimal Values
ASCII
Mars Messages
Run I/O
Registers
Coproc 1
Coproc 0
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 25
$t2 10 5
$t3 11 0
$t4 12 0
$t5 13 0
$t6 14 0
$t7 15 0
$s0 16 5
$s1 17 5
$s2 18 25
$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:


Edit
Execute
Programa17.asm
Programa18.asm
.text
.globl main
main:
lui $t0, 0x1001 # Carrega o endereço base da memória (0x1001 0000) no registrador $t0
lw $s0, 0x0000($t0) # Carrega o valor de x da memória (MEM[0]) para o registrador $s0
lw $s1, 0x0004($t0) # Carrega o valor de y da memória (MEM[1]) para o registrador $s1
addi $s2, $zero, 1 # Inicializa k com 1 (k = 1)
addi $t1, $zero, 0 # Inicializa o contador (contadorY) com 0
slt $t3, $s1, $zero # t3 = (y < 0)
bne $t3, $zero, nega # Se y < 0, vá para neg
beq $s1, $zero, zero # Se y == 0, vá para zero
loop:
beq $t1, $s1, store # Se contadorY == y, vá para store
mul $s2, $s2, $s0 # k = k * x
addi $t1, $t1, 1 # contadorY = contadorY + 1
j loop # Volta para o inicio do loop
zero:
addi $s2, $zero, 1 # k = 1
j store
nega:
addi $s2, $zero, -1 # k = -1
j store
store:
sw $s2, 0x0008($t0) # Armazena o valor de k na memória (MEM[2])
j end
end:
nop # No-operation (final do programa)
.data
x: .word 2
y: .word 5
k: .word -1
Line: 23 Column: 6
Show Line Numbers
Mars Messages
Run I/O
Registers
Coproc 1
Coproc 0
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 5
$t2 10 0
$t3 11 0
$t4 12 0
$t5 13 0
$t6 14 0
$t7 15 0
$s0 16 2
$s1 17 5
$s2 18 32
$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 4194380
hi 0
lo 32

