НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

КАФЕДРА АВТОМАТИЗАЦІЇ СИСТЕМ ОБРОБКИ ІНФОРМАЦІЇ ТА УПРАВЛІННЯ

Лабораторна робота №2

з дісципліни **«**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. Бібліотека Win32

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

**Бібліотека:** Win32.

**Засоби взаємодії:** семафори, мьютекси, критичні секції, події

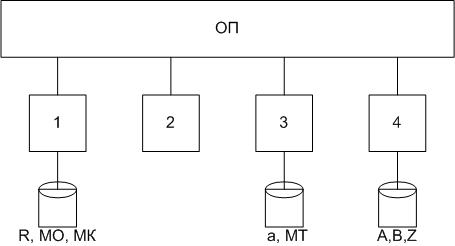


Рис.1 Структурна схема ПКС

**Виконання роботи**

**Етап 1.** Побудова паралельного алгоритму

1. bi = , i = {1..4}
2. b = max(bi,b)
3. =

Спільний ресурс: R, b, Z, , MK

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

**Задача Т1**

|  |  |  |
| --- | --- | --- |
| **№** | **Дія** | **ТС/КД** |
| 1. | Ввід МО, МК,R |  |
| 2. | Надіслати сигнал Т2, 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. | Очікувати сигнал про завершення обчислень b в Т2,Т3,Т4 |  |
| 8. | Надіслати Т2,Т3,Т4 сигнал про завершення обчислення b |  |
| 9. | Копіювати b1 = b | КД |
| 10. | Обчислити = |  |
| 11. | Надіслати Т4 сигнал про завершення обчислень |  |

**Задача Т2**

|  |  |  |
| --- | --- | --- |
| **№** | **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 , Z2 = Z, MK2 = MK, R2 = R | КД |
| 5. | Обчислити b2 = |  |
| 6. | Обчислити b = max(b2,b) | КД |
| 7. | Надіслати сигнал Т1 про завершення обчислень b |  |
| 8. | Очікувати сигнал від Т1 про завершення обчислень 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,Т2,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 Т1 про завершення обчислень b |  |
| 9. | Очікувати сигнал від Т1 про завершення обчислень 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,Т2,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 Т1 про завершення обчислень b |  |
| 9. | Очікувати сигнал від Т1 про завершення обчислень b |  |
| 10. | Копіювати b4 = b | КД |
| 11. | Обчислити = |  |
| 12. | Очікувати сигнал від Т1,Т2,Т3 про завершення обчислень |  |
| 13. | Вивід А |  |

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

На структурній схемі взаємодії задач уведено такі умовні позначення:

1. evSynchInput[] – масив примітивів для вирішення завдання синхронізації вводу даних (елементи 1-2 – сигнали, 0 – семафор );
2. maxNumEvents[] - масив примітивів для вирішення завдання синхронізації обчислення max(B) в Т2-4 та 1 (0-2 – сигнали);
3. Ev1\_all\_2 – сигнал Т2-4 від Т1 про остаточне завершення обчислень max(B);
4. endTaskSignals[3] – сигнал Т4 від інших задач про завершення обчислень виразу в них (елементи 0-1 – події, 2 – семафор);
5. CrSec – критична секція для копіювання спільних ресурсів: a,Z,R,MT;
6. Mut - мьютекс для копіювання спільного ресурсу b;
7. semSharedResource – семафор для вирішення завдання взаємного виключення при записі до спільного ресурсу b.

