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

# Лабораторна робота №5

з дисципліни

Програмне забезпечення високопродуктивних комп'ютерних систем

ВИКОНАЛА:
Студентка групи ІМ-21
Рабійчук Дар’я Олександрівна
№ у списку(варіант) - 18

ПЕРЕВІРИВ:
доц. Корочкін О. В.

**Київ 2025**

**Завдання**

**Варіант 22**

a = ((B*MX)*(Z*MS)) * min(B)

Введення-виведення даних:

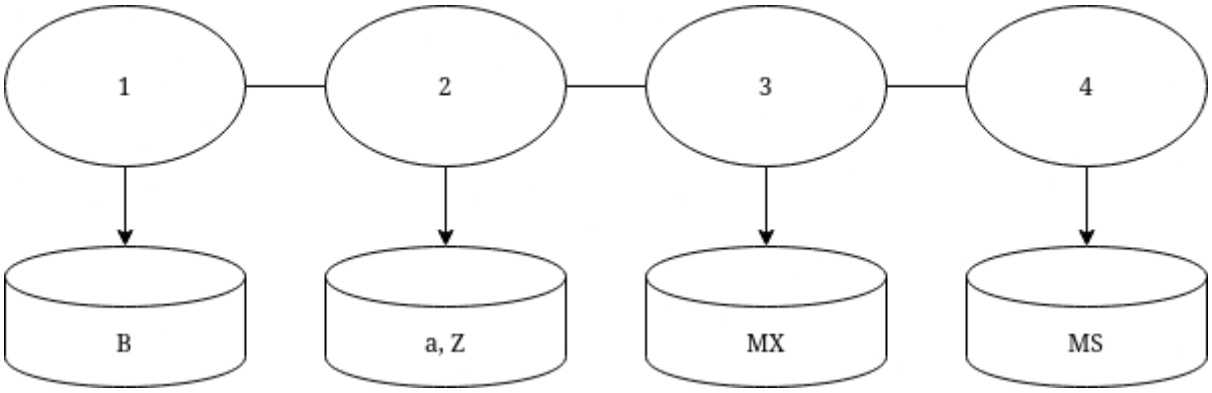
1 B

2 a, Z

3 MX

4 MS

Структура 1




**Етап 1. Побудова паралельного математичного алгоритму**

1) bi = min(Bн)
2) b = min(b, bi) | CP: b
3) ai = ((B*MXн)*(Z*MSн)) * b | CP: B, Z, b
4) a = a + ai | CP: a

**Етап 2. Розробка алгоритмів потоків**

T1

1. Ввести B
2. Передати в T2 дан B
3. Отримати від T2 дані Z, MXн, MSн
4. Обчислити b1 = min(Bн)

5. Передати T2 дані b1
6. Отримати від T2 дані b
7. Обчислити  a1 = ((B*MXн)*(Z*MSн)) * b
8. Передати T2 дані a1

T2

1. Ввести Z
2. Отримати від T1 дані B
3. Передати в T3 дан B, Z
4. Отримати від T3 дані MX2н, MS2н
5. Передати в T1 дані Z, MXн, MSн
6. Обчислити b2 = min(Bн)
7. Отримати від T1 дані b1
8. Обчислити b2 = min(b1, b2)
9. Передати T3 дані b2
10. Отримати від T3 дані b
11. Передати T1 дані b
12. Обчислити  a2 = ((B*MXн)*(Z*MSн)) * b
13. Отримати від T1 дані a1
14. Обчислити a2 = a1 + a2
15. Отримати від T3 дані a3
16. Обчислити a = a2 + a3
17. Вивести a

