UNIVERSIDADE DO VALE DO ITAJAÍ - UNIVALI
ENGENHARIA DE COMPUTAÇÃO
ARQUITETURA E ORGANIZAÇÃO DE PROCESSADORES

PROFESSOR: DOUGLAS ROSSI DE MELO

CAUÃ DOMINGOS
TARYCK GEAN SANTOS PEGO

AVALIAÇÃO 02: PROGRAMAÇÃO EM LINGUAGEM DE MONTAGEM

ITAJAÍ - SC
16 de Abril, 2023

Exercício: Implemente um programa que leia um vetor via console, carregue todos os elementos do vetor na memória e, após, solicite ao usuário que escolha um índice do vetor para ter o seu valor impresso. Após, o programa deve apresentar no terminal a identificação do elemento e o seu valor.

Código-fonte em linguagem C

```
    int Vetor_A[8] = {0};
    int tamanho_vetor;
    int indice = 0;
    int busca_posicao;

    while (tamanho_vetor < 2 || tamanho_vetor > 8){
        cout << "\n\n\t\tEntre com o tamanho do Vetor_A (máx.
= 8): ";
        cin >> tamanho_vetor;

        if(tamanho_vetor < 2 || tamanho_vetor > 8){
            cout << "\n\t\tValor inválido! ";
        }
    }

    while (indice < tamanho_vetor){
        cout << "\n\n\t\tVetor_A[" << indice << "]: ";
        cin >> Vetor_A[indice];
        indice++;
    }

    do{
        cout << "\n\n\t\tDigite o índice do valor a ser
impresso: ";
        cin >> busca_posicao;

        if (busca_posicao < 0 || busca_posicao >
tamanho_vetor-1){
            cout << "\n\n\t\tValor inválido! ";
        }
    } while (busca_posicao < 0 || busca_posicao >
tamanho_vetor-1);

    cout << "\n\n\n\t\tO elemento do vetor na posição " <<
busca_posicao << " possui o valor: "<< Vetor_A[busca_posicao]
<< endl << endl;
```

Conforme pedido no enunciado e nos tópicos de seus requisitos, foi implementado um código em C que interpreta como seria esta implementação em alto nível. Primeiramente foi realizado a pretensão do tópico 1:

> *“ Na seção de declaração de variáveis , o vetor deve ser declarado com 8 elementos inicializados em 0. Ele deve ser claramente identificado com um nome como Vetor_A, por exemplo. ”*

Como visto no código abaixo, o Vetor_A foi inicialmente declarado com todos os elementos 0:

```
int Vetor_A[8] = {0};
```

Após isso foi criado um laço de repetição que comporta os tópicos 2 e 3 do enunciado:

> *2. O programa deve solicitar o número de elementos do vetor, aceitando no máximo vetores com 8 elementos. Para leitura, deve ser apresentada uma mensagem solicitando a entrada desse valor, indicando o seu limite máximo. Ex: “Entre com o tamanho do vetor (máx. = 8)”.*

> *3. O programa deve solicitar a entrada do número de elementos até que ele seja maior que 1 e menor ou igual a 8. Ou seja, deve-se implementar um mecanismo de filtragem que não aceite entrada diferente da especificada. No caso de entrada invalida, o programa deve imprimir uma mensagem de advertência antes de solicitar novamente a entrada. Ex: “Valor inválido”.*

Como pode ser visto abaixo, foi implementado um laço de repetição **`while`** que funcionará infinitamente caso o valor de **`tamanho_vetor`** seja menor que 2 ou maior que 8. Foi criado uma condição dentro desse laço que, se houver a confirmação dessa condição de repetição, sempre imprimirá na tela a mensagem de “valor inválido”. Caso seja um número entre 2 e 8 acontecerá apenas a carga do valor na variável **`tamanho_vetor`**.

```
     while (tamanho_vetor < 2 || tamanho_vetor > 8){
        cout << "\n\n\t\tEntre com o tamanho do Vetor_A (máx.
= 8): ";
        cin >> tamanho_vetor;

        if(tamanho_vetor < 2 || tamanho_vetor > 8){
            cout << "\n\t\tValor inválido! ";
        }
    }
```

Temos então o tópico 4 do enunciado:

Para leitura, o programa deve solicitar ao usuário a entrada de cada elemento do vetor, um a um, com mensagens do tipo:

Vetor_A[0] =

Vetor_A[1] =

...

