**UNIVALI – Universidade do Vale do Itajaí**

Curso – Ciências da Computação

Disciplina – Arquitetura e Organização de Processadores

Professor – Douglas Rossi de Melo

Alunos – Felipe dos Santos e Nathalia Suzin

**PROGRAMAÇÃO EM LINGUAGEM DE MONTAGEM**

22/04/2022

**Enunciado**

Implemente um programa que leia dois vetores via console e, após a leitura dos vetores, produza um terceiro vetor em que cada elemento seja o maior valor de mesmo índice dos outros dois vetores. Por fim, o programa deve imprimir esse novo vetor na tela.

**Em C++:**

//Disciplina: Arquitetura e Organização de Processadores

//Atividade: Avaliação 02 – Programação em Linguagem de Montagem

//Atividade 01

//Alunos: Felipe dos Santos e Nathalia Suzin

#include <iostream>

using namespace std;

int main()

{

int i, j, n, maior=0, vetA[8], vetB[8], vetC[8];

do{

cout << "Entre com o tamanho dos vetores (máx=8): ";

cin >> n;

if(n<1 or n>8)

cout << "\nValor inválido.\n";

}while(n<1 or n>8);

for(i=0;i<n;i++){

cout << "\nEntre com o valor Vetor A[" << i << "]: ";

cin >> vetA[i];

}

for(j=0;j<n;j++){

cout << "\nEntre com o valor Vetor B[" << j << "]: ";

cin >> vetB[j];

}

for(i=0;i<n;i++){

for(j=0;j<n;j++){

if(vetA[i]>vetB[j]){

maior=vetA[i];

vetC[i]=maior;

}else{

if(vetB[j]>vetA[i]){

maior=vetB[j];

vetC[j]=maior;

}

}

}

}

for(i=0;i<n;i++){

cout << "\nVetor C [" << i << "]: " << vetC[i];

}

return 0;

}

**Em MIPS:**

#Disciplina: Arquitetura e Organização de Processadores

#Atividade: Avaliação 02 – Programação em Linguagem de Montagem

#Exercício 01

#Alunos: Felipe dos Santos e Nathalia Suzin

.data

contSize: .asciiz "\nEntre com o tamanho dos vetores (max. = 8): "

msgInvalid: .asciiz "\nValor inválido."

contStrA: .asciiz "\nVetor\_A["

contStrB: .asciiz "\nVetor\_B["

contStrC: .asciiz "\nVetor\_C["

contEnd: .asciiz "]: "

array1: .word 0,0,0,0,0,0,0,0

array2: .word 0,0,0,0,0,0,0,0

array3: .word 0,0,0,0,0,0,0,0

.text

main:

la $s1, array1 # guarda no $s1 o endereco base do array1

la $s2, array2 # guarda no $s2 o endereco base do array2

la $s3, array3 # guarda no $s3 o endereco base do array3

addi $s4, $zero, 0 # inicializando indice=0 (i=0)

j Loop

Invalidez:

li $v0, 4 # Carrega o serviço 4 (Ponteiro para string)

la $a0, msgInvalid # Carega ptr p/ string (Mostra a mensagem na tela)

syscall # Chama o serviço 4

Loop:

li $v0, 4 # Carrega o servico 4 (Ponteiro para string)

la $a0, contSize # Carega ptr p/ string (Mostra a mensagem na tela)

syscall # Chama o servico 4

li $v0, 5 # Carrega o servico 5

syscall # Chama o servico 5

add $s0, $v0, $zero # O que o usuario digitar sera adicionado ao $s0

blt $s0, 1, Invalidez # Se o valor digitado for menor que 1, volta para "Invalidez"

bgt $s0, 8, Invalidez # Se o valor digitado for maior que 8, volta para "Invalidez"

