G6 - Laboratório 1
Assembly - Risc V

Ana Luísa Padilha Alves, 20/2006546
Bruno Vargas de Souza, 20/2006564
Harisson Freitas Magalhães, 20/2006466

[1]Dep. Ciência da Computação – Universidade de Brasília (UnB)
CIC 116394 – Organização e Arquitetura de Computadores – Turma A

202006546@aluno.com.br, 202006564@aluno.unb.br, 202006466@aluno.unb.br

1. Simulador/Montador Rars

1.1. No diretório System, abra o Rars15Custom1 e carregue o programa de ordenamento sort.s. Dado o vetor: V[30]=9,2,5,1,8,2,4,3,6,7,10,2,32,54,2,12,6,3,1,78,54,23,1,54,2,65,3,6,55,31, ordená-lo em ordem crescente e contar o número de instruções por tipo e o número total exigido pelo procedimento sort. Qual o tamanho em bytes do código executável? E da memória de dados usada?

O código possui um tamanho de 276 Bytes. e a memória usada foi de 128 Bytes

1.2. Considere a execução deste algoritmo em um processador RISC-V com frequência de clock de 50MHz que necessita 1 ciclo de clock para a execução de cada instrução (CPI=1). Para os vetores de entrada de n elementos já ordenados Vo[n] = 1, 2, 3, 4, ...n e ordenados inversamente Vi[n] = n, n-1, n-2,..., 2, 1:

$$T_{exec} = Instruções * CPI * T$$
$$\text{CPI} = \frac{Ciclos_{clock}}{Inst} \quad \text{T} = \frac{Segundos}{Ciclos_{clock}}$$

a) Paro o procedimento sort, escreva as equações dos tempos de execução em função de n, to(n) e ti(n).

Para o vetor ordenado:

```
# Ordenado
# N - Instruções
# 1 - 23
# 2 - 33
# 3 - 43
# 10 - 113
```

Figure 1. Número de Instruções Ordenado

Foi possível notar um padrão envolvendo as saídas, assim foi obtida a seguite equação para o vetor ordenado:

$$T_0(n) = (30n + 13) * 1 * (50 * 10^{-6})$$

Para o vetor não-ordenado:

```
# Não ordenado
# N - Instruções
# 1 - 23
# 2 - 46
# 3 - 87
# 4 - 146
# 5 - 223
```

Figure 2. Número de Instruções Não-Ordenado

Sabendo que o Bubble Sort no pior caso gera uma complexidade O(n^2), teremos uma equação de segundo grau. Então, foi possível chegar na equação através de um sistema de matrizes utilizando o sistema linear feito na Figura 2, da qual foi o usado Método de Gauss para um padrão $ax^2 + bx + c$.

$$T_i(n) = (9n^2 - 4n + 18) * 1 * (50 * 10^{-6})$$

b) Para n=10,20,30,40,50,60,70,80,90,100, plote (em escala!) as duas curvas em um mesmo gráfico n×t. Comente os resultados obtidos

