Лабораторная работа №2

по курсу:

«Паралельные и распределённые вычисления»

Тема: «Win32»

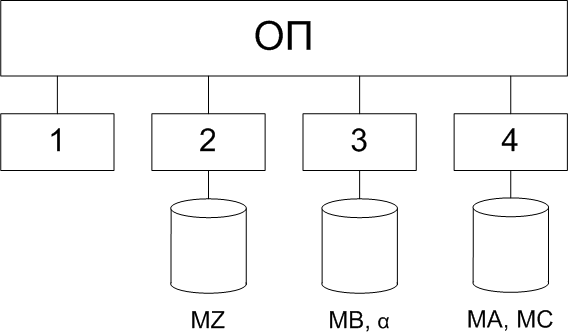
Выполнил: студент группы ИВ-83

НТУУ «КПИ» ФИВТ

Гаврилюк Ольга

**Техническое задание**

MA = MB + α \* (MC \* MZ)



**Этап 1. Построение параллельного алгоритма**

Вычисление данного матричного уравнения можно разбить на шаги:

1. MAH = MBH + α (MC \* MZH)

**Общие ресурсы:** MC, α

**Этап 2. Разработка алгоритмов процессов (задач)**

**№ Т1 ТС/КУ**

1. Ждать введения в T2, T3, T4 W2.1, W3.1, W4.1
2. Копирование α1 := α КУ
3. Копирование MC1 := MC КУ
4. Cчёт MAH = MBH + α1 (MC1 \* MZH)
5. Сигнал задаче Т4 о завершении счёта. S4.1

**№ Т2 ТС/КУ**

1. Ввод МZ
2. Сигнал задачам Т1, Т3, Т4 о завершении ввода МZ S1.1, S3.1, S4.1
3. Ждать введения в Т3, Т4 W3.1, W4.1
4. Копирование α2 := α КУ
5. Копирование MC2 := MC КУ
6. Cчёт MAH = MBH + α2 (MC2 \* MZH)
7. Сигнал задаче Т4 о завершении счёта. S4.2

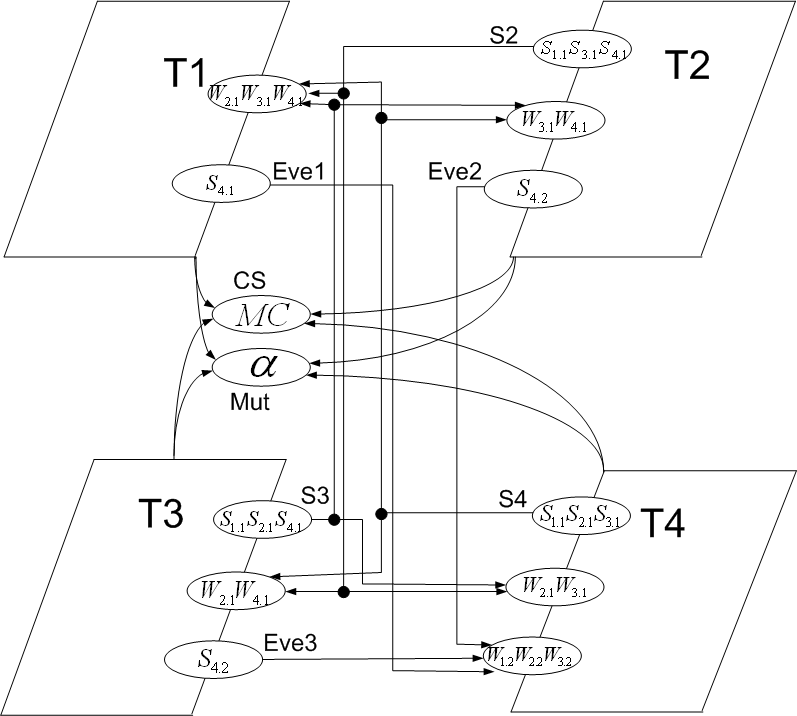
**№ Т3 ТС/КУ**

1. Ввод МB, α
2. Сигнал задачам Т1, Т2, Т4 о завершении ввода S1.1, S2.1, S4.1
3. Ждать введения в Т2, Т4 W2.1, W4.1
4. Копирование α3 := α КУ
5. Копирование MC3 := MC КУ
6. Cчёт MAH = MBH + α3 (MC3 \* MZH)
7. Сигнал задаче Т4 о завершении счёта. S4.2

**№ Т4 ТС/КУ**

1. Ввод MC
2. Сигнал задачам Т1, Т2, Т3 о завершении ввод S1.1, S2.1, S3.1
3. Ждать введения в Т2, Т3 W2.1, W3.1
4. Копирование α4 := α КУ
5. Копирование MC4 := MC КУ
6. Cчёт MAH = MBH + α4 (MC4 \* MZH)
7. Ждать завершения счёта MA в задачах Т1, Т2, Т3 W1.2, W2.2, W3.2
8. Вывод MA.

