МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки комп’ютерної та програмної інженерії

Кафедра комп’ютеризованих систем управління

Лабораторна робота №1

З дисципліни «Паралельні та розподілені обчислення»

Тема: «Семафори та критичні ділянки»

Виконав:

Студент групи СП-425

Остапенко О.В.

Первірив:

Корочкін О.В.

КИЇВ 2019

**ЗАВДАННЯ НА РОБОТУ**

1. Структура паралельної комп’ютерної системи:



1. Математичний вираз: ***A* = min(*z*) *\* E +* max(*z*) \* (*D \* MS*)**
2. Мова програмування: **C#**
3. Засоби програмування взаємодії потоків(задач): **семафори C#, клас Semaphore**

**ВИКОНАННЯ РОБОТИ**

**Етап 1**. Побудова паралельного алгоритму

1. 𝑎 𝑖 = min (*zH*), 𝑖=1, 2, 3
2. 𝑎 = min (*𝑎*, *𝑎1*) – критична ділянка
3. *b* 𝑖 = min (*zH*), 𝑖=1, 2, 3
4. *b* = max (*b*, *b* 1) – критична ділянка
5. *AH* = *a* \* *E* + *b* \* (*D* \* *MSH*)

Спільні ресурси: *a, b, D*

**Етап 2**. Розроблення алгоритмів потоків(задач)

Т1

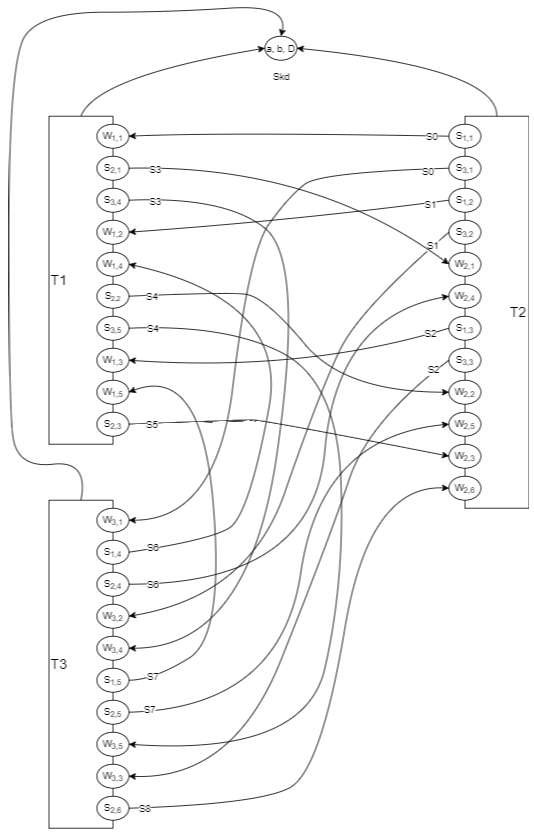
|  |  |  |
| --- | --- | --- |
| № | Дія | Точки синхронізації |
| 1 | Чекати введення даних в Т2 | W1,1 |
| 2 | Обчислення *а*1 = min(*z*H) |  |
| 3 | Обчислення *а* = min(*a*, *a*1) | Критична ділянка |
| 4 | Сигнал задачам Т2 та Т3 про завершення обчислення *а* | S2,1, S3,4 |
| 5 | Чекати на завершення обчислення *а* в задачах Т2 та Т3 | W1,2, W1,4 |
| 6 | Обчислення *b*1 = max(*z*H) |  |
| 7 | Обчислення *b* = max(*b*, *b*1) | Критична ділянка |
| 8 | Сигнал задачам Т2 та Т3 про завершення обчислення *b* | S2,2, S3,5 |
| 9 | Чекати на завершення обчислення *b* в задачах Т2 та Т3 | W1,3, W1,5 |
| 10 | Копіювати: *а*­1 = *а, b*1 = *b, D*1 = *D* | Критична ділянка |
| 11 | Обчислення *А*Н |  |
| 12 | Сигнал задачі Т2 про завершення обчислення *А*Н | S2,3 |

|  |  |  |
| --- | --- | --- |
| № | Дія | Точки синхронізації |
| 1 | Введення: *z, E, D, MS* |  |
| 2 | Сигнал задачам Т1 та Т3 завершення введення | S1,1, S3,1 |
| 3 | Обчислення *а­*2 = min( *z*H ) |  |
| 4 | Обчислення *а* = max(*a*, *a*2) | Критична ділянка |
| 5 | Сигнал задачам Т1 та Т3 про завершення обчислення *а* | S1,2, S3,2 |
| 6 | Чекати на завершення обчислення *а* в задачах Т1 та Т3 | W2,1, W2,4 |
| 7 | Обчислення *b*2 = max(*zH*) |  |
| 8 | Обчислення *b* = max(*b*, *b*2) | Критична ділянка |
| 9 | Сигнал задачам Т1 та Т3 про завершення обчислення *b* | S1,3, S3,3 |
| 10 | Чекати на завершення обчислення *b* в задачах Т1 та Т3 | W2,2, W2,5 |
| 11 | Копіювати: *а*­2 = *а, b*2 = *b, D*2 = *D* | Критична ділянка |
| 12 | Обчислення *А*Н |  |
| 13 | Чекати на завершення *А*Н в задачах Т1 та Т3 | W2,3, W2,6 |
| 14 | Виведення  *А*Н |  |

