PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

GABRIEL VARGAS BENTO DE SOUZA

EXERCÍCIO PRÁTICO 05

Arquitetura de Computadores II

Belo Horizonte - MG
2023

Parte 1 - Responda

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

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

3. Qual o caracter que, na linguagem assembly do SPIM, inicia um comentário?
A. #
B. $
C. / /
D. *

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

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

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

7. O que é uma diretiva, tal como a diretiva .text?
A. uma instrução em linguagem assembly que resulta em uma instrução em linguagem de máquina.
B. uma das opções de menu do sistema SPIM.
C. uma instrução em linguagem de máquina que faz com que uma operação sobre os dados ocorra.
D. uma declaração que diz o montador algo sobre o que o programador quer, mas não corresponde diretamente a uma instrução de máquina.

8. O que é um endereço simbólico?
A. um local de memória que contém dados simbólicos.
B. um byte na memória que contém o endereço de dados.
C. símbolo dado como argumento para uma directiva.
D. um nome usado no código-fonte em linguagem assembly para um local na memória.

9. Em qual endereço o simulador SPIM coloca a primeira instrução de máquina quando ele está sendo executado?
A. 0x00000000
B. 0x00400000
C. 0x10000000
D. 0xFFFFFFFF

10. Algumas instruções de máquina possuem uma constante como um dos operandos. Como é chamado tal operando?
A. operando imediato
B. operando embutido
C. operando binário
D. operando de máquina

11. Como é chamada uma operação lógica executada entre bits de cada coluna dos operandos para produzir um bit de resultado para cada coluna?
A. operação lógica
B. operação bitwise
C. operação binária
D. operação coluna

12. Quando uma operação é de fato executada, como estão os operandos na ALU?
A. Pelo menos um operando deve ser de 32 bit.
B. Cada operando pode ser de qualquer tamanho.
C. Ambos operandos devem que vir de registros.
D. Cada um dos registradores deve possuir 32 bit.

13. Dezesseis bits de dados de uma instrução de ori são usados como um operando imediato. Durante a execução, o que deve ser feito primeiro?
A. Os dados são estendidos em zero à direita por 16 bits.
B. Os dados são estendidos em zero à esquerda por 16 bits.
C. Nada precisa ser feito.
D. Apenas 16 bits são usados pelo outro operando.

14. Qual das instruções seguintes armazenam no registrador $5 um padrão de bits que representa positivo 48?
A. ori $5,$0,0x48
B. ori $5,$5,0x48
C. ori $5,$0,48
D. ori $0,$5,0x48

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

16. Qual das instruções seguintes limpa todos os bits no registrador $8 com exceção do byte de baixa ordem que fica inalterado?
A. ori $8,$8,0xFF
B. ori $8,$0,0x00FF
C. xori $8,$8,0xFF
D. andi $8,$8,0xFF

17. Qual é o resultado de um ou exclusivo de padrão sobre ele mesmo?

A. Todos os bits em zero.

B. Todos os bits em um.

C. O padrão original utilizado.

D. O resultado é o contrário do original.

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

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

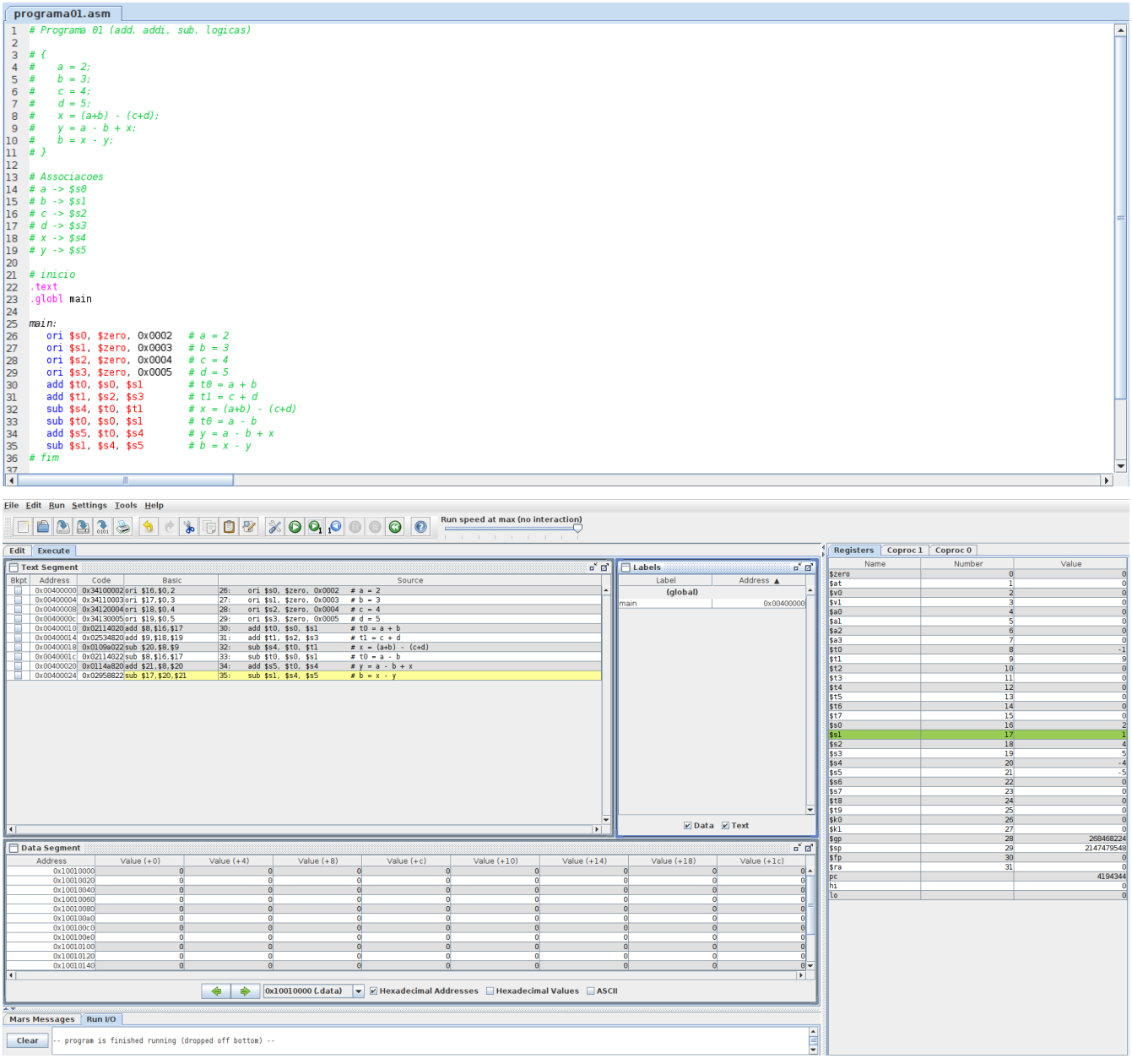
B. Não. Cada instrução de máquina é completamente diferente de qualquer outra.

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

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

Parte 2 - Implementar em MIPS/MARS os seguintes programas (usando apenas as instruções indicadas)

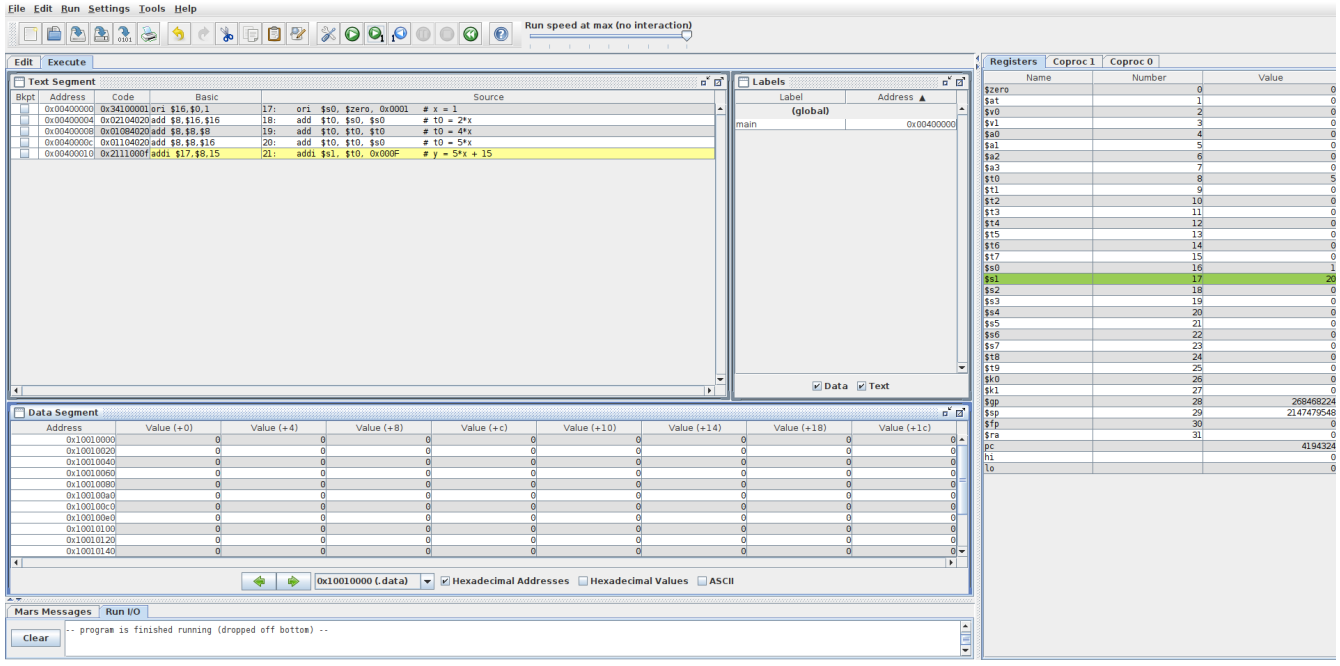
Programa 1:



Programa 2:

programa01.asm | programa02.asm

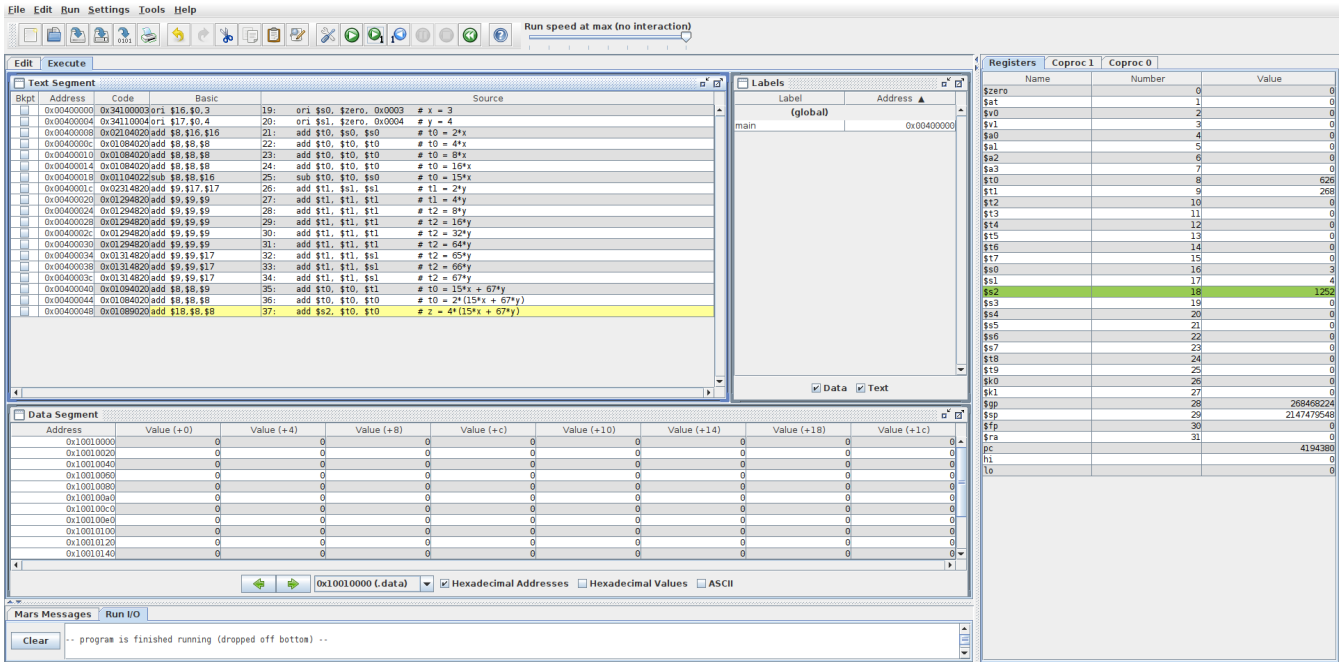
```
1  # Programa 02 (add, addi, sub, logicas)
2
3  # {
4  #    x = 1;
5  #    y = 5*x + 15;
6  # }
7
8  # Associacoes
9  # x -> $s0
10 # y -> $s1
11
12 # inicio
13 .text
14 .globl main
15
16 main:
17    ori  $s0, $zero, 0x0001   # x = 1
18    add  $t0, $s0, $s0        # t0 = 2*x
19    add  $t0, $t0, $t0        # t0 = 4*x
20    add  $t0, $t0, $s0        # t0 = 5*x
21    addi $s1, $t0, 0x000F     # y = 5*x + 15
22 # fim
23
24
25
```



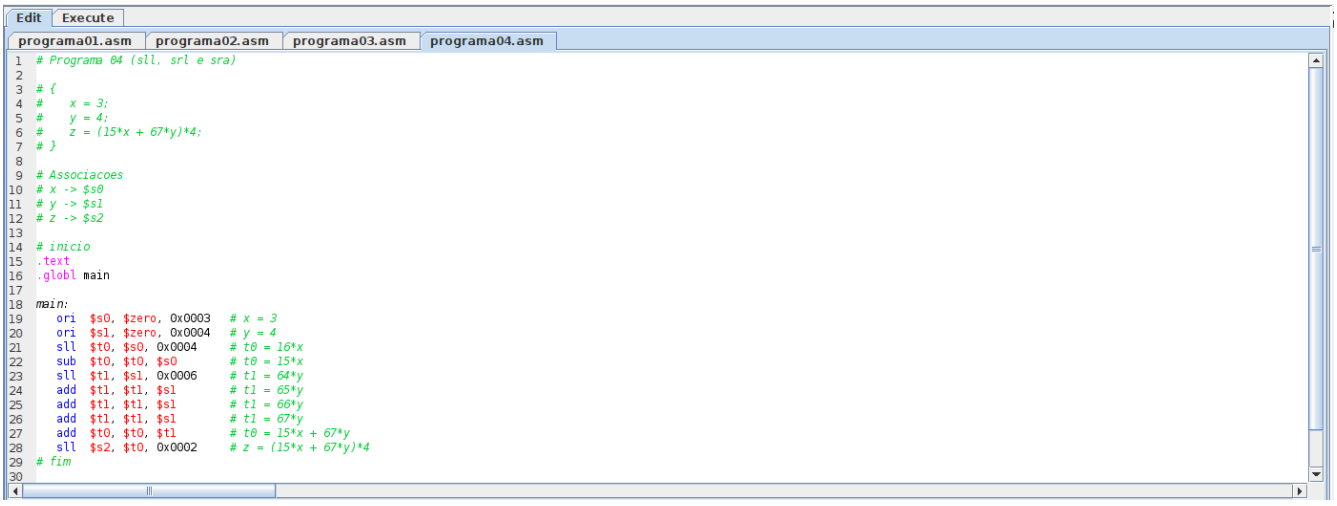
Programa 3:

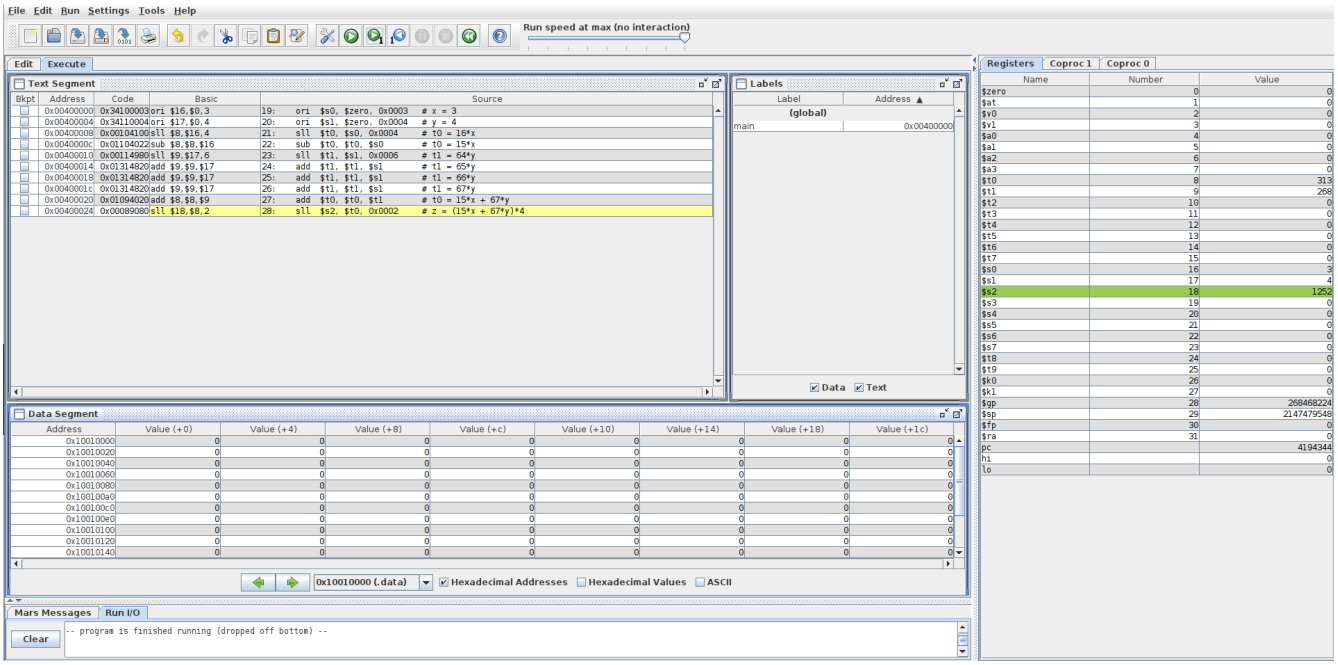
programa01.asm | programa02.asm | programa03.asm

```
1  # Programa 03 (add, addi, sub, logicas)
2
3  # {
4  #    x = 3;
5  #    y = 4;
6  #    z = (15*x + 67*y)*4;
7  # }
8
9  # Associacoes
10 # x -> $s0
11 # y -> $s1
12 # z -> $s2
13
14 # inicio
15 .text
16 .globl main
17
18 main:
19    ori $s0, $zero, 0x0003   # x = 3
20    ori $s1, $zero, 0x0004   # y = 4
21    add $t0, $s0, $s0        # t0 = 2*x
22    add $t0, $t0, $t0        # t0 = 4*x
23    add $t0, $t0, $t0        # t0 = 8*x
24    add $t0, $t0, $t0        # t0 = 16*x
25    sub $t0, $t0, $s0        # t0 = 15*x
26    add $t1, $s1, $s1        # t1 = 2*y
27    add $t1, $t1, $t1        # t1 = 4*y
28    add $t1, $t1, $t1        # t2 = 8*y
29    add $t1, $t1, $t1        # t2 = 16*y
30    add $t1, $t1, $t1        # t2 = 32*y
31    add $t1, $t1, $t1        # t2 = 64*y
32    add $t1, $t1, $s1        # t2 = 65*y
33    add $t1, $t1, $s1        # t2 = 66*y
34    add $t1, $t1, $s1        # t2 = 67*y
35    add $t0, $t0, $t1        # t0 = 15*x + 67*y
36    add $t0, $t0, $t0        # t0 = 2*(15*x + 67*y)
37    add $s2, $t0, $t0        # z = 4*(15*x + 67*y)
38 # fim
39
```


File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Text Segment
Bkpt Address Code Basic Source
0x00400000 0x34100003 ori $16,$0,3 19: ori $s0, $zero, 0x0003 # x = 3
0x00400004 0x34110004 ori $17,$0,4 20: ori $s1, $zero, 0x0004 # y = 4
0x00400008 0x02104020 add $8,$16,$16 21: add $t0, $s0, $s0 # t0 = 2*x
0x0040000c 0x01084020 add $8,$8,$8 22: add $t0, $t0, $t0 # t0 = 4*x
0x00400010 0x01084020 add $8,$8,$8 23: add $t0, $t0, $t0 # t0 = 8*x
0x00400014 0x01084020 add $8,$8,$8 24: add $t0, $t0, $t0 # t0 = 16*x
0x00400018 0x01104022 sub $8,$8,$16 25: sub $t0, $t0, $s0 # t0 = 15*x
0x0040001c 0x02314820 add $9,$17,$17 26: add $t1, $s1, $s1 # t1 = 2*y
0x00400020 0x01294820 add $9,$9,$9 27: add $t1, $t1, $t1 # t1 = 4*y
0x00400024 0x01294820 add $9,$9,$9 28: add $t1, $t1, $t1 # t1 = 8*y
0x00400028 0x01294820 add $9,$9,$9 29: add $t1, $t1, $t1 # t2 = 16*y
0x0040002c 0x01294820 add $9,$9,$9 30: add $t1, $t1, $t1 # t2 = 32*y
0x00400030 0x01294820 add $9,$9,$9 31: add $t1, $t1, $t1 # t2 = 64*y
0x00400034 0x01314820 add $9,$9,$17 32: add $t1, $t1, $s1 # t2 = 65*y
0x00400038 0x01314820 add $9,$9,$17 33: add $t1, $t1, $s1 # t2 = 66*y
0x0040003c 0x01314820 add $9,$9,$17 34: add $t1, $t1, $s1 # t2 = 67*y
0x00400040 0x01094020 add $8,$8,$9 35: add $t0, $t0, $t1 # t0 = 15*x + 67*y
0x00400044 0x01084020 add $8,$8,$8 36: add $t0, $t0, $t0 # t0 = 2*(15*x + 67*y)
0x00400048 0x01089020 add $18,$8,$8 37: add $s2, $t0, $t0 # z = 4*(15*x + 67*y)
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 8 626
$t1 9 268
$t2 10 0
$t3 11 0
$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 4194380
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
0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII
Mars Messages Run I/O
Clear
-- program is finished running (dropped off bottom) --

