

Parte 1:

1. O que é um arquivo fonte?

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

2. O que é um registrador?

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

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

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

4. Quantos bits há em cada instrução de máquina MIPS?

- A. 8
- B. 16
- C. 32
- D. instruções diferentes possuem diferentes comprimentos.

5. O que é o contador de programa?

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

6. Ao executarmos uma instrução, quanto será adicionado ao contador de programa?

- A. 1
- B. 2
- C. 4
- D. 8

7. O que é uma diretiva, tal como a diretiva .text?

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

8. O que é um endereço simbólico?

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

9. Em qual endereço o simulador SPIM coloca a primeira instrução de máquina quando ele está sendo executado?

- A. 0x00000000
- B. 0x00400000
- C. 0x10000000
- D. 0xFFFFFFFF

10. Algumas instruções de máquina possuem uma constante como um dos operandos. Como é chamado tal operando?

- A. operando imediato
- B. operando embutido
- C. operando binário
- D. operando de máquina

11. Como é chamada uma operação lógica executada entre bits de cada coluna dos operandos para produzir um bit de resultado para cada coluna?

- A. operação lógica
- B. operação bitwise
- C. operação binária
- D. operação coluna

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

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

13. Dezesseis bits de dados de uma instrução de ori são usados como um operando imediato. Durante execução, o que deve ser feito primeiro?

- A. Os dados são estendidos em zero à direita por 16 bits.
- B. Os dados são estendidos em zero à esquerda por 16 bits.
- C. Nada precisa ser feito.
- D. Apenas 16 bits são usados pelo outro operando.

14. Qual das instruções seguintes armazenam no registrador \$5 um padrão de bits que representa positivo 48?

- A. ori \$5,\$0,0x48
- B. ori \$5,\$5,0x48
- C. ori \$5,\$0,48
- D. ori \$0,\$5,0x48

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

- A. Não.
- B. Sim.

16. Qual das instruções seguintes limpa todos os bits no registrador \$8 com exceção do byte de baixa ordem que fica inalterado?

- A. ori \$8,\$8,0xFF
- B. ori \$8,\$0,0x00FF
- C. xori \$8,\$8,0xFF
- D. andi \$8,\$8,0xFF

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

- A. Todos os bits em zero.
- B. Todos os bits em um.
- C. O padrão original utilizado.
- D. O resultado é o contrário do original.

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

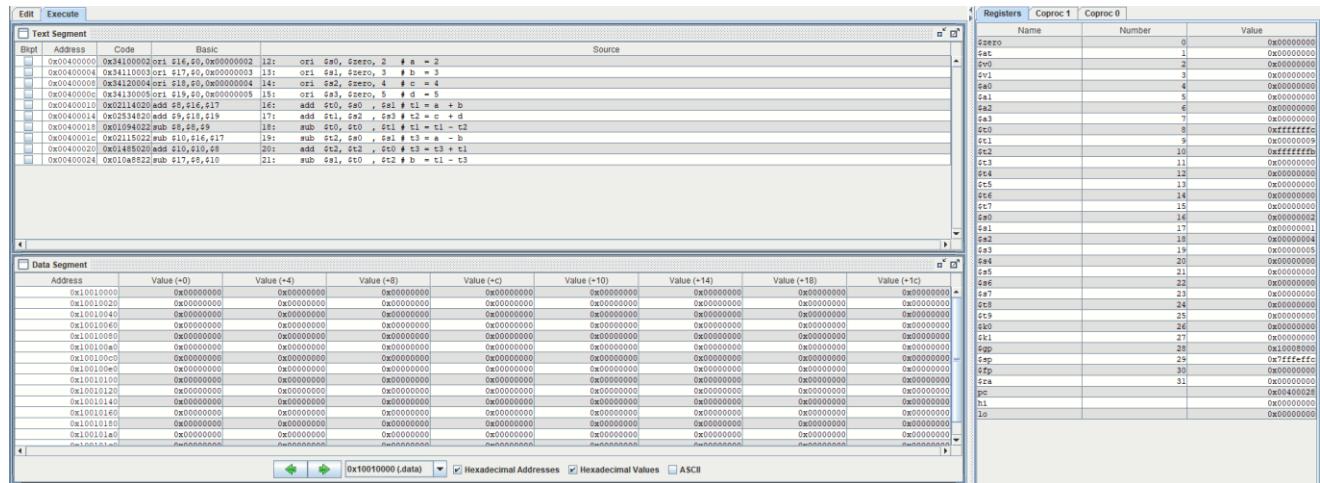
- A. Não. Diferentes de instruções de máquina possuem campos diferentes.
- B. Não. Cada instrução de máquina é completamente diferente de qualquer outra.
- C. Sim. Todas as instruções de máquina têm os mesmos campos na mesma ordem.
- D. Sim. Todas as instruções de máquina têm os mesmos campos, mas eles podem estar em ordens diferentes.

Programa 01:

```

1  # a =2;
2  # b =3;
3  # c =4;
4  # d =5;
5  # x = (a+b) - (c+d);
6  # y = a - b + x;
7  # b = x - y;
8  # add, addi, sub, lógicas
9
10 .text
11 main:
12     ori $s0, $zero, 2    # a = 2
13     ori $s1, $zero, 3    # b = 3
14     ori $s2, $zero, 4    # c = 4
15     ori $s3, $zero, 5    # d = 5
16     add $t0, $s0 , $s1 # t1 = a + b
17     add $t1, $s2 , $s3 # t2 = c + d
18     sub $t0, $t0 , $t1 # t1 = t1 - t2
19     sub $t2, $s0 , $s1 # t3 = a - b
20     add $t2, $t2 , $t0 # t3 = t3 + t1
21     sub $s1, $t0 , $t2 # b = t1 - t3
22
23 # EOF

```



Programa 02:

The screenshot shows a debugger interface with several windows:

- Text Segment:** Displays the assembly code for P02.asm.
- Registers:** Shows the state of various寄存器 (Registers) from \$zero to \$t9.
- Data Segment:** Shows the state of memory locations from \$10010000 to \$10011000.
- Stack:** Shows the stack starting at address \$10010000.

```

P02.asm

1 # x = 1;
2 # y = 5*x + 15;
3 # add, addi, sub, lógicas
4
5 .text
6 main:
7     ori $s0, $zero, 1    # x = 1
8     add $t1, $s0 , $s0 # t1 = x + x = 2x
9     add $t1, $t1 , $t1 # t1 = 2x + 2x = 4x
10    add $t1, $t1 , $s0 # t1 = 4x + x = 5x
11    addi $s1, $t1 , 15 # y = t1 + 15
12
13 # EOF

```

Programa 03:

P03.asm

```

1  # x = 3;
2  # y = 4 ;
3  # z = ( 15*x + 67*y)*4
4  # add, addi, sub, lógicas

5

6 .text
7 main:

8

9     ori $s0, $zero, 3      # x = 3
10    ori $s1, $zero, 4      # y = 4
11    # 15 * x
12    add $t1, $s0 , $s0 # t1 = x + x = 2x
13    add $t1, $t1 , $t1 # t1 = 2x + 2x = 4x
14    add $t1, $t1 , $t1 # t1 = 4x + 4x = 8x
15    add $t1, $t1 , $t1 # t1 = 8x + 8x = 16x
16    sub $t1, $t1 , $s0 # t1 = 16x - x = 15x
17    # 67 * y
18    add $t2, $s1 , $s1 # t2 = y + y = 2y
19    add $t2, $t2 , $t2 # t2 = 2y + 2y = 4y
20    add $t2, $t2 , $t2 # t2 = 4y + 4y = 8y
21    add $t2, $t2 , $t2 # t2 = 8y + 8y = 16y
22    add $t2, $t2 , $t2 # t2 = 16y + 16y = 32y
23    add $t2, $t2 , $t2 # t2 = 32y + 32y = 64y
24    add $t2, $t2 , $s1 # t2 = 64y + y = 65y
25    add $t2, $t2 , $s1 # t2 = 65y + y = 66y
26    add $t2, $t2 , $s1 # t2 = 66y + y = 67y
27    # 15x + 67y
28    add $t3, $t1 , $t2 # t3 = t1 + t2 = a
29    # ( 15x + 67y ) * 4
30    add $t3, $t3 , $t3 # t3 = s + a = 2a
31    add $t3, $t3 , $t3 # t3 = 2a + 2a = 4a
32    add $s2, $zero, $t3 # z = t3
33
34 # END

```

Text Segment

| Bxpl | Address | Codr | Basic | Source |
|------|------------|------------|-------------------------|------------------------|
| | 0x00400000 | 0x34100003 | ori \$t1,\$0,0x00000003 | 9: ori \$s0,\$zero, 3 |
| | 0x00400004 | 0x34100004 | ori \$t1,\$0,0x00000004 | 10: ori \$s1,\$zero, 4 |
| | 0x00400008 | 0x02104040 | add \$t1,\$t1,\$t1 | 11: add \$t1,\$s0,\$t1 |
| | 0x00400010 | 0x02104040 | add \$t1,\$t1,\$t1 | 12: add \$t1,\$t1,\$t1 |
| | 0x00400012 | 0x02104040 | add \$t1,\$t1,\$t1 | 13: add \$t1,\$t1,\$t1 |
| | 0x00400014 | 0x02104040 | add \$t1,\$t1,\$t1 | 14: add \$t1,\$t1,\$t1 |
| | 0x00400016 | 0x02104040 | add \$t1,\$t1,\$t1 | 15: add \$t1,\$t1,\$t1 |
| | 0x00400018 | 0x02104042 | sub \$t1,\$t1,\$s0 | 16: sub \$t1,\$t1,\$s0 |
| | 0x00400020 | 0x02104040 | add \$t1,\$t1,\$t1 | 17: add \$t1,\$t1,\$t1 |
| | 0x00400022 | 0x02104040 | add \$t1,\$t1,\$t1 | 18: add \$t1,\$t1,\$t1 |
| | 0x00400024 | 0x02104040 | add \$t1,\$t1,\$t1 | 19: add \$t1,\$t1,\$t1 |
| | 0x00400026 | 0x02104040 | add \$t1,\$t1,\$t1 | 20: add \$t2,\$t2,\$t2 |
| | 0x00400028 | 0x02104040 | add \$t1,\$t1,\$t1 | 21: add \$t2,\$t2,\$t2 |
| | 0x00400030 | 0x02104040 | add \$t1,\$t1,\$t1 | 22: add \$t2,\$t2,\$t2 |
| | 0x00400032 | 0x02104040 | add \$t1,\$t1,\$t1 | 23: add \$t2,\$t2,\$t2 |
| | 0x00400034 | 0x02104040 | add \$t1,\$t1,\$t1 | 24: add \$t2,\$t2,\$t2 |
| | 0x00400036 | 0x02104040 | add \$t1,\$t1,\$t1 | 25: add \$t2,\$t2,\$t2 |
| | 0x00400038 | 0x02104040 | add \$t1,\$t1,\$t1 | 26: add \$t2,\$t2,\$t2 |
| | 0x00400040 | 0x02104040 | add \$t1,\$t1,\$t1 | 27: add \$t2,\$t2,\$t2 |

Data Segment

| Address | Value (+0) | Value (+4) | Value (+8) | Value (+C) | Value (+10) | Value (+14) | Value (+18) | Value (+1c) |
|------------|------------|------------|------------|------------|-------------|-------------|-------------|-------------|
| 0x10010000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010020 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010040 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010060 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010080 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x100100a0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x100100c0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x100100e0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010100 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010120 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010140 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010160 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010180 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x100101a0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x100101c0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |

Programa 04:

P04.asm

```

1  # x = 3;
2  # y = 4 ;
3  # z = ( 15*x + 67*y) *4
4  # sll, srl, sra
5
6 .text
7 main:
8     ori $s0, $zero, 3      # x = 3
9     ori $s1, $zero, 4      # y = 4
10    # 15*x
11    sll $t1, $s0 , 4      #
12    sub $t1, $t1 , $s0 # t1 = t1 - x
13    # 67*z
14    sll $t2, $s1 , 6      #
15    add $t2, $t2 , $s1 # 
16    add $t2, $t2 , $s1 # 
17    add $t2, $t2 , $s1 #
18
19 # EOF