**Этап 3. Разработка схемы взаимодействия задач**

****

**Этап 4. Разработка программы**

/\*\*

\* @author IO-83

\* Paralel and distributed computing

\* Laboratory work #2. Win32

\* 21.03.2011

\*/

#include "stdafx.h"

#include "windows.h"

#include <iostream>

#include <fstream>

#include <algorithm>

#include <string>

using namespace std;

int N = 4000;

const int P = 4;

const int H = N/P;

//========================================================

int alfa;

int \*\* MA = new int\*[N];

int \*\* MB = new int\*[N];

int \*\* MC = new int\*[N];

int \*\* MZ = new int\*[N];

//Semaphores

HANDLE S2, S3, S4;

//Critical section

CRITICAL\_SECTION CS;

//Mutex

HANDLE Mut;

//Events

HANDLE Eve1, Eve2, Eve3;

static void inputVector(int\* vec) {

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

vec[j] = 1;

}

}

static void inputMatrix(int\*\* matr) {

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

{

matr[i] = new int[N];

}

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

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

matr[i][j] = 1;

}

}

}

//========================TASK 1===========================

void T1()

{

int alfa1, sum;

int\*\* MC1 = new int\*[N];

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

{

MC1[i] = new int[N];

}

int task = 1; //Task number

cout << "T1: started" << endl;

//1. Ждать введения в T2, T3, T4 W2.1, W3.1, W4.1

WaitForSingleObject(S2, INFINITE); //W2.1

WaitForSingleObject(S3, INFINITE); //W3.1

WaitForSingleObject(S4, INFINITE); //W4.1

//2. Копирование α1 := α (МЮТЕКС)

WaitForSingleObject(Mut, INFINITE);

alfa1 = alfa;

ReleaseMutex(Mut);

//3. Копирование MC1 := MC (КРИТИЧЕСКАЯ СЕКЦИЯ)

EnterCriticalSection(&CS);

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

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

MC1[i][j] = MC[i][j];

LeaveCriticalSection(&CS);

//4. Cчёт MAH = MBH + α1 (MC1 \* MZH)

for (int z = (task - 1) \* H; z < task \* H; z++){

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

sum = 0;

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

sum = sum + MZ[z][j] \* MC1[j][i]; //(MC1 \* MZH)

}

sum = sum \* alfa1; //α1 (MC1 \* MZH)

MA[z][i] = sum + MB[z][i]; //MBH + α1 (MC1 \* MZH)

}

}

//5. Сигнал задаче Т4 о завершении счёта.

SetEvent(Eve1); //S4.1

cout << "T1 finished" << endl;

}

//=============================TASK 2=====================

void T2()

{

int alfa2, sum;

int\*\* MC2 = NULL;

MC2 = new int\*[N];

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

{

MC2[i] = new int[N];

}

int task = 2; //Task number

cout << "T2: started" << endl;

//1. Ввод МZ

inputMatrix(MZ);

//2. Сигнал задачам Т1, Т3, Т4 о завершении ввода МZ

ReleaseSemaphore(S2, 3, NULL); //S1.1, S3.1, S4.1

//3. Ждать введения в Т3, Т4

WaitForSingleObject(S3, INFINITE); //W3.1

WaitForSingleObject(S4, INFINITE); //W4.1

//4. Копирование α2 := α (МЮТЕКС)

WaitForSingleObject(Mut, INFINITE);

alfa2 = alfa;

ReleaseMutex(Mut);

//5. Копирование MC2 := MC (КРИТИЧЕСКАЯ СЕКЦИЯ)

EnterCriticalSection(&CS);

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

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

MC2[i][j] = MC[i][j];

LeaveCriticalSection(&CS);

//6. Cчёт MAH = MBH + α2 (MC2 \* MZH)

for (int z = (task - 1) \* H; z < task \* H; z++){

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

sum = 0;

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

sum = sum + MZ[z][j] \* MC2[j][i]; //(MC2 \* MZH)

}

sum = sum \* alfa2; //α1 (MC1 \* MZH)

MA[z][i] = sum + MB[z][i]; //MBH + α2 (MC2 \* MZH)

}

}

//7. Сигнал задаче Т4 о завершении счёта.

