Лабораторная работа №2

по курсу:

«Паралельные и распределённые вычисления»

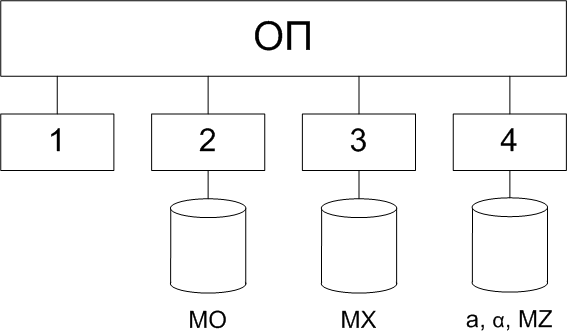
Тема: «Win32»

Выполнил: студент группы ИВ-83

НТУУ «КПИ» ФИВТ

**Техническое задание**

а = min(MO + MX \* MZ \* α)



**Этап 1. Построение параллельного алгоритма**

Вычисление данного матричного уравнения можно разбить на шаги:

1. ai = min(MOH + MX \* MZH \* α), i = , где Р – количество процессоров

**Общие ресурсы:** MX, α

1. a = min(a, ai)

**Общие ресурсы:** a

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

**№ Т1 ТС/КУ**

1. Ждать введения в задачах Т2, Т3, Т4 W2.1 W3.1 W4.1
2. Копирование α1 := α, МX1 := MX КУ
3. Счёт a1 = min(MOH + MX1 \* MZH \* α1)
4. Счёт a = min(a, a1) KУ
5. Сигнал задаче Т4 о завершении счёта S4.1

**№ Т2 ТС/КУ**

1. Ввод MО
2. Сигнал задачам Т1, Т3, Т4 о завершении ввода S1.1 S3.1 S4.1
3. Ждать введения в Т3, Т4 W3.1 W4.1
4. Копирование α2 := α, МХ2 := MХ КУ
5. Счёт a2 = min(MOH + MX2 \* MZH \* α2)
6. Счёт a = min(a, a2) KУ
7. Сигнал задаче Т4 о завершении счёта S4.2

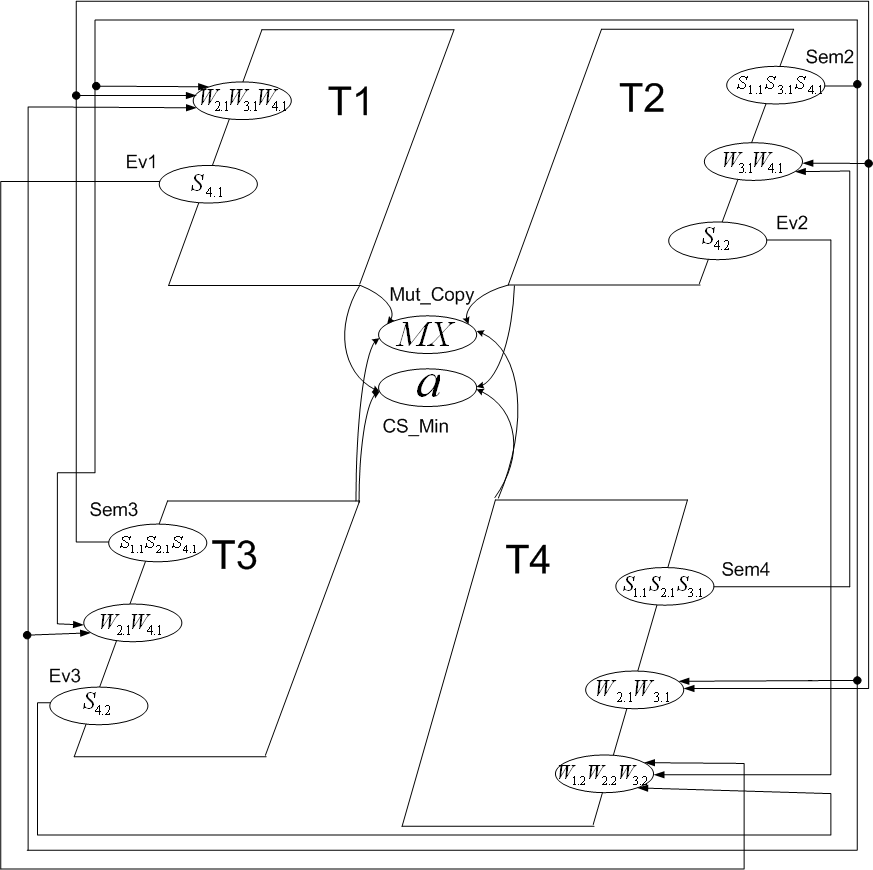
**№ Т3 ТС/КУ**

1. Ввод MX
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, МХ3 := MХ КУ
5. Счёт a3 = min(MOH + MX3 \* MZH \* α3)
6. Счёт a = min(a, a3) KУ
7. Сигнал задаче Т4 о завершении счёта S4.2