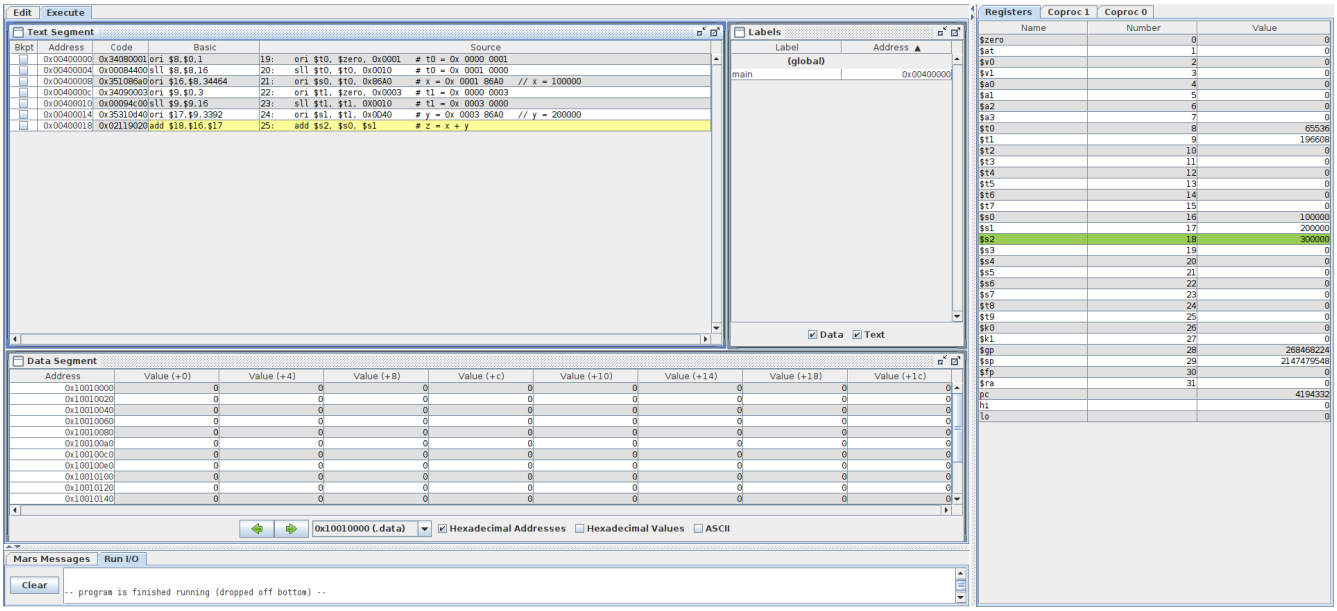
Programa 4:


Edit Execute
programa01.asm programa02.asm programa03.asm programa04.asm
Programa 04 (sll, srl e sra)
{
x = 3;
y = 4;
z = (15*x + 67*y)*4;
}
Associacoes
x -> $s0
y -> $s1
z -> $s2
inicio
.text
.globl main
main:
ori $s0, $zero, 0x0003 # x = 3
ori $s1, $zero, 0x0004 # y = 4
sll $t0, $s0, 0x0004 # t0 = 16*x
sub $t0, $t0, $s0 # t0 = 15*x
sll $t1, $s1, 0x0006 # t1 = 64*y
add $t1, $t1, $s1 # t1 = 65*y
add $t1, $t1, $s1 # t1 = 66*y
add $t1, $t1, $s1 # t1 = 67*y
add $t0, $t0, $t1 # t0 = 15*x + 67*y
sll $s2, $t0, 0x0002 # z = (15*x + 67*y)*4
fim


File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Text Segment
Bkpt Address Code Basic Source
0x00400000 0x34100003 ori $16,$0,3 19: ori $s0, $zero, 0x0003 # x = 3
0x00400004 0x34110004 ori $17,$0,4 20: ori $s1, $zero, 0x0004 # y = 4
0x00400008 0x00104100 sll $8,$16,4 21: sll $t0, $s0, 0x0004 # t0 = 16*x
0x0040000c 0x01104022 sub $8,$8,$16 22: sub $t0, $t0, $s0 # t0 = 15*x
0x00400010 0x00114980 sll $9,$17,6 23: sll $t1, $s1, 0x0006 # t1 = 64*y
0x00400014 0x01314820 add $9,$9,$17 24: add $t1, $t1, $s1 # t1 = 65*y
0x00400018 0x01314820 add $9,$9,$17 25: add $t1, $t1, $s1 # t1 = 66*y
0x0040001c 0x01314820 add $9,$9,$17 26: add $t1, $t1, $s1 # t1 = 67*y
0x00400020 0x01094020 add $8,$8,$9 27: add $t0, $t0, $t1 # t0 = 15*x + 67*y
0x00400024 0x00089080 sll $18,$8,2 28: sll $s2, $t0, 0x0002 # z = (15*x + 67*y)*4
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 8 313
$t1 9 268
$t2 10 0
$t3 11 0
$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
0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII
Mars Messages Run I/O
Clear
-- program is finished running (dropped off bottom) --

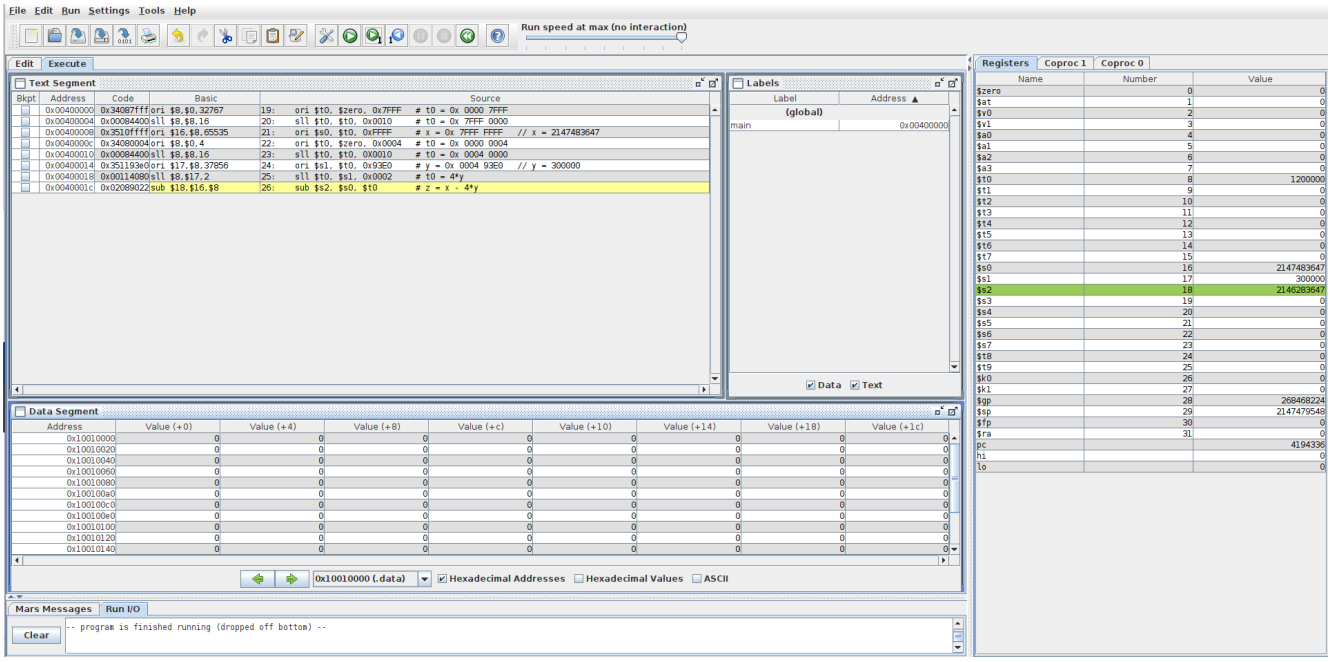
Programa 5:

```
# Programa 05 (sll, srl e sra)

# {
#    x = 100000;    // 0x 0001 86A0
#    y = 200000;    // 0x 0003 0D40
#    z = x + y;
# }

# Associacoes
# x -> $s0
# y -> $s1
# z -> $s2

# inicio
.text
.globl main

main:
    ori $t0, $zero, 0x0001   # t0 = 0x 0000 0001
    sll $t0, $t0, 0x0010     # t0 = 0x 0001 0000
    ori $s0, $t0, 0x86A0     # x = 0x 0001 86A0   // x = 100000
    ori $t1, $zero, 0x0003   # t1 = 0x 0000 0003
    sll $t1, $t1, 0X0010     # t1 = 0x 0003 0000
    ori $s1, $t1, 0x0D40     # y = 0x 0003 86A0   // y = 200000
    add $s2, $s0, $s1        # z = x + y
# fim
```

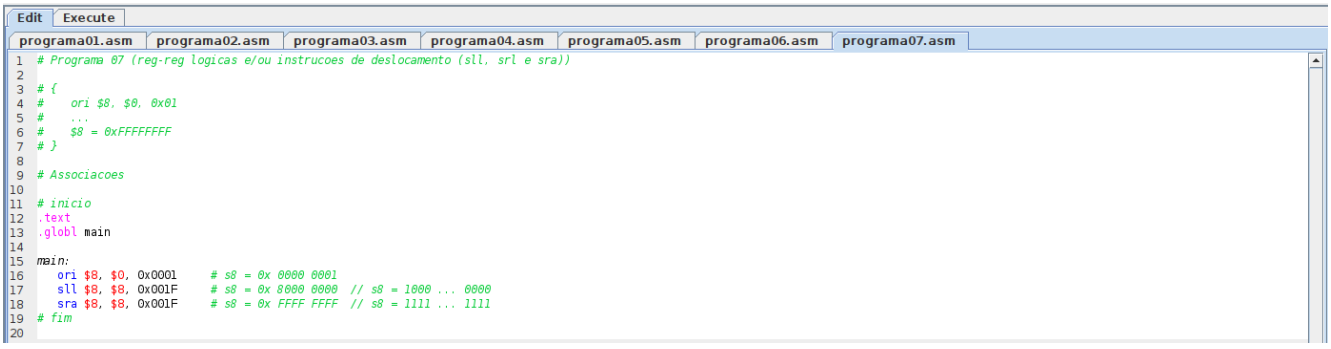


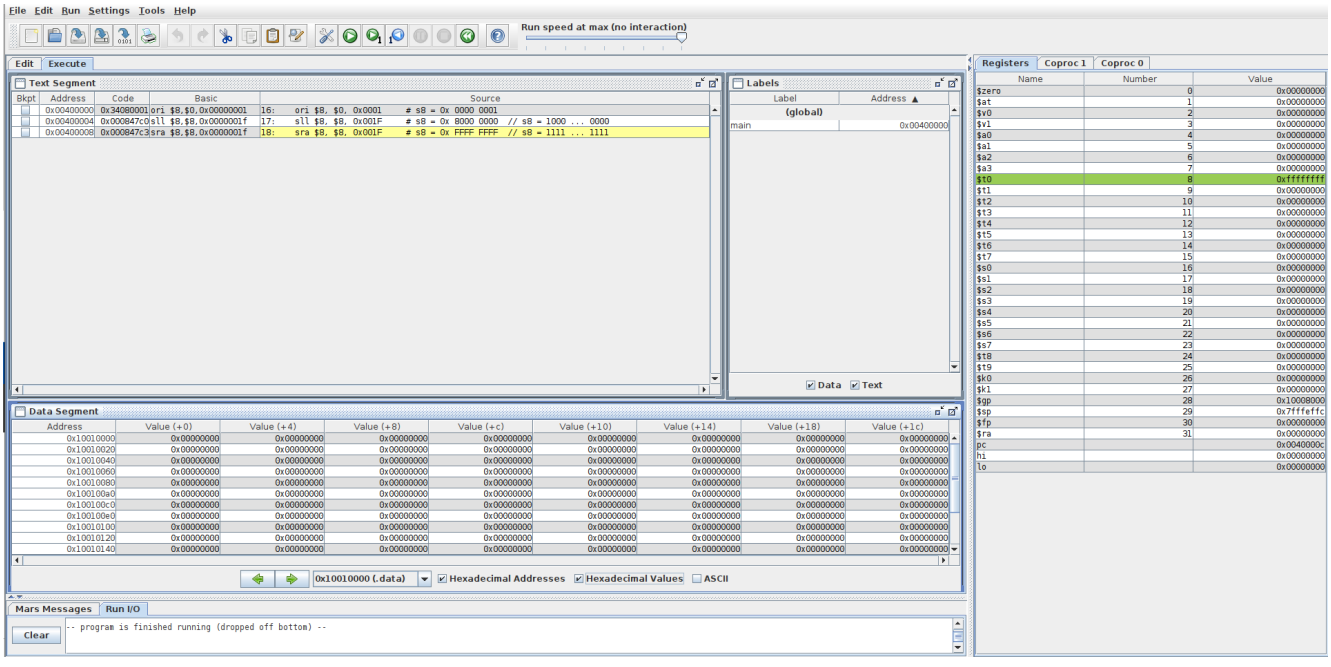
Programa 6:

```
# Programa 06 (sll, srl e sra)

# {
#    x = o maior inteiro possivel;
#    y = 300000;   //0004 93E0
#    z = x - 4y
# }

# Associacoes
# x -> $s0
# y -> $s1
# z -> $s2

# inicio
.text
.globl main

main:
    ori $t0, $zero, 0x7FFF   # t0 = 0x 0000 7FFF
    sll $t0, $t0, 0x0010     # t0 = 0x 7FFF 0000
    ori $s0, $t0, 0xFFFF     # x = 0x 7FFF FFFF   // x = 2147483647
    ori $t0, $zero, 0x0004   # t0 = 0x 0000 0004
    sll $t0, $t0, 0X0010     # t0 = 0x 0004 0000
    ori $s1, $t0, 0x93E0     # y = 0x 0004 93E0   // y = 300000
    sll $t0, $s1, 0x0002     # t0 = 4*y
    sub $s2, $s0, $t0        # z = x - 4*y
# fim
```


File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Text Segment
Bkpt Address Code Basic Source
0x00400000 0x34087fff ori $8,$0,32767 19: ori $t0, $zero, 0x7FFF # t0 = 0x 0000 7FFF
0x00400004 0x00084400 sll $8,$8,16 20: sll $t0, $t0, 0x0010 # t0 = 0x 7FFF 0000
0x00400008 0x3510ffff ori $16,$8,65535 21: ori $s0, $t0, 0xFFFF # x = 0x 7FFF FFFF // x = 2147483647
0x0040000c 0x34080004 ori $8,$0,4 22: ori $t0, $zero, 0x0004 # t0 = 0x 0000 0004
0x00400010 0x00084400 sll $8,$8,16 23: sll $t0, $t0, 0X0010 # t0 = 0x 0004 0000
0x00400014 0x351193e0 ori $17,$8,37856 24: ori $s1, $t0, 0x93E0 # y = 0x 0004 93E0 // y = 300000
0x00400018 0x00114080 sll $8,$17,2 25: sll $t0, $s1, 0x0002 # t0 = 4*y
0x0040001c 0x02089022 sub $18,$16,$8 26: sub $s2, $s0, $t0 # z = x - 4*y
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 1200000
$t1 9 0
$t2 10 0
$t3 11 0
$t4 12 0
$t5 13 0
$t6 14 0
$t7 15 0
$s0 16 2147483647
$s1 17 300000
$s2 18 2146283647
$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
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
0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII
Mars Messages Run I/O
Clear
-- program is finished running (dropped off bottom) --

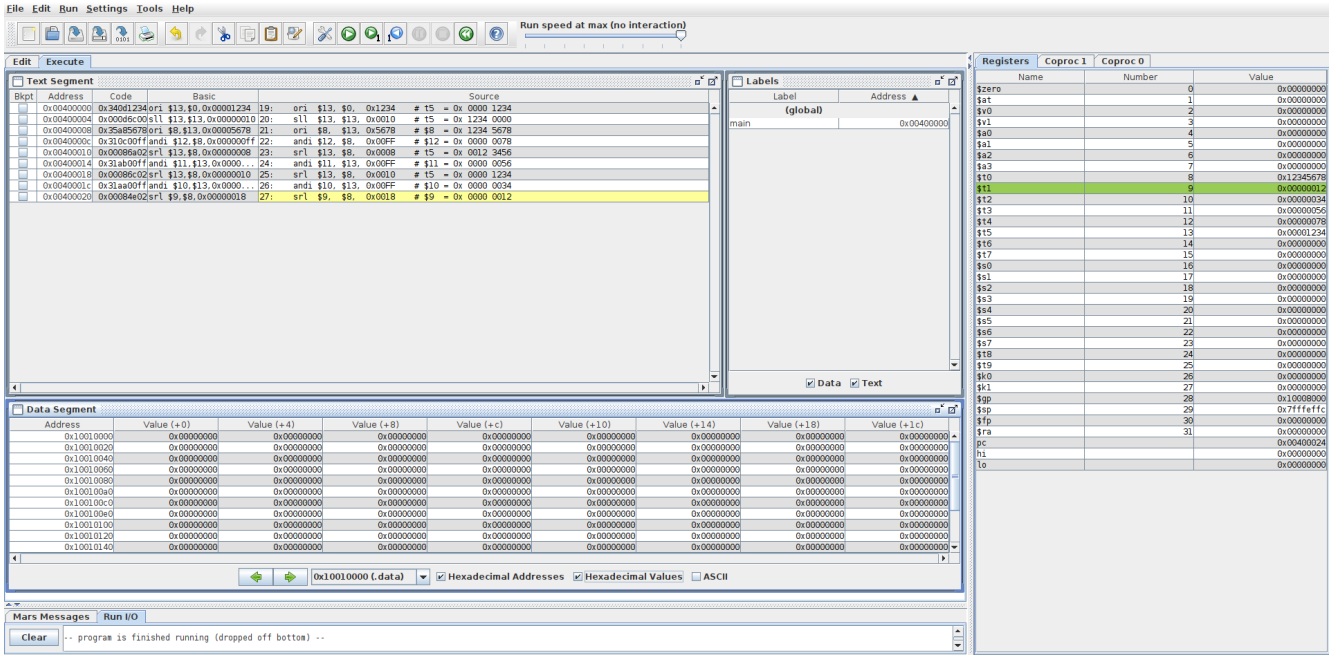
Programa 7:


Edit Execute
programa01.asm programa02.asm programa03.asm programa04.asm programa05.asm programa06.asm programa07.asm
1 # Programa 07 (reg-reg logicas e/ou instrucoes de deslocamento (sll, srl e sra))
2
3 # {
4 # ori $8, $0, 0x01
5 # ...
6 # $8 = 0xFFFFFFFF
7 # }
8
9 # Associacoes
10
11 # inicio
12 .text
13 .globl main
14
15 main:
16 ori $8, $0, 0x0001 # s8 = 0x 0000 0001
17 sll $8, $8, 0x001F # s8 = 0x 8000 0000 // s8 = 1000 ... 0000
18 sra $8, $8, 0x001F # s8 = 0x FFFF FFFF // s8 = 1111 ... 1111
19 # fim
20


File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Text Segment
Bkpt Address Code Basic Source
0x00400000 0x34080001 ori $8,$0,0x00000001 16: ori $8, $0, 0x0001 # s8 = 0x 0000 0001
0x00400004 0x000847c0 sll $8,$8,0x0000001f 17: sll $8, $8, 0x001F # s8 = 0x 8000 0000 // s8 = 1000 ... 0000
0x00400008 0x000847c3 sra $8,$8,0x0000001f 18: sra $8, $8, 0x001F # s8 = 0x FFFF FFFF // s8 = 1111 ... 1111
Labels
Label Address
(global)
main 0x00400000
Data Text
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
Data Segment
Address Value (+0) Value (+4) Value (+8) Value (+c) Value (+10) Value (+14) Value (+18) Value (+1c)
0x10010000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010020 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010040 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010060 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010080 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100100a0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100100c0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100100e0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010100 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010120 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010140 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII
Mars Messages Run I/O
Clear
-- program is finished running (dropped off bottom) --

