МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

Лабораторна робота №11

з предмету “Системне програмування”

Виконав:

студент 2-го курсу ФІОТ

групи ІО-31

Долинний О.В.

Київ 2015

Лабораторна робота №11

Програмування команд SSE у модулях на асемблері

Мета: Навчитися програмувати модулі на асемблері, у яких містяться команди SSE, команди x87 FPU, а також використовувати такі модулі у проектах C++.

**Завдання:**

1. Створити проект Visual C++ Win32 з ім'ям Lab11.

2. Написати на асемблері процедуру обчислення скалярного добутку двох векторів із 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++ .

**Варіант завдання:** 11

Кількість елементів векторів A та B має бути N = 40 \* 11 = 440

Роздруківка тексту програми

**WinProject11.cpp**

void Test\_SSE(HWND hWnd) {

\_\_declspec(align(16)) float A[440];

\_\_declspec(align(16)) float B[440];

\_\_declspec(align(16)) float res;

std::fill(std::begin(A), std::end(A), 1.5);

std::fill(std::begin(B), std::end(B), 2.5);

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++)

{

MyDotProduct\_SSE(&res, A, B, 440);

}

GetLocalTime(&st);

ten = 60000 \* (long)st.wMinute

+ 1000 \* (long)st.wSecond

+ (long)st.wMilliseconds - tst;

char sBuffer[100];

sprintf\_s(sBuffer, "Result = %f, time = %d :)", res, ten);

MessageBox(nullptr, sBuffer, "SSE", MB\_OK);

}

void Test\_FPU(HWND hWnd) {

\_\_declspec(align(16)) float A[440];

\_\_declspec(align(16)) float B[440];

\_\_declspec(align(16)) float res = 0;

std::fill(std::begin(A), std::end(A), 1.5);

std::fill(std::begin(B), std::end(B), 2.5);

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++)

{

MyDotProduct\_FPU(&res, A, B, 440);

}

GetLocalTime(&st);

ten = 60000 \* (long)st.wMinute

+ 1000 \* (long)st.wSecond

+ (long)st.wMilliseconds - tst;

char sBuffer[100];

sprintf\_s(sBuffer, "Result = %f, time = %d :)", res, ten);

MessageBox(nullptr, sBuffer, "FPU", MB\_OK);

}

void Test\_CPP(HWND hWnd) {

\_\_declspec(align(16)) float A[440];

\_\_declspec(align(16)) float B[440];

\_\_declspec(align(16)) float res;

std::fill(std::begin(A), std::end(A), 1.5);

std::fill(std::begin(B), std::end(B), 2.5);

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++)

{

MyDotProduct(res, A, B, 440);

}

GetLocalTime(&st);

ten = 60000 \* (long)st.wMinute

+ 1000 \* (long)st.wSecond

+ (long)st.wMilliseconds - tst;

char sBuffer[100];

sprintf\_s(sBuffer, "Result = %f, time = %d :)", res, ten);

MessageBox(hWnd, sBuffer, "C++", MB\_OK);

}

void MyDotProduct(float &res, float \*A, float \*B, long N) {

float r = 0;

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

r += A[i] \* B[i];

}

res = r;

}

**vectsse.asm**

.686

.xmm

.model flat, C

.data

temp dd 4 dup (0)

.code

MyDotProduct\_SSE proc dest:dword, pA:dword, pB:dword, N:dword;

mov edx, N ; n

mov esi, pA ; a

mov ebx, pB ; b

mov edi, dest ; res

xorps xmm2, xmm2

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

MyDotProduct\_SSE endp

end

**vectfpu.asm**

.586

.model flat, c

.code

MyDotProduct\_FPU proc res:dword, A:dword, B:dword, N:dword;

mov eax, A

mov ebx, B

mov ecx, N

dec ecx

fld dword ptr [eax+4\*ecx]

fmul dword ptr [ebx+4\*ecx]

@cycle:

dec ecx

js @exit

fld dword ptr [eax+4\*ecx]

fmul dword ptr [ebx+4\*ecx]

faddp st(1), st(0)

jmp @cycle

@exit:

mov edx, res

fstp dword ptr [edx]

ret

MyDotProduct\_FPU endp

end

|  |  |
| --- | --- |
| **Результати: (1.5\*2.5\*440=1650)** | **Аналіз результатів:** |
|  | Програма виконує обчислення скалярного добутку двох векторів. |
|  | Результат отримано вірний для усіх способів. |
|  | Програмування на мові високого рівня (C++) виявилося найбільш повільним, найшвидшим виявилося застосування команд формату SSE, що у приблизно 2 рази швидше за формат FPU та у 6 разів програмування на C++. |

**Висновок:**

Під час виконання лабораторної роботи були покращені навички написання власних модулів, роботи з циклами, а також були закріплено основні навички в операціях роботи із форматом SSE та доведено швідкість виконання операцій на мові асемблеру.