НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Лабораторна робота №9

з дісципліни **«**Системне програмування**»**

Виконав:

студент 2 курсу

ФІОТ гр. ІО-21

Кузьменко Володимир

Перевірив:

Порєв В.М.

Київ 2014 р.

**Мета:** Навчитися програмувати модулі на асемблері, у яких використовуються

команди SSE, а також команди x87 FPU.

**Завдання:**

1. Створити проект Visual C++ Win32 з ім'ям Lab10.

2. Написати на асемблері процедуру обчислення скалярного добутку двох

векторів з використанням команд SSE. Ім'я процедури: MyDotProduct\_SSE.

Процедуру оформити у окремому модулі і записати файли vectsse.asm,

vectsse.h. Додати файл vectsse.asm у проект.

3. Запрограмувати на асемблері процедуру обчислення скалярного добутку

двох векторів на основі команд x87 FPU без використання команд SSE. Ім'я

процедури: MyDotProduct\_FPU. Процедуру оформити у окремому модулі і

записати файли vectfpu.asm, vectfpu.h. Додати файл vectfpu.asm у проект.

4. Запрограмувати на C++ обчислення скалярного добутку тих самих векторів

як звичайну функцію C++ з ім'ям MyDotProduct, яка приймає значення двох

масивів і записує результат у числову перемінну (будь-яка оптимізація при

компіляції повинна бути відсутня).

5. Зробити меню для вікна програми так, щоб користувач програми мав

можливість викликати процедури на асемблері MyDotProduct\_SSE,

MyDotProduct\_FPU з модулів vectsse, vectfpu, а також функцію MyDotProduct.

6. Запрограмувати вивід результатів обчислень та виміри часу виконання

скалярного добутку для трьох варіантів реалізації.

7. Отримати дизасемблерний текст функції C++ MyDotProduct. Проаналізувати

код дизасемблеру, порівняти з кодом на асемблері процедури

MyDotProduct\_FPU.

8. Зробити висновки щодо використання модулів на асемблері у програмах на

мові C++ .

Текст програми:

Головний файл С++:

// lab9C.cpp: определяет точку входа для приложения.

//

#include <ctime>

#include "stdafx.h"

#include "lab9C.h"

#include "longop.h"

#include "module.h"

#include "vectsse.h"

#include "vectfpu.h"

#define MAX\_LOADSTRING 100

// Глобальные переменные:

HINSTANCE hInst; // текущий экземпляр

TCHAR szTitle[MAX\_LOADSTRING]; // Текст строки заголовка

TCHAR szWindowClass[MAX\_LOADSTRING]; // имя класса главного окна

// Отправить объявления функций, включенных в этот модуль кода:

ATOM MyRegisterClass(HINSTANCE hInstance);

BOOL InitInstance(HINSTANCE, int);

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

INT\_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM);

void MyWork1(HWND hWnd);

void MyMulVectFPU(HWND hWnd);

void MyMulVectSSE(HWND hWnd);

void MyMulVectC(HWND hWnd);

int APIENTRY \_tWinMain(HINSTANCE hInstance,

HINSTANCE hPrevInstance,

LPTSTR lpCmdLine,

int nCmdShow)

{

UNREFERENCED\_PARAMETER(hPrevInstance);

UNREFERENCED\_PARAMETER(lpCmdLine);

// TODO: разместите код здесь.

MSG msg;

HACCEL hAccelTable;

// Инициализация глобальных строк

LoadString(hInstance, IDS\_APP\_TITLE, szTitle, MAX\_LOADSTRING);

LoadString(hInstance, IDC\_LAB9C, szWindowClass, MAX\_LOADSTRING);

MyRegisterClass(hInstance);

// Выполнить инициализацию приложения:

if (!InitInstance (hInstance, nCmdShow))

{

return FALSE;

}

hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_LAB9C));

// Цикл основного сообщения:

while (GetMessage(&msg, NULL, 0, 0))

{

if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

return (int) msg.wParam;

}

//

// ФУНКЦИЯ: MyRegisterClass()

//

// НАЗНАЧЕНИЕ: регистрирует класс окна.

//

// КОММЕНТАРИИ:

//

// Эта функция и ее использование необходимы только в случае, если нужно, чтобы данный код

// был совместим с системами Win32, не имеющими функции RegisterClassEx'

// которая была добавлена в Windows 95. Вызов этой функции важен для того,

// чтобы приложение получило "качественные" мелкие значки и установило связь

// с ними.

