

ORGCAMP L11 – Laboratório de Organização de Computadores




Prof. Jean Marcos Laine
Prof. Bruno da Silva Rodrigues
Prof. Wallace Rodrigues de Santana

Faculdade de Computação e Informática – FCI
Univ. Presbiteriana Mackenzie

*Siga as instruções abaixo e resolva os exercícios solicitados. Depois, elabore um relatório para a atividade usando o template **Relatório de Práticas de Lab Org.docx**, disponível no Moodle, seguindo e respeitando o modelo.*

Lab #3 – Uso de Vetores e Acesso à Memória

ORIENTAÇÕES INICIAIS

1. Abra o simulador MARS para editar, montar (gerar o código binário) e testar o(s) programa(s) que será(ão) desenvolvido(s).
2. Lembrando, a montagem do programa pode ser feita em Run -> Assemble ou clicando no botão de atalho .
3. Em um primeiro momento, execute seu código no modo passo-a-passo, clicando em  e observando o conteúdo dos registradores, das memórias, as instruções que estão sendo executadas etc. Analise se sua lógica está adequada ao exercício proposto.
4. Depois, faça a execução normal para realizar seus casos de teste e validar seu programa, clicando em .

Exercício 1 – Considere o seguinte vetor de inteiros:

[2, -5, 12, 7, -3, 99, 8, 54, 21, -45, 67, 61, 55]

Declare explicitamente este vetor no seu programa, conforme exemplificado abaixo:

```
.data
vetor: .word 2, -5, 12, 7, -3, 99, 8, 54, 21, -45, 67, 61, 55
tamanho: .word 13 # Tamanho do vetor
```

Exemplo de declaração do vetor

Em seguida, crie uma função para cada uma das ações listadas abaixo:

a) Imprimir o vetor original:

- Formate a impressão conforme o exemplo acima.

b) Encontrar e imprimir o maior elemento dentro do vetor.

c) Encontrar e imprimir o menor elemento dentro do vetor.

d) Calcular e imprimir a média dos elementos do vetor.

e) Verificar se um elemento está presente no vetor

- Criar uma função que solicite ao usuário um inteiro de entrada, depois verifica se o número está presente no vetor e imprima uma mensagem indicando o resultado: Número encontrado no vetor! Ou Número não encontrado no vetor!

Lembre-se de implementar as funções correspondentes e invocá-las no "Main" do programa.

A seguir, temos exemplos de saídas esperadas para cada uma das funções do programa:

Saídas

"O vetor original é: [2, -5, 12, 7, -3, 99, 8, 54, 21, -45, 67, 61, 55]"

"O maior elemento do vetor é: 99"

"O menor elemento do vetor é: -45"

"A média dos elementos do vetor é: 25.615"

"Digite um número: 67"

Número encontrado no vetor" "

Resposta:

Código:

```
1  .data
2      vetor: .word 2, -5, 12, 7, -3, 99, 8, 54, 21, -45, 67, 61, 55
3      tamanho: .word 13
4      colchete: .asciiz "]"
5      virgula: .asciiz ", "
6      saidaA: .asciiz "O vetor original é: ["
7      saidaB: .asciiz "O maior elemento do vetor é: "
8      saidaC: .asciiz "\nO menor elemento do vetor é: "
9      saidaD: .asciiz "\nA média dos elementos do vetor é: "
10     saidaE: .asciiz "\nDigite um número: "
11     encontrado: .asciiz "Número encontrado no vetor"
12     naoEncontrado: .asciiz "Número não encontrado no vetor"
13
14  .text
15      j main
16
17     imprime_original: # Imprimir o vetor original
18
19     move $t9, $ra
20     jal carrega # carrega o vetor ($s0) e tamanho ($t0)
21
22     li $v0, 4
23     la $a0, saidaA
24     syscall
25
26     loop:
27
28     lw $s1, 0($s0)
29
30     li $v0, 1
31     move $a0, $s1
32     syscall
33
34     add $s0, $s0, 4
35     add $t1, $t1, 1
36
37     beq $t1, $t0, fim_loop
38
39     li $v0, 4
40     la $a0, virgula
41     syscall
42
43     j loop
44
45     fim_loop:
46
47     li $v0, 4
48     la $a0, colchete
49     syscall
50
51     jr $t9
```

```

53     imprime_maior: # Encontrar e imprimir o maior elemento dentro do vetor
54
55     move $t9, $ra
56     jal carrega # carrega o vetor ($s0) e tamanho ($t0)
57
58     lw $t3, 0($s0)
59
60     # while( $t1 < tamanho )
61     while_01:
62     addi $s0, $s0, 4
63     lw $t2, 0($s0)
64     bge $t3, $t2, final_while01
65
66     move $t3, $t2
67
68     final_while01:
69     addi $t1, $t1, 1
70     blt $t1, $t0, while_01
71
72     li $v0, 4
73     la $a0, saidaB
74     syscall
75
76     li $v0, 1
77     move $a0, $t3
78     syscall