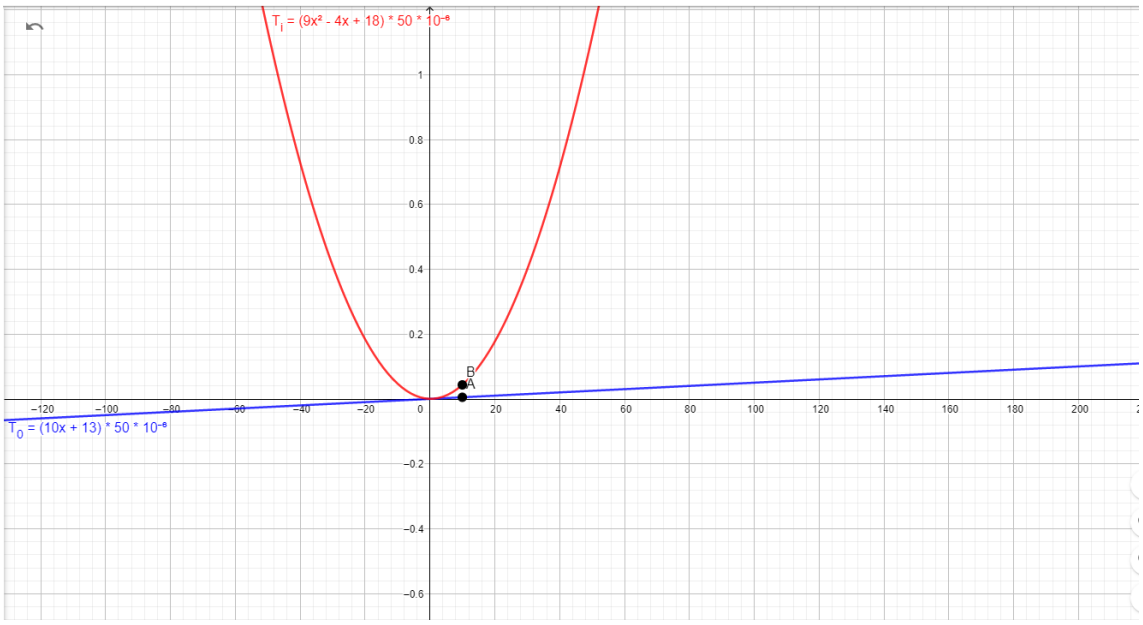

Figure 3. Curvas do tempo de execução

1.3. O programa foi executado.

2. Compilador cruzado GCC

2.1.

2.2. Dado o programa sortc.c, compile-o com a diretiva –O0 e obtenha o arquivo sortc.s. Indique as modificações necessárias no código Assembly gerado para que possa ser executado corretamente no Rars.
Dica: Uso de Assembly em um programa em C. Use a função show definida no sort.s para não precisar implementar a função printf, conforme mostrado no sortc-mod.c

Para executar o programa corretamente no Rars foi necessário:

- Acrescentar o .data e o .text nos lugares corretos
- Fazer um Jump para a label Main no início do programa
- Colocar uma label no fim para encerrar a execução do Rars

2.3. Compile o programa sortc-mod.c e, com a ajuda do Rars, monte uma tabela comparativa com o número total de instruções executadas pelo programa todo, e o tamanho em bytes dos códigos em linguagem de máquina gerados para cada diretiva de otimização da compilação -O0, -O1, -O2, -O3, -Os. Compare ainda com os resultados obtidos no item 1.1) com o programa sort.s que foi implementado diretamente em Assembly. Analise os resultados obtidos usando o mesmo vetor de entrada.

Como visto no 1.1 o programa possui 276 bytes. Conforme a figura a seguir, podemos comparar as otimizações:

Flag	Instruções	Bytes
O0	9788	528
O1	3888	372
O2	2176	280
O3	2175	280
Os	4999	348

Figure 4. Tabela de comparação dos resultados

Pode-se notar que o programa original escrito em Assembly acaba sendo mais ”otimizado”, já que necessitou de menos bytes na memória de instruções.

3. Senha (Mastermind)

3.1. Crie um programa no Rars que emule o jogo com máximo de 10 tentativas e N inicial igual a 5

O jogo foi criado com um N que vai até 19 cores.

3.2. Cada cor possui um efeito sonoro único quando colocada no tabuleiro. O preto possui uma pequena música e o branco outra pequena música.

O preto e branco possui uma pequena musiquinha para que não tome muito tempo de execução.

3.3. As cores são escolhidas através do teclado, escreva a codificação das teclas em cores na tela

Utilizamos as letras de A até T para codificar.

3.4. A cada vitória N=N+1, isto é, o número de cores é incrementado, aumentando a dificuldade do próximo nível.

O N vai incrementando na medida que vai aumentando, e também não é possível colocar uma letra que ainda não foi revelada para utilizar nas bolinhas.

