Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Звіт до лабораторної роботи №3

«Програмування для комп'ютерних систем зі спільною пам'яттю. Семафори, критичні секції, атомік-змінні, бар'єри»

з дисципліни **«Програмне забезпечення високопродуктивних комп'ютерних систем»**

Виконала:
студентка групи ІМ-13
Мартинюк Марія Павлівна
номер у списку групи 13

Перевірив:
Корочкін О. В.

Київ 2024

Мета роботи: розробка програми для ПКС зі СП
Мова програмування: C#
Засоби організації взаємодії процесів: семафори, мютекси, події, критичні секції, атомік змінні (типи), бар'єри

Вхідні дані:

- комп'ютерна система зі спільною пам'яттю включає чотири процесори і чотири пристрої введення-виведення (рис. 1.1);
- математичне завдання згідно **варіанту №7**:
  MA = (MR * MC) * d + (B * Z) * (MZ * MM) * p,
  де B, Z - вектори розміру N; MA, MR, MC, MZ, MM - матриці розміру N; p, d - скаляри;
- введення значень для MC, B виконується у процесорі 1;
  введення значення для Z, d, p виконується у процесорі 2;
  введення значень для MM та виведення результату MA виконується у процесорі 3;
  введення значень для MR, MZ виконується у процесорі 4.

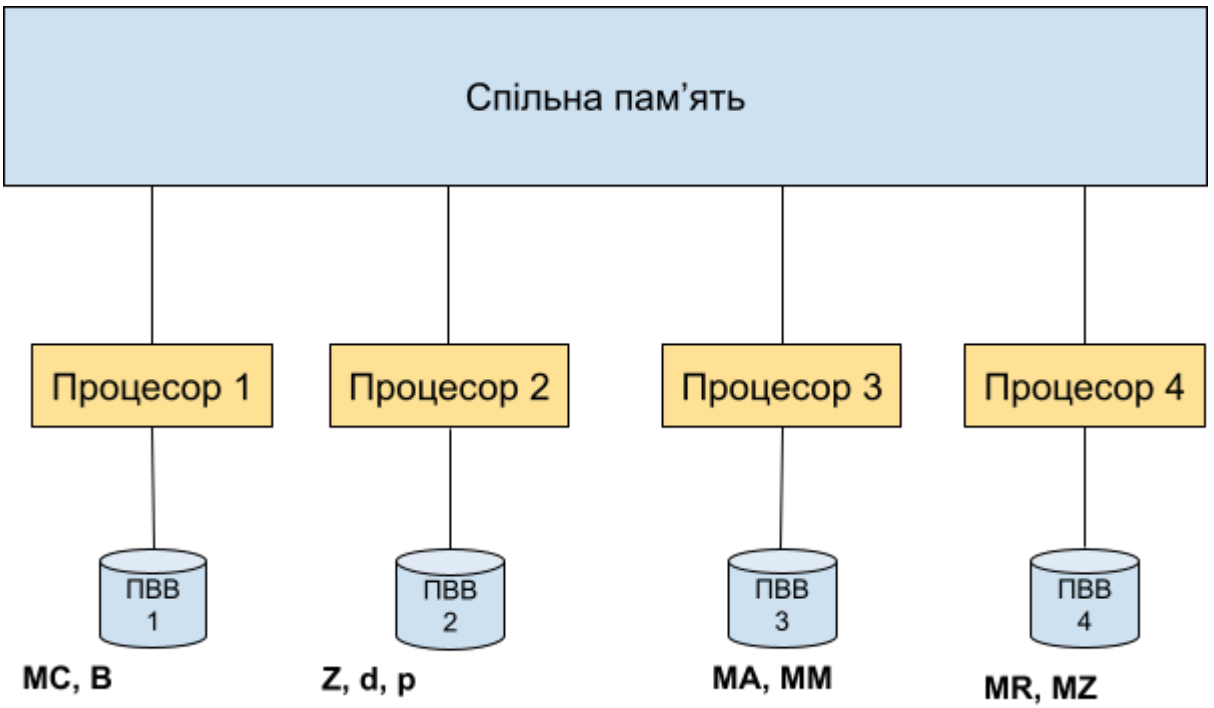


рис. 1.1 Структурна схема 4 процесорної системи

ВИКОНАННЯ РОБОТИ

Етап 1. Побудова паралельного алгоритму

$$MA = (MR * MC) * d + (B * Z) * (MZ * MM) * p$$

1. ai = (Bн * Zн), i = 1...P
2. a = a + ai

3 MAн = (MR * MCн) * d + * a * (MZ * MMн) * p d, a, p

,де н = N/P частина вектору або матриці, N - розмір вектору/матриці, P = 4 - кількість процесорів (потоків).

