Національний технічний університет України

«Київський політехнічний інститут ім. І. Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Паралельне програмування

Лабораторна робота №4

«Мова Java. Монітори»

Виконав:

студент групи ІО-21

Безщасний Р. Р.

Номер у списку групи: 2

Перевірив:

Корочкін Олександр Володимирович

Київ 2024р.

**Лабораторна робота №4.2**

**Тема:** «Мова Java. Монітори».

**Мета:** розробка програми для ПКС зі СП

**Індивідуальне завдання:**



**Етап перший:** Побудова паралельного алгоритму

1) аi = min(Bh)

2) а = min(а, аi)

3) Fh =d\*Bh +Z\*(MM\*MXh)

4) Lh = sort(Fh)

5) L2h = sort(Lh, Lh)

6) L = sort(L2h, L2h)

7) XH = Lh \* а

СР: MM, Z, d, а, a

Структура паралельної комп’ютерної системи зі спільною пам’яттю:

**Зображення, що містить текст, схема, знімок екрана, дизайн

Автоматично згенерований опис**

**Етап другий:** Розроблення паралельних потоків

Задача Т1

|  |  |  |
| --- | --- | --- |
| 1 | Очікування закінчення введення даних в T2, T4 | W2,0 W0,4 |
| 2 | Обчислення 1) a1= min(BH) | - |
| 3 | Обчислення 2) a= min(a, a1) | КД1 |
| 4 | Сигнал задачам T2, T3, T4 про обчислення 2) | S2,1 S3,1 S4,1 |
| 5 | Сигнал про обчислення 2) у задачах T2, T3, T4 | W2,1 W3,1 W4,1 |
| 6 | Копія: d1 = d | КД2 |
| 7 | Копія: a1 = a | КД3 |
| 8 | Обчислення 3) FH =d1 \*BH +Z1 \*(MM1 \*MXH) | - |
| 9 | Обчислення 4) LH = sort(FH) | - |
| 10 | Очікування закінчення обчислення 4) в задачі Т2 | W2,2 |
| 11 | Обчислення 5) L2H = sort(LH, LH) | - |
| 12 | Очікування закінчення обчислення 5 в Т3 | W3,2 |
| 13 | Обчислення 6) L = sort(L2H, L2H) | - |
| 14 | Сигнал T2, T3, T4 задачам про обчислення 6) | S2,2 S3,2 S4,2 |
| 15 | Обчислення 7) XH = LH \* a | - |
| 16 | Сигнал 4 про завершення обчислення 7) | S4,6 |

Задача Т2

|  |  |  |
| --- | --- | --- |
| 1 | Введення: B, MX | - |
| 2 | Сигнал T4 про введення даних | S2,0 |
| 3 | Очікування закінчення введення даних в Т4 | W4,0 |
| 4 | Обчислення 1) a2 = min(BH) | - |
| 5 | Обчислення 2) a = min(a, a2) | КД1 |
| 6 | Сигнал задачам T1, T3, T4 про обчислення 2) | S1,2 S3,2 S4,2 |
| 7 | Сигнал про обчислення 2) у задачах T1, T3, T4 | W1,2, W3,2, W4,2 |
| 8 | Копія: а2 = а | КД2 |
| 9 | Копія: d2 = d | КД3 |
| 10 | Обчислення 3) FH =d2 \*BH +Z2 \*(MM2 \*MXH) | - |
| 11 | Обчислення 4) LH = sort(FH) | - |
| 12 | Сигнал задачі T1 про завершення обчислення 4) | S1,3 |
| 13 | Сигнал про обчислення 6) у Т1 | W1,3 |
| 14 | Обчислення 7) XH = LH \* a | - |
| 15 | Сигнал 4 про завершення обчислення 7) | S4,6 |