Foi visto como necessária a implementação de outro laço de repetição do tipo *while,* que funcionará enquanto o valor do índice seja menor que o tamanho da variável *tamanho_vetor* (as posições no vetor começam em 0 e vão até tamanho - 1). Como o índice foi declarado no valor de 0, a inserção no vetor começará na sua posição 0 também. A cada repetição será inserido um valor na posição índice e a cada repetição é incrementado 1 no valor do índice.

```
    while (indice < tamanho_vetor){
        cout << "\n\n\t\tVetor_A[" << indice << "]: ";
        cin >> Vetor_A[indice];
        indice++;
    }
```

Após todas as inserções no vetor, será feita a requisição do tópico 5:

5. Só após a entrada de todos os elementos do vetor e do armazenamento deles na memória, o programa deve solicitar o índice cujo valor será impresso, como por exemplo: “Digite o índice do valor a ser impresso: ”.

Pelo fato do vetor ter um número limitado de posições inseridas, há a necessidade de uma limitação do acesso pedido no tópico 5. Foi visto como necessário a implementação de um laço *do - while*, pois inicialmente não sabemos o valor de *busca_posicao* a fim de realizar apenas um while. A condição de repetição desse laço se dá no valor inserido em *busca_posicao*: esse valor tem que ser menor do que *tamanho_vetor* - 1 e maior do que 0 (todas as posições existentes no vetor). Dentro do laço há uma condição que monitora o valor inserido de *busca_posicao* e, se ela não estiver dentro dos padrões do laço, imprimirá na tela a mensagem “valor inválido!”.

```
    do{
        cout << "\n\n\t\tDigite o índice do valor a ser
impresso: ";
        cin >> busca_posicao;

        if (busca_posicao < 0 || busca_posicao >
tamanho_vetor-1){
            cout << "\n\n\t\tValor inválido! ";
        }
    } while (busca_posicao < 0 || busca_posicao >
tamanho_vetor-1);
```

Então, depois da permissão de todas as limitações empregadas nos códigos acima, é implementado o tópico 6 do enunciado:

> *Ao final, o programa deve informar ao usuário as informações utilizando uma mensagem como o seguinte exemplo: “O elemento do vetor na posição 5 possui o valor 23”.*

Com um fácil desenvolvimento de amostragem de valores, foi criada a última expressão do código: foi usado o *busca_posicao* para mostrar em qual posição foi retirada o valor e a mesma variável para indicar o valor inserido inserido naquela posição do vetor.

```
    cout << "\n\n\n\t\tO elemento do vetor na posição " <<
busca_posicao << " possui o valor: "<< Vetor_A[busca_posicao]
<< endl << endl;
```