Етап 2. Розробка алгоритмів потоків

Задача T1	**Точки синхронізації, КД**
1. Введення MC, B	
2. Сигнал задачі T2, T3, T4 про введення MC, B	S2-1 S3-1 S4-1
3. Чекати на введення Z, d, p у задачі T2; MM у задачі T3; MR, MZ у задачі T4	W2-1 W3-1 W4-1
4. Обчислення1 a1 = (Bн * Zн)	
5. Обчислення2 a = a + a1	КД1
6. Сигнал T2, T3, T4 про завершення обчислень a	S2-2 S3-2 S4-2
7. Чекати на завершення обчислень a в T2, T3, T4	W2-2 W3-2 W4-2
8. Копія a1 = a	КД2
9. Копія p1 = p	КД3

10. Обчислення3 МВн = a1 * (MZ * ММн) * p1	
11. Копія d1 = d	КД4
12. Обчислення4 МАн = (MR * МСн) * d1 + МВн	
13. Сигнал Т3 про завершення обчислень МАн	S3-3

Задача Т2	**Точки синхронізації, КД**
1. Введення Z, d, p	
2. Сигнал задачі Т1, Т3, Т4 про введення Z, d, p	S1-1 S3-1 S4-1
3. Чекати на введення МС, В у задачі Т1; ММ у задачі Т3; MR, MZ у задачі Т4	W1-1 W3-1 W4-1
4. Обчислення1 a2 = (Вн * Zн)	
5. Обчислення2 a = a + a2	КД1
6. Сигнал Т1, Т3, Т4 про завершення обчислень a	S1-2 S3-2 S4-2
7. Чекати на завершення обчислень a в Т1, Т3, Т4	W1-2 W3-2 W4-2
8. Копія a2 = a	КД2
9. Копія p2 = p	КД3
10. Обчислення3 МВн = a2 * (MZ * ММн) * p2	
11. Копія d2 = d	КД4
12. Обчислення4 МАн = (MR * МСн) * d2 + МВн	
13. Сигнал Т3 про завершення обчислень МАн	S3-3

Задача T3	**Точки синхронізації, КД**
1. Введення MM	
2. Сигнал задачі T1, T2, T4 про введення MM	S1-1 S2-1 S4-1
3. Чекати на введення MC, B у задачі T1; Z, d, p у задачі T2; MR, MZ у задачі T4	W1-1 W2-1 W4-1
4. Обчислення1 a3 = (Bн * Zн)	
5. Обчислення2 a = a + a1	КД1
6. Сигнал T1, T2, T4 про завершення обчислень a	S1-2 S2-2 S4-2
7. Чекати на завершення обчислень a в T1, T2, T4	W1-2 W2-2 W4-2
8. Копія a3 = a	КД2
9. Копія p3 = p	КД3
10. Обчислення3 MBн = a3 * (MZ * MMн) * p3	
11. Копія d3 = d	КД4
12. Обчислення4 MAн = (MR * MCн) * d3 + MBн	
13. Чекати на завершення обчислень MAн в T1, T2, T4	W1-3 W2-3 W4-3
14. Вивід MA	

Задача T4	**Точки синхронізації, КД**
1. Введення MR, MZ	

2. Сигнал задачі T1, T2, T3 про введення MR, MZ	S1-1 S2-1 S3-1
3. Чекати на введення MC, B у задачі T1; Z, d, p у задачі T2; MM у задачі T3	W1-1 W2-1 W3-1
4. Обчислення1 a4 = (Bн * Zн)	
5. Обчислення2 a = a + a4	КД1
6. Сигнал T1, T2, T3 про завершення обчислень a	S1-2 S2-2 S3-2
7. Чекати на завершення обчислень a в T1, T2, T3	W1-2 W2-2 W3-2
8. Копія a4 = a	КД2
9. Копія p4 = p	КД3
10. Обчислення3 MBн = a4 * (MZ * MMн) * p4	
11. Копія d4 = d	КД4
12. Обчислення4 MAн = (MR * MCн) * d4 + MBн	
13. Сигнал T3 про завершення обчислень MAн	S3-3

