Laboratório 1 - Organização e arquitetura de computadores

Gabriel Rodrigues Diógenes, Higor Gabriel,
Marina Carvalho Soares de Queiroz, Luis Filipe Campos Cardoso

11 de fevereiro de 2020

Universidade de Brasília - Instituto de Ciências Exatas
Departamento de Ciência da Computação - CIC 116394 - Organização e Arquitetura de Computadores
2019.1 - Turma A - Professor Marcus Vinicius Lamar
Prédio CIC/EST - Campus Universitário Darcy Ribeiro
Asa Norte 70919-970 Brasília, DF

1 Introdução

O laboratório 1 teve como objetivo familiarizar os alunos com o Simulador/Montador Rars, permitir o desenvolvimento da capacidade de codificação de algoritmos em linguagem *Assembly* e desenvolver a capacidade de análise de desempenho de algoritmos em *Assembly*.

2 Simulador/Montador Rars

2.1 Item 1.1

O programa em *assembly* "sort.s"ocupa 276 bytes de código executável e na memória de dados "x"bytes armazenando os dados do vetor. Ao executar a rotina *sort*, a quantidade de instruções executadas foram "x", sendo "x"do tipo R, "x"do tipo I, "x"do tipo S e "x"do tipo U.

Para ordenção decrescente foi modificada a linha de comando de número 47 para: ble t4,t3,exit2. Usando a ferramenta instruction counter do RARS, os tipos de instruções e quantidades na função sort estão descritas nas tabela abaixo.

Type	Número de instruções
R-type	993
R4-type	0
I-type	1058
S-type	317
B-type	400
U-type	0
J-type	343

Quadro 1: Ordenação crescente

Type	Número de instruções
R-type	1595
R4-type	0
I-type	1660
S-type	519
B-type	600
U-type	0
J-type	545

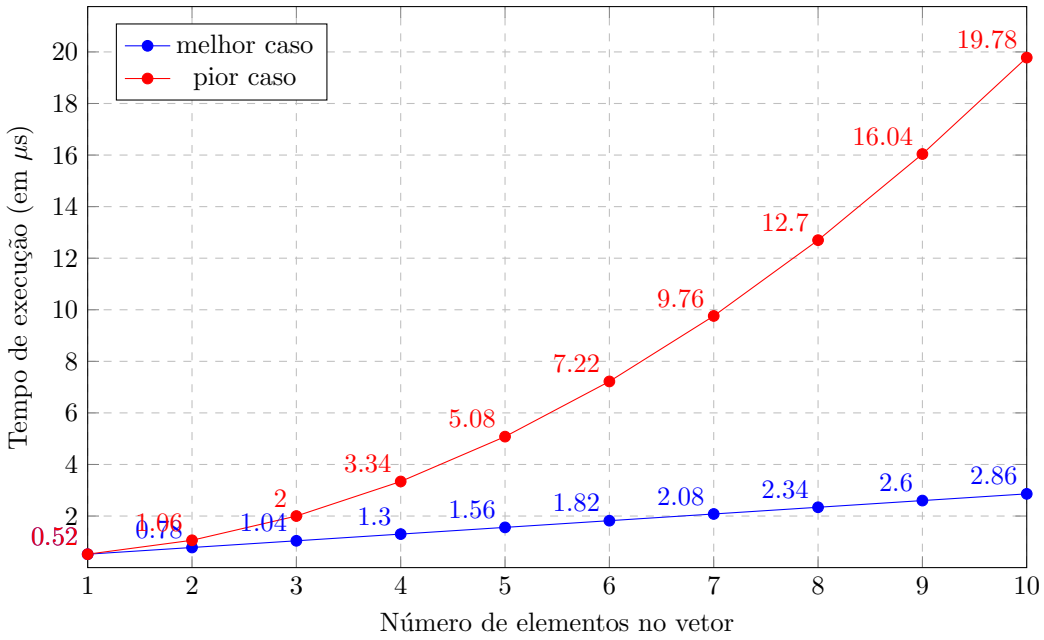
Quadro 2: Ordenação decrescente

O tamanho do executável em bytes 110 em hexadecimal mais 4 bytes em decimal, fazendo a transformação temos 272+4 = 276 bytes.

