

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. 0xFFFFFFF

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 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:

The screenshot shows a MIPS assembly debugger interface with the following sections:

- Code View:** Displays the assembly code for `program01.asm`. The code initializes variables `a`, `b`, `c`, and `d`, calculates intermediate values `t0` through `t5`, and finally computes `x` and `y`.
- Registers View:** Shows the register values for the current program counter (PC), including \$zero through \$t10, \$sp, \$gp, \$tp, \$ra, \$s1, and \$lo.
- Data Segment View:** Shows the memory dump starting at address `0x10010000`, with columns for Address, Value (+0) through Value (+18).
- Labels View:** Lists global labels and their addresses, such as `main` at `0x00400000`.
- Text Segment View:** Lists breakpoints (Bkpt), address, code, basic, and source for each instruction.
- Mars Messages View:** Displays a message indicating the program has finished running.

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

```

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Text Segment

| Bkpt | Address | Code | Basic | Source |
|------------|------------|--------------------------|-------|----------------|
| 0x00400000 | 0x34100001 | ori \$s0, \$zero, 0x0001 | 17: | # x = 1 |
| 0x00400004 | 0x02104020 | add \$t0, \$s0, \$s0 | 18: | # t0 = 2*x |
| 0x00400008 | 0x01084020 | add \$t0, \$t0, \$t0 | 19: | # t0 = 4*x |
| 0x00400012 | 0x01104020 | add \$t0, \$t0, \$s0 | 20: | # t0 = 5*x |
| 0x00400016 | 0x2111000f | addi \$s1, \$t0, 0x000F | 21: | # y = 5*x + 15 |

Labels

| Label | Address |
|----------|------------|
| (global) | 0x00400000 |
| main | 0x00400000 |

Registers

| 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 |
| \$t8 | 16 | 1 |
| \$s1 | 17 | 20 |
| \$s2 | 18 | 0 |
| \$s3 | 19 | 0 |
| \$s4 | 20 | 0 |
| \$s5 | 21 | 0 |
| \$s6 | 22 | 0 |
| \$s7 | 23 | 0 |
| \$t9 | 24 | 0 |
| \$t10 | 25 | 0 |
| \$t11 | 26 | 0 |
| \$t12 | 27 | 0 |
| \$sp | 28 | 268468224 |
| \$fp | 29 | 2147479548 |
| \$ra | 30 | 0 |
| \$pc | 31 | 4194324 |
| 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 |
| 0x10010004 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010040 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010050 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010080 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010090 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x100100a0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x100100d0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x100100e0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x100100f0 | 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 Decimal Addresses Hexadecimal Values ASCII

Mars Messages Run I/O

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

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

```

Screenshot of the Mars Simulation Environment showing assembly code, registers, and memory dump.

Registers:

| 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 | 625 |
| \$t1 | 9 | 268 |
| \$t2 | 10 | 0 |
| \$t3 | 11 | 0 |
| \$t4 | 12 | 0 |
| \$t5 | 13 | 0 |
| \$t6 | 14 | 0 |
| \$t7 | 15 | 0 |
| \$t8 | 16 | 0 |
| \$t9 | 17 | 4 |
| \$s2 | 18 | 125 |
| \$s3 | 19 | 0 |
| \$s4 | 20 | 0 |
| \$s5 | 21 | 0 |
| \$s6 | 22 | 0 |
| \$s7 | 23 | 0 |
| \$s8 | 24 | 0 |
| \$s9 | 25 | 0 |
| \$k0 | 26 | 0 |
| \$k1 | 27 | 0 |
| \$gp | 28 | 268468224 |
| \$sp | 29 | 2147479548 |
| \$fp | 30 | 0 |
| \$ra | 31 | 4194380 |
| pc | | 4194380 |
| hi | | 0 |
| lo | | 0 |

Text Segment:

```

Bkpt Address Code Basic Source
0x00400000 0x34100003 ori $16,$0,3 19: ori $0,$zero,0x0003 # x = 3
0x00400004 0x34110040 ori $17,$0,4 20: ori $1,$zero,0x0004 # y = 4
0x00400005 0x02104020 add $8,$16,$16 21: add $10,$0,$0 # t0 = 2^x
0x00400006 0x02104020 add $8,$18,$8 22: add $10,$10,$0 # t0 = 64*x
0x00400007 0x02104020 add $8,$18,$16 23: add $10,$10,$0 # t0 = 15*x
0x00400008 0x02104020 add $8,$17,$17 24: add $10,$10,$0 # t0 = 67*y
0x00400009 0x02104020 add $8,$17,$17 25: sub $10,$10,$0 # t0 = 15*x
0x0040000a 0x02104020 add $8,$17,$17 26: add $11,$11,$1 # t1 = 2^y
0x0040000b 0x02104020 add $8,$17,$17 27: add $11,$11,$1 # t1 = 4^y
0x0040000c 0x02104020 add $8,$17,$17 28: add $11,$11,$1 # t1 = 64^y
0x0040000d 0x02104020 add $8,$17,$17 29: add $11,$11,$1 # t2 = 16^y
0x0040000e 0x02104020 add $8,$17,$17 30: add $11,$11,$1 # t2 = 32^y
0x0040000f 0x02104020 add $8,$17,$17 31: add $11,$11,$1 # t2 = 64^y
0x00400010 0x02104020 add $8,$17,$17 32: add $11,$11,$1 # t2 = 128^y
0x00400011 0x02104020 add $8,$17,$17 33: add $11,$11,$1 # t2 = 256^y
0x00400012 0x02104020 add $8,$17,$17 34: add $11,$11,$1 # t2 = 67*y
0x00400013 0x02104020 add $8,$17,$17 35: add $10,$10,$1 # t0 = 15*x + 67*y
0x00400014 0x02104020 add $8,$17,$17 36: add $10,$10,$0 # t0 = 2^(15*x + 67*y)
0x00400015 0x02104020 add $8,$17,$17 37: add $12,$12,$0 # z = 4^(15*x + 67*y)
0x00400016 0x02104020 add $8,$17,$17

```

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 |

Mars Messages:

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

Programa 4:

Screenshot of the Mars Simulation Environment showing assembly code, registers, and memory dump.

Registers:

| 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 |
| \$t8 | 16 | 0 |
| \$t9 | 17 | 4 |
| \$s2 | 18 | 125 |
| \$s3 | 19 | 0 |
| \$s4 | 20 | 0 |
| \$s5 | 21 | 0 |
| \$s6 | 22 | 0 |
| \$s7 | 23 | 0 |
| \$s8 | 24 | 0 |
| \$s9 | 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 |

Text Segment:

```

1 # Programa 04 (sll, srl e sra)
2
3 # {
4 #   x = 3;
5 #   y = 4;
6 #   z = (15*x + 67*y)*4;
7 # }
8
9 # Associações
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   sll $t0,$s0,0x0004 # t0 = 16^x
22   sub $t0,$t0,$s0 # t0 = 15*x
23   sll $t1,$s1,0x0006 # t1 = 64^y
24   add $t1,$t1,$s1 # t1 = 65*y
25   add $t1,$t1,$s1 # t1 = 66*y
26   add $t1,$t1,$s1 # t1 = 67*y
27   add $t0,$t0,$t1 # t0 = 15*x + 67*y
28   sll $s2,$t0,0x0002 # z = (15*x + 67*y)*4
29 # fim
30

```

Screenshot of the Mars Simulation Environment showing assembly code, registers, and memory dump.

Registers:

| 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 |
| \$t8 | 16 | 0 |
| \$t9 | 17 | 4 |
| \$s2 | 18 | 125 |
| \$s3 | 19 | 0 |
| \$s4 | 20 | 0 |
| \$s5 | 21 | 0 |
| \$s6 | 22 | 0 |
| \$s7 | 23 | 0 |
| \$s8 | 24 | 0 |
| \$s9 | 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 |

Text Segment:

```

Bkpt Address Code Basic Source
0x00400000 0x34100003 ori $16,$0,3 19: ori $0,$zero,0x0003 # x = 3
0x00400004 0x34110040 ori $17,$0,4 20: ori $1,$zero,0x0004 # y = 4
0x00400005 0x02104020 add $8,$16,$16 21: add $10,$0,$0 # t0 = 2^x
0x00400006 0x02104020 add $8,$18,$8 22: add $10,$10,$0 # t0 = 64*x
0x00400007 0x02104020 add $8,$18,$16 23: add $10,$10,$0 # t0 = 15*x
0x00400008 0x02104020 add $8,$17,$17 24: add $10,$10,$0 # t0 = 67*y
0x00400009 0x02104020 add $8,$17,$17 25: sub $10,$10,$0 # t0 = 15*x
0x0040000a 0x02104020 add $8,$17,$17 26: add $11,$11,$1 # t1 = 2^y
0x0040000b 0x02104020 add $8,$17,$17 27: add $11,$11,$1 # t1 = 4^y
0x0040000c 0x02104020 add $8,$17,$17 28: add $11,$11,$1 # t1 = 64^y
0x0040000d 0x02104020 add $8,$17,$17 29: add $11,$11,$1 # t2 = 16^y
0x0040000e 0x02104020 add $8,$17,$17 30: add $11,$11,$1 # t2 = 32^y
0x0040000f 0x02104020 add $8,$17,$17 31: add $11,$11,$1 # t2 = 64^y
0x00400010 0x02104020 add $8,$17,$17 32: add $11,$11,$1 # t2 = 128^y
0x00400011 0x02104020 add $8,$17,$17 33: add $11,$11,$1 # t2 = 256^y
0x00400012 0x02104020 add $8,$17,$17 34: add $11,$11,$1 # t2 = 67*y
0x00400013 0x02104020 add $8,$17,$17 35: add $10,$10,$1 # t0 = 15*x + 67*y
0x00400014 0x02104020 add $8,$17,$17 36: add $10,$10,$0 # t0 = 2^(15*x + 67*y)
0x00400015 0x02104020 add $8,$17,$17 37: add $12,$12,$0 # z = 4^(15*x + 67*y)
0x00400016 0x02104020 add $8,$17,$17

```

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 |

Mars Messages:

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

Programa 5:

The screenshot shows the QEMU debugger interface with the following panes:

- Assembly Editor:** Displays the assembly code for Program 5. The code initializes variables \$t0, \$t1, and \$t2, performs arithmetic operations (\$t0 = x + y), and prints the result.
- Registers:** Shows the state of general-purpose registers (\$zero to \$t7) and floating-point registers (\$f0 to \$f2). \$t2 is highlighted in green.
- Data Segment:** Displays memory dump from address 0x10010000 to 0x10010400. The value at \$t2 (0x20000) is highlighted in green.
- Mars Messages:** Shows the message: "program is finished running (dropped off bottom)"

Programa 6:

The screenshot shows the QEMU debugger interface with the following panes:

- Assembly Editor:** Displays the assembly code for Program 6. The code initializes variables \$t0, \$t1, and \$t2, performs arithmetic operations (\$t0 = 4*y), and prints the result.
- Registers:** Shows the state of general-purpose registers (\$zero to \$t7) and floating-point registers (\$f0 to \$f2). \$t2 is highlighted in green.
- Data Segment:** Displays memory dump from address 0x10010000 to 0x10010400. The value at \$t2 (0x2147483647) is highlighted in green.
- Mars Messages:** Shows the message: "program is finished running (dropped off bottom)"

Screenshot of the MARS assembly editor showing the assembly code for Programa 7:

```

    .text
    .globl main
main:
    ori $8, $0, 0x0001      # s8 = 0x 0000 0001
    sll $8,$8, 0x001F        # s8 = 0x 0000 0000 // s8 = 1000 ... 0000
    sra $8,$8, 0x001F        # s8 = 0x FFFF FFFF // s8 = 1111 ... 1111

