Relatório

Cauã Ribas, Haran Souza, Nilson Andrade

Universidade do Vale do Itajaí - Univali
Escola do Mar, Ciência e Tecnologia
Ciência da Computação
(cauaribas, haran ,nilson.neto) @edu.univali.br

Arquitetura e Organização de Processadores

Avaliação 01 - Programação em linguagem de montagem

Thiago Felski Pereira

02/10/2023

1. Introdução:

Este relatório descreve a implementação de um programa usando a Linguagem de Montagem do Risc-V, O objetivo deste programa é ler um vetor de inteiros de um tamanho especificado pelo usuário via console, ordenar o vetor em ordem crescente usando o algoritmo de ordenação bolha (bubble sort) e, por fim, imprimir o vetor ordenado na tela.

2. Programa:

2.1 Enunciado: Implemente um programa que leia um vetor via console e, após a leitura, ordene o vetor. Por fim, o programa deve imprimir esse novo vetor ordenado na tela.

2.2 Código fonte em Linguagem de Alto Nível C/C++

```
int v[8]={1,2,3,4,5,6,7,8};
int temp;
for (int i=0; i<7; i++) {
        for (int j=0; j<7; j++) {
                if ( v[j]>v[j+1] ) {
                temp=v[j];
                v[j]=v[j+1];
                v[j+1]=temp;
                 }
        }
 }
```

2.3 Código fonte em Linguagem de Montagem do Risc-V

Explicação Lógica do Código:

O programa solicita ao usuário o tamanho do vetor, em seguida, solicita ao usuário os valores que compõem o vetor, em seguida o vetor passa por um algoritmo de ordenação bolha (buuble sort), e por fim o vetor ordenado e impresso na tela.

O código é organizado em seções de dados e código (.data e .text, respectivamente). As variáveis e mensagens de texto são definidas na seção de dados, enquanto a lógica do programa está na seção de código.

O programa começa solicitando ao usuário que digite o tamanho do vetor desejado. O tamanho deve estar entre os valores mínimos e máximos permitidos (definidos pelas variáveis min_tam e max_tam). O programa continua pedindo ao usuário até que um tamanho válido seja fornecido.

Após o tamanho válido do vetor ser especificado, o programa entra em um loop para ler os elementos do vetor um por um a partir da entrada do usuário. Cada elemento é impresso na tela com um rótulo que mostra a posição

no vetor (índice). Os elementos lidos são armazenados no vetor array na posição apropriada.

Neste código, os registradores s0, s1, s2, s3, s4, s5, e s6 são alocados para uso, começando em s0 para armazenar o vetor e os valores temporários durante a ordenação. O restante dos registradores t0, t1, t2, e t3 são usados para contadores e cálculos temporários.

Após a entrada do vetor ser concluída, o programa entra na fase de ordenação. Ele implementa o algoritmo de ordenação bolha, que compara pares de elementos adjacentes no vetor e os troca se estiverem fora de ordem. O algoritmo continua fazendo isso até que nenhum elemento precise ser trocado, o que indica que o vetor está ordenado.

Depois de ordenar o vetor, o programa imprime a mensagem "Vetor ordenado:" e, em seguida, itera sobre o vetor ordenado, imprimindo cada elemento na tela, juntamente com seu índice.

Por fim, o programa faz um syscall Exit para encerrar a execução do código.

