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

Лабораторна робота №4

з дисципліни
«Програмне забезпечення високопродуктивних комп’ютерних систем»

на тему

«Монітори»

Виконав
Студент групи ІМ-13
Котенко Ярослав Олегович

Перевірив
доц. Корочкін О.В.

Київ 2024

Завдання

Розробити паралельну програму для обчислення в паралельної системі (ПКС СП) функції :

Варіант: 6

A= (R*MC)*MD *p + (B*Z)*E*d

Введення – виведення даних			
1	2	3	4
MC, E	MD, d	A,B,p	R,Z

Мова програмування: Java

Засоби організації взаємодії процесів : монітори.

Виконання лабораторної роботи

Етап 1. Побудова паралельного математичного алгоритму.

A = (R*MC)*MD*p + (B*Z)*E*d

1. ai = (B_H*Z_H) i = 1...P
2. a = a + ai CP: a
3. Сн = R*MCн - синхронізація по обч. C CP: R
4. Ан = C*MDн*p + a*Eн*d CP: C, a, d, p

N - розмірність вектора/матриці.

P - кількість потоків, які виконують обчислення.

H = N / P

Етап 2. Розробка алгоритмів потоків.

Задача T1	**Точки синхронізації**
1. Ведення MC, E	
2. **Сигнал** задачам T2, T3, T4 про введення MC, E	-- **S_{2-1}, S_{3-1}, S_{4-1}**
3. **Чекати** на введення даних в задачах T2, T3, T4	-- **W_{2-1}, W_{3-1} W_{4-1}**
4. Обчислення 1: a1 = (B_H*Z_H)	
5. Обчислення 2: a = a + ai	-- КД1
6. **Сигнал** задачам T2, T3, T4 про завершення обчислення 2	-- **S_{2-2}, S_{3-2}, S_{4-2}**

7. **Чекати** на завершення обчислення 2 в задачах Т2, Т3, Т4 -- W_{2-2}, W_{3-2}, W_{4-2}

8. Обчислення 3 Sн = R*MCн

9. **Сигнал** задачам Т2, Т3, Т4 про завершення обчислення 3 -- S_{2-3}, S_{3-3}, S_{4-3}

10. **Чекати** на завершення обчислення 3 в задачах Т2, Т3, Т4 -- W_{2-3}, W_{3-3}, W_{4-3}

11. **Копія** a1 = a -- КД2

12. **Копія** d1 = d -- КД3

13. **Копія** p1 = p -- КД4

14. Обчислення 4: Ан = S*MDн*p1 + a1*Ен*d1

15. **Сигнал** задачі Т3 про завершення обчислення 4 -- S_{3-4}

Задача Т2

1. Ведення MD, d

2. **Сигнал** задачам Т1, Т3, Т4 про введення MD, d -- S_{1-1}, S_{3-1}, S_{4-1}

3. **Чекати** на введення даних в задачах Т1, Т3, Т4 -- W_{1-1}, $W_{3-1}W_{4-1}$

4. Обчислення 1: a2 = (B_H*Z_H)

5. Обчислення 2: a = a + ai -- КД1

6. **Сигнал** задачам Т1, Т3, Т4 про завершення обчислення 2 -- S_{1-2}, S_{3-2}, S_{4-2}

7. **Чекати** на завершення обчислення 2 в задачах Т1, Т3, Т4 -- W_{1-2}, W_{3-2}, W_{4-2}

8. Обчислення 3 Sн = R*MCн

9. Сигнал задачам Т1, Т3, Т4 про завершення обчислення 3 -- S_{1-3}, S_{3-3}, S_{4-3}

10. **Чекати** на завершення обчислення 3 в задачах Т1, Т3, Т4 -- W_{1-3}, W_{3-3}, W_{4-3}

