Trabalho Prático 1 - Assembly

Luiz Felipe Gondim Araujo - 2023028188

Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte - MG - Brasil

luizfelipegondimaraujo@ufmg.br

1 Introdução

Este trabalho teve como objetivo desenvolver rotinas em linguagem Assembly RISC-V para manipular vetores e realizar operações matemáticas simples. Sabendo disso, foram implementadas funcionalidades como a verificação se os elementos de um vetor estão em ordem crescente, e o ajuste de preços de produtos baseado em um multiplicador e uma taxa informados no sistema. As implementações foram organizadas em funções específicas, com testes individuais para verificar a correção dos resultados esperados.

2 Decisões de implementação

Durante a implementação, algumas decisões foram tomadas visando a clareza e modularidade do código. Dessa forma, foi possível adotar as seguintes abordagens:

- A função e_crescente foi construída comparando pares de elementos adjacentes em um vetor para determinar se estavam em ordem crescente.
- Já para o ajuste de preços, optou-se por seguir a sugestão do professor e dividir o processo em duas funções: ajustar_preco, responsável pela multiplicação do preço pelo fator, e aplicar_taxa, responsável pela aplicação de uma taxa adicional.
- Para obter o valor final de todos os preços ajustados, foi criada a função soma_estoque, que percorre o vetor e acumula os resultados em um registrador somador.

3 Dificuldades enfrentadas

Em relação às dificuldades encontradas, a principal delas foi a manipulação de ponteiros e offsets para acessar corretamente os elementos dos vetores. Além disso, o gerenciamento dos registradores exigiu atenção, sendo necessário salvar e restaurar valores na pilha para garantir o funcionamento correto entre chamadas de função. A depuração também foi um desafio, já que a linguagem Assembly não fornece mensagens de erro detalhadas, o que dificultou a localização de falhas lógicas no código.

4 Testes realizados

Foram realizados diversos testes para validar as funcionalidades. Vetores com diferentes configurações e valores (positivos, nulos e negativos) foram utilizados para verificar a função e_crescente, abrangendo casos em que o vetor estava ordenado corretamente e outros em que havia quebras na ordem.

Ademais, a função soma_estoque foi testada com valores de entrada conhecidos, para garantir que os resultados multiplicados estivessem corretos. Ao final dos testes, os resultados eram armazenados em registradores e comparados com os valores esperados, contabilizando o número de testes bem-sucedidos. Também foram utilizados valores nulos e negativos para os preços dos produtos e parametros modificadores (fator e taxa), o que permitiu verificar se as instruções de multiplicação estavam funcionando corretamente.

5 Aprendizados adquiridos

Como principais aprendizados, destaca-se a compreensão prática dos fundamentos da arquitetura RISC-V, especialmente no que diz respeito à manipulação de registradores, uso da pilha e chamadas de função. Além disso, a experiência reforçou a importância de manter a documentação do código clara, especialmente em uma linguagem de baixo nível, onde a interpretação de erros é mais trabalhosa.