

## Exercício Prático 05

A partir de agora, nos exercícios com programação você deverá usar o [MARS](#). Você deverá apresentar um print da tela contendo o programa e a sua execução.

Exemplo:

Seja o exercício de implementar o seguinte programa:

$x=1$ ;

$x=x+1$ ;

A solução é a seguinte:

# Associações:  $x \rightarrow \$s0$

# inicio

addi \$s0, \$zero, 1 #  $x = 1$

addi \$s0, \$s0, 1 #  $x = x + 1$

#fim

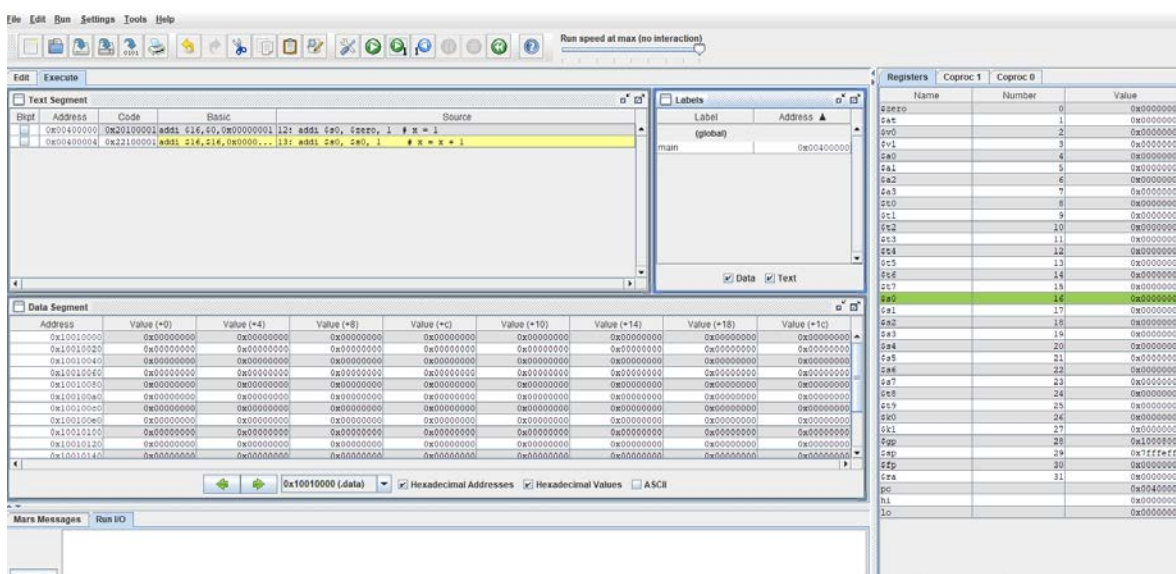
**O que deverá ser então apresentado no relatório:**

*O programa implementado:*



```
1 # Exemplo 01
2 # x=1;
3 # x=x+1;
4 # Associações:
5 # x -> $s0
6
7 # inicio
8
9 .text
10 .globl main
11 main:
12 addi $s0, $zero, 1 # x = 1
13 addi $s0, $s0, 1 # x = x + 1
14
15 #fim
16
```

*A sua execução:*



The screenshot shows the MARS execution window with the following components:

- Text Segment:** Displays the assembly code from the previous block, with the instruction `addi $s0, $s0, 1` highlighted.
- Labels:** Shows the `main` label at address `0x00400000`.
- Data Segment:** A table showing memory addresses and their corresponding values in different byte sizes.
- Registers:** A table showing the state of MIPS registers.

Register	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	0x00000000
\$t1		9	0x00000000
\$t2		10	0x00000000
\$t3		11	0x00000000
\$t4		12	0x00000000
\$t5		13	0x00000000
\$t6		14	0x00000000
\$t7		15	0x00000000
\$s0		16	0x00000001
\$s1		17	0x00000000
\$s2		18	0x00000000
\$s3		19	0x00000000
\$s4		20	0x00000000
\$s5		21	0x00000000
\$s6		22	0x00000000
\$s7		23	0x00000000
\$s8		24	0x00000000
\$s9		25	0x00000000
\$t0		26	0x00000000
\$t1		27	0x00000000
\$t2		28	0x00000000
\$t3		29	0x00000000
\$t4		30	0x00000000
\$t5		31	0x00000000
\$f0		32	0x00000000
\$f1		33	0x00000000
\$f2		34	0x00000000
\$f3		35	0x00000000
\$f4		36	0x00000000
\$f5		37	0x00000000
\$f6		38	0x00000000
\$f7		39	0x00000000
\$f8		40	0x00000000
\$f9		41	0x00000000
\$f10		42	0x00000000