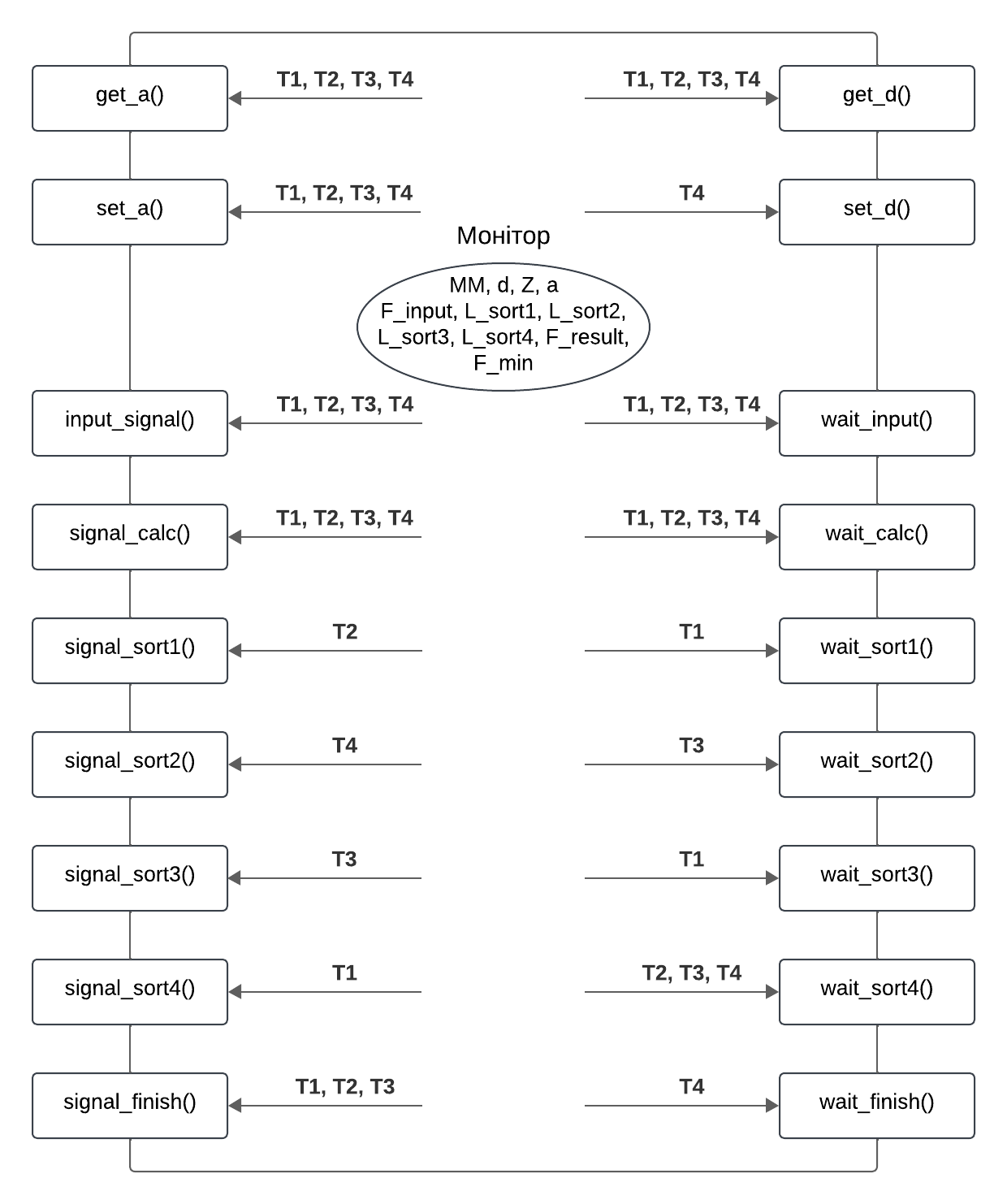
Задача Т3

|  |  |  |
| --- | --- | --- |
| 1 | Очікування закінчення введення даних в T2, T4 | W2,0 W4,0 |
| 2 | Обчислення 1) a3 = min(BH) | - |
| 3 | Обчислення 2) a = min(a, a3) | КД1 |
| 4 | Сигнал задачам T1, T2, T4 про обчислення 2) | S1,3 S2,3 S4,3 |
| 5 | Сигнал про обчислення 2) у задачах T1, T2, T4 | W1,3, W2,3, W4,3 |
| 6 | Копія: а3 = а | КД2 |
| 7 | Копія: d3 = d | КД3 |
| 8 | Обчислення 3) FH =d3 \*BH +Z3 \*(MM3 \*MXH) | - |
| 9 | Обчислення 4) LH = sort(FH) | - |
| 10 | Очікування закінчення обчислення 4) в задачі Т4 | W4,4 |
| 11 | Обчислення 5) K2H = sort(LH, LH) | - |
| 12 | Сигнал задачі T1 про обчислення 5) | S1,4 |
| 13 | Сигнал про обчислення 6) у Т1 | W1,4 |
| 14 | Обчислення 7) XH = LH \* a | - |
| 15 | Сигнал 4 про завершення обчислення 7) | S4,6 |

