Лабораторная работа №2

по курсу:

«Паралельные и распределённые вычисления»

Тема: «Win32»

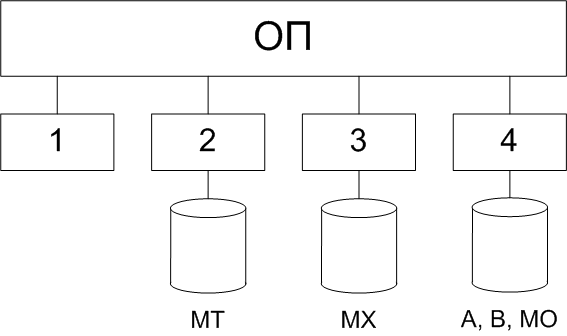
Выполнил: студент группы ИВ-83

НТУУ «КПИ» ФИВТ

Воробйов Виталий

**Техническое задание**

A = (B \* MX)(MO \* MT)



**Этап 1. Построение параллельного алгоритма**

Вычисление данного матричного уравнения можно разбить на шаги:

1. CH = B \* MXH

**Общие ресурсы:** B

1. MCH = MO \* MTH

**Общие ресурсы:** MO

1. AH = C \* MCH

**Общие ресурсы:** C

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

**№ Т1 ТС/КУ**

1. Ждать введения MX в Т3 и B в Т4 W3.1, W4.1(1)
2. Копирование B1 := B КУ
3. Cчёт CH = B1 \* MXH
4. Сигнал задачам Т2, Т3, Т4 о завершении счёта С S2.2, S3.2, S4.2
5. Ждать введения МО в Т4 и МТ в Т2 W2.1, W4.1(2)
6. Копирование МО1 := MO КУ
7. Счёт MCH = MO1 \* MTH
8. Ждать завершения счёта С в задачах Т2, Т3, Т4 W2.2, W3.2, W4.2
9. Копирование С1 := C КУ
10. Счёт AH = C1 \* MCH
11. Сигнал задаче Т4 о завершении счёта. S4.3