Етап 3. Розробка схеми взаємодії задач

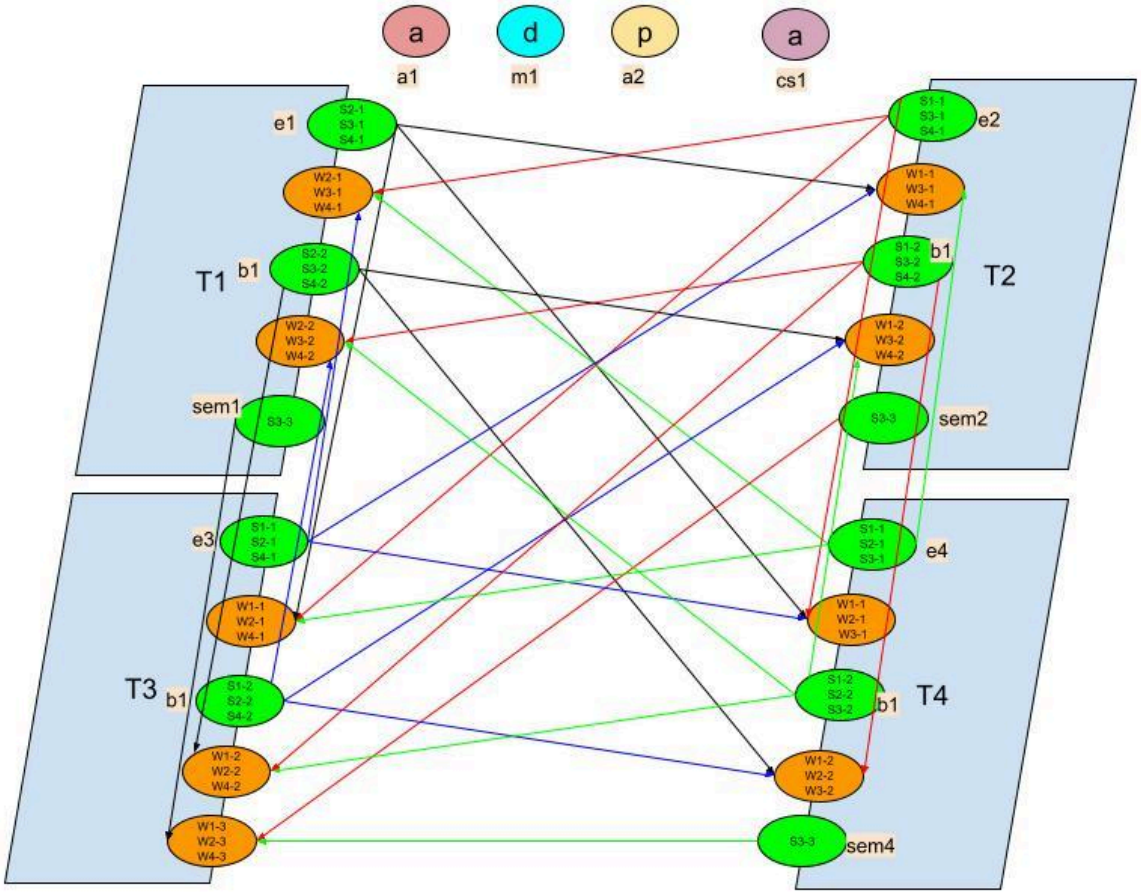


Рис.1.2 Структурна схема взаємодії задач

На структурній схемі взаємодії задач (рис.1.2) наведено такі засоби синхронізації потоків:

- Події: **e1, e2, e3, e4 -** для синхронізації завершення введення даних в потоках T1, T3, T4 відповідно;
- Семафори: **sem1, sem2, sem3** - для синхронізації завершення обчислень МАн в потоках T1, T2, T4 відповідно;
- Atomic змінні (типи):
    - **a1** - для захисту КД1 при обрахунку значення a в потоках T1, T2, T3, T4 відповідно;
    - **a2** - для захисту КД3 при копіюванні змінної p в потоках T1, T2, T3, T4 відповідно;
- Критичні секції: **cs1** - для захисту КД2 при копіюванні змінної a в потоках T1, T2, T3, T4 відповідно;
- Бар'єри: **b1 -** для синхронізації завершення обрахунку значення a в потоках T1, T2, T3, T4 відповідно;

- М’ютекси: **m1** - для захисту КД4 при копіюванні змінної d в потоках T1, T2, T3, T4 відповідно;

Структурна схема взаємодії базується на розробленому в етапі 2 алгоритмі взаємодії потоків та дозволяє наочно проконтролювати зв’язок у багатопотоковій програмі, використовуючи при цьому вище описані засоби синхронізації.