Código-fonte em linguagem Assembly do MIPS

```
# Disciplina: Arquitetura e Organização de Processadores
# Atividade: Avaliação 02 – Programação em Linguagem de
Montagem
# Programa 02
# Aluno: Caua Domingos e Taryck Santos

.data # segmento de dados

     Vetor_A: .word 0, 0, 0, 0, 0, 0, 0, 0
     newline: .asciiz "\n"
     Msg1: .asciiz "\n\n Entre com o tamanho do Vetor_A (máx.
= 8): "
     Msg2: .asciiz "\n Valor inválido! "
     Msg3: .asciiz "\n\n Vetor_A["
     Msg4: .asciiz "]: "
     Msg5: .asciiz "\n\n Digite o índice do valor a ser
impresso: "
     Msg6: .asciiz "\n\n O elemento do vetor na posição "
     Msg7: .asciiz " possui o valor: "

.text # segmento de código

main:

     li $s1, 1 # Declarando s1 = 1
     li $s6, 8 # Declarando s6 = 8

     # Print na mensagem 1
     li $v0, 4 # Chama o serviço 4 (print_string)
     la $a0, Msg1 # Msg1
     syscall

     # Lê a variável
     li $v0, 5 # Chama o serviço 5 (read_int)
     syscall

     move $s7, $v0 # Move o inteiro para o endereço s7

     slt $t8, $s6, $s7 # Caso o inteiro inserido seja maior
que 8, t8 = 1
     beq $t8, $s1, mensagem_invalida # Caso t8 e s1 sejam
iguais, saltam para "mensagem invalida"

     li $t8, 0 # limpa t8

     slti $t8, $s7, 2 # caso s7 seja menor que 2, t8 = 1
     beq $t8, $s1, mensagem_invalida # Caso t8 e s1 sejam
```

```
iguais, saltam para "mensagem invalida"

     loop_inserir_vetor:

          # Print na mensagem 3
          li $v0, 4
          la $a0, Msg3
          syscall

          # Print na constante $t0
          li $v0, 1 # Chama o serviço 1 (print_int)
          move $a0, $t0 # Carrega a constante de $t0 para o
syscall
          syscall

          # Print na mensagem 4
          li $v0, 4
          la $a0, Msg4
          syscall

          # Lê a variável
          li $v0, 5 # Chama o serviço 5 (read_int)
          syscall

          move $t1, $v0 # Move o inteiro para o endereço t1

          sw $t1, Vetor_A($s0) # Coloca na posição $s0 do
Vetor_A o valor contido em $t1

          addi $t0, $t0, 1 # adiciona 1 em t0
          addi $s0, $s0, 4 # adiciona 4 (1 word) em s0
          bne $t0, $s7, loop_inserir_vetor # Caso $t0 seja
diferente de $s7 retorna ao loop

     inserir_indice_busca:

          li $t0, 0 # zera t0
          li $t1, 0 # zera t1
          li $s0, 0 # zera s0

          # Print na mensagem 5
          li $v0, 4
          la $a0, Msg5
          syscall

          # Lê a variável
          li $v0, 5 # Chama o serviço 5 (read_int)
          syscall

          move $s2, $v0 # Move o inteiro para o endereço s2
```

```
          sub $s5, $s7, $s1 # Subtrai 1 de s7 e insere em s5
(para a busca no vetor que vai de 0 até s7 - 1)

          slt $t8, $s5, $s2 #  Caso o indice inserido seja
maior que o numero de posições no vetor, t8 = 1
          beq $t8, $s1, mensagem_invalida_busca # se t8 for
igual a s1

          li $t8, 0

          slti $t8, $s2, 0 #  Caso o indice inserido seja
menor que 0, t8 = 1
          beq $t8, $s1, mensagem_invalida_busca # se t8 for
igual a s1

          li $s0, 0 # zera s0

          j loop_buscar_vetor

     mensagem_invalida: # função de print para ser chamada na
invalidez do tamanho do vetor

          # Print na mensagem 2
          li $v0, 4
          la $a0, Msg2
          syscall

          j main

     mensagem_invalida_busca: # função de print para ser
chamada na invalidez da busca no vetor

          # Print na mensagem 2
          li $v0, 4
          la $a0, Msg2
          syscall

          j inserir_indice_busca

     loop_buscar_vetor: # função de buscar no vetor

          beq $t0, $s2, mostrar_vetor_buscado # caso a
contagem de posição for igual a posição buscada, mostrar o
valor naquela posição do vetor

          # caso $t0 != $s2:
          addi $s0, $s0, 4 # adiciona 4 (1 word) em s0
          addi $t0, $t0, 1 # adiciona 1 em t0
          j loop_buscar_vetor # retorna ao início do loop
```

```
     mostrar_vetor_buscado: # Mostra o vetor buscado

          # Print na mensagem 6
          li $v0, 4
          la $a0, Msg6
          syscall

          # Print na constante $t0
          li $v0, 1
          move $a0, $t0
          syscall

          # Print na mensagem 7
          li $v0, 4
          la $a0, Msg7
          syscall

          lw $s3, Vetor_A($s0) # carrega em s3 o valor
contido naquela posição do vetor

          # Print na constante $s3
          li $v0, 1
          move $a0, $s3
          syscall
```