```

The Registers window shows the following values:

| 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 |
| \$t1 | 17 | 5000000 |
| \$s2 | 18 | 2147483647 |
| \$s3 | 19 | 0 |
| \$s4 | 20 | 0 |
| \$s5 | 21 | 0 |
| \$s6 | 22 | 0 |
| \$s7 | 23 | 0 |
| \$t8 | 24 | 0 |
| \$t9 | 25 | 0 |
| \$t0 | 26 | 0 |
| \$t1 | 27 | 0 |
| \$gp | 28 | 269468224 |
| \$sp | 29 | 2147479548 |
| \$fp | 30 | 0 |
| \$ra | 31 | 0 |
| pc | | 419436 |
| hi | | 0 |
| lo | | 0 |

Mars Messages: -- program is finished running (dropped off bottom) --

Programa 7:

Screenshot of the MARS assembly editor showing the assembly code for Programa 7:

```

    .text
    .globl main
main:
    ori $8, $0, 0x0001      # s8 = 0x 0000 0001
    sll $8,$8, 0x001F        # s8 = 0x 0000 0000 // s8 = 1000 ... 0000
    sra $8,$8, 0x001F        # s8 = 0x FFFF FFFF // s8 = 1111 ... 1111

```

Screenshot of the MARS assembly editor showing the assembly code for Programa 7:

```

    .text
    .globl main
main:
    ori $8, $0, 0x0001      # s8 = 0x 0000 0001
    sll $8,$8, 0x001F        # s8 = 0x 0000 0000 // s8 = 1000 ... 0000
    sra $8,$8, 0x001F        # s8 = 0x FFFF FFFF // s8 = 1111 ... 1111

```

The Registers window shows the following values:

| 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 |
| \$t1 | 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 |
| \$t0 | 26 | 0x00000000 |
| \$t1 | 27 | 0x00000000 |
| \$gp | 28 | 0x10000000 |
| \$sp | 29 | 0x7fffffc |
| \$fp | 30 | 0x00000000 |
| \$ra | 31 | 0x00000000 |
| pc | | 0x00000000 |
| hi | | 0x00000000 |
| lo | | 0x00000000 |

Mars Messages: -- program is finished running (dropped off bottom) --

Programa 8:

```

Edit Execute
programa01.asm programa02.asm programa03.asm programa04.asm programa05.asm programa06.asm programa07.asm programa08.asm
1 # Programa 08 (logicas (or, ori, and, xor, xori) e instrucoes de deslocamento (sll, srl e sra))
2
3 #
4 #   $8 = 0x12345678
5 # ... // nao nessa ordem
6 #   $9 = 0x12
7 #   $10 = 0x34
8 #   $11 = 0x56
9 #   $12 = 0x78
10 #
11
12 # Associacoes
13
14 # inicio
15 .text
16 .globl main
17
18 main:
19     ori $13, $0, 0x1234    # t5 = 0x 0000 1234
20     sll $13, $13, 0x0010    # t5 = 0x 1234 0000
21     ori $8, $13, 0x5678    # $8 = 0x 1234 5678
22     andi $12, $8, 0x00FF    # $12 = 0x 0000 0078
23     srl $13, $8, 0x0008    # t5 = 0x 0012 3456
24     andi $11, $13, 0x00FF    # $11 = 0x 0000 0056
25     srl $13, $8, 0x0010    # t5 = 0x 0000 1234
26     andi $10, $13, 0x00FF    # $10 = 0x 0000 0034
27     srl $9, $8, 0x0018    # $9 = 0x 0000 0012
28 # fim
29

```

The screenshot shows the Mars 32-bit debugger interface with several windows open:

- Registers:** Shows the CPU register state. The \$11 register is highlighted in green.
- Labels:** Shows the global label `main` at address `0x00400000`.
- Text Segment:** Shows the assembly code for the `main` function.
- Data Segment:** Shows the memory dump starting at address `0x10010000`, with columns for Address, Value (+0), Value (+4), Value (+8), Value (+c), Value (+10), Value (+14), Value (+18), and Value (+1c).
- Mars Messages:** Shows a message indicating the program has finished running.

Programa 9:

```

Edit Execute
programa01.asm programa02.asm programa03.asm programa04.asm programa05.asm programa06.asm programa07.asm programa08.asm programa09.asm
1 # Programa 09 (lw e sw)
2
3 #
4 #   x1 = MEM[0];
5 #   x2 = MEM[1];
6 #   x3 = MEM[2];
7 #   x4 = MEM[3];
8 #   MEM[4] = x1 + x2 + x3 + x4;
9 #
10
11 # Associacoes
12 # x1 -> $s1
13 # x2 -> $s2
14 # x3 -> $s3
15 # x4 -> $s4
16 # soma -> $s5
17
18 # inicio
19 .text
20 .globl main
21
22 main:
23     ori $t0, $0, 0x1001    # t0 = 0x 0000 1001
24     sll $t0, $t0, 0x0010    # t0 = 0x 1001 0000
25     lw $s1, 0x0000($t0)    # x1 = MEM[0]
26     lw $s2, 0x0004($t0)    # x2 = MEM[1]
27     lw $s3, 0x0008($t0)    # x3 = MEM[2]
28     lw $s4, 0x000C($t0)    # x4 = MEM[3]
29     add $t1, $s1, $s2      # t1 = x1 + x2
30     add $t1, $t1, $s3      # t1 = x1 + x2 + x3
31     add $s5, $t1, $s4      # soma = x1 + x2 + x3 + x4
32     sw $s5, 0x0010($t0)    # MEM[4] = soma
33
34 .data
35 x1: .word 15
36 x2: .word 25
37 x3: .word 13
38 x4: .word 17
39 soma: .word -1
40 # fim
41

```

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Edit Execute

Text Segment

| Blkpt | Address | Code | Basic | Source |
|------------|------------|---------------------------|----------------------------|--------|
| 0x00400000 | 0x34081000 | ori \$t0, \$0, 0x1001 | # t0 = 0x 1001 0000 | |
| 0x00400004 | 0x00084400 | sll \$t1, \$t0, 0x0000010 | # t1 = 0x 1001 0000 | |
| 0x00400008 | 0x8d100000 | lw \$t2, 0x00000000(\$t0) | # x1 = MEM[0] | |
| 0x0040000c | 0x8d120004 | lw \$t3, 0x00000004(\$t0) | # x2 = MEM[1] | |
| 0x00400010 | 0x8d140000 | lw \$t4, 0x00000008(\$t0) | # x3 = MEM[2] | |
| 0x00400014 | 0x8d14000c | lw \$t5, 0x00000010(\$t0) | # x2 = MEM[3] | |
| 0x00400018 | 0x02342402 | add \$t1, \$t1, \$t2 | # t1 = x1 + x2 | |
| 0x0040001c | 0x01343402 | add \$t1, \$t1, \$t3 | # t1 = x1 + x2 + x3 | |
| 0x00400020 | 0x01343402 | add \$t1, \$t1, \$t4 | # soma = x1 + x2 + x3 + x4 | |
| 0x00400024 | 0xad500100 | sw \$t5, 0x00000010(\$t0) | # MEM[4] = soma | |

Labels

| Label | Address |
|----------------|------------|
| (global) | 0x00400000 |
| main | 0x10010000 |
| programa09.asm | |
| x1 | 0x10010000 |
| x2 | 0x10010004 |
| x3 | 0x10010008 |
| x4 | 0x1001000c |
| soma | 0x10010010 |

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 |
| \$s3 | 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 |
| \$t8 | 16 | 0x00000000 |
| \$t9 | 17 | 0x00000001 |
| \$s2 | 18 | 0x00000019 |
| \$s3 | 19 | 0x00000000 |
| \$s4 | 20 | 0x00000000 |
| \$s5 | 21 | 0x00000000 |
| \$s6 | 22 | 0x00000000 |
| \$s7 | 23 | 0x00000000 |
| \$t8 | 24 | 0x00000000 |
| \$t9 | 25 | 0x00000000 |
| \$t0 | 26 | 0x00000000 |
| \$t1 | 27 | 0x00000000 |
| \$gp | 28 | 0x10000000 |
| \$sp | 29 | 0x7fffffc |
| \$fp | 30 | 0x00000000 |
| \$ra | 31 | 0x00000000 |
| pc | | 0x00400028 |
| 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 |
| 0x10010004 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010008 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x1001000c | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010010 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010014 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010018 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010020 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010024 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |

Mars Messages Run I/O

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

Programa 10:

Edit Execute

programa04.asm programma05.asm programma06.asm programma07.asm programma08.asm programma09.asm programma02.asm programma03.asm

programma01.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 sobreescrito apois a execucao do programa
32 # fim
33

```

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Edit Execute

Text Segment

| Blkpt | Address | Code | Basic | Source |
|------------|------------|-------------------------|---------------------------|--------|
| 0x00400000 | 0x34081000 | lw \$t0, 0x4007 | # t0 = 0x 1001 0000 | |
| 0x00400004 | 0x8d100000 | lw \$s0, 0(\$t0) | # x = MEM[0] | |
| 0x00400008 | 0x8d120004 | lw \$s2, 0x4(\$t0) | # z = MEM[1] | |
| 0x0040000c | 0x00040000 | ori \$t1, \$0, 0x16,7 | # t1 = 128*x | |
| 0x00400010 | 0x00040000 | sll \$t1, \$s0, 0x0007 | # t1 = 127*x | |
| 0x00400014 | 0x00040000 | sub \$t1, \$t1, \$s0 | # t1 = 127x | |
| 0x00400018 | 0x01240000 | add \$t2, \$t2, \$s2 | # t2 = 65*z | |
| 0x00400020 | 0x01240000 | add \$t1, \$t1, \$t2 | # t1 = 127x - 65z | |
| 0x00400024 | 0x01240000 | addi \$s1, \$t1, 0x0001 | # y = 127x - 65z + 1 | |
| 0x00400028 | 0xad500100 | sw \$s1, 0x0008(\$t0) | # MEM[2] = 127x - 65z + 1 | |

Labels

| Label | Address |
|-----------------|------------|
| (global) | 0x00400000 |
| main | 0x10010000 |
| programma10.asm | |
| x | 0x10010000 |
| z | 0x10010004 |
| y | 0x10010008 |

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 |
| \$s3 | 7 | 0x00000000 |
| \$t0 | 8 | 2685000000 |
| \$t1 | 9 | 180 |
| \$t2 | 10 | 455 |
| \$t3 | 11 | 0 |
| \$t4 | 12 | 0 |
| \$t5 | 13 | 0 |
| \$t6 | 14 | 0 |
| \$t7 | 15 | 0 |
| \$t8 | 16 | 0 |
| \$t9 | 17 | 181 |
| \$s0 | 18 | 0 |
| \$s1 | 19 | 0 |
| \$s2 | 20 | 0 |
| \$s3 | 21 | 0 |
| \$s4 | 22 | 0 |
| \$s5 | 23 | 0 |
| \$s6 | 24 | 0 |
| \$s7 | 25 | 0 |
| \$s8 | 26 | 0 |
| \$s9 | 27 | 0 |
| \$fp | 28 | 268468224 |
| \$sp | 29 | 2147479748 |
| \$ra | 30 | 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 | 5 | 7 | 181 | 0 | 0 | 0 | 0 | 0 |
| 0x10010004 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010008 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x1001000c | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010010 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010014 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010018 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010020 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010024 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Mars Messages Run I/O

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

Programa 11:

```

Edit Execute
programa04.asm programa05.asm programa06.asm programa07.asm programa08.asm programa09.asm programa10.asm programa11.asm
programa01.asm programma02.asm programma03.asm
1 # Programa 11 (lw e sw)
2
3 # {
4 #   y = x - z + 300000;
5 # }
6
7 # Associações
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    sub $t1, $s0, $s2      # t1 = x - z
21    lui $t2, 0x0004      # t2 = 0x 0004 0000
22    ori $t2, $t2, 0x93E0  # t2 = 0x 0004 93E0 // 300000
23    add $s1, $t1, $t2      # y = x - z + 300000
24    sw $s1, 0x0008($t0)  # MEM[2] = x - z + 300000
25
26 .data
27 x: .word 100000
28 z: .word 200000
29 y: .word 0          # esse valor deverá ser sobreescrito após a execução do programa
30 # fim

```

The screenshot shows the debugger interface with multiple windows:

- Registers:** Shows the CPU registers (\$zero to \$t1, \$s0 to \$s2, \$t2, \$sp, \$fp, \$ra, \$pc, \$hi, \$lo) with their names, numbers, and values.
- Labels:** Shows the global labels and their addresses: main (0x00400000), \$s0 (0x10010000), \$s1 (0x10010004), \$s2 (0x10010008), \$t0 (0x1001000C), \$t1 (0x10010010), \$t2 (0x10010014), \$s3 (0x10010018), \$s4 (0x1001001C), \$s5 (0x10010020), \$s6 (0x10010024), \$s7 (0x10010028), \$s8 (0x1001002C), \$s9 (0x10010030), \$s10 (0x10010034), \$s11 (0x10010038), \$s12 (0x10010040), \$s13 (0x10010044), \$s14 (0x10010048), \$s15 (0x1001004C), \$s16 (0x10010050), \$s17 (0x10010054), \$s18 (0x10010058), \$s19 (0x1001005C), \$s20 (0x10010060), \$s21 (0x10010064), \$s22 (0x10010068), \$s23 (0x10010070), \$s24 (0x10010074), \$s25 (0x10010078), \$s26 (0x10010080), \$s27 (0x10010084), \$s28 (0x10010088), \$s29 (0x10010090), \$s30 (0x10010094), \$s31 (0x10010098).
- Text Segment:** Shows the assembly code for the main function with comments explaining each instruction.
- Data Segment:** Shows the memory dump of the data segment from address 0x10010000 to 0x10010040.
- Mars Messages:** Shows a message indicating the program has finished running.

Programa 12:

```

Edit Execute
programa05.asm programa06.asm programa07.asm programa08.asm programa09.asm programa10.asm programa11.asm programa12.asm
programa01.asm programma02.asm programma03.asm programma04.asm
1 # Programa 12 (lw e sw)
2
3 # {
4 #   int ***x;
5 #   K = MEM [ MEM [ MEM [ x ] ] ];
6 #   K = 2^K;
7 #   MEM [ MEM [ MEM [ x ] ] ] = k;
8 # }
9
10 # Associações
11 # x -> $s0
12 # k -> $s1
13
14 # inicio
15 .text
16 .globl main
17
18 main:
19    lui $s0, 0x1001      # x = 0x 1001 0000
20    lw $t0, 0x0000($s0)  # t0 = MEM[x]
21    lw $t1, 0x0000($t0)  # t1 = MEM[MEM[x]]
22    lw $s1, 0x0000($t1)  # k = MEM [ MEM[MEM[x]] ]
23    sll $s1, $s1, 0x0001  # k = 2^k
24    sw $s1, 0x0000($t1)  # MEM [ MEM[MEM[x]] ] = 2^k
25
26 .data
27 x: .word 0x10010004
28 pont_x: .word 0x10010008
29 pont_pont_x: .word 20
30 # fim
31

```

The screenshot shows the Mars 32-bit CPU Emulator interface with several windows open:

- Text Segment**: Shows assembly code for `program12.asm`. The highlighted line is:


```
lw $t1, MEM[x]    # t1 = MEM[MEM[x]]
```
- Registers**: Displays CPU registers (r0-r31, pc, hi, lo) with their names, numbers, and current values.
- Data Segment**: Displays memory dump starting at address `0x00010000`, showing values for `r0` through `r31`.
- Labels**: Shows labels and their addresses in the program.
- Run I/O**: Shows the current run status.

Programa 13:

```
1 # Programa 13 (beg, bne, j)
2
3 #
4 #   int A = MEM[0];
5 #   if (A < 0) A = -A;
6 #   MEM[0] = A;
7 #
8
9 # Associações
10 # A -> $s0
11
12 # inicio
13 .text
14 .globl main
15
16 main:
17    lui $t0, 0x1001      # x = 0x1001 0000
18    lw $s0, 0x0000($t0) # A = MEM[0]
19    srl $t1, $s0, 0x001F # tl = 0x00000 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
```

The screenshot shows the Mars 32-bit Simulator interface with several windows open:

- Text Segment**: Displays assembly code for the program13.asm file. The code includes instructions like `lui`, `sll`, `add`, `beq`, and `sw`. A yellow highlight covers the instruction at address 0x00400014.
- Registers**: Shows the state of various registers. The highlighted register \$r0 contains the value 16.
- Labels**: Lists labels and their addresses, including `main` at 0x00400000 and `program13.asm`.
- Data Segment**: Displays memory dump starting at address 0x10010000, showing values for `hi` and `lo`.
- Mars Messages**: Shows a message indicating the program has finished running.

Edit Execute

programa06.asm programma07.asm programma08.asm programma09.asm programma10.asm programma11.asm programma12.asm programma13.asm programma14.asm programma15.asm programma05.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: 7
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 \$t0, 0x00000001 | 17: | lui \$t0, 0x1001 # x = 0x 1001 0000 |
| 0x00400008 | 0x00000000 | lw \$s0, 0x0000(\$t0) | 18: | lw \$s0, 0x0000(\$t0) # A = MEM[0] |
| 0x0040000F | 0x00000000 | srl \$t1, \$s0, 0x001F | 19: | srl \$t1, \$s0, 0x001F # t1 = 0x0000 000? // ? = 0 se positivo; ? = 1 se negativo |
| 0x00400016 | 0x00000002 | beq \$t1, \$0, end | 20: | beq \$t1, \$0, end # if (A > 0) end |
| 0x00400020 | 0x00000002 | sub \$s0, \$0, \$s0 | 21: | sub \$s0, \$0, \$s0 # else A = A |
| 0x00400024 | 0x00000000 | sw \$s0, 0(\$t0) | 22: | sw \$s0, 0(\$t0) # MEM[0] = A |

Registers

| Name | Number | Value |
|------|--------|------------|
| zero | 0 | 0x00000000 |
| \$t1 | 1 | 0x00000000 |
| \$v0 | 2 | 0x00000000 |
| \$v1 | 3 | 0x00000000 |
| \$s0 | 4 | 0x00000000 |
| \$s1 | 5 | 0x00000000 |
| \$s2 | 6 | 0x00000000 |
| \$s3 | 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 |
| \$t0 | 26 | 0x00000000 |
| \$t1 | 27 | 0x00000000 |
| \$sp | 28 | 0x10000000 |
| \$fp | 29 | 0x7fffffc; |
| \$ra | 30 | 0x00000000 |
| pc | 31 | 0x00000000 |
| hi | | 0x00000000 |
| lo | | 0x00000000 |

Data Segment

| Address | Value (+0) | Value (+4) | Value (+8) | Value (+c) | Value (+10) | Value (+14) | Value (+18) | Value (+lc) |
|------------|------------|------------|------------|------------|-------------|-------------|-------------|-------------|
| 0x10000000 | 0x00000007 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10000020 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10000040 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10000060 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10000080 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x100000A0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x100000C0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x100000E0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10000100 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10000120 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10000140 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |

Mars Messages Run I/O

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

Programa 14:

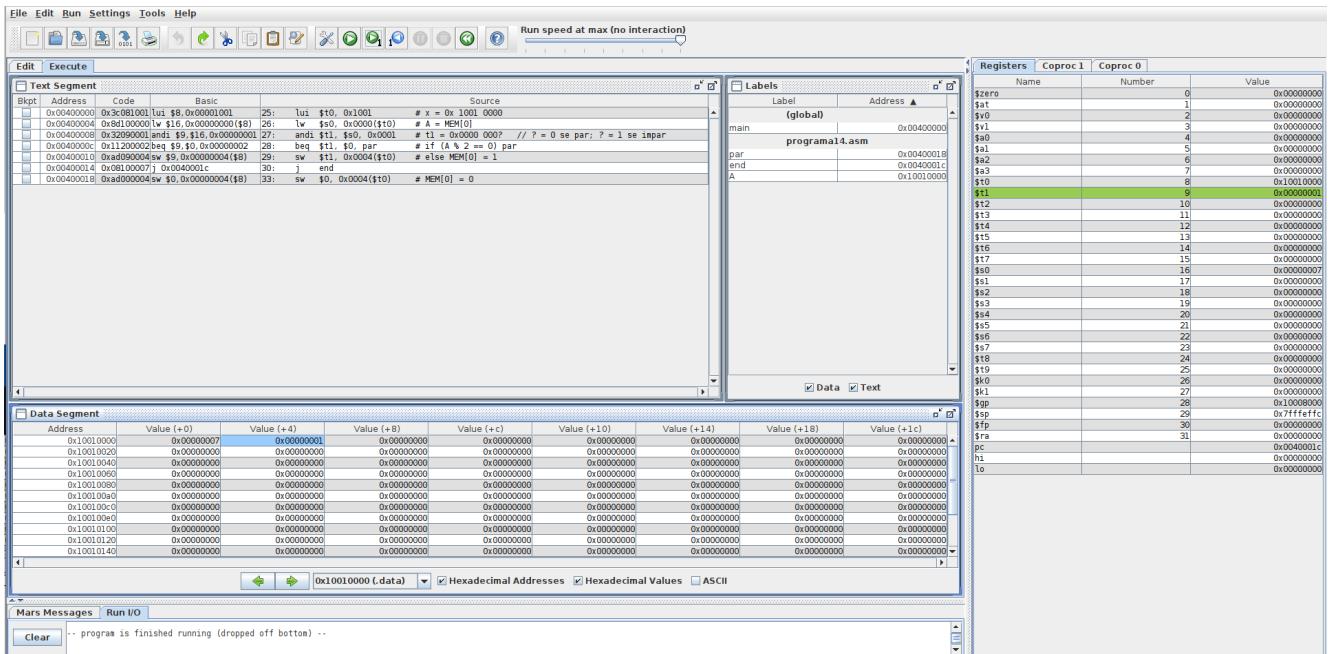
Edit Execute

programa07.asm programma08.asm programma09.asm programma10.asm programma11.asm programma12.asm programma13.asm programma14.asm programma15.asm programma05.asm programma06.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 # SOLUCAO SEM USAR DESVIO
17 # main:
18 #    lui $t0, 0x1001      # x = 0x 1001 0000
19 #    lw $s0, 0x0000($t0)  # A = MEM[0]
20 #    andi $t1, $s0, 0x0001 # t1 = 0x0000 000? // ? = 0 se par; ? = 1 se impar
21 #    beq $t1, $0, par     # if (A % 2 == 0) par
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: 7
39
40 # fim

```

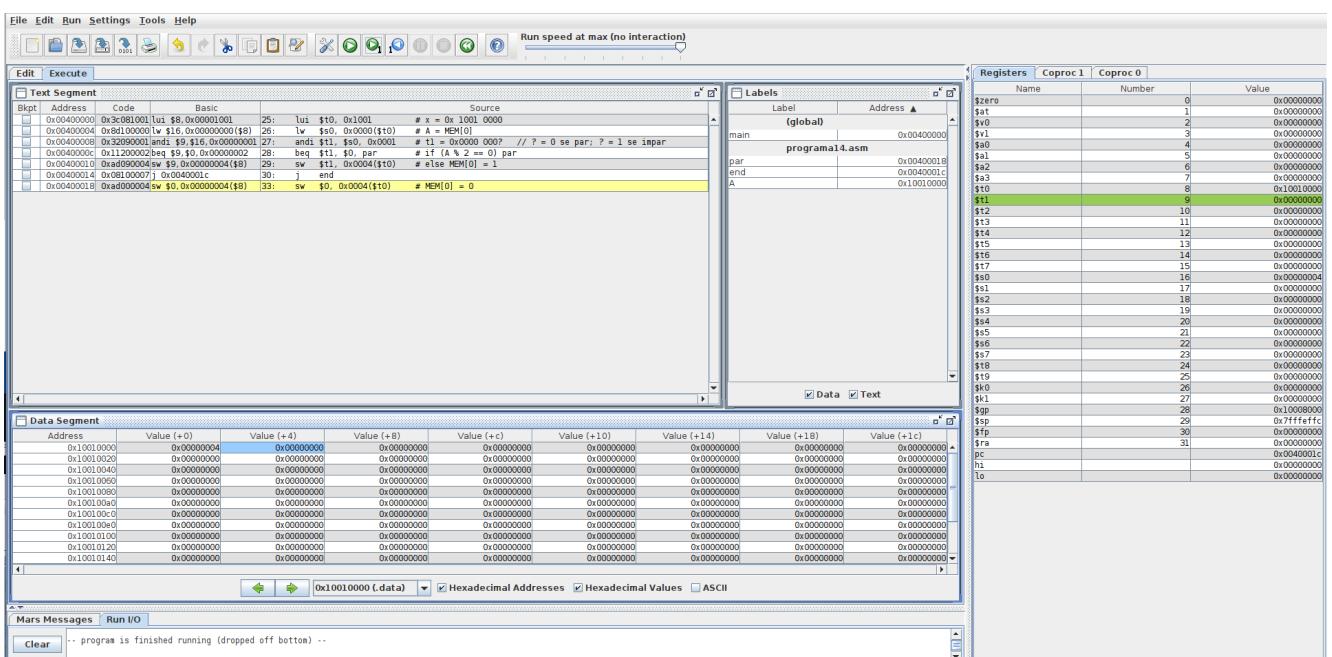


```

Edit Execute
programa07.asm programa08.asm programa09.asm programa10.asm programma11.asm programma12.asm programma13.asm programma14.asm
programma01.asm programma02.asm programma03.asm programma04.asm programma05.asm programma06.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 # Associações
10 # A -> $s0
11
12 # inicio
13 .text
14 .globl main
15
16
17 # SOLUÇÃO 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   $s0, 0x0000($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

