**CUDA:**

#include "cuda\_runtime.h"

#include "device\_launch\_parameters.h"

#include <iostream>

#include <vector>

/\* --------------------------------------------------------------------

Implementati un program CUDA ce efectueaza adunarea a doua matrici patratice

cu N linii si N coloane

Urmati, pe rand, pasii de mai jos.

--------------------------------------------------------------------\*/

\_\_global\_\_ void matrix\_matrix\_add(float \*a, float \*b, float \*c, int N);

int main()

{

// Numarul N de linii/coloane

int N = 1024;

// Prima matricea

float \*a\_h;

// A 2-a matrice

float \*b\_h;

// Matricea rezultat

float \*c\_h;

/\* --------------------------------------------------------------------

1. Alocati memorie pe host pentru cele trei matrici:

\* Matricea este de N linii si N coloane si este alocată ca un singur

bloc de memorie (un o singura alocare de N\*N elemente)

0.7142 puncte

----------------------------------------------------------------------\*/

/\* ------------------------ REZOLVARE PCT 1 -----------------------------\*/

/\* ------------------------ CODUL TAU DE AICI ---------------------------\*/

a\_h = new float[N\*N];

b\_h = new float[N\*N];

c\_h = new float[N\*N];

/\* ------------------------ CODUL TAU PANA AICI ---------------------------\*/

/\* --------------------------------------------------------------------

2. Initializati cele doua matrici a si b după cum urmeaza:

\* Prima matrice va fi initializata cu zero peste tot exceptand diagonala

principala unde elementele au valoarea 5

\* A 2-a matrice va fi initializată cu 1 peste tot exceptand diagonala

secundara unde elementele au valoarea 3

0.7142 puncte

----------------------------------------------------------------------\*/

/\* ------------------------ REZOLVARE PCT 2 -----------------------------\*/

/\* ------------------------ CODUL TAU DE AICI ---------------------------\*/

for (int i = 0; i < N; i++)

{

for (int j = 0; j < N; j++)

{

if (i == j)

{

a\_h[i\*N + j] = 5;

}

else

{

a\_h[i\*N + j] = 0;

}

}

}

for (int i = 0; i < N; i++)

{

for (int j = 0; j < N; j++)

{

if (j == (N-1) - i)

{

b\_h[i\*N + j] = 3;

}

else

{

b\_h[i\*N + j] = 1;

}

}

}

/\* ------------------------ CODUL TAU PANA AICI ---------------------------\*/

// Cele trei matrici pentru memoria device

float \*a\_d;

float \*b\_d;

float \*c\_d;

/\* --------------------------------------------------------------------

3. Alocati memorie pe device pentru cele trei matrici

0.7142 puncte

----------------------------------------------------------------------\*/

/\* ------------------------ REZOLVARE PCT 3 -----------------------------\*/

/\* ------------------------ CODUL TAU DE AICI ---------------------------\*/

cudaMalloc((void\*\*)&a\_d, N\*N \* sizeof(float));

cudaMalloc((void\*\*)&b\_d, N\*N \* sizeof(float));

cudaMalloc((void\*\*)&c\_d, N\*N \* sizeof(float));

/\* ------------------------ CODUL TAU PANA AICI ---------------------------\*/

/\* --------------------------------------------------------------------

4. Copiati continutul celor doua matrici a si b de pe host pe device

0.7142 puncte

----------------------------------------------------------------------\*/

/\* ------------------------ REZOLVARE PCT 4 -----------------------------\*/

/\* ------------------------ CODUL TAU DE AICI ---------------------------\*/

cudaMemcpy(a\_d, a\_h, N\*N \* sizeof(float), cudaMemcpyHostToDevice);

cudaMemcpy(b\_d, b\_h, N\*N \* sizeof(float), cudaMemcpyHostToDevice);

/\* ------------------------ CODUL TAU PANA AICI ---------------------------\*/

// Numarul de thread-uri pe bloc

dim3 threadsPerBlock;

threadsPerBlock.x = 32;

threadsPerBlock.y = 32;

threadsPerBlock.z = 1;

// Numarul de blocuri

dim3 numBlocks;

/\* --------------------------------------------------------------------

5. Calculati numarul de blocuri astfel incat:

\* se obtine un grid 2D de thread-uri

\* Se va lansa in executie cate un thread pentru fiecare element din matrice

deci NxN in total

0.7142 puncte

----------------------------------------------------------------------\*/

/\* ------------------------ REZOLVARE PCT 5 -----------------------------\*/

/\* ------------------------ CODUL TAU DE AICI ---------------------------\*/

numBlocks.x = N / threadsPerBlock.x;

numBlocks.y = N / threadsPerBlock.y;

numBlocks.z = 1;

/\* ------------------------ CODUL TAU PANA AICI ---------------------------\*/

// Se lanseaza in executie kernel-ul cuda

matrix\_matrix\_add << <numBlocks, threadsPerBlock >> > (a\_d, b\_d, c\_d, N);

