

Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

**Лабораторна робота №6**

“Java. Монітори”

з дисципліни “Програмування для Паралельних Компьютерних Систем”

Виконав:

студент 3 курсу групи ІО-52

Березинець Артем

Номер заліковки: 5205

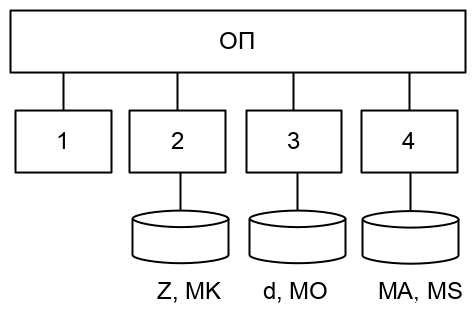
Перевірив:

Корочкін О.В.

Київ 2018 р.

**Техническое задание**

Структура паралельной компъютерной системы с общей памятью:



Выражение для подсчёта:

MA = min(Z) \* (MO\*MK) - d\*MS

Язык программирования:

Java

Средства взаимодействия задач:

Мониторы

**Выполнение работы**

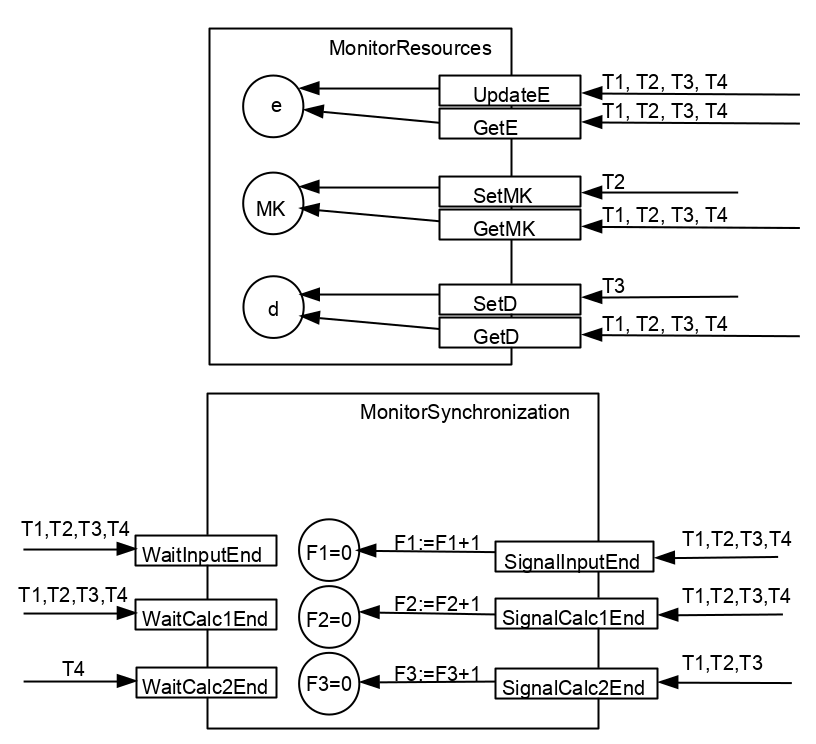
**Этап 1:** Разработка параллельного математического алгоритма

1. ei = min(ZH) , i = (1..P), P – количество потоков, P = 4
2. e = min(e; ei) , i = (1..P)
   * ОР: е
3. MAH = e \* (MOH \* MK) - d\*MSH
   * ОР: MK, d, e

**Этап 2:** Разработка алгоритмов потоков

|  |  |  |
| --- | --- | --- |
| # | Ti (1 <= i <= 4) | ТС | КУ |
| 1 | Если i=2 то: Ввод Z, MK  Если i=3 то: Ввод d, MO  Если i=4 то: Ввод MS |  |
| 2 | Если i != 1 то: сигнал Tj (j=1..4, j != i) о завершении ввода | S1..4-1 |
| 3 | Ждать сигнала о завершения ввода в Tj (j=1..4, j != i) | W1..4-1 |
| 6 | ei = min(ZH) |  |
| 7 | e = min(e, ei) | КУ |
| 8 | Сигнал Tj (j=1..4, j != i) о завершении счёта1 | S1..4-2 |
| 9 | Ждать сигнала о завершения счёта1 в Tj (j=1..4, j != i) | W1..4-2 |
| 10 | Копирование MKi := MK, di := d, ei := e | КУ |
| 11 | Счёт MAH = e \* (MOH \* MK) - d\*MSH |  |
| 12 | Если i != 4 то: сигнал T4 о завершении счёта2.  Иначе Если i = 4 то: ждать завершения счёта в Tj (1 <= j <= 3) | S4-3  W1,2,3-3 |
| 13 | Если i=4 то: Вывод MA |  |