**№ Т2 ТС/КУ**

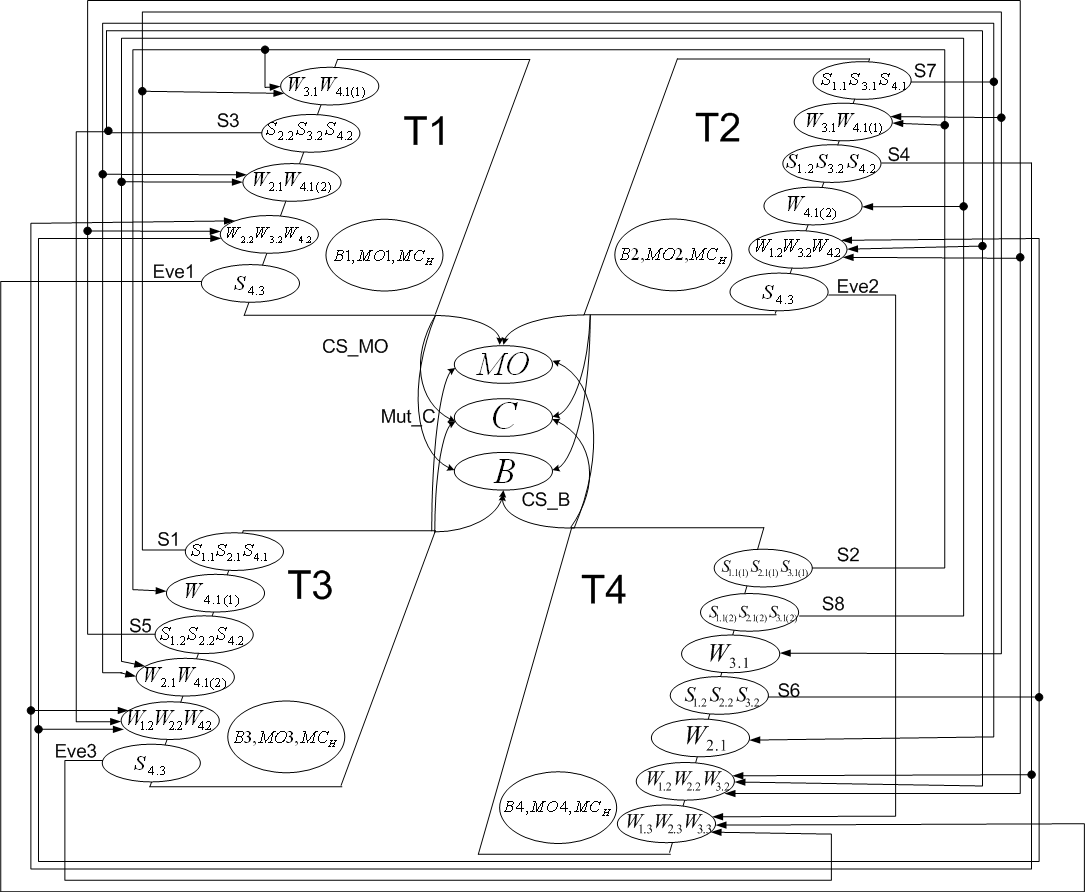
1. Ввод МТ
2. Сигнал задачам Т1, Т3, Т4 о завершении ввода МТ S1.1, S3.1, S4.1
3. Ждать введения MX в Т3 и B в Т4 W3.1, W4.1(1)
4. Копирование B2 := B КУ
5. Cчёт CH = B2 \* MXH
6. Сигнал задачам Т1, Т3, Т4 о завершении счёта С S1.2, S3.2, S4.2
7. Ждать введения МО в Т4 W4.1(2)
8. Копирование МО2 := MO КУ
9. Счёт MCH = MO2 \* MTH
10. Ждать завершения счёта С в задачах Т1, Т3, Т4 W1.2, W3.2, W4.2
11. Копирование С2 := C КУ
12. Счёт AH = C2 \* MCH
13. Сигнал задаче Т4 о завершении счёта. S4.3

**№ Т3 ТС/КУ**

1. Ввод МX
2. Сигнал задачам Т1, Т2, Т4 о завершении ввода МX S1.1, S2.1, S4.1
3. Ждать введения B в Т4 W4.1(1)
4. Копирование B3 := B КУ
5. Cчёт CH = B3 \* MXH
6. Сигнал задачам Т1, Т2, Т4 о завершении счёта С S1.2, S2.2, S4.2
7. Ждать введения МО в Т4 и МТ в Т2 W2.1, W4.1(2)
8. Копирование МО3 := MO КУ
9. Счёт MCH = MO3 \* MTH
10. Ждать завершения счёта С в задачах Т1, Т2, Т4 W1.2, W2.2, W4.2
11. Копирование С3 := C КУ
12. Счёт AH = C3 \* MCH
13. Сигнал задаче Т4 о завершении счёта. S4.3

**№ Т4 ТС/КУ**

1. Ввод B
2. Cигнал задачам Т1, Т2, Т3 о завершении ввода B S1.1(1), S2.1(1), S3.1(1)
3. Ввод MO
4. Сигнал задачам Т1, Т2, Т3 о завершении ввода MO S1.1(2), S2.1(2), S3.1(2)
5. Ждать введения MX в Т3 W3.1
6. Копирование B4 := B КУ
7. Cчёт CH = B4 \* MXH
8. Сигнал задачам Т1, Т2, Т3 о завершении счёта С S1.2, S2.2, S3.2
9. Ждать введения МТ в Т2 W2.1
10. Копирование МО4 := MO КУ
11. Счёт MCH = MO4 \* MTH
12. Ждать завершения счёта С в задачах Т1, Т2, Т3 W1.2, W2.2, W3.2
13. Копирование С4 := C КУ
14. Счёт AH = C4 \* MCH
15. Ждать завершение счёта А в задачах Т1, Т2, Т3 W1.3, W2.3, W3.3
16. Вывод А.