LoopA: #RESPONSAVEL POR PEDIR OS VALORES DE 'A' E GUARDAR NA MEMEORIA

li $v0, 4 # Carrega o servico 4 (Ponteiro para string)

la $a0, contStrA # Carrega ptr p/ string (Mostra a mensagem na tela)

syscall # Chama o servico 4

li $v0, 1 # Carrega o servico 1 (inteiro)

la $a0, ($s4) # Carrega no $a0 o valor inteiro do indice

syscall # Chama o servico 1

li $v0, 4 # Carrega o servico 4 (Ponteiro para string)

la $a0, contEnd # Carrega ptr p/ string (Mostra a mensagem na tela)

syscall # Chama o servico 4

li $v0, 5 # Carrega o servico 5

syscall # Chama o servico 5

add $t2, $v0, $zero # O que o usuario digitar sera adicionado ao t2

add $t1, $s4, $s4 # $t1 = 2.i

add $t1, $t1, $t1 # $t1 = 4.i

add $t1, $t1, $s1 # $t1 = endereco base + 4.i (deslocamento) = endereco absoluto do vetor A

sw $t2, 0($t1) # $t0 = save[i]

addi $s4, $s4, 1 # i = i + 1

blt $s4, $s0, LoopA # Se indice for menor que o valor escolhido pelo usuario, volta para LoopA

addi $s4, $zero, 0 # Restaurando o indice para 0 (i=0)

LoopB: #RESPONSAVEL POR PEDIR OS VALORES DE 'B' E GUARDAR NA MEMORIA

li $v0, 4 # Carrega o servico 4 (Ponteiro para string)

la $a0, contStrB # Carrega ptr p/ string (Mostra a mensagem na tela)

syscall # Chama o servico 4

li $v0, 1 # Carrega o servico 1 (inteiro)

la $a0, ($s4) # Carrega no $a0 o valor inteiro do indice

syscall # Chama o servico 1

li $v0, 4 # Carrega o servico 4 (Ponteiro para string)

la $a0, contEnd # Carrega ptr p/ string (Mostra a mensagem na tela)

syscall # Chama o servico 4

li $v0, 5 # Carrega o servico 5

syscall # Chama o servico 5

add $t2, $v0, $zero # O que o usuario digitar sera adicionado ao t2

add $t1, $s4, $s4 # $t1 = 2.i

add $t1, $t1, $t1 # $t1 = 4.i

add $t1, $t1, $s2 # $$t1 = endereco base + 4.i (deslocamento) = endereco absoluto do vetor B

sw $t2, 0($t1) # $t0 = save[i]

addi $s4, $s4, 1 # i = i + 1

blt $s4, $s0, LoopB # Se indice for menor que o valor escolhido pelo usuario, volta para LoopB

addi $s4, $zero, 0 # Restaurando o indice para 0 (i=0)

LoopC: #RESPONSAVEL POR CARREGAR A E B, depois guardar o maior valor em C

#Carrega o valor de A[i]

add $t1, $s4, $s4 # $t1 = 2.i

add $t1, $t1, $t1 # $t1 = 4.i

add $t1, $t1, $s1 # $t1 = endereco base + 4.i (deslocamento) = endereco absoluto do vetor A

lw $t4, 0($t1) # $t2 receber o valor de A[i]

#Carrega o valor de B[i]

add $t1, $s4, $s4 # $t1 = 2.i

add $t1, $t1, $t1 # $t1 = 4.i

add $t1, $t1, $s2 # $t1 = endereco base + 4.i (deslocamento) = endereco absoluto do vetor B

lw $t3, 0($t1) # $t3 = recebe valor de A[i]

bgt $t4, $t3, Calc # Se A for maior que B vai para Calc

#Senao, vetor C vai receber valor de B

add $t1, $s4, $s4 # $t1 = 2.i

add $t1, $t1, $t1 # $t1 = 4.i

add $t1, $t1, $s3 # $t1 = endereco base + 4.i (deslocamento) = endereco absoluto do vetor C

sw $t3, 0($t1) # $t0 = save[i]

addi $s4, $s4, 1 # i = i + 1

blt $s4, $s0, LoopC # Se indice for menor que o valor escolhido pelo usuario, volta para LoopC

addi $s4, $zero, 0 # Restaurando o indice para 0 (i=0)

LoopD:

#Carrega o valor de C[i]

add $t1, $s4, $s4 # $t1 = 2.i

add $t1, $t1, $t1 # $t1 = 4.i

add $t1, $t1, $s3 # $t1 = endereco base + 4.i (deslocamento) = endereco absoluto do vetor C

lw $t4, 0($t1) # $t2 receber o valor de C[i]

li $v0, 4 # Carrega o servico 4 (Ponteiro para string)

la $a0, contStrC # Carrega ptr p/ string (Mostra a mensagem na tela)

syscall # Chama o servico 4

li $v0, 1 # Carrega o servico 1 (inteiro)

la $a0, ($s4) # Carrega no $a0 o valor inteiro do indice

syscall # Chama o servico 1

li $v0, 4 # Carrega o servico 4 (Ponteiro para string)

la $a0, contEnd # Carrega ptr p/ string (Mostra a mensagem na tela)

syscall # Chama o servico 4

li $v0, 1 # Carrega o servico 1 (inteiro)

la $a0, ($t4) # Carrega no $a0 o valor inteiro do indice

syscall # Chama o servico 1

addi $s4, $s4, 1 # i = i + 1

blt $s4, $s0, LoopD # Se indice for menor que o valor escolhido pelo usuario, volta para LoopD

j Exit

Calc:

add $t1, $s4, $s4 # $t1 = 2.i

add $t1, $t1, $t1 # $t1 = 4.i

add $t1, $t1, $s3 # $t1 = endereco base + 4.i (deslocamento) = endereço absoluto do vetor C

sw $t4, 0($t1) # $t0 = save[i]

addi $s4, $s4, 1 # i = i + 1

j LoopC

Exit:

nop # Null operation, ele ira terminar o codigo e nao fara mais nada!

**Relatório**

Inicialmente, em *.data*, são declaradas as mensagens que serão chamadas na tela pelo *syscall* e, também, declarados os tamanhos dos vetores utilizados no código. Neste caso, os vetores (*array1, array2, array3*) terão um tamanho máximo de 8 elementos, todos iniciados com 0.

.data

*contSize:* .asciiz "\nEntre com o tamanho dos vetores (max. = 8): "

*msgInvalid:* .asciiz "\nValor inválido."

*contStrA:* .asciiz "\nVetor\_A["

*contStrB:* .asciiz "\nVetor\_B["

*contStrC:* .asciiz "\nVetor\_C["

*contEnd:* .asciiz "]: "

*array1:* .word 0,0,0,0,0,0,0,0

*array2:* .word 0,0,0,0,0,0,0,0

*array3:* .word 0,0,0,0,0,0,0,0

As primeiras instruções, iniciando o programa em *.text*, terão como objetivo guardar estes *arrays* e seu índice em registradores salvos, pois serão utilizados até o fim do programa.

*array1:* .word 0,0,0,0,0,0,0,0

*array2:* .word 0,0,0,0,0,0,0,0

*array3:* .word 0,0,0,0,0,0,0,0

*main:*

la $s1, array1

la $s2, array2

la $s3, array3

addi $s4, $zero, 0

j Loop

* O *array1* será armazenado em *$s1*.

*Endereço: 0x00400000*

*Registrador Número Valor*



* O *array2* será armazenado em *$s2*.

*Endereço: 0x00400008*

*Registrador Número Valor*



* O *array3* será armazenado em *$s3*.

*Endereço: 0x00400010*

*Registrador Número Valor*



* O índice será armazenado em *$s4* e inicializado com 0.

*Endereço: 0x00400018*

*Registrador Número Valor*



Após o armazenamento aos registradores, o programa executará um *jump* para o primeiro laço.

No primeiro laço, denominado *Loop*, será apresentada uma mensagem e, após, a leitura do valor que o usuário digitar no console. Este valor é correspondente a quantidade de elementos que o indivíduo deseja que os vetores possuam, além disso, o número que o usuário digitar deve estar entre o intervalo de 1 a 8, pois 8 é a quantidade máxima de elementos que os vetores podem obter e é necessário que tenha pelo menos 1 elemento cada vetor para a funcionalidade do código.