```

```

80     jr $t9
81
82
83     imprime_menor: # Encontrar e imprimir o menor elemento dentro do vetor
84
85     move $t9, $ra
86     jal carrega # carrega o vetor ($s0) e tamanho ($t0)
87
88     lw $t3, 0($s0)
89
90     # while( $t1 < tamanho )
91     while_02:
92     addi $s0, $s0, 4
93     lw $t2, 0($s0)
94     ble $t3, $t2, final_while02
95
96     move $t3, $t2
97
98     final_while02:
99     addi $t1, $t1, 1
100    blt $t1, $t0, while_02
101
102    li $v0, 4
103    la $a0, saidaC
104    syscall
105

```

```

106     li $v0, 1
107     move $a0, $t3
108     syscall
109
110     jr $t9
111
112     imprime_media: # Calcular e imprimir a média dos elementos do vetor
113
114     move $t9, $ra
115     jal carrega # carrega o vetor ($s0) e tamanho ($t0)
116
117     lw $t3, 0($s0)
118
119     # while( $t1 < tamanho )
120     while_03:
121     addi $s0, $s0, 4
122     lw $t2, 0($s0)
123     add $t3, $t3, $t2
124
125     #final do while
126     addi $t1, $t1, 1
127     blt $t1, $t0, while_03
128
129     li $v0, 4
130     la $a0, saidaD
131     syscall

```

```

133     li $v0, 1
134     div $a0, $t3, $t0
135     syscall
136
137     jr $t9
138
139     verifica_elemento: # Verificar se um elemento está presente no vetor
140
141     move $t9, $ra
142     jal carrega # carrega o vetor ($s0) e tamanho ($t0)
143     addi $t0, $t0, 1
144
145     lw $t3, 0($s0)
146
147     li $v0, 4
148     la $a0, saidaE
149     syscall
150
151     li $v0, 5
152     syscall
153
154     # while( $t1 < tamanho )
155     while_04:
156     beq $v0, $t3, Achou
157     addi $s0, $s0, 4
158     lw $t3, 0($s0)

```

```

160      #final do while
161      addi $t1, $t1, 1
162      blt $t1, $t0, while_04
163
164      li $v0, 4
165      la $a0, naoEncontrado
166      syscall
167
168      jr $t9
169
170      Achou:
171
172      li $v0, 4
173      la $a0, encontrado
174      syscall
175
176      jr $t9
177
178
179      carrega: # carrega as variaveis para evitar que estejam manipuladas ao começo de cada função
180
181      la $s0, vetor
182      lw $t0, tamanho
183      li $t1, 0
184
185      jr $ra

```



```

187      main:
188
189      jal imprime_original
190      jal imprime_maior
191      jal imprime_menor
192      jal imprime_media
193      jal verifica_elemento
194
195      li $v0, 10
196      syscall
197
198

