Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Лабораторна робота №3

з дисципліни «Паралельне програмування -2. Програмування для паралельних систем»

Тема: **С#.** Cемафори, мютекси, критичні секції, події,бар’єри, монітори.

Виконав

студент групи ІП-31

Кобилинський Дмитро

Київ 2016

**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 = α\*max(Z)\*E+B\*(MT\*MK)

**Мова:** C#.

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

A = B(MO\*MKH) + α\*max(Z)\*E

**ОП**

5

6

3

4

2

1

A,B, α MO,Z MK,E

**Виконання роботи:**

**Етап 1. Побудова паралельного алгоритму**

1. mi = max(ZH)
2. m = max (m, mi)
3. AH = B\*(MO\*MKH) + α\*m\*E

Спільний ресурс: m, B, MO,E, α

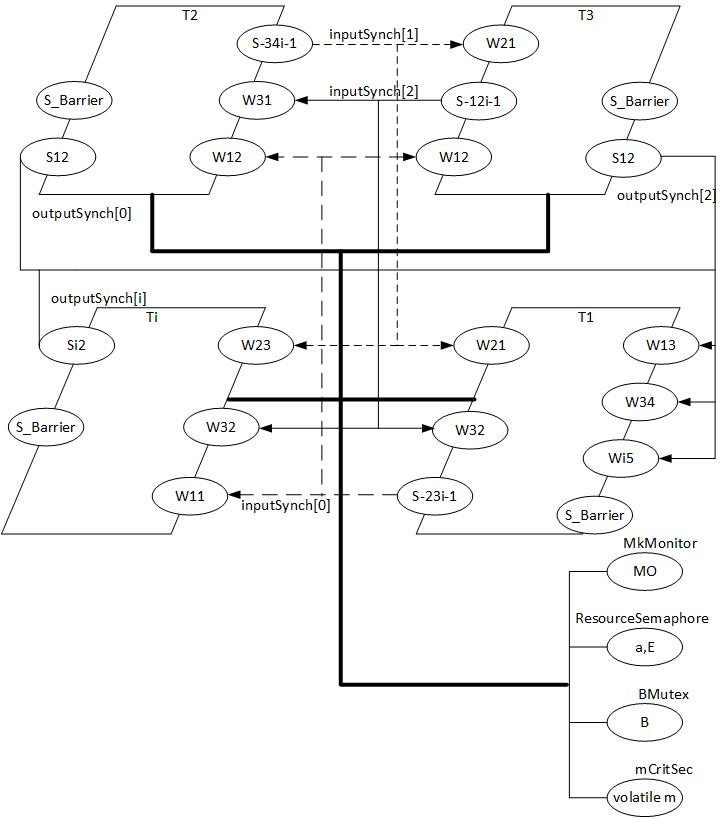
**Етап 2. Розроблення алгоритмів роботи кожного процесу**

|  |  |
| --- | --- |
| **Задача Т1** | ТС, КД |
| 1. Введення B, α |  |
| 2. Сигнал задачам Т2, Т3, Т4 про введення B , α | S2,3,4 – 1 |
| 3. Чекати на введення MO, Z у задачі Т2 | W2 – 1 |
| 4. Чекати на введення MK, E у задачі T3 | W3 – 2 |
| 5. Копіювати B1 = B, α1= α,E1=E,MO1=MO | КД |
| 6. Обчислення m1 = max(ZH) |  |
| 7. Обчислення m = max (m, m1) | КД |
| 8. Сигнал T2, T3, T4 про завершення обчислень m | S2,3,4 – 2 |
| 9. Чекати на завершення обчислень m в T2, T3, T4  10. Копіювання m1 = m | W2,3,4 – 3  КД |
| 11. Обчислення AH = B1(MO\*MKH) + α1\*m\*E1 |  |
| 12. Чекати на завершення обчислень MA в T2, T3, T4 | W2,3,4 – 4 |
| 13. Виведення A |  |
| **Задача T2** | ТС, КД |
| 1. Введення MO, Z |  |
| 2. Сигнал задачам Т1, Т3, Т4 про введення MO, Z | S1,3,4 – 1 |
| 3. Чекати на введення B, α у задачі T1 | W1 – 1 |
| 4. Чекати на введення MK,E у задачі T3 | W3 – 2 |
| 5. Копіювати B2 = B, α2= α,E2=E,MO2=MO | КД |
| 6. Обчислення m2 := max(ZH) |  |
| 7. Обчислення m: = max(m,m2) | КД |
| 8. Сигнал Т1 про про завершення обчислень m | S1 – 2 |
| 9. Чекати сигналу від T1 про завершення обчислень m | W1 –3 |
| 10. Копіювання m2: = m | КД |
| 11. Обчислення AH = B2(MO\*MKH) + α2\*m\*E2 |  |
| 12. Сигнал T1 про завершення обчислень MA | S1 – 3 |
| **Задача Т3** | ТС, КД |
| 1. Введення МК , E |  |
| 2. Сигнал задачам Т1, Т2, Т4 про введення МK, E | S1,2,4 – 1 |
| 3. Чекати на введення B, α у задачі T1 | W1 – 1 |
| 4. Чекати на введення MО,Z у задачі T2 | W3 – 2 |
| 5. Копіювати B3 = B, α3= α,E3=E,MO3=MO | КД |
| 6. Обчислення m3 := max(ZH) |  |
| 7. Обчислення m: = max(m,m3) | КД |
| 8. Сигнал Т1 про про завершення обчислень m | S1 – 2 |
| 9. Чекати сигналу від T1 про завершення обчислень m | W1 – 3 |
| 10. Копіювання m3: = m | КД |
| 11. Обчислення AH = B3(MO\*MKH) + α3\*m\*E3 |  |
| 12. Сигнал T1 про завершення обчислень MA | S1 – 3 |
| **Задача Тi** | ТС, КД |
| 1. Чекати на введення B, α у задачі T1 | W1 – 1 |
| 2. Чекати на введення MО,Z у задачі T2 | W2 – 2 |
| 3. Чекати на введення MK, E у задачі T3 | W3 – 3 |
| 4. Копіювати Bi = B, αi= α,Ei=E,MOi=MO | КД |
| 5. Обчислення mi := max(ZH) |  |
| 6. Обчислення m: = max(m,m4) | КД |
| 7. Сигнал Т1 про про завершення обчислень m | S1 – 1 |
| 8. Чекати сигналу від T1 про завершення обчислень m | W1 – 4 |
| 9. Копіювання mi: = m | КД |
| 10. Обчислення Ai = Bi(MO\*MKi) + αi\*m\*Ei |  |
| 11. Сигнал T1 про завершення обчислень MA | S1 – 2 |