```

| Text Segment | | | | | Data Segment | | | | |
|--------------|------------|----------------------|---------------|--|--------------|--------|--------|------------|--|
| Brpt | Address | Code | Source | | | Name | Number | Value | |
| | 0x00400000 | lw \$t1, 0(\$a1) | | | | \$zero | 0 | 0x00000000 | |
| | 0x00400004 | ori \$t1, \$zero, 3 | # x = 3 | | | \$at | 1 | 0x00000001 | |
| | 0x00400008 | lw \$t1, 4(\$a1) | # y = 4 | | | \$v0 | 2 | 0x00000000 | |
| | 0x0040000c | ori \$t1, \$zero, 4 | # | | | \$v1 | 3 | 0x00000000 | |
| | 0x00400010 | lw \$t1, 12(\$a1) | # | | | \$a0 | 4 | 0x00000000 | |
| | 0x00400014 | sll \$t1, \$t1, 4 | # | | | \$a1 | 5 | 0x00000000 | |
| | 0x00400018 | sub \$t1, \$t1, \$s0 | # t1 = t1 - x | | | \$a2 | 6 | 0x00000000 | |
| | 0x0040001c | lw \$t1, 16(\$a1) | # | | | \$a3 | 7 | 0x00000000 | |
| | 0x00400020 | sll \$t2, \$s1, 6 | # | | | \$s0 | 8 | 0x00000000 | |
| | 0x00400024 | add \$t2, \$t2, \$s1 | # | | | \$t1 | 9 | 0x00000000 | |
| | 0x00400028 | add \$t2, \$t2, \$s1 | # | | | \$t2 | 10 | 0x00000000 | |
| | 0x0040002c | add \$t2, \$t2, \$s1 | # | | | \$t3 | 11 | 0x00000000 | |
| | 0x00400030 | add \$t2, \$t2, \$s1 | # | | | \$t4 | 12 | 0x00000000 | |
| | 0x00400034 | add \$t2, \$t2, \$s1 | # | | | \$t5 | 13 | 0x00000000 | |
| | 0x00400038 | add \$t2, \$t2, \$s1 | # | | | \$t6 | 14 | 0x00000000 | |
| | 0x00400040 | add \$t2, \$t2, \$s1 | # | | | \$t7 | 15 | 0x00000000 | |
| | 0x00400044 | add \$t2, \$t2, \$s1 | # | | | \$s0 | 16 | 0x00000000 | |
| | 0x00400048 | add \$t2, \$t2, \$s1 | # | | | \$a2 | 17 | 0x00000000 | |
| | 0x00400052 | add \$t2, \$t2, \$s1 | # | | | \$a3 | 18 | 0x00000000 | |
| | 0x00400056 | add \$t2, \$t2, \$s1 | # | | | \$s1 | 19 | 0x00000000 | |
| | 0x00400060 | add \$t2, \$t2, \$s1 | # | | | \$t1 | 20 | 0x00000000 | |
| | 0x00400064 | add \$t2, \$t2, \$s1 | # | | | \$t2 | 21 | 0x00000000 | |

Programa 05:

P05.asm

```

1  # x = 100000 = 0x186A0;
2  # y = 200000 = 0x30D40;
3  # z = x + y;
4  # sll, srl e sra

5

6 .text
7 main:
8     ori $t1, $zero, 0x186A # t1 = 0x186A
9     sll $s1, $t1 , 4      # x = t1 << 4b
10    ori $t2, $zero, 0x30D4 # t2 = 0x30D4
11    sll $s2, $t2 , 4      # y = t2 << 4b
12    add $s3, $s1 , $s2    # z = x + y
13
14 # EOF

```

| Text Segment | | |
|--------------|------------|---|
| Begr. | Address | Code |
| | 0x00400000 | 0x3409104a ori \$t1,\$zero,0x186A |
| | 0x00400004 | 0x00009300 sll \$s1,\$t1,4 # x = t1 << 4b |
| | 0x00400008 | 0x00009300 sll \$s2,\$t2,4 # y = t2 << 4b |
| | 0x0040000c | 0x0000a910 add \$s3,\$s1,\$s2 # z = x + y |
| | 0x00400010 | 0x02329820 add \$s3,\$s1,\$s2 # z = x + y |

| Data Segment | | | | | | | | |
|--------------|------------|------------|------------|------------|-------------|-------------|-------------|-------------|
| Address | Value (+0) | Value (+4) | Value (+8) | Value (+c) | Value (+10) | Value (+14) | Value (+18) | Value (+1c) |
| 0x10010000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010020 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010040 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010060 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010080 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x100100a0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x100100c0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x100100e0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010100 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010120 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010140 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010160 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010180 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |

| Name | Number | Value |
|--------|--------|------------|
| \$zero | 0 | 0x00000000 |
| \$t1 | 1 | 0x00000000 |
| \$v0 | 2 | 0x00000000 |
| \$v1 | 3 | 0x00000000 |
| \$a0 | 4 | 0x00000000 |
| \$a1 | 5 | 0x00000000 |
| \$a2 | 6 | 0x00000000 |
| \$a3 | 7 | 0x00000000 |
| \$t2 | 8 | 0x00000000 |
| \$t3 | 9 | 0x00000000 |
| \$t4 | 10 | 0x00000000 |
| \$t5 | 11 | 0x00000000 |
| \$t6 | 12 | 0x00000000 |
| \$t7 | 13 | 0x00000000 |
| \$t8 | 14 | 0x00000000 |
| \$t9 | 15 | 0x00000000 |
| \$t10 | 16 | 0x00000000 |
| \$t11 | 17 | 0x00001fa5 |
| \$s2 | 18 | 0x00030d40 |
| \$s3 | 19 | 0x000493e0 |
| \$s4 | 20 | 0x00000000 |
| \$t4t | 21 | 0x00000000 |
| \$t6t | 22 | 0x00000000 |
| \$t7t | 23 | 0x00000000 |
| \$t8t | 24 | 0x00000000 |
| \$t9t | 25 | 0x00000000 |
| \$k0 | 26 | 0x00000000 |
| \$k1 | 27 | 0x00000000 |
| \$k2 | 28 | 0x00000000 |
| \$t9p | 29 | 0xffffffff |
| \$fp | 30 | 0x00000000 |
| \$ra | 31 | 0x00000000 |
| \$gp | 32 | 0x00000000 |
| \$hi | 33 | 0x00000000 |
| \$lo | 34 | 0x00000000 |

Programa 06:

P06.asm

```

1 # x = o maior inteiro possível;
2 # y = 300000 = 493e0;
3 # z = x - 4y
4 # sll, srl e sra
5
6 .text
7 main:
8     ori $s0, $zero, 0x7FFF # x = maior inteiro de 16 bits positivo
9     ori $t0, $zero, 0x493E # t1 = 0x493E
10    sll $s1, $t0, 4      # y = t1 << 4b
11    sll $t1, $s1, 2      # t1 = y << 2b
12    sub $s2, $s0, $t1    # z = x - t1
13
14 # EOF

```

Text Segment

| Beg. | Address | Code | Basic | Source |
|------------|------------|--------------------------|---|---|
| 0x00400000 | 0x34107fff | ori \$t0, \$zero, 0x7FFF | # x = maior inteiro de 16 bits positivo | 8: ori \$s0, \$zero, 0x7FFF # x = maior inteiro de 16 bits positivo |
| 0x00400004 | 0x3410493e | ori \$t0, \$zero, 0x493E | # t1 = 0x493E | 9: ori \$t0, \$zero, 0x493E |
| 0x00400008 | 0x00008990 | sll \$s1, \$t0, 4 | # y = t1 << 4b | 10: sll \$s1, \$t0, 4 # y = t1 << 4b |
| 0x0040000c | 0x00000002 | sll \$t1, \$s1, 2 | # t1 = y << 2b | 11: sll \$t1, \$s1, 2 # t1 = y << 2b |
| 0x00400010 | 0x02099022 | sub \$s2, \$s0, \$t1 | # z = x - t1 | 12: sub \$s2, \$s0, \$t1 # z = x - t1 |

Data Segment

| Address | Value (+0) | Value (+4) | Value (+8) | Value (+c) | Value (+10) | Value (+14) | Value (+18) | Value (+1c) |
|------------|------------|------------|------------|------------|-------------|-------------|-------------|-------------|
| 0x10010000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010020 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010040 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010060 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010080 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x100100a0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x100100c0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x100100e0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010100 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010120 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010140 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010160 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010180 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x100101a0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x100101c0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |

Registers:

| Name | Number | Value |
|--------|--------|------------|
| \$zero | 0 | 0x00000000 |
| \$at | 1 | 0x00000000 |
| \$v0 | 2 | 0x00000000 |
| \$v1 | 3 | 0x00000000 |
| \$a0 | 4 | 0x00000000 |
| \$a1 | 5 | 0x00000000 |
| \$a2 | 6 | 0x00000000 |
| \$a3 | 7 | 0x00000000 |
| \$t0 | 8 | 0x0000493e |
| \$t1 | 9 | 0x000124f5 |
| \$t2 | 10 | 0x00000000 |
| \$t3 | 11 | 0x00000000 |
| \$t4 | 12 | 0x00000000 |
| \$t5 | 13 | 0x00000000 |
| \$t6 | 14 | 0x00000000 |
| \$t7 | 15 | 0x00000000 |
| \$s0 | 16 | 0x0000ffff |
| \$s1 | 17 | 0xffffffff |
| \$s2 | 18 | 0xffffffff |
| \$s3 | 19 | 0x00000000 |
| \$s4 | 20 | 0x00000000 |
| \$s5 | 21 | 0x00000000 |
| \$s6 | 22 | 0x00000000 |
| \$s7 | 23 | 0x00000000 |
| \$t8 | 24 | 0x00000000 |
| \$t9 | 25 | 0x00000000 |
| \$k0 | 26 | 0x00000000 |
| \$k1 | 27 | 0x00000000 |
| \$gp | 28 | 0x00000000 |
| \$sp | 29 | 0xffffffff |
| \$fp | 30 | 0x00000000 |
| \$ra | 31 | 0x00000000 |
| pc | | 0x00400024 |
| hi | | 0x00000000 |
| lo | | 0x00000000 |

Programa 07:

P07.asm

```

1 # Considere a seguinte instrução iniciando um programa:
2 #      ori $8, $0, 0x01
3 # Usando apenas instruções reg-reg lógicas e/ou instruções de deslocamento (sll, srl, sra), continuar o
4 # programa de forma que ao final, tenhamos o seguinte conteúdo no registrador $8:
5 #      $8 = 0xFFFFFFFF
6
7 .text
8 main:
9     solucao_1:
10    ori $8, $zero, 0x01    # $8 = 0x00000001
11    ori $8, $8 , 0xFFFF # $8 = 0x0000FFFF
12    sll $8, $8 , 16    # $8 = 0xFFFF0000
13    ori $8, $8 , 0xFFFF # $8 = 0xFFFFFFFF
14
15     solucao_2:
16     ori $8, $zero, 0x01    # $8 = 0x00000001
17     addi $8, $8 , -2     # $8 = 0xFFFFFFFF
18
19 # EOF

```

Text Segment

| Byte | Address | Code | Basic | Source |
|------------|--------------|---------------------------|---------------------|--|
| 0x04000000 | 0x34080001 | ori \$8,\$0,0x00000001 | ori \$8,\$zero,0x01 | 10: ori \$8,\$zero,0x01 # \$8 = 0x00000001 |
| 0x04000004 | 0x3508ffff | ori \$8,\$8,0x0000ffff | ori \$8,\$8,0xFFFF | 11: ori \$8,\$8,0xFFFF # \$8 = 0x0000FFFF |
| 0x04000008 | 0x35008440 | sll \$8,\$8,16 | sll \$8,\$8,16 | 12: sll \$8,\$8,16 # \$8 = 0xFFFF0000 |
| 0x0400000C | 0x3500ffff | ori \$8,\$8,0x0000ffff | ori \$8,\$8,0xFFFF | 13: ori \$8,\$8,0xFFFF # \$8 = 0xFFFFFFFF |
| 0x04000010 | 0x31008001 | ori \$8,\$zero,0x00000001 | ori \$8,\$zero,0x01 | 14: ori \$8,\$zero,0x01 # \$8 = 0x00000001 |
| 0x04000014 | 0x3100fffffe | addi \$8,\$8,-2 | addi \$8,\$8,-2 | 17: addi \$8,\$8,-2 # \$8 = 0xFFFFFFFF |