Етап 4. Розробка програми

Програма складається з основного класу Lab3, допоміжного класу з методами для обчислення математичних виразів Data та 4 класів для кожного з 4 потоків: T1, T2, T3, T4.

Lab3.cs

```
//Дисципліна "Програмне забезпечення високопродуктивних комп’ютерних систем"
//Лабораторна робота ЛР3 Варіант 7
//MA = (MR * MC) * d + (B * Z) * (MZ * MM) * p
//Мартинюк Марія Павлівна
//Дата 11.04.2024

using System.Diagnostics;

namespace lab3
{
    internal class Lab3
    {
        public static int N = 8;
        public static int P = 4;
        public static int H = N/P;
        static Stopwatch watchApp = new Stopwatch();

        public static void Main(string[] args)
        {
            watchApp.Start();

            var t1 = new Thread(() => T1.inputAndCalculate());
            var t2 = new Thread(() => T2.inputAndCalculate());
            var t3 = new Thread(() => T3.inputAndCalculate());
            var t4 = new Thread(() => T4.inputAndCalculate());
```

```
            t1.Start();
            t2.Start();
            t3.Start();
            t4.Start();

            t1.Join();
            t2.Join();
            t3.Join();
            t4.Join();

            watchApp.Stop();
            Console.WriteLine("Time: " + watchApp.ElapsedMilliseconds +
" ms");

        }
    }
}
```