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

На структурній схемі взаємодії задач уведено такі умовні позначення:

1. MoMonitor – монітор для вирішення завдання взаємного виключення при доступі до СР MO;
2. ResourceSemaphore – семафор для вирішення завдання взаємного виключення при доступі до СР α, E;
3. BMutex – мьютекс для вирішення завдання взаємного виключення при доступі до СР B;
4. mCritSec – критична секція для вирішення завдання взаємного виключення при доступі до СР m;
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
7. maxCalcBarrier – бар’єр для синхронізації завершення обчислень max(Z).
8. S\_Barrier – сигнал бар’єру про завершення обчислень max(Z)



**Етап 4. Лістинг коду**

**Program.cs**

using System;

using System.Threading;

using System.Threading.Tasks;

/\*------------------------------------------------------------------

--

-- Parallel programming --

-- Laboratory work #3 C# --

-- Task: A = B\*(MO\*MK)+alfa\*max(Z)\*E --

-- --

-- Author: Kobylynskiy Dmytro, group IP-31 --

-- Date:30.03.2016 --

-- --

------------------------------------------------------------------ \*/

namespace lab3{

class Program{

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

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

private static int a, N, H, P;

private static object zCritSec = new object();

private static Semaphore resourceSemaphore;

private static Mutex BMutex;

private static WaitHandle[] inputSynch;

private static WaitHandle[] outputSynch;

private static Barrier maxCalcBarrier;

private static volatile int maxZ;

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);

BMutex = 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];

outputSynch[0] = new AutoResetEvent(true);

outputSynch[2] = new AutoResetEvent(false);

outputSynch[1] = new AutoResetEvent(false);

outputSynch[3] = new AutoResetEvent(false);

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, mi;

int[] Zi = null, Bi = null;

int[][] MKi = null;

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

switch (tID){

case 0:

B = MatrixOperations.inputVector(N);

a = MatrixOperations.inputConstant();

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

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

inputSynch[1].WaitOne();

inputSynch[2].WaitOne();

break;

case 1:

Z = MatrixOperations.inputVector(N);

MO = MatrixOperations.inputMatrix(N);

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

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

inputSynch[0].WaitOne();

inputSynch[2].WaitOne();

break;

case 2:

E = MatrixOperations.inputVector(N);

MK = MatrixOperations.inputMatrix(N);

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

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

inputSynch[0].WaitOne();

inputSynch[1].WaitOne();

break;

default:

WaitHandle.WaitAll(inputSynch);

break;

}

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

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

mi = MatrixOperations.max(Z, startIndex, endIndex);

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

compareWithMax(mi);

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

maxCalcBarrier.SignalAndWait();

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

mi = copyMaxZ();

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

calculateMatrixEquation(ai, mi, Zi, Bi, MKi, startIndex, endIndex);

switch (tID)

{

case 2:

Console.WriteLine(tID+"LOL");((AutoResetEvent)outputSynch[tID]).Set();

break;

case 1:

Console.WriteLine(tID+"LOL");((AutoResetEvent)outputSynch[tID]).Set();

break;

case 0:

WaitHandle.WaitAll(outputSynch);

if (N <= 8)

Console.WriteLine(tID+"LOL");

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

break;

default:

((AutoResetEvent)outputSynch[tID]).Set();Console.WriteLine(tID+"LOL"); //сигнал о завершении вычисления

break;

}

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

}

private static void calculateMatrixEquation(int copy\_a, int copy\_b, int[] copyE, int[] copyB, int[][] copyMO, int startIndex, int endIndex)

{

int[][] Buf2 = MatrixOperations.multiplyMatrixMatrix(copyMO, MK, startIndex, endIndex);

int[] vect1 = MatrixOperations.multiplyVectorMatrix(copyB, Buf2, startIndex, endIndex);

int[] vect2 = MatrixOperations.multiplyVectorСonstant(copyE, copy\_b, copy\_a, startIndex, endIndex);

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

}

private static void copySharedResources(ref int copy\_a, ref int[] copyE, ref int[] copyB, ref int[][] copyMO)

{

resourceSemaphore.WaitOne();

copy\_a = a;

copyE = MatrixOperations.copyVector(E);

resourceSemaphore.Release();

Monitor.Enter(MO);

copyMO = MatrixOperations.copyMatrix(MO);

Monitor.Exit(MO);

BMutex.WaitOne();

copyB = MatrixOperations.copyVector(B);

BMutex.ReleaseMutex();

}

private static void compareWithMax(int c){

lock(zCritSec){

if (maxZ < c)

maxZ = c;

}

}

private static int copyMaxZ(){

lock (zCritSec)

return maxZ;

}

}

}

**ThreadWorker.cs**

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);

}

}

}