Código-fonte em linguagem Assembly do RISCV

```
# Disciplina: Arquitetura e Organização de Processadores
# Atividade: Avaliação 02 – Programação em Linguagem de
Montagem
# Programa 01
# Aluno: Caua Domingos e Taryck Santos

.data # segmento de dados

     Vetor_A: .word 0, 0, 0, 0, 0, 0, 0, 0
     newline: .asciz "\n"
     Msg1: .asciz "\n\n Entre com o tamanho do Vetor_A (máx.
= 8): "
     Msg2: .asciz "\n Valor inválido! "
     Msg3: .asciz "\n\n Vetor_A["
     Msg4: .asciz "]: "
     Msg5: .asciz "\n\n Digite o índice do valor a ser
impresso: "
     Msg6: .asciz "\n\n O elemento do vetor na posição "
     Msg7: .asciz " possui o valor: "

.text # segmento de código
```

```
main:

     li s1, 1 # Declarando s1 = 1
     li s6, 8 # Declarando s6 = 8
     la s0, Vetor_A # Declarando s0 como posição 0 do vetor

     # Print na mensagem 1
     li a7, 4 # Chama o serviço 4 (print_string)
     la a0, Msg1 # Msg1
     ecall

     # Lê a variável
     li a7, 5 # Chama o serviço 5 (read_int)
     ecall

     mv s7, a0 # Move o inteiro para o endereço s7

     slt t6, s6, s7 # Caso o inteiro inserido seja maior que
8, t6 = 1
     beq t6, s1, mensagem_invalida # Caso t6 e s1 sejam
iguais, saltam para "mensagem invalida"

     li t6, 0 # limpa t6

     slti t6, s7, 2 # caso s7 seja menor que 2, t6 = 1
     beq t6, s1, mensagem_invalida # Caso t6 e s1 sejam
iguais, saltam para "mensagem invalida"

     loop_inserir_vetor:

          # Print na mensagem 3
          li a7, 4
          la a0, Msg3
          ecall

          # Print na constante t0
          li a7, 1 # Chama o serviço 1 (print_int)
          mv a0, t0 # Carrega a constante de t0 para o
syscall
          ecall

          # Print na mensagem 4
          li a7, 4
          la a0, Msg4
          ecall

          # Lê a variável
          li a7, 5 # Chama o serviço 5 (read_int)
          ecall
```

```
          mv t1, a0 # Move o inteiro para o endereço t1

          sw t1, 0(s0) # Coloca na posição s0 do Vetor_A o
valor contido em t1

          addi t0, t0, 1 # adiciona 1 em t0
          addi s0, s0, 4 # adiciona 4 (1 word) em s0
          bne t0, s7, loop_inserir_vetor # Caso t0 seja
diferente de s7 retorna ao loop

     inserir_indice_busca:

          li t0, 0 # zera t0
          li t1, 0 # zera t1
          la s0, Vetor_A # Declarando s0 como posição 0 do
vetor

          # Print na mensagem 5
          li a7, 4
          la a0, Msg5
          ecall

          # Lê a variável
          li a7, 5 # Chama o serviço 5 (read_int)
          ecall

          mv s2, a0 # Move o inteiro para o endereço s2

          sub s5, s7, s1 # Subtrai 1 de s7 e insere em s5
(para a busca no vetor que vai de 0 até s7 - 1

          slt t6, s5, s2 #  Caso o indice inserido seja maior
que o numero de posições no vetor, t6 = 1
          beq t6, s1, mensagem_invalida_busca # se t6 for
igual a s1, pula pra mensagem_invalida_busca

          li t6, 0 # zera t6

          slti t6, s2, 0 #  Caso o indice inserido seja menor
que 0, t6 = 1
          beq t6, s1, mensagem_invalida_busca # se t6 for
igual a s1, pula pra mensagem_invalida_buscala s0, Vetor_A #

          j loop_buscar_vetor

     mensagem_invalida: # função de print para ser chamada na
invalidez do tamanho do vetor

          # Print na mensagem 2
          li a7, 4
```

```
          la a0, Msg2
          ecall

          j main

     mensagem_invalida_busca: # função de print para ser
chamada na invalidez da busca no vetor

          # Print na mensagem 2
          li a7, 4
          la a0, Msg2
          ecall

          j inserir_indice_busca

     loop_buscar_vetor: # função de buscar no vetor

          beq t0, s2, mostrar_vetor_buscado # caso a contagem
de posição for igual a posição buscada, pula pra
mostrar_vetor_buscado

          # caso $t0 != $s2:
          addi s0, s0, 4 # adiciona 4 (1 word) em s0
          addi t0, t0, 1 # adiciona 1 em t0
          j loop_buscar_vetor # retorna ao início do loop

     mostrar_vetor_buscado: # Mostra o vetor buscado

          # Print na mensagem 6
          li a7, 4
          la a0, Msg6
          ecall

          # Print na constante t0
          li a7, 1
          mv a0, t0
          ecall

          # Print na mensagem 7
          li a7, 4
          la a0, Msg7
          ecall

          lw t5, 0(s0) # carrega em t5 o valor contido
naquela posição do vetor

          # Print na constante s3
          li a7, 1
          mv a0, t5
          ecall
```