11. **Копія** a2 = a -- КД2

12. **Копія** d2 = d -- КД3

13. **Копія** p2 = p -- КД4

14. Обчислення 4: Ан = S*MDн*p2 + a2*Ен*d2

15. **Сигнал** задачі Т3 про завершення обчислення 4 -- S_{3-4}

Задача Т3

1. Ведення A, B, p
2. **Сигнал** задачам Т1, Т2, Т4 про введення A, B, p -- S_{1-1}, S_{2-1}, S_{4-1}
3. **Чекати** на введення даних в задачах Т1, Т2, Т4 -- W_{1-1}, $W_{2-1}W_{4-1}$
4. Обчислення 1: a3 = (B_H*Z_H)
5. Обчислення 2: a = a + ai -- КД1
6. **Сигнал** задачам Т1, Т2, Т4 про завершення обчислення 2 -- S_{1-2}, S_{2-2}, S_{4-2}
7. **Чекати** на завершення обчислення 2 в задачах Т1, Т2, Т4 -- W_{1-2}, W_{2-2}, W_{4-2}
8. Обчислення 3 Sн = R*MCн
9. **Сигнал** задачам Т1, Т2, Т4 про завершення обчислення 3 -- S_{1-3}, S_{2-3}, S_{4-3}
10. **Чекати** на завершення обчислення 3 в задачах Т1, Т2, Т4 -- W_{1-3}, W_{2-3}, W_{4-3}
11. **Копія** a3 = a -- КД2
12. **Копія** d3 = d -- КД3
13. **Копія** p3 = p -- КД4
14. Обчислення 4: Ан = S*MDн*p3 + a3*Eн*d3
15. **Чекати** на завершення обчислення 4 в задачах Т1, Т2, Т4 -- W_{1-4}, W_{2-4}, W_{4-4}
16. Виведення результату A

Задача Т4

1. Ведення R, Z
2. **Сигнал** задачам Т1, Т2, Т4 про введення R, Z -- S_{1-1}, S_{2-1}, S_{4-1}
3. **Чекати** на введення даних в задачах Т1, Т2, Т3 -- W_{1-1}, $W_{2-1}W_{3-1}$
4. Обчислення 1: a4 = (B_H*Z_H)
5. Обчислення 2: a = a + ai -- КД1
6. **Сигнал** задачам Т1, Т2, Т3 про завершення обчислення 2 -- S_{1-2}, S_{2-2}, S_{3-2}
7. **Чекати** на завершення обчислення 2 в задачах Т1, Т2, Т3 -- W_{1-2}, W_{2-2}, W_{3-2}
8. Обчислення 3 Sн =R*MCн
9. **Сигнал** задачам Т1, Т2, Т3 про завершення обчислення 3 -- S_{1-3}, S_{2-3}, S_{3-3}

10. **Чекати** на завершення обчислення 3 в задачах T1, T2, T3 -- W_{1-3}, W_{2-3}, W_{3-3}