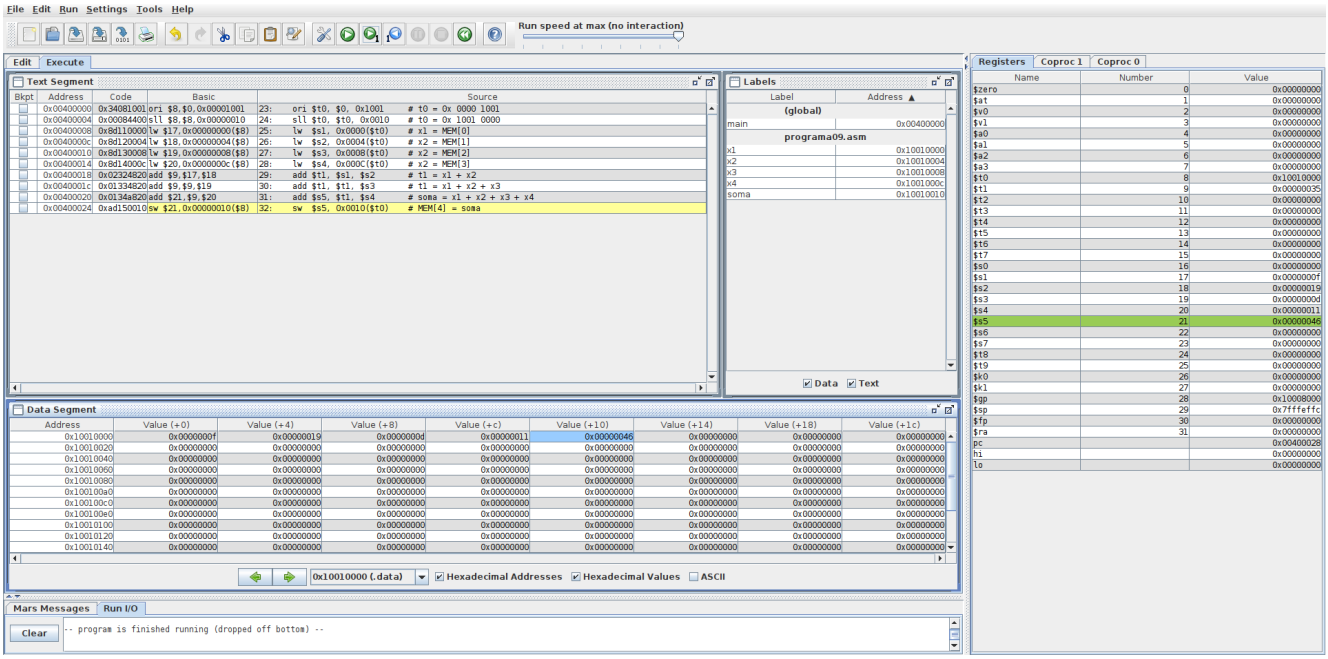
Programa 8:

```
# Programa 08 (logicas (or, ori, and, andi, xor, xori) e instrucoes de deslocamento (sll, srl e sra))

# {
#    $8 = 0x12345678
#    ... // nao nessa ordem
#    $9  = 0x12
#    $10 = 0x34
#    $11 = 0x56
#    $12 = 0x78
# }

# Associacoes

# inicio
.text
.globl main

main:
    ori  $13, $0,  0x1234    # t5  = 0x 0000 1234
    sll  $13, $13, 0x0010    # t5  = 0x 1234 0000
    ori  $8,  $13, 0x5678    # $8  = 0x 1234 5678
    andi $12, $8,  0x00FF    # $12 = 0x 0000 0078
    srl  $13, $8,  0x0008    # t5  = 0x 0012 3456
    andi $11, $13, 0x00FF    # $11 = 0x 0000 0056
    srl  $13, $8,  0x0010    # t5  = 0x 0000 1234
    andi $10, $13, 0x00FF    # $10 = 0x 0000 0034
    srl  $9,  $8,  0x0018    # $9  = 0x 0000 0012
# fim
```

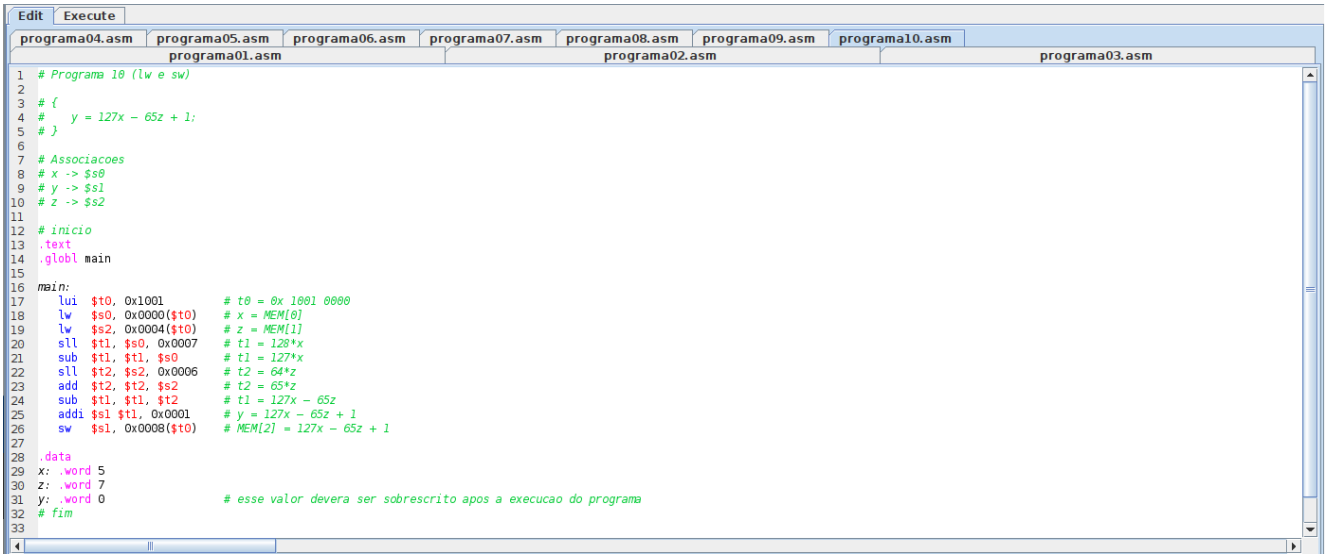


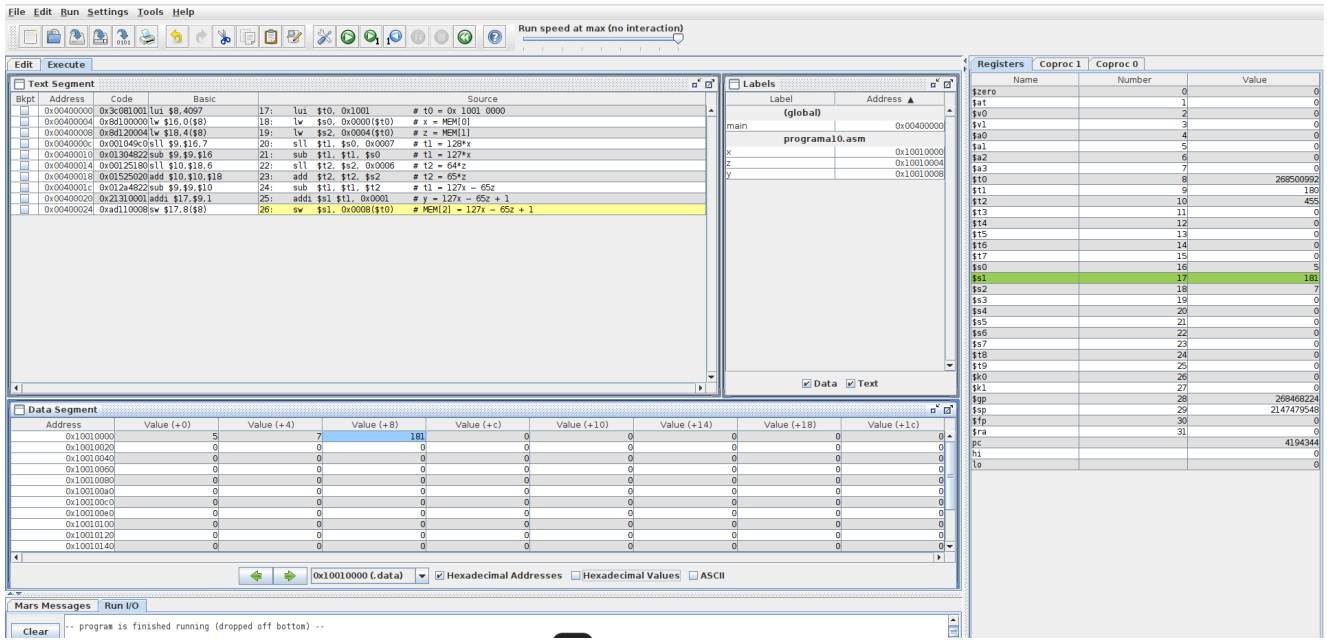
Programa 9:

```
# Programa 09 (lw e sw)

# {
#    x1 = MEM[0];
#    x2 = MEM[1];
#    x3 = MEM[2];
#    x4 = MEM[3];
#    MEM[4] = x1 + x2 + x3 + x4;
# }

# Associacoes
# x1 -> $s1
# x2 -> $s2
# x3 -> $s3
# x4 -> $s4
# soma -> $s5

# inicio
.text
.globl main

main:
    ori $t0, $0, 0x1001      # t0 = 0x 0000 1001
    sll $t0, $t0, 0x0010     # t0 = 0x 1001 0000
    lw  $s1, 0x0000($t0)     # x1 = MEM[0]
    lw  $s2, 0x0004($t0)     # x2 = MEM[1]
    lw  $s3, 0x0008($t0)     # x2 = MEM[2]
    lw  $s4, 0x000C($t0)     # x2 = MEM[3]
    add $t1, $s1, $s2        # t1 = x1 + x2
    add $t1, $t1, $s3        # t1 = x1 + x2 + x3
    add $s5, $t1, $s4        # soma = x1 + x2 + x3 + x4
    sw  $s5, 0x0010($t0)     # MEM[4] = soma

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

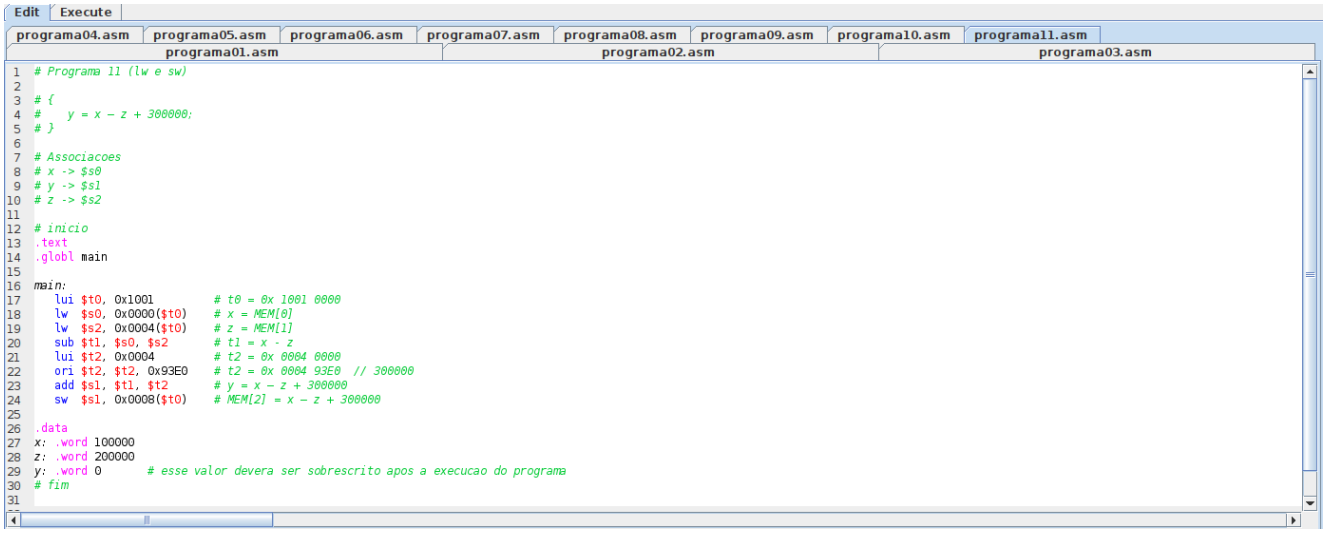

File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Text Segment
Bkpt Address Code Basic Source
0x00400000 0x34081001 ori $8,$0,0x00001001 23: ori $t0, $0, 0x1001 # t0 = 0x 0000 1001
0x00400004 0x00084400 sll $8,$8,0x00000010 24: sll $t0, $t0, 0x0010 # t0 = 0x 1001 0000
0x00400008 0x8d110000 lw $17,0x00000000($8) 25: lw $s1, 0x0000($t0) # x1 = MEM[0]
0x0040000c 0x8d120004 lw $18,0x00000004($8) 26: lw $s2, 0x0004($t0) # x2 = MEM[1]
0x00400010 0x8d130008 lw $19,0x00000008($8) 27: lw $s3, 0x0008($t0) # x2 = MEM[2]
0x00400014 0x8d14000c lw $20,0x0000000c($8) 28: lw $s4, 0x000C($t0) # x2 = MEM[3]
0x00400018 0x02324820 add $9,$17,$18 29: add $t1, $s1, $s2 # t1 = x1 + x2
0x0040001c 0x01334820 add $9,$9,$19 30: add $t1, $t1, $s3 # t1 = x1 + x2 + x3
0x00400020 0x0134a820 add $21,$9,$20 31: add $s5, $t1, $s4 # soma = x1 + x2 + x3 + x4
0x00400024 0xad150010 sw $21,0x00000010($8) 32: sw $s5, 0x0010($t0) # MEM[4] = soma
Labels
Label Address
(global)
main 0x00400000
programa09.asm
x1 0x10010000
x2 0x10010004
x3 0x10010008
x4 0x1001000c
soma 0x10010010
Data Segment
Address Value (+0) Value (+4) Value (+8) Value (+c) Value (+10) Value (+14) Value (+18) Value (+1c)
0x10010000 0x0000000f 0x00000019 0x0000000d 0x00000011 0x00000046 0x00000000 0x00000000 0x00000000
0x10010020 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010040 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010060 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010080 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100100a0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100100c0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100100e0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010100 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010120 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010140 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII
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 0x10010000
$t1 9 0x00000035
$t2 10 0x00000000
$t3 11 0x00000000
$t4 12 0x00000000
$t5 13 0x00000000
$t6 14 0x00000000
$t7 15 0x00000000
$s0 16 0x00000000
$s1 17 0x0000000f
$s2 18 0x00000019
$s3 19 0x0000000d
$s4 20 0x00000011
$s5 21 0x00000046
$s6 22 0x00000000
$s7 23 0x00000000
$t8 24 0x00000000
$t9 25 0x00000000
$k0 26 0x00000000
$k1 27 0x00000000
$gp 28 0x10008000
$sp 29 0x7fffeffc
$fp 30 0x00000000
$ra 31 0x00000000
pc 0x00400028
hi 0x00000000
lo 0x00000000
Mars Messages Run I/O
Clear
-- program is finished running (dropped off bottom) --

Programa 10:

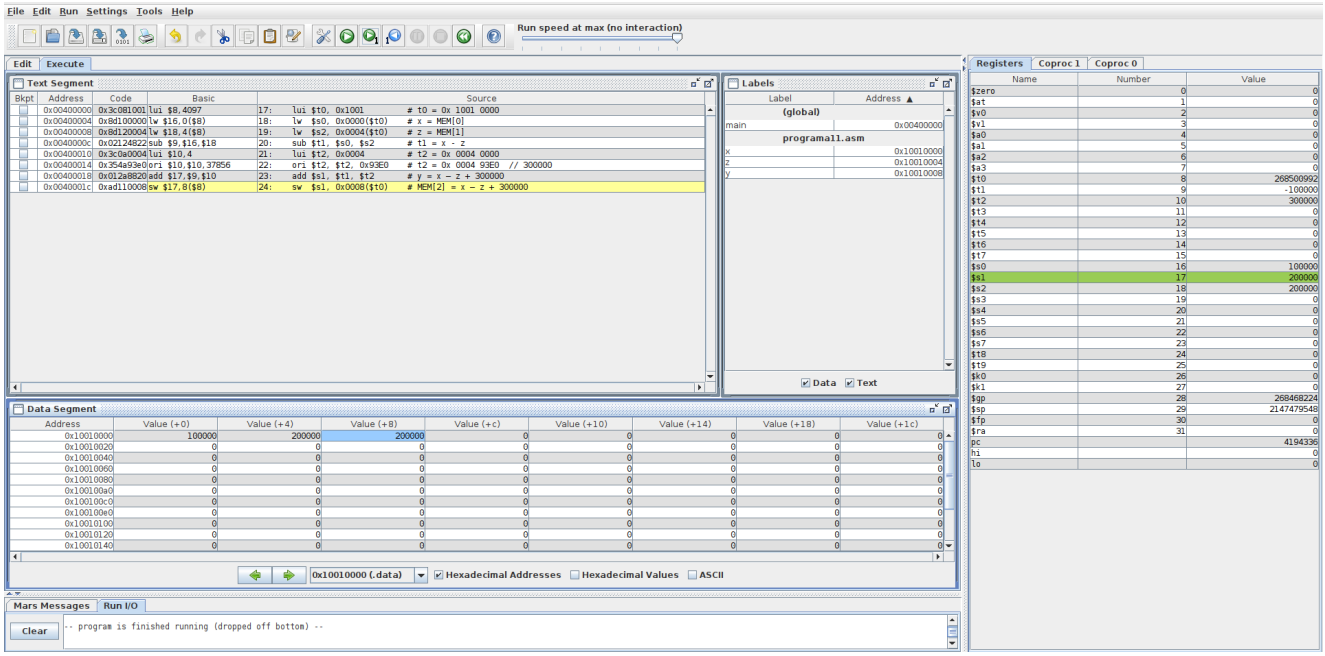

Edit Execute
programa04.asm programa05.asm programa06.asm programa07.asm programa08.asm programa09.asm programa10.asm
programa01.asm programa02.asm programa03.asm
1 # Programa 10 (lw e sw)
2
3 # {
4 # y = 127x - 65z + 1;
5 # }
6
7 # Associacoes
8 # x -> $s0
9 # y -> $s1
10 # z -> $s2
11
12 # inicio
13 .text
14 .globl main
15
16 main:
17 lui $t0, 0x1001 # t0 = 0x 1001 0000
18 lw $s0, 0x0000($t0) # x = MEM[0]
19 lw $s2, 0x0004($t0) # z = MEM[1]
20 sll $t1, $s0, 0x0007 # t1 = 128*x
21 sub $t1, $t1, $s0 # t1 = 127*x
22 sll $t2, $s2, 0x0006 # t2 = 64*z
23 add $t2, $t2, $s2 # t2 = 65*z
24 sub $t1, $t1, $t2 # t1 = 127x - 65z
25 addi $s1 $t1, 0x0001 # y = 127x - 65z + 1
26 sw $s1, 0x0008($t0) # MEM[2] = 127x - 65z + 1
27
28 .data
29 x: .word 5
30 z: .word 7
31 y: .word 0 # esse valor devera ser sobrescrito apos a execucao do programa
32 # fim
33


File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Text Segment
Bkpt Address Code Basic Source
0x00400000 0x3c081001 lui $8,4097 17: lui $t0, 0x1001 # t0 = 0x 1001 0000
0x00400004 0x8d100000 lw $16,0($8) 18: lw $s0, 0x0000($t0) # x = MEM[0]
0x00400008 0x8d120004 lw $18,4($8) 19: lw $s2, 0x0004($t0) # z = MEM[1]
0x0040000c 0x001049c0 sll $9,$16,7 20: sll $t1, $s0, 0x0007 # t1 = 128*x
0x00400010 0x01304822 sub $9,$9,$16 21: sub $t1, $t1, $s0 # t1 = 127*x
0x00400014 0x00125180 sll $10,$18,6 22: sll $t2, $s2, 0x0006 # t2 = 64*z
0x00400018 0x01525020 add $10,$10,$18 23: add $t2, $t2, $s2 # t2 = 65*z
0x0040001c 0x012a4822 sub $9,$9,$10 24: sub $t1, $t1, $t2 # t1 = 127x - 65z
0x00400020 0x21310001 addi $17,$9,1 25: addi $s1 $t1, 0x0001 # y = 127x - 65z + 1
0x00400024 0xad110008 sw $17,8($8) 26: sw $s1, 0x0008($t0) # 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
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 268500992
$t1 9 180
$t2 10 455
$t3 11 0
$t4 12 0
$t5 13 0
$t6 14 0
$t7 15 0
$s0 16 5
$s1 17 181
$s2 18 7
$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
Mars Messages Run I/O
Clear
-- program is finished running (dropped off bottom) --

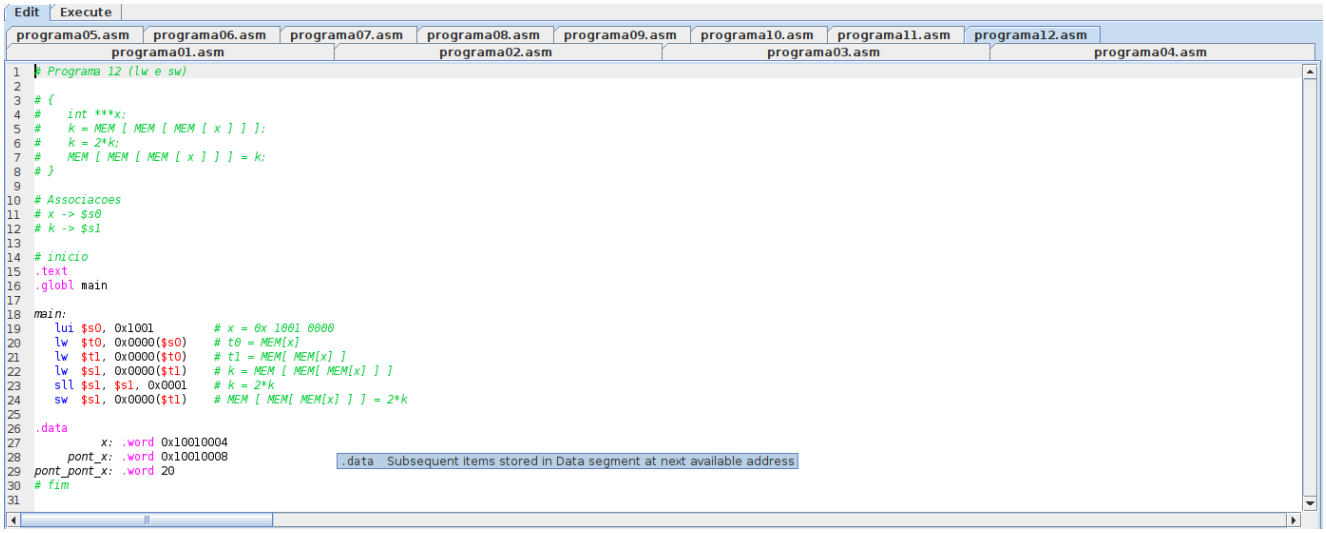
Programa 11:



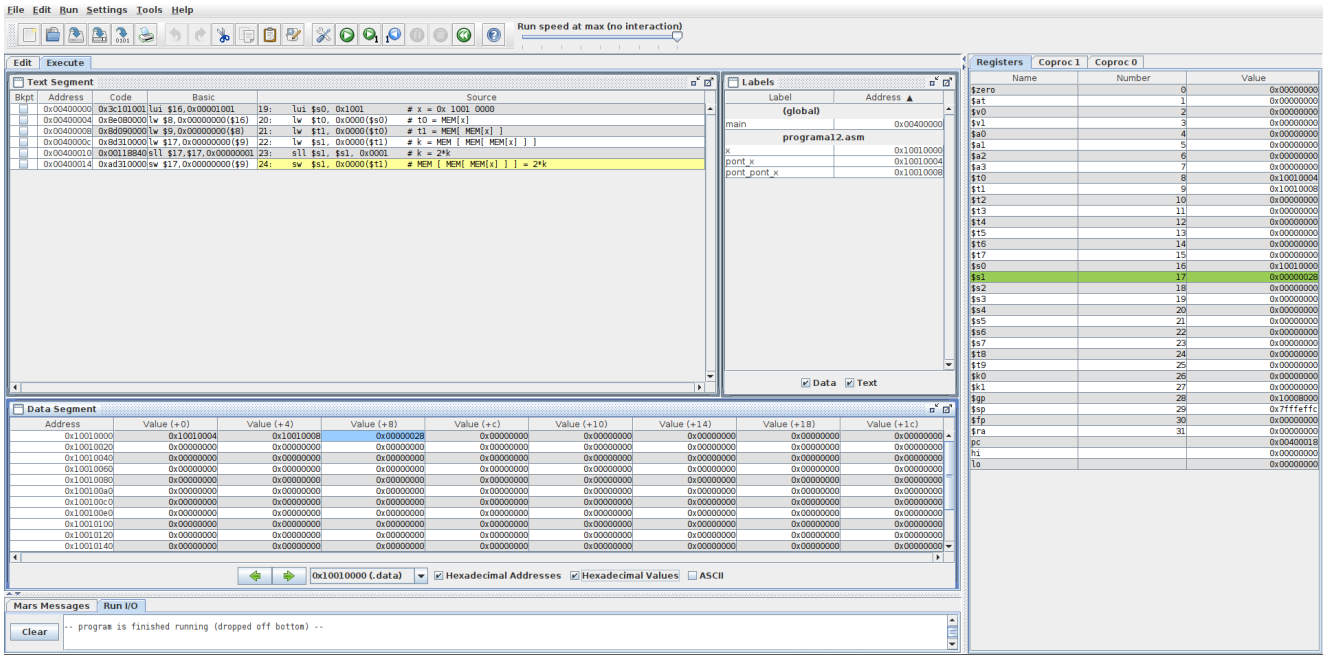
```
# Programa 11 (lw e sw)