Para fazer a implementação em linguagem Assembly, foi usado o código em C como parâmetro. No primeiro momento, dentro do segmento de dados foram declarados o vetor com as 8 posições inicializadas em 0, e as mensagens que virão a ser utilizadas pelos syscall e ecall.

Seguindo, após as declarações das constantes 1 e 8 em registradores (utilizados para fins de comparação), foram utilizadas as primeiras chamadas de sistema: para impressão da mensagem 1 e para a leitura da constante do tamanho do vetor.

Equivalente as comparações feitas no *while* no código em C, foram realizadas as comparações no Assembly:

Caso o número inserido for maior que s6 (8)

```
    slt $t8, $s6, $s7
    beq $t8, $s1, mensagem_invalida
```

Caso o número inserido for menor que 2

```
    slti $t8, $s7, 2
    beq $t8, $s1, mensagem_invalida
```

Caso o elemento inserido seja maior ou menor que as constantes usadas como parâmetro, o compilador irá para a função *mensagem_invalida*, onde será mostrada a mensagem 2 e voltará para o início do código. Segue a implementação da função:

```
mensagem_invalida:

          li $v0, 4
          la $a0, Msg2
          syscall

          j main
```

Caso o número de posições do vetor se encaixe nos padrões estabelecidos pelas comparações anteriores, o código se encaminha para a função de inserção dos valores no vetor. Primeiramente são feitas as chamadas de mostra de mensagens e constantes (índices), e após são feitas as comparações do loop.

Usando t0 como contador e comparador, s0 como índice de posição no vetor e t1 como auxiliar para a inserção, o loop é utilizado até que t0 seja igual ao tamanho do vetor (contagem começa do 0 e vai até tamanho -1). Caso t0 seja diferente do tamanho total, o loop continuará a ser feito. A cada loop completo, o valor de s0 aumenta em 4 (1 word). Esse laço é implementado da seguinte maneira:

Loop de inserção nos vetores - MIPS

```
     loop_inserir_vetor:

          …(mensagens e leitura de constantes)…

          move $t1, $v0 # Move o inteiro para o endereço t1

          sw $t1, Vetor_A($s0) # Coloca na posição s0 do
Vetor_A o valor contido em $t1

          addi $t0, $t0, 1 # adiciona 1 em t0
          addi $s0, $s0, 4 # adiciona 4 (1 word) em s0
          bne $t0, $s7, loop_inserir_vetor # Caso $t0 seja
diferente de $s7 retorna ao loop
```