//

ATOM MyRegisterClass(HINSTANCE hInstance)

{

WNDCLASSEX wcex;

wcex.cbSize = sizeof(WNDCLASSEX);

wcex.style = CS\_HREDRAW | CS\_VREDRAW;

wcex.lpfnWndProc = WndProc;

wcex.cbClsExtra = 0;

wcex.cbWndExtra = 0;

wcex.hInstance = hInstance;

wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_LAB9C));

wcex.hCursor = LoadCursor(NULL, IDC\_ARROW);

wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW+1);

wcex.lpszMenuName = MAKEINTRESOURCE(IDC\_LAB9C);

wcex.lpszClassName = szWindowClass;

wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI\_SMALL));

return RegisterClassEx(&wcex);

}

//

// ФУНКЦИЯ: InitInstance(HINSTANCE, int)

//

// НАЗНАЧЕНИЕ: сохраняет обработку экземпляра и создает главное окно.

//

// КОММЕНТАРИИ:

//

// В данной функции дескриптор экземпляра сохраняется в глобальной переменной, а также

// создается и выводится на экран главное окно программы.

//

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)

{

HWND hWnd;

hInst = hInstance; // Сохранить дескриптор экземпляра в глобальной переменной

hWnd = CreateWindow(szWindowClass, szTitle, WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0, NULL, NULL, hInstance, NULL);

if (!hWnd)

{

return FALSE;

}

ShowWindow(hWnd, nCmdShow);

UpdateWindow(hWnd);

return TRUE;

}

//

// ФУНКЦИЯ: WndProc(HWND, UINT, WPARAM, LPARAM)

//

// НАЗНАЧЕНИЕ: обрабатывает сообщения в главном окне.

//

// WM\_COMMAND - обработка меню приложения

// WM\_PAINT -Закрасить главное окно

// WM\_DESTROY - ввести сообщение о выходе и вернуться.

//

//

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)

{

int wmId, wmEvent;

PAINTSTRUCT ps;

HDC hdc;

switch (message)

{

case WM\_COMMAND:

wmId = LOWORD(wParam);

wmEvent = HIWORD(wParam);

// Разобрать выбор в меню:

switch (wmId)

{

case ID\_32772: // Вибір пункту меню

MyWork1(hWnd);

break;

case ID\_32774: // Вибір пункту меню

MyMulVectC(hWnd);

break;

case ID\_32776: // Вибір пункту меню

MyMulVectSSE(hWnd);

break;

case ID\_32777: // Вибір пункту меню

MyMulVectFPU(hWnd);

break;

case IDM\_ABOUT:

DialogBox(hInst, MAKEINTRESOURCE(IDD\_ABOUTBOX), hWnd, About);

break;

case IDM\_EXIT:

DestroyWindow(hWnd);

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

break;

case WM\_PAINT:

hdc = BeginPaint(hWnd, &ps);

// TODO: добавьте любой код отрисовки...

EndPaint(hWnd, &ps);

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

return 0;

}

// Обработчик сообщений для окна "О программе".

INT\_PTR CALLBACK About(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)

{

UNREFERENCED\_PARAMETER(lParam);

switch (message)

{

case WM\_INITDIALOG:

return (INT\_PTR)TRUE;

case WM\_COMMAND:

if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)

{

EndDialog(hDlg, LOWORD(wParam));

return (INT\_PTR)TRUE;

}

break;

}

return (INT\_PTR)FALSE;

}

void MyWork1(HWND hWnd)

{

long oA[12]={0x00000002,0x00000000,0x00000000,0x00000000,

0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000};

long oB[12]={0x00000001,0x00000000,0x00000000,0x00000000,

0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000};

long oN[12]={0x00000002,0x00000000,0x00000000,0x00000000,

0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000};

long oC[12]={0x00000004,0x00000000,0x00000000,0x00000000,

0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000};

long result1[384];

long result2[384];

long result3[384];

long result4[384];

long result5[24] ={0x00000000,0x00000000,0x00000000,0x00000000,

0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,

0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000};

char TextBuf[384];

Sub\_Long(result4, oB, oA);

Add\_Long(result2, oC, oN);

Multlong(result5, result4, result2);

StrHex\_MY(768, result5, TextBuf);

MessageBox(hWnd, TextBuf, "Результат (A-B)(C+2)", MB\_OK);

}

void MyMulVectC(HWND hWnd)

{

double oA[8]={1.0,2.0,3.0,4.0,

5.0,6.0,7.0,8.0};

double oB[8]={1.0,1.0,1.0,1.0,

1.0,1.0,1.0,1.0};

double result = 0;

SYSTEMTIME st;

long tst,ten;

GetLocalTime(&st);

tst = 60000\*(long)st.wMinute

+ 1000\*(long)st.wSecond

+ (long)st.wMilliseconds;

for (long i=0; i<1000000; i++)

{

result = 0;

for(long i = 0; i < 8; i++)

{

result = result + oA[i]\*oB[i];