**Етап 4. Лістинг коду**

#include "matrixOperations.h"

#include<windows.h>

#include<iostream>

using namespace std;

typedef int\* vector;

typedef int\*\* matrix;

int a,b,N,H,P;

vector A,B,Z,R;

matrix MO,MK,MT;

CRITICAL\_SECTION CrSec;

HANDLE semSharedResource, mut, //доступ до СР

maxNumEvents[3],

evSynchInput[3], endTaskSignals[3],

ev1\_all\_2;

void calculateEquation(int startIndex, int endIndex, int c1, int c2, vector v1, vector v2, matrix matr1){

//multConstVect(c1,v1, N);

//1multConstVect(c2,v2, N);

vector bufVect1 = multVectMatr(v1,MO,startIndex,endIndex, N); //Buf1= R\*MO

multConstVect(c1, bufVect1, startIndex, endIndex); //max(B)\*Buf1

matrix bufMatr = multMatr(MT,matr1,startIndex,endIndex, N);

vector bufVect2 = multVectMatr(v2,bufMatr,startIndex,endIndex, N);

addVectors(bufVect1,bufVect2,A,startIndex,endIndex);

}

void copySharedResources(int &constant, vector &vect1, vector &vect2, matrix &matr){

EnterCriticalSection(&CrSec);

constant = a;

vect1 = copyVector(Z, N);

vect2 = copyVector(R, N);

matr = copyMatrix(MT, N);

LeaveCriticalSection(&CrSec);

}

int compare(int a, int b){

if (a >= b)

return a;

else return b;

}

void compareWithMax(int c){

WaitForSingleObject(semSharedResource,INFINITE);

b = compare(b,c);

ReleaseSemaphore(semSharedResource,1,NULL);

}

void task1(){

int a1, b1;

vector Z1, R1;

matrix MK1 = NULL;

Z1 = NULL;

R1 = NULL;

cout << "t1 started" << endl;

//1.

MO = generateMatrix(N);

MK = generateMatrix(N);

R = generateVector(N);

//2.

ReleaseSemaphore(evSynchInput[0], 3, NULL);

//3.

WaitForSingleObject(evSynchInput[2], INFINITE);

//4.

WaitForSingleObject(evSynchInput[1], INFINITE);

//5.

copySharedResources(a1, Z1, R1, MK1);

//6.

b1 = maxInVector(B, 0, H);

//7.

WaitForMultipleObjects(3, maxNumEvents, TRUE, INFINITE);

b = compare(b, b1);

//8.

SetEvent(ev1\_all\_2);

//9.

WaitForSingleObject(mut, INFINITE);

b1 = b;

ReleaseMutex(mut);

//10.

calculateEquation(0, H, b1, a1, R1, Z1, MK1);

//11.

cout << "t1 finished" << endl;

SetEvent(endTaskSignals[0]);

}

void task2(){

int a2, b2;

vector Z2 = NULL, R2 = NULL;

matrix MK2 = NULL;

cout << "t2 started" << endl;

//1-3

WaitForMultipleObjects(3, evSynchInput, TRUE, INFINITE);

//4.

copySharedResources(a2, Z2, R2, MK2);

//5.

b2 = maxInVector(B, H, 2 \* H);

//6.

compareWithMax(b2);

//7.

SetEvent(maxNumEvents[0]);

//8.

WaitForSingleObject(ev1\_all\_2, INFINITE);

//9.

WaitForSingleObject(mut, INFINITE);

b2 = b;

ReleaseMutex(mut);

//10. task10

calculateEquation(H, 2 \* H, b2, a2, R2, Z2, MK2);

//11.

cout << "t2 finished" << endl;

SetEvent(endTaskSignals[1]);

}

void task3(){

cout << "t3 started" << endl;

int a3, b3;

vector Z3 = NULL, R3 = NULL;

matrix MK3 = NULL;

//1.

a = generateConstant();

MT = generateMatrix(N);

//2.

SetEvent(evSynchInput[1]);

//3.

WaitForSingleObject(evSynchInput[0], INFINITE);

//4.

WaitForSingleObject(evSynchInput[2], INFINITE);

//5.

copySharedResources(a3, Z3, R3, MK3);

//6.

b3 = maxInVector(B, 2 \* H, 3 \* H);

//7.

compareWithMax(b3);

//8.

SetEvent(maxNumEvents[1]);

//9.

WaitForSingleObject(ev1\_all\_2, INFINITE);

//10.

