Laboratório 1
Assembly RISC-V

André Dornelas, 17/0099369
José Marcos Gois, 13/0143081

[1]Dep. Ciência da Computação – Universidade de Brasília (UnB)
CIC0099 - Organização e Arquitetura de Computadores

andredornelas23@gmail.com, goisinho_5@hotmail.com

1. Introdução

Inicialmente é apresentado a conceito de compilação cruzada, estudado através de exemplos simples na plataforma *Godbolt*, onde foram inseridos códigos em C e extraídos os códigos em Assembly com as limitações da ISA RV32I através de diferentes tipos de otimização. Então é criado um código em Assembly para solução de labirintos e estuda-se sua performance.

2. Objetivos

Familiarização com o Simulador/Montador Rars, desenvolvimento da capacidade de codificação de algoritmos em linguagem Assembly e também análise de desempenho de algoritmos em Assembly.

3. Compilador Cruzado GCC

2.2)

- Foi necessário incluir *.data* antes do vetor v para que o mesmo fosse lido como dado, assim como incluir *.text* antes das instruções.
- O programa principal *main* foi movido para o começo do código para ser executado primeiro.
- Como última instrução, foi chamada a função *ecall* com o registrador a7 carregado com o imediato 10 para devolver o controle ao sistema operacional para que o mesmo não terminasse com erros.

O código em Assembly pode ser visto no arquivo *sortc.s*

2.3)

Diretiva	-O0	-O1	-O2	-O3	-Os
Instruções	9786	3886	2174	2173	4097
Tamanho em bytes	524	368	276	276	346

Nota-se que quanto mais aumenta a otimização, menor são as instruções e menor o tamanho do programa em bytes, sendo que da diretiva -O2 para -O3 foi diminuída apenas uma instrução, tal diferença provavelmente se deve pelo fato de ser um programa simples. Já a diretiva -Os parece não ser tão otimizada, pois possui um número de instruções maior do que a -O1 e o tamanho do programa maior do que a da diretiva -O2.

4. Solução de Labirintos

Não foi possível concluir o procedimento *solve_maze*, portanto não é possível responder as perguntas desta seção.
Contudo, o código em assembly com os procedimentos *draw_maze* e *animate* pode ser visto no arquivo *q3.s*
Utilizou-se a lógica de recursão para tentar solucionar os labirintos, porém houve problemas na lógica que não houve como concluir.
Pode aferir que muitos dos conteúdos lecionados como recursão na busca de um novo caminho, utilização da memória como uma estrutura de dados, pilha.

draw_maze Primeiramente há a problemática de mostrar os labirintos de diversos tamanhos que foi uma parcela concluída com sucesso, todos centralizados. Utilizando a lógica de encontrar o ponto central e deste ponto em diante calcular de acordo com os parâmetros dados.

Desempenho dos Algoritmos Como não foi possível concluir o algorítimo principal proposto no laboratório, para responder alguma questão do relatório, fizemos as métricas pedidas em cima das funções bem sucedidas. Como a draw maze. Instruções para o draw maze: 172752 Tempo de execução para o draw maze: 1807541418 Link para Vídeo: da execução parcelada. O algorítimo para solução apesar de parcelado, está apenas com a falha no retorno ao entrar em um caminho sem saída. Porém bem completo quanto a lógica.