**Этап 3. Разработка схемы взаимодействия задач**

**Этап 4. Разработка программы**

/\*\*

\* @author Vorobyev Vitaliy IO-73

\* Paralel and distributed computing

\* Laboratory work #2. Win32

\* 21.02.2011

\*/

#include "stdafx.h"

#include "windows.h"

#include <iostream>

#include <fstream>

#include <algorithm>

using namespace std;

const int N = 8000;

const int P = 4;

const int H = N/P;

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

Vector A, B, C;

Matrix MX, MO, MT;

//Semaphores

HANDLE S1, S2, S3, S4, S5, S6, S7, S8;

//Critical sections

CRITICAL\_SECTION CS\_MO, CS\_B;

//Mutex

HANDLE Mut\_C;

//Events

HANDLE Eve1, Eve2, Eve3;

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

void T1()

{

Vector B1, C1;

Matrix MO1, MC;

int task = 1; //Task number

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

//1. Ждать введения MX в Т3 и B в Т4

WaitForSingleObject(S1, INFINITE); //W3.1

WaitForSingleObject(S2, INFINITE); //W4.1(1)

//2. Копирование B1 := B (КРИТИЧЕСКИЙ УЧАСТОК)

EnterCriticalSection(&CS\_B);

B1 = B;

LeaveCriticalSection(&CS\_B);

//3. Cчёт CH = B1 \* MXH

int sum;

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

sum = 0;

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

sum = sum + B1.vect[j] \* MX.mas[j].vect[i];

}

C.vect[i] = sum;

}

//4. Сигнал задачам Т2, Т3, Т4 о завершении счёта СH

ReleaseSemaphore(S3, 3, NULL); //S2.2 S3.2 S4.2

//5. Ждать введения МО в Т4 и МТ в Т2

WaitForSingleObject(S7, INFINITE); //W2.1

WaitForSingleObject(S8, INFINITE); //W4.1(2)

//6. Копирование МО1 := MO (КРИТИЧЕСКИЙ УЧАСТОК)

EnterCriticalSection(&CS\_MO);

MO1 = MO;

LeaveCriticalSection(&CS\_MO);

//7. Счёт MCH = MO1 \* MTH

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 + MT.mas[z].vect[j] \* MO1.mas[j].vect[i];

}

MC.mas[z].vect[i] = sum;

}

}

//8. Ждать завершения счёта С в задачах Т2, Т3, Т4

WaitForSingleObject(S4, INFINITE); //W2.2

WaitForSingleObject(S5, INFINITE); //W3.2

WaitForSingleObject(S6, INFINITE); //W4.2

//9. Копирование С1 := C (КРИТИЧЕСКИЙ УЧАСТОК)

WaitForSingleObject(Mut\_C, INFINITE);

C1 = C;

ReleaseMutex(Mut\_C);

//10. Счёт AH = C1 \* MCH

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

sum = 0;

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

sum = sum + C1.vect[j] \* MC.mas[i].vect[j];

}

A.vect[i] = sum;

}

cout << "T1: calculating A finished" << endl;

//11. Сигнал задаче Т4 о завершении счёта.

SetEvent(Eve1); //S4.3

cout << "T1: finished" << endl;

}

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

void T2()

{

Vector B2, C2;

Matrix MO2, MC;

int task = 2; //Task number

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

//1. Ввод МТ

MT.InputMatrix();

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

ReleaseSemaphore(S7, 3, NULL); //S1.1, S3.1, S4.1

//3. Ждать введения MX в Т3 и B в Т4

WaitForSingleObject(S1, INFINITE); //W3.1

WaitForSingleObject(S2, INFINITE); //W4.1(1)

//4. Копирование B2 := B (КРИТИЧЕСКИЙ УЧАСТОК)

EnterCriticalSection(&CS\_B);

B2 = B;

LeaveCriticalSection(&CS\_B);

//5. Cчёт CH = B2 \* MXH

int sum;

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

sum = 0;