Data Segment

| Address | Value (+0) | Value (+4) | Value (+8) | Value (+c) | Value (+10) | Value (+14) | Value (+18) | Value (+1c) |
|------------|------------|------------|------------|------------|-------------|-------------|-------------|-------------|
| 0x10010000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010020 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010040 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010060 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010080 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x100100A0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x100100C0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x100100E0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010100 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010120 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010140 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010160 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010180 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x100101A0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x100101C0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |

0x10010000 (data) Hexadecimal Addresses Hexadecimal Values ASCII

| Name | Number | Value |
|--------|--------|------------|
| \$zero | 0 | 0x00000000 |
| \$at | 1 | 0x00000000 |
| \$v0 | 2 | 0x00000000 |
| \$v1 | 3 | 0x00000000 |
| \$a0 | 4 | 0x00000000 |
| \$a1 | 5 | 0x00000000 |
| \$a2 | 6 | 0x00000000 |
| \$a3 | 7 | 0x00000000 |
| \$t0 | 8 | 0xffffffff |
| \$t1 | 9 | 0x00000000 |
| \$t2 | 10 | 0x00000000 |
| \$t3 | 11 | 0x00000000 |
| \$t4 | 12 | 0x00000000 |
| \$t5 | 13 | 0x00000000 |
| \$t6 | 14 | 0x00000000 |
| \$t7 | 15 | 0x00000000 |
| \$s0 | 16 | 0x00000000 |
| \$s1 | 17 | 0x00000000 |
| \$s2 | 18 | 0x00000000 |
| \$s3 | 19 | 0x00000000 |
| \$s4 | 20 | 0x00000000 |
| \$s5 | 21 | 0x00000000 |
| \$s6 | 22 | 0x00000000 |
| \$s7 | 23 | 0x00000000 |
| \$t8 | 24 | 0x00000000 |
| \$t9 | 25 | 0x00000000 |
| \$t10 | 26 | 0x00000000 |
| \$t11 | 27 | 0x00000000 |
| \$t12 | 28 | 0x10000000 |
| \$t13 | 29 | 0xffffffff |
| \$t14 | 30 | 0xffffffff |
| \$t15 | 31 | 0x00000000 |
| pc | | 0x04000018 |
| hi | | 0x00000000 |
| lo | | 0x00000000 |

Programa 08:

```

P08.asm
1 # Inicialmente escreva um programa que faça:
2 #     $8 = 0x12345678.
3 # A partir do registrador $8 acima, usando apenas instruções lógicas (or, ori, and, andi, xor, xori)
4 # e instruções de deslocamento (sll, srl e sra), você deverá obter os seguintes valores nos respectivos
5 # registradores:
6 #     $9 = 0x12
7 #     $10 = 0x34
8 #     $11 = 0x56
9 #     $12 = 0x78
10
11 .text
12 main:
13     ori $8, $zero, 0x1234 # $8 = 0x00001234
14     sll $8, $8, 16 # $8 = 0x12340000
15     ori $8, $8, 0x5678 # $8 = 0x12345678
16     sra $9, $8, 24 # $9 = $8 >> 24
17     sra $10, $8, 16 # $10 = $10 >> 16
18     andi $10, $10, 0xFF # $10 = andi( 0x00001234, 0x000000FF ) -> mascarando os bits
19     sra $11, $8, 8 # $11 = $8 >> 8
20     andi $11, $11, 0xFF # $11 = andi( 0x00123456, 0x000000FF ) -> mascarando os bits
21     andi $12, $8, 0xFF # $12 = andi( 0x12345678, 0x000000FF ) -> mascarando os bits
22
23 # EOF

```

Text Segment

| Bp# | Address | Code | Basic | Source |
|-----|------------|-------------------------------------|---|--------|
| | 0x00400000 | 0x345678ori \$8,\$0,0x00001234 | 13: ori \$8, \$zero, 0x1234 # \$8 = 0x00001234 | |
| | 0x00400004 | 0x31000000sll \$8,\$8,16 | 14: sll \$8, \$8, 16 # \$8 = 0x12340000 | |
| | 0x00400008 | 0x355678ori \$8,\$8,0x5678 | 15: ori \$8, \$8, 0x5678 # \$8 = 0x12345678 | |
| | 0x00400010 | 0x30004e03sra \$9,\$8,0x00000010 | 16: sra \$9, \$8, 24 # \$9 = \$8 >> 24 | |
| | 0x00400014 | 0x30005403sra \$10,\$8,0x00000010 | 17: sra \$10, \$8, 16 # \$10 = \$10 >> 16 | |
| | 0x00400018 | 0x30005003sra \$11,\$8,0x00000008 | 18: sra \$11, \$8, 8 # \$11 = \$8 >> 8 | |
| | 0x0040001c | 0x310000ffandi \$11,\$11,0x0000000f | 19: andi \$11, \$11, 0xFF # \$11 = andi(0x00123456, 0x000000FF) -> mascarando os bits | |
| | 0x00400020 | 0x310000ffandi \$12,\$8,0x000000ff | 20: andi \$12, \$8, 0xFF # \$12 = andi(0x12345678, 0x000000FF) -> mascarando os bits | |

Data Segment

| Address | Value (+0) | Value (+4) | Value (+8) | Value (+C) | Value (+10) | Value (+14) | Value (+18) | Value (+1C) | Value (+0) |
|------------|------------|------------|------------|------------|-------------|-------------|-------------|-------------|------------|
| 0x10010000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010020 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010040 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010060 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010080 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x100100a0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x100100c0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x100100e0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010100 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010120 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010140 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010160 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010180 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x100101a0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x100101c0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |

| Name | Number | Value |
|--------|--------|------------|
| \$zero | 0 | 0x00000000 |
| \$at | 1 | 0x00000000 |
| \$v0 | 2 | 0x00000000 |
| \$v1 | 3 | 0x00000000 |
| \$a0 | 4 | 0x00000000 |
| \$a1 | 5 | 0x00000000 |
| \$a2 | 6 | 0x00000000 |
| \$a3 | 7 | 0x00000000 |
| \$t0 | 8 | 0x12345678 |
| \$t1 | 9 | 0x00000012 |
| \$t2 | 10 | 0x00000034 |
| \$t3 | 11 | 0x00000054 |
| \$t4 | 12 | 0x00000078 |
| \$t5 | 13 | 0x00000000 |
| \$t6 | 14 | 0x00000000 |
| \$t7 | 15 | 0x00000000 |
| \$t8 | 16 | 0x00000000 |
| \$t9 | 17 | 0x00000000 |
| \$t10 | 18 | 0x00000000 |
| \$t11 | 19 | 0x00000000 |
| \$t12 | 20 | 0x00000000 |
| \$t13 | 21 | 0x00000000 |
| \$t14 | 22 | 0x00000000 |
| \$t15 | 23 | 0x00000000 |
| \$t16 | 24 | 0x00000000 |
| \$k0 | 25 | 0x00000000 |
| \$k1 | 26 | 0x00000000 |
| \$gp | 27 | 0x00000000 |
| \$sp | 28 | 0x10000000 |
| \$fp | 29 | 0x7fffffe0 |
| \$ta | 30 | 0x00000000 |
| \$pc | 31 | 0x00400024 |
| \$hi | 32 | 0x00000000 |
| \$lo | 33 | 0x00000000 |

Programa 09:

P09.asm

```

7 # x4: .word 17
8 # soma: .word -1
9 # Escrever um programa que leia todos os números, calcule e substitua o valor da variável soma por este valor.
10 # lw, sw

11

12 .data
13     x1: .word 15
14     x2: .word 25
15     x3: .word 13
16     x4: .word 17
17     soma: .word -1

18

19 .text
20 main:
21     ori $t0, $zero, 0x1001 # $t0 = 0x00001001
22     sll $t0, $t0 , 16    # $t0 = 0x10010000 -> onde começa a memoria de dados
23
24     lw $t1, 0 ($t0)      # $t1 = $t0[0] = x1
25     lw $t2, 4 ($t0)      # $t2 = $t0[1] = x2 -> de 4 em 4 bits
26     lw $t3, 8 ($t0)      # $t3 = $t0[2] = x3
27     lw $t4, 12($t0)     # $t4 = $t0[3] = x4
28
29     add $t5, $t1, $t2      # $t5 = x1 + x2
30     add $t5, $t5, $t3      # $t5 = $t5 + x3
31     add $t5, $t5, $t4      # $t5 = $t5 + x4
32
33     sw $t5, 16($t0)      # soma = $t0[4] = $t5
34
35 # EOF

```

| Text Segment | | Data Segment | | | | | | | | | | | | |
|--------------|------------|-------------------------------------|-------|----------------------------|---|------------|------------|------------|------------|-------------|-------------|-------------|-------------|------------|
| Byte | Address | Code | Basic | Source | Address | Value (+0) | Value (+4) | Value (+8) | Value (+c) | Value (+10) | Value (+14) | Value (+18) | Value (+1c) | |
| | 0x00400000 | 0x3401001ori \$t0,\$zero,0x10010000 | 21: | ori \$t0,\$zero,0x10010000 | 0x10010000 | 0x0000000f | 0x00000019 | 0x0000000d | 0x00000011 | 0x00000046 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| | 0x00400004 | 0x000084400all \$t0,\$t0,16 | 22: | sll \$t0,\$t0,16 | # \$t0 = 0x10010000 -> onde começa a memoria de dados | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| | 0x00400008 | 0x8d0090000lw \$t1,0(\$t0) | 24: | lw \$t1,0(\$t0) | # \$t1 = \$t0[0] = x1 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| | 0x0040000c | 0x8d00a0000lw \$t2,4(\$t0) | 25: | lw \$t2,4(\$t0) | # \$t2 = \$t0[1] = x2 -> de 4 em 4 bits | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| | 0x00400010 | 0x8d00b0000lw \$t3,8(\$t0) | 26: | lw \$t3,8(\$t0) | # \$t3 = \$t0[2] = x3 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| | 0x00400014 | 0x8d00c0000lw \$t4,12(\$t0) | 27: | lw \$t4,12(\$t0) | # \$t4 = \$t0[3] = x4 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| | 0x00400018 | 0x0012a620add \$t1,\$t2,\$t1 | 29: | add \$t1,\$t2,\$t1 | # \$t1 = x1 + x2 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| | 0x0040001c | 0x0012a620add \$t1,\$t3,\$t1 | 30: | add \$t1,\$t3,\$t1 | # \$t1 = x1 + x3 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| | 0x00400020 | 0x0012a620add \$t1,\$t4,\$t1 | 31: | add \$t1,\$t4,\$t1 | # \$t1 = x1 + x4 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| | 0x00400024 | 0x0ad0d010sw \$t1,0x00000010(\$t0) | 33: | sw \$t1,0x00000010(\$t0) | # soma = \$t0[4] = \$t5 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |

Programa 10:

```
P10.asm
10
11 .data
12     x: .word 5
13     z: .word 7
14     y: .word 0
15
16 .text
17 main:
18     ori $t0, $zero, 0x1001 # $t0 = 0x00001001
19     sll $t0, $t0 , 16      # $t0 = 0x10010000 -> onde começa a memoria de dados
20
21     lw  $t1, 0($t0)        # $t1 = $t0[0] = x
22     lw  $t2, 4($t0)        # $t2 = $t0[1] = z
23
24     calcular_127x:
25         sll $t3, $t1, 7    # $t3 = x << 7
26         sub $t3, $t3, $t1 # $t3 = $t3 - x
27
28     calcular_65z:
29         sll $t4, $t2, 6    # $t4 = z << 6
30         add $t4, $t4, $t2 # $t4 = $t4 + z
31
32     calcular_exp:
33         add $t4, $t4, $t3 # $t4 = 65z + 127x
34         addi $t4, $t4, 1   # $t4 = $t4 + 1
35
36     sw  $t4, 8($t0)        # y = $t0[2] = $t4
37
38 # EOF
```

| Text Segment | | | | | | | | | | |
|--------------|------------|------------|--|--------|--|---|--|--|--|--|
| Byte | Address | Code | Basic | Source | | | | | | |
| 0x00000000 | 0x00000000 | 0x34001001 | ori \$t0, \$zero, 0x1001 # \$t0 = 0x00001001 | 18: | ori \$t0, \$zero, 0x1001 # \$t0 = 0x00001001 | | | | | |
| 0x00000001 | 0x00000001 | 0x00000000 | | 19: | sll \$t0, \$t0 , 16 | # \$t0 = 0x10010000 -> onde começa a memoria de dados | | | | |
| 0x00000002 | 0x00000002 | 0x00000000 | | 20: | lw \$t1, 0(\$t0) | # \$t1 = \$t0[0] = x | | | | |
| 0x00000003 | 0x00000003 | 0x00000000 | | 21: | lw \$t2, 4(\$t0) | # \$t2 = \$t0[1] = z | | | | |
| 0x00000004 | 0x00000004 | 0x00000000 | | 22: | lw \$t3, 8(\$t0) | # \$t3 = \$t0[2] = \$t4 | | | | |
| 0x00000005 | 0x00000005 | 0x00000000 | | 23: | calcular_127x: | | | | | |
| 0x00000006 | 0x00000006 | 0x00000000 | | 24: | sll \$t3, \$t1, 7 | # \$t3 = x << 7 | | | | |
| 0x00000007 | 0x00000007 | 0x00000000 | | 25: | sub \$t3, \$t3, \$t1 | # \$t3 = \$t3 - x | | | | |
| 0x00000008 | 0x00000008 | 0x00000000 | | 26: | sub \$t3, \$t3, \$t1 | # \$t3 = \$t3 - x | | | | |
| 0x00000009 | 0x00000009 | 0x00000000 | | 27: | calcular_65z: | | | | | |
| 0x0000000A | 0x0000000A | 0x00000000 | | 28: | sll \$t4, \$t2, 6 | # \$t4 = z << 6 | | | | |
| 0x0000000B | 0x0000000B | 0x00000000 | | 29: | add \$t4, \$t4, \$t2 | # \$t4 = \$t4 + z | | | | |
| 0x0000000C | 0x0000000C | 0x00000000 | | 30: | calcular_exp: | | | | | |
| 0x0000000D | 0x0000000D | 0x00000000 | | 31: | add \$t4, \$t4, \$t3 | # \$t4 = 65z + 127x | | | | |
| 0x0000000E | 0x0000000E | 0x00000000 | | 32: | addi \$t4, \$t4, 1 | # \$t4 = \$t4 + 1 | | | | |
| 0x0000000F | 0x0000000F | 0x00000000 | | 33: | addi \$t4, \$t4, 1 | # \$t4 = \$t4 + 1 | | | | |
| 0x00000010 | 0x00000010 | 0x00000000 | | 34: | addi \$t4, \$t4, 1 | # \$t4 = \$t4 + 1 | | | | |
| 0x00000011 | 0x00000011 | 0x00000000 | | 35: | sw \$t4, 8(\$t0) | # y = \$t0[2] = \$t4 | | | | |
| 0x00000012 | 0x00000012 | 0x00000000 | | 36: | | | | | | |
| 0x00000013 | 0x00000013 | 0x00000000 | | 37: | | | | | | |
| 0x00000014 | 0x00000014 | 0x00000000 | | 38: | # EOF | | | | | |

| Data Segment | | | | | | | | | | |
|--------------|------------|------------|------------|------------|-------------|-------------|-------------|-------------|--|--|
| Address | Value (+0) | Value (+4) | Value (+8) | Value (+C) | Value (+10) | Value (+14) | Value (+18) | Value (+1C) | | |
| 0x10010000 | 0x00000005 | 0x00000007 | 0x0000043 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | | |
| 0x10010020 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | | |
| 0x10010040 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | | |
| 0x10010060 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | | |
| 0x10010080 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | | |
| 0x100100A0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | | |
| 0x100100C0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | | |
| 0x100100E0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | | |
| 0x10010100 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | | |
| 0x10010120 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | | |
| 0x10010140 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | | |
| 0x10010160 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | | |
| 0x10010180 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | | |
| 0x100101A0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | | |

Programa 11:

```
P11.asm
1 # Considere o seguinte programa:
2 #      y = x - z + 300000
3 # Faça um programa que calcule o valor de y conhecendo os valores de x e z. Os valores de x e z estão armazenados na
4 # memória e, na posição imediatamente a seguir, o valor de y deverá ser escrito, ou seja:
5 #      .data
6 #      x: .word 100000
7 #      z: .word 200000
8 #      y: .word 0 # esse valor deverá ser sobreescrito após a execução do programa.
9
10 .data
11     x: .word 100000
12     z: .word 200000
13     y: .word 0
14
15 .text
16 main:
17     ori $t0, $zero, 0x1001 # $t0 = 0x00001001
18     sll $t0, $t0 , 16    # $t0 = 0x10010000 -> onde começa a memoria de dados
19
20     lw $t1, 0($t0)        # $t1 = $t0[0] = x
21     lw $t2, 4($t0)        # $t2 = $t0[1] = z
22
23     sub $t3, $t1, $t2      # $t3 = x - z
24     ori $t4, $t4, 0x493E # $t4 = $t4 + 0x493E
25     sll $t4, $t4, 4       # $t4 = $t4 << 4
26     add $t3, $t3, $t4      # $t3 = $t3 + 300000
27
28     sw $t3, 8($t0)        # y = $t0[2] = $t3
29
30 # EOF
```

The screenshot shows a debugger interface with several windows:

- Text Segment:** Displays the assembly code from P11.asm.
- Data Segment:** Displays memory dump starting at address 0x10010000, showing values for x, z, and y.
- Registers:** Shows the state of all general-purpose registers (\$t0 to \$t7, \$s0 to \$s3, \$a0 to \$a3, \$d0 to \$d3).
- Coproc 1:** Shows the state of Coprocessor 1 registers (\$f0 to \$f3).
- Coproc 0:** Shows the state of Coprocessor 0 registers (\$r0 to \$r3).

Programa 12:

```
P12.asm
9 # Crie um programa que implemente a estrutura de dados acima, leia o valor de K, o multiplique por 2 e o reescreva no
10 # local correto conhecendo-se apenas o valor de x.
11
12 # *x      = &p2;
13 # *p2     = &p1;
14 # *p1     = &valor;
15 # *valor  = valor;
16
17 .data
18     x:     .word p2          # Terceiro ponteiro que aponta para 'p2' (**x)
19     p2:    .word p1          # Segundo ponteiro que aponta para 'p1'
20     p1:    .word valor       # Primeiro ponteiro que aponta para 'valor'
21     valor: .word 10          # Inteiro inicial armazenado
22
23 .text
24 main:
25     ori $t0, $zero, 0x1001 # $t0 = 0x00001001
26     sll $t0, $t0 , 16      # $t0 = 0x10010000 -> onde começa a memoria de dados
27
28     lw $t1, 0($t0)        # $t1 = MEM[$t0] -> p2
29     lw $t2, 0($t1)        # $t2 = MEM[$t1] -> p1
30     lw $t3, 0($t2)        # $t3 = MEM[$t2] -> valor
31     lw $t4, 0($t3)        # $t4 = valor
32
33     add $t4, $t4, $t4      # $t4 = k * 2
34
35     sw $t4, 0($t3)        # MEM[$t3] = k * 2
36
37 # EOF
```