```

Testes:

```
O vetor original é: [2, -5, 12, 7, -3, 99, 8, 54, 21, -45, 67, 61, 55]
O maior elemento do vetor é: 99
O menor elemento do vetor é: -45
A média dos elementos do vetor é: 25
Digite um número: 99
Número encontrado no vetor
-- program is finished running --

O vetor original é: [2, -5, 12, 7, -3, 99, 8, 54, 21, -45, 67, 61, 55]
O maior elemento do vetor é: 99
O menor elemento do vetor é: -45
A média dos elementos do vetor é: 25
Digite um número: 2
Número encontrado no vetor
-- program is finished running --

O vetor original é: [2, -5, 12, 7, -3, 99, 8, 54, 21, -45, 67, 61, 55]
O maior elemento do vetor é: 99
O menor elemento do vetor é: -45
A média dos elementos do vetor é: 25
Digite um número: 55
Número encontrado no vetor
-- program is finished running --

O vetor original é: [2, -5, 12, 7, -3, 99, 8, 54, 21, -45, 67, 61, 55]
O maior elemento do vetor é: 99
O menor elemento do vetor é: -45
A média dos elementos do vetor é: 25
Digite um número: 68
Número não encontrado no vetor
-- program is finished running --

O vetor original é: [2, -5, 12, 7, -3, 99, 8, 54, 21, -45, 67, 61, 55]
O maior elemento do vetor é: 99
O menor elemento do vetor é: -45
A média dos elementos do vetor é: 25
Digite um número: 13
Número não encontrado no vetor
-- program is finished running --
```

Comentários:

Nesse exercício eu precisei percorrer a array diversas vezes, e para isso decidi deixar na primeira função os um loop para que ficasse mais evidente o que está acontecendo no programa, nas outras funções apliquei versões de loop semelhantes.

É possível ver que dentro de todas as funções eu chamei outra função, a “Carrega”, foi a primeira vez que implementei uma função dentro de outra, e notei que não era possível mais usar o valor do registrador \$ra pois era sobrescrito pelo endereço da nova função. para contornar isso todas as vezes antes de chamar a carrega eu passava o valor de \$ra para \$t9, dessa forma o jr chamava o valor dentro de \$t9 para voltar a main.

A formatação foi uma das partes mais fáceis, apenas chamando as syscalls alternadamente para formar o resultado final.

Para encontrar um número no vetor, eu percorri ele e caso encontrasse antes do loop acabar, ele pulava para um trecho do código que imprime a mensagem de encontrado, caso não encontrasse, saíria do loop e já iria de encontro a um trecho de código que imprime a mensagem de não encontrado.

Exercício 2 – Exercício de Estatísticas de Notas em Assembly MIPS

Você deve desenvolver um programa em assembly MIPS que realize as seguintes tarefas:

1. Solicitação de Notas:

- O programa deve pedir ao usuário que insira as notas de 10 alunos.

2. Armazenamento em Vetor:

- As notas fornecidas pelo usuário devem ser armazenadas em memória, seguindo o formato de um vetor. O espaço necessário para este vetor deve ser alocado inicialmente no segmento de dados do programa.

3. Cálculo de Aprovação e Reprovação:

- O programa deve calcular quantos alunos foram aprovados (nota maior ou igual a 6) e quantos foram reprovados (nota menor que 6).

4. Contagem de Alunos com Nota igual a Zero

- O programa deve contar quantos alunos tem nota igual a zero.

5. Impressão dos Resultados:

Após realizar os cálculos, o programa deve imprimir na tela:

- O vetor de notas
- A quantidade de alunos aprovados.
- A quantidade de alunos reprovados.
- A quantidade de alunos com nota igual a zero.

