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

Trabalho 1 – 2023.1

Parte I:
Aprimorar o módulo escrito em linguagem C, chamado *equation.c*, implementado em sala de aula, com a utilização de **instruções vetoriais (AVX/FMA)** usando a biblioteca Intel Intrinsics.

Considere um sistema de equações do tipo:

$$a_{0,0}x_0 + a_{0,1}x_1 + a_{0,2}x_2 + \ldots + a_{0,n-1}x_{n-1} = b_0$$
$$a_{1,0}x_0 + a_{1,1}x_1 + a_{1,2}x_2 + \ldots + a_{1,n-1}x_{n-1} = b_1$$
$$a_{2,0}x_0 + a_{2,1}x_1 + a_{2,2}x_2 + \ldots + a_{2,n-1}x_{n-1} = b_2$$
$$\ldots$$
$$a_{n-1,0}x_0 + a_{n-1,1}x_1 + a_{n-1,2}x_2 + \ldots + a_{n-1,n-1}x_{n-1} = b_{n-1}$$

Este sistema pode ser também descrito de forma matricial:

$$Ax = B$$

Onde a matriz *A* representa os coeficientes da equação que multiplicam o vetor *x* como mostrado a seguir:

$$A = \begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} & \ldots & a_{0,n-1} \\ a_{1,0} & a_{1,1} & a_{1,2} & \ldots & a_{1,n-1} \\ a_{2,0} & a_{2,1} & a_{2,2} & \ldots & a_{2,n-1} \\ & & \ldots & & \\ a_{n-1,0} & a_{n-1,1} & a_{n-1,2} & \ldots & a_{n-1,n-1} \end{bmatrix}$$

Os vetores *x* e *B* podem ser escritos da seguinte forma:

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \ldots \\ x_{n-1} \end{bmatrix}$$

$$b = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \ldots \\ b_{n-1} \end{bmatrix}$$

Considerando que as *n* equações são linearmente independentes, o sistema possui apenas uma resposta para os valores do vetor x e operações de soma entre linhas e multiplicações dentro de uma linha não modificam a resposta do sistema. Por exemplo, podemos multiplicar a linha *i* por um fator *k* e usar essa nova equação no lugar da equação da linha *i*. Outra operação permitida é somar a linha *i* com a linha *j* e substituir a linha *i* (ou a linha *j*) pelo resultado da operação.

Implementar a função *processaVetores* que transforma a matriz *A* e o vetor *B*. Essa função deverá zerar todos os valores abaixo da diagonal principal, mantendo integro o sistema de equações representada por eles. Essa função deverá utilizar instruções vetoriais (AVX). O uso dessas instruções será o foco desse trabalho.

A função deverá receber um ponteiro para a matriz *A*, para o vetor *B* e um inteiro representando o número de incógnitas do sistema de acordo com o protótipo a seguir.

```
void processaVetores(double *mA, double *vB, int nIncognitas);
```

A função deverá modificar a matriz *A* e o vetor *B* para zerar toda a matriz A abaixo da diagonal principal.

Parte II:
Crie um programa em linguagem C, chamado *equation*, que implemente um código para testar a biblioteca *equation.c*. Esse programa deve receber:

a) o nome do arquivo binário com a matriz A (-m <arquivo>);
b) o nome do arquivo binário com o vetor B (-v <arquivo>);
c) a quantidade de incógnitas (-n <quantidade de incógnitas>
d) opcionalmente, o nome do arquivo de saída para contabilizar o tempo (-o <arquivo>).

O arquivo *main.cpp*, *comum.cpp* e *timer.cpp* implementam essas funcionalidades e realizam a chamada correta para a função *processaVetores*. Esses arquivos utilizam os headers comum.h, timer.h e gpu.h.

Exemplo de linha de comando:

equation -m matrizA.bin -n 32 -o tempos.txt -v vetorB.bin

Onde:

- 32 é o número de incógnitas do sistema;
- matrizA.bin é o nome do arquivo com 32x32 doubles representando a matriz *A*;
- vetorB.bin é o nome do arquivo com 32 doubles representando o vetor *B*.

O programa principal deve cronometrar o tempo de execução da função *processaVetores*. 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.cpp*, fornecido.

Observação 1:

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

```
gcc -Wall -mavx -march=native -o equation main.cpp equation.cpp comum.cpp
timer.cpp
```

Onde:

- *equation* = nome do programa executável;
- *main.cpp* = nome do programa fonte que tem a função main();

- *comum.cpp = nome do módulo com funções auxiliares;*
- *equation.cpp* = nome do programa fonte do módulo com a sua resposta;
- *timer.cpp* = 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 sistemas pequenos para facilitar a depuração, mas, a versão final deve ser testada com sistemas grandes, com dimensão 1024 x 1024 ou superior.

Observação 3:

Apenas o módulo fontes *equation.cpp* deve ser carregado no site de EAD da disciplina até o prazo de entrega. **Somente **UM** integrante do grupo deve fazer a carga.**

Prazo de entrega: veja plataforma EaD