Loop de inserção nos vetores - RISCV

```
     loop_inserir_vetor:

          …(mensagens e leitura de constantes)…

          sw t1, Vetor_A, s0 # Coloca na posição s0 do
Vetor_A o valor contido em t1

          addi t0, t0, 1 # adiciona 1 em t0
          addi s0, s0, 4 # adiciona 4 (1 word) em s0
          bne t0, s7, loop_inserir_vetor # Caso t0 seja
diferente de s7 retorna ao loop
```

Após a leitura exitosa dos valores, o código seguirá para o chamado de busca. Os registradores usados anteriormente são zerados para a reutilização dos mesmos evitando a sobrecarga de valores. Após isso é chamada a mensagem 5, a leitura do índice para busca e o movimento do inteiro para o registrador s2.

É subtraído 1 do valor de s7 (e colocado em s5) para que as buscas sejam feitas apenas nas posições existentes do vetor (de 0 até tamanho -1), e da mesma forma que foram feitas as regras da inserção do tamanho do vetor, foram feitas as regras de busca: caso o número seja menor que 0 ou maior que tamanho -1, o loop será acionado - caso contrário, o código seguirá em frente. Segue a implementação do loop de busca:

```
inserir_indice_busca:

          …(Chamadas do sistema e movimento da constante)…

          sub $s5, $s7, $s1 # Subtrai 1 de s7 e insere em s5
(para a busca no vetor que vai de 0 até s7 - 1)

          slt $t8, $s5, $s2 #  Caso o indice inserido seja
maior que o número de posições no vetor, t8 = 1
          beq $t8, $s1, mensagem_invalida_busca # se t8 for
igual a s1, irá para a chamada da mensagem invalida

          li $t8, 0 # limpeza de t8

          slti $t8, $s2, 0 #  Caso o indice inserido seja
menor que 0, t8 = 1
          beq $t8, $s1, mensagem_invalida_busca # se t8 for
igual a s1, irá para a chamada da mensagem invalida

          li $s0, 0 # zera s0

          j loop_buscar_vetor
```

Após a verificação exitosa da inserção do índice de busca, essa função de inserção pula para o loop de busca dentro do vetor. Este loop faz com que t0 (índice) e s0 (posição no vetor) se atualizem utilizando a posição buscada como parâmetro de comparação.

Enquanto t0 não for igual a s2 (posição desejada), o loop atualiza o valor de t0 incrementando 1 e atualiza o valor de s0 incrementando 4. No momento que t0 se torna igual a s2, há um *jump* para a função mostrar_vetor_buscado.

```
loop_buscar_vetor: # função de buscar no vetor

          beq $t0, $s2, mostrar_vetor_buscado # caso a
contagem de posição for igual a posição buscada, mostrar o
valor naquela posição do vetor

          # caso $t0 != $s2:
          addi $s0, $s0, 4 # adiciona 4 (1 word) em s0
          addi $t0, $t0, 1 # adiciona 1 em t0
          j loop_buscar_vetor # retorna ao início do loop
```

Na função mostrar_vetor_buscado, são feitas as mostrar do syscall e ecall para as frases e valores desejados. O t0 é utilizado como valor de posição, em s3 é inserido o valor do Vetor_A na posição s0 e no final é mostrado o conteúdo de s3.

```
     mostrar_vetor_buscado:

          # Print na mensagem 6
          li $v0, 4
          la $a0, Msg6
          syscall

          # Print na constante $t0
          li $v0, 1
          move $a0, $t0
          syscall

          # Print na mensagem 7
          li $v0, 4
          la $a0, Msg7
          syscall

          lw $s3, Vetor_A($s0) # carrega em s3 o valor
contido naquela posição do vetor

          # Print na constante $s3
          li $v0, 1
          move $a0, $s3
          syscall
```