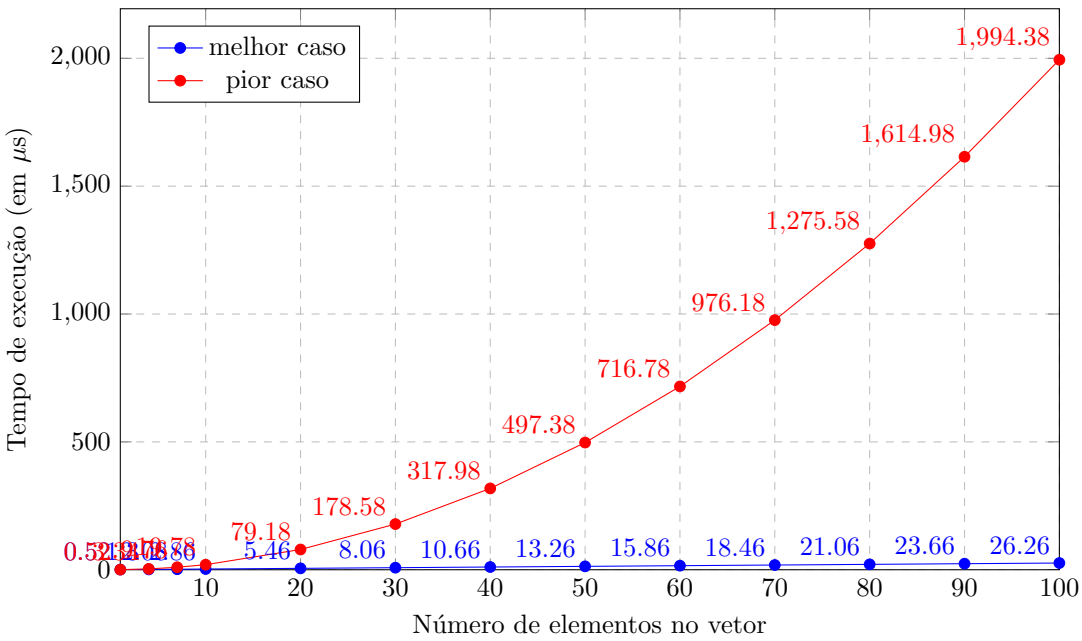
O tamanho da memória utilizada depende do tamanho do vetor utilizado, ou seja, do número de elementos e do tamanho máximo que a pilha do programa pode ter. Com o número de elementos igual a 30, já temos um tamanho de memória de 120 bytes, somados ao tamanho máximo da pilha da função swap de 60x4=240 bytes. No total, o tamanho da memória utilizada é de 360 bytes.

2.2 Item 1.2

Utilizando a equação $t_{exec} = I \cdot CPI \cdot T_{clk}$ e os dados disponíveis, obteve-se gráfico abaixo:



Comparação: algoritmo de ordenação (no arquivo sort.s)



Nota-se que no pior caso possível, no qual temos um vetor ordenado inversamente, há um crescimento no esforço computacional de modo exponencial conforme o número de elementos aumenta. Já nos casos tais que o vetor já esta ordenado, ele cresce linearmente

a uma baixa taxa.

3 Problema do Polígono Fechado

3.1 Item 3.1

SORTEIO: add s1 zero zero

SORTEIOFOR: addi a7 zero 42 //ecall que chama os rand li a1 319 definindo o teto li a0 1 ecall mv t0 a0 /// salvando o valor aleatorio em s2 addi a7 zero 42 ecall que chama os rand li a1 239 //definindo o teto li a0 1 ecall mv t1 a0 // salvando o valor aleatorio em s3

slli t0 t0 16 // movendo os 16 bits or t0 t0 t1 //fazendo um or pra juntar na word

slli t1 s1 2 //multiplicando por 4 s1, e salvando em t1 add t2 t1 s2 //colocando o valor de t1 em t0

bge s1 s4 SORTEIOMORREU //Verificando se s0 é maior que s1, se for ele morre sw t0 0(t2) //carregando na memoria s2 addi s1 s1 1 //add 1 no contador s1 j SORTEIOFOR //retorna o loop

SORTEIOMORREU: ret

3.2 Item 3.2

twi

3.3 Item 3.3

Foi criada a função polígono no arquivo 3.3.asm. Primeiramente, essa função recebe os dados do polígono (quantidade de lados, cor, eixos X e Y). Em seguida, carrega as rotinas do sistema Ecall. Após, carrega os dados do polígono nos registradores. Depois, é criado um loop. Ele é rodado enquanto o contador não chega no tamanho do polígono. Dentro do loop, são carregados os pares ordenados e a cor da linha nos registradores responsáveis pela renderização da linha. Em cada interação do loop, é chamado o serviço utilizado para renderizar linha (47) para mostrá-la em tela. Ao terminar o loop, o programa entra em uma instrução (FECHA) responsável por fechar completamente o polígono e renderizar a

última linha que falta. Por fim, o programa entra em estado de loop na linha 46 através da label FIM para que fique em espera e as linhas renderizadas permaneçam na tela.