T3

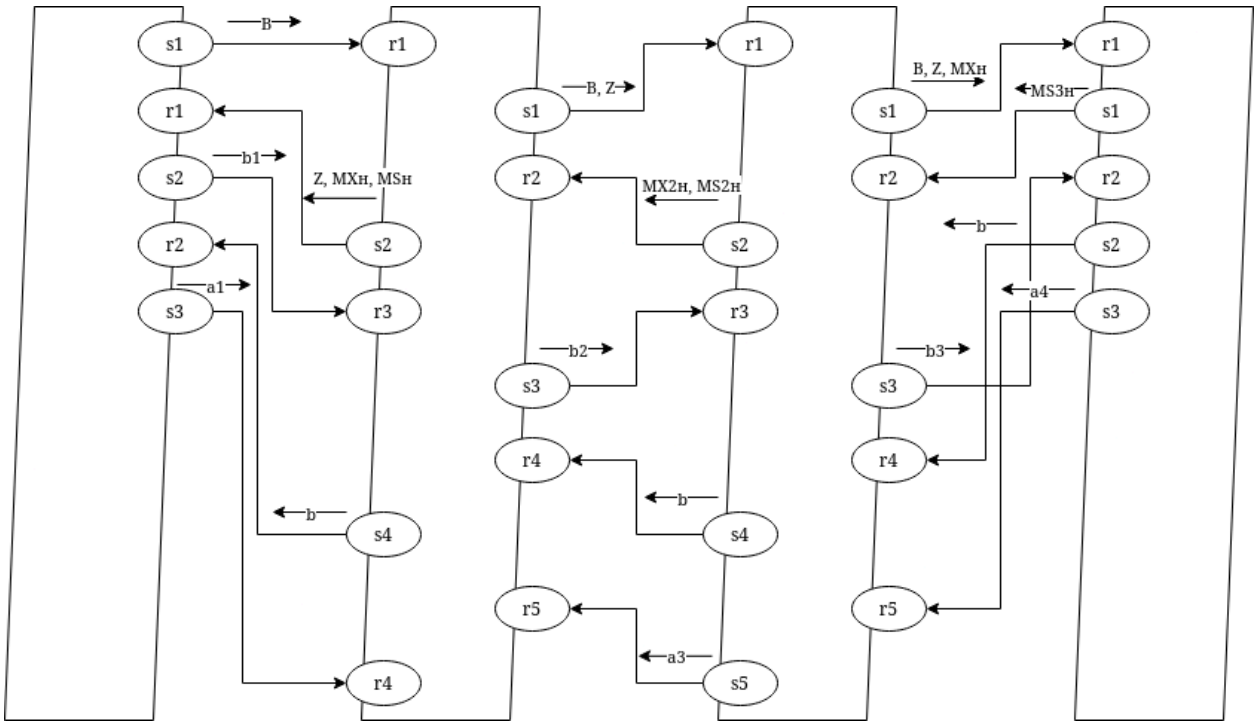
1. Ввести MX
2. Отримати від T2 дані B, Z
3. Передати в T4 дані B, Z, MXн
4. Отримати від T4 дані MS3н

5. Передати в T2 дан MX2н, MS2н
6. Обчислити b3 = min(Bн)
7. Отримати від T2 дані b2
8. Обчислити b3 = min(b2, b3)
9. Передати T4 дані b3
10. Отримати від T4 дані b
11. Передати T2 дані b
12. Обчислити  a3 = ((B*MXн)*(Z*MSн)) * b
13. Отримати від T4 дані a4
14. Обчислити a3 = a3 + a4
15. Передати T2 дані a3

T3

1. Ввести MS
2. Отримати від T3 дані B, Z, MXн
3. Передати в T3 дані MS3н
4. Обчислити b4 = min(Bн)
5. Отримати від T3 дані b3
6. Обчислити b = min(b3, b4)
7. Передати T3 дані b
8. Обчислити  a4 = ((B*MXн)*(Z*MSн)) * b
9. Передати T3 дані a4

