UNIVERSIDADE DO VALE DO ITAJAÍ

NICOLE MIGLIORINI MAGAGNIN

PROGRAMAÇÃO EM LINGUAGEM DE MONTAGEM

Relatório para a disciplina de Arquitetura e Organização de computadores, do 3º Período do curso de Engenharia Da Computação da Escola do Mar, Ciências e Tecnologia.

Professor: Douglas Rossi de Melo.

Itajaí

24 de Outubro de 2019

PROPOSTA

Realizar a implementação de um procedimento de ordenação de vetores baseado em bolha (Bubble Sort) em linguagem assembly, utilizando procedimentos e pilha.

BUBBBLE SORT

Código em C++:

```
void bubblesort(int vet[], int n) {
int i, j, cond, temp;
cond = 1;
for (i=n-1; (i >= 1) && (cond == 1); i--) {
cond = 0;
        for (j=0; j < i ;j++) {
            if (vet[j+1] < vet[j]) {
                temp = vet[j];
                vet[j] = vet[j+1];
                vet[j+1] = temp;
                cond = 1;
                }
            }
}
}
```

Código em ASSEMBLY:

```
#########################################################################
#######################################
# Disciplina: Arquitetura e Organização de Computadores
# Atividade: Avaliação 02 – Programação em Linguagem de Montagem
# Bubblesort
# Aluno: Nicole Migliorini Magagnin
#
#


#########################################################################
########################
```

```
.data
Vetor: .word 0,0,0,0,0,0,0,0   #Vetor de 8 elementos
EntraVetor:.asciiz "Entre com o tamanho do vetor (máx. = 8)"       #Mensagem de solicitação do tamanho do vetor
PulaLinha:.asciiz "\n"
Invalido: .asciiz "Valor inválido"                                 #Caso o valor seja maior que 8
LeVet: .asciiz "Vetor["                                            #Lê primeira parte do vetor
FechaVet: .asciiz "] = "                                           #Fecha chaves do vetor
ElementosVet: .asciiz "Digite os elementos do vetor"
msg: .asciiz "Vetor ordenado:   "
space: .asciiz "\t"



#S0 = armazena o vetor
#S1 = armazena o número de elementos do vetor


.text


j main
#########################################################################
#Bubblesort#############################################################
Bubblesort:
addi $sp, $sp,-24
sw $s0,20($sp)                          #Armazena na memória
sw $s1,16($sp)
sw $s2,12($sp)
sw $s3,8($sp)
```

```
        sw $s4, 4($sp)
        sw $s5, 0($sp)



        add $s1, $a0, $0            #S1 = Número de vetores
        add $s2, $0, $0             #S2 = Será usado para o swap
        add $s3, $s3, $0            #S3 = Será usado para o swap
        add $s4, $0, 1              #S4 = i*4
        add $s4, $s4, $s4           #i*4
        add $s4, $s4, $s4
        addi $s1, $s1, -1           #i = n-1
        move $s0, $a1                   #Move endereço do vetor para s0



Loop1:

        la $s0, Vetor               #Carrega a primeira posição do vetor
        add $s5, $0, $0                 #Zera j
        beq $s1,$0, LoopFim         #Até que i seja igual a 0, faz o sort

Loop2:
        beq $s5,$s1, Decrementa     #Se j for igual a i
        lw $s2, 0($s0)              #Carrega j
        lw $s3, 4($s0)                  #Carrega j+1
        bgt $s2, $s3, swap          #Se j > j+1, troca
        add $s0, $s0, $s4           #Próxima posição do vetor
        addi $s5, $s5, 1            #j++
        j Loop2                         #Volta para o loop2, até que j seja = i

Decrementa:
```

```
        add $s1, $s1, -1            #i--
        j Loop1                         #Volta ao loop externo

swap:
        sw $s2, 4($s0)              #Troca o valor de j por j+1
        sw $s3, 0($s0)              #Troca o valor de j+1 por j

        beq $s5,$s1, Decrementa         #Se j for igual a i
        add $s0, $s0, $s4           #Próxima posição do vetor
        addi $s5, $s5, 1            #j++


        j Loop2


LoopFim:

        lw $s5, 0($sp)              #Desempilha
        lw $s4, 4($sp)
        lw  $s3, 8($sp)
        lw $s2, 12($sp)
        lw $s1, 16($sp)
        lw $s0, 20($sp)
        jr $ra                      #Retorna ao main

#####################################################################main#############################################################

main:
                addi $s0, $0, 1         #Incializa todos registradores em valores diferentes de zero
                addi $s1, $0, 2
```

```
                addi $s2, $0, 3
                addi $s3, $0, 4
                addi $s4, $0, 5
                addi $s5, $0, 6

                la $s0, Vetor           #Carrega endereço do vetor em $s0

Elementos:
                li $v0, 4
                la $a0, PulaLinha
                syscall
                la $s0, Vetor
                li $v0,4                #Imprime String
                la $a0, EntraVetor      #Solicita o tamanho do vetor
                syscall
                li $v0,5                #Lê o tamanho do vetor
                syscall
                add $s1, $zero, $v0     #Guarda o tamanho em s1
                blt $s1,2,invalido      #Se o número de elementos for menor que 2, é
exibido erro
                bgt $s1,8,invalido      #Se o número de elementos for maior que 8, é
exibido erro
                add $t0, $zero, $zero #Zera o contador i
                li $v0,4
                la $a0, ElementosVet #Chama mensagem "Digite os elementos do vetor"
                syscall
                li $v0, 4
                la $a0, PulaLinha
                syscall
              la $s0, Vetor             #Carrega novamente o vetor A
```

```
Leitura:
                li $v0, 4                #Chama mensagem
                la $a0, LeVet            #Chama "Vetor["
                syscall
                li $v0, 1                #Imprime int
                add $a0, $zero,$t0       #imprime indíce
                syscall
                li $v0, 4
                la $a0, FechaVet         #Fecha vetor
                syscall
                li $v0,5                 #Lê int - Número correspondente ao elemento x do
vetor
                syscall
                sw $v0, 0($s0)           #salva  o valor digitado no vetor , carregado no s0
                addi $t1, $0, 1          #i*4
                add $t1, $t1, $t1
                add $t1, $t1, $t1
                add $s0, $t1, $s0        #Soma i ao endereço-base do vetor
                addi $t0,$t0,1           # i++
                blt $t0, $s1, Leitura    #i<número de elementos continua no loop
                add $t0, $zero, $zero    #zera reg t0
                la $s0, Vetor
                move $a1, $s0                    #Carrega endereço base do vetor em a1
                add $a0, $0, $s1         #Carrega número de elementos do vetor em a0

                jal Bubblesort           #Pula para o bubblesort

                li $v0, 4
                la $a0, msg              #Imprime mensagem "Vetor ordenado: "
                syscall
                addi $t1, $0, 1          #i*4
```

```
                add $t1, $t1, $t1
                add $t1, $t1, $t1
                add $t0, $0, $0                 #Zera contador
                la $a1, Vetor           #Carrega vetor de volta em a1
########################################################################
######## Imprime
#########################################################33
Imprime:
                li $v0, 1
                lw $a0, 0($a1)          #Imprime o vetor
                syscall
                add $a1, $a1, $t1       #i*4
                addi $t0, $t0, 1        #cont++
                li $v0, 4
                la $a0, space           #Imprime espaço entre os vetores
                syscall
                blt $t0, $s1, Imprime #Até o contador ser igual ao número de elementos,
faz o loop de impressão
                j Exit                  #Senão, sai



########################################################################
#####Label inválido#########################################################
invalido:                       #Mensagem de erro sobre o número de elementos


                li $v0, 4               #Imprimir string
                la $a0, Invalido        #Mensgem de número inválido
                syscall
                j  Elementos            #Volta para elementos



Exit:
```

```
li $v0, 10          #Chama função de saída
syscall
```

EXECUÇÃO COMENTADA EM ASSEMBLY

Uso dos registradores no main:

$s0 = Salva o endereço-base do vetor.

