МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ІВАНА ФРАНКА

Звіт із
Розпаралелення додавання/віднімання матриць Лабораторна робота №1
з курсу «Паралельні та розподілені обчислення»

Виконав:
Гуменюк Станіслав
Група Пмі-33с

Оцінка ___
Перевірив:
Пасічник
Т.В.

2024

Завдання:

Написати програми обчислення суми та різниці двох матриць (послідовний та паралельний алгоритми). Порахувати час роботи кожної з програм, обчисліть прискорення та ефективність роботи паралельного алгоритму.
В матрицях розмірності (n,m) робити змінними, щоб легко змінювати величину матриці.
Кількість потоків k - також змінна величина.
Програма повинна показувати час при послідовному способі виконання програми, а також при розпаралеленні на k потоків.
Звернути увагу на випадки, коли розмірність матриці не кратна кількості потоків.

Програмна реалізація:

Програма написана на пакеті .NET з використанням мови C#.
В меню є вибір функціоналу

```
   1. Add two random matrix with one thread
   2. Add two large random matrices with one thread
   3. Subtract two large random matrices with one thread
   4. Add two random matrix with multiple threads
   5. Add two large random matrices with multiple threads
   6. Subtract two large random matrices with multiple threads
   7. Calculate difference
-> 8. Best efficiency
   Exit
```

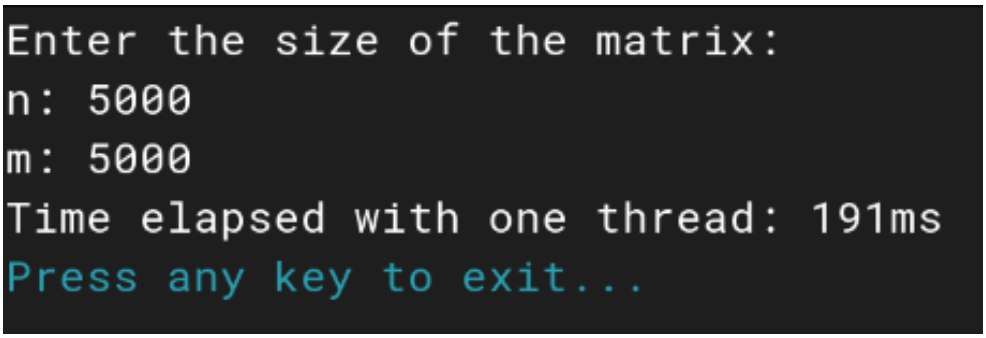
1. Додавання послідовно для перевірки правильності виконання операції

```
Enter the size of the matrix:
n: 3
m: 3
Matrix A:
7 2 4
4 0 5
6 7 4
Matrix B:
1 7 4
5 1 4
4 3 0
Matrix C:
8 9 8
9 1 9
10 10 4
Press any key to exit...
```

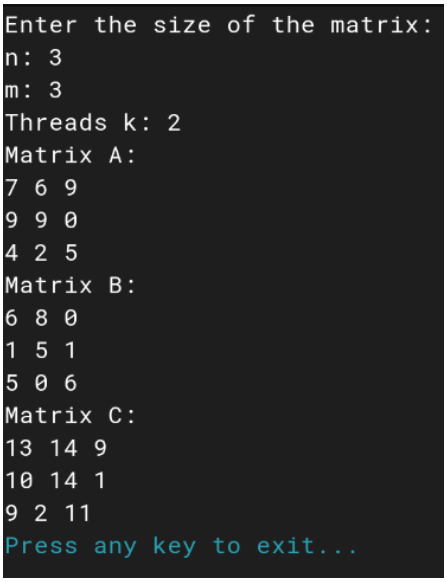
2. Додавання великих матриць послідовно для засікання часу

```
Enter the size of the matrix:
n: 5000
m: 5000
Time elapsed with one thread: 172ms
Press any key to exit...
```

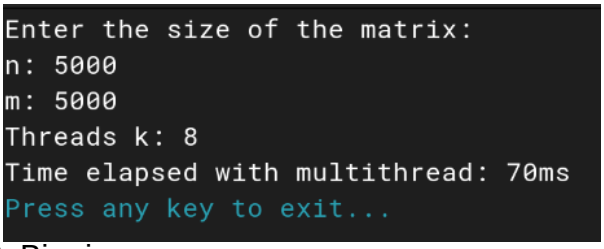
3. Віднімання великих матриць послідовно для засікання часу