**Етап 3. Розробка схеми**




## Етап 4. Розробка програми

## Lab5.java

```
// Програмне забезпечення високопродуктивних комп'ютерних систем

// Лабораторна робота No5
// Варіант 22
// a = ((B*MX)*(Z*MS)) * min(B)
// Рабійчук Дар'я Олександрівна
// 24.05.2025

import mpi.*;
import java.io.*;
import java.util.*;

public class Lab5 {
    private static final int N = 8; // Розмір векторів/матриць
    private static final int P = 4; // Кількість потоків
    private static final int H = N / P; // Розмір підмасиву для кожного потоку

    public static void main(String[] args) throws MPIException {
        MPI.Init(args);

        int rank = MPI.COMM_WORLD.Rank();
```

```
        int size = MPI.COMM_WORLD.Size();

        if (size != P) {
            if (rank == 1) {
                System.out.println("Програма потребує рівно " + P + " потоків!");
            }
            MPI.Finalize();
            return;
        }

        long startTime = System.currentTimeMillis();

        switch (rank) {
            case 0:
                thread1();
                break;
            case 1:
                thread2();
                break;
            case 2:
                thread3();
                break;
            case 3:
                thread4();
                break;
        }

        if (rank == 0) {
            long endTime = System.currentTimeMillis();
            System.out.println("Загальний час виконання: " + (endTime -
startTime) + " мс");
        }

        MPI.Finalize();
    }

    // T1
    private static void thread1() throws MPIException {
        System.out.println("Потік T1 почався");

        // Локальні змінні для T1
        int[] B = new int[N];
        int[] Z = new int[N];
        int[][] MX = new int[N][N];
        int[][] MS = new int[N][N];

        // 1. Ввести B
        System.out.println("T1: Крок 1 - Введення вектору B");
        fillVector(B, 1);

        // 2. Передати в T2 дані B
        System.out.println("T1: Крок 2 - Передача B до T2");
        MPI.COMM_WORLD.Send(B, 0, N, MPI.INT, 1, 1);

        // 3. Отримати від T2 дані Z, MXн, MSн
        System.out.println("T1: Крок 3 - Отримання Z, MXн, MSн від T2");
        MPI.COMM_WORLD.Recv(Z, 0, N, MPI.INT, 1, 2);

        int[][] MXh = new int[H][N];
```

```
        int[][] MSh = new int[H][N];
        for (int i = 0; i < H; i++) {
            MPI.COMM_WORLD.Recv(MXh[i], 0, N, MPI.INT, 1, 3 + i);
            MPI.COMM_WORLD.Recv(MSh[i], 0, N, MPI.INT, 1, 3 + H + i);
        }

        // Копіювання отриманих даних у повні матриці
        for (int i = 0; i < H; i++) {
            System.arraycopy(MXh[i], 0, MX[i], 0, N);
            System.arraycopy(MSh[i], 0, MS[i], 0, N);
        }

        // 4. Обчислити b1 = min(Bн)
        System.out.println("T1: Крок 4 - Обчислення b1 = min(Bн)");
        int b1 = findMinInRange(B, 0, H);

        // 5. Передати T2 дані b1
        System.out.println("T1: Крок 5 - Передача b1 до T2");
        int[] b1Array = {b1};
        MPI.COMM_WORLD.Send(b1Array, 0, 1, MPI.INT, 1, 4);

        // 6. Отримати від T2 дані b
        System.out.println("T1: Крок 6 - Отримання b від T2");
        int[] bArray = new int[1];
        MPI.COMM_WORLD.Recv(bArray, 0, 1, MPI.INT, 1, 5);
        int b = bArray[0];

        // 7. Обчислити a1 = ((B*MXн)*(Z*MSн)) * b
        System.out.println("T1: Крок 7 - Обчислення a1");
        int a1 = computePartialResult(B, MX, Z, MS, 0, H, b);


        // 8. Передати T2 дані a1
        System.out.println("T1: Крок 8 - Передача a1 до T2");
        int[] a1Array = {a1};
        MPI.COMM_WORLD.Send(a1Array, 0, 1, MPI.INT, 1, 6);

        System.out.println("Потік T1 закінчився");
    }