$s1 = Número de elementos do vetor.

$a0 = Armazena o número de elementos do vetor para passá-los por parâmetro.

$a1 = Armazena o número de elementos do vetor.

$t1 = Armazena o valor de i*4.

$t0 = Contador.

Uso dos registradores no procedimento:

$s0 = Recebe novamente o endereço do vetor.

$s1 = Recebe o número de elementos do vetor e é subtraído 1, tornando-se i.

$s2 e $s3 = Usados na comparação como j e j+1.

$s4 = i*4.

$s5 = Armazena o valor do contador j.

$t7	15	0
$s0	16	1
$s1	17	2
$s2	18	3
$s3	19	4
$s4	20	5
$s5	21	6
$s6	22	0
$s7	23	0

Figura 1 - Registradores tipo S utilizados, inicializados em valores diferentes de 0

```
Entre com o tamanho do vetor (máx. = 8)9
Valor inválido
Entre com o tamanho do vetor (máx. = 8)1
Valor inválido
```

Figura 2 - Entrada de número de elementos maiores e menores que o permitido

```
Entre com o tamanho do vetor (máx. = 8)8
Digite os elementos do vetor
Vetor[0] = -10
Vetor[1] = 0
Vetor[2] = -20
Vetor[3] = 650
Vetor[4] = -60
Vetor[5] = 8
Vetor[6] = 9
Vetor[7] = 1
```

Figura 3 - Inserção válida dos elementos a serem ordenados

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	-10	0	-20	650	-60	8	9	1
0x10010020	1920233029	1868767333	544153709	1634558324	544172142	1981837156	1919906917	-512940000

Figura 4 - Elementos inseridos

$zero	0	0
$at	1	268500992
$v0	2	1
$v1	3	0
$a0	4	8
$a1	5	268500992
$a2	6	0
$a3	7	0
$t0	8	0
$t1	9	4
$t2	10	0
$t3	11	0
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	268500992
$s1	17	8
$s2	18	3
$s3	19	4
$s4	20	5
$s5	21	6
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	4194732
pc		4194308
hi		0
lo		0

Figura 5 - Chamado o procedimento, $a1 recebe o endereço do vetor e $a0 o número de elementos

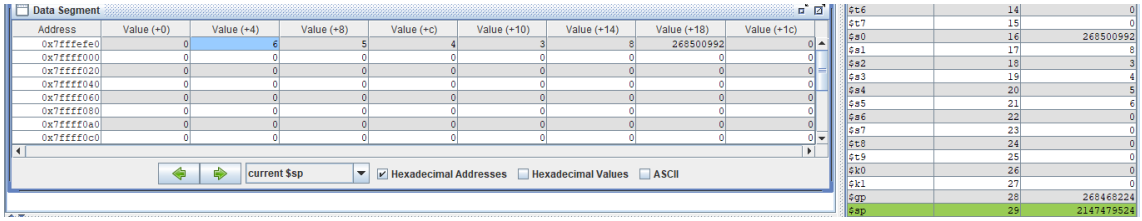

Figura 6 - Empilhamento realizado

```
Entre com o tamanho do vetor (máx. = 8)8
Digite os elementos do vetor
Vetor[0] = -10
Vetor[1] = 0
Vetor[2] = -20
Vetor[3] = 650
Vetor[4] = -60
Vetor[5] = 8
Vetor[6] = 9
Vetor[7] = 1
```

Figura 7 - Inserção dos valores do vetor

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	-60	-20	-10	0	1	8	9	650
0x10010020	1920233029	1868767333	544153709	1634558324	544172142	1981837156	1919906917	-512940000
0x10010040	1025519224	2701344	1633026058	544370540	-512332183	1868851564	1952798208	5993071

Figura 8 - Vetor ordenado

$s0	16	268500992
$s1	17	8
$s2	18	3
$s3	19	4
$s4	20	5
$s5	21	6
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0

