

# Lista de exercícios 2 – Parte 1

## FAQ:

- 1) Esta lista de exercícios **é individual**, cada um deve submeter o seu trabalho.
- 2) Os exercícios devem ser resolvidos de **forma manuscrita** em uma folha (**Não é** para apresentar telas do MARS).
- 3) Um documento final deverá ser escaneado e submetido na plataforma **CANVAS** em apenas um arquivo no **formato PDF**.
- 4) A data de entrega será o dia da segunda avaliação até 23:59.

Para todos os exercícios a seguir, use a linguagem de montagem do processador MIPS visto em sala.

1)

```
a = 10;  
b = -1;  
a = a + 1;  
c = a + b;
```

2)

```
x = 3;  
y = x * 4 ;
```

3)

```
x = 3;  
y = x * 1025 ;
```

4)

```
x = 3;  
y = x / 4 ;
```

5)

```
x = 305419896;
```

6)

```
x = -1;  
y = x / 32 ;
```

7)

```
A [ 12 ] = h + A [ 8 ];
```

8)

```
h = k + A [ i ] ;
```

9)

```
A [ j ] = h + A [ i ] ;
```

10)

```
h = A [ i ] ;  
A [ i ] = A [ i + 1 ] ;  
A [ i + 1 ] = h ;
```

**11)**

```
j = 0;
i = 10;
do
{
    j = j + 1;
}
while (j != i);
```

**12)**

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.

**13)**

Escreva um programa que leia da memória um valor de Temperatura TEMP. Se  $TEMP \geq 30$  e  $TEMP \leq 50$  uma variável FLAG, também na memória, deverá receber o valor 1, caso contrário, FLAG deverá ser zero.

**14)**

Considere que a partir da primeira posição livre da memória temos um vetor com 100 elementos. Escrever um programa que ordene esse vetor de acordo com o algoritmo da bolha. Faça o teste colocando um vetor totalmente desordenado e verifique se o algoritmo funciona.

**15)**

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

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.

**16)**

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

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.

**17)**

Escreva um programa que compute a série de Fibonacci, a série é definida como:

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...

Cada termo da soma é a soma dos dois termos predecessores.

Exemplo: o termo 13 é a soma de 5 e 8.

Escreva o programa que compute os primeiros 100 termos da série. Se não for possível computar estes 100 termos, identifique a maior quantidade possível suportada pelo emulador.

Cada termo deverá estar em uma posição da memória. O primeiro termo na primeira posição livre da memória.

**18)**

Escreva um programa que leia um número armazenado na primeira posição livre da memória. Após a leitura desse número, um registrador qualquer será um flag, isto é, se esse número lido estiver na faixa de 50 a 100 ou 150 a 200 esse registrador deverá conter um “1”, caso contrário esse registrador deverá conter “0”.

Exemplo:

```
leia número;
se ( 50 <= número <=100 ou 150 <= número <= 200)
    registrador flag = 1;
caso contrário
    registrador flag = 0;
```

**19)**

Escreva um programa que calcule a mediana de 3 números armazenados na memória. Após encontrar essa mediana, escrevê-la na posição seguinte aos 3 números.

Exemplo:

.data

A: .word 23

B: .word 98

C: .word 17

Os três números acima serão armazenados na memória quando o programa for iniciado.

A mediana é maior ou igual a um dos inteiros e menor ou igual ao outro.

No caso acima, a mediana é o número 23

Um outro caso possível:

.data

A: .word 9

B: .word 98

C: .word 9

Nesse caso a mediana é o "9".

Considere que os números nas posições A, B e C podem ser trocados de rodada para rodada do seu programa.

**20)**

Para os dois exercícios a seguir, considere que a máquina opera a 100MHz e os CPIs das instruções são:

Instruções da ALU -> 3;

Instruções de Desvio -> 4;

Instruções de MEM -> 5;

- Considere que um vetor de 100 números inteiros está armazenado na memória e o endereço base está em \$S1. Escrever um programa que some todos os elementos do vetor e armazene esta soma na primeira posição de memória após o vetor. Calcule o CPI médio, o tempo de execução do programa, implemente alguma melhoria nesse seu programa e calcule o speedup (se o seu programa já está na menor versão possível, insira dois nops dentro do loop e calcule o speedup do programa original sobre esse com os dois nops).

21)

- Repita os cálculos anteriores para o seguinte programa em MIPS:

```
addi $S3, $S2, 396
```

LOOP:

```
lw $S1, 0($S2)
```

```
addi $S1, $S1, 1
```

```
sw $S1, 0($S2)
```

```
addi $S2, $S2, 4
```

```
sub $S4, $S3, $S2
```

```
bne $S4, $zero, LOOP
```

22)

Escreva uma função que receba como argumentos 2 números inteiros de 32 bits. Essa função deverá também retornar um inteiro.

O primeiro número recebido como parâmetro representa um endereço de memória e o segundo uma quantidade de elementos. Sua função deverá criar um vetor que tem início nesse endereço de memória (primeiro argumento) e a quantidade de elementos desse vetor dadas pelo segundo argumento.

Cada elemento do vetor é um elemento da série:

$y[i] = 2i - 1$  se  $i$  for par;

$y[i] = i$  se  $i$  for ímpar.

O valor retornado será a soma de todos os elementos de  $y[]$ .

23)

Escreva um programa que solicite ao usuário que digite dois números, seu programa deverá conter uma função que receba esses dois números e retorne o primeiro elevado ao segundo. Esse resultado deverá ser mostrado na tela. O programa rodará indefinidamente até que o primeiro número digitado seja 0 (zero).

**Obs.: Caso você não tenha visto a utilização de handlers e a leitura de valores pelo teclado, os dois números deverão ser lidos da primeira e segunda posição livre da memória. O resultado será escrito na terceira posição livre da memória e o programa irá executar apenas uma vez.**

**Comece já, você nunca terá tanto tempo para terminar !!!!!**