Data.cs

```
using System.Text;

namespace lab3 {
public class Data {
    public static Mutex m1 = new Mutex();
    public static Barrier b1 = new Barrier(4);
    public static EventWaitHandle e1 = new EventWaitHandle(false,
EventResetMode.ManualReset);
    public static EventWaitHandle e2 = new EventWaitHandle(false,
EventResetMode.ManualReset);
    public static EventWaitHandle e3 = new EventWaitHandle(false,
EventResetMode.ManualReset);
    public static EventWaitHandle e4 = new EventWaitHandle(false,
EventResetMode.ManualReset);
    public static Semaphore sem1 = new Semaphore(0, 3);
    public static Semaphore sem2 = new Semaphore(0, 3);
    public static Semaphore sem3 = new Semaphore(0, 3);

    public static object cs1 = new object();
    public static int a;

    // Helper method to fill a vector with ones
    public static int[] VectorInput(int N) {
        int[] vector = new int[N];
```

```
        Array.Fill(vector, 1);
        return vector;
    }

    // Helper method to fill a matrix with ones
    public static int[,] MatrixInput(int N) {
        int[,] result = new int[N,N];

        for (int i = 0; i < N; i++) {
            for (int j = 0; j < N; j++) {
                result[i,j] = 1;
            }
        }
        return result;
    }

    // Helper method to calculate matrix multiplication
    public static int[,] MultiplyMatricesPartialResult(int[,] MX,
int[,] MTн, int startRow, int endRow) {
        int rows = endRow - startRow;
        int cols = MTн.GetLength(1);
        int[,] partialResult = new int[rows, cols];

        for (int i = startRow; i < endRow; i++)
        {
            for (int j = 0; j < cols; j++)
            {
                for (int k = 0; k < MTн.GetLength(0); k++)
                {
                    partialResult[i - startRow, j] += MX[i, k] * MTн[k,
j];
                }
            }
        }

        return partialResult;
    }

    // Helper method to calculate ai = Bн * Zн
    public static int VectorMultiply(int[] Bн, int[] Zн) {
        return Bн.Zip(Zн, (b, z) => b * z).Sum();
    }
```

```
    // Helper method to multiply a matrix by a scalar
    public static int[,] MatrixByScalarMultiply(int[,] matrix, int
scalar) {
        int rows = matrix.GetLength(0);
        int cols = matrix.GetLength(1);

        int[,] result = new int[rows, cols];

        for (int i = 0; i < rows; i++)
        {
            for (int j = 0; j < cols; j++)
            {
                result[i, j] = matrix[i, j] * scalar;
            }
        }

        return result;
    }

    //  Helper method to calculate matrices addition
        public static int[,] MatricesAdd(int[,] matrix1, int[,]
matrix2) {
        int rows = matrix1.GetLength(0);
        int cols = matrix1.GetLength(1);

        int[,] result = new int[rows, cols];

        for (int i = 0; i < rows; i++)
        {
            for (int j = 0; j < cols; j++)
            {
                result[i,j] = matrix1[i, j] + matrix2[i, j];
            }
        }

        return result;
    }

    // Helper method to calculate MBн  = a * (MZ * MMн) * p
    public static int[,] CalculateMBн(int[,] MZ, int[,] MMн, int
startRow, int endRow, int a, int p) {
        int[,] MZ_MMн = MultiplyMatricesPartialResult(MZ, MMн,
startRow, endRow); // MZ * MMн
```

```
        int[,] a_MBн = MatrixByScalarMultiply(MZ_MMн, a); // a * (MZ *
MMн)
        int[,] MBн = MatrixByScalarMultiply(a_MBн, p); // a * (MZ *
MMн) * p

        return MBн;
    }

    // Helper method to calculate MA = (MR * MCн) * d + MBн
    public static int[,] CalculateMA(int[,] MR, int[,] MC, int
startRow, int endRow, int d, int[,] MBн) {
        int[,] MR_MCн = MultiplyMatricesPartialResult(MR, MC, startRow,
endRow); // MR * MCн
        int[,] d_MR_MCн = MatrixByScalarMultiply(MR_MCн, d); // (MR *
MCн) * d
        int[,] MA = MatricesAdd(d_MR_MCн, MBн); // (MR * MCн) * d + MBн

        return MA;
    }

    // Method to copy a part of a matrix to the final matrix MA
    public static void CopyToFinalMatrixMA(int[,] sourceMatrix, int[,]
destinationMatrix, int startRow, int startCol)
    {
        int rows = sourceMatrix.GetLength(0);
        int cols = sourceMatrix.GetLength(1);

        for (int i = 0; i < rows; i++)
        {
        for (int j = 0; j < cols; j++)
        {
            destinationMatrix[startRow + i, startCol + j] =
sourceMatrix[i, j];
        }
        }
    }

    public static int[,] GetResult(int[,] MA1, int[,] MA2, int[,] MA3,
int[,] MA4) {
        int rows = MA4.GetLength(0);
        int cols = MA4.GetLength(1);

        int[,] result = new int[4* rows, cols];
```

```
        CopyToFinalMatrixMA(MA1, result, 0, 0);
        CopyToFinalMatrixMA(MA2, result, rows, 0);
        CopyToFinalMatrixMA(MA3, result, 2* rows, 0);
        CopyToFinalMatrixMA(MA4, result, 3*rows, 0);

        return result;
    }

    public static string ArrayToString(int[,] data) {
        int rows = data.GetLength(0);
        int cols = data.GetLength(1);

        StringBuilder builder = new StringBuilder();

        for (int i = 0; i < rows; i++)
        {
            for (int j = 0; j < cols; j++)
            {
             builder.Append(data[i, j]).Append(' ');
            }

            builder.AppendLine();
        }

        return builder.ToString();
    }
}
}
```

T1.cs

```
namespace lab3
{
    internal class T1 {
        public static int[,] MC, MAн;
        public static int[] B;
        private static int threadIndex = 1;

    public static void inputAndCalculate() {
           try {
                Console.WriteLine("T1 is started");

                // Введення MC, B
```

```
                B = Data.VectorInput(Lab3.N);
                MC = Data.MatrixInput(Lab3.N);

                //Сигнал задачі T2, T3, T4 про введення MC, B
                Data.e1.Set();

                // Чекати на введення Z, d, p у задачі T2; MM у задачі
T3; MR, MZ у задачі T4
                Data.e2.WaitOne();
                Data.e3.WaitOne();
                Data.e4.WaitOne();

                // Обчислення1 a1 = ( Bн * Zн)
                int startIndex = (threadIndex * Lab3.H) - Lab3.H;

                int[] Zн = new int[Lab3.H];
                Array.Copy(T2.Z, startIndex, Zн, 0, Lab3.H);
                int[] Bн = new int[Lab3.H];
                Array.Copy(B, startIndex, Bн, 0, Lab3.H);

                int a1 = Data.VectorMultiply(Bн, Zн);

                // Обчислення2 a = a + a1
                Interlocked.Add(ref Data.a, a1);    // КД1

                // Чекати на завершення обчислень a в T2, T3, T4
                Data.b1.SignalAndWait();

                // Копія a1 = a
                lock (Data.cs1) {
                    a1 = Data.a;    // КД2
                }

                int p1 = (int) Interlocked.Read(ref T2.p);  // КД3

                // Обчислення3 MBн  = a1 * (MZ * MMн) * p1
                int[,] MBн = Data.CalculateMBн(T4.MZ, T3.MM,
startIndex, startIndex + Lab3.H, a1, p1);

                // Копія d1 = d
                Data.m1.WaitOne();
                int d1 = T2.d;  // КД4
                Data.m1.ReleaseMutex();
```

```
                // Обчислення4 МАн = (MR * МСн) * d1 + МВн
                МАн = Data.CalculateMA(T4.MR, MC, startIndex,
startIndex + Lab3.H, d1, МВн);

                // Сигнал Т3 про завершення обчислень МАн
                Data.sem1.Release();

                Console.WriteLine("T1 is finished");
        } catch (Exception e) {
            Console.WriteLine("EXCEPTION: " + e);
        }
        }
}
}
```

