Laborator 5
Execuție sincronă și execuție asincronă

Problemă de rezolvat

Să se calculeze:

E = A * B + C * D

unde A, B, C, D, E sunt matrice nxn.

Să se paralelizeze algoritmul secvențial implementat. Să se compare timpii de execuție între algoritmul secvențial și cel paralel.

Algoritm secvențial

```
#include "stdafx.h"
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include <omp.h>


#define N 1000
double A[N][N], B[N][N], C[N][N], D[N][N], E1[N][N], E2[N][N],
E[N][N];

int main()
{
	clock_t start, stop;
	double durata_secv, durata_par;

	srand((unsigned)time(NULL));
	for (int i = 0; i < N; i++) {
		for (int j = 0; j<N; j++) {
			A[i][j] = (double)(rand() % 1000);
			B[i][j] = (double)(rand() % 1000);
			C[i][j] = (double)(rand() % 1000);
			D[i][j] = (double)(rand() % 1000);
		}
```

```
	}

	start = clock();
	for (int i = 0; i < N; i++) {
		for (int j = 0; j < N; j++) {
			E1[i][j] = 0;
			for (int k = 0; k < N; k++)
				E1[i][j] += A[i][k] * B[k][i];
		}
	}
	for (int i = 0; i < N; i++) {
		for (int j = 0; j < N; j++) {
			E2[i][j] = 0;
			for (int k = 0; k < N; k++)
				E2[i][j] += C[i][k] * D[k][i];
		}
	}
	for (int i = 0; i < N; i++) {
		for (int j = 0; j < N; j++) {
			E[i][j] = E1[i][j] + E2[i][j];
		}
	}
	stop = clock();
	durata_secv = (double)(stop - start) / CLOCKS_PER_SEC;
	printf("SECV %2.10f seconds\n", durata_secv);

	getchar();
    return 0;
}
```

Să se restrângă codul necesar calculului matricei E într-o singură grupare for (i,j,k). Comparați timpii de execuție necesari celor două implementări secvențiale.

Algoritm paralel sincron

```
#pragma omp parallel sections
	{
#pragma omp section
		{
			for (int i = 0; i < N; i++)
			for (int j = 0; j < N; j++) {
				E1[i][j] = 0;
				for (int k = 0; k < N; k++)
					E1[i][j] += A[i][k] * B[k][i];
```

```
                        }
                  }
#pragma omp section
            {
                  for (int i = 0; i < N; i++) {
                        for (int j = 0; j < N; j++) {
                              E2[i][j] = 0;
                              for (int k = 0; k < N; k++)
                                    E2[i][j] += C[i][k] * D[k][i];
                        }
                  }
            }
      }
#pragma omp barrier
            for (int i = 0; i < N; i++) {
                  for (int j = 0; j < N; j++) {
                        E[i][j] = E1[i][j] + E2[i][j];
                  }
            }
```

Algoritm paralel asincron

```
#define    P 2
…
#pragma omp parallel for num_threads(P)
      for (int i = 0; i < N; i++) {
                        for (int j = 0; j < N; j++) {
                              E1[i][j] = 0;
                              E2[i][j] = 0;
                              for (int k = 0; k < N; k++) {
                                    E1[i][j] += A[i][k] * B[k][i];
                                    E2[i][j] += C[i][k] * D[k][i];
                              }
                              E[i][j] = E1[i][j] + E2[i][j];
                        }
                  }
```

Cum explicați diferențele de performanță între algoritmul asincron și cel sincron?

Propuneți alte forme de implementare pentru algoritmul asincron care să permită creșterea gradului de paralelism.