    // T2
    private static void thread2() throws MPIException {
        System.out.println("Потік T2 почався");

        // Локальні змінні для T2
        int[] B = new int[N];
        int[] Z = new int[N];
        int[][] MX = new int[N][N];
        int[][] MS = new int[N][N];

        // 1. Ввести Z
        System.out.println("T2: Крок 1 - Введення вектору Z");
        fillVector(Z, 2);

        // 2. Отримати від T1 дані B
        System.out.println("T2: Крок 2 - Отримання B від T1");
        MPI.COMM_WORLD.Recv(B, 0, N, MPI.INT, 0, 1);

        // 3. Передати в T3 дані B, Z
```

```
        System.out.println("T2: Крок 3 - Передача B, Z до T3");
        MPI.COMM_WORLD.Send(B, 0, N, MPI.INT, 2, 7);
        MPI.COMM_WORLD.Send(Z, 0, N, MPI.INT, 2, 8);

        // 4. Отримати від T3 дані MX2н, MS2н
        System.out.println("T2: Крок 4 - Отримання MX2н, MS2н від T3");
        int[][] MX2h = new int[2*H][N];
        int[][] MS2h = new int[2*H][N];
        for (int i = 0; i < 2*H; i++) {
            MPI.COMM_WORLD.Recv(MX2h[i], 0, N, MPI.INT, 2, 9 + i);
            MPI.COMM_WORLD.Recv(MS2h[i], 0, N, MPI.INT, 2, 9 + 2*H + i);
        }

        // Копіювання у повні матриці
        for (int i = 0; i < 2*H; i++) {
            System.arraycopy(MX2h[i], 0, MX[i], 0, N);
            System.arraycopy(MS2h[i], 0, MS[i], 0, N);
        }

        // 5. Передати в T1 дані Z, MXн, MSн
        System.out.println("T2: Крок 5 - Передача Z, MXн, MSн до T1");
        MPI.COMM_WORLD.Send(Z, 0, N, MPI.INT, 0, 2);
        for (int i = 0; i < H; i++) {
            MPI.COMM_WORLD.Send(MX[i], 0, N, MPI.INT, 0, 3 + i);
            MPI.COMM_WORLD.Send(MS[i], 0, N, MPI.INT, 0, 3 + H + i);
        }

        // 6. Обчислити b2 = min(Bн)
        System.out.println("T2: Крок 6 - Обчислення b2 = min(Bн)");
        int b2 = findMinInRange(B, H, 2*H);

        // 7. Отримати від T1 дані b1
        System.out.println("T2: Крок 7 - Отримання b1 від T1");
        int[] b1Array = new int[1];
        MPI.COMM_WORLD.Recv(b1Array, 0, 1, MPI.INT, 0, 4);
        int b1 = b1Array[0];

        // 8. Обчислити b2 = min(b1, b2)
        System.out.println("T2: Крок 8 - Обчислення b2 = min(b1, b2)");
        b2 = Math.min(b1, b2);

        // 9. Передати T3 дані b2
        System.out.println("T2: Крок 9 - Передача b2 до T3");
        int[] b2Array = {b2};
        MPI.COMM_WORLD.Send(b2Array, 0, 1, MPI.INT, 2, 10);

        // 10. Отримати від T3 дані b
        System.out.println("T2: Крок 10 - Отримання b від T3");
        int[] bArray = new int[1];
        MPI.COMM_WORLD.Recv(bArray, 0, 1, MPI.INT, 2, 11);
        int b = bArray[0];

        // 11. Передати T1 дані b
        System.out.println("T2: Крок 11 - Передача b до T1");
        MPI.COMM_WORLD.Send(bArray, 0, 1, MPI.INT, 0, 5);

        // 12. Обчислити a2 = ((B*MXн)*(Z*MSн)) * b
        System.out.println("T2: Крок 12 - Обчислення a2");
        int a2 = computePartialResult(B, MX, Z, MS, H, 2*H, b);
```

```
        // 13. Отримати від T1 дані a1
        System.out.println("T2: Крок 13 - Отримання a1 від T1");
        int[] a1Array = new int[1];
        MPI.COMM_WORLD.Recv(a1Array, 0, 1, MPI.INT, 0, 6);
        int a1 = a1Array[0];

        // 14. Обчислити a2 = a1 + a2
        System.out.println("T2: Крок 14 - Обчислення a = a1 + a2");
        a2 = a1 + a2;

        // 15. Отримати від T3 дані a3
        System.out.println("T2: Крок 15 - Отримання a3 від T3");
        int[] a3Array = new int[1];
        MPI.COMM_WORLD.Recv(a3Array, 0, 1, MPI.INT, 2, 12);
        int a3 = a3Array[0];

        // 16. Обчислити a = a2 + a3
        System.out.println("T2: Крок 15 - Обчислення a2 = a2 + a3");