Задача Т4

|  |  |  |
| --- | --- | --- |
| 1 | Введення: Z, MM, d | - |
| 2 | Сигнал T2 про введення даних | S4,0 |
| 3 | Очікування закінчення введення даних в Т4 | W2,0 |
| 3 | Обчислення 1) a4 = min(BH) | - |
| 4 | Обчислення 2) a = min(a, a4) | КД1 |
| 5 | Сигнал задачам T1, T2, T3 про обчислення 2) | S1,4 S2,4 S3,4 |
| 6 | Сигнал про обчислення 2) у задачах T1, T2, T3 | W1,4 W2,4 W3,4 |
| 7 | Копія: а4 = а | КД2 |
| 8 | Копія: d4 = d | КД3 |
| 9 | Обчислення 3) FH =d4 \*BH +Z4 \*(MM4 \*MXH) | - |
| 10 | Обчислення 4) LH = sort(FH) | - |
| 11 | Сигнал задачі T3 про обчислення 4) | S3,5 |
| 12 | Сигнал про обчислення 6) у Т1 | W1,5 |
| 13 | Обчислення 7) XH = LH \* a | - |
| 14 | Чекати сигнали T1, T2, T3 про обчислення 7) | W1,6, W2,6, W3,6 |
| 15 | Виведення результату X | - |

**Етап третій:**



**Четвертий етап:** Код програми

*Lab3.java:*

/\*  
Лабораторна робота №4 Варіант 23  
𝑋 = 𝑠𝑜𝑟𝑡(𝑑 ∗ 𝐵 + 𝑍 ∗ (𝑀𝑀 ∗ 𝑀𝑋)) ∗ min(𝐵)  
T1:  
T2: B, MX  
T3:   
T4: X, MM, d, Z

Безщасний Роман ІО-21  
Дата 22.12.2024  
\*/  
  
import java.util.Scanner;  
  
public class Lab4 {  
 public static void main(String[] args) {  
  
 Monitor monitor = new Monitor();  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.print("Enter the size of the matrices:");  
 Data.*N* = scanner.nextInt();  
 T1 T1 = new T1(monitor);  
 T2 T2 = new T2(monitor);  
 T3 T3 = new T3(monitor);  
 T4 T4 = new T4(monitor);  
 long startTime = System.*currentTimeMillis*();  
 T1.start();  
 T2.start();  
 T3.start();  
 T4.start();  
 try {  
 T1.join();  
 T2.join();  
 T3.join();  
 T4.join();  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 long endTime = System.*currentTimeMillis*();  
 long duration = (endTime - startTime);  
 System.*out*.println("Execution time in milliseconds:"+duration);  
 }  
}

*Monitor.java:*

public class Monitor {  
 private int d = 0;  
 private int a = 0;  
 private int F\_input = 0;  
 private int L\_sort1 = 0;  
 private int L\_sort2 = 0;  
 private int L\_sort3 = 0;  
 private int L\_sort4 = 0;  
 private int F\_min = 0;  
 private int F\_result = 0;  
 public synchronized int get\_a(){ return a; }  
 public synchronized int get\_d(){ return d; }  
 public synchronized void set\_d(int d){ this.d = d; }  
 public synchronized void set\_a(int ai){ this.a = Math.*max*(this.a, ai); }  
   
   
 public synchronized void input\_signal(){  
 F\_input += 1;  
 if(F\_input == 2){  
 notifyAll();  
 }  
 }  
 public synchronized void wait\_input() {  
 while(F\_input != 2) {  
 try {  
 wait();  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 public synchronized void signal\_sort1(){  
 L\_sort1 += 1;  
 notifyAll();  
  
