

PUC-Rio – Departamento de Informática
Curso de Ciência da Computação
INF1029 – Introdução à Arquitetura de Computadores
Prof.: Anderson Oliveira da Silva



Programa base dos trabalhos de INF1029

Na disciplina de INF1029, vamos aprender a utilizar vários recursos de hardware e software para aumentar o desempenho (aceleração) dos nossos programas. Para melhor visualizarmos o impacto desses novos recursos, criaremos um programa base principal que utilizará uma biblioteca de funções para realizar dois cálculos: (i) *produto de valor escalar por matriz*; e (ii) *produto de duas matrizes*. Nosso objetivo é aplicar os novos recursos computacionais para melhorar o desempenho dessas duas funções e modificar o programa base principal minimamente.

Parte I:

Crie um módulo escrito em linguagem C, chamado *matrix_lib.c*, que implemente duas funções para fazer operações aritméticas com matrizes, conforme descrito 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. 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. 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.

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) e os nomes 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. A terceira matriz (C) deve ser inicializada com zeros. A função *scalar_matrix_mult* deve ser chamada com os seguintes argumentos: o valor escalar fornecido e a primeira matriz (A). O resultado do produto (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 do produto (retornado na matriz C) deve ser armazenado em um **arquivo binário** com o nome do quarto arquivo de floats.

Exemplo de linha de comando:

```
matrix_lib_test 5.0 8 16 16 8 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;

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 base 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 abaixo:

Listagem do arquivo fonte: timer.h

```
#include <sys/time.h>

float timedifference_msec(struct timeval t0, struct timeval t1);
```

Listagem do arquivo fonte: timer.c

```
#include "timer.h"

/*
 * float timedifference_msec(struct timeval t0, struct timeval t1)
 *
 * Recebe uma marca de tempo t0 e outra marca de tempo t1 (ambas do
 * tipo struct timeval) e retorna a diferenca de tempo (delta) entre
 * t1 e t0 em milisegundos (tipo float).
 */
float timedifference_msec(struct timeval t0, struct timeval t1)
{
    return (t1.tv_sec - t0.tv_sec) * 1000.0f + (t1.tv_usec - t0.tv_usec) / 1000.0f;
}
```

Exemplo de uso do módulo timer.c: timer_test.c

```
#include <stdio.h>
#include <stdlib.h>
#include "timer.h"

int main(int argc, char *argv[]) {
    struct timeval start, stop, overall_t1, overall_t2;
    unsigned long int i, VECTOR_SIZE, *vector;
    char *eptr = NULL;

    // Mark overall start time
    gettimeofday(&overall_t1, NULL);

    if (argc != 2) {
        printf("Usage: %s <vector_size>\n", argv[0]);
        return 0;
    }

    // Convert arguments
    VECTOR_SIZE = strtol(argv[1], &eptr, 10);

    // Allocate vector
    vector = (unsigned long int*)malloc(VECTOR_SIZE*sizeof(unsigned long int));

    if (vector == NULL) {
        printf("%s: vector allocation problem.", argv[0]);
        return 1;
    }

    // Mark init start time
    gettimeofday(&start, NULL);

    // Initialize vector
    for (i = 0; i < VECTOR_SIZE; ++i)
        vector[i] = i;

    // Mark init stop time
    gettimeofday(&stop, NULL);

    // Show init exec time
    printf("Init time: %f ms\n", timedifference_msec(start, stop));

    // Mark reorder start time
    gettimeofday(&start, NULL);

    // Reorder vector
    for (i = 0; i < VECTOR_SIZE; ++i)
        vector[i] = VECTOR_SIZE - 1 - i;

    // Mark reorder stop time
    gettimeofday(&stop, NULL);

    // Show reorder exec time
    printf("Reorder time: %f ms\n", timedifference_msec(start, stop));

    // Free vector
    free(vector);

    // Mark overall stop time
```

```

gettimeofday(&overall_t2, NULL);

// Show elapsed overall time
printf("Overall time: %f ms\n", timedifference_msec(overall_t1, overall_t2));

return 0;
}

```

Linha de comando para compilar o programa *timer_test.c*:

```
gcc -o timer_test timer_test.c timer.c
```

Onde,

timer_test = nome do programa executável.

timer_test.c = nome do programa fonte de teste do módulo do cronômetro.

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

Linha de comando para testar o programa *timer_test.c*:

```
timer_test 1024000000
```

Observação 1:

Todas as práticas da disciplina serão executadas no ambiente Linux. Como o objetivo é usar esse programa base nos próximos trabalhos desta disciplina, este programa deve ser compilado e executado no ambiente Linux. Para fazer a compilação do programa no Linux, pode-se usar o GCC, com os seguintes argumentos:

```
gcc -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.

Uma máquina virtual padrão VMware com o sistema Linux está disponível na área de download do site da Equipe de Suporte do DI, em:

URL: <http://suporte.inf.puc-rio.br/download/vms/VMCCPP-FC23-64-DI-PUC-Rio-V1.0.zip>

Para executar a máquina virtual, basta baixar o VMware Workstation Player gratuitamente a partir do site da VMware.

URL: <https://www.vmware.com/br/products/workstation-player/workstation-player-evaluation.html>

Observação 2:

Os arquivos de entrada usados para popular as matrizes e o arquivo de saída com o resultado do programa estão no **formato binário com floats**, ou seja, não podem ser manipulados com *fprintf* e *fscanf*, mas, sim, com *fread* e *fwrite*.

Crie um programa simples para gerar os arquivos de entrada com a quantidade necessária de floats para popular as matrizes, ou seja, com $M \times N$ floats onde $M \times N$ é a dimensão de uma matriz.

Observação 3:

Os nomes dos programas fontes `matrix_lib.c`, `matrix_lib.h` e `matrix_lib_test.c` não podem ser alterados pelo grupo e não devem ser submetidos no EAD pois não serão corrigidos.

Prazo de preparação: 1/9/2025 - 23:59h.