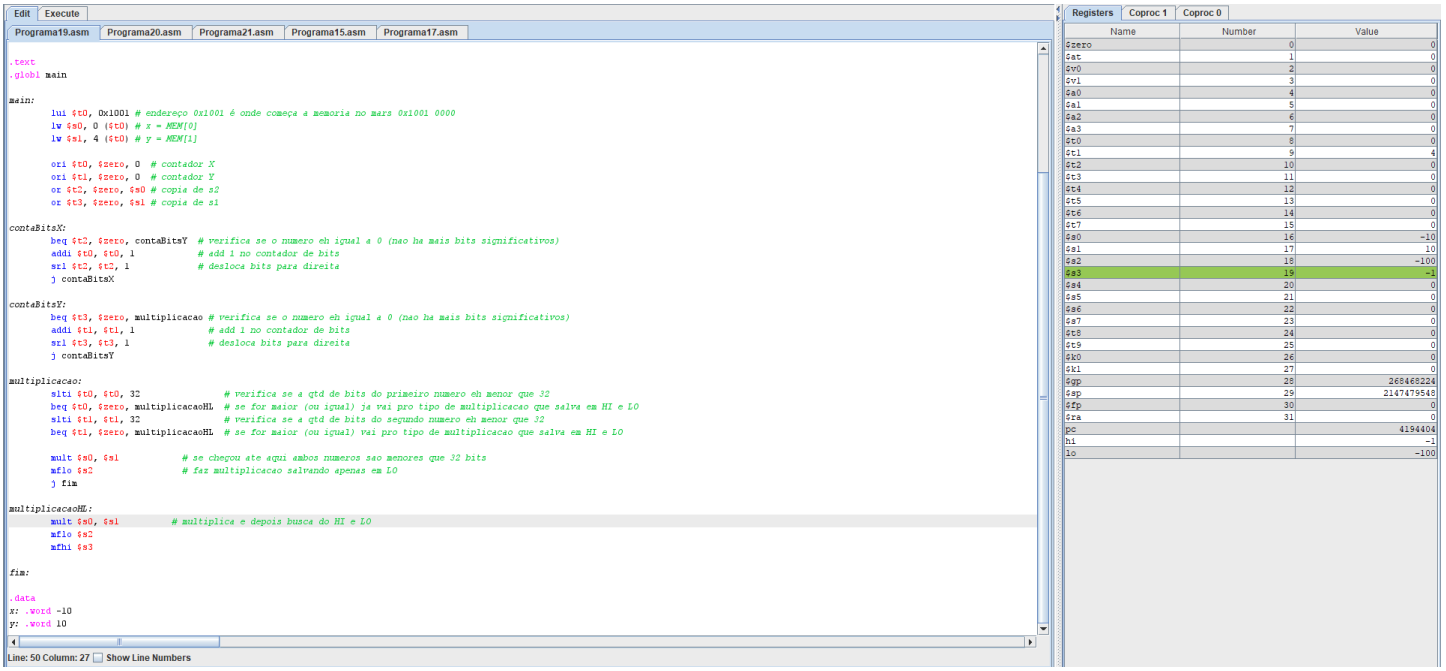

Edit
Execute
Text Segment
Bkpt Address Code Basic Source
0x00400000 0x3c081001 lui $8,4097 27: lui $t0, 0x1001 # Carrega o endereço base da memória (0x1001 0000) no registrado...
0x00400004 0x8d100000 lw $16,0($8) 28: lw $s0, 0x0000($t0) # Carrega o valor de x da memória (MEM[0]) para o registrador $s0
0x00400008 0x8d110004 lw $17,4($8) 29: lw $s1, 0x0004($t0) # Carrega o valor de y da memória (MEM[1]) para o registrador $s1
0x0040000c 0x20120001 addi $18,$0,1 30: addi $s2, $zero, 1 # Inicializa k com 1 (k = 1)
0x00400010 0x20090000 addi $9,$0,0 31: addi $t1, $zero, 0 # Inicializa o contador (contadorY) com 0
0x00400014 0x0220582a slt $11,$17,$0 33: slt $t3, $s1, $zero # t3 = (y < 0)
0x00400018 0x15600007 bne $11,$0,7 34: bne $t3, $zero, nega # Se y < 0, vá para neg
0x0040001c 0x12200004 beq $17,$0,4 35: beq $s1, $zero, zero # Se y == 0, vá para zero
0x00400020 0x11310007 beq $9,$17,7 38: beq $t1, $s1, store # Se contadorY == y, vá para store
0x00400024 0x72509002 mul $18,$18,$16 39: mul $s2, $s2, $s0 # k = k * x
0x00400028 0x21290001 addi $9,$9,1 40: addi $t1, $t1, 1 # contadorY = contadorY + 1
0x0040002c 0x08100008 j 0x00400020 41: j loop # Volta para o inicio do loop
0x00400030 0x20120001 addi $18,$0,1 44: addi $s2, $zero, 1 # k = 1
0x00400034 0x08100010 j 0x00400040 45: j store
0x00400038 0x2012ffff addi $18,$0,-1 48: addi $s2, $zero, -1 # k = -1
0x0040003c 0x08100010 j 0x00400040 49: j store
0x00400040 0xad120008 sw $18,8($8) 52: sw $s2, 0x0008($t0) # Armazena o valor de k na memória (MEM[2])
0x00400044 0x08100012 j 0x00400048 53: j end
0x00400048 0x00000000 nop 56: nop # No-operation (final do programa)
Labels
Label Address
(global)
main 0x00400000
Programa18.asm
loop 0x00400020
zero 0x00400030
nega 0x00400038
store 0x00400040
end 0x00400048
x 0x10010000
y 0x10010004
k 0x10010008
Data Text
Data Segment
Address Value (+0) Value (+4) Value (+8) Value (+c) Value (+10) Value (+14) Value (+18) Value (+1c)
0x10010000 2 5 32 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
0x100101e0 0 0 0 0 0 0 0 0
0x10010000 (.data)
Hexadecimal Addresses
Hexadecimal Values
ASCII
Mars Messages
Run I/O
Registers
Coproc 1
Coproc 0
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 5
$t2 10 0
$t3 11 0
$t4 12 0
$t5 13 0
$t6 14 0
$t7 15 0
$s0 16 2
$s1 17 5
$s2 18 32
$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 4194380
hi 0
lo 32



Para os programas a seguir use instruções mult, div, mflo e mfhi. Responda

1) C
2) B
3) A
4) C
5) B
6) A
7) D
8) A
9) A
10) A

PROGRAMA 19:



Edit | Execute

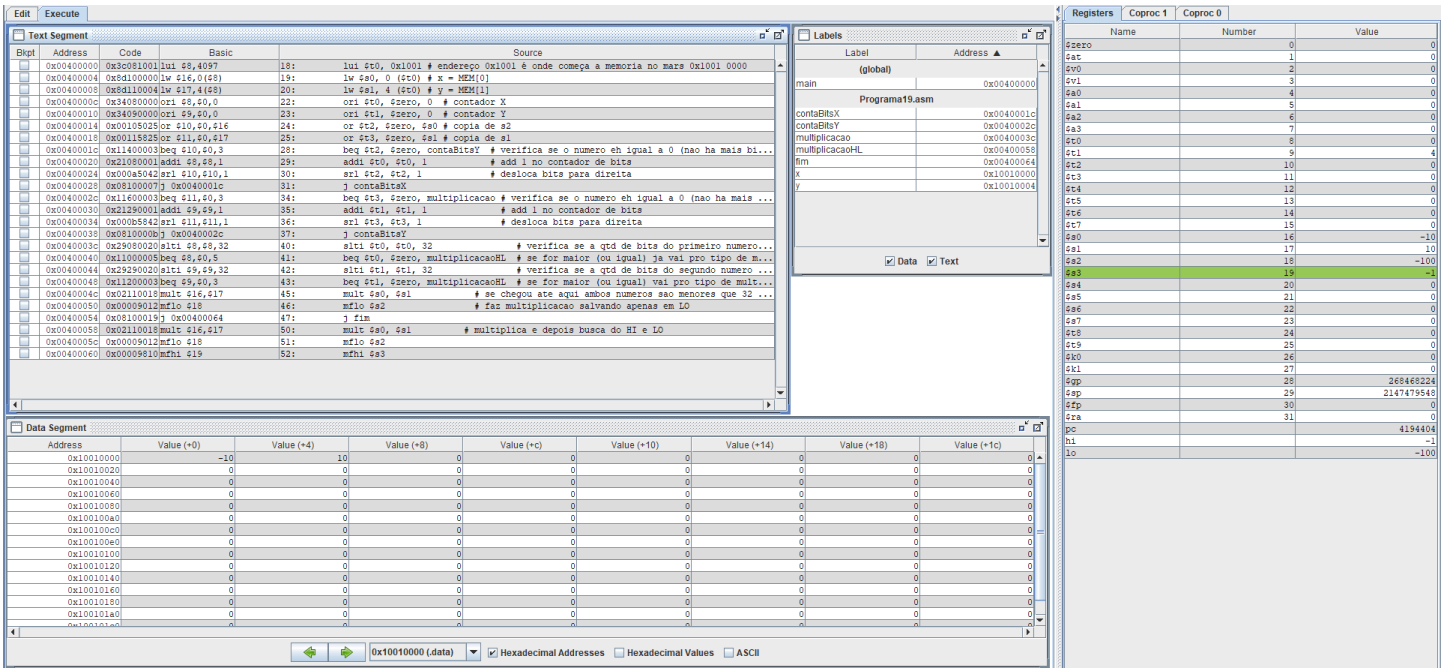
Programa19.asm | Programa20.asm | Programa21.asm | Programa15.asm | Programa17.asm

```
.text
.globl main

main:
        lui $t0, 0x1001 # endereço 0x1001 é onde começa a memoria no mars 0x1001 0000
        lw $s0, 0 ($t0) # x = MEM[0]
        lw $s1, 4 ($t0) # y = MEM[1]

        ori $t0, $zero, 0  # contador X
        ori $t1, $zero, 0  # contador Y
        or $t2, $zero, $s0 # copia de s2
        or $t3, $zero, $s1 # copia de s1

contaBitsX:
        beq $t2, $zero, contaBitsY  # verifica se o numero eh igual a 0 (nao ha mais bits significativos)
        addi $t0, $t0, 1            # add 1 no contador de bits
        srl $t2, $t2, 1             # desloca bits para direita
        j contaBitsX

contaBitsY:
        beq $t3, $zero, multiplicacao # verifica se o numero eh igual a 0 (nao ha mais bits significativos)
        addi $t1, $t1, 1              # add 1 no contador de bits
        srl $t3, $t3, 1               # desloca bits para direita
        j contaBitsY

multiplicacao:
        slti $t0, $t0, 32               # verifica se a qtd de bits do primeiro numero eh menor que 32
        beq $t0, $zero, multiplicacaoHL  # se for maior (ou igual) ja vai pro tipo de multiplicacao que salva em HI e LO
        slti $t1, $t1, 32               # verifica se a qtd de bits do segundo numero eh menor que 32
        beq $t1, $zero, multiplicacaoHL  # se for maior (ou igual) vai pro tipo de multiplicacao que salva em HI e LO

        mult $s0, $s1           # se chegou ate aqui ambos numeros sao menores que 32 bits
        mflo $s2                # faz multiplicacao salvando apenas em LO
        j fim

multiplicacaoHL:
        mult $s0, $s1          # multiplica e depois busca do HI e LO
        mflo $s2
        mfhi $s3

fim:

.data
x: .word -10
y: .word 10
```