 }  
 public synchronized void wait\_sort1() {  
 while(L\_sort1 != 1) {  
 try {  
 wait();  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 public synchronized void signal\_sort2(){  
 L\_sort2 += 1;  
 notifyAll();  
 }  
 public synchronized void wait\_sort2() {  
 while(L\_sort2 != 1) {  
 try {  
 wait();  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 public synchronized void signal\_sort3(){  
 L\_sort3 += 1;  
 notifyAll();  
 }  
 public synchronized void wait\_sort3() {  
 while(L\_sort3 != 1) {  
 try {  
 wait();  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 public synchronized void signal\_sort4(){  
 L\_sort4 += 1;  
 notifyAll();  
 }  
 public synchronized void wait\_sort4() {  
 while(L\_sort4 != 1) {  
 try {  
 wait();  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
  
 public synchronized void signal\_calc(){  
 F\_min += 1;  
 if(F\_min == 4){ notifyAll();}  
 }  
  
 public synchronized void wait\_calc() {  
 while(F\_min != 4) {  
 try {  
 wait();  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
   
 public synchronized void signal\_finish(){  
 F\_result += 1;  
 if(F\_result == 3){  
 notify();  
 }  
 }  
 public synchronized void wait\_finish() {  
 while(F\_result != 3) {  
 try {  
 wait();  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
}

*Data.java:*

import java.util.Arrays;  
import java.util.stream.IntStream;  
  
public class Data {  
 public static int *N*;  
 public static int[] *B*;  
 public static int[] *Z*;  
   
 public static int[][] *MM*;  
 public static int[][] *MX*;  
   
 public static int[] *Lh1*;  
 public static int[] *Lh2*;  
 public static int[] *Lh3*;  
 public static int[] *Lh4*;  
 public static int[] *Lh2\_1*;  
 public static int[] *Lh2\_2*;  
 public static int[] *X*;  
   
 //Функція для отримання частини вектора  
 public static int[] fixArrayLength(int[] A, int start, int end) {  
 int[] res = new int[end - start];  
 IntStream.*range*(0, (end - start)).forEach(i -> res[i] = A[i]);  
  
 return res;  
 }  
   
 // Функція, що повертає певну частину матриці  
 public static int[][] fixMatrixLength(int[][] matrix, int start, int end) {  
 int[][] res = new int[matrix.length][end - start];  
 for (int i = 0; i < matrix.length; i++) {  
 int ind = 0;  
 for (int j = start; j < end; j++) {  
 res[i][ind] = matrix[i][j];  
 ind++;  
 }  
 }  
 return res;  
 }  
   
 //Функція для множення матриць  
 public static int[][] mulMatrix(int[][] vec1, int[][] vec2){  
 int[][] result= new int[vec1.length][vec2[0].length];  
 for (int i = 0; i < vec1.length; i++)  
 {  
 for (int j = 0; j < vec2[0].length; j++)  
 {  
 for (int k = 0; k < vec1[0].length; k++)  
 {  
 result[i][j] += vec1[i][k] \* vec2[k][j];  
 }  
 }  
 }  
 return result;  
 }  
  
  
 // Функція, що додає вектори  
 public static int[] addArrays(int[] A, int[] B) {  
 int[] res = new int[A.length];  
 for (int i = 0; i < A.length; i++) {  
 res[i] = A[i] + B[i];  
 }  
 return res;  
 }  
  
 // Функція, що множить вектор на матрицю  
 static public int[] mulArrayMatrix(int[] A, int[][] matrix){  
 int[] res = new int[matrix[0].length];  
 for (int i = 0; i < matrix[0].length; i++) {  
 for (int j = 0; j < A.length; j++) {  
 res[i] += A[j] \* matrix[j][i];  
 }  
 }  
 return res;  
 }  
  
 // Функція, що вставляє вектор в частину іншого вектора  
 public static void insertArrayRes(int[] A, int[] B, int start, int end) {  
 int ind = 0;  
 for (int i = start; i < end; i++) {  
 B[i] = A[ind++];  
 }  
 }  
  