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

sum = sum + B2.vect[j] \* MX.mas[j].vect[i];

}

C.vect[i] = sum;

}

//6. Сигнал задачам Т1, Т3, Т4 о завершении счёта СH

ReleaseSemaphore(S4, 3, NULL); //S1.2 S3.2 S4.2

//7. Ждать введения МО в Т4 W4.1(2)

WaitForSingleObject(S8, INFINITE); //W4.1(2)

//8. Копирование МО2 := MO (КРИТИЧЕСКИЙ УЧАСТОК)

EnterCriticalSection(&CS\_MO);

MO2 = MO;

LeaveCriticalSection(&CS\_MO);

//9. Счёт MCH = MO2 \* MTH

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 + MT.mas[z].vect[j] \* MO2.mas[j].vect[i];

}

MC.mas[z].vect[i] = sum;

}

}

//10. Ждать завершения счёта С в задачах Т1, Т3, Т4

WaitForSingleObject(S3, INFINITE); //W1.2

WaitForSingleObject(S5, INFINITE); //W3.2

WaitForSingleObject(S6, INFINITE); //W4.2

//11. Копирование С2 := C (КРИТИЧЕСКИЙ УЧАСТОК)

WaitForSingleObject(Mut\_C, INFINITE);

C2 = C;

ReleaseMutex(Mut\_C);

//12. Счёт AH = C2 \* MCH

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

sum = 0;

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

sum = sum + C2.vect[j] \* MC.mas[i].vect[j];

}

A.vect[i] = sum;

}

cout << "T2: calculating A finished" << endl;

//13. Сигнал задаче Т4 о завершении счёта.

SetEvent(Eve2); //S4.3

cout << "T2: finished" << endl;

}

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

void T3()

{

Vector B3, C3;

Matrix MO3, MC;

int task = 3; //Task number

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

//1. Ввод МX

MX.InputMatrix();

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

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

//3. Ждать введения B в Т4 W4.1(1)

WaitForSingleObject(S2, INFINITE); //W4.1(1)

//4. Копирование B3 := B (КРИТИЧЕСКИЙ УЧАСТОК)

EnterCriticalSection(&CS\_B);

B3 = B;

LeaveCriticalSection(&CS\_B);

//5. Cчёт CH = B3 \* MXH

int sum;

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

sum = 0;

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

sum = sum + B3.vect[j] \* MX.mas[j].vect[i];

}

C.vect[i] = sum;

}

//6. Сигнал задачам Т1, Т2, Т4 о завершении счёта СH

ReleaseSemaphore(S5, 3, NULL); //S1.2 S2.2 S4.2

//7. Ждать введения МО в Т4 и МТ в Т2 W2.1, W4.1(2)

WaitForSingleObject(S8, INFINITE); //W2.1

WaitForSingleObject(S7, INFINITE); //W4.1(2)

//8. Копирование МО3 := MO (КРИТИЧЕСКИЙ УЧАСТОК)

EnterCriticalSection(&CS\_MO);

MO3 = MO;

LeaveCriticalSection(&CS\_MO);

//9. Счёт MCH = MO3 \* MTH

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 + MT.mas[z].vect[j] \* MO3.mas[j].vect[i];

}

MC.mas[z].vect[i] = sum;

}

}

//10. Ждать завершения счёта С в задачах Т1, Т2, Т4

WaitForSingleObject(S3, INFINITE); //W1.2

WaitForSingleObject(S4, INFINITE); //W2.2

WaitForSingleObject(S6, INFINITE); //W4.2

//11. Копирование С3 := C (КРИТИЧЕСКИЙ УЧАСТОК)

WaitForSingleObject(Mut\_C, INFINITE);

C3 = C;

ReleaseMutex(Mut\_C);

//12. Счёт AH = C3 \* MCH

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

sum = 0;

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

sum = sum + C3.vect[j] \* MC.mas[i].vect[j];

}

A.vect[i] = sum;

}

cout << "T3: calculating A finished" << endl;

//13. Сигнал задаче Т4 о завершении счёта.