# {
#    y = x - z + 300000;
# }

# Associacoes
# x -> $s0
# y -> $s1
# z -> $s2

# inicio
.text
.globl main

main:
   lui $t0, 0x1001         # t0 = 0x 1001 0000
   lw  $s0, 0x0000($t0)    # x = MEM[0]
   lw  $s2, 0x0004($t0)    # z = MEM[1]
   sub $t1, $s0, $s2       # t1 = x - z
   lui $t2, 0x0004         # t2 = 0x 0004 0000
   ori $t2, $t2, 0x93E0    # t2 = 0x 0004 93E0  // 300000
   add $s1, $t1, $t2       # y = x - z + 300000
   sw  $s1, 0x0008($t0)    # MEM[2] = x - z + 300000

.data
x: .word 100000
z: .word 200000
y: .word 0       # esse valor devera ser sobrescrito apos a execucao do programa
# fim
```



Programa 12:

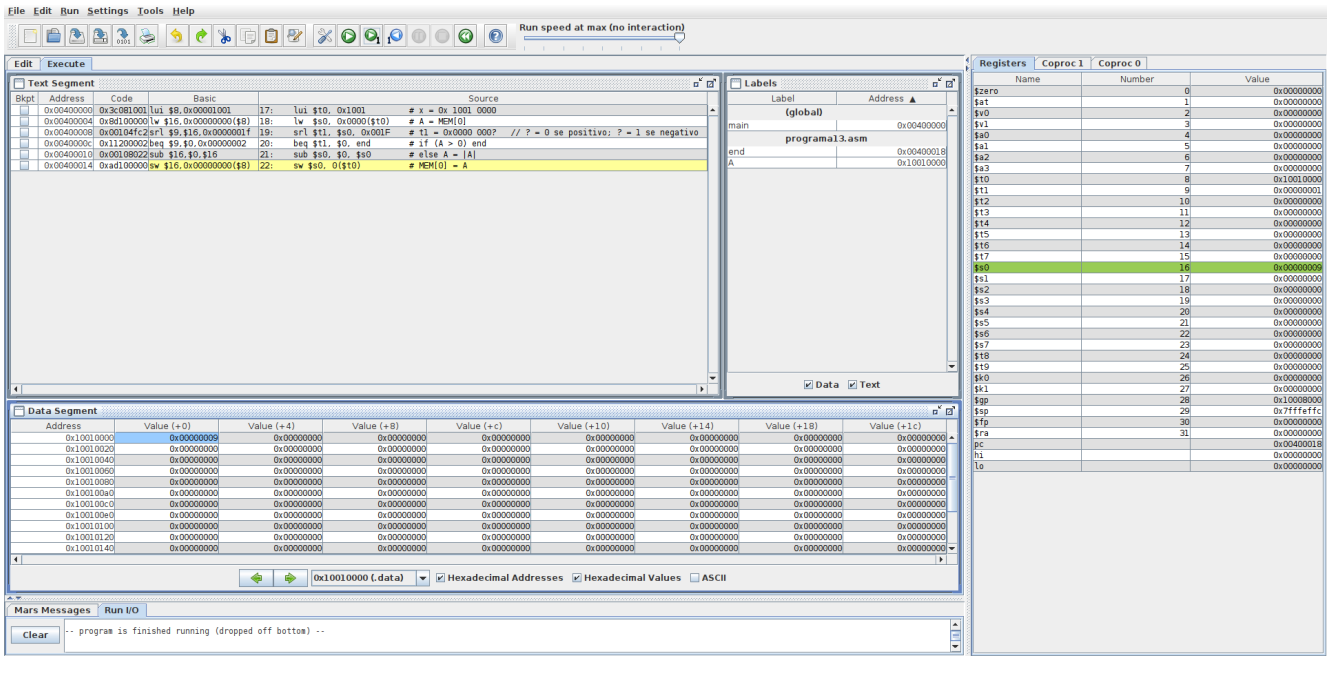


```
# Programa 12 (lw e sw)

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

# Associacoes
# x -> $s0
# k -> $s1

# inicio
.text
.globl main

main:
   lui $s0, 0x1001         # x = 0x 1001 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, 0x0001    # k = 2*k
   sw  $s1, 0x0000($t1)    # MEM [ MEM[ MEM[x] ] ] = 2*k

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

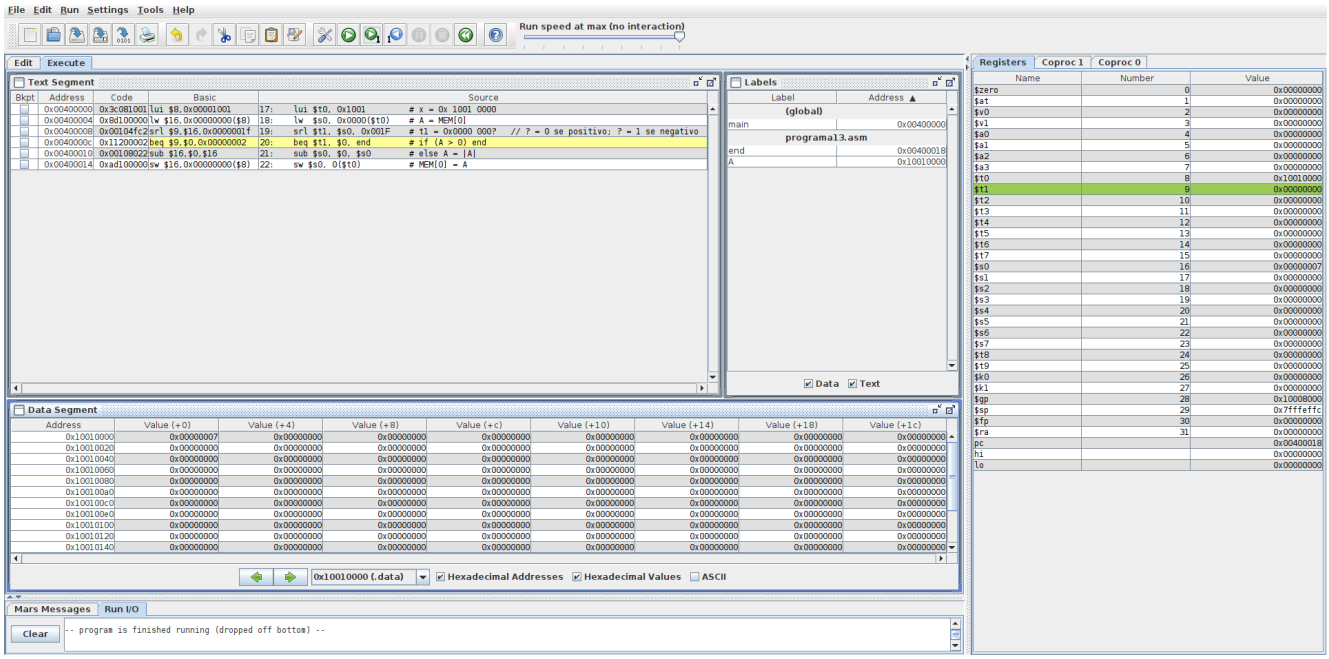

File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Text Segment
Bkpt Address Code Basic Source
0x00400000 0x3c101001 lui $16,0x00001001 19: lui $s0, 0x1001 # x = 0x 1001 0000
0x00400004 0x8e080000 lw $8,0x00000000($16) 20: lw $t0, 0x0000($s0) # t0 = MEM[x]
0x00400008 0x8d090000 lw $9,0x00000000($8) 21: lw $t1, 0x0000($t0) # t1 = MEM[MEM[x]]
0x0040000c 0x8d310000 lw $17,0x00000000($9) 22: lw $s1, 0x0000($t1) # k = MEM [MEM[MEM[x]]]
0x00400010 0x00118840 sll $17,$17,0x00000001 23: sll $s1, $s1, 0x0001 # k = 2*k
0x00400014 0xad310000 sw $17,0x00000000($9) 24: sw $s1, 0x0000($t1) # MEM [MEM[MEM[x]]] = 2*k
Labels
Label Address
(global)
main 0x00400000
programa12.asm
x 0x10010000
pont_x 0x10010004
pont_pont_x 0x1001000B
Data Segment
Address Value (+0) Value (+4) Value (+8) Value (+c) Value (+10) Value (+14) Value (+18) Value (+1c)
0x10010000 0x10010004 0x10010008 0x0000002B 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010020 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010040 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010060 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010080 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100100a0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100100c0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100100e0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010100 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010120 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010140 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII
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 0x1001000B
$t2 10 0x00000000
$t3 11 0x00000000
$t4 12 0x00000000
$t5 13 0x00000000
$t6 14 0x00000000
$t7 15 0x00000000
$s0 16 0x10010000
$s1 17 0x0000002B
$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
Mars Messages Run I/O
Clear
-- program is finished running (dropped off bottom) --

Programa 13:

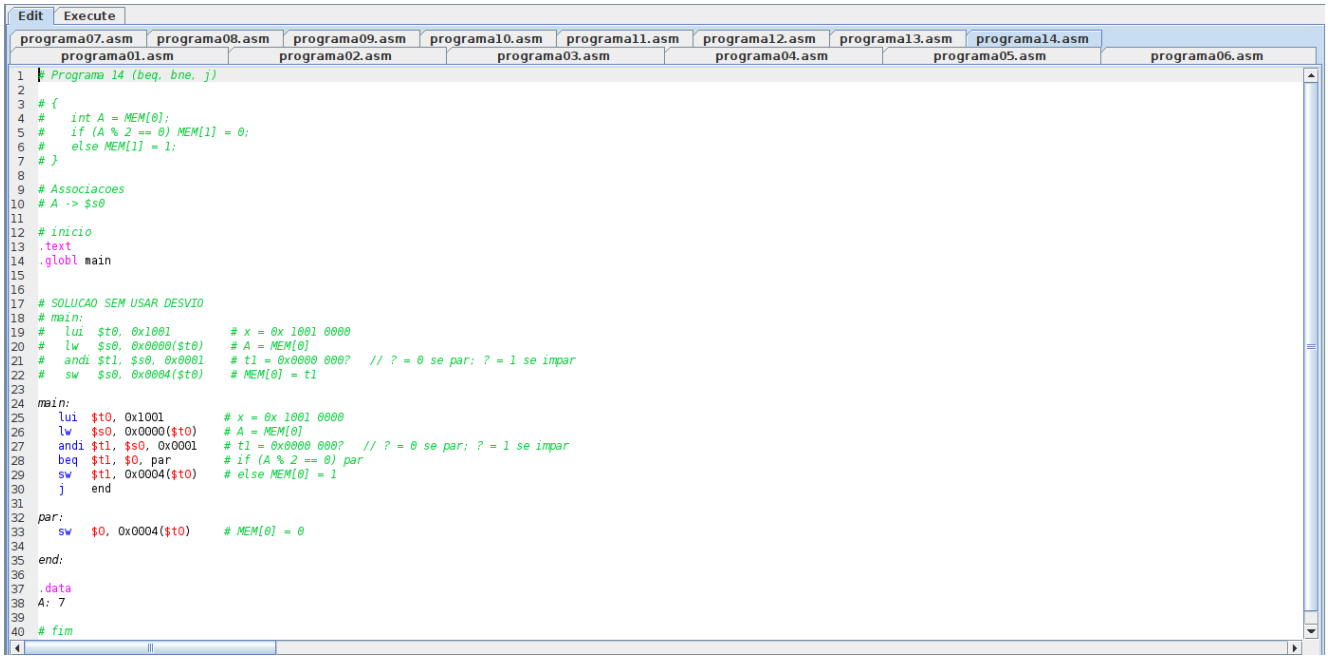

Edit Execute
programa06.asm programa07.asm programa08.asm programa09.asm programa10.asm programa11.asm programa12.asm programa13.asm
programa01.asm programa02.asm programa03.asm programa04.asm programa05.asm
1 # Programa 13 (beq, bne, j)
2
3 # {
4 # int A = MEM[0];
5 # if (A < 0) A = -A;
6 # MEM[0] = A;
7 # }
8
9 # Associacoes
10 # A -> $s0
11
12 # inicio
13 .text
14 .globl main
15
16 main:
17 lui $t0, 0x1001 # x = 0x 1001 0000
18 lw $s0, 0x0000($t0) # A = MEM[0]
19 srl $t1, $s0, 0x001F # t1 = 0x0000 000? // ? = 0 se positivo; ? = 1 se negativo
20 beq $t1, $0, end # if (A > 0) end
21 sub $s0, $0, $s0 # else A = |A|
22 sw $s0, 0($t0) # MEM[0] = A
23
24 end:
25
26 .data
27 A: -9
28
29 # fim
30

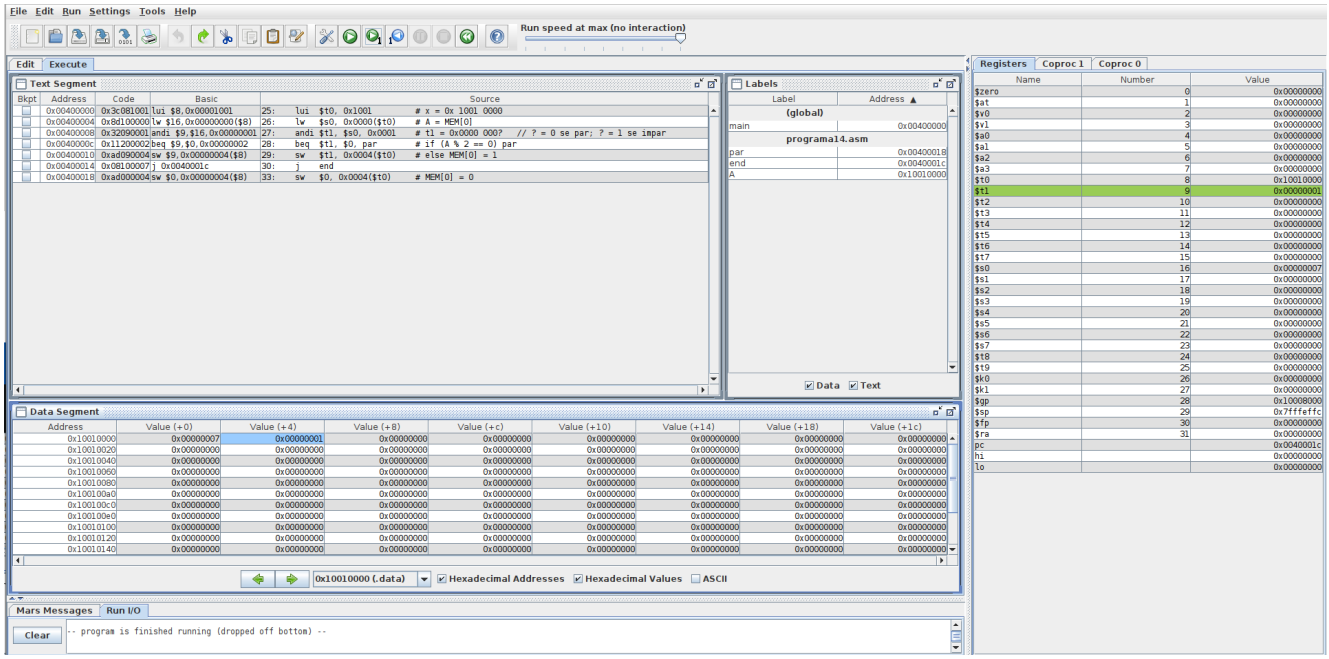

File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Text Segment
Bkpt Address Code Basic Source
0x00400000 0x3c081001 lui $8,0x00001001 17: lui $t0, 0x1001 # x = 0x 1001 0000
0x00400004 0x8d100000 lw $16,0x00000000($8) 18: lw $s0, 0x0000($t0) # A = MEM[0]
0x00400008 0x00104fc2 srl $9,$16,0x0000001f 19: srl $t1, $s0, 0x001F # t1 = 0x0000 000? // ? = 0 se positivo; ? = 1 se negativo
0x0040000c 0x11200002 beq $9,$0,0x00000002 20: beq $t1, $0, end # if (A > 0) end
0x00400010 0x00108022 sub $16,$0,$16 21: sub $s0, $0, $s0 # else A = |A|
0x00400014 0xad100000 sw $16,0x00000000($8) 22: sw $s0, 0($t0) # MEM[0] = A
Labels
Label Address
(global)
main 0x00400000
programa13.asm
end 0x00400018
A 0x10010000
Data Segment
Address Value (+0) Value (+4) Value (+8) Value (+c) Value (+10) Value (+14) Value (+18) Value (+1c)
0x10010000 0x00000009 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010020 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010040 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010060 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010080 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100100a0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100100c0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100100e0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010100 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010120 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010140 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII
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 0x10010000
$t1 9 0x00000001
$t2 10 0x00000000
$t3 11 0x00000000
$t4 12 0x00000000
$t5 13 0x00000000
$t6 14 0x00000000
$t7 15 0x00000000
$s0 16 0x00000009
$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 0x00400018
hi 0x00000000
lo 0x00000000
Mars Messages Run I/O
Clear
-- program is finished running (dropped off bottom) --