```



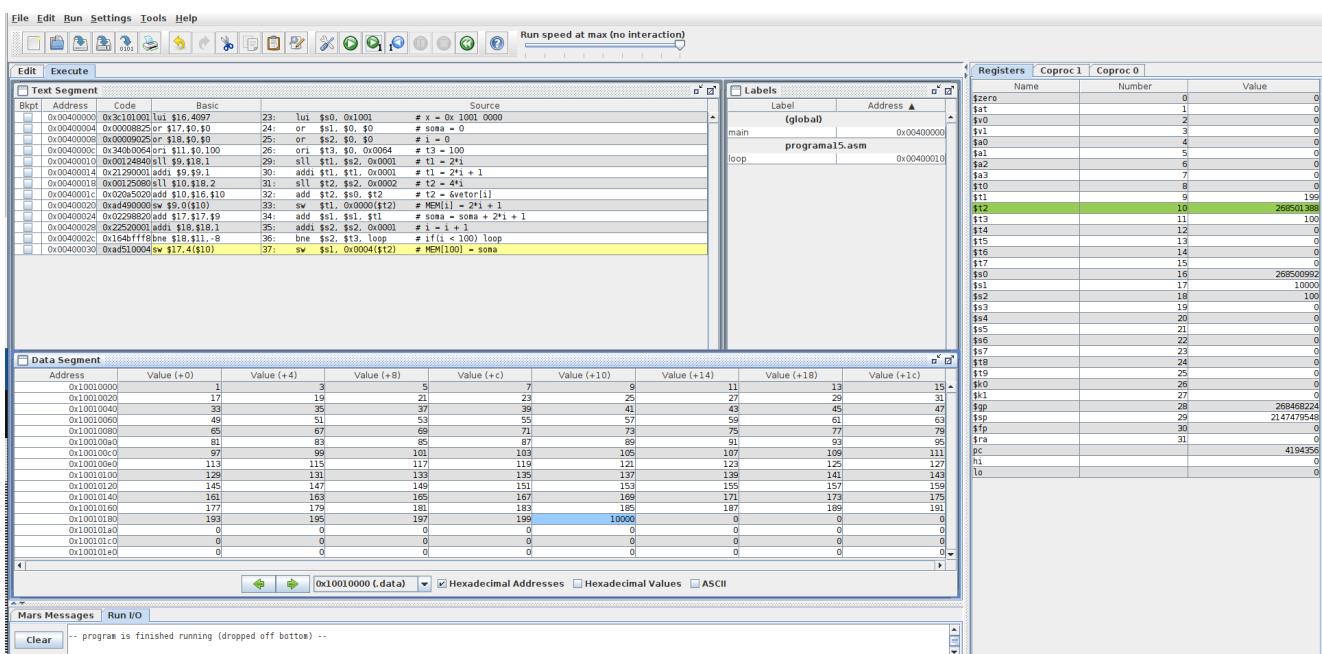
Programa 15:

```

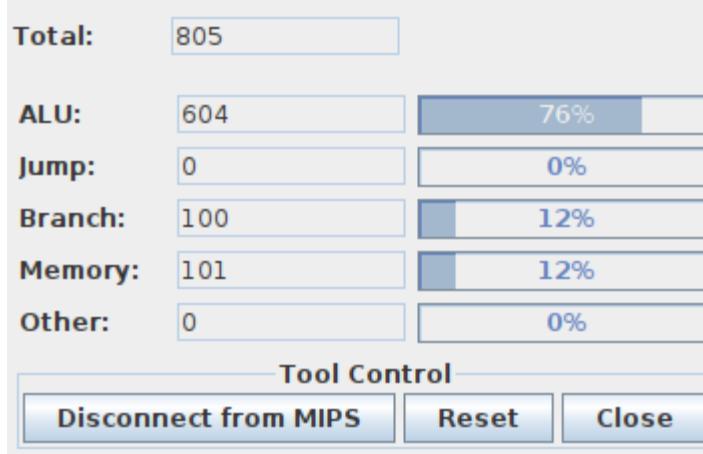
Edit Execute
programa08.asm programa09.asm programa10.asm programa11.asm programa12.asm programa13.asm programa14.asm programa15.asm
programa01.asm programa02.asm programa03.asm programa04.asm programa05.asm programa06.asm programa07.asm

1 # Programa 15 (beg, bne, j)
2
3 #
4 #   int vetor[100];
5 #   int soma = 0;
6 #   for(int i = 0; i < 100; i++) {
7 #       vetor[i] = 2*i + 1; // MEM[i] = 2*i + 1;
8 #       soma += 2*i + 1;
9 #   }
10 #   MEM[100] = soma;
11 #
12
13 # Associacoes
14 # vetor[0] -> $s0
15 # soma -> $s1
16 # i -> $s2
17
18 # inicio
19 .text
20 .globl main
21
22 main:
23    lui    $s0, 0x1001      # x = 0x 1001 0000
24    or     $s1, $0, $0      # soma = 0
25    or     $s2, $0, $0      # i = 0
26    ori    $t3, $0, 0x0064  # t3 = 100
27
28 loop:
29    sll   $t1, $s2, 0x0001  # t1 = 2*i
30    addi $t1, $t1, 0x0001  # t1 = 2*i + 1
31    sll   $t2, $s2, 0x0002  # t2 = 4*i
32    addi $t2, $s0, $t2      # t2 = &vetor[i]
33    sw    $t1, 0x0000($t2)  # MEM[i] = 2*i + 1
34    addi $s1, $s1, $t1      # soma = soma + 2*i + 1
35    addi $s2, $s2, 0x0001  # i = i + 1
36    bne  $s2, $t3, loop    # if(i < 100) loop
37    sw    $s1, 0x0004($t2)  # MEM[100] = soma
38
39 .data
40 # fim

```



Instruction Statistics, Version 1.0 (Ingo Kofler)

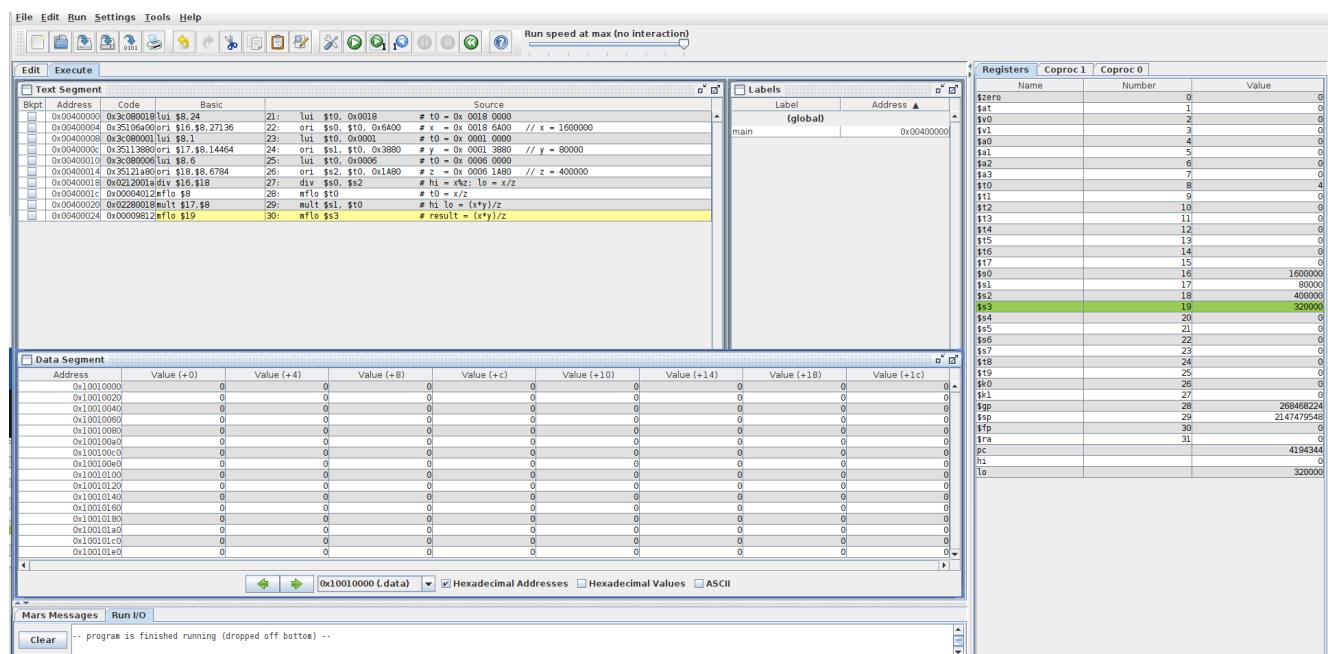


Programa 16:

```

Edit Execute
programa08.asm programma09.asm programma10.asm programma11.asm programma12.asm programma13.asm programma14.asm programma15.asm programma16.asm
programma01.asm programma02.asm programma03.asm programma04.asm programma05.asm programma06.asm programma07.asm
1 # Programa 16 (beq, bne, j)
2
3 # {
4 #   x = 1600000; // x = 0x 0018 6400
5 #   y = 80000; // y = 0x 0001 3880
6 #   z = 400000; // z = 0x 0006 1A80
7 #   result = (x*y)/z;
8 # }
9
10 # Associacoes
11 # x -> $s0
12 # y -> $s1
13 # z -> $s2
14 # result -> $s3
15
16 # inicio
17 .text
18 .globl main
19
20 main:
21   lui $t0, 0x0018          # t0 = 0x 0018 0000
22   ori $s0, $t0, 0x6A00      # x = 0x 0018 6400 // x = 1600000
23   lui $t0, 0x0001          # t0 = 0x 0001 0000
24   ori $s1, $t0, 0x3880      # y = 0x 0001 3880 // y = 80000
25   lui $t0, 0x0006          # t0 = 0x 0006 0000
26   ori $s2, $t0, 0x1A80      # z = 0x 0006 1A80 // z = 400000
27   div $s0, $s2              # hi = x%z; lo = x/z
28   mflo $t0                  # t0 = x/z
29   mult $s1, $t0              # hi_lo = (x*y)/z
30   mflo $s3                  # result = (x*y)/z
31
32 .data
33 # fim