        int a = a2 + a3;

        // 17. Вивести a
        System.out.println("T1: Крок 10 - Виведення результату");
        System.out.println("Фінальний результат: " + a);


        System.out.println("Потік T2 закінчився");
    }

    // T3
    private static void thread3() throws MPIException {
        System.out.println("Потік T3 почався");

        // Локальні змінні для T3
        int[] B = new int[N];
        int[] Z = new int[N];
        int[][] MX = new int[N][N];
        int[][] MS = new int[N][N];

        // 1. Ввести MX
        System.out.println("T3: Крок 1 - Введення матриці MX");
        fillMatrix(MX, 3);

        // 2. Отримати від T2 дані B, Z
        System.out.println("T3: Крок 2 - Отримання B, Z від T2");
        MPI.COMM_WORLD.Recv(B, 0, N, MPI.INT, 1, 7);
        MPI.COMM_WORLD.Recv(Z, 0, N, MPI.INT, 1, 8);

        // 3. Передати в T4 дані B, Z, MXн
        System.out.println("T3: Крок 3 - Передача B, Z, MXн до T4");
        MPI.COMM_WORLD.Send(B, 0, N, MPI.INT, 3, 13);
        MPI.COMM_WORLD.Send(Z, 0, N, MPI.INT, 3, 14);
        for (int i = 2*H; i < 3*H; i++) {
            MPI.COMM_WORLD.Send(MX[i], 0, N, MPI.INT, 3, 15 + (i - 2*H));
        }

        // 4. Отримати від T4 дані MS3н
        System.out.println("T3: Крок 4 - Отримання MS3н від T4");
        int[][] MS3h = new int[H][N];
```

```
        for (int i = 0; i < H; i++) {
            MPI.COMM_WORLD.Recv(MS3h[i], 0, N, MPI.INT, 3, 16 + i);
        }

        // Копіювання у повну матрицю
        for (int i = 0; i < H; i++) {
            System.arraycopy(MS3h[i], 0, MS[2*H + i], 0, N);
        }

        // 5. Передати в T2 дані MX2н, MS2н
        System.out.println("T3: Крок 5 - Передача MX2н, MS2н до T2");
        for (int i = 0; i < 2*H; i++) {
            MPI.COMM_WORLD.Send(MX[i], 0, N, MPI.INT, 1, 9 + i);
            MPI.COMM_WORLD.Send(MS[i], 0, N, MPI.INT, 1, 9 + 2*H + i);
        }

        // 6. Обчислити b3 = min(Bн)
        System.out.println("T3: Крок 6 - Обчислення b3 = min(Bн)");
        int b3 = findMinInRange(B, 2*H, 3*H);

        // 7. Отримати від T2 дані b2
        System.out.println("T3: Крок 7 - Отримання b2 від T2");
        int[] b2Array = new int[1];
        MPI.COMM_WORLD.Recv(b2Array, 0, 1, MPI.INT, 1, 10);
        int b2 = b2Array[0];

        // 8. Обчислити b3 = min(b2, b3)
        System.out.println("T3: Крок 8 - Обчислення b3 = min(b2, b3)");
        b3 = Math.min(b2, b3);

        // 9. Передати T4 дані b3
        System.out.println("T3: Крок 9 - Передача b3 до T4");
        int[] b3Array = {b3};
        MPI.COMM_WORLD.Send(b3Array, 0, 1, MPI.INT, 3, 17);

        // 10. Отримати від T4 дані b
        System.out.println("T3: Крок 10 - Отримання b від T4");
        int[] bArray = new int[1];
        MPI.COMM_WORLD.Recv(bArray, 0, 1, MPI.INT, 3, 18);
        int b = bArray[0];

        // 11. Передати T2 дані b
        System.out.println("T3: Крок 11 - Передача b до T2");
        MPI.COMM_WORLD.Send(bArray, 0, 1, MPI.INT, 1, 11);

        // 12. Обчислити a3 = ((B*MXн)*(Z*MSн)) * b
        System.out.println("T3: Крок 12 - Обчислення a3");
        int a3 = computePartialResult(B, MX, Z, MS, 2*H, 3*H, b);

        // 13. Отримати від T4 дані a4
        System.out.println("T3: Крок 13 - Отримання a4 від T4");
        int[] a4Array = new int[1];
        MPI.COMM_WORLD.Recv(a4Array, 0, 1, MPI.INT, 3, 19);
        int a4 = a4Array[0];

        // 14. Обчислити a3 = a3 + a4
        System.out.println("T3: Крок 14 - Обчислення a3 = a3 + a4");
        a3 = a3 + a4;
```

```
        // 15. Передати T2 дані a3
        System.out.println("T3: Крок 15 - Передача a3 до T2");
        int[] a3FinalArray = {a3};