Código do Risc-V:

```
# Disciplina: Arquitetura e Organização de Processadores
# Programa: Ler um vetor via console, ordenar o vetor, e por fim imprimir o vetor na tela
# Atividade M1
# Grupo: - Cauã Ribas, Haran Souza, Nilson Andrade

.data # Dados
	array: .word 0,0,0,0,0,0,0,0
	min_tam: .word 2
	max_tam: .word 8
	texto_input_max_tam: .asciz "Digite o tamanho do vetor (max: 8): "
	texto_input_vetor: .asciz "Vetor["
	texto_fecha_index: .asciz "] = "
	texto_output_vetor_ordenado: .asciz "\nVetor ordenado:"
	texto_output_index: .asciz "\nVetor["

.text # Codigo
	lw t0, min_tam
	lw t1, max_tam
```

```
        la s0, array # Carrega o vetor no registrador s0

loop_define_tam_vetor:
        # Imprime: String texto_input_max_tam, ate que o usuario defina um valor valido para o tamanho do vetor
        addi a7, zero, 4 # Adiciona o valor 4 ao registrador de serviço a7 (PrintString)
        la a0, texto_input_max_tam # Carrega o texto texto_input_max_tam no registrador a0
        ecall # Chama o syscall

        # Solicita: Int tamanho do vetor
        addi a7, zero, 5 # Adiciona o valor 5 ao registrador de serviço a7 (ReadInt)
        ecall # Chama o syscall

        blt a0, t0, loop_define_tam_vetor # Verifica se a0 é menor que o tamanho mínimo
        bgt a0, t1, loop_define_tam_vetor # Verifica se a0 é maior que o tamanho máximo

        addi s1, a0, 0 # Atualiza o limite do vetor
        add t0, zero, zero # Carrega o valor 0 ao registrador t0, (contador/i)

for:
        bge t0, s1, fim_for # se t0 >= s1 termina o loop

        # Imprime: String texto_input_vetor
        addi a7, zero, 4 # Adiciona o valor 4 ao registrador de serviço a7 (PrintString)
        la a0, texto_input_vetor # Carrega o texto texto_input_vetor no registrador a0
        ecall # Chama o syscall

        # Imprime: Int contador/i
        addi a7, zero, 1 # Adiciona o valor 1 ao registrador de serviço a7
        add a0, zero, t0 # Carrega ao registrador a0 o contador
```

```
        ecall # Chama o syscall

        # Imprime: String texto_fecha_index
        addi a7, zero, 4 # Adiciona o valor 4 ao registrador de serviço a7 (PrintString)
        la a0, texto_fecha_index # Carrega o texto texto_fecha_index no registrador a0
        ecall # Chama o syscall

        # Solicita: Int valor para o vetor
        addi a7, zero, 5 # Adiciona o valor 5 ao registrador de serviço a7 (ReadInt)
        ecall # Chama o syscall
        add s2, zero, a0 # s2 recebe o valor digitado

        # Armazena
        slli t1, t0, 2 # Move 2 bits para a esquerda: 4 * i
        add s3, s0, t1 # Calcula a posicao no vetor desde o seu comeco: comeco do vetor + (4 * i)

        # Adiciona o valor na posicao calculada do vetor
        sw s2, 0(s3) # Guarda o valor informado pelo usuario no s2, com um offset de 0 bits, no vetor (s0)

        addi t0, t0, 1 # Incremento do contador: i = i + 1
        jal zero, for # "Reinicia o loop"

fim_for:

ordenacao:
        # Ordenação:
        add t0, zero, zero # Define o contador para o valor 0, (int i = 0)
        add t1, zero, zero # Define o contador para o valor 0, (int j = 0)

loop_externo:
        bge t0, s1, fim_loop_externo  # Se i(t0) >= tamanho do vetor(s1), termina a ordenação
```

```
    loop_interno:
        bge t1, s1, fim_loop_interno  # Se j >= tamanho do vetor, termina a iteração interna
        addi t2, t1, 1 # Incremento do contador: = j + 1
        bge t2, s1, fim_loop_interno  # Se j + 1 >= tamanho do vetor, termina a iteração interna

        # Carrega Vetor[j] em s4
        slli t3, t1, 2  # t3 = 4 * t1 (deslocamento para a posição j)
        add s4, s0, t3  # s4 = base (s0) + deslocamento (t3)
        lw s5, 0(s4)  # s5 carrega o valor do array Vetor[j] s5 = vetor[j]

        # Carrega Vetor[j+1] em s6
        slli t3, t2, 2 # Move 2 bits para a esquerda: 4 * i(deslocamento para a posição j+1)
        add s4, s0, t3 # Calcula a posicao no vetor desde o seu comeco: comeco do vetor + t3(4 * i)
        lw s6, 0(s4) # Carrega o valor do vetor[j+1] no registrador s6

        bgt s5, s6, ordenar_vetor  # Se Vetor[j] > Vetor[j+1], troque os valores

        jal zero, incremento_loop_interno  # Caso contrario, continue com a próxima iteração

    ordenar_vetor:
        # Troca os valores do vetor[j] e vetor[j+1]
        slli t3, t1, 2  # t3 = 4 * t1 (deslocamento para a posição j)
        add s4, s0, t3  # s4 = base (s0) + deslocamento (t3)
        sw s6, 0(s4)  # Armazena o valor s6 no array na posição j

        # Troca os valores do vetor[j+1] e vetor[j]
        slli t3, t2, 2  # t3 = 4 * t2 (deslocamento para a posição j + 1)
        add s4, s0, t3  # s4 = base (s0) + deslocamento (t3)
        sw s5, 0(s4)  # Armazena o valor s5 no array na posição j + 1
```

```
            jal zero, incremento_loop_interno  # Continua com a próxima
iteração

        incremento_loop_interno:
            addi t1, t1, 1 # Incremento do contador: j = j + 1
            jal zero, loop_interno  # Volta para o loop interno(loop_interno)

        fim_loop_interno:
            add t1, zero, zero # Reinicia o contador para o valor 0, (int j = 0)
            addi t0, t0, 1 # Incremento do contador: i = i + 1
            jal zero, loop_externo  # Volta para o loop externo(loop_externo)

fim_loop_externo:
        # Imprime: String texto_output_vetor_ordenado
        addi a7, zero, 4 # Adiciona o valor 4 ao registrador de serviço a7
(PrintString)
        la a0, texto_output_vetor_ordenado # Carrega o texto
texto_output_vetor_ordenado no registrador a0
        ecall # Chama o syscall

        add t0, zero, zero # Define o contador para o valor 0, (int i = 0)

for2:
        bge t0, s1, fim_for2 # se t0 >= s1 termina o loop

        # Imprime: String texto_output_index
        addi a7, zero, 4 # Adiciona o valor 4 ao registrador de serviço a7
(PrintString)
        la a0, texto_output_index # Carrega o texto_output_index no registrador
a0
        ecall # Chama o syscall

        # Imprime: Int contador/i
        addi a7, zero, 1 # Adiciona o valor 1 ao registrador de serviço a7
        add a0, zero, t0 # Carrega ao registrador a0 o contador
        ecall # Chama o syscall
```