**№ Т4 ТС/КУ**

1. Ввод MZ, α
2. Сигнал задачам Т1, Т2, Т3 о завершении ввода S1.1 S2.1 S3.1
3. Ждать введения в Т2, Т3 W2.1 W3.1
4. Копирование α4 := α, МХ4 := MХ КУ
5. Счёт a4 = min(MOH + MX4 \* MZH \* α4)
6. Счёт a = min(a, a4) KУ
7. Ждать завершения счёта в задачах Т1, Т2, Т3 W1.2 W2.2 W3.2
8. Вывод а

**Этап 3. Разработка схемы взаимодействия задач**



**Этап 4. Разработка программы**

#include "stdafx.h"

#include "windows.h"

#include <iostream>

#include <fstream>

#include <algorithm>

#include <string>

using namespace std;

int N = 8;

const int P = 4;

const int H = N/P;

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

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

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

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

int alfa;

int a = 999999999;

//Semaphores

HANDLE Sem2, Sem3, Sem4;

//Critical sections

CRITICAL\_SECTION CS\_Min;

//Mutex

HANDLE Mut\_Copy;

//Events

HANDLE Ev1, Ev2, Ev3;

static void inputVector(int\* vec, int temp) {

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

vec[j] = temp;

}

}

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

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] = temp;

}

}

}

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

void T1()

{

int alfa1, sum;

int a1 = 999999999;

int \*\* MX1 = NULL;

MX1 = new int\*[N];

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

{

MX1[i] = new int[N];

}

int task = 1; //Task number

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

//1. Ждать введения в задачах Т2, Т3, Т4

WaitForSingleObject(Sem2, INFINITE); // W2-1

WaitForSingleObject(Sem3, INFINITE); // W3-1

WaitForSingleObject(Sem4, INFINITE); // W4-1

//2. Копирование alfa1 := alfa, МX1 := MX

WaitForSingleObject(Mut\_Copy, INFINITE);

alfa1 = alfa;

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

{

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

{

MX1[i][j] = MX[i][j];

}

}

ReleaseMutex(Mut\_Copy);

//3. Счёт a1 = min(MOH + MX1 \* MZH \* alfa1)

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] \* MX1[j][i] \* alfa1;

}

sum = sum + MO[z][i];

if (sum < a1) {

a1 = sum;

}

}

}

//4. Счёт a = min(a, a1)

EnterCriticalSection(&CS\_Min);

if (a > a1) {

a = a1;

}

LeaveCriticalSection(&CS\_Min);

//5. Сигнал задаче Т4 о завершении счёта

SetEvent(Ev1); //S4.1

cout << "T1 finished" << endl;

}

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

void T2()

{

int alfa2, sum;

int a2 = 999999999;

int \*\* MX2 = NULL;

MX2 = new int\*[N];

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

{

MX2[i] = new int[N];

}

int task = 2; //Task number

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

//1. Ввод MО

inputMatrix(MO, 1);

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

ReleaseSemaphore(Sem2, 3, NULL); //S1.1 S3.1 S4.1

//3. Ждать введения в Т3, Т4

WaitForSingleObject(Sem3, INFINITE); // W3-1

WaitForSingleObject(Sem4, INFINITE); // W4-1

//4. Копирование alfa2 := alfa, МX2 := MX

WaitForSingleObject(Mut\_Copy, INFINITE);

alfa2 = alfa;

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

{

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

{

MX2[i][j] = MX[i][j];

}

}

ReleaseMutex(Mut\_Copy);

//5. Счёт a2 = min(MOH + MX2 \* MZH \* alfa2)

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] \* MX2[j][i] \* alfa2;

}

sum = sum + MO[z][i];

if (sum < a2) {

a2 = sum;

}

}

}

//6. Счёт a = min(a, a2)