**Этап 3:** Разработка схемы взаимодействия потоков



**Этап 4:** Разработка программы

// Lab6: Java. Monitors

// MA = min(Z) \* (MO\*MK) - d\*MS

// Artem Berezynec

// 10.05.2018

import java.util.List;

import java.util.ArrayList;

public class Main {

private static final int N = 4;

private static final int P = 4;

private static final int H = N / P;

private MonitorResources monitorResources = new MonitorResources();

private MonitorSynchronization monitorSynchronization = new MonitorSynchronization();

private Matrix MA = new Matrix(N, 0);

private Matrix MO;

private Matrix MS;

private Vector Z;

public class MyThread extends Thread {

private final int tid;

private final int start;

private final int finish;

public MyThread(int tid, int start, int finish) {

this.tid = tid;

this.start = start;

this.finish = finish;

}

@Override

public void run() {

System.out.println("Starting Thread " + tid);

// Input

switch (tid) {

case 1:

monitorSynchronization.waitInputEnd();

break;

case 2:

Z = fillVectorOnes();

final Matrix MK = fillMatrixOnes();

monitorResources.setMK(MK);

monitorSynchronization.signalInputEnd();

monitorSynchronization.waitInputEnd();

break;

case 3:

final int d = 1;

monitorResources.setD(d);

MO = fillMatrixOnes();

monitorSynchronization.signalInputEnd();

monitorSynchronization.waitInputEnd();

break;

case 4:

MS = fillMatrixOnes();

monitorSynchronization.signalInputEnd();

monitorSynchronization.waitInputEnd();

break;

default:

break;

}

// Calc1 e

int ei = Z.get(start);

for (int i = start; i < finish; i++) {

if (Z.get(i) < ei) {

ei = Z.get(i);

}

}

monitorResources.updateE(ei);

// Sync on calc1

monitorSynchronization.signalCalc1End();

monitorSynchronization.waitCalc1End();

// Copies

ei = monitorResources.getE();

final Matrix MKi = monitorResources.getMK();

final int d = monitorResources.getD();

// Calc2

for (int h = start; h < finish; h++) {

for (int i = 0; i < N; i++) {

int prod = 0;

for (int j = 0; j < N; j++) {

prod += MO.get(h, j) \* MKi.get(j, i);

}

MA.set(h, i, ei \* prod - d \* MS.get(h, i));

}

}

// Sync on calc2

switch (tid) {

case 4:

monitorSynchronization.waitCalc2End();

break;

default:

monitorSynchronization.signalCalc2End();

break;

}

// Output

if (tid == 4) {

if (N <= 8) {

outputMatrix(MA);

}

}

System.out.println("Finished Thread " + tid);

}

}

public Main() {

MyThread[] threads = new MyThread[P];

for (int i = 0; i < P; i++) {

threads[i] = new MyThread(i + 1, i \* H, (i+1) \* H);

}

for (int i = 0; i < P; i++) {

threads[i].start();

}

for (int i = 0; i < P; i++) {

try {

threads[i].join();

} catch (Exception e) {}

}

}

//-----------------------------------------------------------------------------

public static void main(String[] args) {

new Main();

}

public void outputVector(Vector vector) {

for (int i = 0; i < N; i++) {

System.out.print(vector.get(i) + " ");

}

System.out.println();

}

public void outputMatrix(Matrix matrix) {

for (int row = 0; row < N; row++) {

for (int column = 0; column < N; column++) {

System.out.print(matrix.get(row, column) + " ");

}

System.out.println();

}

System.out.println();

}

public Vector fillVectorOnes() {

return new Vector(N, 1);

}

public Matrix fillMatrixOnes() {

return new Matrix(N, 1);

}

}