*contSize:* .asciiz "\nEntre com o tamanho dos vetores

(max. = 8): "

*Loop:*

li $v0, 4

la $a0, contSize

syscall

li $v0, 5

syscall

add $s0, $v0, $zero

blt $s0, 1, Invalidez

bgt $s0, 8, Invalidez

O código 4 (impressão de frase) é carregado e será apresentada a mensagem guardada em *contSize* na tela:

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

*Endereço: 0x00400034*

*Registrador Número Valor*



O código 5 (leitura de valor inteiro) é carregado e o número que o usuário digitar será armazenado no registrador $s0.

*Endereço: 0x00400048*

*Registrador Número Valor*



Mas, antes de prosseguir com o restante do código, serão feitas as validações, denominadas por:

* *be less than* (*blt* – se menor que) → onde se o número digitado for menor que 1, então vá para o destino informado. ● *Endereço: 0x0040004c*
* *be greater than* (*bgt* – se maior que) → onde se o número digitado for maior que 8, então vá para o destino informado. ● *Endereço: 0x00400054*

Assim, como descrito no programa, se o valor não for validado terá um desvio para o espaço denominado *Invalidez*.

*Invalidez:*

li $v0, 4

la $a0, msgInvalid

syscall

*Endereço: 0x00400024*

*Registrador Número Valor*



Nele, é carregado o código 4 (impressão de frase) e feita uma chamada do *syscall,* que apresentará a mensagem *“Valor inválido”*, declarada em *msgInvalid*, na tela e, em seguida, retornará ao *Loop* e será refeita a pergunta até que o valor inserido seja válido, como mostrado nas imagens abaixo.

Interface gráfica do usuário, Texto, Aplicativo, Word

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Word

Descrição gerada automaticamente

Feito isto e o valor inserido validado, o programa prosseguirá para o *LoopA*. Este laço é responsável por coletar os dados do *vetor\_A* (*$s1*), ou seja, pedirá e guardará os valores do tipo inteiro inseridos em seus respectivos índices até o término das inserções, respeitando a quantidade de elementos determinada pelo usuário anteriormente, no primeiro laço, em *Loop*.

*LoopA:*

li $v0, 4

la $a0, contStrA

syscall

li $v0, 1

la $a0, ($s4)

syscall

li $v0, 4

la $a0, contEnd

syscall

li $v0, 5

syscall

add $t2, $v0, $zero

add $t1, $s4, $s4

add $t1, $t1, $t1

add $t1, $t1, $s1

sw $t2, 0($t1)

addi $s4, $s4, 1

blt $s4, $s0, LoopA

addi $s4, $zero, 0

*contStrA:* .asciiz "\nVetor\_A["

*(contador – registrador $s4)*

*contEnd:* .asciiz "]: "

O código 4 (impressão de frase) é carregado e a mensagem é enviada ao console. Observa-se que há uma chamada do índice, carregado pelo código 1 (impressão de número inteiro) e em seguida outra chamada do código 4, a qual fecha a mensagem que foi enviada. Foi feito assim para dar o efeito de contagem do índice, pois ao final, em “*addi $s4, $s4, 1*” é feito o acréscimo de 1 ao *$s4* (registrador do índice), assim como um contador. Consequentemente, é feita a leitura dos números que o usuário impor, carregados pelo código 5 (leitura de número inteiro). Assim, o LoopA continuará sendo executado até que o limite de elementos declarado pelo usuário chegue ao limite, sendo validado pelo *blt* (*be less than* – se menor que). Exemplo:

Quantidade de elementos: 3

Contador (índice): 0 a 2

Leitura: 3 números inteiros

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Frase em *contStrA*:

*Endereço: 0x00400064*

*Registrador Número Valor*





Contador em *$s4*:

*Endereço: 0x00400074*

*Registrador Número Valor*



Frase em *contEnd*:

*Endereço: 0x00400080*

*Registrador Número Valor*





Após a leitura de cada inserção, é calculado o endereço absoluto de cada índice, multiplicando por 4, ou como é feito no Assembly, soma o índice com ele mesmo e em seguida soma seu resultado com o próprio. Logo, é feita a adição do endereço base com o resultado da multiplicação por quatro e é encontrado o endereço absoluto em *$t2* pela *store word* (guarda o endereço do processador para a memória).

