Национальный технический университет

«Киевский политехнический институт»

Факультет информатики и вычислительной техники

## Кафедра Вычислительной Техники

#### Лабораторная работа №5

По курсу: Параллельное программирование

студента III-го курса,

группы ИВ-92

Петрука В.О.

## Киев 2011 г.

/\*Laboratory work 5

Petruk Vadim, IO-92

Variant: 1.13 C = A - B + D

2.28 MD = MIN(MA)\*MB\*MC

3.14 D = (A + B)\*(MA - MB)

\*/

#include <windows.h>

#include <stdio.h>

const int N = 4;

//--------------------------------------------------------------------

void VectorIn(int X[N]){

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

X[i] = 1;

}

}

//--------------------------------------------------------------------

void VectorOut(int X[N]){

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

printf("%d ", X[i]);

}

printf("\n");

}

//--------------------------------------------------------------------

void MatrixIn(int MX[N][N]){

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

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

MX[i][j] = 1;

}

}

}

//--------------------------------------------------------------------

void MatrixOut(int MX[N][N]){

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

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

printf("%d ",MX[i][j]);

}

printf("\n");

}

}

//--------------------------------------------------------------------

//C = A - B + D

void Function1(void){

printf("Function1 is started \n");

int A[N];

int B[N];

int C[N];

int D[N];

VectorIn(A);

VectorIn(B);

VectorIn(D);

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

{

C[i] = A[i] - B[i] + D[i];

}

if (N<=8)

{

VectorOut(C);

}

printf("Function1 s finished \n");

}

//--------------------------------------------------------------------

//MD = MIN(MA)\*MB\*MC

void Function2(void){

printf("Function2 is started \n");

int MA[N][N] , MB[N][N] , MC[N][N] , MD[N][N];

MatrixIn(MA);

MatrixIn(MB);

MatrixIn(MC);

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

{

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

{

MD[i][j]=0;

}

}

//MD := MB\*MC

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

{

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

{

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

{

MD[i][j] = MD[i][j]+MB[i][k]\*MC[k][j];

}

}

}

//search MIN(MA)

int min\_MA = MA[0][0];

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

{

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

{

if (MA[i][j] < min\_MA)

{

min\_MA = MA[i][j];

}

}

}

//MD := MIN(MA)\*MB\*MC

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

{

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

{

MD[i][j]=MD[i][j]\*min\_MA;

}

}

if (N<=8)

{

Sleep(500);

MatrixOut(MD);

}

printf("Function2 s finished \n");

}

//--------------------------------------------------------------------

// D = (A + B)\*(MA - MB)

void Function3(void)

{

printf("Function3 is started \n");

int A[N];

int B[N];

int MA[N][N];

int MB[N][N];

int D[N];

VectorIn(A);

VectorIn(B);

MatrixIn(MA);

MatrixIn(MB);

int C[N];

//C:=A + B

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

{

C[i] = A[i]+B[i];

}

int MC[N][N];

//MC:=MA - MB

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

{

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

{

MC[i][j] = MA[i][j] - MB[i][j];

}

}

//MD = 0

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

{

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

{

D[i]=0;

}

}

//D := C \* MC

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

{

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

{

D[j] = D[j] + C[j]\*MC[j][i];

}

}

if (N<=8)

{

Sleep(1000);

VectorOut(D);

}

printf("Function3 is finished \n");

}

//--------------------------------------------------------------------

int main(void){

DWORD TidA, TidB, TidC;

HANDLE thread1, thread2, thread3;

thread1 = CreateThread(NULL, 0, (LPTHREAD\_START\_ROUTINE) Function1, NULL, 0, &TidA);

thread2 = CreateThread(NULL, 0, (LPTHREAD\_START\_ROUTINE) Function2, NULL, 0, &TidB);

thread3 = CreateThread(NULL, 0, (LPTHREAD\_START\_ROUTINE) Function3, NULL, 0, &TidC);

SetThreadPriority(thread1, THREAD\_PRIORITY\_HIGHEST);

SetThreadPriority(thread2, THREAD\_PRIORITY\_NORMAL);

SetThreadPriority(thread3, THREAD\_PRIORITY\_LOWEST);

CloseHandle(thread1);

CloseHandle(thread2);

CloseHandle(thread3);

system("pause");

}