 // Функція для множення скаляру на вектор  
 public static int[] mulScalArray(int scal, int[] A) {  
 IntStream.*range*(0, A.length).forEach(i -> A[i] \*= scal);  
 return A;  
 }  
  
 //Функція, що перемножує вектори між собою  
 public static int mulArrays(int[] A, int[] B) {  
 int res = 0;  
 for (int i = 0; i < A.length; i++) {  
 res += A[i] \* B[i];  
 }  
 return res;  
 }  
   
 //Функція для виведення вектора  
 public static void printvec(int[] VH){  
 for (int j : VH) {  
 System.*out*.print(j + " ");  
 }  
 System.*out*.println();  
 }  
 //Функція для початовго сортування  
 public static int[] Sort(int[] vec1) {  
 Arrays.*sort*(vec1);  
 return vec1;  
 }  
 //Функція для сортування злиттям  
 public static int[] sortTwoVecs(int[] vector1, int[] vector2) {  
 int[] result = new int[vector1.length + vector2.length];  
 for (int i = 0; i < result.length; i++) {  
 if (i < vector1.length) {  
 result[i] = vector1[i];  
 }  
 else {  
 result[i] = vector2[i-vector1.length];  
 }  
 }  
 Arrays.*sort*(result);  
 return result;  
 }  
}

*T1.java:*

public class T1 extends Thread {  
 Monitor monitor;  
 private final int start;  
 private final int end;  
 public T1(Monitor monitor) {  
   
 start = 0;  
 end = Data.*N* / 4;  
 this.monitor = monitor;  
 }  
  
 @Override  
 public void run(){  
 System.*out*.println("Start Thread 1");  
   
 // Очікування сигналів від Т2, Т4  
 monitor.wait\_input();  
   
 // КД1  
 int a1 = 1;  
 monitor.set\_a(a1);  
   
 // Сигнал задачам T2, T3, T4 про обчислення 2)  
 monitor.signal\_calc();  
 // Очікування сигналів Т2, Т3, Т4 про обчислення  
 monitor.wait\_calc();  
   
 int d1 = monitor.get\_d(); // КД2  
 a1 = monitor.get\_a(); // КД3  
 int[][] MX1 = Data.*fixMatrixLength*(Data.*MX*, start, end);  
 // Обчислення 3)  
 int[] F1 = Data.*addArrays*(Data.*mulScalArray*(d1, Data.*fixArrayLength*(Data.*B*, start, end)),  
 Data.*mulArrayMatrix*(Data.*fixArrayLength*(Data.*Z*, start, end), Data.*mulMatrix*(Data.*MM*, MX1)));  
   
 Data.*Lh1* = Data.*Sort*(F1); // Обчислення 4)  
 // Очікування закінчення обчислення 4) в задачі Т2  
 monitor.wait\_sort1();  
 // Обчислення 5) L2H = sort(LH, LH)  
 Data.*Lh2\_1* = Data.*sortTwoVecs*(Data.*Lh1*, Data.*Lh2*);  
 // Очікування закінчення обчислення 5 в Т3  
 monitor.wait\_sort3();  
 // Обчислення 6) L = sort(L2H, L2H)  
 Data.*X* = Data.*sortTwoVecs*(Data.*Lh2\_1*, Data.*Lh2\_2*);  
 // Сигнал T2, T3, T4 задачам про обчислення 6)  
 monitor.signal\_sort4();  
 //Обчислення 7) XH = LH \* a  
 Data.*insertArrayRes*(Data.*mulScalArray*(a1, Data.*X*), Data.*X*, start, end);  
 // Сигнал 4 про завершення обчислення 7)  
 monitor.signal\_finish();  
  