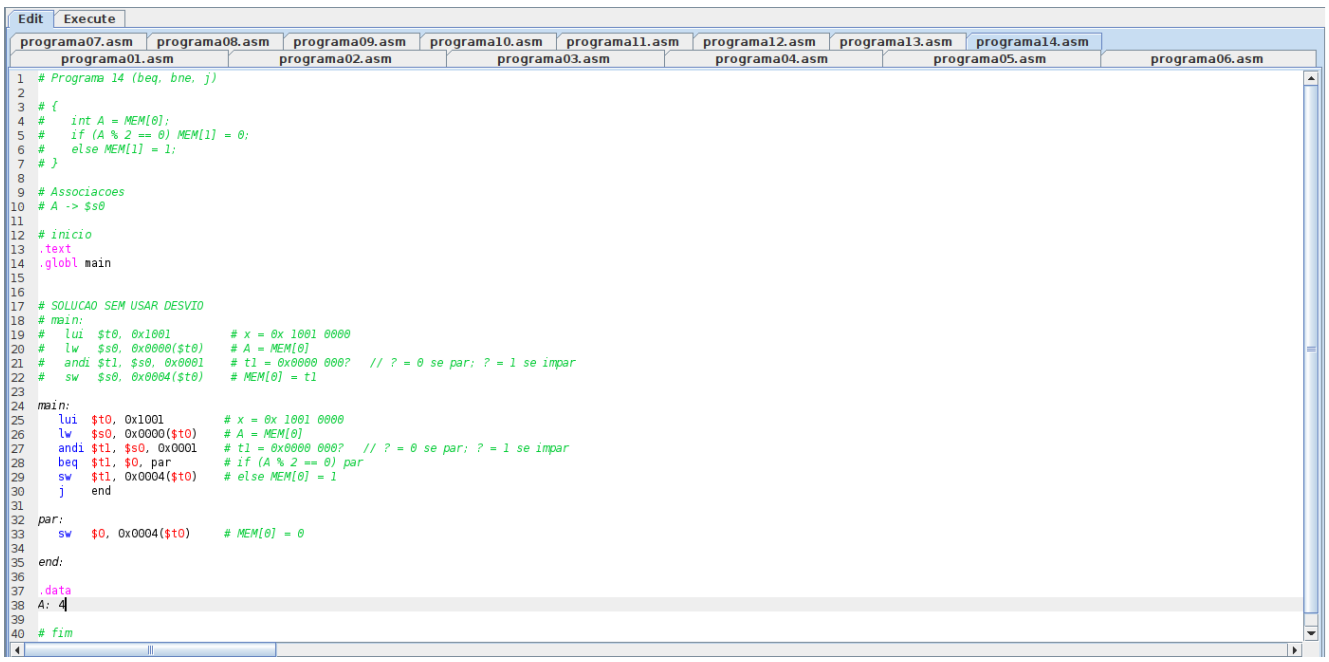

Edit
Execute
programa06.asm
programa07.asm
programa08.asm
programa09.asm
programa10.asm
programa11.asm
programa12.asm
programa13.asm
programa01.asm
programa02.asm
programa03.asm
programa04.asm
programa05.asm
Programa 13 (beq, bne, j)
{
int A = MEM[0];
if (A < 0) A = -A;
MEM[0] = A;
}
Associacoes
A -> $s0
inicio
.text
.globl main
main:
lui $t0, 0x1001 # x = 0x 1001 0000
lw $s0, 0x0000($t0) # A = MEM[0]
srl $t1, $s0, 0x001F # t1 = 0x0000 000? // ? = 0 se positivo; ? = 1 se negativo
beq $t1, $0, end # if (A > 0) end
sub $s0, $0, $s0 # else A = |A|
sw $s0, 0($t0) # MEM[0] = A
end:
.data
A: 7
fim

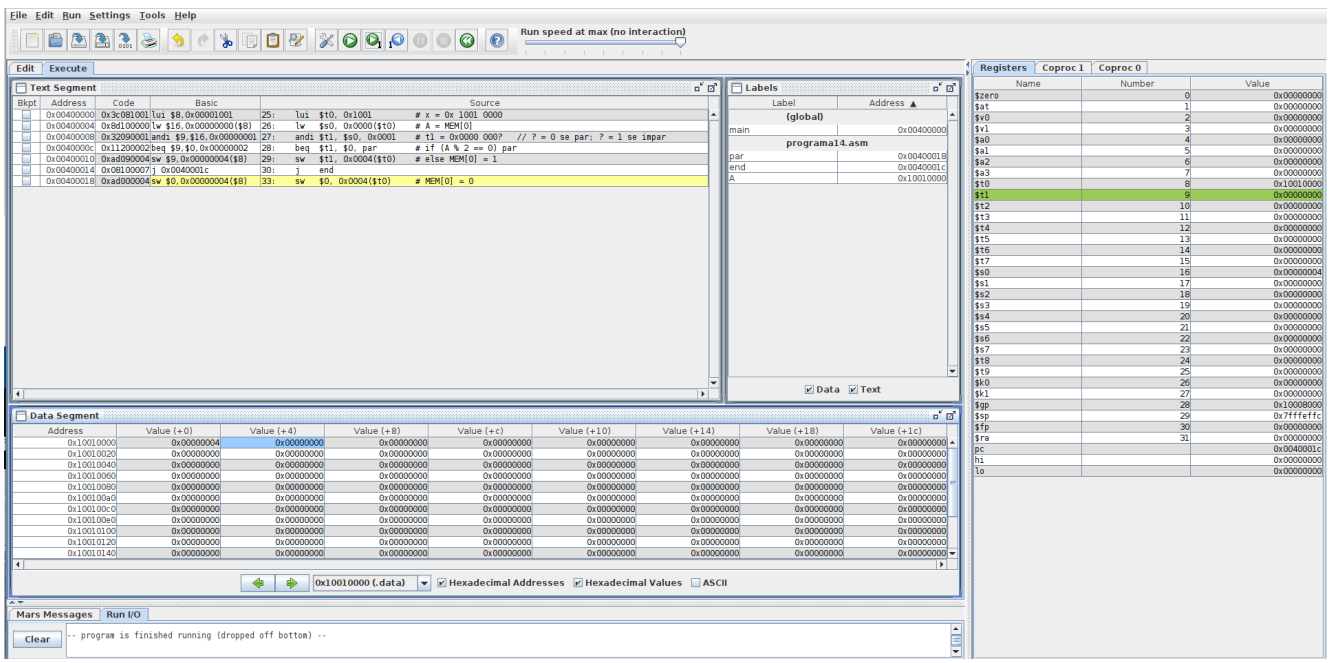

File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Text Segment
Bkpt Address Code Basic Source
0x00400000 0x3c081001 lui $8,0x00001001 17: lui $t0, 0x1001 # x = 0x 1001 0000
0x00400004 0x8d100000 lw $16,0x00000000($8) 18: lw $s0, 0x0000($t0) # A = MEM[0]
0x00400008 0x00104fc2 srl $9,$16,0x0000001f 19: srl $t1, $s0, 0x001F # t1 = 0x0000 000? // ? = 0 se positivo; ? = 1 se negativo
0x0040000c 0x11200002 beq $9,$0,0x00000002 20: beq $t1, $0, end # if (A > 0) end
0x00400010 0x00108022 sub $16,$0,$16 21: sub $s0, $0, $s0 # else A = |A|
0x00400014 0xad100000 sw $16,0x00000000($8) 22: sw $s0, 0($t0) # MEM[0] = A
Labels
Label Address
(global)
main 0x00400000
programa13.asm
end 0x00400018
A 0x10010000
Data Text
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 0x10010000
$t1 9 0x00000000
$t2 10 0x00000000
$t3 11 0x00000000
$t4 12 0x00000000
$t5 13 0x00000000
$t6 14 0x00000000
$t7 15 0x00000000
$s0 16 0x00000007
$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 0x00400018
hi 0x00000000
lo 0x00000000
Data Segment
Address Value (+0) Value (+4) Value (+8) Value (+c) Value (+10) Value (+14) Value (+18) Value (+1c)
0x10010000 0x00000007 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010020 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010040 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010060 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010080 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100100a0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100100c0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100100e0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010100 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010120 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010140 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010000 (.data)
Hexadecimal Addresses
Hexadecimal Values
ASCII
Mars Messages
Run I/O
Clear
-- program is finished running (dropped off bottom) --

Programa 14:

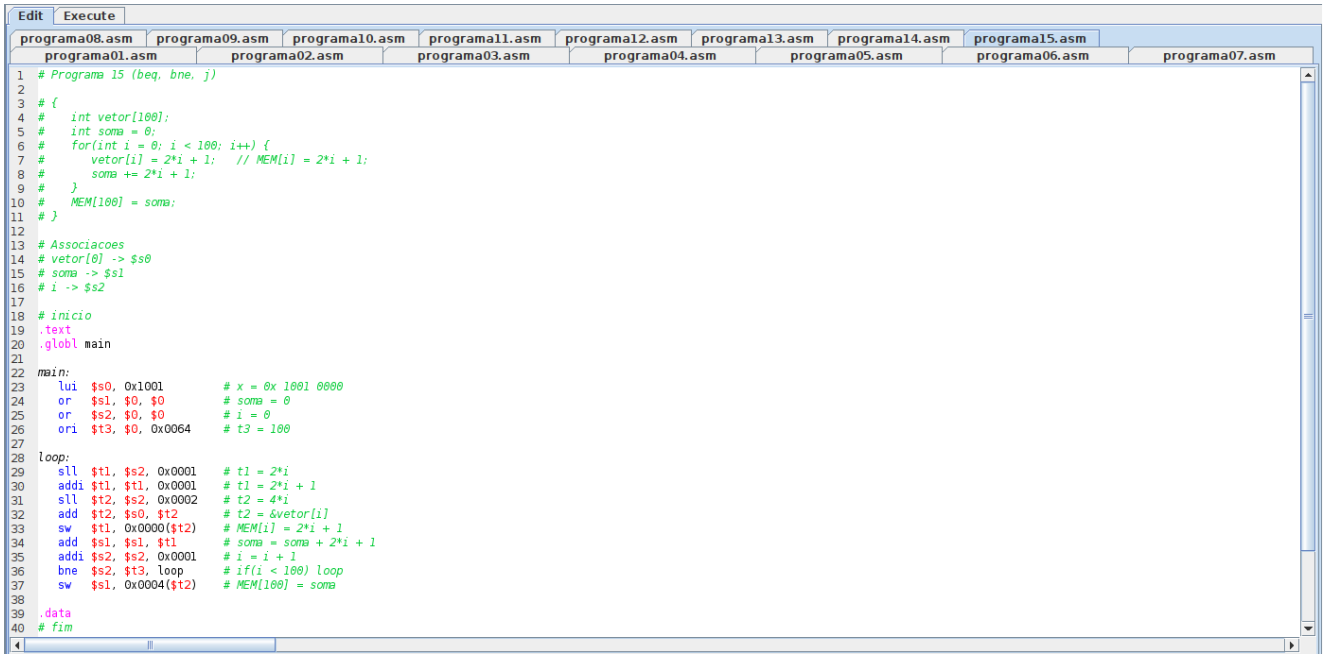

Edit
Execute
programa07.asm
programa08.asm
programa09.asm
programa10.asm
programa11.asm
programa12.asm
programa13.asm
programa14.asm
programa01.asm
programa02.asm
programa03.asm
programa04.asm
programa05.asm
programa06.asm
Programa 14 (beq, bne, j)
{
int A = MEM[0];
if (A % 2 == 0) MEM[1] = 0;
else MEM[1] = 1;
}
Associacoes
A -> $s0
inicio
.text
.globl main
SOLUCAO SEM USAR DESVIO
main:
lui $t0, 0x1001 # x = 0x 1001 0000
lw $s0, 0x0000($t0) # A = MEM[0]
andi $t1, $s0, 0x0001 # t1 = 0x0000 000? // ? = 0 se par; ? = 1 se impar
sw $s0, 0x0004($t0) # MEM[0] = t1
main:
lui $t0, 0x1001 # x = 0x 1001 0000
lw $s0, 0x0000($t0) # A = MEM[0]
andi $t1, $s0, 0x0001 # t1 = 0x0000 000? // ? = 0 se par; ? = 1 se impar
beq $t1, $0, par # if (A % 2 == 0) par
sw $t1, 0x0004($t0) # else MEM[0] = 1
j end
par:
sw $0, 0x0004($t0) # MEM[0] = 0
end:
.data
A: 7
fim


File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Text Segment
Bkpt Address Code Basic Source
0x00400000 0x3c081001 lui $8,0x00001001 25: lui $t0, 0x1001 # x = 0x 1001 0000
0x00400004 0x8d100000 lw $16, 0x00000000($8) 26: lw $s0, 0x0000($t0) # A = MEM[0]
0x00400008 0x32090001 andi $9,$16,0x00000001 27: andi $t1, $s0, 0x0001 # t1 = 0x0000 000? // ? = 0 se par; ? = 1 se impar
0x0040000c 0x11200002 beq $9,$0,0x00000002 28: beq $t1, $0, par # if (A % 2 == 0) par
0x00400010 0xad090004 sw $9,0x00000004($8) 29: sw $t1, 0x0004($t0) # else MEM[0] = 1
0x00400014 0x08100007 j 0x0040001c 30: j end
0x00400018 0xad000004 sw $0,0x00000004($8) 33: sw $0, 0x0004($t0) # MEM[0] = 0
Labels
Label Address
(global)
main 0x00400000
programa14.asm
par 0x00400018
end 0x0040001c
A 0x10010000
Data Text
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 0x10010000
$t1 9 0x00000001
$t2 10 0x00000000
$t3 11 0x00000000
$t4 12 0x00000000
$t5 13 0x00000000
$t6 14 0x00000000
$t7 15 0x00000000
$s0 16 0x00000007
$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
Data Segment
Address Value (+0) Value (+4) Value (+8) Value (+c) Value (+10) Value (+14) Value (+18) Value (+1c)
0x10010000 0x00000007 0x00000001 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010020 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010040 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010060 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010080 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100100a0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100100c0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100100e0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010100 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010120 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010140 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII
Mars Messages Run I/O
Clear
-- program is finished running (dropped off bottom) --

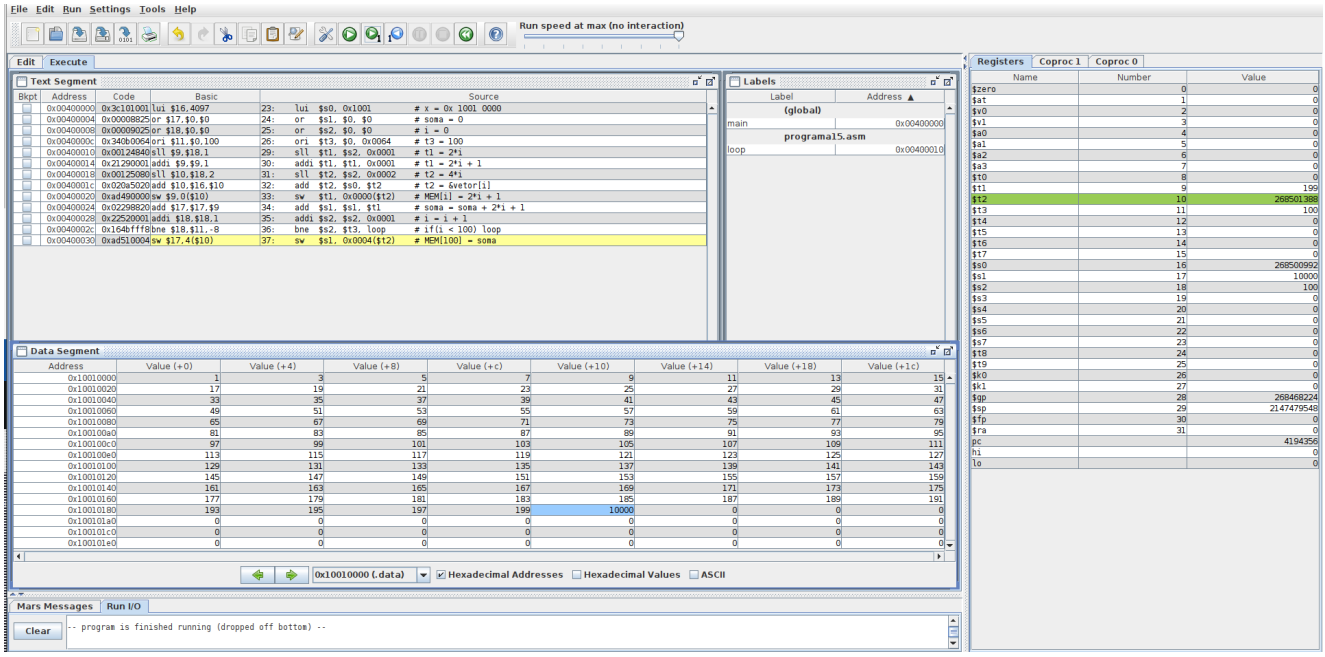

Edit Execute
programa07.asm programa08.asm programa09.asm programa10.asm programa11.asm programa12.asm programa13.asm programa14.asm
programa01.asm programa02.asm programa03.asm programa04.asm programa05.asm programa06.asm
1 # Programa 14 (beq, bne, j)
2
3 # {
4 # int A = MEM[0];
5 # if (A % 2 == 0) MEM[1] = 0;
6 # else MEM[1] = 1;
7 # }
8
9 # Associacoes
10 # A -> $s0
11
12 # inicio
13 .text
14 .globl main
15
16
17 # SOLUCAO SEM USAR DESVIO
18 # main:
19 # lui $t0, 0x1001 # x = 0x 1001 0000
20 # lw $s0, 0x0000($t0) # A = MEM[0]
21 # andi $t1, $s0, 0x0001 # t1 = 0x0000 000? // ? = 0 se par; ? = 1 se impar
22 # sw $s0, 0x0004($t0) # MEM[0] = t1
23
24 main:
25 lui $t0, 0x1001 # x = 0x 1001 0000
26 lw $s0, 0x0000($t0) # A = MEM[0]
27 andi $t1, $s0, 0x0001 # t1 = 0x0000 000? // ? = 0 se par; ? = 1 se impar
28 beq $t1, $0, par # if (A % 2 == 0) par
29 sw $t1, 0x0004($t0) # else MEM[0] = 1
30 j end
31
32 par:
33 sw $0, 0x0004($t0) # MEM[0] = 0
34
35 end:
36
37 .data
38 A: 4
39
40 # fim

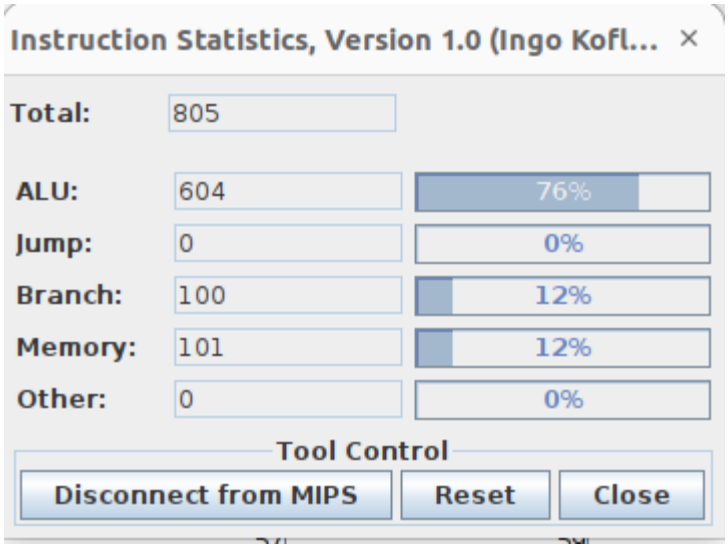

File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Text Segment
Bkpt Address Code Basic Source
0x00400000 0x3c081001 lui $8,0x00001001 25: lui $t0, 0x1001 # x = 0x 1001 0000
0x00400004 0x8d100000 lw $16, 0x00000000($8) 26: lw $s0, 0x0000($t0) # A = MEM[0]
0x00400008 0x32090001 andi $9,$16,0x00000001 27: andi $t1, $s0, 0x0001 # t1 = 0x0000 000? // ? = 0 se par; ? = 1 se impar
0x0040000c 0x11200002 beq $9,$0,0x00000002 28: beq $t1, $0, par # if (A % 2 == 0) par
0x00400010 0xad090004 sw $9,0x00000004($8) 29: sw $t1, 0x0004($t0) # else MEM[0] = 1
0x00400014 0x08100007 j 0x0040001c 30: j end
0x00400018 0xad000004 sw $0,0x00000004($8) 33: sw $0, 0x0004($t0) # MEM[0] = 0
Labels
Label Address
(global)
main 0x00400000
programa14.asm
par 0x00400018
end 0x0040001c
A 0x10010000
Data Text
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 0x10010000
$t1 9 0x00000000
$t2 10 0x00000000
$t3 11 0x00000000
$t4 12 0x00000000
$t5 13 0x00000000
$t6 14 0x00000000
$t7 15 0x00000000
$s0 16 0x00000004
$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
Data Segment
Address Value (+0) Value (+4) Value (+8) Value (+c) Value (+10) Value (+14) Value (+18) Value (+1c)
0x10010000 0x00000004 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010020 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010040 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010060 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010080 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100100a0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100100c0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100100e0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010100 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010120 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010140 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII
Mars Messages Run I/O
Clear
-- program is finished running (dropped off bottom) --

Programa 15:



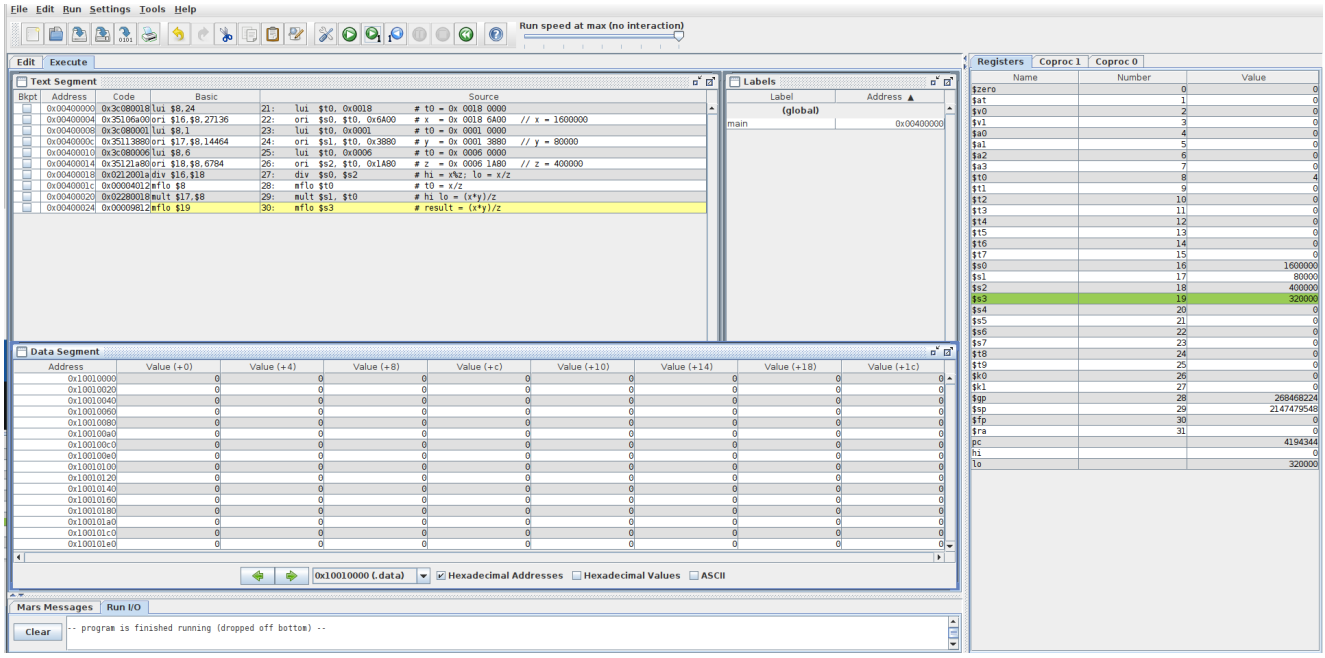
```
# Programa 15 (beq, bne, j)

# {
#    int vetor[100];
#    int soma = 0;
#    for(int i = 0; i < 100; i++) {
#       vetor[i] = 2*i + 1;   // MEM[i] = 2*i + 1;
#       soma += 2*i + 1;
#    }
#    MEM[100] = soma;
# }

# Associacoes
# vetor[0] -> $s0
# soma -> $s1
# i -> $s2

# inicio
.text
.globl main

main:
    lui  $s0, 0x1001          # x = 0x 1001 0000
    or   $s1, $0, $0          # soma = 0
    or   $s2, $0, $0          # i = 0
    ori  $t3, $0, 0x0064      # t3 = 100

loop:
    sll  $t1, $s2, 0x0001     # t1 = 2*i
    addi $t1, $t1, 0x0001     # t1 = 2*i + 1
    sll  $t2, $s2, 0x0002     # t2 = 4*i
    add  $t2, $s0, $t2        # t2 = &vetor[i]
    sw   $t1, 0x0000($t2)     # MEM[i] = 2*i + 1
    add  $s1, $s1, $t1        # soma = soma + 2*i + 1
    addi $s2, $s2, 0x0001     # i = i + 1
    bne  $s2, $t3, loop       # if(i < 100) loop
    sw   $s1, 0x0004($t2)     # MEM[100] = soma

.data
# fim
```



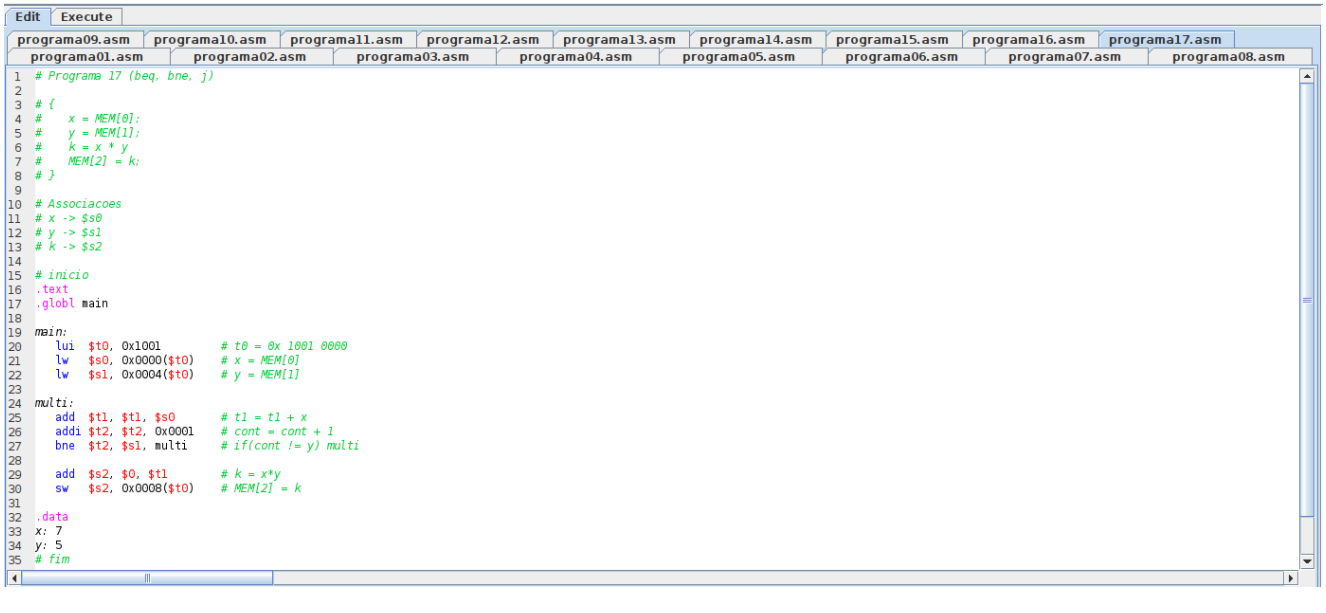


Programa 16:

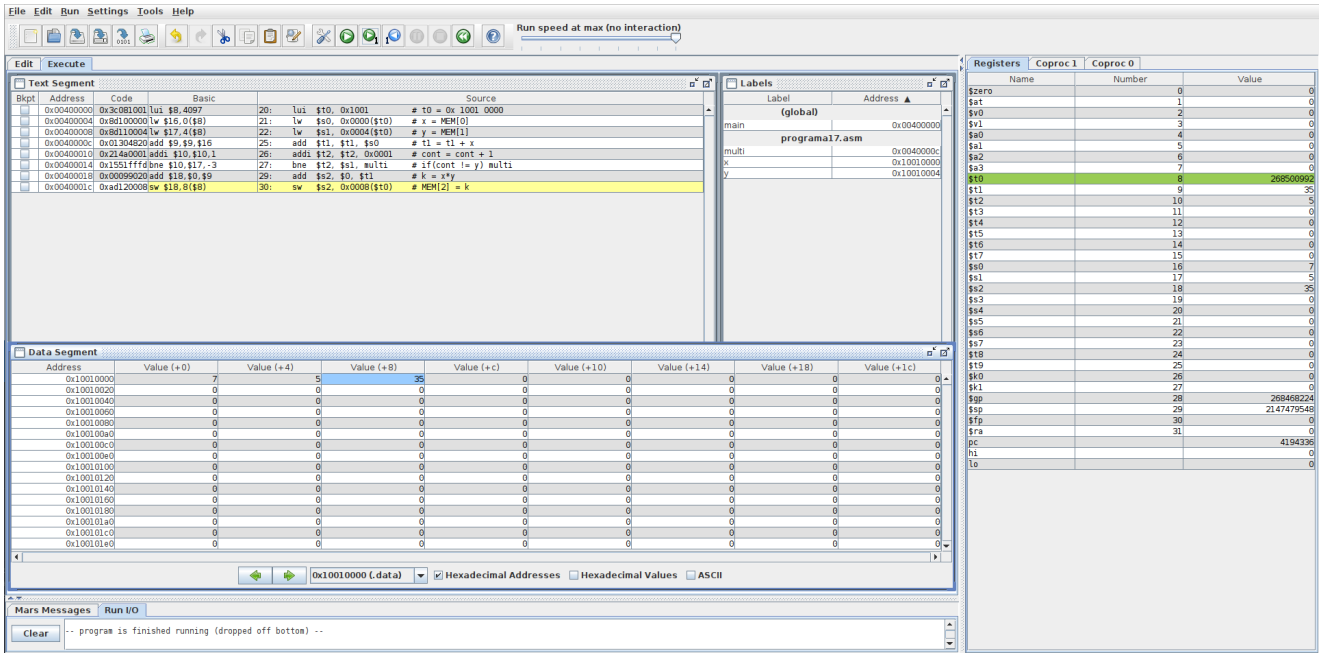
```
# Programa 16 (beq, bne, j)

# {
#    x = 1600000;    // x = 0x 0018 6A00
#    y =   80000;    // y = 0x 0001 3880
#    z =  400000;    // z = 0x 0006 1A80
#    result = (x*y)/z;
# }

# Associacoes
# x -> $s0
# y -> $s1
# z -> $s2
# result -> $s3

# inicio
.text
.globl main

main:
    lui  $t0, 0x0018        # t0 = 0x 0018 0000
    ori  $s0, $t0, 0x6A00   # x  = 0x 0018 6A00   // x = 1600000
    lui  $t0, 0x0001        # t0 = 0x 0001 0000
    ori  $s1, $t0, 0x3880   # y  = 0x 0001 3880   // y = 80000
    lui  $t0, 0x0006        # t0 = 0x 0006 0000
    ori  $s2, $t0, 0x1A80   # z  = 0x 0006 1A80   // z = 400000
    div  $s0, $s2           # hi = x%z; lo = x/z
    mflo $t0                # t0 = x/z
    mult $s1, $t0           # hi lo = (x*y)/z
    mflo $s3                # result = (x*y)/z

.data
# fim
```

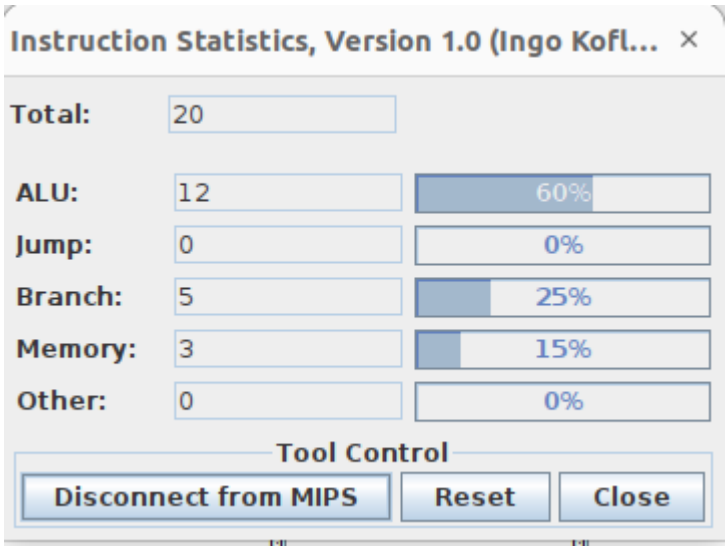


Programa 17:

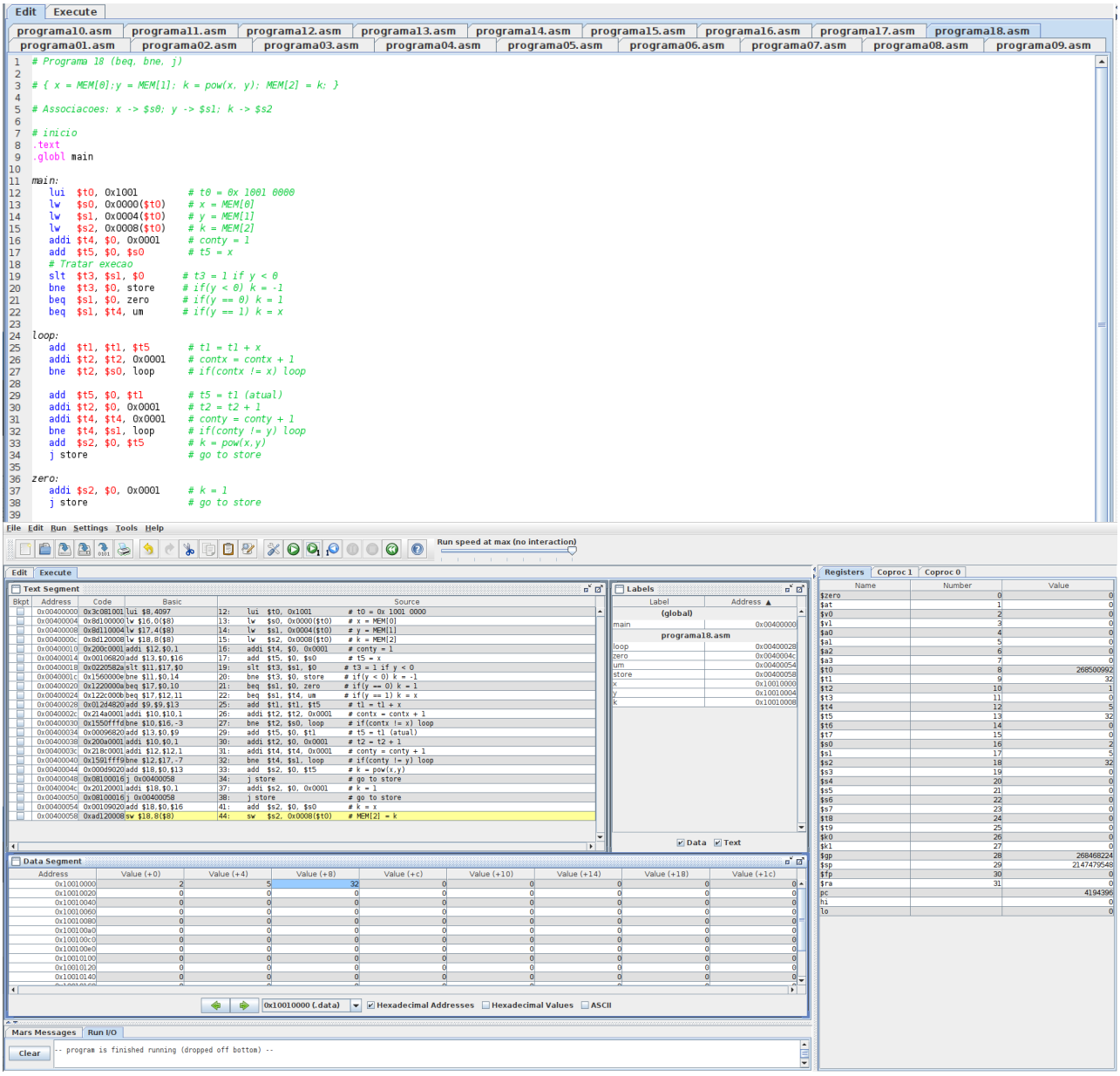


```
# Programa 17 (beq, bne, j)

# {
#    x = MEM[0];
#    y = MEM[1];
#    k = x * y
#    MEM[2] = k;
# }

# Associacoes
# x -> $s0
# y -> $s1
# k -> $s2

# inicio
.text
.globl main

main:
    lui  $t0, 0x1001        # t0 = 0x 1001 0000
    lw   $s0, 0x0000($t0)   # x = MEM[0]
    lw   $s1, 0x0004($t0)   # y = MEM[1]

multi:
    add  $t1, $t1, $s0      # t1 = t1 + x
    addi $t2, $t2, 0x0001   # cont = cont + 1
    bne  $t2, $s1, multi    # if(cont != y) multi

    add  $s2, $0, $t1       # k = x*y
    sw   $s2, 0x0008($t0)   # MEM[2] = k

.data
x: 7
y: 5
# fim
```

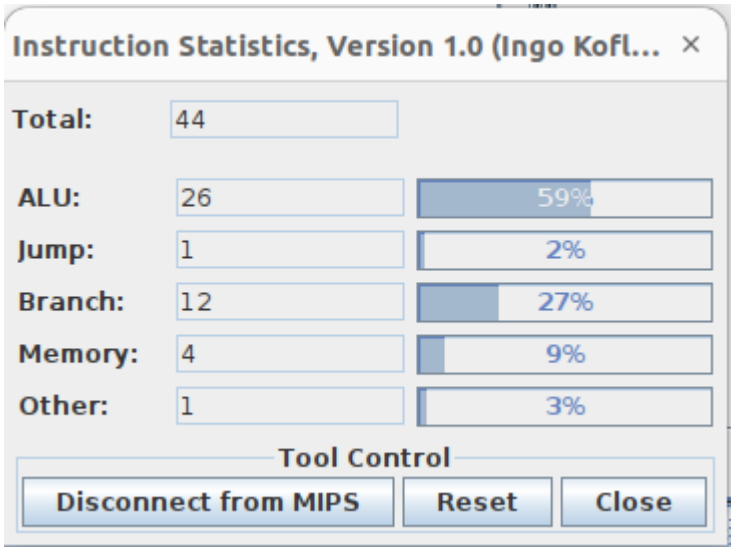

File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Text Segment
Bkpt Address Code Basic Source
0x00400000 0x3c081001 lui $8,4097 20: lui $t0, 0x1001 # t0 = 0x 1001 0000
0x00400004 0x8d100000 lw $16,0($8) 21: lw $s0, 0x0000($t0) # x = MEM[0]
0x00400008 0x8d110004 lw $17,4($8) 22: lw $s1, 0x0004($t0) # y = MEM[1]
0x0040000c 0x01304820 add $9,$9,$16 25: add $t1, $t1, $s0 # t1 = t1 + x
0x00400010 0x214a0001 addi $10,$10,1 26: addi $t2, $t2, 0x0001 # cont = cont + 1
0x00400014 0x1551fffd bne $10,$17,-3 27: bne $t2, $s1, multi # if(cont != y) multi
0x00400018 0x00099020 add $18,$0,$9 29: add $s2, $0, $t1 # k = x*y
0x0040001c 0xad120008 sw $18,8($8) 30: sw $s2, 0x0008($t0) # MEM[2] = k
Labels
Label Address
(global)
main 0x00400000
programa17.asm
multi 0x0040000c
x 0x10010000
y 0x10010004
Data Segment
Address Value (+0) Value (+4) Value (+8) Value (+c) Value (+10) Value (+14) Value (+18) Value (+1c)
0x10010000 7 5 35 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
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 35
$t2 10 5
$t3 11 0
$t4 12 0
$t5 13 0
$t6 14 0
$t7 15 0
$s0 16 7
$s1 17 -5
$s2 18 35
$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
Mars Messages Run I/O
Clear
-- program is finished running (dropped off bottom) --


Instruction Statistics, Version 1.0 (Ingo Kofl...
Total: 20
ALU: 12 60%
Jump: 0 0%
Branch: 5 25%
Memory: 3 15%
Other: 0 0%
Tool Control
Disconnect from MIPS
Reset
Close

Programa 18:



```
# Programa 18 (beq, bne, j)

# { x = MEM[0];y = MEM[1]; k = pow(x, y); MEM[2] = k; }

# Associacoes: x -> $s0; y -> $s1; k -> $s2

# inicio
.text
.globl main

main:
   lui  $t0, 0x1001         # t0 = 0x 1001 0000
   lw   $s0, 0x0000($t0)    # x = MEM[0]
   lw   $s1, 0x0004($t0)    # y = MEM[1]
   lw   $s2, 0x0008($t0)    # k = MEM[2]
   addi $t4, $0, 0x0001     # conty = 1
   add  $t5, $0, $s0        # t5 = x
   # Tratar execao
   slt  $t3, $s1, $0       # t3 = 1 if y < 0
   bne  $t3, $0, store      # if(y < 0) k = -1
   beq  $s1, $0, zero       # if(y == 0) k = 1
   beq  $s1, $t4, um        # if(y == 1) k = x

loop:
   add  $t1, $t1, $t5       # t1 = t1 + x
   addi $t2, $t2, 0x0001    # contx = contx + 1
   bne  $t2, $s0, loop      # if(contx != x) loop

   add  $t5, $0, $t1        # t5 = t1 (atual)
   addi $t2, $0, 0x0001     # t2 = t2 + 1
   addi $t4, $t4, 0x0001    # conty = conty + 1
   bne  $t4, $s1, loop      # if(conty != y) loop
   add  $s2, $0, $t5        # k = pow(x,y)
   j store                  # go to store

zero:
   addi $s2, $0, 0x0001     # k = 1
   j store                  # go to store
```



Parte 2 - Responda

1. Se tivermos 2 inteiros, cada um com 32 bits, quantos bits podemos esperar para o produto?
A.16
B.32
C.64
D.128

2. Quais os registradores que armazenam os resultados na multiplicação?
A.high e low
B.hi e lo
C.R0 e R1
D.$0 e $1

3. Qual a operação usada para multiplicar inteiros em comp. de dois?
A.mult
B.multu
C.multi
D.mutt

4. Qual instrução move os bits menos significativos da multiplicação para o reg. 8?
A.move $8,lo
B.mvlo $8,lo
C.mflo $8
D.addu $8,$0,lo

5. Se tivermos dois inteiros, cada um com 32 bits, quantos bits deveremos estar preparados para receber no quociente?
A.16
B.32
C.64
D.128

6. Após a instrução div, qual registrador possui o quociente?
A.lo
B.hi
C.high
D.$2

7. Qual a inst. Usada para dividir dois inteiros em comp. de dois?
A.dv
B.divide
C.divu
D.div

8. Faça um arithmetic shift right de dois no seguinte padrão de bits: 1001 1011

A.1110 0110
B.0010 0110
C.1100 1101
D.0011 0111

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

10. Qual sequencia de instruções avalia 3x+7, onde x é iniciado no reg. $8 e o resultado armazenado em $9?
A.
ori $3,$0,3
mult $8,$3
mflo $9
addi $9,$9,7