Primeiramente, aquilo que o usuário digitar será salvo em *$t2*:

*Endereço: 0x00400094*

*Registrador Número Valor*



Cálculo para encontrar o endereço absoluto:

add $t1, $s4, $s4 → $t1 = 2.i / $t1 = i+i

*Endereço: 0x00400098*

*Registrador Número Valor*



add $t1, $t1, $t1 → $t1 = 4.i / $t1 = result(i+i) + result(i+i)

*Endereço: 0x0040009c*

*Registrador Número Valor*



add $t1, $t1, $s1 → endereço base + 4.i (deslocamento) = endereço absoluto do vetor A

*Endereço: 0x004000a0*

*Registrador Número Valor*



sw $t2, 0($t1) → $t2 = save[i]

*Endereço: 0x004000a4*

Tabela

Descrição gerada automaticamente

Para finalizar, então, o *LoopA*, o contador é zerado em sua última instrução e prossegue para o próximo laço.

*Endereço: 0x004000b4* (Contador zerado)

*Registrador Número Valor*



O laço do *LoopB* não é tão diferente do laço do *LoopA*. Na verdade, sua única diferença é que a leitura será feita a partir dos valores inseridos no *vetor\_B* (*$s2*).

*LoopB:*

li $v0, 4

la $a0, contStrB

syscall

li $v0, 1

la $a0, ($s4)

syscall

li $v0, 4

la $a0, contEnd

syscall

li $v0, 5

syscall

add $t2, $v0, $zero

add $t1, $s4, $s4

add $t1, $t1, $t1

add $t1, $t1, $s2

sw $t2, 0($t1)

addi $s4, $s4, 1

blt $s4, $s0, LoopB

addi $s4, $zero, 0

*contStrB:* .asciiz "\nVetor\_B["

*(contador – registrador $s4)*

*contEnd:* .asciiz "]: "

Quantidade de elementos: 3

Contador (índice): 0 a 2

Leitura: 3 números inteiros

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Frase em *contStrB*:

*Endereço: 0x004000bc*

*Registrador Número Valor*





Contador em *$s4*:

*Endereço: 0x004000cc*

*Registrador Número Valor*



Frase em *contEnd*:

*Endereço: 0x004000d8*

*Registrador Número Valor*





Aquilo que o usuário digitar será salvo em *$t2*:

*Endereço: 0x004000ec*

*Registrador Número Valor*



Cálculo para encontrar o endereço absoluto:

add $t1, $s4, $s4 → $t1 = 2.i / $t1 = i+i

*Endereço: 0x004000f0*

*Registrador Número Valor*



add $t1, $t1, $t1 → $t1 = 4.i / $t1 = result(i+i) + result(i+i)

*Endereço: 0x0040009c*

*Registrador Número Valor*



add $t1, $t1, $s2 → endereço base + 4.i (deslocamento) = endereço absoluto do vetor A

*Endereço: 0x004000a0*

*Registrador Número Valor*



sw $t2, 0($t1) → $t2 = save[i]

*Endereço: 0x004000fc*

Tabela

Descrição gerada automaticamente

E para finalizar também o *LoopB*, o contador é zerado em sua última instrução e prossegue para o próximo laço.

*Endereço: 0x0040010c* (Contador zerado)

*Registrador Número Valor*



Agora, entramos no terceiro laço, o *LoopC*. Ele está responsável por carregar os vetores A e B, com os valores já inseridos, e compará-los em seus respectivos índices. Assim que ele achar o maior número entre ambos os valores de uma determinada posição, ele irá o armazenar no *vetor\_C* (*$s3*) em seu respectivo índice.

*LoopC:*

add $t1, $s4, $s4

add $t1, $t1, $t1

add $t1, $t1, $s1

lw $t4, 0($t1)

add $t1, $s4, $s4

add $t1, $t1, $t1

add $t1, $t1, $s2

lw $t3, 0($t1)

bgt $t4, $t3, Calc

add $t1, $s4, $s4

add $t1, $t1, $t1

add $t1, $t1, $s3

sw $t3, 0($t1)

addi $s4, $s4, 1

blt $s4, $s0, LoopC

addi $s4, $zero, 0

Como visto, o laço é iniciado com o cálculo de endereçamento do *vetor\_A* (*$s1*) e em seguida do *vetor\_B* (*$s2*), pois pelos endereços absolutos estarem armazenados em registradores temporários é necessário que o cálculo seja feito novamente, de um laço ao outro, para que tenhamos certeza de que estamos obtendo os vetores nas posições certas, tendo como intuito trazê-los ao *LoopC* para poder iniciar a comparação.