Т2

Т3

|  |  |  |
| --- | --- | --- |
| № | Дія | Точки синхронізації |
| 1 | Чекати введення даних в Т2 | W3,1 |
| 2 | Обчислення *а*3 = min(*z*H) |  |
| 3 | Обчислення *а* = min(*a*, *a*3) | Критична ділянка |
| 4 | Сигнал задачам Т1 та Т2 про завершення обчислення *а* | S1,4, S2,4 |
| 5 | Чекати на завершення обчислення *а* в задачах Т2 та Т3 | W3,2, W3,4 |
| 6 | Обчислення *b*3 = max(*z*H) |  |
| 7 | Обчислення *b* = max(*b*, *b*3) | Критична ділянка |
| 8 | Сигнал задачам Т1 та Т2 про завершення обчислення *b* | S1,5, S2,5 |
| 9 | Чекати на завершення обчислення *b* в задачах Т1 та Т2 | W3,5, W3,3 |
| 10 | Копіювати: *а*­3 = *а, b*3 = *b, D*3 = *D* | Критична ділянка |
| 11 | Обчислення *А*Н |  |
| 12 | Сигнал задачі Т2 про завершення обчислення *А*Н | S2,6 |

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

Призначення засобів синхронізації:

* Skd – критична ділянка, для керування доступом до спільного ресурсу
* S0 – для синхронізації по завершенню введення в Т2
* S1 – для синхронізації по завершенню розрахунку a в Т2
* S2 – для синхронізації по завершенню обчислень b в Т2
* S3 - для синхронізації по завершенню розрахунку a в Т1
* S4– для синхронізації по завершенню розрахунку b в Т1
* S5 - для синхронізації по завершенню обчислень AН в Т1
* S6 - для синхронізації по завершенню розрахунку a в Т3
* S7– для синхронізації по завершенню розрахунку b в Т3
* S8 - для синхронізації по завершенню обчислень AН в Т3

**Етап 4.** Розроблення програми

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

namespace Lab1\_Semaphore

{

class Program

{

public static int N = 4;

public static int[] E, Z, D, D1, D2, D3, A = new int[N];

public static int[,] MS, MS1, MS2, MS3;

public static int a = 1000, b = 0, a1 = a, a2 = a, a3 = a, b1 = b, b2 = b, b3 = b;

public static Semaphore S0 = new Semaphore(0, 2); // semaphore that controls input

public static Semaphore S1 = new Semaphore(0, 2), S3 = new Semaphore(0, 2), S6 = new Semaphore(0, 2); //semaphore for calc min(Z)

public static Semaphore S2 = new Semaphore(0, 2), S4 = new Semaphore(0, 2), S7 = new Semaphore(0, 2); //semaphore for calc max(Z)

public static Semaphore S5 = new Semaphore(0, 2), S8 = new Semaphore(0, 1); //Semaphore that controls finishing calculating the result A

public static Thread T1 = new Thread(T1Part), T2 = new Thread(T2Part), T3 = new Thread(T3Part);

public static void T1Part()

{

Console.WriteLine("T1 Started");

S0.WaitOne();

//calc min(Z2)

for (int i = Z.Length / 3; i < 2 \* Z.Length / 3; i++)

{

if (a2 > Z[i])

{

a2 = Z[i];

}

}

Thread.BeginCriticalRegion();

if (a > a2) a = a2;

Thread.EndCriticalRegion();

S3.Release(2);

S1.WaitOne();

S5.WaitOne();

//calc max(Z2)

for (int i = Z.Length / 3; i < 2 \* Z.Length / 3; i++)

{

if (b2 > Z[i])

{

b2 = Z[i];

}

}

Thread.BeginCriticalRegion();

if (b < b2) b = b2;

Thread.EndCriticalRegion();

S2.Release(2);

S4.WaitOne();

S7.WaitOne();

//CRITICAL REGION BEGIN

Thread.BeginCriticalRegion();

b1 = b;

a1 = a;

MS2 = (int[,])MS.Clone();

D2 = (int[])D.Clone();

Thread.EndCriticalRegion();

//CRITICAL REGION END

//CALCULATING

Console.WriteLine("T1 Calculate");

//CALCULATING A

int[] res1 = new int[N];

for (int i = 0; i < N / 3; i++)

{

res1[i] = a1 \* E[i];