        MPI.COMM_WORLD.Send(a3FinalArray, 0, 1, MPI.INT, 1, 12);

        System.out.println("Потік T3 закінчився");
    }

    // T4
    private static void thread4() throws MPIException {
        System.out.println("Потік T4 почався");

        // Локальні змінні для T4
        int[] B = new int[N];
        int[] Z = new int[N];
        int[][] MX = new int[N][N];
        int[][] MS = new int[N][N];

        // 1. Ввести MS
        System.out.println("T4: Крок 1 - Введення матриці MS");
        fillMatrix(MS, 4);

        // 2. Отримати від T3 дані B, Z, MXн
        System.out.println("T4: Крок 2 - Отримання B, Z, MXн від T3");
        MPI.COMM_WORLD.Recv(B, 0, N, MPI.INT, 2, 13);
        MPI.COMM_WORLD.Recv(Z, 0, N, MPI.INT, 2, 14);
        for (int i = 0; i < H; i++) {
            MPI.COMM_WORLD.Recv(MX[2*H + i], 0, N, MPI.INT, 2, 15 + i);
        }

        // 3. Передати в T3 дані MS3н
        System.out.println("T4: Крок 3 - Передача MS3н до T3");
        for (int i = 2*H; i < 3*H; i++) {
            MPI.COMM_WORLD.Send(MS[i], 0, N, MPI.INT, 2, 16 + (i - 2*H));
        }

        // 4. Обчислити b4 = min(Bн)
        System.out.println("T4: Крок 4 - Обчислення b4 = min(Bн)");
        int b4 = findMinInRange(B, 3*H, N);

        // 5. Отримати від T3 дані b3
        System.out.println("T4: Крок 5 - Отримання b3 від T3");
        int[] b3Array = new int[1];
        MPI.COMM_WORLD.Recv(b3Array, 0, 1, MPI.INT, 2, 17);
        int b3 = b3Array[0];

        // 6. Обчислити b = min(b3, b4)
        System.out.println("T4: Крок 6 - Обчислення b = min(b3, b4)");
        int b = Math.min(b3, b4);

        // 7. Передати T3 дані b
        System.out.println("T4: Крок 7 - Передача b до T3");
        int[] bArray = {b};
        MPI.COMM_WORLD.Send(bArray, 0, 1, MPI.INT, 2, 18);

        // 8. Обчислити a4 = ((B*MXн)*(Z*MSн)) * b
        System.out.println("T4: Крок 8 - Обчислення a4");
        int a4 = computePartialResult(B, MX, Z, MS, 3*H, N, b);
```

```
        // 9. Передати T3 дані a4
        System.out.println("T4: Крок 9 - Передача a4 до T3");
        int[] a4Array = {a4};
        MPI.COMM_WORLD.Send(a4Array, 0, 1, MPI.INT, 2, 19);

        System.out.println("Потік T4 закінчився");
    }

    // Допоміжні функції
    private static void fillVector(int[] vector, int threadNum) {
        for (int i = 0; i < vector.length; i++) {
            vector[i] = threadNum; // Заповнення номером потоку
        }
    }

    private static void fillMatrix(int[][] matrix, int threadNum) {
        for (int i = 0; i < matrix.length; i++) {
            for (int j = 0; j < matrix[i].length; j++) {
                matrix[i][j] = threadNum; // Заповнення номером потоку
            }
        }
    }

    private static int findMinInRange(int[] array, int start, int end) {
        int min = array[start];
        for (int i = start + 1; i < end; i++) {
            if (array[i] < min) {
                min = array[i];
            }
        }
        return min;
    }

    private static int computePartialResult(int[] B, int[][] MX, int[] Z, int[][]
MS,
                                            int startRow, int endRow, int b) {
        int result = 0;

        for (int i = startRow; i < endRow; i++) {
            // Обчислення B*MXн для рядка i
            int bmx = 0;
            for (int j = 0; j < N; j++) {
                bmx += B[j] * MX[i][j];
            }

            // Обчислення Z*MSн для рядка i
            int zms = 0;
            for (int j = 0; j < N; j++) {
                zms += Z[j] * MS[i][j];
            }

            // Додавання до результату
            result += bmx * zms;
        }

        return result * b;
    }
}
```