11. **Копія** a4 = a -- КД2

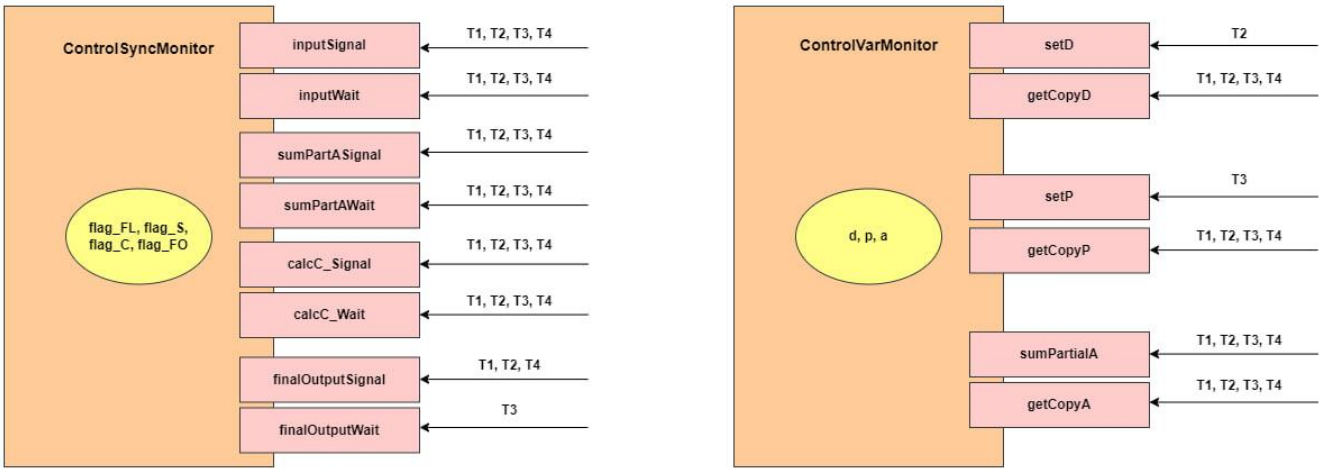
12. **Копія** d4 = d -- КД3

13. **Копія** p4 = p -- КД4

14. Обчислення 4: Ан = S*MDн*p4 + a4*Ен*d4

15. **Сигнал** задачі T3 про завершення обчислення 4 -- S_{3-4}

Етап 3. Розробка схеми взаємодії задач



Монітори:

ControlSyncMonitor – призначений для синхронізації взаємодії потоків.

Методи:

- ***inputSignal*** **–** призначений для надсилання сигналу про завершення введення даних.
- ***inputWait*** **–** призначений для очікування завершення введення даних.
- ***sumPartASignal*** **–** призначений для надсилання сигналу про завершення обчислення **a.**
- ***sumPartAWait*** **–** призначений для очікування завершення обчислення **a.**
- ***calcC_Signal*** **–** призначений для надсилання сигналу про завершення обчислення **C**.
- ***calcC_Wait*** **–** призначений для очікування завершення обчислення **C**.
- ***finalOutputSignal*** **–** призначений для надсилання сигналу про завершення обчислення $\mathbf{A_н}$.
- ***finalOutputWait*** **–** призначений для очікування завершення обчислення $\mathbf{A_н}$.

Змінні

- ***flag_FL*** **–** прапор, призначений для синхронізації введення даних.
- ***flag_S*** **-** прапор, призначений для синхронізації обчислення **a.**
- ***flag_C*** **-** прапор, призначений для синхронізації обчислення **C.**
- ***flag_FO*** **-** прапор, призначений для синхронізації виведення фінального результату**.**

ControlVarMonitor – призначений для вирішення задачі взаємного виключення. Захист спільних ресурсів : **a, d, p** .

Методи:

- ***setD*** **–** призначений для задання змінної **d**.
- ***getCopyD*** **–** призначений для отримання копії **d**.
- ***setP*** **-** призначений для задання змінної **p**.
- ***getCopyP*** **-** призначений для отримання копії **p**.
- ***sumPartialA*** **–** призначений для перезаписування значення **a** (додавання до початкового значення **a** значення $\mathbf{a_i}$).
- ***getCopyA*** **-** призначений для отримання копії **a**.

Змінні

- ***a*** **–** CP a.
- ***d*** **–** CP d.
- ***p*** **–** CP p.

Етап 4. Розроблення програми.

Код програми:

Main.java

```
// Програмне забезпечення високопродуктивних комп'ютерних систем
// Лабораторна робота 4
// Варіант 6
// A=(R*MC)*MD*p+(B*Z)*E*d
// Котенко Ярослав Олегович
// Група ІМ-13
// 30.04.2024

public class Main {
    final static int N = 16;
    final static int P = 4;

    public static void main(String[] args) throws InterruptedException {
        long start = System.currentTimeMillis();
```

```
        Data data = new Data(N, P);

        T1 t1 = new T1(data);
        T2 t2 = new T2(data);
        T3 t3 = new T3(data);
        T4 t4 = new T4(data);

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

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

        long end = System.currentTimeMillis();
        long totalTime = end - start;
        System.out.println("Executed time: " + totalTime + " ms");
    }
}
```