 System.*out*.println("Thread 1 has completed.");  
 }  
}

*T2.java:*

public class T2 extends Thread {  
 private final Monitor monitor;  
 private final int start;  
 private final int end;  
 public T2(Monitor monitor) {  
 Data.*B* = new int[Data.*N*];  
 Data.*MX* = new int[Data.*N*][Data.*N*];  
 start = Data.*N* / 4;  
 end = Data.*N* / 2;  
 this.monitor = monitor;  
 }  
 @Override  
 public void run() {  
 System.*out*.println("Start Thread 2");  
 for (int i = 0; i < Data.*N*; i++) { // Введення В, МХ  
 Data.*B*[i] = 1;  
 for (int j = 0; j < Data.*N*; j++){  
 Data.*MX*[i][j] = 1;  
 }  
 }  
 // Сигнал T4 про введення даних  
 monitor.input\_signal();  
 // Очікування закінчення введення даних в Т4  
 monitor.wait\_input();  
 // КД1  
 int a2 = 1;  
 monitor.set\_a(a2);  
 // Сигнал задачам T1, T3, T4 про обчислення 2)  
 monitor.signal\_calc();  
 // Очікування сигналів Т1, Т3, Т4 про обчислення  
 monitor.wait\_calc();  
  
 a2 = monitor.get\_a(); // КД2  
 int d2 = monitor.get\_d(); // КД3  
 int[][] MX2 = Data.*fixMatrixLength*(Data.*MX*, start, end);  
 // Обчислення 3)  
 int[] F2 = Data.*addArrays*(Data.*mulScalArray*(d2, Data.*fixArrayLength*(Data.*B*, start, end)),  
 Data.*mulArrayMatrix*(Data.*fixArrayLength*(Data.*Z*, start, end), Data.*mulMatrix*(Data.*MM*, MX2)));  
 // Обчислення 4)  
 Data.*Lh2* = Data.*Sort*(F2);  
 // Сигнал задачі T1 про завершення обчислення 4)  
 monitor.signal\_sort1();  
 // Очікування на сигнал про обчислення 6) у Т1  
 monitor.wait\_sort4();  
 // Обчислення 7) XH = LH \* a  
 Data.*insertArrayRes*(Data.*mulScalArray*(a2, Data.*X*), Data.*X*, start, end);  
 // Сигнал 4 про завершення обчислення 7)  
 monitor.signal\_finish();  
  
 System.*out*.println("Thread 2 has completed.");  
 }  
}

*T3.java:*

public class T3 extends Thread {  
 private final Monitor monitor;  
 private final int start;  
 private final int end;  
 public T3(Monitor monitor) {  
 start = Data.*N* / 2;  
 end = Data.*N* / 4 \* 3;  
 this.monitor = monitor;  
 }  
 @Override  
 public void run() {  
 System.*out*.println("Start Thread 3");  
 // Очікування закінчення введення даних в T2, T4  
 monitor.wait\_input();  
 // КД1  
 int a3 = 1;  
 monitor.set\_a(a3);  
 // Сигнал задачам T1, T2, T4 про обчислення 2)  
 monitor.signal\_calc();  
 // Очікування сигналів Т1, Т2, Т4 про обчислення  
 monitor.wait\_calc();  
  
 a3 = monitor.get\_a(); // КД 2  
 int d3 = monitor.get\_d(); // КД3  
 int[][] MX3 = Data.*fixMatrixLength*(Data.*MX*, start, end);  
 // Обчислення 3)  
 int[] F3 = Data.*addArrays*(Data.*mulScalArray*(d3, Data.*fixArrayLength*(Data.*B*, start, end)),  
 Data.*mulArrayMatrix*(Data.*fixArrayLength*(Data.*Z*, start, end), Data.*mulMatrix*(Data.*MM*, MX3)));  
 // Обчислення 4)  
 Data.*Lh3* = Data.*Sort*(F3);  
 // Очікування закінчення обчислення 4) в задачі Т4  
 monitor.wait\_sort2();  
 // Обчислення 5) K2H = sort(LH, LH)  
 Data.*Lh2\_2* = Data.*sortTwoVecs*(Data.*Lh3*, Data.*Lh4*);  
 // Сигнал задачі T1 про обчислення 5)  
 monitor.signal\_sort3();  
 // Сигнал про обчислення 6) у Т1  
 monitor.wait\_sort4();  
 // Обчислення 7) XH = LH \* a  
 Data.*insertArrayRes*(Data.*mulScalArray*(a3, Data.*X*), Data.*X*, start, end);  
 // Сигнал 4 про завершення обчислення 7)  
 monitor.signal\_finish();  
   