}

}

GetLocalTime(&st);

ten = 60000\*(long)st.wMinute

+ 1000\*(long)st.wSecond

+ (long)st.wMilliseconds - tst;

TCHAR buf[32];

\_itot(result, buf, 10);

MessageBox(hWnd, buf, "Скалярний добуток С++", MB\_OK);

\_itot(ten, buf, 10);

MessageBox(hWnd, buf, "Час виконання", MB\_OK);

}

void MyMulVectSSE(HWND hWnd)

{

double oA[8]={1.0,2.0,3.0,4.0,

5.0,6.0,7.0,8.0};

double oB[8]={1.0,1.0,1.0,1.0,

1.0,1.0,1.0,1.0};

long result0[3] ={0x00000000,0x00000000,0x00000000};

char TextBuf[384];

SYSTEMTIME st;

long tst,ten;

GetLocalTime(&st);

tst = 60000\*(long)st.wMinute

+ 1000\*(long)st.wSecond

+ (long)st.wMilliseconds;

for (long i=0; i<1000000; i++) //повторюємо мільйон разів

{

//код, для якого потрібно виміряти час

}

GetLocalTime(&st);

ten = 60000\*(long)st.wMinute

+ 1000\*(long)st.wSecond

+ (long)st.wMilliseconds - tst;

// VectSSE(result0, oB, oA, 8);

StrHex\_MY(96, result0, TextBuf);

MessageBox(hWnd, TextBuf, "Скалярний добуток SSE", MB\_OK);

MessageBox(hWnd, "SSE", "Скалярний добуток SSE", MB\_OK);

}

void MyMulVectFPU(HWND hWnd)

{

double oA[8]={1.0,2.0,3.0,4.0,

5.0,6.0,7.0,8.0};

double oB[8]={1.0,1.0,1.0,1.0,

1.0,1.0,1.0,1.0};

long result0[3] ={0x00000000,0x00000000,0x00000000};

char TextBuf[384];

SYSTEMTIME st;

long tst,ten;

GetLocalTime(&st);

tst = 60000\*(long)st.wMinute

+ 1000\*(long)st.wSecond

+ (long)st.wMilliseconds;

for (long i=0; i<1000000; i++) //повторюємо мільйон разів

{

//код, для якого потрібно виміряти час

}

GetLocalTime(&st);

ten = 60000\*(long)st.wMinute

+ 1000\*(long)st.wSecond

+ (long)st.wMilliseconds - tst;

// VectFPU(result0, oB, oA, 8);

StrHex\_MY(96, result0, TextBuf);

MessageBox(hWnd, TextBuf, "Скалярний добуток FPU", MB\_OK);

MessageBox(hWnd, "FPU", "Скалярний добуток FPU", MB\_OK);

}

Заголовочний файл до модулю vectsse.asm:

extern "C"

{

void VectSSE (long \*dest, double \*pB, double \*pA, long bits);

}

Заголовочний файл до модулю vectfpu.asm:

extern "C"

{

void VectFPU (long \*dest, double \*pB, double \*pA, long bits);

}

Модуль vectsse.asm адаптований:

.586

.xmm

.model flat, c

.data

temp dd 4 dup(0)

.code

VectFPU proc dest:DWORD, pB:DWORD, pA:DWORD, bits:DWORD

mov edx, bits ;[ebp+20] ; n

mov esi, pA ;[ebp+16]; a

mov ebx, pB ;[ebp+12]; b

mov edi, dest ;[ebp+8] ; res

fld dword ptr[temp]

cycle:

dec edx

fld dword ptr[esi+edx\*4]

fmul dword ptr[ebx+edx\*4]

faddp st(1), st(0)

cmp edx, 0

jne cycle

fstp dword ptr[edi]

ret 16

VectFPU endp

end

Модуль vectfpu.asm адаптований:

.586

.xmm

.model flat, c

.data

temp dd 4 dup(0)

.code

VectSSE proc dest:DWORD, pB:DWORD, pA:DWORD, bits:DWORD

mov edx, bits ;[ebp+20] ; n

mov esi, pA ;[ebp+16]; a

mov ebx, pB ;[ebp+12]; b

mov edi, dest ;[ebp+8] ; res

cycle:

sub edx, 4

movups xmm0, [esi+edx\*4]

movups xmm1, [ebx+edx\*4]

mulps xmm0, xmm1

haddps xmm0, xmm0

haddps xmm0, xmm0

addps xmm2, xmm0

cmp edx, 0

jne cycle

movups temp, xmm2

mov eax, dword ptr[temp]

mov dword ptr[edi], eax

ret 16

VectSSE endp

end

Висновок:

Під час виконання даної лабораторної роботи були закріплені навички використання команд SSE, а також команди FPU. Для цього були написані модулі, які обчислювали скалярний добуток векторів.