}

int[] res2 = new int[N];

res2 = MultiplicationV(MS, D1);

int[] res3 = new int[N];

for (int i = 0; i < N / 3; i++)

{

res3[i] = b1 \* res2[i];

}

for (int i = 0; i < N / 3; i++)

{

A[i] = res3[i] + res1[i];

}

//end calc A

Console.WriteLine("T1 FINISHED");

S6.Release();

}

public static void T2Part()

{

Console.WriteLine("T2 Started");

// INIT MS

MS = new int[N, N];

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

{

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

{

MS[i, j] = 1;

}

}

// INIT B

E = new int[N];

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

{

E[i] = 1;

}

// INIT Z

Z = new int[N];

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

{

Z[i] = 1;

}

// INIT R

D = new int[N];

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

{

D[i] = 1;

}

Console.WriteLine("T2 END INPUT");

S0.Release(2);

//calc min(Z1)

for (int i = 0; i < Z.Length / 3; i++)

{

if (a2 > Z[i])

{

a2 = Z[i];

}

}

Thread.BeginCriticalRegion();

if (a > a2) a = a2;

Thread.EndCriticalRegion();

S1.Release(2);

S3.WaitOne();

S5.WaitOne();

//calc max(Z1)

for (int i = 0; i < Z.Length / 3; i++)

{

if (b2 < Z[i])

{

b2 = Z[i];

}

}

Thread.BeginCriticalRegion();

if (b < b1) b = b1;

Thread.EndCriticalRegion();

S4.Release(2);

S2.WaitOne();

S7.WaitOne();

//CALCULATING

Console.WriteLine("T2 Calculate");

//CRITICAL REGION BEGIN

Thread.BeginCriticalRegion();

b2 = b;

a2 = a;

MS1 = (int[,])MS.Clone();

D1 = (int[])D.Clone();

Thread.EndCriticalRegion();

//CRITICAL REGION END

//CALCULATING A2

int[] res1 = new int[N];

for (int i = N / 3; i < 2 \* N / 3; i++)

{

res1[i] = a2 \* E[i];

}

int[] res2 = new int[N];

res2 = MultiplicationV(MS, D1);

int[] res3 = new int[N];

for (int i = N / 3; i < 2 \* N / 3; i++)

{

res3[i] = b1 \* res2[i];

}

for (int i = N / 3; i < 2 \* N / 3; i++)

{

A[i] = res3[i] + res1[i];

}

//end calc A2

S6.WaitOne();

S8.WaitOne();

Console.WriteLine("T2 FINISHED");

//print A

Console.WriteLine("RESULT:");

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

{

Console.WriteLine(A[i]);

}

Console.WriteLine("PROGRAM FINISHED");

Console.WriteLine("---------------------------------------------");

}

public static void T3Part()

{

Console.WriteLine("T3 Started");

S0.WaitOne();

//calc min(Z3)

for (int i = 2 \* Z.Length / 3; i < Z.Length; i++)

{

if (a3 > Z[i])

{

a3 = Z[i];

}

}

Thread.BeginCriticalRegion();

if (a > a3) a = a3;

Thread.EndCriticalRegion();

S5.Release(2);

S1.WaitOne();

S3.WaitOne();

//calc max(Z3)

for (int i = 2 \* Z.Length / 3; i < Z.Length; i++)

{

if (b3 < Z[i])

{

b3 = Z[i];

}

}

Thread.BeginCriticalRegion();

if (b < b3) b = b3;

Thread.EndCriticalRegion();

S7.Release(2);

S2.WaitOne();

S4.WaitOne();

//CRITICAL REGION BEGIN

Thread.BeginCriticalRegion();

b3 = b;

a3 = a;

MS3 = (int[,])MS.Clone();

D3 = (int[])D.Clone();

Thread.EndCriticalRegion();

//CRITICAL REGION END

//CALCULATING

Console.WriteLine("T3 Calculate");

//CALCULATING A3

int[] res1 = new int[N];

for (int i = 2 \* N / 3; i < N; i++)

{

res1[i] = a3 \* E[i];

}

int[] res2 = new int[N];

res2 = MultiplicationV(MS, D3);

int[] res3 = new int[N];

for (int i = 2 \* N / 3; i < N; i++)

{

res3[i] = b1 \* res2[i];

}

for (int i = 2 \* N / 3; i < N; i++)

{

A[i] = res3[i] + res1[i];

}

//end calc A3

Console.WriteLine("T3 FINISHED");

S8.Release();

}

static int[] MultiplicationV(int[,] a, int[] b)

{

int[] r = new int[a.GetLength(0)];

for (int i = 0; i < a.GetLength(0); i++)

{

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

{

r[i] += a[i, j] \* b[j];

}

}

return r;

}

static void Main(string[] args)

{

Console.WriteLine("PROGRAM STARTED");

T1.Start();

T2.Start();

T3.Start();

Console.ReadKey();

}

}

}