T2.cs

```
namespace lab3
{
    internal class T2 {
        public static int[] Z;
        public static int d;
        public static long p;
        public static int[,] МАн;
        private static int threadIndex = 2;

        public static void inputAndCalculate() {
            try {
                Console.WriteLine("T2 is started");

                // Введення Z, d, p
                Z = Data.VectorInput(Lab3.N);
                d = 1;
                p = 1;

                //Сигнал задачі T1, T3, T4 про введення Z, d, p
                Data.e2.Set();

                // Чекати на введення МС, B у задачі T1; ММ у задачі
T3; MR, MZ у задачі T4
                Data.e1.WaitOne();
                Data.e3.WaitOne();
                Data.e4.WaitOne();
```

```
                // Обчислення1 a2 = ( Bн * Zн)
                int startIndex = (threadIndex * Lab3.H) - Lab3.H;

                int[] Zн = new int[Lab3.H];
                Array.Copy(Z, startIndex, Zн, 0, Lab3.H);
                int[] Bн = new int[Lab3.H];
                Array.Copy(T1.B, startIndex, Bн, 0, Lab3.H);

                int a2 = Data.VectorMultiply(Bн, Zн);

                // Обчислення2 a = a + a2
                Interlocked.Add(ref Data.a, a2);    // КД1

                // Чекати на завершення обчислень a в T1, T3, T4
                Data.b1.SignalAndWait();

                // Копія a2 = a
                lock (Data.cs1){
                    a2 = Data.a;    // КД2
                }

                int p2 = (int) Interlocked.Read(ref p); // КД3

                // Обчислення3 MBн  = a2 * (MZ * MMн) * p1
                int[,] MBн = Data.CalculateMBн(T4.MZ, T3.MM,
startIndex, startIndex + Lab3.H, a2, p2);

                // Копія d2 = d
                Data.m1.WaitOne();
                int d2 = d; // КД4
                Data.m1.ReleaseMutex();

                // Обчислення4 MAн = (MR * MCн) * d2 + MBн
                MAн = Data.CalculateMA(T4.MR, T1.MC, startIndex,
startIndex + Lab3.H, d2, MBн);

                // Сигнал T3 про завершення обчислень MAн
                Data.sem2.Release();

                Console.WriteLine("T2 is finished");
            } catch (Exception e) {
            Console.WriteLine("EXCEPTION: " + e);
```

```
        }
        }
    }
}
```