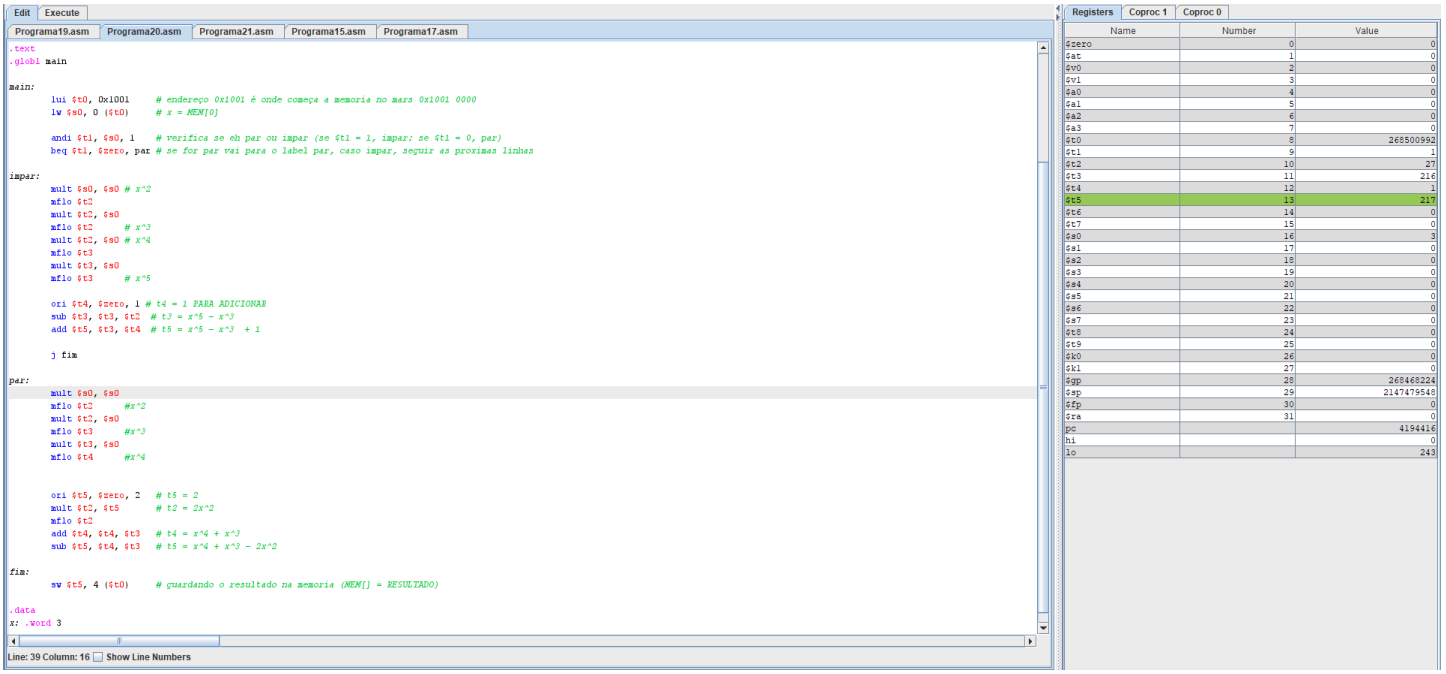
Line: 50 Column: 27 | Show Line Numbers

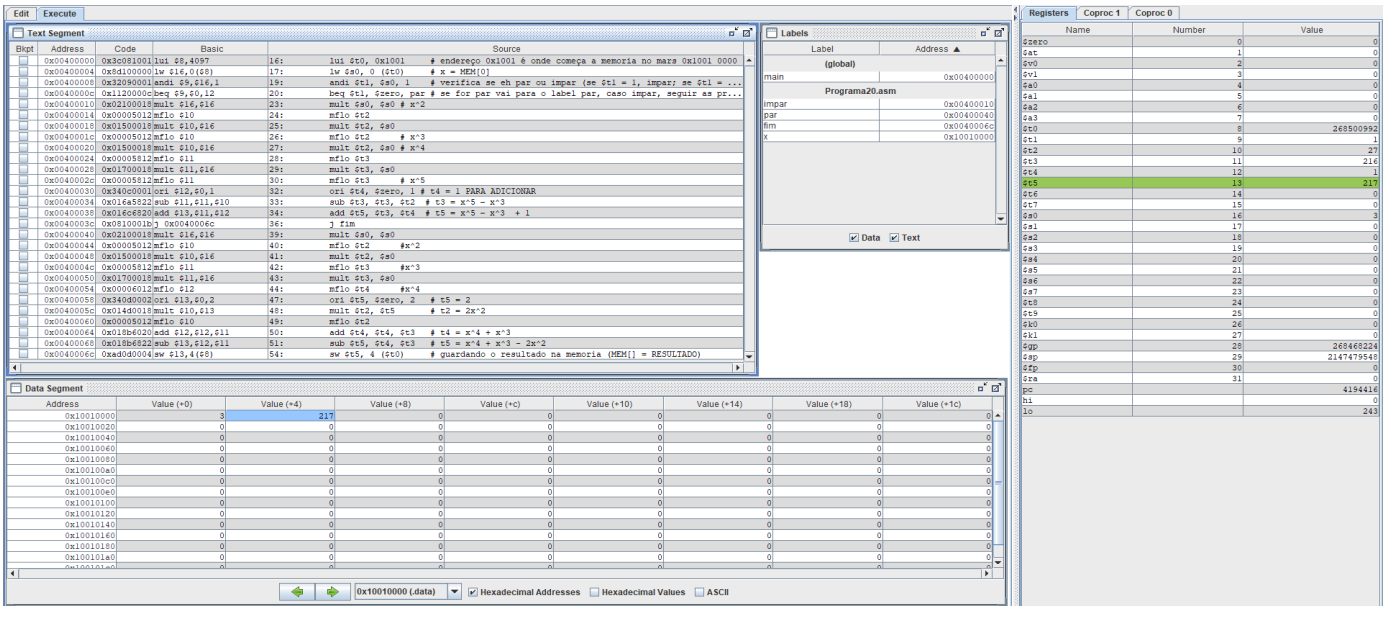
Registers | Coproc 1 | Coproc 0

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	0
$t1	9	4
$t2	10	0
$t3	11	0
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	-10
$s1	17	10
$s2	18	-100
$s3	19	-1
$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		4194404
hi		-1
lo		-100