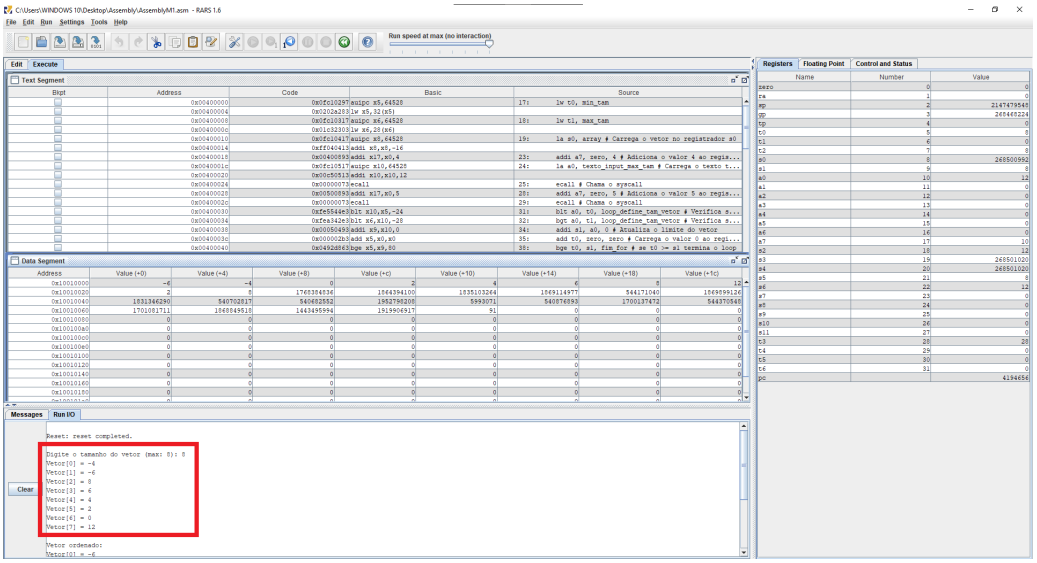
```
        # Imprime: String texto_fecha_index
        addi a7, zero, 4 # Adiciona o valor 4 ao registrador de serviço a7
(PrintString)
        la a0, texto_fecha_index # Carrega o texto_fecha_index no registrador a0
        ecall # Chama o syscall

        # Armazena
        slli t3, t0, 2 # Move 2 bits para a esquerda: 4 * i
        add s4, s0, t3 # Calcula a posicao no vetor desde o seu comeco: comeco
do vetor + (4 * i)
        lw a0, 0(s4) # Carrega o vetor no registrador s0

        # Imprime: Int contador/i
        addi a7, zero, 1 # Adiciona o valor 1 ao registrador de serviço a7
        ecall # Chama o syscall

        addi t0, t0, 1 # Incremento do contador: i = i + 1

        jal zero, for2 # "Reinicia" o loop
fim_for2:

        addi a7, zero, 10 # Encerra o programa com Exit(10)
        ecall # Syscall para sair do programa (10)
```