| Text Segment | | | | | | Data Segment | | | | |
|--------------|------------|------------------------------------|------------------------------|---|--|--------------|--------|------------|--|--|
| Bp# | Address | Code | Basic | Source | | Name | Number | Value | | |
| | 0x00000000 | 0x34010001lw \$t0,\$t0,0x00001001 | 28: lw \$t0,\$t0,0x00001001 | # \$t0 = 0x00001001 | | \$zero | 0 | 0x00000000 | | |
| | 0x00000004 | 0x44010001lw \$t1,\$t0,0x00000001 | 29: lw \$t1,\$t0,0x00000001 | # \$t1 = 0x10010000 -> onde começa a memoria de dados | | \$t1 | 1 | 0x00000000 | | |
| | 0x00000008 | 0x44010001lw \$t2,\$t1,0x00000001 | 29: lw \$t2,\$t1,0x00000001 | # \$t2 = MEM[\$t1] -> p2 | | \$t2 | 2 | 0x00000000 | | |
| | 0x0000000C | 0x44010001lw \$t3,\$t2,0x00000001 | 29: lw \$t3,\$t2,0x00000001 | # \$t3 = MEM[\$t2] -> p1 | | \$t3 | 3 | 0x00000000 | | |
| | 0x00000010 | 0x44010001lw \$t4,\$t3,0x00000001 | 29: lw \$t4,\$t3,0x00000001 | # \$t4 = MEM[\$t3] -> valor | | \$t4 | 4 | 0x00000000 | | |
| | 0x00000014 | 0x34010001lw \$t5,\$t4,0x00000001 | 30: lw \$t5,\$t4,0x00000001 | # \$t5 = MEM[\$t4] -> k | | \$t5 | 5 | 0x00000000 | | |
| | 0x00000018 | 0x34010001lw \$t6,\$t5,0x00000001 | 30: lw \$t6,\$t5,0x00000001 | # \$t6 = MEM[\$t5] -> k * 2 | | \$t6 | 6 | 0x00000000 | | |
| | 0x00000020 | 0x24010001add \$t4,\$t4,\$t4 | 33: add \$t4,\$t4,\$t4 | # \$t4 = k * 2 | | \$a1 | 7 | 0x00000000 | | |
| | 0x00000024 | 0x34010001sw \$t4,0x00000001(\$t3) | 35: sw \$t4,0x00000001(\$t3) | # MEM[\$t3] = k * 2 | | \$a2 | 8 | 0x00000000 | | |
| | 0x00000028 | | | | | \$t0 | 9 | 0x00000000 | | |
| | 0x0000002C | | | | | \$t1 | 10 | 0x10010000 | | |
| | 0x00000030 | | | | | \$t2 | 11 | 0x10010000 | | |
| | 0x00000034 | | | | | \$t3 | 12 | 0x00000000 | | |
| | 0x00000038 | | | | | \$t4 | 13 | 0x00000000 | | |
| | 0x00000040 | | | | | \$t5 | 14 | 0x00000000 | | |
| | 0x00000044 | | | | | \$t6 | 15 | 0x00000000 | | |
| | 0x00000048 | | | | | \$t7 | 16 | 0x00000000 | | |
| | 0x00000050 | | | | | \$t8 | 17 | 0x00000000 | | |
| | 0x00000054 | | | | | \$a1 | 18 | 0x00000000 | | |
| | 0x00000058 | | | | | \$a2 | 19 | 0x00000000 | | |
| | 0x00000060 | | | | | \$t9 | 20 | 0x00000000 | | |
| | 0x00000064 | | | | | \$a5 | 21 | 0x00000000 | | |
| | 0x00000068 | | | | | \$a6 | 22 | 0x00000000 | | |
| | 0x00000070 | | | | | \$t10 | 23 | 0x00000000 | | |
| | 0x00000074 | | | | | \$t11 | 24 | 0x00000000 | | |
| | 0x00000078 | | | | | \$t12 | 25 | 0x00000000 | | |
| | 0x00000080 | | | | | \$t13 | 26 | 0x00000000 | | |
| | 0x00000084 | | | | | \$t14 | 27 | 0x00000000 | | |
| | 0x00000088 | | | | | \$t15 | 28 | 0x0000ffff | | |
| | 0x00000090 | | | | | \$t16 | 29 | 0x7fffffc | | |
| | 0x00000094 | | | | | \$t17 | 30 | 0x00000000 | | |
| | 0x00000098 | | | | | \$t18 | 31 | 0x00000000 | | |
| | 0x000000A0 | | | | | \$t19 | 32 | 0x00000000 | | |
| | 0x000000A4 | | | | | \$t20 | 33 | 0x00000000 | | |
| | 0x000000A8 | | | | | \$t21 | 34 | 0x00000000 | | |
| | 0x000000B0 | | | | | \$t22 | 35 | 0x00000000 | | |
| | 0x000000B4 | | | | | \$t23 | 36 | 0x00000000 | | |
| | 0x000000B8 | | | | | \$t24 | 37 | 0x00000000 | | |
| | 0x000000C0 | | | | | \$t25 | 38 | 0x00000000 | | |
| | 0x000000C4 | | | | | \$t26 | 39 | 0x00000000 | | |
| | 0x000000C8 | | | | | \$t27 | 40 | 0x00000000 | | |
| | 0x000000D0 | | | | | \$t28 | 41 | 0x00000000 | | |
| | 0x000000D4 | | | | | \$t29 | 42 | 0x00000000 | | |
| | 0x000000D8 | | | | | \$t30 | 43 | 0x00000000 | | |
| | 0x000000E0 | | | | | \$t31 | 44 | 0x00000000 | | |
| | 0x000000E4 | | | | | \$t32 | 45 | 0x00000000 | | |
| | 0x000000E8 | | | | | \$t33 | 46 | 0x00000000 | | |
| | 0x000000F0 | | | | | \$t34 | 47 | 0x00000000 | | |
| | 0x000000F4 | | | | | \$t35 | 48 | 0x00000000 | | |
| | 0x000000F8 | | | | | \$t36 | 49 | 0x00000000 | | |
| | 0x000000F0 | | | | | \$t37 | 50 | 0x00000000 | | |
| | 0x000000F4 | | | | | \$t38 | 51 | 0x00000000 | | |
| | 0x000000F8 | | | | | \$t39 | 52 | 0x00000000 | | |
| | 0x000000F0 | | | | | \$t40 | 53 | 0x00000000 | | |
| | 0x000000F4 | | | | | \$t41 | 54 | 0x00000000 | | |
| | 0x000000F8 | | | | | \$t42 | 55 | 0x00000000 | | |
| | 0x000000F0 | | | | | \$t43 | 56 | 0x00000000 | | |
| | 0x000000F4 | | | | | \$t44 | 57 | 0x00000000 | | |
| | 0x000000F8 | | | | | \$t45 | 58 | 0x00000000 | | |
| | 0x000000F0 | | | | | \$t46 | 59 | 0x00000000 | | |
| | 0x000000F4 | | | | | \$t47 | 60 | 0x00000000 | | |
| | 0x000000F8 | | | | | \$t48 | 61 | 0x00000000 | | |
| | 0x000000F0 | | | | | \$t49 | 62 | 0x00000000 | | |
| | 0x000000F4 | | | | | \$t50 | 63 | 0x00000000 | | |
| | 0x000000F8 | | | | | \$t51 | 64 | 0x00000000 | | |
| | 0x000000F0 | | | | | \$t52 | 65 | 0x00000000 | | |
| | 0x000000F4 | | | | | \$t53 | 66 | 0x00000000 | | |
| | 0x000000F8 | | | | | \$t54 | 67 | 0x00000000 | | |
| | 0x000000F0 | | | | | \$t55 | 68 | 0x00000000 | | |
| | 0x000000F4 | | | | | \$t56 | 69 | 0x00000000 | | |
| | 0x000000F8 | | | | | \$t57 | 70 | 0x00000000 | | |
| | 0x000000F0 | | | | | \$t58 | 71 | 0x00000000 | | |
| | 0x000000F4 | | | | | \$t59 | 72 | 0x00000000 | | |
| | 0x000000F8 | | | | | \$t60 | 73 | 0x00000000 | | |
| | 0x000000F0 | | | | | \$t61 | 74 | 0x00000000 | | |
| | 0x000000F4 | | | | | \$t62 | 75 | 0x00000000 | | |
| | 0x000000F8 | | | | | \$t63 | 76 | 0x00000000 | | |
| | 0x000000F0 | | | | | \$t64 | 77 | 0x00000000 | | |
| | 0x000000F4 | | | | | \$t65 | 78 | 0x00000000 | | |
| | 0x000000F8 | | | | | \$t66 | 79 | 0x00000000 | | |
| | 0x000000F0 | | | | | \$t67 | 80 | 0x00000000 | | |
| | 0x000000F4 | | | | | \$t68 | 81 | 0x00000000 | | |
| | 0x000000F8 | | | | | \$t69 | 82 | 0x00000000 | | |
| | 0x000000F0 | | | | | \$t70 | 83 | 0x00000000 | | |
| | 0x000000F4 | | | | | \$t71 | 84 | 0x00000000 | | |
| | 0x000000F8 | | | | | \$t72 | 85 | 0x00000000 | | |
| | 0x000000F0 | | | | | \$t73 | 86 | 0x00000000 | | |
| | 0x000000F4 | | | | | \$t74 | 87 | 0x00000000 | | |
| | 0x000000F8 | | | | | \$t75 | 88 | 0x00000000 | | |
| | 0x000000F0 | | | | | \$t76 | 89 | 0x00000000 | | |
| | 0x000000F4 | | | | | \$t77 | 90 | 0x00000000 | | |
| | 0x000000F8 | | | | | \$t78 | 91 | 0x00000000 | | |
| | 0x000000F0 | | | | | \$t79 | 92 | 0x00000000 | | |
| | 0x000000F4 | | | | | \$t80 | 93 | 0x00000000 | | |
| | 0x000000F8 | | | | | \$t81 | 94 | 0x00000000 | | |
| | 0x000000F0 | | | | | \$t82 | 95 | 0x00000000 | | |
| | 0x000000F4 | | | | | \$t83 | 96 | 0x00000000 | | |
| | 0x000000F8 | | | | | \$t84 | 97 | 0x00000000 | | |
| | 0x000000F0 | | | | | \$t85 | 98 | 0x00000000 | | |
| | 0x000000F4 | | | | | \$t86 | 99 | 0x00000000 | | |
| | 0x000000F8 | | | | | \$t87 | 100 | 0x00000000 | | |

Programa 13:

P13.asm

```

1 # Escreva um programa que leia um valor A da memória, identifique se o número é negativo ou não e encontre o seu módulo.
2 # O valor deverá ser reescrito sobre A.
3
4 .data
5     a: .word -10
6
7 .text
8 .globl main
9 main:
10    ori $t0, $zero, 0x1001
11    sll $t0, $t0, 16
12
13    lw $t1, 0($t0)
14
15    sra $t1, $t1, 31
16    beq $t1, $zero, is_positive
17    sub $t1, $zero, $t1
18
19    is_positive:
20        sw $t1, 0($t0)
21
22 # EOF

```

Text Segment

| Beg | Address | Code | Basic | Source |
|------------|------------|-----------------------------|---------------------------------|--------|
| 0x00400000 | 0x34081001 | ori \$t0,\$zero,0x00001001 | 10: ori \$t0,\$zero,0x1001 | |
| 0x00400004 | 0x00084400 | sll \$t0,\$t0,16 | 11: sll \$t0,\$t0,16 | |
| 0x00400008 | 0x00000000 | lw \$t1,0(\$t0) | 13: lw \$t1,0(\$t0) | |
| 0x0040000c | 0x000094fc | sra \$t1,31 | 14: sra \$t1,31 | |
| 0x00400010 | 0x11120001 | beq \$t1,\$zero,is_positive | 16: beq \$t1,\$zero,is_positive | |
| 0x00400014 | 0x00009422 | sub \$t1,\$zero,\$t1 | 17: sub \$t1,\$zero,\$t1 | |
| 0x00400018 | 0xadd90000 | sw \$t1,0(\$t0) | 20: sw \$t1,0(\$t0) | |

Data Segment

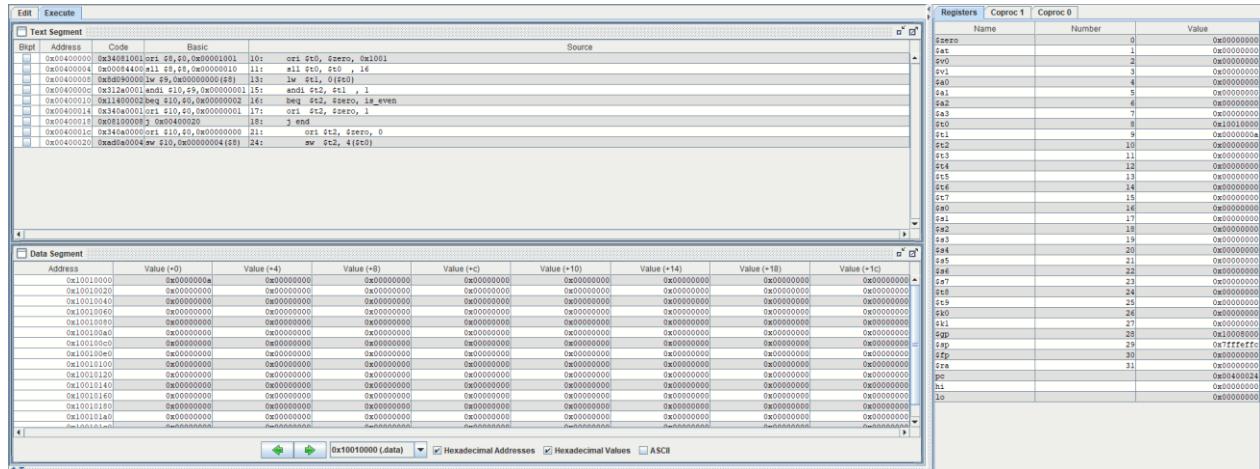
| Address | Value (+0) | Value (+4) | Value (+8) | Value (+c) | Value (+10) | Value (+14) | Value (+18) | Value (+1c) |
|------------|------------|------------|------------|------------|-------------|-------------|-------------|-------------|
| 0x10010000 | 0x00000001 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010020 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010040 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010060 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010080 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x100100a0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x100100c0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x100100e0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010100 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010120 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010140 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010160 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010180 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x100101a0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x100101c0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |

0x10010000 (data) Hexadecimal Addresses Hexadecimal Values ASCII

| Name | Number | Value |
|--------|--------|------------|
| \$zero | 0 | 0x00000000 |
| \$at | 1 | 0x00000000 |
| \$v0 | 2 | 0x00000000 |
| \$v1 | 3 | 0x00000000 |
| \$a0 | 4 | 0x00000000 |
| \$a1 | 5 | 0x00000000 |
| \$a2 | 6 | 0x00000000 |
| \$a3 | 7 | 0x00000000 |
| \$t0 | 8 | 0x10010000 |
| \$t1 | 9 | 0x00000001 |
| \$t2 | 10 | 0x00000000 |
| \$t3 | 11 | 0x00000000 |
| \$t4 | 12 | 0x00000000 |
| \$t5 | 13 | 0x00000000 |
| \$t6 | 14 | 0x00000000 |
| \$t7 | 15 | 0x00000000 |
| \$s0 | 16 | 0x00000000 |
| \$s1 | 17 | 0x00000000 |
| \$s2 | 18 | 0x00000000 |
| \$s3 | 19 | 0x00000000 |
| \$s4 | 20 | 0x00000000 |
| \$s5 | 21 | 0x00000000 |
| \$s6 | 22 | 0x00000000 |
| \$s7 | 23 | 0x00000000 |
| \$t8 | 24 | 0x00000000 |
| \$t9 | 25 | 0x00000000 |
| \$t10 | 26 | 0x00000000 |
| \$t11 | 27 | 0x00000000 |
| \$t12 | 28 | 0x00000000 |
| \$t13 | 29 | 0x1000ffff |
| \$t14 | 30 | 0x00000000 |
| \$t15 | 31 | 0x00000000 |
| \$pc | | 0x00400010 |
| \$hi | | 0x00000000 |
| \$lo | | 0x00000000 |

Programa 14:

```
P14.asm
1  # Escreva um programa que leia um valor A da memória, identifique se o número é par ou não.
2  # Um valor deverá ser escrito na segunda posição livre da memória (0 para par e 1 para ímpar).
3
4  .data
5      a: .word 10
6
7  .text
8  .globl main
9  main:
10     ori $t0, $zero, 0x1001
11     sll $t0, $t0, 16
12
13     lw $t1, 0($t0)
14
15     andi $t2, $t1, 1
16     beq $t2, $zero, is_even
17     ori $t2, $zero, 1
18     j end
19
20 is_even:
21     ori $t2, $zero, 0
22
23 end:
24     sw $t2, 4($t0)
25
26 # EOF
```



