Relatório para o trabalho III

Trabalho Programação Assembler

Aluno: Cristiano Tolentino Santos

Matrícula: 21102850

Professor: Ricardo Pezzuol Jacobi

Turma: T02 (2024.2 - 35T23)

Objetivo:

Este trabalho tem como objetivo a prática da programação em assembly na arquitetura RISCV, por meio do desenvolvimento de funções que executam operações sobre matrizes. As funções a serem implementadas incluem soma, multiplicação, transposição e exibição da matriz.

Pseudo código

O código implementa cada operação como uma função separada, as quais atuam nas matrizes por meio de duas funções bases (**read_cell** e **write_cell**), em seu funcionamento, a0 e a1 permanecem reservados para indicarem as matrizes a serem operadas, a2 reservado para o endereço da matriz resultado e a3 deve indicar o tamanho da lateral das matrizes analisadas.

A função **read_cell** realiza a leitura de um elemento específico de uma matriz. A função recebe a linha e a coluna desejadas, e o endereço do primeiro elemento da matriz a ser lida. A linha e a coluna são ajustadas subtraindo 1 de seus valores, já que a contagem das posições começa do índice 0. Em seguida, o deslocamento a partir de 0 é calculado multiplicando o número de linhas pelo tamanho do lado, e somando o número de colunas. O endereço do elemento desejado é obtido somando esse deslocamento ao endereço base da matriz. O valor da célula é então lido da memória e armazenado em a0.

A função **write_cell** escreve um valor em uma célula específica de uma matriz. O funcionamento para encontrar a posição a ser alterada é idêntico ao funcionamento de **read_cell**. Uma vez que o endereço da célula é obtido e, o valor fornecido sobrescreve o valor armazenado naquela posição de memória.

A função **sum** realiza a soma de duas matrizes. Funciona recebendo os endereços das duas matrizes de entrada e o da matriz resultado. A função percorre todas as linhas e

colunas da matriz resultado, lendo a mesma posição das matrizes sendo operadas em cada iteração. Para cada elemento, chama-se **read_cell** para cada matriz sendo somada. A soma é realizada e o resultado é armazenado na matriz resultado com **write_cell**. O processo continua até que todos os elementos da matriz resultado sejam preenchidos. O controle dos loops é feito com instruções de salto condicional que verificam quando todas as linhas e colunas foram processadas.

A função **multiply** realiza a multiplicação de duas matrizes. Seu processo se inicia recebendo os endereços das duas matrizes de entrada e o da matriz resultado. Em seguida, são estabelecidos dois loops: um para percorrer as posições da matriz resultado e outro para percorrer os fatores do somatório:

$$r_{ij} = \sum_{n=1}^{L} a_{in} \cdot b_{nj}.$$

Onde *"r"* representa uma posição da matriz resultado, *"a"* elementos da primeira matriz e *"b"* elementos da segunda, ao final das iterações do loop interno o elemento da matriz resultado é definido e o loop externo prossegue para a próxima posição.

A função **transpose** realiza a transposição de uma matriz. Opera recebendo o endereço da matriz original e o da matriz resultado. A função percorre os elementos da matriz original e inverte seus índices de linha e coluna. O valor é lido da matriz original com a função **read_cell** e armazenado na nova posição de índices invertidos da matriz transposta com a função **write_cell**. O processo continua até que todas as células da matriz original tenham sido processadas e a matriz transposta esteja completamente preenchida.

A função **print** exibe a matriz na tela. Recebe o endereço da matriz a ser exibida e percorre todos os elementos da matriz, utilizando o comando **read_cell** para ler cada valor da célula e fornecer o valor a ser impresso. Quando uma linha é completada, é inserida uma quebra de linha para separar as linhas da matriz.