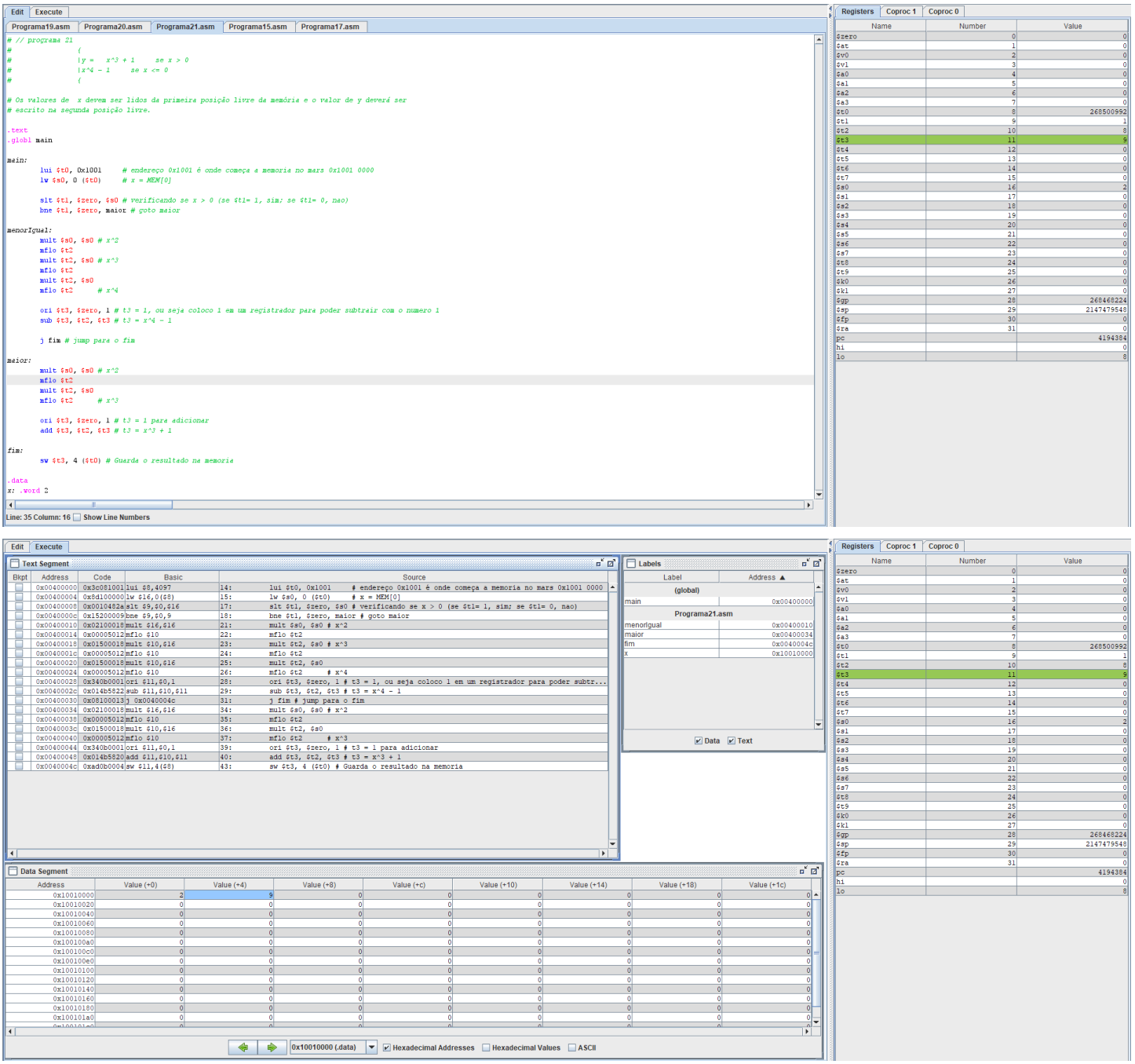

Edit
Execute
Text Segment
Bkpt Address Code Basic Source
0x00400000 0x3c081001 lui $8,4097 18: lui $t0, 0x1001 # endereço 0x1001 é onde começa a memoria no mars 0x1001 0000
0x00400004 0x8d1d0000 lw $16,0($8) 19: lw $s0, 0 ($t0) # x = MEM[0]
0x00400008 0x8d110004 lw $17,4($8) 20: lw $s1, 4 ($t0) # y = MEM[1]
0x0040000c 0x34080000 ori $8,$0,0 22: ori $t0, $zero, 0 # contador X
0x00400010 0x34090000 ori $9,$0,0 23: ori $t1, $zero, 0 # contador Y
0x00400014 0x00105025 or $10,$0,$16 24: or $t2, $zero, $s0 # copia de s2
0x00400018 0x00115825 or $11,$0,$17 25: or $t3, $zero, $s1 # copia de s1
0x0040001c 0x11400003 beq $10,$0,3 28: beq $t2, $zero, contaBitsY # verifica se o numero eh igual a 0 (nao ha mais bi...
0x00400020 0x21080001 addi $8,$8,1 29: addi $t0, $t0, 1 # add 1 no contador de bits
0x00400024 0x000a5042 srl $10,$10,1 30: srl $t2, $t2, 1 # desloca bits para direita
0x00400028 0x08100007 j 0x0040001c 31: j contaBitsX
0x0040002c 0x11600003 beq $11,$0,3 34: beq $t3, $zero, multiplicacao # verifica se o numero eh igual a 0 (nao ha mais ...
0x00400030 0x21290001 addi $9,$9,1 35: addi $t1, $t1, 1 # add 1 no contador de bits
0x00400034 0x000b5842 srl $11,$11,1 36: srl $t3, $t3, 1 # desloca bits para direita
0x00400038 0x0810000b j 0x0040002c 37: j contaBitsY
0x0040003c 0x29080020 slti $8,$8,32 40: slti $t0, $t0, 32 # verifica se a qtd de bits do primeiro numero...