EnterCriticalSection(&CS\_Min);

if (a > a2) {

a = a2;

}

LeaveCriticalSection(&CS\_Min);

//7. Сигнал задаче Т4 о завершении счёта

SetEvent(Ev2); //S4.2

cout << "T2 finished" << endl;

}

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

void T3()

{

int alfa3, sum;

int a3 = 999999999;

int \*\* MX3 = NULL;

MX3 = new int\*[N];

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

{

MX3[i] = new int[N];

}

int task = 3; //Task number

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

//1. Ввод MX

inputMatrix(MX, 1);

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

ReleaseSemaphore(Sem3, 3, NULL); //S1.1 S2.1 S4.1

//3. Ждать введения в Т2, Т4

WaitForSingleObject(Sem2, INFINITE); // W2-1

WaitForSingleObject(Sem4, INFINITE); // W4-1

//4. Копирование alfa3 := alfa, МX3 := MX

WaitForSingleObject(Mut\_Copy, INFINITE);

alfa3 = alfa;

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

{

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

{

MX3[i][j] = MX[i][j];

}

}

ReleaseMutex(Mut\_Copy);

//5. Счёт a3 = min(MOH + MX3 \* MZH \* alfa3)

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] \* MX3[j][i] \* alfa3;

}

sum = sum + MO[z][i];

if (sum < a3) {

a3 = sum;

}

}

}

//6. Счёт a = min(a, a3)

EnterCriticalSection(&CS\_Min);

if (a > a3) {

a = a3;

}

LeaveCriticalSection(&CS\_Min);

//7. Сигнал задаче Т4 о завершении счёта

SetEvent(Ev3); //S4.2

cout << "T3 finished" << endl;

}

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

void T4()

{

int alfa4, sum;

int a4 = 999999999;

int \*\* MX4 = NULL;

MX4 = new int\*[N];

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

{

MX4[i] = new int[N];

}

int task = 4; //Task number

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

//1. Ввод MZ, alfa

inputMatrix(MZ, 1);

alfa = 1;

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

ReleaseSemaphore(Sem4, 3, NULL); //S1.1 S2.1 S3.1

//3. Ждать введения в Т2, Т3

WaitForSingleObject(Sem2, INFINITE); // W2-1

WaitForSingleObject(Sem3, INFINITE); // W3-1

//4. Копирование alfa4 := alfa, МX4 := MX

WaitForSingleObject(Mut\_Copy, INFINITE);

alfa4 = alfa;

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

{

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

{

MX4[i][j] = MX[i][j];

}

}

ReleaseMutex(Mut\_Copy);

//5. Счёт a4 = min(MOH + MX4 \* MZH \* alfa4)

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] \* MX4[j][i] \* alfa4;

}

sum = sum + MO[z][i];

if (sum < a4) {

a4 = sum;

}

}

}

//6. Счёт a = min(a, a4)

EnterCriticalSection(&CS\_Min);

if (a > a4) {

a = a4;

}

LeaveCriticalSection(&CS\_Min);

//7. Ждать завершения счёта в задачах Т1, Т2, Т3 W1.2 W2.2 W3.2

WaitForSingleObject(Ev1, INFINITE); // W1-2

WaitForSingleObject(Ev2, INFINITE); // W2-2

WaitForSingleObject(Ev3, INFINITE); // W3-2

//14. Вывод A

cout << "\t result" << endl;

cout << "\t is" << endl;

cout << " " << endl;

cout << " " << endl;

cout<< a << "\t";

cout << "T4 finished" << endl;

}

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

int main()

{

char a;

DWORD Tid1, Tid2, Tid3, Tid4;

HANDLE Thread1, Thread2, Thread3, Thread4;

cout << "Program started with H = " << H << endl;

Ev1 = CreateEvent(NULL, 0, 1, NULL);

Ev2 = CreateEvent(NULL, 0, 1, NULL);

Ev3 = CreateEvent(NULL, 0, 1, NULL);

Sem2 = CreateSemaphore(NULL, 0, 3, NULL);

Sem3 = CreateSemaphore(NULL, 0, 3, NULL);

Sem4 = CreateSemaphore(NULL, 0, 3, NULL);

Mut\_Copy = CreateMutex(NULL, FALSE, NULL);

InitializeCriticalSection(&CS\_Min);

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();

cin.get();

cin.get();

return 0;

}