SetEvent(Eve2); //S4.2

cout << "T2 finished" << endl;

}

//=============================TASK 3=====================

void T3()

{

int alfa3, sum;

int\*\* MC3 = NULL;

MC3 = new int\*[N];

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

{

MC3[i] = new int[N];

}

int task = 3; //Task number

cout << "T3: started" << endl;

//1. Ввод МB, α

inputMatrix(MB);

alfa = 1;

//2. Сигнал задачам Т1, Т2, Т4 о завершении ввода МZ

ReleaseSemaphore(S3, 3, NULL); //S1.1, S2.1, S4.1

//3. Ждать введения в Т2, Т4

WaitForSingleObject(S2, INFINITE); //W2.1

WaitForSingleObject(S4, INFINITE); //W4.1

//4. Копирование α3 := α (МЮТЕКС)

WaitForSingleObject(Mut, INFINITE);

alfa3 = alfa;

ReleaseMutex(Mut);

//5. Копирование MC3 := MC (КРИТИЧЕСКАЯ СЕКЦИЯ)

EnterCriticalSection(&CS);

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

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

MC3[i][j] = MC[i][j];

LeaveCriticalSection(&CS);

//6. Cчёт MAH = MBH + α3 (MC2 \* MZH)

for (int z = (task - 1) \* H; z < task \* H; z++){

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

sum = 0;

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

sum = sum + MZ[z][j] \* MC3[j][i]; //(MC3 \* MZH)

}

sum = sum \* alfa3; //α3 (MC1 \* MZH)

MA[z][i] = sum + MB[z][i]; //MBH + α3 (MC3 \* MZH)

}

}

//7. Сигнал задаче Т4 о завершении счёта.

SetEvent(Eve3); //S4.2

cout << "T3 finished" << endl;

}

//=============================TASK 4======================

void T4()

{

int alfa4, sum;

int\*\* MC4 = new int\*[N];

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

{

MC4[i] = new int[N];

}

int task = 4; //Task number

cout << "T4: started" << endl;

//1. Ввод MC

inputMatrix(MC);

//2. Сигнал задачам Т1, Т2, Т3 о завершении вводa

ReleaseSemaphore(S4, 3, NULL); //S1.1, S2.1, S3.1

//3. Ждать введения в Т2, Т3

WaitForSingleObject(S2, INFINITE); //W2.1

WaitForSingleObject(S3, INFINITE); //3.1

//4. Копирование α4 := α (МЮТЕКС)

WaitForSingleObject(Mut, INFINITE);

alfa4 = alfa;

ReleaseMutex(Mut);

//5. Копирование MC4 := MC (КРИТИЧЕСКАЯ СЕКЦИЯ)

EnterCriticalSection(&CS);

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

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

MC4[i][j] = MC[i][j];

LeaveCriticalSection(&CS);

//6. Cчёт MAH = MBH + α4 (MC4 \* MZH)

for (int z = (task - 1) \* H; z < task \* H; z++){

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

sum = 0;

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

sum = sum + MZ[z][j] \* MC4[j][i]; //(MC4 \* MZH)

}

sum = sum \* alfa4; //α4 (MC4 \* MZH)

MA[z][i] = sum + MB[z][i]; //MBH + α4 (MC4 \* MZH)

}

}

//7. Ждать завершения счёта MA в задачах Т1, Т2, Т3

WaitForSingleObject(Eve1, NULL);//W1.2

WaitForSingleObject(Eve2, NULL);//W2.2

WaitForSingleObject(Eve3, NULL);//W3.2

//8. Вывод MA.

cout << " " << endl;

cout << " " << endl;

cout << " " << endl;

cout << " " << endl;

if( N < 12 ) {

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

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

cout << MA[i][j] << " ";

}

cout << "\n";

}

}

cout << "T4 finished" << endl;

}

//=========================MAIN===========================

int main()

{

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

{

MA[i] = new int[N];

}

DWORD Tid1, Tid2, Tid3, Tid4;

HANDLE Thread1, Thread2, Thread3, Thread4;

Eve1 = CreateEvent(NULL, 0, 1, NULL);

Eve2 = CreateEvent(NULL, 0, 1, NULL);

Eve3 = CreateEvent(NULL, 0, 1, NULL);

S2 = CreateSemaphore(NULL, 0, 3, NULL);

S3 = CreateSemaphore(NULL, 0, 3, NULL);

S4 = CreateSemaphore(NULL, 0, 3, NULL);

Mut = CreateMutex(NULL, FALSE, NULL);

InitializeCriticalSection(&CS);

size\_t st = 50 \* 1024 \* 1024;

Thread1 = CreateThread(NULL, 0, (LPTHREAD\_START\_ROUTINE) T1, NULL, 0, &Tid1);

Thread2 = CreateThread(NULL, 0, (LPTHREAD\_START\_ROUTINE) T2, NULL, 0, &Tid2);

Thread3 = CreateThread(NULL, 0, (LPTHREAD\_START\_ROUTINE) T3, NULL, 0, &Tid3);

Thread4 = CreateThread(NULL, 0, (LPTHREAD\_START\_ROUTINE) T4, NULL, 0, &Tid4);

CloseHandle(Thread1);

CloseHandle(Thread2);

CloseHandle(Thread3);

cin.get();

return 0;

}