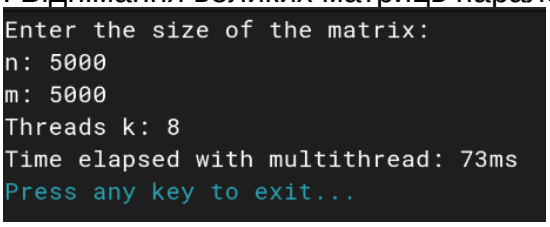
4. Додавання паралельно для перевірки правильності виконання операції



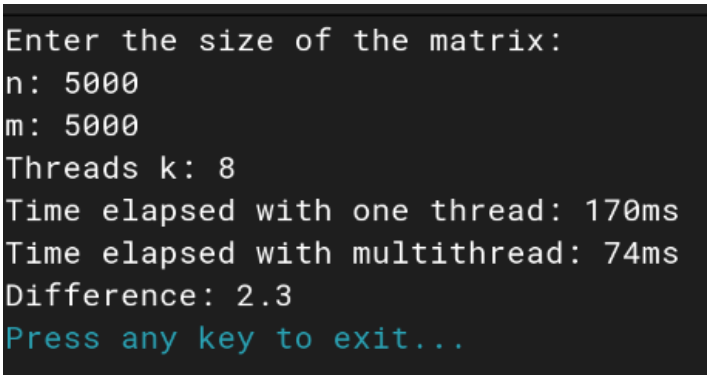
5. Додавання великих матриць паралельно для засікання часу



6. Віднімання великих матриць паралельно для засікання часу



7. Розрахунок прискорення додавання матриць



8. Найкраще прискорення для різної кількості потоків (2-16)

```
Enter the size of the matrix:
n: 5000
m: 5000
Calculating efficiency with 2 threads...
Efficiency: 1.04
Calculating efficiency with 3 threads...
Efficiency: 1.42
Calculating efficiency with 4 threads...
Efficiency: 1.42
Calculating efficiency with 5 threads...
Efficiency: 2.7
Calculating efficiency with 6 threads...
Efficiency: 1.93
Calculating efficiency with 7 threads...
Efficiency: 2.79
Calculating efficiency with 8 threads...
Efficiency: 2.93
Calculating efficiency with 9 threads...
Efficiency: 1.63
Calculating efficiency with 10 threads...
Efficiency: 1.89
Calculating efficiency with 11 threads...
Efficiency: 2.24
Calculating efficiency with 12 threads...
Efficiency: 2.54
Calculating efficiency with 13 threads...
Efficiency: 2.73
Calculating efficiency with 14 threads...
Efficiency: 2.98
Calculating efficiency with 15 threads...
Efficiency: 1.59
Calculating efficiency with 16 threads...
Efficiency: 1.37
Best efficiency: 2.98 with 14 threads
Press any key to exit...
```

Код додавання матриць послідовно:

```
3 references
static int[,] AddMatrices(int[,] A, int[,] B)
{
    int n = A.GetLength(0);
    int m = A.GetLength(1);
    int[,] C = new int[n, m];
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < m; j++)
        {
            C[i, j] = A[i, j] + B[i, j];
        }
    }
    return C;

}
```

Код додавання матриць паралельно з використанням класу Thread

```
static int[,] AddMatrices(int[,] A, int[,] B, int k)
{
    int n = A.GetLength(0);
    int m = A.GetLength(1);
    int[,] C = new int[n, m];

    // create threads
    Thread[] threads = new Thread[k];

    for (int i = 0; i < k; i++)
    {
        threads[i] = new Thread((object obj) =>
        {
            for (int j = (int)obj; j < m; j += k)
            {
                for (int l = 0; l < n; l++)
                {
                    C[l, j] = A[l, j] + B[l, j];
                }
            }
        });
        threads[i].Start(i);
    }

    // wait for all threads to finish
    for (int i = 0; i < k; i++)
    {
        threads[i].Join();
    }

    return C;
}
```

Висновок:
Обрахунок суми і різниці двох матриць є ресурсозатратним процесом для великих розмірів, тому паралельне обчислення допомагає скоротити час на виконання таких операцій. В залежності від потужності компєютера, можна використовувати різну кількість потоків. Для мого 16-ти поточного процесору, враховуючи що система забирає декілька потоків собі, найефективнішою конфігурацією виявилась 8-14 потоків