3.5. Filme vc jogando até o máximo N que seu grupo conseguir. (Não vale usar cheat!!!)

Link do vídeo no youtube

3.6. Faça os gráficos N×texec e N×I, onde texec é o tempo de execução e I o número de instruções requeridas pelo seu algoritmo para a análise do pior caso da tentativa do jogador. Sabendo que o Rars simula uma CPU RISC-V com CPI=1, qual a frequência de clock desta CPU?

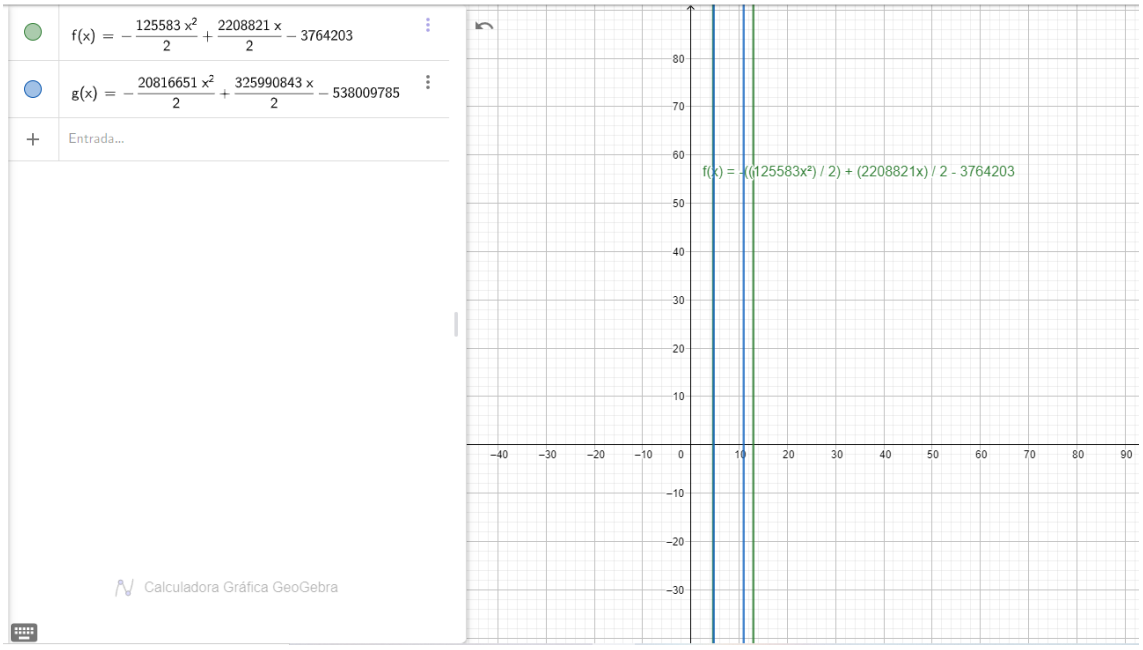

Figure 5. Curvas do Tempo de Execução e Instruções

Frequência = 89.11 MHz.

4. Conclusão

Infelizmente nosso jogo possui ainda pequenos erros que não conseguimos arrumar a tempo, que provavelmente tem a ver com a acesso à memória. aqui está um caso de erro, em que ele começa com o Vetor GABARITO-CORES correto, porém durante o programa, um dos elementos das cores não aparece mais.

```
CORES: .byte 5                          # Quantidade de Cores N
GABARITO_CORES: .byte 0, 0, 0, 0        # Gabarito Final de cada Fase
TENTATIVA_CORES: .byte 0, 0, 0, 0       # Tentativa do jogador
LINHA: .byte 1                          # Linha Atual do Jogador
COLUNA: .byte 1                         # Coluna Atual do Jogador
PINOS_QTD: .byte 0                      # Quantidade de Pinos, brancos ou pretos por fase
```

Figure 6. Vetor que está presente os dados

Figure 7. Pequeno erro no jogo

O erro mostrado pode confirmar que na primeira linha o azul deveria ser a última coluna, porém na linha 7 ele mostra que o azul não está no gabarito, e no final, ele não mostra a última coluna.