

**PUC-Rio – Departamento de Informática**  
**Ciência da Computação**  
**Introdução à Arquitetura de Computadores**  
**Prof.: Alexandre Meslin**



## **Trabalho 3 – 2022.1**

### **Parte I:**

Aprimorar o módulo escrito em linguagem C, chamado *matrix\_lib.c*, implementado no trabalho 2, com a utilização de **instruções vetoriais (AVX/FMA)**, usando a biblioteca Intel Intrinsics, e **processamento paralelo**, usando a biblioteca pthread. As duas funções de operações aritméticas com matrizes estão descritas abaixo.

- a. Função *int scalar\_matrix\_mult(float scalar\_value, struct matrix \*matrix)*

Essa função recebe um valor escalar e uma matriz como argumentos de entrada e calcula o produto do valor escalar pela matriz utilizando instruções vetoriais (AVX). *Cada thread deve calcular o resultado do produto entre o valor escalar e uma das linhas da matriz.* O resultado da operação deve ser retornado na matriz de entrada. Em caso de sucesso, a função deve retornar o valor 1. Em caso de erro, a função deve retornar 0.

- b. Função *int matrix\_matrix\_mult(struct matrix \*matrixA, struct matrix \*matrixB, struct matrix \*matrixC)*

Essa função recebe 3 matrizes como argumentos de entrada e calcula o valor do produto da matriz A pela matriz B utilizando instruções vetoriais (AVX/FMA). O **algoritmo otimizado de produto de matrizes** apresentado em aula deve ser utilizado nesta função. *Cada thread deve calcular o resultado de N linhas da matriz C, onde N é o número de linhas da matriz C dividido pelo número de threads disparadas em paralelo.* O resultado da operação deve ser retornado na matriz C. Em caso de sucesso, a função deve retornar o valor 1. Em caso de erro, a função deve retornar 0.

- c. Função *void set\_number\_threads(int num\_threads)*

Essa função recebe o número de threads que devem ser disparadas em paralelo durante o processamento das operações aritméticas com as matrizes e deve ser chamada pelo programa principal antes das outras funções. Caso não seja chamada, o valor default do número de threads do módulo é 1.

O tipo estruturado matrix é definido da seguinte forma:

```
struct matrix {  
    unsigned long int height;  
    unsigned long int width;  
    float *rows;  
};
```

Onde:

height = número de linhas da matriz (múltiplo de 8)  
width = número de colunas da matriz (múltiplo de 8)  
rows = sequência de linhas da matriz (height\*width elementos)

## Parte II:

Crie um programa em linguagem C, chamado *matrix\_lib\_test.c*, que implemente um código para testar a biblioteca *matrix\_lib.c*. Esse programa deve receber um valor escalar float, a dimensão da primeira matriz (A), a dimensão da segunda matriz (B), **o número de threads a serem disparadas**, e o nome de quatro arquivos binários de floats na linha de comando de execução. O programa deve inicializar as duas matrizes (A e B) respectivamente a partir dos dois primeiros arquivos binários de floats e uma terceira matriz (C) com zeros. **A inicialização da matriz C deve utilizar instruções vetoriais (AVX)**. A função *set\_number\_threads* deve ser chamada com o número de threads a serem disparadas. A função *scalar\_matrix\_mult* deve ser chamada com os seguintes argumentos: o valor escalar fornecido e a primeira matriz (A). O resultado (retornado na matriz A) deve ser armazenado em um arquivo binário usando o nome do terceiro arquivo de floats. Depois, a função *matrix\_matrix\_mult* deve ser chamada com os seguintes argumentos: a matriz A resultante da função *scalar\_matrix\_mult*, a segunda matriz (B) e a terceira matriz (C). O resultado (retornado na matriz C) deve ser armazenado com o nome do quarto arquivo de floats.

Exemplo de linha de comando:

```
matrix_lib_test 5.0 8 16 16 8 4 floats_256_2.0f.dat floats_256_5.0f.dat result1.dat result2.dat
```

Onde,

5.0 é o valor escalar que multiplicará a primeira matriz;

8 é o número de linhas da primeira matriz;

16 é o número de colunas da primeira matriz;

16 é o número de linhas da segunda matriz;

8 é o número de colunas da segunda matriz;

**4 é o número de threads a serem disparadas;**

*floats\_256\_2.0f.dat* é o nome do arquivo de floats que será usado para carregar a primeira matriz;

*floats\_256\_5.0f.dat* é o nome do arquivo de floats que será usado para carregar a segunda matriz;

*result1.dat* é o nome do arquivo de floats onde o primeiro resultado será armazenado;

*result2.dat* é o nome do arquivo de floats onde o segundo resultado será armazenado.

O programa principal deve cronometrar o tempo de execução geral do programa (overall time) e o tempo de execução das funções *scalar\_matrix\_mult* e *matrix\_matrix\_mult*. Para marcar o início e o final do tempo em cada uma das situações, deve-se usar a função padrão *gettimeofday* disponível em *<sys/time.h>*. Essa função trabalha com a estrutura de dados *struct timeval* definida em *<sys/time.h>*. Para calcular a diferença de tempo (delta) entre duas marcas de tempo *t0* e *t1*, deve-se usar a função *timedifference\_msec*, implementada no módulo *timer.c*, fornecido no roteiro do trabalho 1.

## Observação 1:

O programa deve ser desenvolvido em linguagem C e com a biblioteca Intel Intrinsics e a biblioteca pthread. A compilação do programa fonte deve ser realizada com o compilador GCC, usando os seguintes argumentos:

```
gcc -Wall -std=c11 -pthread -mfma -o matrix_lib_test matrix_lib_test.c matrix_lib.c timer.c
```

Onde,

*matrix\_lib\_test* = nome do programa executável.

*matrix\_lib\_test.c* = nome do programa fonte que tem a função main().

*matrix\_lib.c* = nome do programa fonte do módulo de funções de matrizes.

*timer.c* = nome do programa fonte do módulo do cronômetro.

O servidor do DI está disponível para acesso remoto, conforme informado anteriormente, e pode ser usado para executar o programa de teste.

**Observação 2:**

O programa deve inicialmente ser testado com matrizes pequenas para facilitar a depuração, mas, a versão final deve ser testada com matrizes grandes, com dimensão 1024 x 1024.

**Observação 3:**

O número de linhas da matriz C deve ser múltiplo do número de threads que devem ser disparadas e o número de colunas da matriz C deve ser múltiplo de 8.

**Observação 4:**

Apenas os programas fontes *matrix\_lib.c*, *matrix\_lib.h* e *matrix\_lib\_test.c* e a saída do programa com os tempos de execução das funções matriciais e tempo total devem ser carregados no site de EAD da disciplina até o prazo de entrega. **Somente \*\*UM\*\* integrante do grupo deve fazer a carga.**

**Prazo de entrega: 04/05/2022**