НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

КАФЕДРА АВТОМАТИЗАЦІЇ СИСТЕМ ОБРОБКИ ІНФОРМАЦІЇ ТА УПРАВЛІННЯ

Лабораторна робота №3

з дисципліни **«**Паралельне програмування**»**

Виконав:

студент 3 курсу

ФІОТ гр. ІП-31

Кахерський О.І.

Перевірив:

Корочкін О. В.

Київ – 2016 р.

**Тема:** Програмування для комп’ютерних систем зі спільною пам’яттю. Мова C#

**Технічне завдання:** Розробити програму для розв’язання ПКС із СП (структура на рис. 1) математичної задачі: А = R\*max(B)\*MO+aZ\*(MT\*MK)

**Мова:** C#.

**Засоби взаємодії:** семафори, мьютекси, критичні секції, події, бар’єри

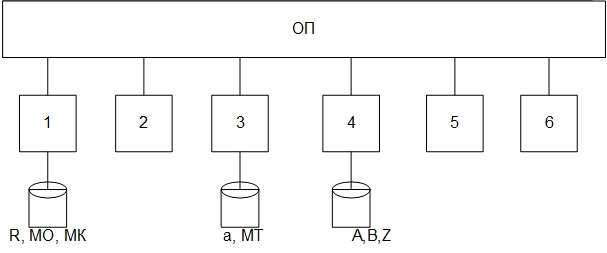


Рис.1 Структурна схема ПКС

**Виконання роботи**

**Етап 1.** Побудова паралельного алгоритму

1. bi = , i = (1..6)
2. b = max(bi,b)
3. =

Спільний ресурс: R, b, Z, , MK

**Етап 2. Розробка алогоритмів роботи кожного процесу**

(i = 2,5,6, tID – номер поточної задачі, j = i - tID)

**Задача Т1**

|  |  |  |
| --- | --- | --- |
| **№** | **Дія** | **ТС/КД** |
| 1. | Ввід МО, МК,R |  |
| 2. | Надіслати сигнал Тi, Т3, Т4 про завершення вводу |  |
| 3. | Очікувати сигнал про завершення вводу в Т4 |  |
| 4. | Очікувати сигнал про завершення вводу в Т3 |  |
| 5. | Копіювати , Z1 = Z, MK1 = MK, R1 = R | КД |
| 6. | Обчислити b1 = |  |
| 7. | Обчислити b = max(b1,b) | КД |
| 8. | Надіслати Тi,Т3,Т4 сигнал про завершення обчислення b |  |
| 9. | Очікувати сигнал про завершення обчислень b інших задачах |  |
| 10. | Копіювати b1 = b | КД |
| 11. | Обчислити = |  |
| 12. | Надіслати Т4 сигнал про завершення обчислень |  |

**Задача Тi**

|  |  |  |
| --- | --- | --- |
| **№** | **Дія** | **ТС/КД** |
| 1. | Очікувати сигнал про завершення вводу в Т1 |  |
| 2. | Очікувати сигнал про завершення вводу в Т4 |  |
| 3. | Очікувати сигнал про завершення вводу в Т3 |  |
| 4. | Копіювати , Zi = Z, MKi = MK, Ri = R | КД |
| 5. | Обчислити bi = |  |
| 6. | Обчислити b = max(bi,b) | КД |
| 7. | Надіслати сигнал іншим задачам про завершення обчислення b |  |
| 8. | Очікувати сигнал від інших задач про завершення обчислення b |  |
| 9. | Копіювати b2 = b | КД |
| 10. | Обчислити = |  |
| 11. | Надіслати Т4 сигнал про завершення обчислень |  |

**Задача Т3**

|  |  |  |
| --- | --- | --- |
| **№** | **Дія** | **ТС/КД** |
| 1. | Ввід , МТ |  |
| 2. | Надіслати сигнал Т1,Тi,Т4 про завершення вводу |  |
| 3. | Очікувати сигнал про завершення вводу в Т4 |  |
| 4. | Очікувати сигнал про завершення вводу в Т1 |  |
| 5. | Копіювати , Z3 = Z, MK3 = MK, R3 = R | КД |
| 6. | Обчислити b3 = |  |
| 7. | Обчислити b = max(b3,b) | КД |
| 8. | Надіслати сигнал іншим задачам про завершення обчислення b |  |
| 9. | Очікувати сигнал від інших задач про завершення обчислення b |  |
| 10. | Копіювати b3 = b | КД |
| 11. | Обчислити = |  |
| 12. | Надіслати Т4 сигнал про завершення обчислень |  |