0x00400040 0x11000005 beq $8,$0,5 41: beq $t0, $zero, multiplicacaoHL # se for maior (ou igual) ja vai pro tipo de m...
0x00400044 0x29290020 slti $9,$9,32 42: slti $t1, $t1, 32 # verifica se a qtd de bits do segundo numero ...
0x00400048 0x11200003 beq $9,$0,3 43: beq $t1, $zero, multiplicacaoHL # se for maior (ou igual) vai pro tipo de mult...
0x0040004c 0x02110018 mult $16,$17 45: mult $s0, $s1 # se chegou ate aqui ambos numeros sao menores que 32 ...
0x00400050 0x00009012 mflo $18 46: mflo $s2 # faz multiplicacao salvando apenas em LO
0x00400054 0x08100019 j 0x00400064 47: j fim
0x00400058 0x02110018 mult $16,$17 50: mult $s0, $s1 # multiplica e depois busca do HI e LO
0x0040005c 0x00009012 mflo $18 51: mflo $s2
0x00400060 0x00009810 mfhi $19 52: mfhi $s3
Labels
Label Address
(global)
main 0x00400000
Programa19.asm
contaBitsX 0x0040001c
contaBitsY 0x0040002c
multiplicacao 0x0040003c
multiplicacaoHL 0x00400058
fim 0x00400064
x 0x10010000
y 0x10010004
Data Text
Registers Coproc 1 Coproc 0
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 0
$t1 9 4
$t2 10 0
$t3 11 0
$t4 12 0
$t5 13 0
$t6 14 0
$t7 15 0
$s0 16 -10
$s1 17 10
$s2 18 -100
$s3 19 -1
$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 4194404
hi -1
lo -100
Data Segment
Address Value (+0) Value (+4) Value (+8) Value (+c) Value (+10) Value (+14) Value (+18) Value (+1c)
0x10010000 -10 10 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

PROGRAMA 20:


Edit
Execute
Programa19.asm Programa20.asm Programa21.asm Programa15.asm Programa17.asm
.text
.globl main
main:
lui $t0, 0x1001 # endereço 0x1001 é onde começa a memoria no mars 0x1001 0000
lw $s0, 0 ($t0) # x = MEM[0]
andi $t1, $s0, 1 # verifica se eh par ou impar (se $t1 = 1, impar; se $t1 = 0, par)
beq $t1, $zero, par # se for par vai para o label par, caso impar, seguir as proximas linhas
impar:
mult $s0, $s0 # x^2
mflo $t2
mult $t2, $s0
mflo $t2 # x^3
mult $t2, $s0 # x^4
mflo $t3
mult $t3, $s0
mflo $t3 # x^5
ori $t4, $zero, 1 # t4 = 1 PARA ADICIONAR
sub $t3, $t3, $t2 # t3 = x^5 - x^3
add $t5, $t3, $t4 # t5 = x^5 - x^3 + 1
j fim
par:
mult $s0, $s0
mflo $t2 #x^2
mult $t2, $s0
mflo $t3 #x^3
mult $t3, $s0
mflo $t4 #x^4
ori $t5, $zero, 2 # t5 = 2
mult $t2, $t5 # t2 = 2x^2
mflo $t2
add $t4, $t4, $t3 # t4 = x^4 + x^3
sub $t5, $t4, $t3 # t5 = x^4 + x^3 - 2x^2
fim:
sw $t5, 4 ($t0) # guardando o resultado na memoria (MEM[] = RESULTADO)
.data
x: .word 3
Line: 39 Column: 16 Show Line Numbers
Registers Coproc 1 Coproc 0
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 1
$t2 10 27
$t3 11 216
$t4 12 1
$t5 13 217
$t6 14 0
$t7 15 0
$s0 16 3
$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 4194416
hi 0
lo 243


Edit
Execute
Text Segment
Bkpt Address Code Basic Source
0x00400000 0x3c081001 lui $8,4097 16: lui $t0, 0x1001 # endereço 0x1001 é onde começa a memoria no mars 0x1001 0000
0x00400004 0x8d100000 lw $16,0($8) 17: lw $s0, 0 ($t0) # x = MEM[0]
0x00400008 0x32090001 andi $9,$16,1 19: andi $t1, $s0, 1 # verifica se eh par ou impar (se $t1 = 1, impar; se $t1 = ...
0x0040000c 0x1120000e beq $9,$0,12 20: beq $t1, $zero, par # se for par vai para o label par, caso impar, seguir as pr...
0x00400010 0x02100018 mult $16,$16 23: mult $s0, $s0 # x^2
0x00400014 0x00005012 mflo $10 24: mflo $t2
0x00400018 0x01500018 mult $10,$16 25: mult $t2, $s0
0x0040001c 0x00005012 mflo $10 26: mflo $t2 # x^3
0x00400020 0x01500018 mult $10,$16 27: mult $t2, $s0 # x^4
0x00400024 0x00005812 mflo $11 28: mflo $t3
0x00400028 0x01700018 mult $11,$16 29: mult $t3, $s0
0x0040002c 0x00005812 mflo $11 30: mflo $t3 # x^5
0x00400030 0x340c0001 ori $12,$0,1 32: ori $t4, $zero, 1 # t4 = 1 PARA ADICIONAR
0x00400034 0x016a5822 sub $11,$11,$10 33: sub $t3, $t3, $t2 # t3 = x^5 - x^3
0x00400038 0x016c6820 add $13,$11,$12 34: add $t5, $t3, $t4 # t5 = x^5 - x^3 + 1
0x0040003c 0x0810001b j 0x0040006c 36: j fim
0x00400040 0x02100018 mult $16,$16 39: mult $s0, $s0
0x00400044 0x00005012 mflo $10 40: mflo $t2 #x^2
0x00400048 0x01500018 mult $10,$16 41: mult $t2, $s0
0x0040004c 0x00005812 mflo $11 42: mflo $t3 #x^3
0x00400050 0x01700018 mult $11,$16 43: mult $t3, $s0
0x00400054 0x00006012 mflo $12 44: mflo $t4 #x^4
0x00400058 0x34ddd002 ori $13,$13,2 47: ori $t5, $zero, 2 # t5 = 2
0x0040005c 0x014d0018 mult $10,$13 48: mult $t2, $t5 # t2 = 2x^2
0x00400060 0x00005012 mflo $10 49: mflo $t2
0x00400064 0x018b6020 add $12,$12,$11 50: add $t4, $t4, $t3 # t4 = x^4 + x^3
0x00400068 0x018b6822 sub $13,$12,$11 51: sub $t5, $t4, $t3 # t5 = x^4 + x^3 - 2x^2
0x0040006c 0xadd00004 sw $13,4($8) 54: sw $t5, 4 ($t0) # guardando o resultado na memoria (MEM[] = RESULTADO)
Labels
Label Address
(global)
main 0x00400000
Programa20.asm
impar 0x00400010
par 0x00400040
fim 0x0040006c
x 0x10010000
Data Text
Registers Coproc 1 Coproc 0
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 1
$t2 10 27
$t3 11 216
$t4 12 1
$t5 13 217
$t6 14 0
$t7 15 0
$s0 16 3
$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 4194416
hi 0
lo 243
Data Segment
Address Value (+0) Value (+4) Value (+8) Value (+c) Value (+10) Value (+14) Value (+18) Value (+1c)
0x10010000 3 217 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