Figura 9 - Vetor desempilhado

```
Entre com o tamanho do vetor (máx. = 8)3
Digite os elementos do vetor
Vetor[0] = 3
Vetor[1] = 2
Vetor[2] = 1
Vetor ordenado:   1     2       3
```

Figura 10 - Exemplo de ordenação

```
Entre com o tamanho do vetor (máx. = 8)5
Digite os elementos do vetor
Vetor[0] = 65
Vetor[1] = 32
Vetor[2] = 52
Vetor[3] = 6
Vetor[4] = 5
Vetor ordenado:   5     6       32      52      65
-- program is finished running --
```

Figura 11 - Exemplo de ordenação

```
Entre com o tamanho do vetor (máx. = 8)7
Digite os elementos do vetor
Vetor[0] = -50
Vetor[1] = 665
Vetor[2] = 0
Vetor[3] = 0
Vetor[4] = 5
Vetor[5] = 5
Vetor[6] = -90
Vetor ordenado:   -90   -50     0       0       5       5       665
-- program is finished running --
```

Figura 12 - Exemplo de ordenação

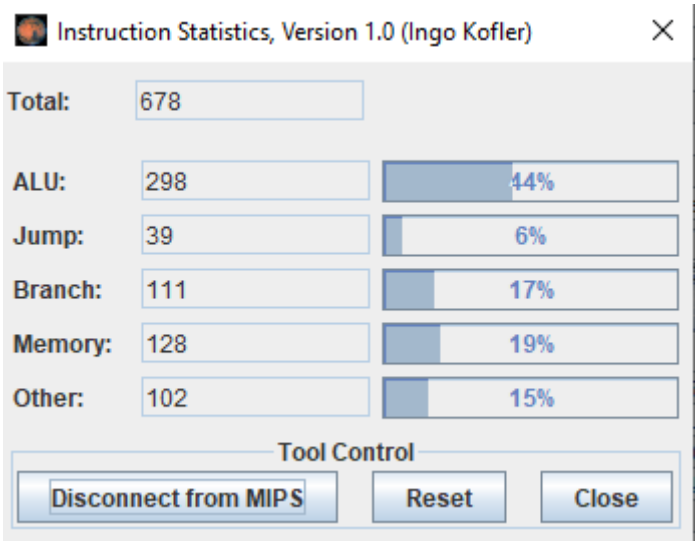

Figura 13 - Estátisticas do programa

O procedimento Bubble Sort:

Para realização do Bubble Sort, inicializei no main os registradores a serem usados em valores diferentes de 0, movi os valores de $s0 (vetor-base) para $a1 e adicionei os valores de $s1(número de elementos) em $a0 e chamei o procedimento via **jal.**

Após, empilhei todos registradores a serem utilizados, voltei o valor de $a0 para $s1 e movi novamente $a1 para $s0. Inicializei i em n-1 (Reduzindo 1 de $s1).

O Loop externo do procedimento, serviu para o for externo, sendo ele responsável por retornar a primeira posição do vetor, zerar j e dar fim ao procedimento caso i fosse zerado, a redução de i a cada loop externo, ficou por conta da label “Decrementa”.

O Loop interno é responsável por carregar a primeira posição do vetor (j) em $s2 e a próxima (j+1) em $s3, se j for maior que j+1, isso significa que haverá troca, chamando a label “swap”.

Caso haja troca, ela é realizada via sw e é incrementada a posição do vetor, via soma sucessiva realizada no início do procedimento, é verificado se j já é igual a i (a fim de voltar ao loop externo) e é incrementado o j. Caso não haja troca, o j é incrementado, a posição do vetor também e é retornado ao loop interno, para uma próxima comparação. Ao início do loop interno, é verificado novamente se j é igual a i, uma vez que quando isso acontece, é feita a diminuição de i (i--) e voltado ao Loop externo.

Quando i chega a 0, é passado ao loop final, onde os registradores são desempilhados e é dado o **jr,** de retorno ao main.