Data.java

```
public class Data {
    public int N;
    public int P;
    public int H;

    public Data(int n, int p) {
        this.N = n;
        this.P = p;
        this.H = n / p;
    }

    public int p;
    public int d;
    public int[] R = new int[this.N];
    public int[] B = new int[this.N];
    public int[] Z = new int[this.N];
    public int[] E = new int[this.N];
    public int[] A = new int[this.N];
    public int[] C = new int[this.N];
    public int[][] MC = new int[this.N][this.N];
    public int[][] MD = new int[this.N][this.N];

    public ControlSyncMonitor controlSyncMonitor = new ControlSyncMonitor();
    public ControlVarMonitor controlVarMonitor = new ControlVarMonitor();

    // Метод для заповнення скаляру
    public int fillScalar() {
        return 1;
    }

    // Метод для заповнення вектору
    public int[] fillVector() {
        int[] result = new int[this.N];
        for (int i = 0; i < this.N; i++) {
            result[i] = 1;
        }
```

```
    return result;
}

// Метод для заповнення матриці
public int[][] fillMatrix() {
    int[][] result = new int[this.N][this.N];
    for (int i = 0; i < this.N; i++) {
        for (int j = 0; j < this.N; j++) {
            result[i][j] = 1;
        }
    }
    return result;
}

// Метод для множення двох векторів
public int multiplyTwoVectors(int[] firstVector, int[] secondVector) {
    int length = Math.min(firstVector.length, secondVector.length);
    int result = 0;
    for (int i = 0; i < length; i++) {
        result += firstVector[i] * secondVector[i];
    }
    return result;
}

// Метод для отримання підвектору
public int[] getSubvector(int[] sourceVector, int startPos, int endPos) {
    int size = endPos - startPos;
    int[] subvector = new int[size];
    for (int i = startPos; i < endPos; i++) {
        subvector[i - startPos] = sourceVector[i];
    }
    return subvector;
}

// Метод для отримання підматриці
public int[][] getSubmatrixFromColumns(int[][] sourceMatrix, int startColumn, int endColumn) {
    int rowCount = sourceMatrix.length;
    int columnCount = endColumn - startColumn;
    int[][] submatrix = new int[rowCount][columnCount];
    for (int i = 0; i < rowCount; i++) {
        for (int j = startColumn; j < endColumn; j++) {
            submatrix[i][j - startColumn] = sourceMatrix[i][j];
        }
    }
    return submatrix;
}

// Метод для множення вектору на матрицю
public int[] multiplyVectorByMatrix(int[] vector, int[][] matrix) {
    int vectorLength = vector.length;
    int matrixColumns = matrix[0].length;
    int[] result = new int[matrixColumns];
    for (int i = 0; i < matrixColumns; i++) {
        int sum = 0;
        for (int j = 0; j < vectorLength; j++) {
            sum += vector[j] * matrix[j][i];
        }
        result[i] = sum;
    }
    return result;
}

// Метод для множення скаляру на вектор
```

```
    public static int[] scalarOnVectorMultiply(int scalar, int[] vector) {
        int[] result = new int[vector.length];
        for (int i = 0; i < vector.length; i++) {
            result[i] = scalar * vector[i];
        }
        return result;
    }

    // Метод для додавання двох векторів
    public static int[] addVectors(int[] vectorA, int[] vectorB) {
        int[] result = new int[vectorA.length];
        for (int i = 0; i < vectorA.length; i++) {
            result[i] = vectorA[i] + vectorB[i];
        }
        return result;
    }

    // Метод для обчислення кроку 4
    public int[] calculateStep4(int[][] MD_H, int[] E_H, int p, int a, int d) {
        return addVectors(scalarOnVectorMultiply(p, this.multiplyVectorByMatrix(this.C, MD_H)),
scalarOnVectorMultiply(d, scalarOnVectorMultiply(a, E_H)));
    }

    // Метод для об'єднання частинок вектора в один вектор
    public void insertSubVectorIntoVector(int[] subVector, int startIndex, int endIndex, int statusVector) {
        for (int i = startIndex; i < endIndex; i++) {
            if (statusVector == 0) {
                this.C[i] = subVector[i - startIndex];
            } else {
                this.A[i] = subVector[i - startIndex];
            }
        }
    }


}
```