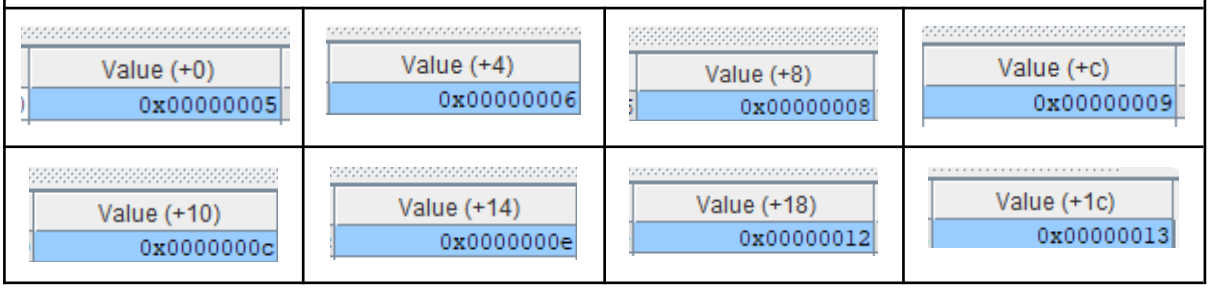
Teste de verificação do número de elementos do vetor - MIPS	Teste de verificação do número de elementos do vetor - RISCV
Entre com o tamanho do Vetor_A (máx. = 8): 0 Valor inválido! Entre com o tamanho do Vetor_A (máx. = 8): 1 Valor inválido! Entre com o tamanho do Vetor_A (máx. = 8): 9 Valor inválido! Entre com o tamanho do Vetor_A (máx. = 8): 8 Vetor_A[0]: 5 Vetor_A[1]: 6 Vetor_A[2]: 8 Vetor_A[3]: 9 Vetor_A[4]: 12 Vetor_A[5]: 14 Vetor_A[6]: 18 Vetor_A[7]: 19 Digite o índice do valor a ser impresso: 4	Entre com o tamanho do Vetor_A (máx. = 8): 0 Valor inválido! Entre com o tamanho do Vetor_A (máx. = 8): 1 Valor inválido! Entre com o tamanho do Vetor_A (máx. = 8): 9 Valor inválido! Entre com o tamanho do Vetor_A (máx. = 8): 8 Vetor_A[0]: 7 Vetor_A[1]: 9 Vetor_A[2]: 20 Vetor_A[3]: 14 Vetor_A[4]: 5 Vetor_A[5]: 6 Vetor_A[6]: 3 Vetor_A[7]: 18 Digite o índice do valor a ser impresso: 7
Entre com o tamanho do Vetor_A (máx. = 8): 0 Valor inválido! Entre com o tamanho do Vetor_A (máx. = 8): 1 Valor inválido! Entre com o tamanho do Vetor_A (máx. = 8): 9 Valor inválido! Entre com o tamanho do Vetor_A (máx. = 8): 2 Vetor_A[0]: 1 Vetor_A[1]: 2 Digite o índice do valor a ser impresso:	Entre com o tamanho do Vetor_A (máx. = 8): 0 Valor inválido! Entre com o tamanho do Vetor_A (máx. = 8): 1 Valor inválido! Entre com o tamanho do Vetor_A (máx. = 8): 9 Valor inválido! Entre com o tamanho do Vetor_A (máx. = 8): 2 Vetor_A[0]: 8 Vetor_A[1]: 5 Digite o índice do valor a ser impresso:

Quanto ao teste de verificação da quantidade de elementos do vetor, foi comprovado o funcionamento perfeito, onde o número mínimo e máximo da quantidade de vetor são respeitados. Quando é inserido um valor menor que dois e maior que 8, o console avisa que estes números são inválidos para aquela aplicação.