T3.cs

```
namespace lab3
{
    internal class T3 {
        public static int[,] MM, MAн;
        private static int threadIndex = 3;

         public static void inputAndCalculate() {
            try {
                Console.WriteLine("T3 is started");

                // Введення MM
                MM = Data.MatrixInput(Lab3.N);

               //Сигнал задачі T1, T2, T4 про введення MM
                Data.e3.Set();

                // Чекати на введення MC, B у задачі T1; Z, d, p у
задачі T2; MR, MZ у задачі T4
                Data.e1.WaitOne();
                Data.e2.WaitOne();
                Data.e4.WaitOne();

                // Обчислення1 a3 = ( Bн * Zн)
                int startIndex = (threadIndex * Lab3.H) - Lab3.H;

                int[] Zн = new int[Lab3.H];
                Array.Copy(T2.Z, startIndex, Zн, 0, Lab3.H);
                int[] Bн = new int[Lab3.H];
                Array.Copy(T1.B, startIndex, Bн, 0, Lab3.H);

                int a3 = Data.VectorMultiply(Bн, Zн);

                // Обчислення2 a = a + a3
                Interlocked.Add(ref Data.a, a3);    // КД1

                // Чекати на завершення обчислень a в T1, T2, T4
                Data.b1.SignalAndWait();
```

```
                // Копія a3 = a
                lock (Data.cs1){
                    a3 = Data.a;     // КД2
                }

                int p3 = (int) Interlocked.Read(ref T2.p);  // КД3

                // Обчислення3 MBн  = a1 * (MZ * MMн) * p3
                int[,] MBн = Data.CalculateMBн(T4.MZ, MM, startIndex,
startIndex + Lab3.H, a3, p3);

                // Копія d3 = d
                Data.m1.WaitOne();
                int d3 = T2.d;  // КД4
                Data.m1.ReleaseMutex();

                // Обчислення4 MAн = (MR * MCн) * d3 + MBн
                MAн = Data.CalculateMA(T4.MR, T1.MC, startIndex,
startIndex + Lab3.H, d3, MBн);

                // Чекати на завершення обчислень MAн в T1, T2, T4
                Data.sem1.WaitOne();
                Data.sem2.WaitOne();
                Data.sem3.WaitOne();

                Console.WriteLine("T1.MA: " +
Data.ArrayToString(T1.MAн));
                Console.WriteLine("T2.MA: " +
Data.ArrayToString(T2.MAн));
                Console.WriteLine("T3.MA: " + Data.ArrayToString(MAн));
                Console.WriteLine("T4.MA: " +
Data.ArrayToString(T4.MAн));

                int[,] MA = Data.GetResult(T1.MAн, T2.MAн, MAн,
T4.MAн);
                Console.WriteLine("MA: \n" + Data.ArrayToString(MA));

                Console.WriteLine("T3 is finished");
            } catch (Exception e) {
                Console.WriteLine("EXCEPTION: " + e);
        }
        }
    }
```

```
}
```

T4.cs

```
namespace lab3
{
    internal class T4 {
        public static int[,] MR, MZ, MAн;
        private static int threadIndex = 4;

        public static void inputAndCalculate() {
            try {
                Console.WriteLine("T4 is started");

                // Введення MR, MZ
                MR = Data.MatrixInput(Lab3.N);
                MZ = Data.MatrixInput(Lab3.N);

                // //Сигнал задачі T1, T3, T4 про введення Z, d, p
                Data.e4.Set();

                // // Чекати на введення MC, B у задачі T1; Z, d, p у
задачі T2; MM у задачі T3
                Data.e1.WaitOne();
                Data.e2.WaitOne();
                Data.e3.WaitOne();

                // Обчислення1 a4 = ( Bн * Zн)
                int startIndex = (threadIndex * Lab3.H) - Lab3.H;

                int[] Zн = new int[Lab3.H];
                Array.Copy(T2.Z, startIndex, Zн, 0, Lab3.H);
                int[] Bн = new int[Lab3.H];
                Array.Copy(T1.B, startIndex, Bн, 0, Lab3.H);

                int a4 = Data.VectorMultiply(Bн, Zн);

                // Обчислення2 a = a + a4
                Interlocked.Add(ref Data.a, a4);    // КД1

                // Чекати на завершення обчислень a в T1, T3, T4
                Data.b1.SignalAndWait();

                // Копія a2 = a
```

```
                lock (Data.cs1){
                    a4 = Data.a;     // КД2
                }

                int p4 = (int) Interlocked.Read(ref T2.p);  // КД3

                // Обчислення3 МВн  = a2 * (MZ * ММн) * p1
                int[,] МВн = Data.CalculateMBн(MZ, T3.MM, startIndex,
startIndex + Lab3.H, a4, p4);

                // Копія d4 = d
                Data.m1.WaitOne();
                int d4 = T2.d;  // КД4
                Data.m1.ReleaseMutex();

                // Обчислення4 МАн = (MR * МСн) * d4 + МВн
                МАн = Data.CalculateMA(MR, T1.MC, startIndex,
startIndex + Lab3.H, d4, МВн);

                // Сигнал Т3 про завершення обчислень МАн
                Data.sem3.Release();

                Console.WriteLine("T4 is finished");
            } catch (Exception e) {
            Console.WriteLine("EXCEPTION: " + e);
        }
        }
    }
}
```