2.4 Resultados

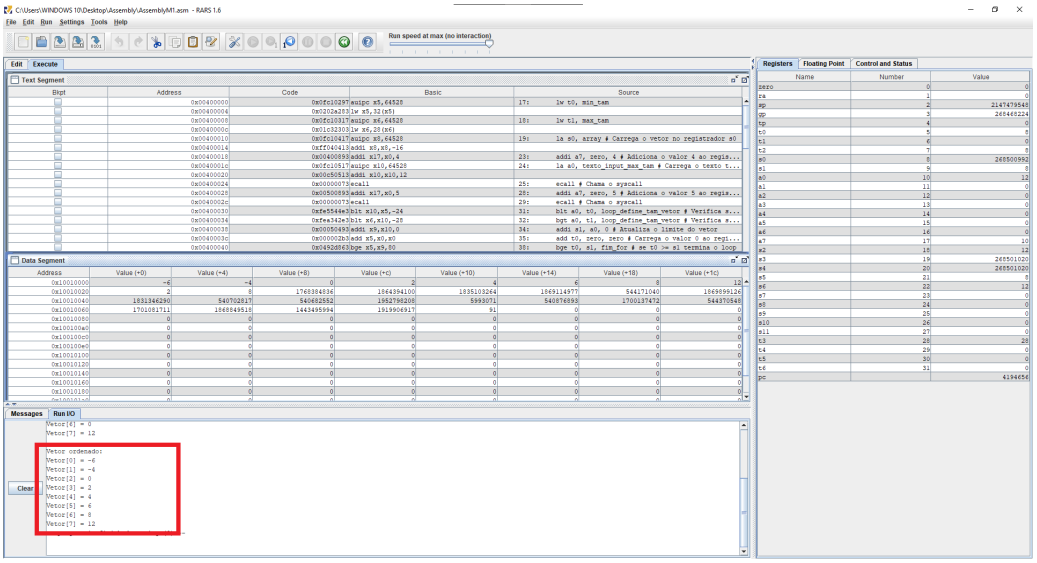
Informações da execução:
— Tamanho do vetor: 8
— Elementos do vetor: -4,-6, 8, 6, 4, 2, 0, 12
— Resultado final após ordenar o vetor: -6, -4, 0, 2, 4, 6, 8, 12

Abaixo estão as capturas de tela (Prints) da execução do programa inserindo os valores informados acima.

Abaixo consta o input do usuário.



Abaixo, consta o resultado do algoritmo de ordenação do vetor.



Abaixo, consta o Instruction Statistic do programa e seus resultados.