A seguir, há uma validação a ser feita pelo *be greater than* (*bgt* – se maior que), onde se o valor do índice em A for maior que o valor do mesmo índice em B, então a instrução desviará o código para *Calc*, senão o programa percorrerá até o fim do *LoopC* normalmente.

Vejamos as opções a seguir:

1. **CASO A MAIOR QUE B**

*Calc:*

add $t1, $s4, $s4

add $t1, $t1, $t1

add $t1, $t1, $s3

sw $t4, 0($t1)

addi $s4, $s4, 1

j LoopC

Através do desvio, em Calc, é feito o cálculo de endereçamento para que o valor armazenado em $t4 na sua respectiva posição seja guardado no vetor\_C ($s3), sempre mantendo o índice correspondente. Seguido da adição de 1 ao contador.

sw $t4, 0($t1)

*Endereço: 0x004001bc*

Tabela

Descrição gerada automaticamente

*Endereço: 0x004001c0* (Contador + 1)

*Registrador Número Valor*



Feito isto, o programa executará um *jump* que retornará ao *LoopC*.

1. **CASO B MAIOR QUE A**

*LoopC:*

add $t1, $s4, $s4

add $t1, $t1, $t1

add $t1, $t1, $s3

sw $t3, 0($t1)

addi $s4, $s4, 1

Neste caso não ocorre desvio algum. O código percorre normalmente pelo *LoopC* e faz o mesmo com o *vetor\_B* (*$s2*)o que fez com o *vetor\_A* ($s1)caso seu valor seja o maior. É calculado seu endereçamento e guardado o valor no *vetor\_C* (*$s3*), seguido da adição de 1 ao contador.

sw $t3, 0($t1)

*Endereço: 0x00400144*

Tabela

Descrição gerada automaticamente

*Endereço: 0x00400148* (Contador + 1)

*Registrador Número Valor*



Valores comparados e os maiores armazenados no *vetor\_C* (*$s3*), o contador é zerado e parte para o próximo e último laço.

O *LoopD* tem como objetivo mostrar no console os valores armazenados no *vetor\_C* em seus respectivos índices, a fim de mostrar o resultado da comparação entre o *vetor\_A* e o *vetor\_B*.

*contStrC:* .asciiz "\nVetor\_C["

*(contador – registrador $s4)*

*contEnd:* .asciiz "]: "

*LoopD:*

add $t1, $s4, $s4

add $t1, $t1, $t1

add $t1, $t1, $s3

lw $t4, 0($t1)

li $v0, 4

la $a0, contStrC

syscall

li $v0, 1

la $a0, ($s4)

syscall

*...*

li $v0, 4

la $a0, contEnd

syscall

li $v0, 1

la $a0, ($t4)

syscall

addi $s4, $s4, 1

blt $s4, $s0, LoopD

j Exit

É feito o cálculo de endereçamento em busca do endereço absoluto do *vetor\_C* para que não haja enganos sobre os valores armazenados, sendo trazido da memória ao processador pelo *load word*. Logo, o programa irá percorrer o *LoopD* até que todos os elementos do *vetor\_C* sejam apresentados no console de interface. Sua condição de parada está em *be less than* (*blt* – se menor que), onde enquanto a posição do vetor for menor que a posição máxima, continuará a transcrever os valores, adicionando 1 ao contador, novamente, para que esteja de acordo com sua posição.

lw $t4, 0($t1)

*Endereço: 0x00400164*

*Registrador Número Valor*



*Endereço: 0x00400148* (Contador + 1)

*Registrador Número Valor*



Ao final ficará assim:

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Logo, quando o limite tiver sido alcançado, o programa desviará o código através do *jump* para *Exit*, onde terminará o código e finalizará o programa.

*Exit – Endereço: 0x004001c8*

*nop*