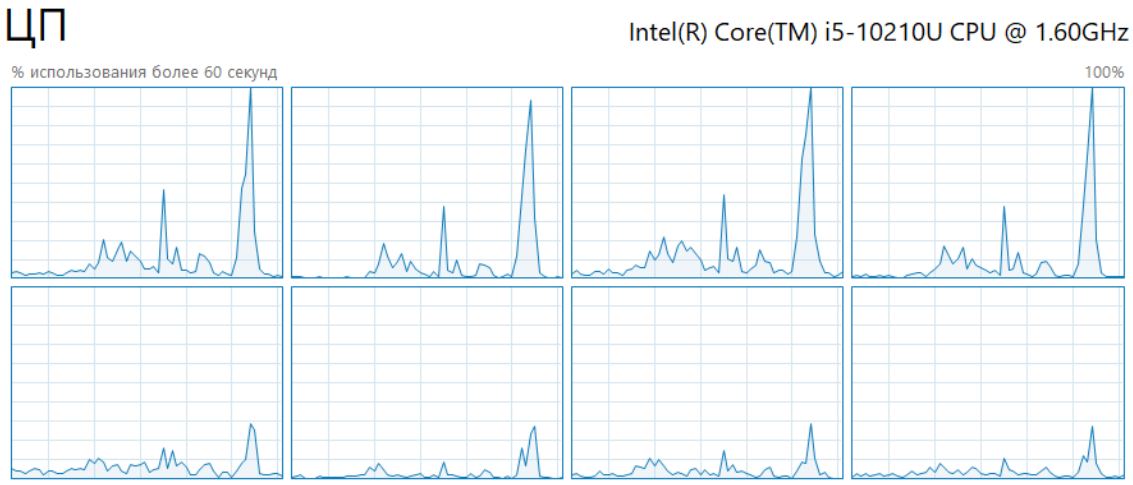
Результат виконання:

На рис. нижче наведений результат виконання багатопотокової програми при N = 8, P = 4. Значення усіх елементів використовуваних матриць та векторів рівні 1. Задля впевненості у правильності розрахунків було додатково виведено у консоль проміжні результати обчислень остаточного результату матриці. Як видно з рис. нижче разрахунок частин та об’єднання їх у остаточний результат відбувається коректно.

```
T4 is started
T1 is started
T2 is started
T3 is started

T1 is finished
T4 is finished
T2 is finished
T1.MA: 72 72 72 72 72 72 72 72
72 72 72 72 72 72 72 72

T2.MA: 72 72 72 72 72 72 72 72
72 72 72 72 72 72 72 72

T3.MA: 72 72 72 72 72 72 72 72
72 72 72 72 72 72 72 72

T4.MA: 72 72 72 72 72 72 72 72
72 72 72 72 72 72 72 72

MA:
8 72 72 72 72 72 72 72 72

T3 is finished
Time: 69 ms
```

Дослідження завантаженості ядер процесору:

Варто зазначити, що конкретно мій комп'ютер містить 8 логічних ядер та 4 фізичних. Нижче наведений рис. демонструє, що при використанні 4 ядер процесору розподілення обчислювальних можливостей відбувається рівномірно між кожним з них. Поєднуючи цю інформацію із раніше отриманими остаточними значеннями обрахунків, можна зробити висновок, що чотирьох потокова програма написана коректно.



Додатково було обраховано коефіцієнт прискорення при запуску програми на 4 ядрах та на 1 ядрі при N = 8: Кп = 368 / 101 = 3,64. Тобто при використанні 4 ядер замість 1, програма завершується у 3,64 рази швидше.

Висновок:

1 Побудовано та розроблено паралельний алгоритм для обчислення математичної задачі. За отриманим алгоритмом було визначено критичні ділянки з використання спільних ресурсів a, d, p, a.

2 Задля реалізації коректної синхронізації паралельних обчислень було використано такі засоби мови C#: події, семафори, м’ютекси, atomic змінні (типи), критичні секції та бар’єри.

3 Побудовано схему взаємодії задач, де позначені всі використані засоби синхронізації паралельних обчислень.

4 Отриманий вивід результату обчислень демонструє, що розрахунок остаточного значення відбувається коректно. Аналіз завантаженості ядер процесору при запуску програми підтвердив рівномірне розподілення навантаження між задіяними ядрами, що свідчить про успішну оптимізацію паралельних обчислень.

5 Обчислений коефіцієнт прискорення Кп = 3,64, що також сповіщає про пришвидшення виконання програми у разі використання 4 ядер процесора замість 1.