ControlSyncMonitor.java

```
public class ControlSyncMonitor {

    private int flag_FL = 0;
    private int flag_S = 0;
    private int flag_C = 0;
    private int flag_FO = 0;

    public synchronized void inputWait() {
        if (flag_FL != 4) {
            try {
                this.wait();
            } catch (InterruptedException e) {
                throw new RuntimeException(e);
            }
        }
    }

    public synchronized void inputSignal() {
        flag_FL++;
        if (flag_FL == 4) {
            this.notifyAll();
        }
```

```
    }

    public synchronized void sumPartASignal() {
        flag_S++;
        if (flag_S == 4) {
            notifyAll();
        }
    }

    public synchronized void sumPartAWait() {
        if (flag_S != 4) {
            try {
                this.wait();
            } catch (InterruptedException e) {
                throw new RuntimeException(e);
            }
        }
    }

    public synchronized void calcC_Signal() {
        flag_C++;
        if (flag_C == 4) {
            notifyAll();
        }
    }

    public synchronized void calcC_Wait() {
        if (flag_C != 4) {
            try {
                this.wait();
            } catch (InterruptedException e) {
                throw new RuntimeException(e);
            }
        }
    }

    public synchronized void finalOutputSignal() {
        flag_FO++;
        if (flag_FO == 3) {
            notify();
        }
    }

    public synchronized void finalOutputWait() {
        if (flag_FO != 3) {
            try {
                wait();
            } catch (InterruptedException e) {
                throw new RuntimeException(e);
            }
        }
    }

}
```

ControlVarMonitor.java

```
public class ControlVarMonitor {
    private int d;
    private int p;
    private int a = 0;
    public synchronized void setD(int d) {
        this.d = d;
    }
    public synchronized void setP(int p) {
        this.p = p;
    }
    public synchronized int getCopyD() {
        return this.d;
    }
    public synchronized int getCopyP() {
        return this.p;
    }
    public synchronized void sumPartialA(int ai) {
        a += ai;
    }
    public synchronized int getCopyA() {
        return this.a;
    }
}
```

T1.java

```
public class T1 extends Thread {
    private final Data data;

    public T1(Data data) {
        this.data = data;
    }

    @Override
    public void run() {
        System.out.println("T1 is started");

        // Ініціалізація C ( для подальшої коректної роботи програми )
        data.C = new int[data.N];

        // Введення MC, E
        data.MC = data.fillMatrix();
        data.E = data.fillVector();

        // Сигнал про введення MC, E задачам T2,T3,T4
        data.controlSyncMonitor.inputSignal();

        // Чекати завершення введення даних у задачах T2,T3,T4
        data.controlSyncMonitor.inputWait();

        // Обчислення 1: a1=(Bн*Zн)
        int[] B_H = data.getSubvector(data.B, 0, data.H);
        int[] Z_H = data.getSubvector(data.Z, 0, data.H);

        int a1 = data.multiplyTwoVectors(B_H, Z_H);

        // КД1. Обчислення 2: a = a + a1
        data.controlVarMonitor.sumPartialA(a1);
```

```
        // Сигнал про обчислення "a" задачам T2,T3,T4
        data.controlSyncMonitor.sumPartASignal();

        // Чекати завершення обчислення "a" у задачах T2,T3,T4
        data.controlSyncMonitor.sumPartAWait();

        // Обчислення 3: Сн = R*MCн
        int[][] MC_H = data.getSubmatrixFromColumns(data.MC, 0, data.H);

        int[] C_H = data.multiplyVectorByMatrix(data.R, MC_H);

        // Об'єднання Сн в C
        data.insertSubVectorIntoVector(C_H, 0, data.H, 0);

        // Сигнал про обчислення "C" задачам T2,T3,T4
        data.controlSyncMonitor.calcC_Signal();

        // Чекати завершення обчислення "C" у задачах T2,T3,T4
        data.controlSyncMonitor.calcC_Wait();

        // КД2. Копія a1 = a
        a1 = data.controlVarMonitor.getCopyA();

        // КД3. Копія d1 = d
        int d1 = data.controlVarMonitor.getCopyD();

        // КД4. Копія p1 = p
        int p1 = data.controlVarMonitor.getCopyP();

        // Обчислення 4: Ан = S*MDн*p1+a1*Ен*d1
        int[][] MD_H = data.getSubmatrixFromColumns(data.MD, 0, data.H);
        int[] E_H = data.getSubvector(data.E, 0, data.H);

        int[] A_H = data.calculateStep4(MD_H, E_H, p1, a1, d1);

        // Об'єднання фінального результату
        data.insertSubVectorIntoVector(A_H, 0, data.H, 1);

        // Сигнал про завершення обчислення Ан потоку T3
        data.controlSyncMonitor.finalOutputSignal();

        System.out.println("T1 is finished");
    }
}
```