## Вивід програми при N=8

```
MPJ Express (0.44) is started in the multicore configuration
Потік T1 почався
T1: Крок 1 – Введення вектору B
T1: Крок 2 – Передача B до T2
Потік T3 почався
Потік T4 почався
T4: Крок 1 – Введення матриці MS
T3: Крок 1 – Введення матриці MX
T4: Крок 2 – Отримання B, Z, MXн від T3
T3: Крок 2 – Отримання B, Z від T2
Потік T2 почався
T2: Крок 1 – Введення вектору Z
T2: Крок 2 – Отримання B від T1
T1: Крок 3 – Отримання Z, MXн, MSн від T2
T2: Крок 3 – Передача B, Z до T3
T2: Крок 4 – Отримання MX2н, MS2н від T3
T3: Крок 3 – Передача B, Z, MXн до T4
T3: Крок 4 – Отримання MS3н від T4
T4: Крок 3 – Передача MS3н до T3
T4: Крок 4 – Обчислення b4 = min(Bн)
T3: Крок 5 – Передача MX2н, MS2н до T2
T4: Крок 5 – Отримання b3 від T3
T3: Крок 6 – Обчислення b3 = min(Bн)
T3: Крок 7 – Отримання b2 від T2
T2: Крок 5 – Передача Z, MXн, MSн до T1
T2: Крок 6 – Обчислення b2 = min(Bн)
T1: Крок 4 – Обчислення b1 = min(Bн)
T1: Крок 5 – Передача b1 до T2
T2: Крок 7 – Отримання b1 від T1
T1: Крок 6 – Отримання b від T2
T2: Крок 8 – Обчислення b2 = min(b1, b2)
T2: Крок 9 – Передача b2 до T3
T2: Крок 10 – Отримання b від T3
T3: Крок 8 – Обчислення b3 = min(b2, b3)
T3: Крок 9 – Передача b3 до T4
T3: Крок 10 – Отримання b від T4
T4: Крок 6 – Обчислення b = min(b3, b4)
T4: Крок 7 – Передача b до T3
T4: Крок 8 – Обчислення a4
T3: Крок 11 – Передача b до T2
T4: Крок 9 – Передача a4 до T3
T3: Крок 12 – Обчислення a3
T2: Крок 11 – Передача b до T1
T3: Крок 13 – Отримання a4 від T4
T2: Крок 12 – Обчислення a2
T1: Крок 7 – Обчислення a1
Потік T4 закінчився
T2: Крок 13 – Отримання a1 від T1
T1: Крок 8 – Передача a1 до T2
T3: Крок 14 – Обчислення a3 = a3 + a4
T3: Крок 15 – Передача a3 до T2
Потік T1 закінчився
T2: Крок 14 – Обчислення a = a1 + a2
T2: Крок 15 – Отримання a3 від T3
Потік T3 закінчився
T2: Крок 15 – Обчислення a2 = a2 + a3
T1: Крок 10 – Виведення результату
Загальний час виконання: 5 мс
Фінальний результат: 3072
Потік T2 закінчився
```