**Apresente estas telas para cada exercício de programação**

## Parte1 - Resposta

1. O que é um arquivo fonte?
  - A. um arquivo de texto que contém instruções de linguagem de programação.
  - B. um subdiretório que contém os programas.
  - C. um arquivo que contém dados para um programa.
  - D. um documento que contém os requisitos para um projeto.
2. O que é um registrador?
  - A. parte do sistema de computador que mantém o controle dos parâmetros do sistema.
  - B. uma parte do processador que possui um padrão de bits.
  - C. parte do processador que contém o seu número de série único.
  - D. parte do bus de sistema que contém dados.
3. Qual o caracter que, na linguagem assembly do SPIM, inicia um comentário?
  - A. #
  - B. \$
  - C. //
  - D. \*
4. Quantos bits há em cada instrução de máquina MIPS?
  - A. 8
  - B. 16
  - C. 32
  - D. instruções diferentes possuem diferentes comprimentos.
5. O que é o contador de programa?
  - A. um registrador que mantém a conta do número de erros durante a execução de um programa.
  - B. uma parte do processador que contém o endereço da primeira palavra de dados.
  - C. uma variável na montadora que os números das linhas do arquivo de origem.
  - D. parte do processador que contém o endereço da próxima instrução de máquina para ser obtida.
6. Ao executarmos uma instrução, quanto será adicionado ao contador de programa?
  - A. 1
  - B. 2
  - C. 4
  - D. 8
7. O que é uma diretiva, tal como a diretiva .text?
  - A. uma instrução em linguagem assembly que resulta em uma instrução em linguagem de máquina.
  - B. uma das opções de menu do sistema SPIM.
  - C. uma instrução em linguagem de máquina que faz com que uma operação sobre os dados ocorra.
  - D. uma declaração que diz o montador algo sobre o que o programador quer, mas não corresponde diretamente a uma instrução de máquina.
8. O que é um endereço simbólico?
  - A. um local de memória que contém dados simbólicos.
  - B. um byte na memória que contém o endereço de dados.
  - C. símbolo dado como argumento para uma directiva.
  - D. um nome usado no código-fonte em linguagem assembly para um local na memória.

9. Em qual endereço o simulador SPIM coloca a primeira instrução de máquina quando ele está sendo executado?
- A. 0x00000000
  - B. 0x00400000
  - C. 0x10000000
  - D. 0xFFFFFFFF
10. Algumas instruções de máquina possuem uma constante como um dos operandos. Como é chamado tal operando?
- A. operando imediato
  - B. operando embutido
  - C. operando binário
  - D. operando de máquina
11. Como é chamada uma operação lógica executada entre bits de cada coluna dos operandos para produzir um bit de resultado para cada coluna?
- A. operação lógica
  - B. operação bitwise
  - C. operação binária
  - D. operação coluna
12. Quando uma operação é de fato executada, como estão os operandos na ALU?
- A. Pelo menos um operando deve ser de 32 bit.
  - B. Cada operando pode ser de qualquer tamanho.
  - C. Ambos operandos devem que vir de registros.
  - D. Cada um dos registradores deve possuir 32 bit.
13. Dezesesseis 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.

## Parte2 - Implementar em MIPS/MARS os seguintes programas (usando apenas as instruções indicadas)

//programa 1 (add, addi, sub, lógicas)

```
{  
    a = 2;  
    b = 3;  
    c = 4;  
    d = 5;  
    x = (a+b) - (c+d);  
    y = a - b + x;  
    b = x - y;  
}
```

//programa 2 (add, addi, sub, lógicas)

```
{  
    x = 1;  
    y = 5*x + 15;  
}
```

// programa 3 (add, addi, sub, lógicas)

```
{  
    x = 3;  
    y = 4 ;  
    z = ( 15*x + 67*y)*4  
}
```

**Nos exercícios a seguir procure usar as inst. sll, srl e sra:**

**// programa 4**

```
{
    x = 3;
    y = 4;
    z = ( 15*x + 67*y)*4
}
```

**// programa 5**

```
{
    x = 100000;
    y = 200000;
    z = x + y;
}
```

**// programa 6**

```
{
    x = o maior inteiro possível;
    y = 300000;
    z = x - 4y
}
```

**// programa 7**

Considere a seguinte instrução iniciando um programa:

ori \$8, \$0, 0x01

Usando apenas instruções reg-reg lógicas e/ou instruções de deslocamento (sll, srl e sra), continuar o programa de forma que ao final, tenhamos o seguinte conteúdo no registrador \$8:

\$8 = 0xFFFFFFFF

**// programa 8**

Inicialmente escreva um programa que faça:

\$8 = 0x12345678.

A partir do registrador \$8 acima, usando apenas instruções lógicas (or, ori, and, andi, xor, xori) e instruções de deslocamento (sll, srl e sra), você deverá obter os seguintes valores nos respectivos registradores:

\$9 = 0x12

\$10 = 0x34

\$11 = 0x56

\$12 = 0x78

## Para os programas a seguir use instruções de Memória (lw e sw)

### // programa 9

Considere a memória inicial da seguinte forma:

.text

.data

x1: .word 15

x2: .word 25

x3: .word 13

x4: .word 17

soma: .word -1

Escrever um programa que leia todos os números, calcule e substitua o valor da variável soma por este valor.