```



Programa 17:

```

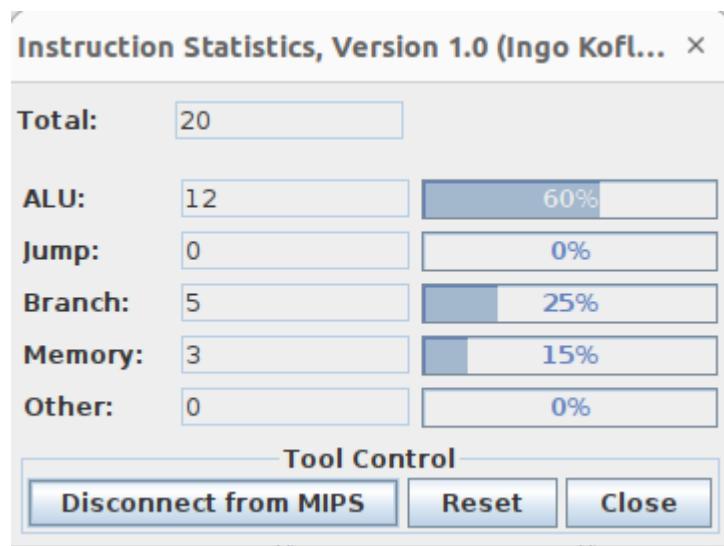
Edit Execute
programa09.asm programma10.asm programma11.asm programma12.asm programma13.asm programma14.asm programma15.asm programma16.asm programma17.asm
programma01.asm programma02.asm programma03.asm programma04.asm programma05.asm programma06.asm programma07.asm programma08.asm
1 # Programa 17 (beq, bne, j)
2
3 # {
4 #   x = MEM[0];
5 #   y = MEM[1];
6 #   k = x * y
7 #   MEM[2] = k;
8 # }
9
10 # Associacoes
11 # x -> $s0
12 # y -> $s1
13 # k -> $s2
14
15 # inicio
16 .text
17 .globl main
18
19 main:
20   lui $t0, 0x1001          # t0 = 0x 1001 0000
21   lw $s0, 0x0000($t0)      # x = MEM[0]
22   lw $s1, 0x0004($t0)      # y = MEM[1]
23
24 multi:
25   add $t1, $t1, $s0          # t1 = t1 + x
26   addi $t2, $t2, 0x0001      # cont = cont + 1
27   bne $t2, $s1, multi        # if(cont != y) multi
28
29   add $s2, $0, $t1            # k = x*y
30   sw $s2, 0x0008($t0)       # MEM[2] = k
31
32 .data
33 x: 7
34 y: 5
35 # fim

```

The screenshot shows the Mars 32-bit CPU Emulator interface with several windows open:

- Text Segment**: Displays assembly code for `program17.asm`. The code includes instructions like `lui`, `add`, `sw`, and `lw`. Address 30 is highlighted.
- Registers**: Shows the state of various registers. The `$zero` register is at address 0, and the `$t1` register is at address 9. The `pc` register is at address 4194336.
- Data Segment**: Displays memory dump starting at address 0x00000000. The value at address 0 is 7, and the value at address 30 is 35.
- Labels**: Shows the labels defined in the assembly code, such as `main` and `multi`.

At the bottom, there are buttons for running the program and a message bar indicating "program is finished running (dropped off bottom) ..".



Programa 18:

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Labels

| Label | Address |
|----------|------------|
| (global) | 0x00400000 |
| main | 0x00400000 |
| loop | 0x00400028 |
| zero | 0x00400040 |
| x | 0x00400058 |
| y | 0x00400064 |
| k | 0x00400068 |

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 | 32 |
| \$t2 | 10 | 0 |
| \$t3 | 11 | 0 |
| \$t4 | 12 | 5 |
| \$t5 | 13 | 32 |
| \$t6 | 14 | 0 |
| \$t7 | 15 | 0 |
| \$t8 | 16 | 2 |
| \$t9 | 17 | 5 |
| \$t10 | 18 | 32 |
| \$t11 | 19 | 0 |
| \$t12 | 20 | 0 |
| \$t13 | 21 | 0 |
| \$t14 | 22 | 0 |
| \$t15 | 23 | 0 |
| \$t16 | 24 | 0 |
| \$t17 | 25 | 0 |
| \$t18 | 26 | 0 |
| \$t19 | 27 | 0 |
| \$t20 | 28 | 268468224 |
| \$t21 | 29 | 214749548 |
| \$t22 | 30 | 0 |
| \$t23 | 31 | 4194396 |
| pc | | |
| hi | | |
| lo | | |

Text Segment

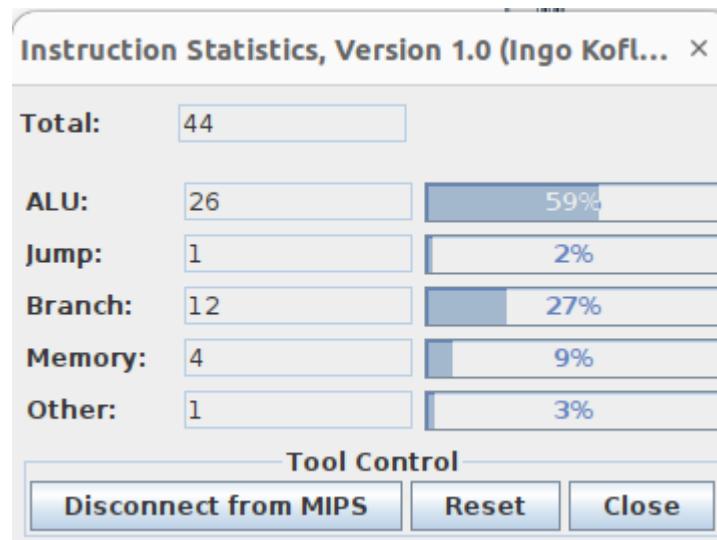
| Bkp | Address | Code | Basic | Source |
|------------|---------------|------------------------|----------------------------|-----------------------|
| 0x00400000 | 0x3c081001 | lui \$t0, \$0,4097 | 12: lui \$t0, 0x1001 | # t0 = 0x 1001 0000 |
| 0x00400004 | 0x8d100000 | lw \$t0, 0x0000(\$t0) | 13: lw \$t0, 0x0000(\$t0) | # x = MEM[0] |
| 0x00400008 | 0x3c081001 | lui \$t1, 0x0004(\$t0) | 14: lui \$t1, 0x0004(\$t0) | # y = MEM[1] |
| 0x00400012 | 0x8d100000 | lw \$t1, 0x0004(\$t0) | 15: lw \$t1, 0x0004(\$t0) | # k = MEM[2] |
| 0x00400016 | 0x00004(\$t0) | addi \$t4, \$0, 0x0001 | 16: addi \$t4, \$0, 0x0001 | # conty = 1 |
| 0x00400020 | 0x00004(\$t0) | addi \$t5, \$0, \$t0 | 17: addi \$t5, \$0, \$t0 | # t5 = x |
| 0x00400024 | 0x00004(\$t0) | # 7ratar execao | | |
| 0x00400028 | 0x00004(\$t0) | slt \$t3, \$t1, \$0 | 18: slt \$t3, \$t1, \$0 | # t3 = 1 if y < 0 |
| 0x00400032 | 0x00004(\$t0) | bne \$t3, \$0, store | 19: bne \$t3, \$0, store | # if(y < 0) k = -1 |
| 0x00400036 | 0x00004(\$t0) | beq \$t1, \$0, zero | 20: beq \$t1, \$0, zero | # if(y == 0) k = 1 |
| 0x00400040 | 0x00004(\$t0) | beq \$t1, \$0, loop | 21: beq \$t1, \$0, loop | # if(conty != y) loop |
| 0x00400044 | 0x00004(\$t0) | add \$t2, \$0, \$t5 | 22: add \$t2, \$0, \$t5 | # k = pow(x,y) |
| 0x00400048 | 0x00004(\$t0) | j store | 23: j store | # go to store |
| 0x00400052 | 0x00004(\$t0) | addi \$t2, \$0, 0x0001 | 24: addi \$t2, \$0, 0x0001 | # k = 1 |
| 0x00400056 | 0x00004(\$t0) | lui \$t0, 0x1010 | 25: lui \$t0, 0x1010 | # t0 = 0x 1010 0000 |
| 0x00400060 | 0x00004(\$t0) | addi \$t2, \$0, 0x0003 | 26: addi \$t2, \$0, 0x0003 | # conty = conty + 1 |
| 0x00400064 | 0x00004(\$t0) | bne \$t3, \$t1, \$0 | 27: bne \$t3, \$t1, \$0 | # if(conty != x) loop |
| 0x00400068 | 0x00004(\$t0) | add \$t2, \$0, \$t0 | 28: add \$t2, \$0, \$t0 | # t5 = t1 (atual) |
| 0x00400072 | 0x00004(\$t0) | addi \$t2, \$0, 0x0001 | 29: addi \$t2, \$0, 0x0001 | # 12 = 12 + 1 |
| 0x00400076 | 0x00004(\$t0) | lui \$t1, 0x0004(\$t0) | 30: lui \$t1, 0x0004(\$t0) | # y = 0x 0004(\$t0) |
| 0x00400080 | 0x00004(\$t0) | add \$t2, \$0, \$t1 | 31: add \$t2, \$0, \$t1 | # conty = conty + 1 |
| 0x00400084 | 0x00004(\$t0) | bne \$t3, \$t1, \$0 | 32: bne \$t3, \$t1, \$0 | # if(y == 0) k = -1 |
| 0x00400088 | 0x00004(\$t0) | beq \$t1, \$0, zero | 33: beq \$t1, \$0, zero | # if(y == 0) k = 1 |
| 0x00400092 | 0x00004(\$t0) | beq \$t1, \$0, um | 34: beq \$t1, \$0, um | # if(y == 1) k = x |
| 0x00400096 | 0x00004(\$t0) | add \$t2, \$0, \$t5 | 35: add \$t2, \$0, \$t5 | # t5 = t1 - 1 |
| 0x00400100 | 0x00004(\$t0) | add \$t2, \$0, \$t0 | 36: add \$t2, \$0, \$t0 | # t5 = t1 - 1 |
| 0x00400104 | 0x00004(\$t0) | lui \$t0, 0x1010 | 37: lui \$t0, 0x1010 | # t0 = 0x 1010 0000 |
| 0x00400108 | 0x00004(\$t0) | add \$t2, \$0, \$t0 | 38: add \$t2, \$0, \$t0 | # conty = conty + 1 |
| 0x00400112 | 0x00004(\$t0) | bne \$t3, \$t1, \$0 | 39: bne \$t3, \$t1, \$0 | # if(conty != x) loop |
| 0x00400116 | 0x00004(\$t0) | add \$t2, \$0, \$t0 | 40: add \$t2, \$0, \$t0 | # t5 = t1 (atual) |
| 0x00400120 | 0x00004(\$t0) | add \$t2, \$0, \$t0 | 41: add \$t2, \$0, \$t0 | # k = x |
| 0x00400124 | 0x00004(\$t0) | sw \$t2, 0x0008(\$t0) | 42: sw \$t2, 0x0008(\$t0) | # MEM[2] = k |
| 0x00400128 | 0x00004(\$t0) | add \$t2, \$0, \$t0 | 43: add \$t2, \$0, \$t0 | # k = x |
| 0x00400132 | 0x00004(\$t0) | sw \$t2, 0x0008(\$t0) | 44: sw \$t2, 0x0008(\$t0) | # MEM[2] = k |

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 |
| 0x10010090 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010098 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x100100a0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x100100b0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x100100c0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x100100d0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x100100e0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x100100f0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010100 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010110 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010120 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010130 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010140 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Mars Messages Run I/O