**Етап 5  Тестування**

На тестовій системі встановлений процесор AMD Ryzen 5 4500U, який має 6 фізичних ядер і підтримує технологію SMT. Це дозволяє отримати 12 логічних ядер для виконання обчислень.

Тестування відбувається при N=3000.

При навантаженні всіх 12 логічних ядер час виконання склав 767 мс.

Після зменшенні доступних логічних ядер процесору до 1 час збільшився до 2019 мс.

**КП**: 2019 / 767 = 2.63.

## Висновки

**Етап 1: Побудова паралельного математичного алгоритму**

Розроблено паралельний алгоритм для обчислення виразу a = ((B*MX)*(Z*MS)) * min(B), який розподіляє задачу між потоками для знаходження мінімального значення масиву B та виконання матричних і векторних операцій.

Алгоритм структуровано з урахуванням розподілу обчислень на чотири потоки (T1–T4) та використання контрольних точок (CP) для синхронізації даних (b, B, Z, a).

Висновок: Алгоритм коректно розпаралелено, що дозволяє ефективно використовувати багатоядерність процесора.

**Етап 2: Розробка алгоритмів потоків**

Детально описано алгоритми для кожного потоку (T1–T4), включаючи введення даних, обчислення, передачу та приймання повідомлень.

Кожен потік виконує частину обчислень (знаходження локального мінімуму bi, обчислення ai) та синхронізується через обмін даними (B, Z, MX, MS, b, a).

Висновок: Алгоритми потоків чітко структуровані, враховують послідовність обчислень і синхронізацію, що забезпечує коректність роботи програми.

**Етап 3: Розробка структурної схеми взаємодії потоків**

Схема взаємодії потоків не представлена в запиті, але передбачається, що вона відображає передачу даних між потоками T1–T4 (B, Z, MX, MS, b, a) з урахуванням напрямів і обсягів повідомлень.

Висновок: Схема мала б ілюструвати ієрархію обміну даними та синхронізацію, що є ключовим для розуміння взаємодії потоків. Її розробка необхідна для візуалізації роботи алгоритму.

**Етап 4: Розробка та налагодження програми**

Програму (Lab5.java) розроблено з урахуванням описаних алгоритмів, включаючи "шапку" та коментарі відповідно до етапу 2.

Проведено налагодження, що забезпечило отримання правильних результатів обчислень.

За допомогою Диспетчера задач Windows проконтрольовано завантаження ядер процесора, що підтверджує використання багатоядерності.

Висновок: Програма успішно реалізована, протестована та відповідає поставленим вимогам, забезпечуючи коректні обчислення.

**Етап 5: Тестування програми**

Проведено тестування на процесорі AMD Ryzen 5 4500U (6 фізичних ядер, 12 логічних) для N=3000.

Час виконання при 12 логічних ядрах: 767 мс; при 1 ядрі: 2019 мс.

Коефіцієнт прискорення: КП = 2019 / 767 ≈ 2.63.

Висновок: Паралельний алгоритм демонструє прискорення у 2.63 раза при використанні 12 логічних ядер порівняно з одним ядром, що свідчить про ефективність розпаралелювання, хоча прискорення не є ідеальним через накладні витрати на синхронізацію та обмін даними між потоками.