WaitForSingleObject(mut, INFINITE);

b3 = b;

ReleaseMutex(mut);

//11.

calculateEquation(2 \* H, 3 \* H, b3, a3, R3, Z3, MK3);

//12.

cout << "t3 finished" << endl;

ReleaseSemaphore(endTaskSignals[2], 1, NULL);

}

void task4(){

int a4, b4;

vector Z4 = NULL, R4 = NULL;

matrix MK4 = NULL;

cout << "t4 started" << endl;

//1.

B = generateVector(N);

Z = generateVector(N);

//2.

SetEvent(evSynchInput[2]);

//3.

WaitForSingleObject(evSynchInput[0], INFINITE);

//4.

WaitForSingleObject(evSynchInput[1], INFINITE);

//5.

copySharedResources(a4, Z4, R4, MK4);

//6.

b4 = maxInVector(B, 3 \* H, 4 \* H);

//7.

compareWithMax(b4);

//8.

SetEvent(maxNumEvents[2]);

//9.

WaitForSingleObject(ev1\_all\_2, INFINITE);

//10.

WaitForSingleObject(mut, INFINITE);

b4 = b;

ReleaseMutex(mut);

//11.

calculateEquation(3 \* H, N, b4, a4, R4, Z4, MK4);

//12.

WaitForMultipleObjects(3, endTaskSignals, TRUE, INFINITE);

//13.

if (N <= 8)

printVector(A,N);

cout << "t4 finished" << endl;

}

int main(int argc, char const \*argv[])

{

cout << "Lab2 started" << endl;

N = 8;

P = 4;

cin >> N;

H = N / 4;

A = new int[N];

InitializeCriticalSection(&CrSec);

mut = CreateMutex(NULL, FALSE, NULL); //вирішення завдання взаємного виключення при доступі до спільних ресурсів

semSharedResource = CreateSemaphore(NULL, 1, 1, NULL);

maxNumEvents[0] = CreateEvent(NULL, FALSE, FALSE, NULL); //сигнали Т1 про завершення обчислень b в інших потоках

maxNumEvents[1] = CreateEvent(NULL, FALSE, FALSE, NULL);

maxNumEvents[2] = CreateEvent(NULL, FALSE, FALSE, NULL);

evSynchInput[0] = CreateSemaphore(NULL, 0, 3, NULL); //сигнали про завершення вводу

evSynchInput[1] = CreateEvent(NULL, TRUE, FALSE, NULL);

evSynchInput[2] = CreateEvent(NULL, TRUE, FALSE, NULL);

ev1\_all\_2 = CreateEvent(NULL, TRUE, FALSE, NULL); //сигнал всім потокам від T1 про остаточне завершення обчислення max(B)

endTaskSignals[0] = CreateEvent(NULL, FALSE, FALSE, NULL); //сигнали Т4 від інших потоків про завершення обчислення виразу.

endTaskSignals[1] = CreateEvent(NULL, FALSE, FALSE, NULL);

endTaskSignals[2] = CreateSemaphore(NULL, 0, 1, NULL);

DWORD Tid1, Tid2, Tid3, Tid4;

HANDLE threads[] = {

CreateThread(NULL, NULL, (LPTHREAD\_START\_ROUTINE)task1, NULL, NULL, &Tid1),

CreateThread(NULL, NULL, (LPTHREAD\_START\_ROUTINE)task2, NULL, NULL, &Tid2),

CreateThread(NULL, NULL, (LPTHREAD\_START\_ROUTINE)task3, NULL, NULL, &Tid3),

CreateThread(NULL, NULL, (LPTHREAD\_START\_ROUTINE)task4, NULL, NULL, &Tid4)

};

WaitForMultipleObjects(4, threads, true, INFINITE);

CloseHandle(threads[0]);

CloseHandle(threads[1]);

CloseHandle(threads[2]);

CloseHandle(threads[3]);

cout << "Lab2 ended" << endl;

char c;

cin >> c;

return 0;

}