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



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:

```

Edit Execute
programa13.asm programma14.asm programma15.asm programma16.asm programma17.asm
programa05.asm programma06.asm programma07.asm programma08.asm
programa01.asm programma02.asm
1 # Programa 19
2
3 {
4 #   x = MEM[0];
5 #   y = MEM[1];
6 #   t0 = bitsSignificantes de x
7 #   tl = bitsSignificantes de y
8 #   if (t0 + tl < 32) s2 = x*y
9 #   else s2, s3 = x*y
10 #
11
12 # Associacoes
13 # x -> $s0
14 # y -> $s1
15
16 # inicio
17 .text
18 .globl main
19
20 main:
21   lui $t0, 0x1001      # t0 = 0x 1001 0000
22   lw $s0, 0x0000($t0)  # x = MEM[0]
23   lw $s1, 0x0004($t0)  # y = MEM[1]
24   and $t0, $t0, $0     # t0 = 0
25   lui $t3, 0xFFFF      # t3 = 0x FFFF 0000
26   ori $t3, $t3, 0xFFFF # t3 = 0x FFFF FFFF
27   srl $t4, $s0, $s0    # t4 = 0x 0000 000? // ? = 0 se x >= 0; ? = 1 se x < 0
28   add $t2, $zero, $s0   # t2 = x
29   beq $t4, $0, positivo1 # if (x % 2 == 0) positivo1
30   j negativo1          # else negativo1
31
32 continue1:
33   srl $t4, $s1, 0x001F # t4 = 0x 0000 000? // ? = 0 se y >= 0; ? = 1 se y < 0
34   add $t2, $zero, $s1   # t2 = y
35   beq $t2, $0, positivo2 # if (y % 2 == 0) positivo2
36   j negativo2          # else negativo2
37
38 continue2:
39   mult $s0, $s1         # hi_lo = x * y
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, 0x0020 # t3 = 32
50     slt $t0, $t0, $t3    # if (contTotal < 32) t0 = 1; else t0 = 0
51     beg $t0, $0, high    # if (t0 == 0) high
52     j low                # else low
53
54 positivo2:
55   loop2:
56     sra $t2, $t2, 0x0001 # t2 = t2 >> 1
57     addi $t0, $t0, 0x0001 # contx = contx + 1
58     bne $t2, $0, loop1  # if(t2 != 0) loop1
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     beg $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, $0, loop4  # if(t2 != 0) 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: 75000000
86 y: -11000000
87 # fim

```

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Edit Execute

Text Segment

| Bkpt | Address | Code | Basic | Source |
|------|------------|------------|--------------------------|---|
| | 0x04000000 | 0x3c081001 | ld \$t0, 0x1000 | 21: lui \$t0, 0x1000 # t0 = 0x 1001 0000 |
| | 0x04000004 | 0x3c081001 | ld \$t1, 0x0000(\$t0) | 22: lw \$t1, 0x0000(\$t0) # y = MEM[0] |
| | 0x04000008 | 0x8d110004 | lw \$t0, 0x17,4(\$s0) | 23: lw \$t0, 0x0004(\$t0) # y = MEM[1] |
| | 0x04000010 | 0x01004024 | and \$t0, \$t0, \$0 | 24: and \$t0, \$t0, \$0 # t0 = 0 |
| | 0x04000014 | 0x3c081001 | ld \$t3, 0x0000(\$t0) | 25: lui \$t3, 0xFFFF # t3 = 0x FFFF 0000 |
| | 0x04000018 | 0x01004024 | and \$t3, 0x17,4(\$s0) | 26: ori \$t3, \$t3, 0xFFFF # t3 = 0x FFFF FFFF |
| | 0x04000020 | 0x01004024 | and \$t3, 0x17,4(\$s1) | 27: srl \$t4, \$s0, \$s0 # t4 = 0x 0000 000? // ? = 0 se x >= 0; ? = 1 se x < 0 |
| | 0x04000024 | 0x01004024 | and \$t3, 0x17,4(\$s1) | 28: add \$t2, \$zero, \$s0 # t2 = x |
| | 0x04000028 | 0x01180008 | beq \$t4, \$0, positivo1 | 29: beq \$t4, \$0, positivo1 # if (x % 2 == 0) positivo1 |
| | 0x04000032 | 0x01180008 | beq \$t2, \$0, 0x11 | 30: j negativo1 # else negativo1 |
| | 0x04000036 | 0x01004024 | and \$t3, 0x17,31 | 31: srl \$t4, \$s1, 0x001F # t4 = 0x 0000 000? // ? = 0 se y >= 0; ? = 1 se y < 0 |
| | 0x04000040 | 0x01180008 | beq \$t4, \$0, 0x17 | 32: add \$t2, \$zero, \$s1 # t2 = y |
| | 0x04000044 | 0x01004024 | and \$t3, 0x17,15 | 33: beq \$t4, \$0, positivo2 # if (y % 2 == 0) positivo2 |
| | 0x04000048 | 0x01004024 | and \$t3, 0x17,15 | 34: j negativo2 # else negativo2 |
| | 0x04000052 | 0x01004024 | and \$t3, 0x17,15 | 35: mult \$s0, \$s1 # elu positivo2 |
| | 0x04000056 | 0x01004024 | and \$t3, 0x17,15 | 36: add \$t0, \$t0, \$t1 # contTotal = contx + conty |
| | 0x04000060 | 0x01004024 | and \$t3, 0x17,15 | 37: addi \$t3, \$0, 0x0020 # t3 = 32 |
| | 0x04000064 | 0x21080001 | slt \$t0, \$t0, \$t3 | 38: slt \$t0, \$t0, \$t3 # if (contTotal < 32) t0 = 1; else t0 = 0 |
| | 0x04000068 | 0x01004024 | and \$t3, 0x17,15 | 39: beq \$t0, \$0, high # if (t0 == 0) high |
| | 0x04000072 | 0x01004024 | and \$t3, 0x17,15 | 40: j low # else low |
| | 0x04000076 | 0x01004024 | and \$t3, 0x17,15 | 41: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000080 | 0x01004024 | and \$t3, 0x17,15 | 42: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000084 | 0x01004024 | and \$t3, 0x17,15 | 43: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000088 | 0x01004024 | and \$t3, 0x17,15 | 44: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000092 | 0x01004024 | and \$t3, 0x17,15 | 45: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000096 | 0x01004024 | and \$t3, 0x17,15 | 46: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000010 | 0x01004024 | and \$t3, 0x17,15 | 47: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000014 | 0x01004024 | and \$t3, 0x17,15 | 48: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000018 | 0x01004024 | and \$t3, 0x17,15 | 49: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000022 | 0x01004024 | and \$t3, 0x17,15 | 50: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000026 | 0x01004024 | and \$t3, 0x17,15 | 51: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000030 | 0x01004024 | and \$t3, 0x17,15 | 52: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000034 | 0x01004024 | and \$t3, 0x17,15 | 53: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000038 | 0x01004024 | and \$t3, 0x17,15 | 54: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000042 | 0x01004024 | and \$t3, 0x17,15 | 55: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000046 | 0x01004024 | and \$t3, 0x17,15 | 56: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000050 | 0x01004024 | and \$t3, 0x17,15 | 57: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000054 | 0x01004024 | and \$t3, 0x17,15 | 58: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000058 | 0x01004024 | and \$t3, 0x17,15 | 59: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000062 | 0x01004024 | and \$t3, 0x17,15 | 60: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000066 | 0x01004024 | and \$t3, 0x17,15 | 61: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000070 | 0x01004024 | and \$t3, 0x17,15 | 62: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000074 | 0x01004024 | and \$t3, 0x17,15 | 63: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000078 | 0x01004024 | and \$t3, 0x17,15 | 64: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000082 | 0x01004024 | and \$t3, 0x17,15 | 65: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000086 | 0x01004024 | and \$t3, 0x17,15 | 66: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000090 | 0x01004024 | and \$t3, 0x17,15 | 67: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000094 | 0x01004024 | and \$t3, 0x17,15 | 68: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000098 | 0x01004024 | and \$t3, 0x17,15 | 69: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000012 | 0x01004024 | and \$t3, 0x17,15 | 70: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000016 | 0x01004024 | and \$t3, 0x17,15 | 71: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000020 | 0x01004024 | and \$t3, 0x17,15 | 72: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000024 | 0x01004024 | and \$t3, 0x17,15 | 73: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000028 | 0x01004024 | and \$t3, 0x17,15 | 74: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000032 | 0x01004024 | and \$t3, 0x17,15 | 75: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000036 | 0x01004024 | and \$t3, 0x17,15 | 76: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000040 | 0x01004024 | and \$t3, 0x17,15 | 77: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000044 | 0x01004024 | and \$t3, 0x17,15 | 78: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000048 | 0x01004024 | and \$t3, 0x17,15 | 79: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000052 | 0x01004024 | and \$t3, 0x17,15 | 80: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000056 | 0x01004024 | and \$t3, 0x17,15 | 81: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000060 | 0x01004024 | and \$t3, 0x17,15 | 82: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000064 | 0x01004024 | and \$t3, 0x17,15 | 83: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000068 | 0x01004024 | and \$t3, 0x17,15 | 84: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000072 | 0x01004024 | and \$t3, 0x17,15 | 85: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000076 | 0x01004024 | and \$t3, 0x17,15 | 86: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000080 | 0x01004024 | and \$t3, 0x17,15 | 87: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000084 | 0x01004024 | and \$t3, 0x17,15 | 88: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000088 | 0x01004024 | and \$t3, 0x17,15 | 89: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000092 | 0x01004024 | and \$t3, 0x17,15 | 90: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000096 | 0x01004024 | and \$t3, 0x17,15 | 91: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000010 | 0x01004024 | and \$t3, 0x17,15 | 92: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000014 | 0x01004024 | and \$t3, 0x17,15 | 93: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000018 | 0x01004024 | and \$t3, 0x17,15 | 94: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000022 | 0x01004024 | and \$t3, 0x17,15 | 95: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000026 | 0x01004024 | and \$t3, 0x17,15 | 96: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000030 | 0x01004024 | and \$t3, 0x17,15 | 97: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000034 | 0x01004024 | and \$t3, 0x17,15 | 98: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000038 | 0x01004024 | and \$t3, 0x17,15 | 99: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000042 | 0x01004024 | and \$t3, 0x17,15 | 100: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000046 | 0x01004024 | and \$t3, 0x17,15 | 101: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000050 | 0x01004024 | and \$t3, 0x17,15 | 102: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000054 | 0x01004024 | and \$t3, 0x17,15 | 103: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000058 | 0x01004024 | and \$t3, 0x17,15 | 104: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000062 | 0x01004024 | and \$t3, 0x17,15 | 105: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000066 | 0x01004024 | and \$t3, 0x17,15 | 106: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000070 | 0x01004024 | and \$t3, 0x17,15 | 107: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000074 | 0x01004024 | and \$t3, 0x17,15 | 108: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000078 | 0x01004024 | and \$t3, 0x17,15 | 109: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000082 | 0x01004024 | and \$t3, 0x17,15 | 110: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000086 | 0x01004024 | and \$t3, 0x17,15 | 111: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000090 | 0x01004024 | and \$t3, 0x17,15 | 112: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000094 | 0x01004024 | and \$t3, 0x17,15 | 113: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000098 | 0x01004024 | and \$t3, 0x17,15 | 114: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000012 | 0x01004024 | and \$t3, 0x17,15 | 115: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000016 | 0x01004024 | and \$t3, 0x17,15 | 116: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000020 | 0x01004024 | and \$t3, 0x17,15 | 117: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000024 | 0x01004024 | and \$t3, 0x17,15 | 118: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000028 | 0x01004024 | and \$t3, 0x17,15 | 119: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000032 | 0x01004024 | and \$t3, 0x17,15 | 120: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000036 | 0x01004024 | and \$t3, 0x17,15 | 121: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000040 | 0x01004024 | and \$t3, 0x17,15 | 122: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000044 | 0x01004024 | and \$t3, 0x17,15 | 123: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000048 | 0x01004024 | and \$t3, 0x17,15 | 124: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000052 | 0x01004024 | and \$t3, 0x17,15 | 125: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000056 | 0x01004024 | and \$t3, 0x17,15 | 126: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000060 | 0x01004024 | and \$t3, 0x17,15 | 127: addi \$t0, \$t0, 0x0020 # t3 = 32 |
| | 0x04000064 | 0x01004024 | and \$t3, 0x17,15 | 128: addi \$t0, \$t0, 0x0020 # |