PROGRAMA 21:



```
# // programa 21
#               (
#               |y =   x^3 + 1     se x > 0
#               |x^4 - 1     se x <= 0
#               (

# 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
.globl main

main:
        lui $t0, 0x1001      # endereço 0x1001 é onde começa a memoria no mars 0x1001 0000
        lw $s0, 0 ($t0)      # x = MEM[0]

        slt $t1, $zero, $s0 # verificando se x > 0 (se $t1= 1, sim; se $t1= 0, nao)
        bne $t1, $zero, maior # goto maior

menorIgual:
        mult $s0, $s0 # x^2
        mflo $t2
        mult $t2, $s0 # x^3
        mflo $t2
        mult $t2, $s0
        mflo $t2      # x^4

        ori $t3, $zero, 1 # t3 = 1, ou seja coloco 1 em um registrador para poder subtrair com o numero 1
        sub $t3, $t2, $t3 # t3 = x^4 - 1

        j fim # jump para o fim

maior:
        mult $s0, $s0 # x^2
        mflo $t2
        mult $t2, $s0
        mflo $t2      # x^3

        ori $t3, $zero, 1 # t3 = 1 para adicionar
        add $t3, $t2, $t3 # t3 = x^3 + 1

fim:
        sw $t3, 4 ($t0) # Guarda o resultado na memoria

.data
x: .word 2
```

Line: 35 Column: 16 Show Line Numbers

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	1
$t2	10	8
$t3	11	9
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	2
$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		4194384
hi		0
lo		8

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x3c081001	lui $8,4097	14: lui $t0, 0x1001 # endereço 0x1001 é onde começa a memoria no mars 0x1001 0000
	0x00400004	0x8d100000	lw $16,0($8)	15: lw $s0, 0 ($t0) # x = MEM[0]
	0x00400008	0x0010482a	slt $9,$0,$16	17: slt $t1, $zero, $s0 # verificando se x > 0 (se $t1= 1, sim; se $t1= 0, nao)
	0x0040000c	0x15200009	bne $9,$0,9	18: bne $t1, $zero, maior # goto maior
	0x00400010	0x02100018	mult $16,$16	21: mult $s0, $s0 # x^2
	0x00400014	0x00005012	mflo $10	22: mflo $t2
	0x00400018	0x01500018	mult $10,$16	23: mult $t2, $s0 # x^3
	0x0040001c	0x00005012	mflo $10	24: mflo $t2
	0x00400020	0x01500018	mult $10,$16	25: mult $t2, $s0
	0x00400024	0x00005012	mflo $10	26: mflo $t2 # x^4
	0x00400028	0x340b0001	ori $11,$0,1	28: ori $t3, $zero, 1 # t3 = 1, ou seja coloco 1 em um registrador para poder subtr...
	0x0040002c	0x014b5822	sub $11,$10,$11	29: sub $t3, $t2, $t3 # t3 = x^4 - 1
	0x00400030	0x08100013	j 0x0040004c	31: j fim # jump para o fim
	0x00400034	0x02100018	mult $16,$16	34: mult $s0, $s0 # x^2
	0x00400038	0x00005012	mflo $10	35: mflo $t2
	0x0040003c	0x01500018	mult $10,$16	36: mult $t2, $s0
	0x00400040	0x00005012	mflo $10	37: mflo $t2 # x^3
	0x00400044	0x340b0001	ori $11,$0,1	39: ori $t3, $zero, 1 # t3 = 1 para adicionar
	0x00400048	0x014b5820	add $11,$10,$11	40: add $t3, $t2, $t3 # t3 = x^3 + 1
	0x0040004c	0xad0b0004	sw $11,4($8)	43: sw $t3, 4 ($t0) # Guarda o resultado na memoria

Labels

Label	Address
(global)	
main	0x00400000
Programa21.asm	
menorIgual	0x00400010
maior	0x00400034
fim	0x0040004c
x	0x10010000

Data Text

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	2	9	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	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	1
$t2	10	8
$t3	11	9
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	2
$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		4194384
hi		0
lo		8