Programa 15:

```
P15.asm
6 .data
7     vetor: .space 400
8     soma: .word 0
9
10 .text
11 .globl main
12 main:
13     ori $t0, $zero, 0x1001 # $t0 = 0x00001001
14     sll $t0, $t0 , 16      # $t0 = 0x10010000 -> onde começa a memoria de dados
15
16     ori $t1, $zero, 0      # endereco = 0
17     ori $t2, $zero, 0      # i = 0
18     ori $t3, $zero, 100    # n = 100
19     ori $t4, $zero, 0      # soma = 0
20
21 loop:
22     sll $t5, $t2, 1       # $t5 = 2 * i
23     addi $t5, $t5, 1       # $t5 = 2 * i + 1
24
25     add $t6, $t0, $t1      # $t6 = $t0 + $t1 -> endereco do vetor[i]
26     sw $t5, 0($t6)        # vetor[i] = 2 * i + 1
27
28     addi $t1, $t1, 4       # endereco = endereco + 4
29     add $t4, $t4, $t5      # soma = soma + vetor[i]
30     addi $t2, $t2, 1       # i = i + 1
31
32     bne $t2, $t3, loop   # if i != n goto loop
33
34     sw $t4, 400($t0)      # MEM[400] = soma
35
```

| Text Segment | | | | | | | | | |
|--------------|------------|--------------------------|------------|---|--|--|--|--|--|
| Beg. | Address | Code | Basic | Source | | | | | |
| 0x00400000 | 0x34000000 | ori \$t0, \$zero, 0x1001 | 0x1001 | ori \$t0, \$zero, 0x1001 # \$t0 = 0x00001001 | | | | | |
| 0x00400010 | 0x34004400 | sll \$t0, \$t0 , 16 | 0x10010000 | # \$t0 = 0x10010000 -> onde começa a memoria de dados | | | | | |
| 0x00400020 | 0x34009000 | ori \$t1, \$zero, 0 | 0 | ori \$t1, \$zero, 0 # endereco = 0 | | | | | |
| 0x00400030 | 0x3400e000 | ori \$t2, \$zero, 0 | 0 | ori \$t2, \$zero, 0 # i = 0 | | | | | |
| 0x00400040 | 0x3400e004 | ori \$t3, \$zero, 100 | 100 | ori \$t3, \$zero, 100 # n = 100 | | | | | |
| 0x00400050 | 0x3400e008 | ori \$t4, \$zero, 0 | 0 | ori \$t4, \$zero, 0 # soma = 0 | | | | | |
| 0x00400060 | 0x3400e00c | add \$t6, \$t0, \$t1 | 0x10010004 | # \$t6 = \$t0 + \$t1 -> endereco do vetor[i] | | | | | |
| 0x00400070 | 0x3400e010 | sw \$t5, 0(\$t6) | 0x10010008 | # vetor[i] = 2 * i + 1 | | | | | |
| 0x00400080 | 0x3400e014 | addi \$t1, \$t1, 4 | 0x10010012 | # endereco = endereco + 4 | | | | | |
| 0x00400090 | 0x3400e018 | add \$t4, \$t4, \$t5 | 0x10010016 | # soma = soma + vetor[i] | | | | | |
| 0x004000a0 | 0x3400e01c | addi \$t2, \$t2, 1 | 0x10010020 | # i = i + 1 | | | | | |
| 0x004000b0 | 0x3400e020 | bne \$t2, \$t3, loop | 0x10010024 | # if i != n goto loop | | | | | |
| 0x004000c0 | 0x3400e024 | sw \$t4, 400(\$t0) | 0x10010028 | # MEM[400] = soma | | | | | |

| Data Segment | | | | | | | | | |
|--------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|
| Address | Value (-4) |
| 0x10010000 | 0x00000001 | 0x00000003 | 0x00000005 | 0x00000007 | 0x00000009 | 0x0000000b | 0x0000000d | 0x0000000f | 0x00000001 |
| 0x10010020 | 0x00000011 | 0x00000013 | 0x00000015 | 0x00000017 | 0x00000019 | 0x0000001b | 0x0000001d | 0x0000001f | 0x00000001 |
| 0x10010040 | 0x00000021 | 0x00000023 | 0x00000025 | 0x00000027 | 0x00000029 | 0x0000002b | 0x0000002d | 0x0000002f | 0x00000001 |
| 0x10010060 | 0x00000031 | 0x00000033 | 0x00000035 | 0x00000037 | 0x00000039 | 0x0000003b | 0x0000003d | 0x0000003f | 0x00000001 |
| 0x10010080 | 0x00000041 | 0x00000043 | 0x00000045 | 0x00000047 | 0x00000049 | 0x0000004b | 0x0000004d | 0x0000004f | 0x00000001 |
| 0x100100a0 | 0x00000051 | 0x00000053 | 0x00000055 | 0x00000057 | 0x00000059 | 0x0000005b | 0x0000005d | 0x0000005f | 0x00000001 |
| 0x100100c0 | 0x00000061 | 0x00000063 | 0x00000065 | 0x00000067 | 0x00000069 | 0x0000006b | 0x0000006d | 0x0000006f | 0x00000001 |
| 0x100100e0 | 0x00000071 | 0x00000073 | 0x00000075 | 0x00000077 | 0x00000079 | 0x0000007b | 0x0000007d | 0x0000007f | 0x00000001 |
| 0x10010100 | 0x00000081 | 0x00000083 | 0x00000085 | 0x00000087 | 0x00000089 | 0x0000008b | 0x0000008d | 0x0000008f | 0x00000001 |
| 0x10010120 | 0x00000091 | 0x00000093 | 0x00000095 | 0x00000097 | 0x00000099 | 0x0000009b | 0x0000009d | 0x0000009f | 0x00000001 |
| 0x10010140 | 0x000000a1 | 0x000000a3 | 0x000000a5 | 0x000000a7 | 0x000000a9 | 0x000000ab | 0x000000ad | 0x000000af | 0x00000001 |
| 0x10010160 | 0x000000b1 | 0x000000b3 | 0x000000b5 | 0x000000b7 | 0x000000b9 | 0x000000bb | 0x000000bd | 0x000000bf | 0x00000001 |
| 0x10010180 | 0x000000c1 | 0x000000c3 | 0x000000c5 | 0x000000c7 | 0x000000c9 | 0x000000cb | 0x000000cd | 0x000000cf | 0x00000001 |
| 0x100101a0 | 0x000000d0 | 0x000000d2 | 0x000000d4 | 0x000000d6 | 0x000000d8 | 0x000000da | 0x000000dc | 0x000000df | 0x00000001 |
| 0x100101c0 | 0x000000e0 | 0x000000e2 | 0x000000e4 | 0x000000e6 | 0x000000e8 | 0x000000ea | 0x000000ec | 0x000000ef | 0x00000001 |

Programa 16:

P16.asm

```

1  # Escreva um programa que avalie a expressão:
2  # (x * y) / z.
3  # Use
4  # x = 1600000 (=0x186A00),
5  # y = 80000 (=0x13880),
6  # z = 400000 (=0x61A80).
7  # Inicializar os registradores com os valores acima.

8

9 .data
10    x: .word 0x186A00
11    y: .word 0x13880
12    z: .word 0x61A80
13
14 .text
15 .globl main
16 main:
17     ori $s0, $0, 0x186A # $s0 = 0x0000186A
18     ori $s1, $0, 0x1388 # $s1 = 0x00001388
19     ori $s2, $0, 0x61A8 # $s2 = 0x000061A8
20     mult $s0, $s1        # $s0 * $s1 = 0x186A_ * 0x1388_ = 0xXXXX_
21     mflo $t0             #
22     div $t0, $s2         # $t0 / $s2 = 0xXXXX_ / 0x61A8_ = 0xXXXX_
23     mflo $t0             #
24     sll $t0, $t0, 8      #

25

26 # EOF

```

Text Segment

| Bxpt | Address | Code | Basic | Source |
|------------|-------------------------------------|------|---|--------|
| 0x00400000 | 0x3411114#ori \$16, \$0, 0x186A | 17: | ori \$s0, \$0, 0x186A # \$s0 = 0x0000186A | |
| 0x00400004 | 0x3411118#ori \$17, \$0, 0x00001388 | 18: | ori \$s1, \$0, 0x1388 # \$s1 = 0x00001388 | |
| 0x00400008 | 0x341261#ori \$18, \$0, 0x000061A8 | 19: | ori \$s2, \$0, 0x61A8 # \$s2 = 0x000061A8 | |
| 0x00400010 | 0x011110#mult \$s0, \$s1 | 20: | mult \$s0, \$s1 # \$s0 * \$s1 = 0x186A_ * 0x1388_ = 0xXXXX_ | |
| 0x00400014 | 0x0112001#div \$t0, \$s2 | 21: | div \$t0, \$s2 # \$t0 / \$s2 = 0xXXXX_ / 0x61A8_ = 0xXXXX_ | |
| 0x00400018 | 0x00000401#mflo \$t0 | 22: | mflo \$t0 # | |
| 0x0040001c | 0x000004200#sll \$t0, \$t0, 8 | 23: | sll \$t0, \$t0, 8 # | |
| 0x00400020 | | 24: | | |

Data Segment

| Address | Value (+0) | Value (+4) | Value (+8) | Value (+C) | Value (+10) | Value (+14) | Value (+18) | Value (+1c) |
|------------|------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| 0x10010000 | 0x00186A00 | 0x000013880 | 0x000061A80 | 0x000000000 | 0x000000000 | 0x000000000 | 0x000000000 | 0x000000000 |
| 0x10010020 | 0x00000000 | 0x00000000 | 0x00000000 | 0x000000000 | 0x000000000 | 0x000000000 | 0x000000000 | 0x000000000 |
| 0x10010040 | 0x00000000 | 0x00000000 | 0x00000000 | 0x000000000 | 0x000000000 | 0x000000000 | 0x000000000 | 0x000000000 |
| 0x10010060 | 0x00000000 | 0x00000000 | 0x00000000 | 0x000000000 | 0x000000000 | 0x000000000 | 0x000000000 | 0x000000000 |
| 0x10010080 | 0x00000000 | 0x00000000 | 0x00000000 | 0x000000000 | 0x000000000 | 0x000000000 | 0x000000000 | 0x000000000 |
| 0x100100a0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x000000000 | 0x000000000 | 0x000000000 | 0x000000000 | 0x000000000 |
| 0x100100c0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x000000000 | 0x000000000 | 0x000000000 | 0x000000000 | 0x000000000 |
| 0x100100e0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x000000000 | 0x000000000 | 0x000000000 | 0x000000000 | 0x000000000 |
| 0x10010100 | 0x00000000 | 0x00000000 | 0x00000000 | 0x000000000 | 0x000000000 | 0x000000000 | 0x000000000 | 0x000000000 |
| 0x10010120 | 0x00000000 | 0x00000000 | 0x00000000 | 0x000000000 | 0x000000000 | 0x000000000 | 0x000000000 | 0x000000000 |
| 0x10010140 | 0x00000000 | 0x00000000 | 0x00000000 | 0x000000000 | 0x000000000 | 0x000000000 | 0x000000000 | 0x000000000 |
| 0x10010160 | 0x00000000 | 0x00000000 | 0x00000000 | 0x000000000 | 0x000000000 | 0x000000000 | 0x000000000 | 0x000000000 |
| 0x10010180 | 0x00000000 | 0x00000000 | 0x00000000 | 0x000000000 | 0x000000000 | 0x000000000 | 0x000000000 | 0x000000000 |
| 0x100101a0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x000000000 | 0x000000000 | 0x000000000 | 0x000000000 | 0x000000000 |

Registers:

| 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 | 0x00004e200 |
| \$t1 | 9 | 0x00000000 |
| \$t2 | 10 | 0x00000000 |
| \$t3 | 11 | 0x00000000 |
| \$t4 | 12 | 0x00000000 |
| \$t5 | 13 | 0x00000000 |
| \$t6 | 14 | 0x00000000 |
| \$t7 | 15 | 0x00000000 |
| \$t8 | 16 | 0x0000186a |
| \$t9 | 17 | 0x00001388 |
| \$t10 | 18 | 0x000061a8 |
| \$t11 | 19 | 0x00000000 |
| \$t12 | 20 | 0x00000000 |
| \$t13 | 21 | 0x00000000 |
| \$t14 | 22 | 0x00000000 |
| \$t15 | 23 | 0x00000000 |
| \$t16 | 24 | 0x00000000 |
| \$t17 | 25 | 0x00000000 |
| \$t18 | 26 | 0x00000000 |
| \$t19 | 27 | 0x00000000 |
| \$t20 | 28 | 0x10000000 |
| \$t21 | 29 | 0xffffffff |
| \$t22 | 30 | 0x00000000 |
| \$t23 | 31 | 0x00000000 |
| pc | | 0x00400020 |
| hi | | 0x00000000 |
| lo | | 0x00004e20 |

Programa 17:

The screenshot shows the assembly code for Program 17, its registers, and memory dump.