File Edit Execute

programa13.asm programma14.asm programma15.asm programma16.asm programma17.as
programma05.asm programma06.asm programma07.asm programma08.asm
programma01.asm programma02.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    loop:
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 continu1          # go to continu1
53
54 negativo1:
55    loop:
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 $t1, $t3, $t1   # contx = 32 - contx
60        j continu1          # go to continu1
61
62 positivo2:
63    loop:
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 continu2          # go to continu2
69
70 negativo2:
71    loop:
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 continu2          # go to continu2
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
88

```

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Text Segment

| Bkpt | Address | Code | Basic | Source |
|------|------------|---|-------|--|
| | 0x00400000 | 0x9c000000 lui \$t0, 0x1001 | 21: | lui \$t0, 0x1001 # t0 = 0x 1001 0000 |
| | 0x00400004 | 0x8d100000 lw \$t0, 0x0000(\$t0) # x = MEM[1] | 22: | lw \$t0, 0x0000(\$t0) # x = MEM[1] |
| | 0x00400008 | 0x8d100000 lw \$t1, 0x0004(\$t0) # y = MEM[1] | 23: | lw \$t1, 0x0004(\$t0) # y = MEM[1] |
| | 0x0040000c | 0x01004024 and \$t0, \$t0, \$0 | 24: | and \$t0, \$t0, \$0 # t0 = 0 |
| | 0x00400010 | 0x9c00ffff andi \$t1, \$t1, 0xFFFF | 25: | andi \$t1, \$t1, 0xFFFF # t1 = 0xFFFF 0000 |
| | 0x00400014 | 0x95e00000 ori \$t1, \$t1, 0x1111_6555 | 26: | ori \$t1, \$t1, 0xFFFF # t1 = 0xFFFF FFFF |
| | 0x00400018 | 0x00000000 add \$t0, \$t0, \$0 | 27: | add \$t0, \$t0, \$0 # t0 = 0 |
| | 0x00400020 | 0x00100020 add \$t0, \$t0, \$0 | 28: | add \$t2, \$t0, \$0 # t2 = x - 0 |
| | 0x00400024 | 0x01180000 beq \$t2, \$0, 0x11 | 29: | beq \$t2, \$0, positivo1 # if (x % 2 == 0) positivo1 |
| | 0x00400028 | 0x08100018 j \$0x400094 | 30: | j negativo1 # else negativo1 |
| | 0x00400032 | 0x01000000 add \$t0, \$t0, \$0 | 31: | add \$t2, \$t0, \$0 # t2 = 0 |
| | 0x00400036 | 0x00115020 add \$t0, \$t0, \$0 | 32: | add \$t2, \$t0, \$0 # t2 = 0 |
| | 0x00400039 | 0x1100000f beq \$t0, \$0, 0x15 | 33: | srli \$t0, \$t0, 0x000F # t4 = (0x 0000 000F) // ? = 0 se y >= 0; ? = 1 se y < 0 |
| | 0x00400043 | 0x08100020 j \$0x400094 | 34: | add \$t2, \$t0, \$0 # t2 = 0 |
| | 0x00400047 | 0x08100020 j \$0x400094 | 35: | beq \$t2, \$0, positivo2 # if (y % 2 == 0) positivo2 |
| | 0x00400051 | 0x08100020 j \$0x400094 | 36: | j negativo2 # else negativo2 |
| | 0x00400055 | 0x02110018 mul \$t0, \$t1 | 39: | mult \$t0, \$t1 # hi_lo = x * y |
| | 0x00400059 | 0x01094020 add \$t0, \$t1, \$0 | 40: | add \$t0, \$t1, \$0 # contTotal = contx + conty |
| | 0x00400063 | 0x01000000 add \$t0, \$t0, \$0 | 41: | add \$t0, \$t0, \$0 # t0 = 0 |
| | 0x00400067 | 0x01000020 add \$t0, \$t0, \$0 | 42: | slt \$t0, \$t0, \$t3 # if (contTotal < 32) t0 = 1; else t0 = 0 |
| | 0x00400071 | 0x11000011 beq \$t0, \$0, 0x17 | 43: | beq \$t0, \$0, high # else low |
| | 0x00400075 | 0x08100025 j \$0x400094 | 44: | j low # if (t0 == 0) high |
| | 0x00400079 | 0x08100025 j \$0x400094 | 45: | j continu1 # go to continu1 |
| | 0x00400083 | 0x21080000 add \$t0, \$t0, \$0 | 46: | sra \$t2, \$t2, 0x0001 # t2 = t2 >> 1 |
| | 0x00400087 | 0x1540fffbne \$t0, \$0, -3 | 47: | addi \$t0, \$t0, 0x0001 # contx = contx + 1 |
| | 0x00400091 | 0x08100000 add \$t0, \$t0, \$0 | 48: | bne \$t2, \$0, loop1 # if(t2 != 0) loop1 |
| | 0x00400095 | 0x08100000 add \$t0, \$t0, \$0 | 49: | addi \$t0, \$t0, 0x0001 # conty = conty + 1 |
| | 0x00400099 | 0x00005043 sra \$t0, \$0, \$10.1 | 50: | bne \$t2, \$0, loop2 # if(t2 != 0) loop2 |
| | 0x0040009d | 0x08100000 add \$t0, \$t0, \$0 | 51: | j continu1 # go to continu1 |
| | 0x004000a1 | 0x00005043 sra \$t0, \$0, \$10.1 | 52: | sra \$t2, \$t2, 0x0001 # t2 = t2 >> 1 |
| | 0x004000a5 | 0x08100000 add \$t0, \$t0, \$0 | 53: | addi \$t0, \$t0, 0x0001 # contx = contx + 1 |
| | 0x004000a9 | 0x00005043 sra \$t0, \$0, \$10.1 | 54: | bne \$t2, \$0, loop3 # if(t2 != 0) loop3 |
| | 0x004000b3 | 0x08100000 add \$t0, \$t0, \$0 | 55: | j continu1 # go to continu1 |
| | 0x004000b7 | 0x1540fffbne \$t0, \$0, -3 | 56: | sra \$t2, \$t2, 0x0001 # t2 = t2 >> 1 |
| | 0x004000bb | 0x08100000 add \$t0, \$t0, \$0 | 57: | addi \$t0, \$t0, 0x0001 # conty = conty + 1 |
| | 0x004000bf | 0x00005043 sra \$t0, \$0, \$10.1 | 58: | bne \$t2, \$0, loop4 # if(t2 != 0) loop4 |
| | 0x004000c3 | 0x08100000 add \$t0, \$t0, \$0 | 59: | j continu2 # go to continu2 |
| | 0x004000c7 | 0x00006910#t1 \$19 | 60: | mfhi \$s3 # s3 = hi(x*y) |
| | 0x004000cb | 0x00006910#t1 \$19 | 61: | mflo \$s2 # s2 = lo(x*y) |
| | 0x004000cf | 0x00000012#t0 \$18 | 62: | mflo \$s2 # s2 = lo(x*y) |

Labels

| Label | Address |
|------------|------------|
| (global) | 0x00400000 |
| main | 0x00400028 |
| continuel | 0x00400038 |
| continuel2 | 0x00400039 |
| positivo1 | 0x00400050 |
| negativo1 | 0x00400060 |
| loop1 | 0x00400055 |
| loop2 | 0x00400065 |
| loop3 | 0x00400070 |
| loop4 | 0x00400070 |
| negativo2 | 0x00400080 |
| loop5 | 0x00400080 |
| high | 0x00400090 |
| low | 0x00400094 |
| s1 | 0x10010004 |

Registers

| 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 | -1 |
| \$t3 | 11 | 32 |
| \$t4 | 12 | 1 |
| \$t5 | 13 | 0 |
| \$t6 | 14 | 0 |
| \$t7 | 15 | 0 |
| \$t8 | 16 | 75 |
| \$t9 | 17 | -11 |
| \$t10 | 18 | -84 |
| \$t11 | 19 | 0 |
| \$t12 | 20 | 0 |
| \$t13 | 21 | 0 |
| \$t14 | 22 | 0 |
| \$t15 | 23 | 0 |
| \$t16 | 24 | 0 |
| \$t17 | 25 | 0 |
| \$t18 | 26 | 0 |
| \$t19 | 27 | 0 |
| \$gp | 28 | 268468224 |
| \$sp | 29 | 214749540 |
| \$fp | 30 | 0 |
| \$ra | 31 | 0 |
| pc | | 4194456 |
| hi | | -1 |
| lo | | -825 |

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 |

Labels

| Label | Address |
|-----------------------|-------------------------------------|
| 0x10010000 (.data) | 0x10010000 |
| Hexadecimal Addresses | <input checked="" type="checkbox"/> |
| Hexadecimal Values | <input type="checkbox"/> |
| ASCII | <input type="checkbox"/> |

Programa 20:

```

Edit Execute
programa13.asm programma14.asm programma15.asm programma16.asm programma17.asm programma18.asm programma19.asm programma20.asm
programa05.asm programma06.asm programma07.asm programma08.asm programma09.asm programma10.asm programma11.asm programma12.asm
programa01.asm programma02.asm programma03.asm programma04.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 # Associações
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
21    beq $t1, $0, par     # if(x % 2 == 0) go to par
22    mult $s0, $s0
23    mflo $t2
24    mult $t2, $s0
25    mflo $t3
26    mult $t2, $t3
27    mflo $t4
28    sub $t4, $t4, $t3
29    addi $s1, $t4, 0x0001
30    j store
31
32 par:
33    mult $s0, $s0
34    mflo $t2
35    mult $t2, $s0
36    mflo $t3
37    mult $t2, $t2
38    mflo $t4
39    add $t4, $t4, $t3
40    sub $t4, $t4, $t2
41    sub $s1, $t4, $t2
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)

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 | 26850092 |
| \$t1 | 9 | 0 |
| \$t2 | 10 | 4 |
| \$t3 | 11 | 8 |
| \$t4 | 12 | 20 |
| \$t5 | 13 | 0 |
| \$t6 | 14 | 0 |
| \$t7 | 15 | 0 |
| \$t8 | 16 | 2 |
| \$t9 | 17 | 16 |
| \$t10 | 18 | 0 |
| \$t11 | 19 | 0 |
| \$t12 | 20 | 0 |
| \$t13 | 21 | 0 |
| \$t14 | 22 | 0 |
| \$t15 | 23 | 0 |
| \$t16 | 24 | 0 |
| \$t17 | 25 | 0 |
| \$t18 | 26 | 0 |
| \$t19 | 27 | 0 |
| \$k0 | 28 | 26849824 |
| \$k1 | 29 | 2147479548 |
| \$fp | 30 | 0 |
| \$ra | 31 | 0 |
| pc | | 4194396 |
| hi | | 16 |