Este exercício visa a prática do uso das instruções LW (Load Word) e SW (Store Word) para manipulação de vetores em assembly MIPS, bem como o cálculo de estatísticas básicas. Certifique-se de testar o programa em um ambiente adequado para as instruções MIPS.

```

1  .data
2  Notas_Alunos:      .space 40      #Memoria alocada para notas (vetor de 10 elementos x 4 bytes de cada nota)
3  Contador_Aprovados: .word 0 #Contador de alunos aprovados
4  Contador_Reprovados: .word 0 #Contador de alunos reprovados
5  Contador_Zeros: .word 0 #Contador de alunos zerados
6  #Mensagens de sistema:
7  Msg_Solicitar: .ascii "Digite a nota do aluno: "
8  Msg_Notas: .ascii "\nNotas dos alunos: "
9  Msg_Aprovados: .ascii "\nA quantidade de alunos aprovados: "
10 Msg_Reprovados: .ascii "\nA quantidade de alunos reprovados: "
11 Msg_Zerados: .ascii "\nA quantidade de alunos com nota igual a zero: "
12 Simbolo_Chave1: .ascii "Vetor de notas: ["
13 Simbolo_Chave2: .ascii "]"
14 Virgula: .ascii ", "
15
16 .text
17 Main:
18     #Inicializa o programa
19     li $t0, 0      #Registrador para guardar o índice vetor de notas que inicia em 0
20     li $t1, 10     #Quantidade de alunos
21     la $t2, Notas_Alunos #Endereço do vetor de notas
22     li $t3, 6      #Variavel de quem foi aprovado (maior que 6) para comparação
23
24 Solicitar_Notas_Dos_Alunos:
25     #Solicita a nota do usuário
26     li $v0, 4
27     la $a0, Msg_Solicitar #Carrega a mensagem de inserir nota
28     syscall
29
30     li $v0, 5      #ler um inteiro (nota)
31     syscall
32
33     #Guarda a nota no vetor de notas
34     sw $v0, 0($t2)
35
36     #Verifica se a nota é zero se for vai para o bloco de zeros
37     beqz $v0, Notas_Zeradas
38

```

```

39     #Verifica se a nota é maior ou igual a 6 (está aprovado)
40     bge $v0, $t3, Notas_Aprovadas
41
42     #Se não, é reprovado
43     j Notas_Reprovadas
44
45 #Label de alunos aprovados
46 Notas_Aprovadas:
47     lw $t4, Contador_Aprovados #Carrega o contador e alunos aprovados
48     addi $t4, $t4, 1          #Incrementa a contagem se o aluno foi aprovado
49     sw $t4, Contador_Aprovados #Guarda a nova contagem
50     j Proximo_Aluno
51
52 #Label de alunos zerados
53 Notas_Zeradas:
54     lw $t4, Contador_Zeros     #Carrega o contador de alunos com nota zero
55     addi $t4, $t4, 1          #Incrementa a contagem se ele tirou 0
56     sw $t4, Contador_Zeros     #Guarda a nova contagem
57
58     #Se ele tirou 0 está reprovado também:
59     lw $t4, Contador_Reprovados #Carrega o contador de alunos reprovados
60     addi $t4, $t4, 1          #Incrementa a contagem se ele foi reprovado
61     sw $t4, Contador_Reprovados #Guarda a nova contagem
62     j Proximo_Aluno
63
64 #Label de alunos reprovados
65 Notas_Reprovadas:
66     lw $t4, Contador_Reprovados #Carrega o contador de alunos reprovados
67     addi $t4, $t4, 1          #Incrementa a contagem se ele foi reprovado
68     sw $t4, Contador_Reprovados #Guarda a nova contagem
69
70 #Label para passar para o proximo aluno do vetor
71 Proximo_Aluno:
72     addi $t0, $t0, 1          #Incrementa o índice no vetor de notas
73     addi $t2, $t2, 4          #Vai para a próxima posição do vetor de notas adicionando 4 bytes de memoria
74     addi $t1, $t1, -1         #Decrementa o contador de alunos
75
76     bnez $t1, Solicitar_Notas_Dos_Alunos #Loop para solicitar a nota dos alunos