T2.java

```
public class T2 extends Thread {

    private final Data data;

    public T2(Data data) {
        this.data = data;
    }

    @Override
    public void run() {
        System.out.println("T2 is started");

        // Введення MD, d
        data.MD = data.fillMatrix();
```

```
        data.d = data.fillScalar();
        data.controlVarMonitor.setD(data.d);

        // Сигнал про введення MD, d задачам T1,T3,T4
        data.controlSyncMonitor.inputSignal();

        // Чекати завершення введення даних у задачах T1,T3,T4
        data.controlSyncMonitor.inputWait();

        // Обчислення 1: a2=(Bн*Zн)
        int[] B_H = data.getSubvector(data.B, data.H, data.H * 2);
        int[] Z_H = data.getSubvector(data.Z, data.H, data.H * 2);

        int a2 = data.multiplyTwoVectors(B_H, Z_H);

        // КД1. Обчислення 2: a = a + a2
        data.controlVarMonitor.sumPartialA(a2);

        // Сигнал про обчислення "a" задачам T1,T3,T4
        data.controlSyncMonitor.sumPartASignal();

        // Чекати завершення обчислення "a" у задачах T1,T3,T4
        data.controlSyncMonitor.sumPartAWait();

        // Обчислення 3: Cн = R*MCн
        int[][] MC_H = data.getSubmatrixFromColumns(data.MC, data.H, data.H * 2);

        int[] C_H = data.multiplyVectorByMatrix(data.R, MC_H);

        // Об'єднання Cн в C
        data.insertSubVectorIntoVector(C_H, data.H, data.H * 2, 0);

        // Сигнал про обчислення "C" задачам T1,T3,T4
        data.controlSyncMonitor.calcC_Signal();

        // Чекати завершення обчислення "C" у задачах T1,T3,T4
        data.controlSyncMonitor.calcC_Wait();

        // КД2. Копія a2 = a
        a2 = data.controlVarMonitor.getCopyA();

        // КД3. Копія d2 = d
        int d2 = data.controlVarMonitor.getCopyD();

        // КД4. Копія p2 = p
        int p2 = data.controlVarMonitor.getCopyP();

        // Обчислення 4: Aн = S*MDн*p2+a2*Eн*d2
        int[][] MD_H = data.getSubmatrixFromColumns(data.MD, data.H, data.H * 2);
        int[] E_H = data.getSubvector(data.E, data.H, data.H * 2);

        int[] A_H = data.calculateStep4(MD_H, E_H, p2, a2, d2);

        // Об'єднання фінального результату
        data.insertSubVectorIntoVector(A_H, data.H, data.H * 2, 1);

        // Сигнал про завершення обчислення Aн потоку T3
        data.controlSyncMonitor.finalOutputSignal();

        System.out.println("T2 is finished");
    }
}
```