Text Segment

| Bkt# | Address | Code | Basic | Source |
|------------|------------|-------------------------|-------|--|
| 0x00400000 | 0x00000000 | lui \$t0, 0x4007 | 0 | 18: lui \$t0, 0x1001 # t0 = 0x 1001 0000 |
| 0x00400004 | 0x00000000 | lw \$s0, 0x0000(\$t0) | 0 | 19: lw \$s0, 0x0000(\$t0) # x = MEM[0] |
| 0x00400008 | 0x32090001 | andi \$t1, \$s0, 0x0001 | 0 | 20: andi \$t1, \$s0, 0x0001 # t1 = 0x0000 0001 // ? = 0 se par; ? = 1 se impar |
| 0x0040000c | 0x11200009 | beq \$t1, \$0, par | 0 | 21: beq \$t1, \$0, par # if(x % 2 == 0) go to par |
| 0x00400010 | 0x00000000 | mult \$s0, \$s0 | 0 | 22: mult \$s0, \$s0 # hi_lo = x^2 |
| 0x00400014 | 0x00000000 | mflo \$t2 | 0 | 23: mflo \$t2 # t2 = x^2 |
| 0x00400018 | 0x00000000 | mult \$t2, \$s0 | 0 | 24: mult \$t2, \$s0 # hi_lo = x^3 |
| 0x0040001c | 0x00000512 | mflo \$t3 | 0 | 25: mflo \$t3 # t3 = x^3 |
| 0x00400020 | 0x00000000 | mult \$t2, \$s0 | 0 | 26: mult \$t2, \$s0 # hi_lo = x^5 |
| 0x00400024 | 0x00000000 | mflo \$t4 | 0 | 27: mflo \$t4 # t4 = x^5 |
| 0x00400028 | 0x00000000 | sub \$t4, \$t4, \$t3 | 0 | 28: sub \$t4, \$t4, \$t3 # t4 = x^5 - x^3 |
| 0x0040002c | 0x21000000 | addi \$s1, \$t4, 0x0001 | 0 | 29: addi \$s1, \$t4, 0x0001 # y = x^5 - x^3 + 1 |
| 0x00400030 | 0x00000000 | j store | 0 | 30: j store # go to store |
| 0x00400034 | 0x00200018 | mult \$t0, \$t0 | 0 | 31: mult \$t0, \$t0 # hi_lo = x^2 |
| 0x00400038 | 0x00000000 | mflo \$t2 | 0 | 32: mflo \$t2 # t2 = x^2 |
| 0x0040003c | 0x00000000 | mult \$t2, \$t0 | 0 | 33: mult \$t2, \$t0 # hi_lo = x^3 |
| 0x00400040 | 0x00000512 | mflo \$t3 | 0 | 34: mflo \$t3 # t3 = x^3 |
| 0x00400044 | 0x00000000 | mult \$t2, \$t0 | 0 | 35: mult \$t2, \$t0 # hi_lo = x^5 |
| 0x00400048 | 0x00140000 | mflo \$t4 | 0 | 36: mflo \$t4 # t4 = x^5 |
| 0x0040004c | 0x00000000 | mult \$t2, \$t2 | 0 | 37: mult \$t2, \$t2 # hi_lo = x^4 |
| 0x00400050 | 0x00000000 | mflo \$t4 | 0 | 38: mflo \$t4 # t4 = x^4 |
| 0x00400054 | 0x00000000 | add \$t4, \$t4, \$t3 | 0 | 39: add \$t4, \$t4, \$t3 # t4 = x^4 + x^3 |
| 0x00400058 | 0x00000000 | sub \$t4, \$t4, \$t2 | 0 | 40: sub \$t4, \$t4, \$t2 # t4 = x^4 + x^3 - x^2 |
| 0x0040005c | 0x01080022 | sub \$s1, \$t4, \$t0 | 0 | 41: sub \$s1, \$t4, \$t0 # y = x^4 + x^3 - x^2 + 1 |
| 0x00400060 | 0x00000000 | sw \$s1, 0x0004(\$t0) | 0 | 42: sw \$s1, 0x0004(\$t0) # MEM[1] = y |

Data Segment

| Address | Value (+0) | Value (+4) | Value (+8) | Value (+c) | Value (+10) | Value (+14) | Value (+18) | Value (+1c) |
|------------|------------|------------|------------|------------|-------------|-------------|-------------|-------------|
| 0x10010000 | 0 | 16 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010004 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010008 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x1001000c | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010010 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010014 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010018 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x1001001c | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010020 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010024 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010028 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x1001002c | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010030 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010034 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010038 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x1001003c | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010040 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010044 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010048 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x1001004c | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010050 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010054 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010058 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x1001005c | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010060 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010064 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010068 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x1001006c | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010070 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010074 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010078 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x1001007c | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010080 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010084 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010088 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x1001008c | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010090 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010094 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010098 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x1001009c | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x100100a0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x100100a4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x100100a8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x100100ac | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x100100b0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x100100b4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x100100b8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x100100bc | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x100100c0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x100100c4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x100100c8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x100100cc | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x100100d0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x100100d4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x100100d8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x100100dc | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x100100e0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x100100e4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x100100e8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x100100ec | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x100100f0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x100100f4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x100100f8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x100100fc | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010100 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010104 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010108 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x1001010c | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010110 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010114 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010118 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x1001011c | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010120 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010124 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010128 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x1001012c | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010130 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010134 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010138 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x1001013c | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010140 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010144 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010148 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x1001014c | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010150 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010154 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010158 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x1001015c | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010160 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010164 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010168 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x1001016c | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010170 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010174 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010178 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x1001017c | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010180 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010184 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010188 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x1001018c | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010190 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010194 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010198 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x1001019c | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x100101a0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x100101a4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x100101a8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x100101ac | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x100101b0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x100101b4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x1 | | | | | | | | |

File Edit Run Settings Tools Help

Run speed at max (no interaction)

```

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 28
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 # Associações
11 # x -> $s0; y -> $s1
12
13 # inicio
14 .text
15 .globl main
16
17 main:
18    lui    $t0, 0x1001      # t0 = 0x1001 0000
19    lw     $s0, 0x0000($t0)  # x = MEM[0]
20    andi   $t1, $s0, 0x0001 # t1 = 0x0000 0001 // ? = 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
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    addi   $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)

| Text Segment | | |
|--------------|------------|--|
| Bkpt | Address | Code |
| | 0x00400000 | ui \$t0, 0x1001 # t0 = 0x1001 0000 |
| | 0x00400004 | lw \$s0, 0x0000(\$t0) # x = MEM[0] |
| | 0x00400008 | andi \$t1, \$s0, 0x0001 # t1 = 0x0000 0001 // ? = 0 se par; ? = 1 se impar |
| | 0x0040000c | beq \$t1, \$0, par # if(x % 2 == 0) go to par |
| | 0x00400010 | mult \$s0, \$s0 # hi_lo = x^2 |
| | 0x00400014 | mflo \$t2 # t2 = x^2 |
| | 0x00400018 | mult \$t2, \$s0 # hi_lo = x^3 |
| | 0x0040001c | mflo \$t3 # t3 = x^3 |
| | 0x00400020 | mult \$t2, \$t3 # hi_lo = x^5 |
| | 0x00400024 | mflo \$t4 # t4 = x^5 |
| | 0x00400028 | sub \$t4, \$t4, \$t3 # t4 = x^5 - x^3 |
| | 0x0040002c | addi \$s1, \$t4, 0x0001 # y = x^5 - x^3 + 1 |
| | 0x00400030 | j store # go to store |
| | 0x00400034 | mult \$s0, \$s0 # hi_lo = x^2 |
| | 0x00400038 | mflo \$t3 # t3 = x^2 |
| | 0x0040003c | mult \$t2, \$s0 # hi_lo = x^3 |
| | 0x00400040 | mflo \$t4 # t4 = x^3 |
| | 0x00400044 | mult \$t2, \$t4 # hi_lo = x^4 |
| | 0x00400048 | mflo \$t5 # t5 = x^4 |
| | 0x0040004c | sub \$t5, \$t5, \$t3 # t5 = x^4 - x^3 |
| | 0x00400050 | sub \$t2, \$t2, \$t1 # t2 = x^4 - x^3 - x^2 |
| | 0x00400054 | sub \$s1, \$t2, \$t1 # y = x^4 - x^3 - 2*x^2 |
| | 0x00400058 | sw \$s1, 0x0004(\$t0) # MEM[1] = y |

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

| Registers | | | |
|-----------|--------|-----------|-----------|
| Name | Number | Value | Value |
| \$zero | 0 | 0 | 0 |
| \$t0 | 1 | 0 | 0 |
| \$t1 | 2 | 0 | 0 |
| \$t2 | 3 | 0 | 0 |
| \$t3 | 4 | 0 | 0 |
| \$t4 | 5 | 0 | 0 |
| \$t5 | 6 | 0 | 0 |
| \$t6 | 7 | 0 | 0 |
| \$t7 | 8 | 26850996 | 26850996 |
| \$t8 | 9 | 1 | 1 |
| \$t9 | 10 | 0 | 0 |
| \$t10 | 11 | 125 | 125 |
| \$t11 | 12 | 3006 | 3006 |
| \$t12 | 13 | 0 | 0 |
| \$t13 | 14 | 0 | 0 |
| \$t14 | 15 | 0 | 0 |
| \$t15 | 16 | 5 | 5 |
| \$t16 | 17 | 3001 | 3001 |
| \$t17 | 18 | 0 | 0 |
| \$t18 | 19 | 0 | 0 |
| \$t19 | 20 | 0 | 0 |
| \$t20 | 21 | 0 | 0 |
| \$t21 | 22 | 0 | 0 |
| \$t22 | 23 | 0 | 0 |
| \$t23 | 24 | 0 | 0 |
| \$t24 | 25 | 0 | 0 |
| \$t25 | 26 | 0 | 0 |
| \$t26 | 27 | 0 | 0 |
| \$t27 | 28 | 26846924 | 26846924 |
| \$sp | 29 | 214747948 | 214747948 |
| \$t29 | 30 | 0 | 0 |
| \$t30 | 31 | 0 | 0 |
| \$t31 | | 4194396 | 4194396 |
| \$t32 | | 0 | 0 |
| \$t33 | | 3125 | 3125 |

| 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 |
| 0x10010004 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010008 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x1001000c | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010010 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010014 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010018 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x1001001c | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010020 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010024 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010028 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x1001002c | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010030 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010034 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010038 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x1001003c | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010040 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010044 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010048 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x1001004c | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010050 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010054 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10010058 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Programa 21:

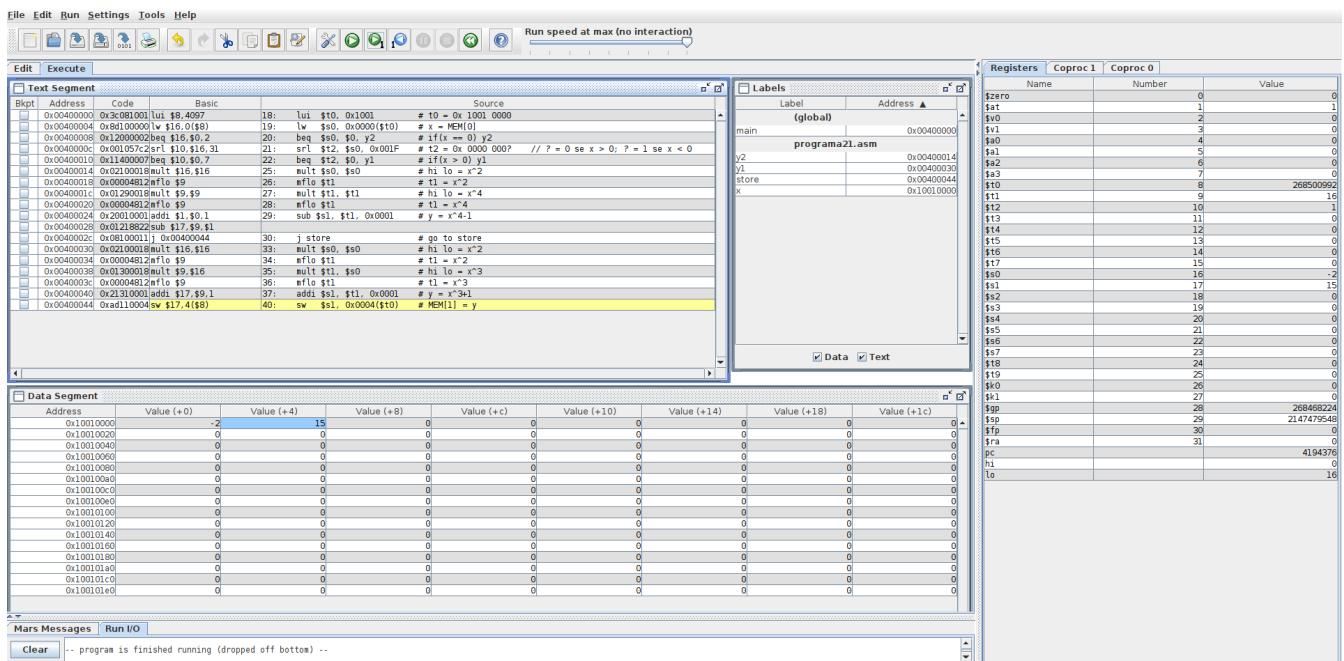
Edit Execute

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 programma11.asm programma12.asm programma13.asm
 programma01.asm programma02.asm programma03.asm programma04.asm programma05.asm

```

1 # Associações
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

```



1 # Associações
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: 5
35 # fim
36
37
38
39
40
41
42
43
44
45
46
47