 System.*out*.println("Thread 3 has completed.");  
 }  
  
}

*T4.java:*

public class T4 extends Thread {  
 private final Monitor monitor;  
 private final int start;  
 private final int end;  
  
 public T4(Monitor monitor) {  
 Data.*MM* = new int[Data.*N*][Data.*N*];  
 Data.*Z* = new int[Data.*N*];  
 Data.*X* = new int[Data.*N*];  
 start = Data.*N* / 4 \* 3;  
 end = Data.*N*;  
 this.monitor = monitor;  
 }  
 @Override  
 public void run(){  
 System.*out*.println("Start Thread 4");  
 for (int i = 0; i < Data.*N*; i++) { // Введення Z, MM, d  
 Data.*X*[i] = 1;  
 Data.*Z*[i] = 1;  
 for (int j = 0; j < Data.*N*; j++){  
 Data.*MM*[i][j] = 1;  
 }  
 }  
   
 monitor.set\_d(1);  
 // Сигнал T2 про введення даних  
 monitor.input\_signal();  
 // Очікування закінчення введення даних в Т4  
 monitor.wait\_input();  
 // КД1  
 int a4 = 1;  
 monitor.set\_a(a4);  
 // Сигнал задачам T1, T2, T3 про обчислення 2)  
 monitor.signal\_calc();  
 // Сигнал про обчислення 2) у задачах T1, T2, T3  
 monitor.wait\_calc();  
  
 a4 = monitor.get\_a(); // КД2  
 int d4 = monitor.get\_d(); // КД3  
 int[][] MX4 = Data.*fixMatrixLength*(Data.*MX*, start, end);  
 // Обчислення 3)   
 int[] F4 = Data.*addArrays*(Data.*mulScalArray*(d4, Data.*fixArrayLength*(Data.*B*, start, end)),  
 Data.*mulArrayMatrix*(Data.*fixArrayLength*(Data.*Z*, start, end), Data.*mulMatrix*(Data.*MM*, MX4)));  
 // Обчислення 4)   
 Data.*Lh4* = Data.*Sort*(F4);  
 // Сигнал задачі T3 про обчислення 4)  
 monitor.signal\_sort2();  
 // Сигнал про обчислення 6) у Т1  
 monitor.wait\_sort4();  
 // Обчислення 7) XH = LH \* a  
 Data.*insertArrayRes*(Data.*mulScalArray*(a4, Data.*X*), Data.*X*, start, end);  
 // Чекати сигнали T1, T2, T3 про обчислення 7)  
 monitor.wait\_finish();  
 // Виведення результату X  
 if (Data.*N* == 4) {  
 Data.*printvec*(Data.*X*);  
 }  
  
 System.*out*.println("Thread 4 has completed.");  
 }  
}

**Результат при** *N = 4 та 4 ядрах***:**

Зображення, що містить текст, знімок екрана, Шрифт, програмне забезпечення

Автоматично згенерований опис

*Підтвердження правильності розрахунків програми:*

X =

**Результат при** *N = 1000 та 4 ядрах***:**

Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис

Зображення, що містить знімок екрана, зелений, Барвистість

Автоматично згенерований опис

**Результат при** *N = 1000 та 1 ядрі***:**

Зображення, що містить текст, знімок екрана, програмне забезпечення, Мультимедійне програмне забезпечення

Автоматично згенерований опис

Зображення, що містить знімок екрана, зелений, ряд, Барвистість

Автоматично згенерований опис

Коефіцієнт прискорення: 3846 / 1313 = 2.929

**Висновок**

Під час виконання лабораторної роботи я:

* Розробив паралельний алгоритм для математичної задачі згідно із заданим варіантом і визначив спільні змінні.
* Описав алгоритм роботи кожного потоку, виділив критичні секції та точки синхронізації.
* Створив структурну схему монітора, що забезпечує контроль взаємодії між задачами.
* Написав програму на Java, яка реалізує паралельне обчислення задачі за розробленим алгоритмом. Перевірив коректність результатів на тестовому масиві розміром 4.
* Провів тестування швидкодії програми на 1 і 4 ядрах для масивів розміром 1000, розрахувавши коефіцієнт прискорення, який становить 3.29.