T3.java

```
import java.util.Arrays;

public class T3 extends Thread {

    private final Data data;

    public T3(Data data) {
        this.data = data;
    }

    @Override
    public void run() {
        System.out.println("T3 is started");

        // Введення A, B, p
        data.A = data.fillVector();
        data.B = data.fillVector();
        data.p = data.fillScalar();
        data.controlVarMonitor.setP(data.p);

        // Сигнал про введення A, B, p задачам T1,T2,T4
        data.controlSyncMonitor.inputSignal();

        // Чекати завершення введення даних у задачах T1,T2,T4
        data.controlSyncMonitor.inputWait();

        // Обчислення 1: a3=(Bн*Zн)
        int[] B_H = data.getSubvector(data.B, data.H * 2, data.H * 3);
        int[] Z_H = data.getSubvector(data.Z, data.H * 2, data.H * 3);

        int a3 = data.multiplyTwoVectors(B_H, Z_H);

        // КД1. Обчислення 2: a = a + a3
        data.controlVarMonitor.sumPartialA(a3);

        // Сигнал про обчислення "a" задачам T1,T2,T4
        data.controlSyncMonitor.sumPartASignal();

        // Чекати завершення обчислення "a" у задачах T1,T2,T4
        data.controlSyncMonitor.sumPartAWait();

        // Обчислення 3: Cн = R*MCн
        int[][] MC_H = data.getSubmatrixFromColumns(data.MC, data.H * 2, data.H * 3);

        int[] C_H = data.multiplyVectorByMatrix(data.R, MC_H);

        // Об'єднання Cн в C
        data.insertSubVectorIntoVector(C_H, data.H * 2, data.H * 3, 0);

        // Сигнал про обчислення "C" задачам T1,T2,T4
        data.controlSyncMonitor.calcC_Signal();

        // Чекати завершення обчислення "C" у задачах T1,T2,T4
        data.controlSyncMonitor.calcC_Wait();

        // КД2. Копія a3 = a
        a3 = data.controlVarMonitor.getCopyA();

        // КД3. Копія d3 = d
        int d3 = data.controlVarMonitor.getCopyD();
```

```
        // КД4. Копія p3 = p
        int p3 = data.controlVarMonitor.getCopyP();

        // Обчислення 4: Ан = S*MDн*p3+a3*Eн*d3
        int[][] MD_H = data.getSubmatrixFromColumns(data.MD, data.H * 2, data.H * 3);
        int[] E_H = data.getSubvector(data.E, data.H * 2, data.H * 3);

        int[] A_H = data.calculateStep4(MD_H, E_H, p3, a3, d3);

        // Об'єднання фінального результату
        data.insertSubVectorIntoVector(A_H, data.H * 2, data.H * 3, 1);

        // Чекати завершення обчислення Ан в потоках T1,T2,T4
        data.controlSyncMonitor.finalOutputWait();

        // Виведення фінального результату
        System.out.println("A = " + Arrays.toString(data.A));

        System.out.println("T3 is finished");

    }
}
```

T4.java

```
public class T4 extends Thread {

    private final Data data;

    public T4(Data data) {
        this.data = data;
    }

    @Override
    public void run() {
        System.out.println("T4 is started");

        // Введення R, Z
        data.R = data.fillVector();
        data.Z = data.fillVector();

        // Сигнал про введення R, Z задачам T1,T2,T3
        data.controlSyncMonitor.inputSignal();

        // Чекати завершення введення даних у задачах T1,T2,T3
        data.controlSyncMonitor.inputWait();

        // Обчислення 1: a4=(Bн*Zн)
        int[] B_H = data.getSubvector(data.B, data.H * 3, data.N);
        int[] Z_H = data.getSubvector(data.Z, data.H * 3, data.N);

        int a4 = data.multiplyTwoVectors(B_H, Z_H);

        // КД1. Обчислення 2: a = a + a4
        data.controlVarMonitor.sumPartialA(a4);

        // Сигнал про обчислення "a" задачам T1,T2,T3
        data.controlSyncMonitor.sumPartASignal();

        // Чекати завершення обчислення "a" у задачах T1,T2,T3
        data.controlSyncMonitor.sumPartAWait();
```

```
        // Обчислення 3: Сн = R*MCн
        int[][] MC_H = data.getSubmatrixFromColumns(data.MC, data.H * 3, data.N);

        int[] C_H = data.multiplyVectorByMatrix(data.R, MC_H);

        // Об'єднання Сн в C
        data.insertSubVectorIntoVector(C_H, data.H * 3, data.N, 0);

        // Сигнал про обчислення "C" задачам T1,T2,T3
        data.controlSyncMonitor.calcC_Signal();

        // Чекати завершення обчислення "C" у задачах T1,T2,T3
        data.controlSyncMonitor.calcC_Wait();

        // КД2. Копія a4 = a
        a4 = data.controlVarMonitor.getCopyA();

        // КД3. Копія d4 = d
        int d4 = data.controlVarMonitor.getCopyD();

        // КД4. Копія p4 = p
        int p4 = data.controlVarMonitor.getCopyP();

        // Обчислення 4: Ан = S*MDн*p4+a4*Eн*d4
        int[][] MD_H = data.getSubmatrixFromColumns(data.MD, data.H * 3, data.N);
        int[] E_H = data.getSubvector(data.E, data.H * 3, data.N);

        int[] A_H = data.calculateStep4(MD_H, E_H, p4, a4, d4);

        // Об'єднання фінального результату
        data.insertSubVectorIntoVector(A_H, data.H * 3, data.N, 1);

        // Сигнал про завершення обчислення Ан потоку T3
        data.controlSyncMonitor.finalOutputSignal();

        System.out.println("T4 is finished");
    }
}
```