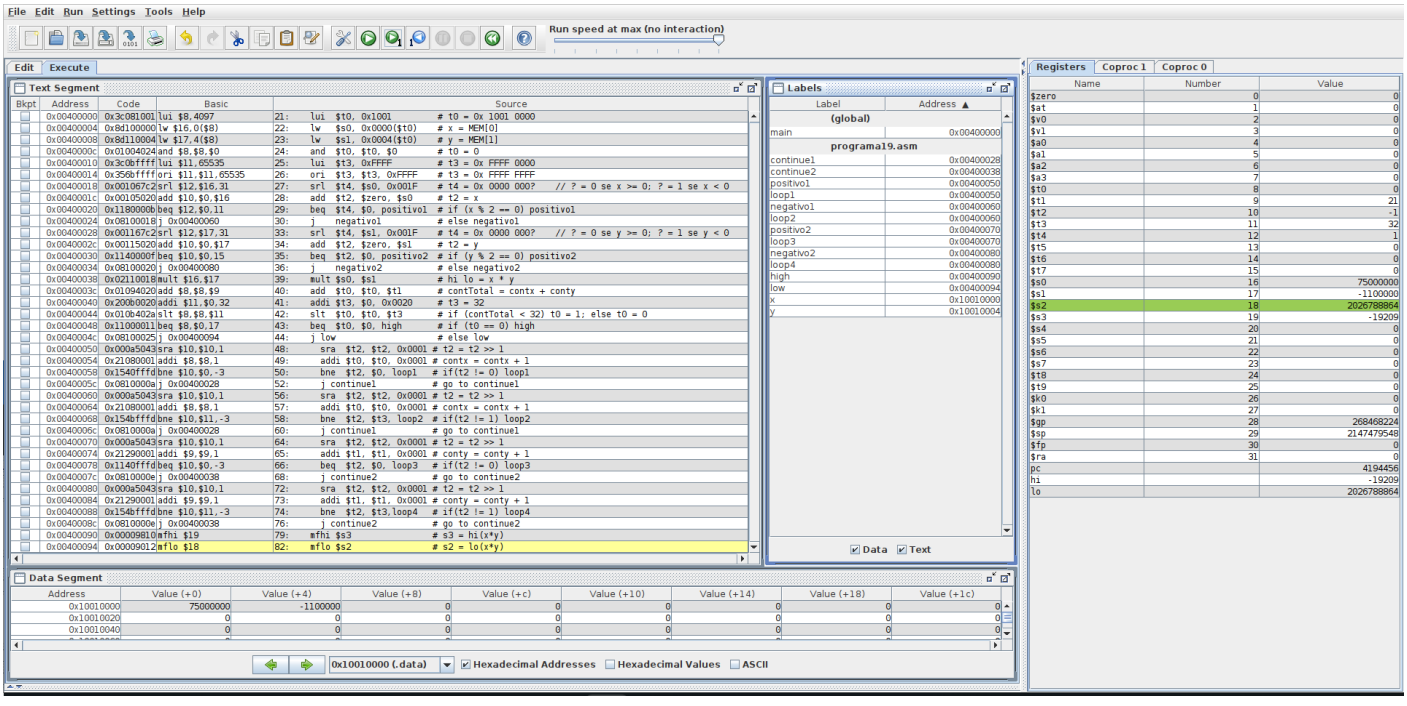
B.
ori $3,$0,3
mult $8,$3
addi $9,$8,7

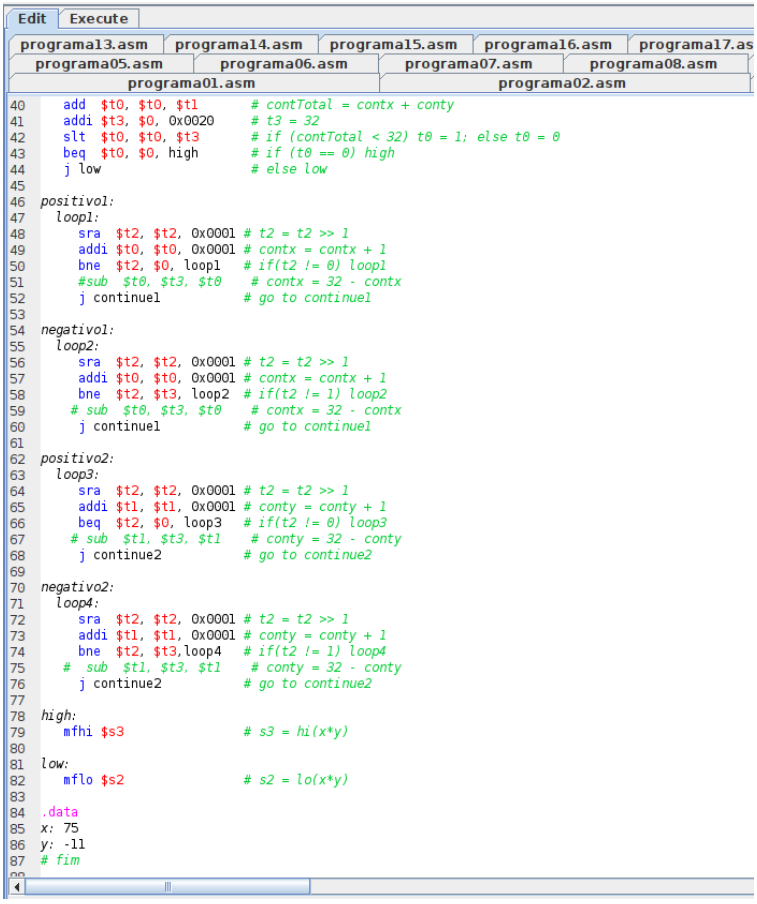
C.
ori $3,$0,3
mult $8,$3
mfhi $9
addi $9,$9,7

D.
mult $8,3
mflo $9
addi $9,$9,7

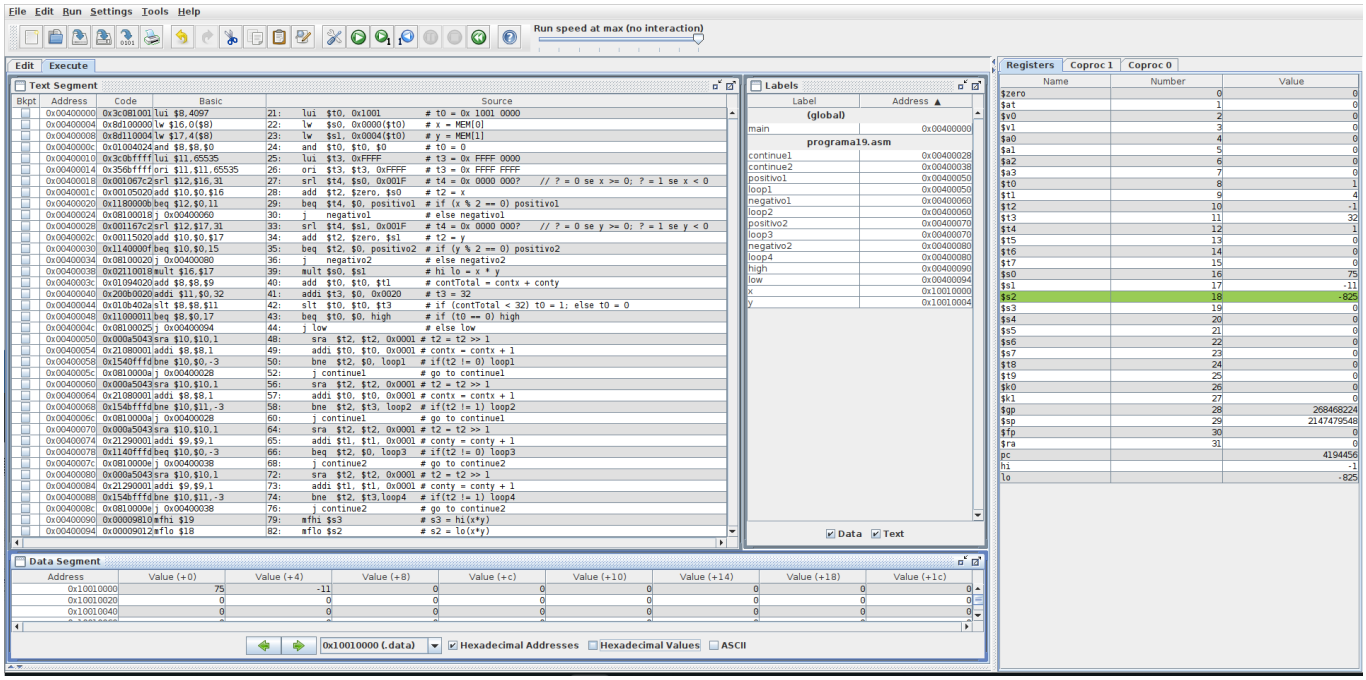
Programa 19:

```
# Programa 19

# {
#    x = MEM[0];
#    y = MEM[1];
#    t0 = bitsSignificantes de x
#    t1 = bitsSignificantes de y
#    if (t0 + t1 < 32) s2 = x*y
#    else s2, s3 = x*y
# }

# Associacoes
# x -> $s0
# y -> $s1

# inicio
.text
.globl main

main:
   lui  $t0, 0x1001          # t0 = 0x 1001 0000
   lw   $s0, 0x0000($t0)     # x = MEM[0]
   lw   $s1, 0x0004($t0)     # y = MEM[1]
   and  $t0, $t0, $0         # t0 = 0
   lui  $t3, 0xFFFF          # t3 = 0x FFFF 0000
   ori  $t3, $t3, 0xFFFF     # t3 = 0x FFFF FFFF
   srl  $t4, $s0, 0x001F     # t4 = 0x 0000 000?    // ? = 0 se x >= 0; ? = 1 se x < 0
   add  $t2, $zero, $s0      # t2 = x
   beq  $t4, $0, positivo1   # if (x % 2 == 0) positivo1
   j    negativo1            # else negativo1

continue1:
   srl  $t4, $s1, 0x001F     # t4 = 0x 0000 000?    // ? = 0 se y >= 0; ? = 1 se y < 0
   add  $t2, $zero, $s1      # t2 = y
   beq  $t2, $0, positivo2   # if (y % 2 == 0) positivo2
   j    negativo2            # else negativo2

continue2:
   mult $s0, $s1             # hi lo = x * y
   add  $t0, $t0, $t1        # contTotal = contx + conty
   addi $t3, $0, 0x0020      # t3 = 32
   slt  $t0, $t0, $t3        # if (contTotal < 32) t0 = 1; else t0 = 0
   beq  $t0, $0, high        # if (t0 == 0) high
   j low                     # else low

positivo1:
  loop1:
     sra  $t2, $t2, 0x0001 # t2 = t2 >> 1
     addi $t0, $t0, 0x0001 # contx = contx + 1
     bne  $t2, $0, loop1   # if(t2 != 0) loop1
     #sub  $t0, $t3, $t0    # contx = 32 - contx
     j continue1           # go to continue1

negativo1:
  loop2:
     sra  $t2, $t2, 0x0001 # t2 = t2 >> 1
     addi $t0, $t0, 0x0001 # contx = contx + 1
     bne  $t2, $t3, loop2  # if(t2 != 1) loop2
   # sub  $t0, $t3, $t0    # contx = 32 - contx
     j continue1           # go to continue1

positivo2:
  loop3:
     sra  $t2, $t2, 0x0001 # t2 = t2 >> 1
     addi $t1, $t1, 0x0001 # conty = conty + 1
     beq  $t2, $0, loop3   # if(t2 != 0) loop3
   # sub  $t1, $t3, $t1    # conty = 32 - conty
     j continue2           # go to continue2

negativo2:
  loop4:
     sra  $t2, $t2, 0x0001 # t2 = t2 >> 1
     addi $t1, $t1, 0x0001 # conty = conty + 1
     bne  $t2, $t3,loop4   # if(t2 != 1) loop4
   #  sub  $t1, $t3, $t1    # conty = 32 - conty
     j continue2           # go to continue2

high:
   mfhi $s3                # s3 = hi(x*y)

low:
   mflo $s2                # s2 = lo(x*y)

.data
x: 75000000
y: -1100000
# fim
```





```
Edit  Execute
programa13.asm  programa14.asm  programa15.asm  programa16.asm  programa17.as
programa05.asm  programa06.asm  programa07.asm  programa08.asm
programa01.asm  programa02.asm
40    add  $t0, $t0, $t1       # contTotal = contx + conty
41    addi $t3, $0, 0x0020     # t3 = 32
42    slt  $t0, $t0, $t3       # if (contTotal < 32) t0 = 1; else t0 = 0
43    beq  $t0, $0, high       # if (t0 == 0) high
44    j low                    # else low
45
46  positivo1:
47    loop1:
48       sra  $t2, $t2, 0x0001 # t2 = t2 >> 1
49       addi $t0, $t0, 0x0001 # contx = contx + 1
50       bne  $t2, $0, loop1   # if(t2 != 0) loop1
51       #sub  $t0, $t3, $t0    # contx = 32 - contx
52       j continue1           # go to continue1
53
54  negativo1:
55    loop2:
56       sra  $t2, $t2, 0x0001 # t2 = t2 >> 1
57       addi $t0, $t0, 0x0001 # contx = contx + 1
58       bne  $t2, $t3, loop2  # if(t2 != 1) loop2
59     # sub  $t0, $t3, $t0    # contx = 32 - contx
60       j continue1           # go to continue1
61
62  positivo2:
63    loop3:
64       sra  $t2, $t2, 0x0001 # t2 = t2 >> 1
65       addi $t1, $t1, 0x0001 # conty = conty + 1
66       beq  $t2, $0, loop3   # if(t2 != 0) loop3
67     # sub  $t1, $t3, $t1    # conty = 32 - conty
68       j continue2           # go to continue2
69
70  negativo2:
71    loop4:
72       sra  $t2, $t2, 0x0001 # t2 = t2 >> 1
73       addi $t1, $t1, 0x0001 # conty = conty + 1
74       bne  $t2, $t3,loop4   # if(t2 != 1) loop4
75     #  sub  $t1, $t3, $t1    # conty = 32 - conty
76       j continue2           # go to continue2
77
78  high:
79       mfhi $s3              # s3 = hi(x*y)
80
81  low:
82       mflo $s2              # s2 = lo(x*y)
83
84  .data
85  x: 75
86  y: -11
87  # fim
```



File Edit Run Settings Tools Help

Run speed at max (no interaction)

Edit Execute

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x3c081001	lui $8,4097	21: lui $t0, 0x1001 # t0 = 0x 1001 0000
	0x00400004	0x8d100000	lw $16,0($8)	22: lw $s0, 0x0000($t0) # x = MEM[0]
	0x00400008	0x8d110004	lw $17,4($8)	23: lw $s1, 0x0004($t0) # y = MEM[1]
	0x0040000c	0x01004024	and $8,$8,$0	24: and $t0, $t0, $0 # t0 = 0
	0x00400010	0x3c0bffff	lui $11,65535	25: lui $t3, 0xFFFF # t3 = 0x FFFF 0000
	0x00400014	0x356bffff	ori $11,$11,65535	26: ori $t3, $t3, 0xFFFF # t3 = 0x FFFF FFFF
	0x00400018	0x001067c2	srl $12,$16,31	27: srl $t4, $s0, 0x001F # t4 = 0x 0000 000? // ? = 0 se x >= 0; ? = 1 se x < 0
	0x0040001c	0x00105020	add $10,$0,$16	28: add $t2, $zero, $s0 # t2 = x
	0x00400020	0x11800000	beq $12,$0,11	29: beq $t4, $0, positivo1 # if (x % 2 == 0) positivo1
	0x00400024	0x08100018	j 0x00400060	30: j negativo1 # else negativo1
	0x00400028	0x001167c2	srl $12,$17,31	33: srl $t4, $s1, 0x001F # t4 = 0x 0000 000? // ? = 0 se y >= 0; ? = 1 se y < 0
	0x0040002c	0x00115020	add $10,$0,$17	34: add $t2, $zero, $s1 # t2 = y
	0x00400030	0x1140000f	beq $10,$0,15	35: beq $t2, $0, positivo2 # if (y % 2 == 0) positivo2
	0x00400034	0x08100020	j 0x00400080	36: j negativo2 # else negativo2
	0x00400038	0x02110018	mult $16,$17	39: mult $s0, $s1 # hi lo = x * y
	0x0040003c	0x01094020	add $8,$8,$9	40: add $t0, $t0, $t1 # contTotal = contx + conty
	0x00400040	0x200b0020	addi $11,$0,32	41: addi $t3, $0, 0x0020 # t3 = 32
	0x00400044	0x010b402a	slt $8,$8,$11	42: slt $t0, $t0, $t3 # if (contTotal < 32) t0 = 1; else t0 = 0
	0x00400048	0x11000011	beq $8,$0,17	43: beq $t0, $0, high # if (t0 == 0) high
	0x0040004c	0x08100025	j 0x00400094	44: j low # else low
	0x00400050	0x000a5043	sra $10,$10,1	48: sra $t2, $t2, 0x0001 # t2 = t2 >> 1
	0x00400054	0x21080001	addi $8,$8,1	49: addi $t0, $t0, 0x0001 # contx = contx + 1
	0x00400058	0x1540fffd	bne $10,$0,-3	50: bne $t2, $0, loop1 # if(t2 != 0) loop1
	0x0040005c	0x08100009	j 0x00400028	52: j continue1 # go to continue1
	0x00400060	0x000a5043	sra $10,$10,1	56: sra $t2, $t2, 0x0001 # t2 = t2 >> 1
	0x00400064	0x21080001	addi $8,$8,1	57: addi $t0, $t0, 0x0001 # contx = contx + 1
	0x00400068	0x154bfffd	bne $10,$11,-3	58: bne $t2, $t3, loop2 # if(t2 != 1) loop2
	0x0040006c	0x0810000a	j 0x00400028	60: j continue1 # go to continue1
	0x00400070	0x000a5043	sra $10,$10,1	64: sra $t2, $t2, 0x0001 # t2 = t2 >> 1
	0x00400074	0x21290001	addi $9,$9,1	65: addi $t1, $t1, 0x0001 # conty = conty + 1
	0x00400078	0x1140fffd	beq $10,$0,-3	66: beq $t2, $0, loop3 # if(t2 != 0) loop3
	0x0040007c	0x0810000e	j 0x00400038	68: j continue2 # go to continue2
	0x00400080	0x000a5043	sra $10,$10,1	72: sra $t2, $t2, 0x0001 # t2 = t2 >> 1
	0x00400084	0x21290001	addi $9,$9,1	73: addi $t1, $t1, 0x0001 # conty = conty + 1
	0x00400088	0x154bfffd	bne $10,$11,-3	74: bne $t2, $t3,loop4 # if(t2 != 1) loop4
	0x0040008c	0x0810000e	j 0x00400038	76: j continue2 # go to continue2
	0x00400090	0x00009810	mfhi $19	79: mfhi $s3 # s3 = hi(x*y)
	0x00400094	0x00009012	mflo $18	82: mflo $s2 # s2 = lo(x*y)

Labels

Label	Address
(global)	
main	0x00400000
programa19.asm	
continue1	0x00400028
continue2	0x00400038
positivo1	0x00400050
loop1	0x00400050
negativo1	0x00400060
loop2	0x00400060
positivo2	0x00400070
loop3	0x00400070
negativo2	0x00400080
loop4	0x00400080
high	0x00400090
low	0x00400094
x	0x10010000
y	0x10010004

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	75	-11	0	0	0	0	0	0
0x10010020	0	0	0	0	0	0	0	0
0x10010040	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	1
$t1	9	4
$t2	10	-1
$t3	11	32
$t4	12	1
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	75
$s1	17	-11
$s2	18	-825
$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		4194456
hi		-1
lo		-825

Programa 20:

Edit | Execute

programa13.asm | programa14.asm | programa15.asm | programa16.asm | programa17.asm | programa18.asm | programa19.asm | programa20.asm

programa05.asm | programa06.asm | programa07.asm | programa08.asm | programa09.asm | programa10.asm | programa11.asm | programa12.asm

programa01.asm | programa02.asm | programa03.asm | programa04.asm

```
1  # Programa 20
2
3  # {
4  #    x = MEM[0];
5  #    if(x % 2 == 0) y = x^4 + x^3 - 2*x^2;
6  #    else y = x^5 - x^3 + 1;
7  #    MEM[1] = y;
8  # }
9
10 # Associacoes
11 # x -> $s0; y -> $s1
12
13 # inicio
14 .text
15 .globl main
16
17 main:
18    lui  $t0, 0x1001          # t0 = 0x 1001 0000
19    lw   $s0, 0x0000($t0)     # x = MEM[0]
20    andi $t1, $s0, 0x0001     # t1 = 0x0000 000?   // ? = 0 se par; ? = 1 se impar
21    beq  $t1, $0,  par        # if(x % 2 == 0) go to par
22    mult $s0, $s0             # hi lo = x^2
23    mflo $t2                  # t2 = x^2
24    mult $t2, $s0             # hi lo = x^3
25    mflo $t3                  # t3 = x^3
26    mult $t2, $t3             # hi lo = x^5
27    mflo $t4                  # t4 = x^5
28    sub  $t4, $t4, $t3        # t4 = x^5 - x^3
29    addi $s1, $t4, 0x0001     # y = x^5 - x^3 + 1
30    j store                   # go to store
31
32 par:
33    mult $s0, $s0             # hi lo = x^2
34    mflo $t2                  # t2 = x^2
35    mult $t2, $s0             # hi lo = x^3
36    mflo $t3                  # t3 = x^3
37    mult $t2, $t2             # hi lo = x^4
38    mflo $t4                  # t4 = x^4
39    add  $t4, $t4, $t3        # t4 = x^4 + x^3
40    sub  $t4, $t4, $t2        # t4 = x^4 + x^3 - x^2
41    sub  $s1, $t4, $t2        # y = x^4 + x^3 - 2*x^2
42
43 store:
44    sw   $s1, 0x0004($t0)     # MEM[1] = y
45
46 .data
47 x: 2
48 # fim
```