SetEvent(Eve3); //S4.3

cout << "T3: finished" << endl;

}

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

void T4()

{

Vector B4, C4;

Matrix MO4, MC;

int task = 4; //Task number

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

//1. Ввод B

B.InputVector();

//2. Cигнал задачам Т1, Т2, Т3 о завершении ввода B

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

//3. Ввод MO

MO.InputMatrix();

//4. Сигнал задачам Т1, Т2, Т3 о завершении ввода MO

ReleaseSemaphore(S8, 3, NULL); //S1.1(2), S2.1(2), S3.1(2)

//5. Ждать введения MX в Т3

WaitForSingleObject(S1, INFINITE); //W3.1

//6. Копирование B4 := B (КРИТИЧЕСКИЙ УЧАСТОК)

EnterCriticalSection(&CS\_B);

B4 = B;

LeaveCriticalSection(&CS\_B);

//5. Cчёт CH = B4 \* MXH

int sum;

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

sum = 0;

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

sum = sum + B4.vect[j] \* MX.mas[j].vect[i];

}

C.vect[i] = sum;

}

//8. Сигнал задачам Т1, Т2, Т3 о завершении счёта СH

ReleaseSemaphore(S6, 3, NULL); //S1.2 S2.2 S3.2

//9. Ждать введения МТ в Т2

WaitForSingleObject(S7, INFINITE); //W2.1

//10. Копирование МО4 := MO (КРИТИЧЕСКИЙ УЧАСТОК)

EnterCriticalSection(&CS\_MO);

MO4 = MO;

LeaveCriticalSection(&CS\_MO);

//11. Счёт MCH = MO4 \* MTH

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 + MT.mas[z].vect[j] \* MO4.mas[j].vect[i];

}

MC.mas[z].vect[i] = sum;

}

}

//12. Ждать завершения счёта С в задачах Т1, Т2, Т3

WaitForSingleObject(S3, INFINITE); //W1.2

WaitForSingleObject(S4, INFINITE); //W2.2

WaitForSingleObject(S5, INFINITE); //W3.2

//13. Копирование С4 := C (КРИТИЧЕСКИЙ УЧАСТОК)

WaitForSingleObject(Mut\_C, INFINITE);

C4 = C;

ReleaseMutex(Mut\_C);

//12. Счёт AH = C4 \* MCH

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

sum = 0;

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

sum = sum + C4.vect[j] \* MC.mas[i].vect[j];

}

A.vect[i] = sum;

}

cout << "T4: calculating A finished" << endl;

//15. Ждать завершение счёта А в задачах Т1, Т2, Т3

WaitForSingleObject(Eve1, NULL); //W1.3

WaitForSingleObject(Eve2, NULL); //W2.3

WaitForSingleObject(Eve3, NULL); //W3.3

//16. Вывод А.

cout << " " << endl;

cout << " " << endl;

cout << " " << endl;

cout << " " << endl;

A.OutputVector();

cout << "T4 finished" << endl;

}

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

int \_tmain(int argc, \_TCHAR\* argv[])

{

DWORD Tid1, Tid2, Tid3, Tid4;

HANDLE Thread1, Thread2, Thread3, Thread4;

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

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

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

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

S1 = CreateSemaphore(NULL, 0, 3, NULL);

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

S3 = CreateSemaphore(NULL, 0, 1, NULL);

S4 = CreateSemaphore(NULL, 0, 1, NULL);

S5 = CreateSemaphore(NULL, 0, 1, NULL);

S6 = CreateSemaphore(NULL, 0, 1, NULL);

S7 = CreateSemaphore(NULL, 0, 3, NULL);

S8 = CreateSemaphore(NULL, 0, 3, NULL);

Mut\_C = CreateMutex(NULL, 0, NULL);

InitializeCriticalSection(&CS\_MO);

InitializeCriticalSection(&CS\_B);

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

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

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

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

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

CloseHandle(Thread1);

CloseHandle(Thread2);

CloseHandle(Thread3);

cin.get();

return 0;

}