Assembly Code (P17.asm):

```

1 # Para a expressão a seguir, escreva um programa que calcule o valor de k:
2 # k = x * y (Você deverá realizar a multiplicação através de somas!)
3 # O valor de x deve ser lido da primeira posição livre da memória e o valor de y deverá lido da segunda posição livre.
4 # O valor de k, após calculado, deverá ainda ser escrito na terceira posição livre da memória.
5
6 .data
7     x: .word 10
8     y: .word 10
9     k: .word -1
10
11 .text
12 .globl main
13 main:
14     ori $t0, $zero, 0x1001      # $t0 = 0x00001001
15     sll $t0, $t0, 16            # $t0 = 0x10010000 -> onde começa a memoria de dados
16
17     lw $t1, 0($t0)             # $t1 = x
18     lw $t2, 4($t0)              # $t2 = y
19
20     loop:
21         add $t3, $t3, $t1      # k = k + x
22         addi $t2, $t2, -1        # y = y - 1
23         bne $t2, $zero, loop    # if y != 0 goto loop
24         sw $t3, 8($t0)           # $t0[8] = st3
25 # EOF

```

Registers:

| Name | Number | Value |
|--------|--------|------------|
| \$zero | 0 | 0x00000000 |
| \$at | 1 | 0x00000000 |
| \$v0 | 2 | 0x00000000 |
| \$v1 | 3 | 0x00000000 |
| \$a0 | 4 | 0x00000000 |
| \$a1 | 5 | 0x00000000 |
| \$a2 | 6 | 0x00000000 |
| \$a3 | 7 | 0x00000000 |
| \$t0 | 8 | 0x10010000 |
| \$t1 | 9 | 0x00000000 |
| \$t2 | 10 | 0x00000000 |
| \$t3 | 11 | 0x00000044 |
| \$t4 | 12 | 0x00000000 |
| \$t5 | 13 | 0x00000000 |
| \$t6 | 14 | 0x00000000 |
| \$t7 | 15 | 0x00000000 |
| \$a0 | 16 | 0x00000000 |
| \$a1 | 17 | 0x00000000 |
| \$a2 | 18 | 0x00000000 |
| \$a3 | 19 | 0x00000000 |
| \$t0 | 20 | 0x00000000 |
| \$t1 | 21 | 0x00000000 |
| \$t2 | 22 | 0x00000000 |
| \$t3 | 23 | 0x00000000 |
| \$t4 | 24 | 0x00000000 |
| \$t5 | 25 | 0x00000000 |
| \$t6 | 26 | 0x00000000 |
| \$t7 | 27 | 0x00000000 |
| \$sp | 28 | 0x10010000 |
| \$fp | 29 | 0x00000000 |
| \$ra | 30 | 0x00000000 |
| \$ra | 31 | 0x00000000 |
| pc | | 0x00000020 |
| hi | | 0x00000000 |
| lo | | 0x00000000 |

Data Segment:

| Address | Value (+0) | Value (+4) | Value (+8) | Value (+C) | Value (+10) | Value (+14) | Value (+18) | Value (+1c) |
|------------|------------|------------|------------|------------|-------------|-------------|-------------|-------------|
| 0x10010000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010020 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010040 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010060 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010080 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x100100a0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x100100c0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x100100e0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010100 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010120 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010140 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010160 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010180 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x100101a0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |

Programa 18:

```

9 .data
10    x: .word 2
11    y: .word 5
12    k: .word -1
13
14 .text
15 .globl main
16 main:
17    ori $t0, $zero, 0x1001      # $t0 = 0x00001001
18    sll $t0, $t0 , 16          # $t0 = 0x10010000 -> onde começa a memória de dados
19
20    lw $s0, 0($t0)            # $s0 = x
21    lw $s1, 4($t0)            # $s1 = y
22
23    ori $s2, $zero, 1          # $s2 = k = 1 (inicializando k)
24    or $t2, $zero, $s1          # $t2 = y (contador)
25
26    pow:
27        beq $t2, $zero, end      # Se y == 0 goto end
28
29        ori $t3, $zero, 0          # $t3 = 0 (inicializando x)
30        or $t4, $zero, $s2          # $t4 = k (valor atual de k para multiplicação)
31
32    mul:
33        add $t3, $t3, $s0          # $t3 = $t3 + x
34        addi $t4, $t4, -1          # $t4 = $t4 - 1
35        bne $t4, $zero, mul        # Se $t4 != 0 goto mul
36
37        or $s2, $t3, $zero          # $s2 = $t3 (atualiza k)
38        addi $t2, $t2, -1          # y = y - 1
39        j pow                      # goto pow
40
41    end:
42        sw $s2, 8($t0)            # Salva o resultado em k
43
44 # EOF

```

| Reg | Name | Number | Value |
|------|------|--------|------------|
| zero | 0 | | 0x00000000 |
| tat | 1 | | 0x00000000 |
| svo | 2 | | 0x00000000 |
| sv1 | 3 | | 0x00000000 |
| svo1 | 4 | | 0x00000000 |
| sa1 | 5 | | 0x00000000 |
| sa2 | 6 | | 0x00000000 |
| ta1 | 7 | | 0x00000000 |
| ta2 | 8 | | 0x00000000 |
| ta11 | 9 | | 0x00000000 |
| ta22 | 10 | | 0x00000000 |
| t3 | 11 | | 0x00000000 |
| ta3 | 12 | | 0x00000000 |
| ta5 | 13 | | 0x00000000 |
| st5 | 14 | | 0x00000000 |
| st6 | 15 | | 0x00000000 |
| ta7 | 16 | | 0x00000000 |
| ta8 | 17 | | 0x00000005 |
| ta9 | 18 | | 0x00000000 |
| ta10 | 19 | | 0x00000000 |
| ta11 | 20 | | 0x00000000 |
| ta12 | 21 | | 0x00000000 |
| ta4 | 22 | | 0x00000000 |
| ta7 | 23 | | 0x00000000 |
| ta8 | 24 | | 0x00000000 |
| ta9 | 25 | | 0x00000000 |
| ta10 | 26 | | 0x00000000 |
| ta11 | 27 | | 0x00000000 |
| ta12 | 28 | | 0x00000000 |
| ta13 | 29 | | 0xffffffff |
| ta14 | 30 | | 0x00000000 |
| ta15 | 31 | | 0x00000000 |
| ta16 | 32 | | 0x00000000 |
| ta17 | 33 | | 0x00000000 |
| ta18 | 34 | | 0x00000000 |
| ta19 | 35 | | 0x00000000 |
| ta20 | 36 | | 0x00000000 |
| ta21 | 37 | | 0x00000000 |
| ta22 | 38 | | 0x00000000 |
| ta23 | 39 | | 0x00000000 |
| ta24 | 40 | | 0x00000000 |
| ta25 | 41 | | 0x00000000 |
| ta26 | 42 | | 0x00000000 |

Programa 19:

```

Edit Execute
P19.asm
6 .data
7     x: .word 3
8     y: .word 2
9     k: .word -1
10
11 .text
12 .globl main
13 main:
14     ori $t0, $zero, 0x1001    # $t0 = 0x00001001
15     sll $t0, $t0 , 16        # $t0 = 0x10010000 -> onde começa a memória de dados
16
17     lw $s0, 0($t0)          # Carrega x em $s0
18     lw $s1, 4($t0)          # Carrega y em $s1
19
20     ori $t1, $zero, 0        # $t1 = 0 = contador de bits significativos de x
21     ori $t2, $zero, 0        # $t2 = 0 = contador de bits significativos de y
22
23     or $t3, $zero, $s0       # $t3 = x
24     count_x:
25         beq $t3, $zero, count_y    # if $t3 == 0 goto count_y
26         srl $t3, $t3, 1           # $t3 = $t3 >> 1
27         addi $t1, $t1, 1           # $t1++
28         j count_x                # goto count_x
29
30     count_y:
31         or $t3, $zero, $s1       # $t3 = y
32     count_y_loop:
33         beq $t3, $zero, multiply    # if $t3 == 0 goto multiply
34         srl $t3, $t3, 1           # $t3 = $t3 >> 1
35         addi $t2, $t2, 1           # $t2++
36
37         j count_y_loop          # goto count_y_loop
38
39     multiply:
40         mult $t1, $t2            # $s0 * $s1
41         mflo $s2                # $s2 = LO
42         mfhi $s3                # $s3 = HI
43
44         slti $t5, $t1, 32        # Se $t1 < 32, $t5 = 1
45         slti $t6, $t2, 32        # Se $t2 < 32, $t6 = 1
46         and $t7, $t5, $t6        # $t7 = 1 se ambos têm menos de 32 bits significativos
47
48         beq $t7, $zero, store_hi_lo    # if $t7 == 0 goto store_hi_lo
49         sw $s2, 8($t0)           # Se $t7 é 1, armazena apenas em $s2
50         j end                    # goto end
51
52     store_hi_lo:
53         sw $s2, 8($t0)           # Armazena parte baixa em posição k
54         sw $s3, 12($t0)          # Armazena parte alta em posição k + 4
55
56     end:
57 # EOF

```

| Bupt | Address | Code | Basic | Source |
|------|------------|------------------------------------|-------|---|
| | 0x00400000 | Cw34000001lw \$t1,0x00000001 | 14: | ext tct, dzero, \$m1001 # \$ct1 = 0x00000001 |
| | 0x00400014 | Cw34000001lw \$t1,45,0x00000010 | 15: | ext tct, \$cto, 16 # \$ct2 = 0x10000000 -> onde começa a memória de dados |
| | 0x00400028 | Cw34000001lw \$t1,45,0x00000000 | 16: | lw \$t2, 0(\$t1) |
| | 0x00400032 | Cw34000001lw \$t1,0x00000000(\$t2) | 17: | lw \$t3, 0(\$t2) # Carrega x em \$t3 |
| | 0x00400036 | Cw34000001lw \$t1,0x00000004(\$t2) | 18: | lw \$t4, 4(\$t2) # Carrega y em \$t4 |
| | 0x00400040 | Cw34000001lw \$t1,0x00000000(\$t2) | 20: | ori \$t5, dzero, 0 # \$t5 = 0 = contador de bits significativos de x |
| | 0x00400044 | Cw34000001lw \$t1,0x00000000(\$t2) | 21: | ori \$t6, dzero, 0 # \$t6 = 0 = contador de bits significativos de y |
| | 0x00400048 | Cw34000001lw \$t1,0x00000000(\$t2) | 22: | addi \$t7, dzero, 1 # \$t7 = 1 |
| | 0x00400052 | Cw34000001lw \$t1,0x00000000(\$t2) | 23: | de \$t5, \$t3, \$t20, \$t20 # \$t5 = x |
| | 0x00400056 | Cw34000001lw \$t1,0x00000000(\$t2) | 24: | beq \$t5, dzero, count_y # if \$t5 == 0 goto count_y |
| | 0x00400060 | Cw34000001lw \$t1,0x00000001(\$t2) | 25: | addi \$t3, \$t3, 1 # \$ct3 = \$t3 >> 1 |
| | 0x00400064 | Cw34000001lw \$t1,0x00000001(\$t2) | 26: | addi \$t4, \$t4, 1 # \$ct4 = \$t4 >> 1 |
| | 0x00400068 | Cw34000001lw \$t1,0x00000001(\$t2) | 27: | addi \$t5, \$t5, 1 # \$ct5 = \$t5 >> 1 |
| | 0x00400072 | Cw34000001lw \$t1,0x00000001(\$t2) | 28: | j count_x # goso count_x |
| | 0x00400076 | Cw34000001lw \$t1,0x00000003(\$t2) | 29: | negt \$t5, dzero, \$t5 # \$t5 = -y |
| | 0x00400080 | Cw34000001lw \$t1,0x00000003(\$t2) | 30: | mult \$t6, \$t3, \$t5 # \$t6 = x * y |
| | 0x00400084 | Cw34000001lw \$t1,0x00000001(\$t2) | 31: | mult \$t7, \$t4, \$t5 # \$t7 = y * z + t5 >> 1 |
| | 0x00400088 | Cw34000001lw \$t1,0x00000001(\$t2) | 32: | addi \$t8, dzero, 1 # \$t8 = 1 |
| | 0x00400092 | Cw34000001lw \$t1,0x00000001(\$t2) | 33: | addi \$t9, \$t8, 1 # \$t9 = 2 |
| | 0x00400096 | Cw34000001lw \$t1,0x00000001(\$t2) | 34: | addi \$t10, \$t9, 1 # \$t10 = 3 |
| | 0x00400010 | Cw34000001lw \$t1,0x00000001(\$t2) | 35: | addi \$t11, \$t10, 1 # \$t11 = 4 |
| | 0x00400014 | Cw34000001lw \$t1,0x00000001(\$t2) | 36: | j count_y_loop # goto count_y_loop |
| | 0x00400018 | Cw34000001lw \$t1,0x00000001(\$t2) | 39: | mult \$t11, \$t2 # \$t2 = z |