/\* --------------------------------------------------------------------

6. Copiati continutul matricei rezultat c de pe device pe host

0.7142 puncte

----------------------------------------------------------------------\*/

/\* ------------------------ REZOLVARE PCT 6 -----------------------------\*/

/\* ------------------------ CODUL TAU DE AICI ---------------------------\*/

cudaMemcpy(c\_h, c\_d, N\*N \* sizeof(float), cudaMemcpyDeviceToHost);

/\* ------------------------ CODUL TAU PANA AICI ---------------------------\*/

return 0;

}

\_\_global\_\_ void matrix\_matrix\_add(float \*a, float \*b, float \*c, int N)

{

// Indicele fiecarui thread

int index;

/\* --------------------------------------------------------------------

7. Calculati valoarea celor doi indici i si j utilizand variabilele

predefinite blockDim, blocIdx si threadIdx astfel incat fiecare thread

sa corespunda unei pozitii din matrice

0.7142 puncte

----------------------------------------------------------------------\*/

/\* ------------------------ REZOLVARE PCT 7 -----------------------------\*/

/\* ------------------------ CODUL TAU DE AICI ---------------------------\*/

int i = blockDim.x\*blockIdx.x + threadIdx.x;

int j = blockDim.y\*blockIdx.y + threadIdx.y;

/\* ------------------------ CODUL TAU PANA AICI ---------------------------\*/

/\* --------------------------------------------------------------------

8. efectuati adunarea matricilor a si b si scrieti rezultatul in c (c = a+b)

2 puncte

----------------------------------------------------------------------\*/

/\* ------------------------ REZOLVARE PCT 8 -----------------------------\*/

/\* ------------------------ CODUL TAU DE AICI ---------------------------\*/

c[i\*N + j] = a[i\*N + j] + b[i\*N + j];

/\* ------------------------ CODUL TAU PANA AICI ---------------------------\*/

}

**OpenMP:**

#include "pch.h"

#include <iostream>

#include <omp.h>

/\* --------------------------------------------------------------------

Implementați un program ce calculează produsul a 2 matrici

cu N linii și N coloane iar apoi paralelizați programul utilizand

OpenMP

Urmati, pe rand, pasii de mai jos.

--------------------------------------------------------------------\*/

int main()

{

// Numarul N de linii/coloane

int N = 1000;

// Prima matrice

int \*a;

// A 2-a matrice

int \*b;

// Matricea rezultat

int \*c;

/\* --------------------------------------------------------------------

1. Alocati memorie pe host pentru cele trei matrici:

0.333 puncte

--------------------------------------------------------------------\*/

/\* ------------------------ REZOLVARE PCT 1 -----------------------------\*/

/\* ------------------------ CODUL TAU DE AICI ---------------------------\*/

a = new int[N\*N];

b = new int[N\*N];

c = new int[N\*N];

/\* ------------------------ CODUL TAU PANA AICI ---------------------------\*/

/\* --------------------------------------------------------------------

2. Initializați matricile a si b dupa cum urmeaza: elementele de pe prima coloana

au valoarea 1, elementele de pe a 2-a coloana au valoarea 2, cele de pe a

3-a au valoarea 4, etc

0.333 puncte

--------------------------------------------------------------------\*/

/\* ------------------------ REZOLVARE PCT 2 -----------------------------\*/

/\* ------------------------ CODUL TAU DE AICI ---------------------------\*/

for (int i = 0; i < N; i++)

{

for (int j = 0; j < N; j++)

{

a[i\*N + j] = j+1;

}

}

for (int i = 0; i < N; i++)

{

for (int j = 0; j < N; j++)

{

b[i\*N + j] = j + 1;

}

}

/\* ------------------------ CODUL TAU PANA AICI ---------------------------\*/

/\* --------------------------------------------------------------------

3. Implementați inmultirea celor doua matrici si scrieti rezultatul in

matiricea c

0.333 puncte

--------------------------------------------------------------------\*/

/\* ------------------------ REZOLVARE PCT 3 -----------------------------\*/

/\* ------------------------ CODUL TAU DE AICI ---------------------------\*/

for (int i = 0; i < N; i++)

{

for (int j = 0; j < N; j++)

for (int k = 0; k < N; k++)

{

c[i\*N + j] = a[i\*N + k] \* b[k\*N + j];

}

}

/\* ------------------------ CODUL TAU PANA AICI ---------------------------\*/

/\* --------------------------------------------------------------------

4. Implementați varianta paralela a calculului de la cerinta precedenta

Indicatie: doar bucla for exterioara (ce parcurge liniile) se distribuie

pe mai multe fire de executie

1 punct

--------------------------------------------------------------------\* /

/\* ------------------------ REZOLVARE PCT 4 -----------------------------\*/

/\* ------------------------ CODUL TAU DE AICI ---------------------------\*/

omp\_set\_num\_threads(4);

#pragma omp parallel

int id = omp\_get\_thread\_num();

int totalThreads = omp\_get\_num\_threads();

int start = N / totalThreads \* id;

int end = N / totalThreads \* (id + 1);

for (int i = start; i < end; i++)

{

for (int j = 0; j < N; j++)

for (int k = 0; k < N; k++)

{

c[i\*N + j] = a[i\*N + k] \* b[k\*N + j];

}

}

/\* ------------------------ CODUL TAU PANA AICI ---------------------------\*/

}