### // programa 10

Considere o seguinte programa:  $y = 127x - 65z + 1$

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 memória e, na posição imediatamente a seguir, o valor de y deverá ser escrito, ou seja:

.data

x: .word 5

z: .word 7

y: .word 0 # esse valor deverá ser sobrescrito após a execução do programa.

### // programa 11

Considere o seguinte programa:  $y = x - z + 300000$

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 memória e, na posição imediatamente a seguir, o valor de y deverá ser escrito, ou seja:

.data

x: .word 100000

z: .word 200000

y: .word 0 # esse valor deverá ser sobrescrito após a execução do programa.

### // programa 12

Considere a seguinte situação:

```
int ***x;
```

onde x contem um ponteiro para um ponteiro para um ponteiro para um inteiro.

Nessa situação, considere que a posição inicial de memória contenha o inteiro em questão.

Coloque todos os outros valores em registradores, use os endereços de memória que quiser dentro do espaço de endereçamento do Mips.

Resumo do problema:

$k = \text{MEM} [ \text{MEM} [ \text{MEM} [ x ] ] ]$ .

Crie um programa que implemente a estrutura de dados acima, leia o valor de K, o multiplique por 2 e o reescreva no local correto conhecendo-se apenas o valor de x.

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

**// programa 13:**

Escreva um programa que leia um valor A da memória, identifique se o número é negativo ou não e encontre o seu módulo. O valor deverá ser reescrito sobre A.

**// programa 14:**

Escreva um programa que leia um valor A da memória, identifique se o número é par ou não. Um valor deverá ser escrito na segunda posição livre da memória (0 para par e 1 para ímpar).

**// programa 15:**

Escrever um programa que crie um vetor de 100 elementos na memória onde  $\text{vetor}[i] = 2*i + 1$ . Após a última posição do vetor criado, escrever a soma de todos os valores armazenados do vetor.

Use o MARS para verificar a quantidade de instruções conforme o tipo (ULA, Desvios, Mem ou Outras)

**// programa 16**

Escreva um programa que avalie a expressão:  $(x*y)/z$ .

Use  $x = 1600000$  ( $=0x186A00$ ),  $y = 80000$  ( $=0x13880$ ), e  $z = 400000$  ( $=0x61A80$ ). Inicializar os registradores com os valores acima.

**// programa 17**

Para a expressão a seguir, escreva um programa que calcule o valor de k:

$k = x * y$  (Você deverá realizar a multiplicação através de somas!)

O valor de x deve ser lido da primeira posição livre da memória e o valor de y deverá lido da segunda posição livre. O valor de k, após calculado, deverá ainda ser escrito na terceira posição livre da memória.

**// programa 18**

Para a expressão a seguir, escreva um programa que calcule o valor de k:

$k = x^y$

Obs: Você poderá utilizar o exercício anterior.

O valor de x deve ser lido da primeira posição livre da memória e o valor de y deverá lido da segunda posição livre. O valor de k, após calculado, deverá ainda ser escrito na terceira posição livre da memória.

Dê um valor para x e y (dê valores pequenos !!) e use o MARS para verificar a quantidade de instruções conforme o tipo (ULA, Desvios, Mem ou Outras)

**Desafio:**

**Todos viram durante a parte aritmética que podemos utilizar a ULA e registradores para multiplicar dois números através de 3 algoritmos.**

**Você deverá escrever um programa que leia dois números da memória (primeira posição e segunda posição) os multiplique e coloque o resultado na terceira posição a memória.**

**Procure usar a versão 3 do algoritmo de multiplicação, pode ser mais simples !!**

**Atenção** que, ao multiplicarmos dois números de 32 bits a resposta poderá ser um número de 64 bits, assim a resposta deverá estar contida em dois registradores temporários, um armazenará a parte superior do número e outro a parte inferior, portanto duas posições de memória serão escritas (a terceira e a quarta).

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

## **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

## Implemente os programas a seguir no MIPS/MARS

### // programa 19

Escrever um programa que leia dois números da memória, a primeira e segunda posições respectivamente (os coloque em \$s0 e \$s1) e determine a quantidade de bits significantes de cada um. Coloque as respostas em \$t0 e \$t1, a partir desse resultado faça a multiplicação. Caso o número de bits significantes de ambos seja menor do que 32 a resposta deverá estar apenas em \$s2, caso contrário a resposta estará em \$s2 e \$s3 (LO e HI respectivamente).

**Para os exercícios a seguir, considere as variáveis com números abaixo de 16 bits, salvo se mencionado ao contrário.**

### // programa 20

$$y = \begin{cases} x^4 + x^3 - 2x^2 & \text{se } x \text{ for par} \\ x^5 - x^3 + 1 & \text{se } x \text{ for impar} \end{cases}$$

Os valores de  $x$  devem ser lidos da primeira posição livre da memória e o valor de  $y$  deverá ser escrito na segunda posição livre.

**// programa 21**

$$y = \begin{cases} x^3 + 1 & \text{se } x > 0 \\ x^4 - 1 & \text{se } x \leq 0 \end{cases}$$

Os valores de  $x$  devem ser lidos da primeira posição livre da memória e o valor de  $y$  deverá ser escrito na segunda posição livre.