```

```

77
78     #Imprime o vetor de notas
79     li $v0, 4
80     la $a0, Simbolo_Chave1 #Carrega e imprime a mensagem "vetor de notas: ["
81     syscall
82
83     la $t2, Notas_Alunos    #Carrega o endereço de memória do vetor de notas em $t2
84     li $t1, 10              #Quantidade total de alunos
85
86 #Imprime as notas do vetor
87 Imprimir_Notas:
88     lw $a0, 0($t2)          #Carrega a nota do vetor em $a0
89     li $v0, 1              #Imprime um inteiro
90     syscall
91
92     addi $t2, $t2, 4        #Avança para a próxima nota/elemento no vetor
93     addi $t1, $t1, -1      #Decrementa no contador de alunos
94
95     bnez $t1, Imprimir_Virgula #Verifica se é a última nota se não continua
96
97     #Imprime o vetor de notas
98     li $v0, 4              #Código para imprimir string
99     la $a0, Simbolo_Chave2 #Carrega a mensagem em $a0
100    syscall
101
102    j Imprimir_Aprovados    #Volta para imprimir os aprovados
103
104 #Imprime a virgula para separar os elementos do vetor
105 Imprimir_Virgula:
106     li $v0, 4
107     la $a0, Virgula        #Carrega e imprime a virgula
108     syscall
109
110    j Imprimir_Notas
111
112 #Imprime a quantidade de alunos aprovados
113 Imprimir_Aprovados:
114     li $v0, 4
115
116
117
118     la $a0, Msg_Aprovados #Imprime uma mensagem de alunos aprovados
119     syscall
120
121
122     lw $a0, Contador_Aprovados #Carrega a quantidade de alunos aprovados em $a0
123     li $v0, 1              #Imprime o inteiro
124     syscall
125
126
127     #Imprime a quantidade de alunos reprovados
128     li $v0, 4
129     la $a0, Msg_Reprovados #Carrega a mensagem de alunos reprovados
130     syscall
131
132     lw $a0, Contador_Reprovados #Carrega a quantidade de alunos reprovados em $a0
133     li $v0, 1              #Imprime o inteiro
134     syscall
135
136
137     #Imprime a quantidade de alunos com nota zero
138     li $v0, 4
139     la $a0, Msg_Zerados    #Carrega a mensagem de alunos com 0
140     syscall
141
142     lw $a0, Contador_Zeros #Carrega a quantidade de alunos com nota zero em $a0
143     li $v0, 1              #Imprime o inteiro
144     syscall
145
146     #Finaliza o programa
147     li $v0, 10
148     syscall
149
150

```

```
Digite a nota do aluno: 0
Digite a nota do aluno: 1
Digite a nota do aluno: 2
Digite a nota do aluno: 3
Digite a nota do aluno: 4
Digite a nota do aluno: 5
Digite a nota do aluno: 6
Digite a nota do aluno: 7
Digite a nota do aluno: 8
Digite a nota do aluno: 9
Vetor de notas: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
A quantidade de alunos aprovados: 4
A quantidade de alunos reprovados: 6
A quantidade de alunos com nota igual a zero: 1
-- program is finished running --
```

```
Digite a nota do aluno: 6
Digite a nota do aluno: 7
Digite a nota do aluno: 8
Digite a nota do aluno: 9
Digite a nota do aluno: 6
Digite a nota do aluno: 7
Digite a nota do aluno: 8
Digite a nota do aluno: 9
Digite a nota do aluno: 6
Digite a nota do aluno: 7
Vetor de notas: [6, 7, 8, 9, 6, 7, 8, 9, 6, 7]
A quantidade de alunos aprovados: 10
A quantidade de alunos reprovados: 0
A quantidade de alunos com nota igual a zero: 0
-- program is finished running --
```

```
Digite a nota do aluno: 5
Digite a nota do aluno: 4
Digite a nota do aluno: 3
Digite a nota do aluno: 2
Digite a nota do aluno: 1
Digite a nota do aluno: 5
Digite a nota do aluno: 4
Digite a nota do aluno: 3
Digite a nota do aluno: 2
Digite a nota do aluno: 1
Vetor de notas: [5, 4, 3, 2, 1, 5, 4, 3, 2, 1]
A quantidade de alunos aprovados: 0
A quantidade de alunos reprovados: 10
A quantidade de alunos com nota igual a zero: 0
-- program is finished running --
```

```
Digite a nota do aluno: 0
Digite a nota do aluno: 0
Digite a nota do aluno: 0
Digite a nota do aluno: 0
Digite a nota do aluno: 0
Digite a nota do aluno: 0
Digite a nota do aluno: 0
Digite a nota do aluno: 0
Digite a nota do aluno: 0
Digite a nota do aluno: 0
Vetor de notas: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
A quantidade de alunos aprovados: 0
A quantidade de alunos reprovados: 10
A quantidade de alunos com nota igual a zero: 10
-- program is finished running --
```

```
Digite a nota do aluno: 7
Digite a nota do aluno: 8
Digite a nota do aluno: 10
Digite a nota do aluno: 8
Digite a nota do aluno: 5
Digite a nota do aluno: 2
Digite a nota do aluno: 0
Digite a nota do aluno: 5
Digite a nota do aluno: 0
Digite a nota do aluno: 7
Vetor de notas: [7, 8, 10, 8, 5, 2, 0, 5, 0, 7]
A quantidade de alunos aprovados: 5
A quantidade de alunos reprovados: 5
A quantidade de alunos com nota igual a zero: 2
-- program is finished running --
```