Результат виконання програми для N = 16.

```
T1 is started
T2 is started
T3 is started
T4 is started
T4 is finished
T1 is finished
T2 is finished
A = [272, 272, 272, 272, 272, 272, 272, 272, 272, 272, 272, 272, 272, 272, 272, 272]
T3 is finished
Executed time: 9 ms
```

Тестування:

Опис комп'ютера:

AMD Ryzen 7 4700U with Radeon Graphics

Базовая скорость:	2,00 ГГц
Сокетов:	1
Ядра:	8
Логических процессоров:	8

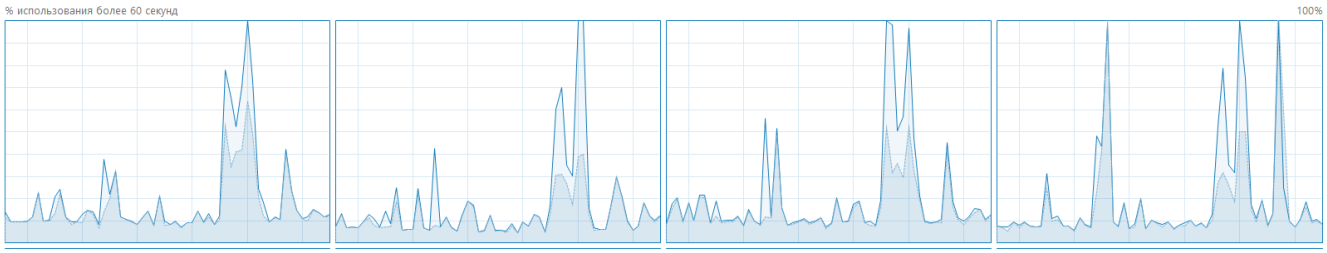
Для N = 2000

Час виконання на 1 ядрі: 2 с .

```
T1 is started
T2 is started
T3 is started
T4 is started
T1 is finished
T2 is finished
T4 is finished
A = [4002000, 4002000, 4002000, 4002000, 4002000, 4002000, 4002000, 4002000, 4002000, 4002000,
T3 is finished
Executed time: 200 ms
```

Час виконання на 4 ядрах: 0,85 с .

```
T1 is started
T2 is started
T3 is started
T4 is started
T2 is finished
T1 is finished
T4 is finished
A = [4002000, 4002000, 4002000, 4002000, 4002000, 4002000, 4002000, 4002000, 4002000, 4002000, 4002000, 4002000, 4002000,
T3 is finished
Executed time: 85 ms
```



Коефіцієнт прискорення дорівнює:

$$K_{\text{у}} = \frac{T_1}{T_4} = \frac{2}{0{,}85} = 2{,}35$$

Висновок

1) У процесі розробки було створено паралельний математичний алгоритм, який включає в себе паралельне множення вектора на матрицю, множення двох векторів, множення скаляра на вектор, а також додавання векторів. Також, було визначено спільні ресурси (СР): скаляри a, d, p та вектори C, R.
2) Були розроблені алгоритми для потоків, в яких кожен потік виконує паралельне обчислення своєї частини задачі. Також було визначено точки синхронізації та критичні ділянки КД1-КД4.
3) Була розроблена структурна схема взаємодії потоків, в якій обрані наступні засоби організації: монітор ControlSyncMonitor для синхронізації взаємодії потоків та монітор ControlVarMonitor для вирішення задачі взаємного виключення.
4) Лабораторна робота була виконана з використанням мови програмування Java. Для створення багатопотокової програми був використаний клас **Thread**. Створені два монітори: **ControlSyncMonitor** та **ControlVarMonitor**. Для створення синхронізованих методів були використані такі конструкції, як ключове слово **synchronized** у оголошенні методу, що гарантує атомарність його виконання у багатопоточному середовищі. Крім того, методи **wait**, **notify** та **notifyAll** були використані для блокування та розблокування потоків у відповідності з виконанням операцій. Також використовувався модифікатор private для забезпечення контролю доступу до спільних ресурсів.
5) Було проведене тестування, результати якого підтвердили ефективність багатопотокової програми. При значенні N = 2000 коефіцієнт прискорення склав 2,35.