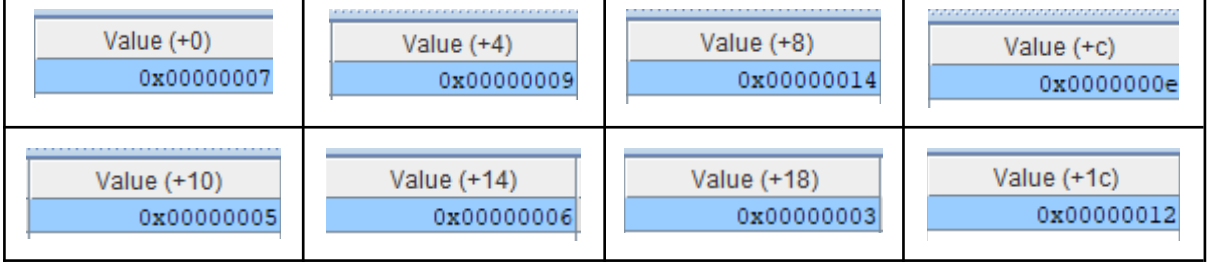
Teste de verificação do índice de busca - MIPS	**Teste de verificação do índice de busca - RISCV**
Digite o índice do valor a ser impresso: 9 Valor inválido! Digite o índice do valor a ser impresso: -1 Valor inválido! Digite o índice do valor a ser impresso: 8 Valor inválido! Digite o índice do valor a ser impresso: 7 O elemento do vetor na posição 7 possui o valor: 19	Digite o índice do valor a ser impresso: 9 Valor inválido! Digite o índice do valor a ser impresso: -1 Valor inválido! Digite o índice do valor a ser impresso: 8 Valor inválido! Digite o índice do valor a ser impresso: 7 O elemento do vetor na posição 7 possui o valor: 18
Digite o índice do valor a ser impresso: -1 Valor inválido! Digite o índice do valor a ser impresso: 3 Valor inválido! Digite o índice do valor a ser impresso: 2 Valor inválido! Digite o índice do valor a ser impresso: 1 O elemento do vetor na posição 1 possui o valor: 2	Digite o índice do valor a ser impresso: -1 Valor inválido! Digite o índice do valor a ser impresso: 3 Valor inválido! Digite o índice do valor a ser impresso: 2 Valor inválido! Digite o índice do valor a ser impresso: 1 O elemento do vetor na posição 1 possui o valor: 5

No teste de verificação do índice de busca, o valor do índice não pode ser menor que 0 e não pode ser maior do que o maior índice pertencente ao vetor (maior índice = tamanho do vetor -1). Como pode ser visto nas imagens acima, o funcionamento se mostrou perfeito.

Registradores após a execução - MIPS



Registradores após a execução - RISCV



Nas tabelas acima é possível ver os valores inseridos ocupando cada posição do vetor que foi direcionada para eles. Quando s0 = 0 (primeira posição do vetor) o primeiro valor inserido vai para a posição 0 (Value (+0)). A cada rotação do loop é incrementado 4 no endereço, onde se torna perceptível que a partir do segundo valor em diante a constante vai para o local apontado para ela: ou seja, a cada loop o valor de s0 aumenta 4 e vai para um novo endereço (Value +4, +8, +c, etc) verificando o funcionamento correto da manipulação do vetor.

Teste de verificação do sistema de busca - MIPS	Teste de verificação do sistema de busca - RISCV
Vetor_A[0]: 5 Vetor_A[1]: 6 Vetor_A[2]: 8 Vetor_A[3]: 9 Vetor_A[4]: 12 Vetor_A[5]: 14 Vetor_A[6]: 18 Vetor_A[7]: 19 Digite o índice do valor a ser impresso: 4 O elemento do vetor na posição 4 possui o valor: 12	Vetor_A[0]: 7 Vetor_A[1]: 9 Vetor_A[2]: 20 Vetor_A[3]: 14 Vetor_A[4]: 5 Vetor_A[5]: 6 Vetor_A[6]: 3 Vetor_A[7]: 18 Digite o índice do valor a ser impresso: 2 O elemento do vetor na posição 2 possui o valor: 20

Após a análise do código e dos resultados obtidos pelas simulações, tanto na linguagem Assembly do MIPS quanto na do RISC-V, é possível afirmar que ambos funcionam corretamente. Os códigos foram desenvolvidos seguindo as especificações necessárias e os resultados obtidos condizem com o que foi proposto. Os testes realizados mostraram que as instruções foram corretamente implementadas, permitindo que os valores fossem corretamente carregados nos registradores indicados e, quando necessário, os acessos a memória também.