Funcionalidade Extra (**Desafio**): experimente ordenar o vetor de notas em ordem crescente, usando um algoritmo de ordenação do tipo *Bubble Sort*, por exemplo, e imprimir as notas nessa ordem.

Não é necessário implementar e entregar esta funcionalidade, mas espero que se motive a tentar!

Observações:

- Esta atividade pode ser feita em grupo, conforme definido pelo professor;
- Não serão aceitas entregas fora do prazo;
- Organize e comente seu código fonte explicando o que faz os principais blocos/trechos programados;
- Teste e valide seu programa para todos os casos que validam as funcionalidades solicitadas no exercício, antes de enviar no Moodle;

O que entregar?

- Entregar o relatório solicitado no formato PDF de acordo com o template disponibilizado;
- Entregar os códigos fontes escritos em MIPS. Nomeie os arquivos de acordo com o exercício, exemplo: ex1.asm, ex2.asm etc. Coloque no cabeçalho o nome dos integrantes;
- Atenção: Apenas um aluno do grupo precisa submeter a atividade no Moodle.

O que será avaliado?

- Corretude do programa (deve estar “compilando sem erros” e realizando o que foi solicitado);
- O quão fiel é o programa quanto à descrição do enunciado;
- Indentação, comentários, organização e legibilidade do código;
- Qualidade e completude do relatório elaborado. Observação: os *prints* de teste inseridos no relatório precisam mostrar todos os testes que confirmam que o programa atende às diferentes solicitações do exercício.

Referência

HENNESSY, J. L.; PATTERSON, D. A.; LARUS, J. R.; MACHADO FILHO, N. Organização e projeto de computadores: a interface hardware/software. 5ª Edição, Rio de Janeiro: Elsevier, 2014.

Disponível em:

- **MINHA BIBLIOTECA - BIBLIOTECA DIGITAL** - oferece mais de 13.310 títulos e é formada pelo consórcio das **quatro principais editoras** de livros acadêmicos do Brasil: Grupo Gen - Atlas, Grupo A, Saraiva e Manole. Essas editoras se uniram para oferecer às instituições de ensino superior uma plataforma prática e inovadora para acesso a um conteúdo científico e técnico de qualidade. Ainda oferece cerca de 1.000 obras das editoras convidadas: Cengage, Cortez, Grupo Autêntica e Zahar. Cada usuário possui uma conta individual no sistema, preservando suas marcações e anotações. Dispõe de link que referencia suas citações diretas e permite imprimir parte do conteúdo.

Clique **aqui** para acessar.

<https://www.mackenzie.br/biblioteca/recursos-de-pesquisa/livros-eletronicos>