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Edit | Execute

Text Segment

Bkpt	Address	Code	Basic		Source
	0x00400000	0x3c081001	lui $8,4097	18: lui $t0, 0x1001	# t0 = 0x 1001 0000
	0x00400004	0x8d100000	lw $16,0($8)	19: lw $s0, 0x0000($t0)	# x = MEM[0]
	0x00400008	0x32090001	andi $9,$16,1	20: andi $t1, $s0, 0x0001	# t1 = 0x0000 000? // ? = 0 se par; ? = 1 se impar
	0x0040000c	0x11200009	beq $9,$0,9	21: beq $t1, $0, par	# if(x % 2 == 0) go to par
	0x00400010	0x02100018	mult $16,$16	22: mult $s0, $s0	# hi lo = x^2
	0x00400014	0x00005012	mflo $10	23: mflo $t2	# t2 = x^2
	0x00400018	0x01500018	mult $10,$16	24: mult $t2, $s0	# hi lo = x^3
	0x0040001c	0x00005812	mflo $11	25: mflo $t3	# t3 = x^3
	0x00400020	0x014b0018	mult $10,$11	26: mult $t2, $t3	# hi lo = x^5
	0x00400024	0x00006012	mflo $12	27: mflo $t4	# t4 = x^5
	0x00400028	0x018b6022	sub $12,$12,$11	28: sub $t4, $t4, $t3	# t4 = x^5 - x^3
	0x0040002c	0x21910001	addi $17,$12,1	29: addi $s1, $t4, 0x0001	# y = x^5 - x^3 + 1
	0x00400030	0x08100016	j 0x00400058	30: j store	# go to store
	0x00400034	0x02100018	mult $16,$16	33: mult $s0, $s0	# hi lo = x^2
	0x00400038	0x00005012	mflo $10	34: mflo $t2	# t2 = x^2
	0x0040003c	0x01500018	mult $10,$16	35: mult $t2, $s0	# hi lo = x^3
	0x00400040	0x00005812	mflo $11	36: mflo $t3	# t3 = x^3
	0x00400044	0x014a0018	mult $10,$10	37: mult $t2, $t2	# hi lo = x^4
	0x00400048	0x00006012	mflo $12	38: mflo $t4	# t4 = x^4
	0x0040004c	0x018b6020	add $12,$12,$11	39: add $t4, $t4, $t3	# t4 = x^4 + x^3
	0x00400050	0x018a6022	sub $12,$12,$10	40: sub $t4, $t4, $t2	# t4 = x^4 + x^3 - x^2
	0x00400054	0x018a8822	sub $17,$12,$10	41: sub $s1, $t4, $t2	# y = x^4 + x^3 - 2*x^2
	0x00400058	0xad110004	sw $17,4($8)	44: sw $s1, 0x0004($t0)	# MEM[1] = y

Labels

Label	Address
(global)	
main	0x00400000
programa20.asm	
par	0x00400034
store	0x00400058
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	16	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

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	4
$t3	11	8
$t4	12	20
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	2
$s1	17	16
$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		4194396
hi		0
lo		16

Edit Execute

programa13.asm | programa14.asm | programa15.asm | programa16.asm | programa17.asm | programa18.asm | programa19.asm | programa20.asm

programa05.asm | programa06.asm | programa07.asm | programa08.asm | programa09.asm | programa10.asm | programa11.asm | programa12.asm

programa01.asm | programa02.asm | programa03.asm | programa04.asm

```
1  # Programa 20
2
3  # {
4  #    x = MEM[0];
5  #    if(x % 2 == 0) y = x^4 + x^3 - 2*x^2;
6  #    else y = x^5 - x^3 + 1;
7  #    MEM[1] = y;
8  # }
9
10 # Associacoes
11 # x -> $s0; y -> $s1
12
13 # inicio
14 .text
15 .globl main
16
17 main:
18    lui  $t0, 0x1001         # t0 = 0x 1001 0000
19    lw   $s0, 0x0000($t0)    # x = MEM[0]
20    andi $t1, $s0, 0x0001    # t1 = 0x0000 000?   // ? = 0 se par; ? = 1 se impar
21    beq  $t1, $0,  par       # if(x % 2 == 0) go to par
22    mult $s0, $s0            # hi lo = x^2
23    mflo $t2                 # t2 = x^2
24    mult $t2, $s0            # hi lo = x^3
25    mflo $t3                 # t3 = x^3
26    mult $t2, $t3            # hi lo = x^5
27    mflo $t4                 # t4 = x^5
28    sub  $t4, $t4, $t3       # t4 = x^5 - x^3
29    addi $s1, $t4, 0x0001    # y = x^5 - x^3 + 1
30    j store                  # go to store
31
32 par:
33    mult $s0, $s0            # hi lo = x^2
34    mflo $t2                 # t2 = x^2
35    mult $t2, $s0            # hi lo = x^3
36    mflo $t3                 # t3 = x^3
37    mult $t2, $t2            # hi lo = x^4
38    mflo $t4                 # t4 = x^4
39    add  $t4, $t4, $t3       # t4 = x^4 + x^3
40    sub  $t4, $t4, $t2       # t4 = x^4 + x^3 - x^2
41    sub  $s1, $t4, $t2       # y = x^4 + x^3 - 2*x^2
42
43 store:
44    sw   $s1, 0x0004($t0)    # MEM[1] = y
45
46 .data
47 x: 5
48 # fim
```

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Edit Execute

Text Segment

Bkpt	Address	Code	Basic		Source
	0x00400000	0x3c081001	lui $8,4097	18: lui $t0, 0x1001	# t0 = 0x 1001 0000
	0x00400004	0x8d100000	lw $16,0($8)	19: lw $s0, 0x0000($t0)	# x = MEM[0]
	0x00400008	0x32090001	andi $9,$16,1	20: andi $t1, $s0, 0x0001	# t1 = 0x0000 000? // ? = 0 se par; ? = 1 se impar
	0x0040000c	0x11200009	beq $9,$0,9	21: beq $t1, $0, par	# if(x % 2 == 0) go to par
	0x00400010	0x02100018	mult $16,$16	22: mult $s0, $s0	# hi lo = x^2
	0x00400014	0x00005012	mflo $10	23: mflo $t2	# t2 = x^2
	0x00400018	0x01500018	mult $10,$16	24: mult $t2, $s0	# hi lo = x^3
	0x0040001c	0x00005812	mflo $11	25: mflo $t3	# t3 = x^3
	0x00400020	0x014b0018	mult $10,$11	26: mult $t2, $t3	# hi lo = x^5
	0x00400024	0x00006012	mflo $12	27: mflo $t4	# t4 = x^5
	0x00400028	0x018b6022	sub $12,$12,$11	28: sub $t4, $t4, $t3	# t4 = x^5 - x^3
	0x0040002c	0x21910001	addi $17,$12,1	29: addi $s1, $t4, 0x0001	# y = x^5 - x^3 + 1
	0x00400030	0x08100016	j 0x00400058	30: j store	# go to store
	0x00400034	0x02100018	mult $16,$16	33: mult $s0, $s0	# hi lo = x^2
	0x00400038	0x00005012	mflo $10	34: mflo $t2	# t2 = x^2
	0x0040003c	0x01500018	mult $10,$16	35: mult $t2, $s0	# hi lo = x^3
	0x00400040	0x00005812	mflo $11	36: mflo $t3	# t3 = x^3
	0x00400044	0x014a0018	mult $10,$10	37: mult $t2, $t2	# hi lo = x^4
	0x00400048	0x00006012	mflo $12	38: mflo $t4	# t4 = x^4
	0x0040004c	0x018b6020	add $12,$12,$11	39: add $t4, $t4, $t3	# t4 = x^4 + x^3
	0x00400050	0x018a6022	sub $12,$12,$10	40: sub $t4, $t4, $t2	# t4 = x^4 + x^3 - x^2
	0x00400054	0x018a8822	sub $17,$12,$10	41: sub $s1, $t4, $t2	# y = x^4 + x^3 - 2*x^2
	0x00400058	0xad110004	sw $17,4($8)	44: sw $s1, 0x0004($t0)	# MEM[1] = y

Labels

Label	Address
(global)	
main	0x00400000
programa20.asm	
par	0x00400034
store	0x00400058
x	0x10010000

Data Text

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	5	3001	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

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	25
$t3	11	125
$t4	12	3000
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	5
$s1	17	3001
$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		4194396
hi		0
lo		3125

Programa 21:

Edit Execute

programa14.asm | programa15.asm | programa16.asm | programa17.asm | programa18.asm | programa19.asm | programa20.asm | programa21.asm

programa06.asm | programa07.asm | programa08.asm | programa09.asm | programa10.asm | programa11.asm | programa12.asm | programa13.asm

programa01.asm | programa02.asm | programa03.asm | programa04.asm | programa05.asm

```
1  # Associacoes
2  # x -> $s0; y -> $s1
3
4  # inicio
5  .text
6  .globl main
7
8  main:
9     lui  $t0, 0x1001         # t0 = 0x 1001 0000
10    lw   $s0, 0x0000($t0)    # x = MEM[0]
11    beq  $s0, $0, y2         # if(x == 0) y2
12    srl  $t2, $s0, 0x001F    # t2 = 0x 0000 000?    // ? = 0 se x > 0; ? = 1 se x < 0
13    beq  $t2, $0, y1         # if(x > 0) y1
14
15 y2:
16    mult $s0, $s0            # hi lo = x^2
17    mflo $t1                 # t1 = x^2
18    mult $t1, $t1            # hi lo = x^4
19    mflo $t1                 # t1 = x^4
20    sub $s1, $t1, 0x0001     # y = x^4-1
21    j store                  # go to store
22
23 y1:
24    mult $s0, $s0            # hi lo = x^2
25    mflo $t1                 # t1 = x^2
26    mult $t1, $s0            # hi lo = x^3
27    mflo $t1                 # t1 = x^3
28    addi $s1, $t1, 0x0001    # y = x^3+1
29
30 store:
31    sw   $s1, 0x0004($t0)    # MEM[1] = y
32
33 .data
34 x: -2
35 # fim
36
37
38
39
40
41
42
43
44
45
46
47
```

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Edit Execute

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x3c081001	lui $8,4097	18: lui $t0, 0x1001 # t0 = 0x 1001 0000
	0x00400004	0x8d100000	lw $16,0($8)	19: lw $s0, 0x0000($t0) # x = MEM[0]
	0x00400008	0x12000002	beq $16,$0,2	20: beq $s0, $0, y2 # if(x == 0) y2
	0x0040000c	0x001057c2	srl $10,$16,31	21: srl $t2, $s0, 0x001F # t2 = 0x 0000 000? // ? = 0 se x > 0; ? = 1 se x < 0
	0x00400010	0x11400007	beq $10,$0,7	22: beq $t2, $0, y1 # if(x > 0) y1
	0x00400014	0x02100018	mult $16,$16	25: mult $s0, $s0 # hi lo = x^2
	0x00400018	0x00004812	mflo $9	26: mflo $t1 # t1 = x^2
	0x0040001c	0x01290018	mult $9,$9	27: mult $t1, $t1 # hi lo = x^4
	0x00400020	0x00004812	mflo $9	28: mflo $t1 # t1 = x^4
	0x00400024	0x20010001	addi $1,$0,1	29: sub $s1, $t1, 0x0001 # y = x^4-1
	0x00400028	0x01218822	sub $17,$9,$1	
	0x0040002c	0x08100011	j 0x00400044	30: j store # go to store
	0x00400030	0x02100018	mult $16,$16	33: mult $s0, $s0 # hi lo = x^2
	0x00400034	0x00004812	mflo $9	34: mflo $t1 # t1 = x^2
	0x00400038	0x01300018	mult $9,$16	35: mult $t1, $s0 # hi lo = x^3
	0x0040003c	0x00004812	mflo $9	36: mflo $t1 # t1 = x^3
	0x00400040	0x21310001	addi $17,$9,1	37: addi $s1, $t1, 0x0001 # y = x^3+1
	0x00400044	0xad110004	sw $17,4($8)	40: sw $s1, 0x0004($t0) # MEM[1] = y

Labels

Label	Address ▲
(global)	
main	0x00400000
programa21.asm	
y2	0x00400014
y1	0x00400030
store	0x00400044
x	0x10010000

Data Text

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0
$at	1	1
$v0	2	0
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	268500992
$t1	9	16
$t2	10	1
$t3	11	0
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	-2
$s1	17	15
$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		4194376
hi		0
lo		16

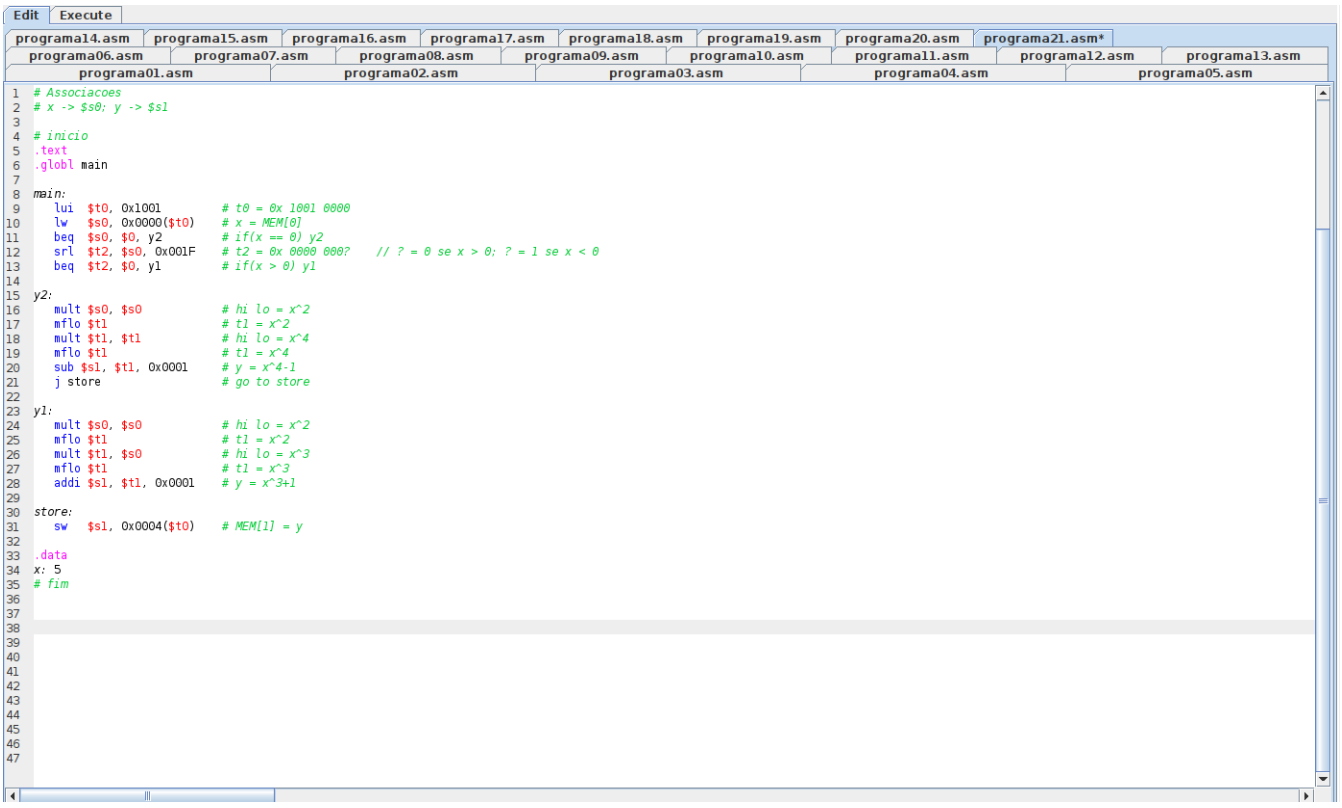
Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	-2	15	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

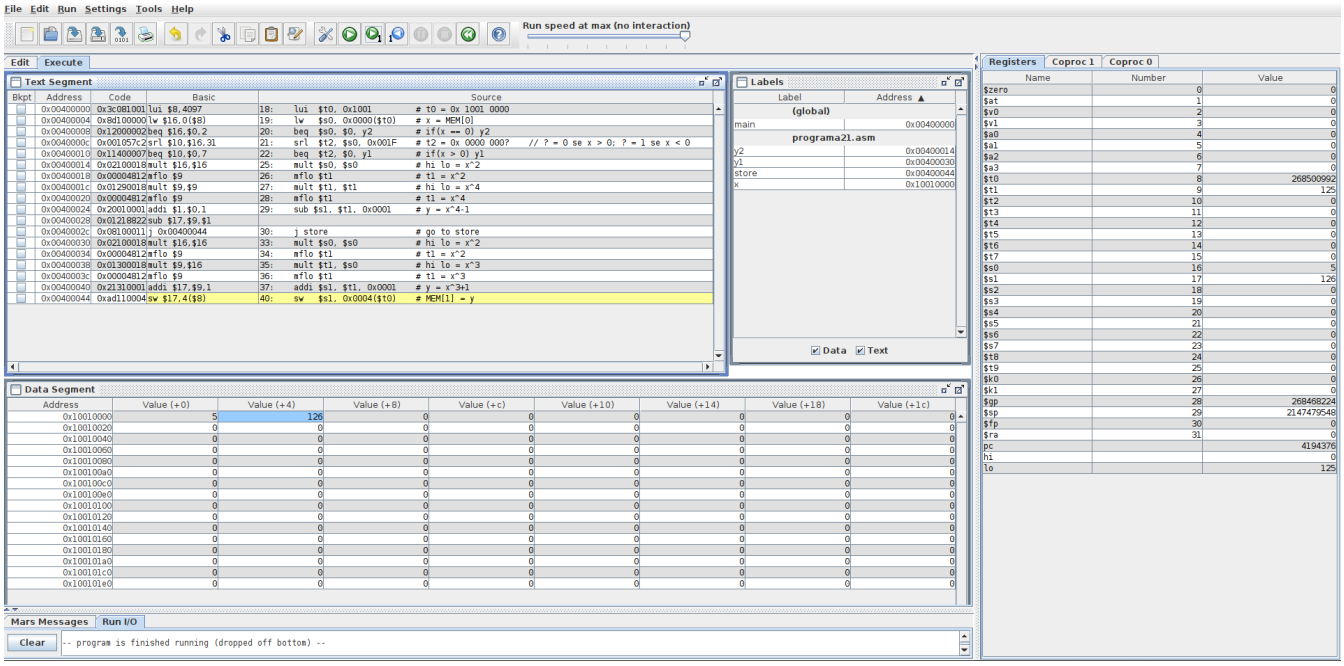
Mars Messages | Run I/O

Clear

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



```
# Associacoes
# x -> $s0; y -> $s1

# inicio
.text
.globl main

main:
    lui  $t0, 0x1001          # t0 = 0x 1001 0000
    lw   $s0, 0x0000($t0)     # x = MEM[0]
    beq  $s0, $0, y2          # if(x == 0) y2
    srl  $t2, $s0, 0x001F     # t2 = 0x 0000 000?    // ? = 0 se x > 0; ? = 1 se x < 0
    beq  $t2, $0, y1          # if(x > 0) y1

y2:
    mult $s0, $s0             # hi lo = x^2
    mflo $t1                  # t1 = x^2
    mult $t1, $t1             # hi lo = x^4
    mflo $t1                  # t1 = x^4
    sub $s1, $t1, 0x0001      # y = x^4-1
    j store                   # go to store

y1:
    mult $s0, $s0             # hi lo = x^2
    mflo $t1                  # t1 = x^2
    mult $t1, $s0             # hi lo = x^3
    mflo $t1                  # t1 = x^3
    addi $s1, $t1, 0x0001     # y = x^3+1

store:
    sw   $s1, 0x0004($t0)     # MEM[1] = y

.data
x: 5
# fim
```



File Edit Run Settings Tools Help

Run speed at max (no interaction)

Edit | Execute

Text Segment

Bkpt	Address	Code	Basic		Source
	0x00400000	0x3c081001	lui $8,4097	18: lui $t0, 0x1001	# t0 = 0x 1001 0000
	0x00400004	0x8d100000	lw $16,0($8)	19: lw $s0, 0x0000($t0)	# x = MEM[0]
	0x00400008	0x12000002	beq $16,$0,2	20: beq $s0, $0, y2	# if(x == 0) y2
	0x0040000c	0x001057c2	srl $10,$16,31	21: srl $t2, $s0, 0x001F	# t2 = 0x 0000 000? // ? = 0 se x > 0; ? = 1 se x < 0
	0x00400010	0x11400007	beq $10,$0,7	22: beq $t2, $0, y1	# if(x > 0) y1
	0x00400014	0x02100018	mult $16,$16	25: mult $s0, $s0	# hi lo = x^2
	0x00400018	0x00004812	mflo $9	26: mflo $t1	# t1 = x^2
	0x0040001c	0x01290018	mult $9,$9	27: mult $t1, $t1	# hi lo = x^4
	0x00400020	0x00004812	mflo $9	28: mflo $t1	# t1 = x^4
	0x00400024	0x20010001	addi $1,$0,1	29: sub $s1, $t1, 0x0001	# y = x^4-1
	0x00400028	0x01218822	sub $17,$9,$1		
	0x0040002c	0x08100011	j 0x00400044	30: j store	# go to store
	0x00400030	0x02100018	mult $16,$16	33: mult $s0, $s0	# hi lo = x^2
	0x00400034	0x00004812	mflo $9	34: mflo $t1	# t1 = x^2
	0x00400038	0x01300018	mult $9,$16	35: mult $t1, $s0	# hi lo = x^3
	0x0040003c	0x00004812	mflo $9	36: mflo $t1	# t1 = x^3
	0x00400040	0x21310001	addi $17,$9,1	37: addi $s1, $t1, 0x0001	# y = x^3+1
	0x00400044	0xad110004	sw $17,4($8)	40: sw $s1, 0x0004($t0)	# MEM[1] = y

Labels

Label	Address
(global)	
main	0x00400000
programa21.asm	
y2	0x00400014
y1	0x00400030
store	0x00400044
x	0x10010000

Data Text

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	5	126	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

Mars Messages | Run I/O

Clear

-- 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	268500992
$t1	9	125
$t2	10	0
$t3	11	0
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	5
$s1	17	126
$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		4194376
hi		0
lo		125