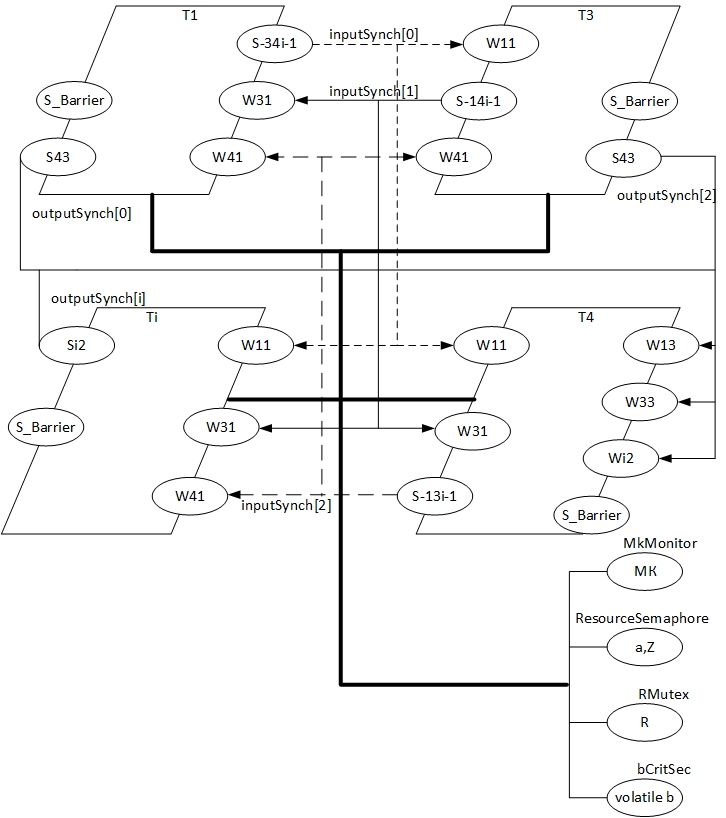
**Задача Т4**

|  |  |  |
| --- | --- | --- |
| **№** | **Дія** | **ТС/КД** |
| 1. | Ввід B,Z |  |
| 2. | Надіслати сигнал Т1,Тi,Т3 про завершення вводу |  |
| 3. | Очікувати сигнал про завершення вводу в Т1 |  |
| 4. | Очікувати сигнал про завершення вводу в Т3 |  |
| 5. | Копіювати , Z4 = Z, MK4 = MK, R4 = R | КД |
| 6. | Обчислити b4 = |  |
| 7. | Обчислити b = max(b4,b) | КД |
| 8. | Надіслати сигнал іншим задачам про завершення обчислень b |  |
| 9. | Очікувати сигнал від інших задач про завершення обчислення b |  |
| 10. | Копіювати b4 = b | КД |
| 11. | Обчислити = |  |
| 12. | Очікувати сигнал від Т1,Тi,Т3 про завершення обчислень |  |
| 13. | Вивід А |  |

**Етап 3. Розробка структурної схеми взаємодії задач**

На структурній схемі взаємодії задач уведено такі умовні позначення:

1. MkMonitor – монітор для вирішення завдання взаємного виключення при доступі до СР MK;
2. ResourceSemaphore – семафор для вирішення завдання взаємного виключення при доступі до СР α, Z;
3. RMutex – мьютекс для вирішення завдання взаємного виключення при доступі до СР R;
4. bCritSec – критична секція для вирішення завдання взаємного виключення при доступі до СР b;
5. inputSynch[] – масив об’єктів синхронізації вводу даних (0 – множинна подія, 1 – множинний семафор, 2 – множинна подія);
6. outputSynch[] – масив об’єктів синхронізації завершення обчислень математичного виразу (0 – одиночний семафор, i – одиночна подія, 2 – одиночний семафор);
7. maxCalcBarrier – бар’єр для синхронізації завершення обчислень max(B).
8. S\_Barrier – сигнал бар’єру про завершення обчислень max(B)



**Етап 4. Лістинг коду**

using System;

using System.Threading;

using System.Threading.Tasks;

namespace lab3{

class Program{

private static int[][] MO, MK, MT;

private static int[] A, B, Z, R;

private static int a, N, H, P;

private static object bCritSec = new object();

private static Semaphore resourceSemaphore;

private static Mutex RMutex;

private static WaitHandle[] inputSynch;

private static WaitHandle[] outputSynch;

private static Barrier maxCalcBarrier;

private static volatile int maxB;

static void Main(string[] args){

Console.WriteLine("lab3 started");