| 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 |
| \$v2 | 8 | 0x00000000 |
| \$t1 | 9 | 0x00000002 |
| \$t2 | 10 | 0x00000002 |
| \$t3 | 11 | 0x00000000 |
| \$t4 | 12 | 0x00000000 |
| \$t5 | 13 | 0x00000001 |
| \$t6 | 14 | 0x00000001 |
| \$t7 | 15 | 0x00000000 |
| \$t8 | 16 | 0x00000000 |
| \$t9 | 17 | 0x00000002 |
| \$t2 | 18 | 0x00000004 |
| \$t3 | 19 | 0x00000000 |
| \$t4 | 20 | 0x00000000 |
| \$t5 | 21 | 0x00000000 |
| \$t6 | 22 | 0x00000000 |
| \$t7 | 23 | 0x00000000 |
| \$t8 | 24 | 0x00000000 |
| \$t9 | 25 | 0x00000000 |
| \$k0 | 26 | 0x00000000 |
| \$r1 | 27 | 0x00000000 |
| \$sp | 28 | 0x10000000 |
| \$fp | 29 | 0x7fffffe0 |
| \$fp | 30 | 0x00000000 |
| \$ra | 31 | 0x00000000 |
| hi | | 0x00000000 |
| lo | | 0x00000004 |

Programa 20:

```
P20.asm
6 .data
7     x: .word 3
8
9 .text
10 .globl main
11 main:
12     ori $t0, $zero, 0x1001      # $t0 = 0x00001001
13     sll $t0, $t0, 16           # $t0 = 0x10010000 (inicio da memória de dados)
14
15     lw $s0, 0($t0)            # Carrega x em $s0
16
17     ori $t1, $zero, 2          # $t1 = 2
18
19     div $s0, $t1               # Divide x por 2
20     mfhi $s1                  # $s1 = resto de x / 2 (0 se par, != 0 se ímpar)
21
22     mult $s0, $s0              # x^2
23     mflo $s2                  # $s2 = x^2
24
25     mult $s2, $s0              # x^3
26     mflo $s3                  # $s3 = x^3
27
28     mult $s2, $s2              # x^4
29     mflo $s4                  # $s4 = x^4
30
31     mult $s4, $s0              # x^5
32     mflo $s5                  # $s5 = x^5
33
34     bne $s1, $zero, impar_case # Se $s1 != 0 (x é ímpar), vai para impar_case
35
36
37     par_case:
38         add $t2, $s4, $s3       # $t2 = x^4 + x^3
39
40         add $t3, $s2, $s2       # $t3 = 2 * x^2
41
42         sub $t4, $t2, $t3       # $t4 = x^4 + x^3 - 2*x^2
43         sw $t4, 4($t0)          # Armazena y na segunda posição da memória
44         j end                   # Fim do programa
45
46     impar_case:
47         sub $t5, $s5, $s3       # $t5 = x^5 - x^3
48         addi $t6, $t5, 1          # $t6 = x^5 - x^3 + 1
49         sw $t6, 4($t0)          # Armazena y na segunda posição da memória
50
51     end:
52 # EOF
```

| Text Segment | | Source | | | |
|--------------|------------|-------------|--------------------------|---|--|
| Brpt | Address | Code | Basic | | |
| | 0x00400000 | 0x340001001 | ori \$t0, \$zero, 0x1001 | # \$t0 = 0x10010000 (início da memória de dados) | |
| | 0x00400004 | 0x000084400 | add \$t0, \$t0, 16 | # \$t0 = 0x10010000 | |
| | 0x00400008 | 0x000000000 | lsl \$t0, \$t0, 16 | # \$t0 = 0x10010000 | |
| | 0x0040000c | 0x340000000 | ori \$t1, \$zero, 2 | # \$t1 = 2 | |
| | 0x00400010 | 0x0209001a | div \$t0, \$t1 | # Divide x por 2 | |
| | 0x00400014 | 0x000000000 | mfhi \$t1 | # \$t1 = resto de x / 2 (0 se par, != 0 se ímpar) | |
| | 0x00400018 | 0x000000000 | mflo \$t0 | # \$t0 = x / 2 | |
| | 0x0040001c | 0x000000000 | mflo \$t0 | # \$t0 = x / 2 | |
| | 0x00400020 | 0x02090001 | mult \$t0, \$t0 | # \$t0 = x^2 | |
| | 0x00400024 | 0x000000000 | mflo \$t0 | # \$t0 = x^2 | |
| | 0x00400028 | 0x000000000 | mflo \$t0 | # \$t0 = x^2 | |
| | 0x00400030 | 0x02090001 | mult \$t0, \$t1 | # \$t0 = x^3 | |
| | 0x00400034 | 0x000000000 | mflo \$t0 | # \$t0 = x^3 | |
| | 0x00400038 | 0x000000000 | mflo \$t0 | # \$t0 = x^3 | |
| | 0x00400040 | 0x02090000 | mult \$t0, \$t0 | # \$t0 = x^6 | |
| | 0x00400044 | 0x02090000 | add \$t2, \$t1, \$t0 | # \$t2 = x^4 + x^3 | |
| | 0x00400048 | 0x02090000 | add \$t3, \$t2, \$t0 | # \$t3 = 2^3 * x^2 | |

| Name | Number | Value |
|--------|--------|------------|
| \$zero | 0 | 0x00000000 |
| \$t0 | 1 | 0x00000000 |
| \$t1 | 2 | 0x00000000 |
| \$t2 | 3 | 0x00000000 |
| \$t3 | 4 | 0x00000000 |
| \$t4 | 5 | 0x00000000 |
| \$t5 | 6 | 0x00000000 |
| \$t6 | 7 | 0x00000000 |
| \$t7 | 8 | 0x00000000 |
| \$t8 | 9 | 0x00000000 |
| \$t9 | 10 | 0x00000000 |
| \$t10 | 11 | 0x00000000 |
| \$t11 | 12 | 0x00000000 |
| \$t12 | 13 | 0x00000000 |
| \$t13 | 14 | 0x00000000 |
| \$t14 | 15 | 0x00000000 |
| \$t15 | 16 | 0x00000000 |
| \$t16 | 17 | 0x00000001 |
| \$t17 | 18 | 0x00000009 |
| \$t18 | 19 | 0x000001b1 |
| \$t19 | 20 | 0x00000000 |
| \$t20 | 21 | 0x000000f3 |
| \$t21 | 22 | 0x00000000 |
| \$t22 | 23 | 0x00000000 |
| \$t23 | 24 | 0x00000000 |
| \$t24 | 25 | 0x00000000 |
| \$t25 | 26 | 0x00000000 |
| \$t26 | 27 | 0x00000000 |
| \$t27 | 28 | 0xffffffff |
| \$t28 | 29 | 0xffffffff |
| \$t29 | 30 | 0x00000000 |
| \$t30 | 31 | 0x00000000 |
| \$pc | | 0x00000000 |
| \$hi | | 0x00000000 |
| \$lo | | 0x000000f3 |

Programa 21:

P21.asm

```

5
6 .data
7     x: .word -2
8
9 .text
10 .globl main
11 main:
12     ori $t0, $zero, 0x1001      # $t0 = 0x00001001
13     sll $t0, $t0 , 16          # $t0 = 0x10010000 -> onde começa a memória de dados
14
15     lw $s0, 0($t0)           # $s0 = x
16
17     mult $s0, $s0             # x^2
18     mflo $s2                  # $s2 = x^2
19
20     mult $s2, $s0             # x^3
21     mflo $s3                  # $s3 = x^3
22
23     mult $s3, $s0             # x^4
24     mflo $s4                  # $s4 = x^4
25
26     slt $t1, $zero, $s0       # $t1 = 1 se $s0 > 0 (x > 0)
27     beq $t1, $zero, menor    # if $s0 <= 0 goto menor
28
29     addi $t2, $s3, 1          # $t2 = x^3 + 1
30     sw $t2, 4($t0)           # $t0[1] = x^3 + 1
31     j fim
32
33     menor:
34         addi $t3, $s4, -1      # $t3 = x^4 - 1
35         sw $t3, 4($t0)           # $t0[1] = x^4 - 1
36
37     fim:
38 # EOF

```

Text Segment

| Bpt | Address | Code | Basic | Source |
|-----|------------|------------|--------------------------|------------------------------|
| | 0x00000000 | 0x40000000 | ori \$t0, \$zero, 0x1001 | 12: ori \$t0, \$zero, 0x1001 |
| | 0x00000004 | 0x40000004 | sll \$t0, \$t0 , 16 | 13: sll \$t0, \$t0 , 16 |
| | 0x00000008 | 0x40000008 | lw \$s0, 0(\$t0) | 15: lw \$s0, 0(\$t0) |
| | 0x0000000C | 0x4000000C | mult \$s0, \$s0 | 17: mult \$s0, \$s0 |
| | 0x00000010 | 0x40000010 | mflo \$s2 | 18: mflo \$s2 |
| | 0x00000014 | 0x40000014 | mult \$s2, \$s0 | 20: mult \$s2, \$s0 |
| | 0x00000018 | 0x40000018 | mflo \$s3 | 21: mflo \$s3 |
| | 0x0000001C | 0x4000001C | mult \$s3, \$s0 | 23: mult \$s3, \$s0 |
| | 0x00000020 | 0x40000020 | mflo \$s4 | 24: mflo \$s4 |
| | 0x00000024 | 0x40000024 | slt \$t1, \$zero, \$s0 | 26: slt \$t1, \$zero, \$s0 |
| | 0x00000028 | 0x40000028 | beq \$t1, \$zero, menor | 27: beq \$t1, \$zero, menor |
| | 0x00000030 | 0x40000030 | addi \$t2, \$s3, 1 | 29: addi \$t2, \$s3, 1 |
| | 0x00000034 | 0x40000034 | sw \$t2, 4(\$t0) | 30: sw \$t2, 4(\$t0) |
| | 0x00000038 | 0x40000038 | j fim | 31: j fim |
| | 0x0000003C | 0x4000003C | addi \$t3, \$s4, -1 | 34: addi \$t3, \$s4, -1 |
| | 0x00000040 | 0x40000040 | sw \$t3, 4(\$t0) | 35: sw \$t3, 4(\$t0) |

Data Segment

| Address | Value (+0) | Value (+4) | Value (+8) | Value (+12) | Value (+16) | Value (+20) | Value (+24) | Value (+28) |
|------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
| 0x10010000 | 0xfffffffffe | 0xffffffffff |
| 0x10010020 | 0x00000000 |
| 0x10010040 | 0x00000000 |
| 0x10010060 | 0x00000000 |
| 0x10010080 | 0x00000000 |
| 0x100100A0 | 0x00000000 |
| 0x100100C0 | 0x00000000 |
| 0x100100E0 | 0x00000000 |
| 0x10010100 | 0x00000000 |
| 0x10010120 | 0x00000000 |
| 0x10010140 | 0x00000000 |
| 0x10010160 | 0x00000000 |
| 0x10010180 | 0x00000000 |
| 0x100101A0 | 0x00000000 |
| 0x100101C0 | 0x00000000 |

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
```

FIM