Console.WriteLine("input N");

N = Int32.Parse(Console.ReadLine());

Console.WriteLine("Input P");

P = Int32.Parse(Console.ReadLine());

H = N / P;

A = new int[N];

resourceSemaphore = new Semaphore(1, 1);

RMutex = new Mutex();

maxCalcBarrier = new Barrier(P);

inputSynch = new WaitHandle[3];

inputSynch[0] = new ManualResetEvent(false);

inputSynch[1] = new Semaphore(0, P - 1);

inputSynch[2] = new ManualResetEvent(false);

outputSynch = new WaitHandle[P]; //[3] = null;

outputSynch[0] = new Semaphore(0, 1);

outputSynch[1] = new AutoResetEvent(false);

outputSynch[2] = new Semaphore(0, 1);

outputSynch[3] = new AutoResetEvent(true);

for (int i = 4; i < P; i++)

outputSynch[i] = new AutoResetEvent(false);

Thread[] t = new Thread[P];

for (int i = 0; i < P-1; i++) {

t[i] = new Thread(new ThreadWorker(H \* i, H \* (i + 1), i).run);

}

t[P - 1] = new Thread(new ThreadWorker(H \* (P - 1), N, (P - 1)).run);

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

{

t[i].Start();

}

t[3].Join();

Console.WriteLine("lab3 finished");

Console.ReadLine();

}

public static void taskIFunction(int startIndex, int endIndex, int tID){

int ai = 0, bi;

int[] Zi = null, Ri = null;

int[][] MKi = null;

Console.WriteLine("task "+ tID+" started");

//Ввод данных

switch (tID){

case 0: //задача 1

//Ввод данных

MO = MatrixOperations.inputMatrix(N);

MK = MatrixOperations.inputMatrix(N);

R = MatrixOperations.inputVector(N);

Console.WriteLine("task " + tID + " finishedInput and waiting");

//Сигнал остальным потокам о завершении ввода

((ManualResetEvent)inputSynch[0]).Set();

//Ожидание ввода в 3 задаче

inputSynch[1].WaitOne();

//Ожидание ввода в 4 задаче

inputSynch[2].WaitOne();

break;

case 2: //задача 3

//Ввод данных

a = MatrixOperations.inputConstant();

MT = MatrixOperations.inputMatrix(N);

Console.WriteLine("task " + tID + " finishedInput and waiting");

//Сигнал остальным потокам о завершении ввода

((Semaphore)inputSynch[1]).Release(P-1);

//Ожидание ввода в 1 задаче

inputSynch[0].WaitOne();

//Ожидание ввода в 4 задаче

inputSynch[2].WaitOne();

break;

case 3: //задача 4

//Ввод данных

B = MatrixOperations.inputVector(N);

maxB = B[0];

Z = MatrixOperations.inputVector(N);

Console.WriteLine("task " + tID + " finishedInput and waiting");

//Сигнал остальным потокам о завершении ввода

((ManualResetEvent)inputSynch[2]).Set();

//Ожидание ввода в 1 задаче

inputSynch[0].WaitOne();

//Ожидание ввода в 3 задаче

inputSynch[1].WaitOne();

break;

default:

//Ожидание ввода в 1,3,4 задачах

WaitHandle.WaitAll(inputSynch);

break;

}

//Копирование общих ресурсов

Console.WriteLine("task " + tID + " is copying shared resources");

copySharedResources(ref ai, ref Zi, ref Ri, ref MKi);

//Вычисление max(B) в диапазоне

bi = MatrixOperations.max(B, startIndex, endIndex);

Console.WriteLine("task " + tID + " is comparing with maxB");

//Сравнивание с общим maxB

compareWithMax(bi);

//Ожидание вычислений maxB во всех других потоках (барьер)

Console.WriteLine("task " + tID + " is in barrier");

maxCalcBarrier.SignalAndWait();

//Копирование maxB

Console.WriteLine("task " + tID + " is copying maxB");

bi = copyMaxB();

Console.WriteLine("task " + tID + " is calculating equation");

//Вычисление выражения

calculateMatrixEquation(ai, bi, Zi, Ri, MKi, startIndex, endIndex);

switch (tID){

case 0: //задача 1

case 2: //задача 2

((Semaphore)outputSynch[tID]).Release(); //сигнал о завершении вычисления

break;

case 3: //задача 4

//ожидание вычисления выражения во всех потоках

WaitHandle.WaitAll(outputSynch);

if (N <= 8)

Console.WriteLine(MatrixOperations.ToString(A));

break;

default:

((AutoResetEvent)outputSynch[tID]).Set(); //сигнал о завершении вычисления

break;

}

Console.WriteLine("task " + tID + " finished");

}

private static void calculateMatrixEquation(int copy\_a, int copy\_b, int[] copyZ, int[] copyR, int[][] copyMK, int startIndex, int endIndex)

{

int[][] Buf1 = MatrixOperations.multiplyMatrixMatrix(MT, copyMK, startIndex, endIndex);

int[] vect1 = MatrixOperations.multiplyVectorMatrix(copyR, MO, copy\_b, startIndex, endIndex);

int[] vect2 = MatrixOperations.multiplyVectorMatrix(copyZ, Buf1, copy\_a, startIndex, endIndex);

MatrixOperations.addVectors(vect1, vect2, A, startIndex, endIndex);

}

private static void copySharedResources(ref int copy\_a, ref int[] copyZ, ref int[] copyR, ref int[][] copyMK) //TODO: тут мб ошибка

{

resourceSemaphore.WaitOne();

copy\_a = a;

copyZ = MatrixOperations.copyVector(Z);

resourceSemaphore.Release();

Monitor.Enter(MK);

copyMK = MatrixOperations.copyMatrix(MK);

Monitor.Exit(MK);

RMutex.WaitOne();

copyR = MatrixOperations.copyVector(R);

RMutex.ReleaseMutex();

}

private static void compareWithMax(int c){

lock(bCritSec){

if (maxB < c)

maxB = c;

}

}

private static int copyMaxB(){

lock (bCritSec)

return maxB;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab3

{

class ThreadWorker

{

private int startIndex;

private int endIndex;

private int tID;

public ThreadWorker(int start, int end, int tID)

{

this.startIndex = start;

this.endIndex = end;

this.tID = tID;

}

public void run()

{

Program.taskIFunction(startIndex, endIndex, tID);

}

}

}

using System;

using System.Text;

namespace lab3

{

public class MatrixOperations

{

/\*

\* Генерує матрицю, заповнює одиницями

\*

\* @param n розмірність

\*/

public static int[][] inputMatrix(int n)

{

int[][] result = new int[n][];

for (int i = 0; i < n; i++)

{

result[i] = new int[n];

for (int j = 0; j < n; j++)

{

result[i][j] = 1;

}

}

return result;

}

//генерирует вектор размерности b

public static int[] inputVector(int n)

{

int[] result = new int[n];

for (int i = 0; i < n; i++)

result[i] = 1;

return result;

}

public static int inputConstant()

{

return 1;

}

//суммирует векторы

public static void addVectors(int[] p1, int[] p2, int[] result, int startIndex, int endIndex)

{

for (int i = startIndex; i < endIndex; i++)

{

result[i] = p1[i] + p2[i];

}

}

//выполняет умножение матрици на матрицу

public static int[][] multiplyMatrixMatrix(int[][] param1, int[][] param2, int startIndex, int endIndex)

{

int[][] result = new int[param1.Length][];

for (int k = startIndex; k < endIndex; k++)

{

result[k] = new int[param1.Length];

for (int i = 0; i < param1.Length; i++)

{

for (int j = 0; j < param2.Length; j++)

{

result[k][i] += param1[i][j] \* param2[j][i];

}

}

}

return result;

}

//выполняет умножение вектора на константу и на матрицу

public static int[] multiplyVectorMatrix(int[] v, int[][] m, int constant, int startIndex, int endIndex)

{

int[] result = new int[v.Length];

for (int i = startIndex; i < endIndex; i++)

{

for (int j = 0; j < m.Length; j++)

{

result[i] += constant \* v[i] \* m[i][j];

}

}

return result;

}

public static int max(int[] vector, int startIndex, int endIndex)

{

int result = vector[startIndex];

for (int i = startIndex + 1; i < endIndex; i++)

if (result < vector[i])

result = vector[i];

return result;

}

public static int[] copyVector(int[] vector)

{

int[] result = new int[vector.Length];

for (int i = 0; i < result.Length; i++)

result[i] = vector[i];

return result;

}

public static int[][] copyMatrix(int[][] matrix)

{

int[][] result = new int[matrix.Length][];

for (int i = 0; i < result.Length; i++)

result[i] = copyVector(matrix[i]);

return result;

}

//выводит вектор на экран

public static String ToString(int[] vector)

{

StringBuilder b = new StringBuilder();

for (int i = 0; i < vector.Length; i++)

b.Append(vector[i] + " ");

return b.ToString();

}

}

}