Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

ЛАБОРАТОРНА РОБОТА № 11

З дисципліни «Системне програмування»

ВИКОНАВ:

студент 2 курсу ФІОТ

групи ІО-41

Смішний Д. М.

ПЕРЕВІРИВ:

ст. вик. Порєв В. М.

Київ – 2016

Лабораторна робота №11

Тема: 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++.

**Варіант завдання:** 25

Кількість елементів векторів A та B має бути N = 40 \* 25 = 1000

Роздруківка тексту програми

**TestLab11.cpp**

void Work\_SSE(HWND hWnd)

{

SYSTEMTIME st;

long tst, ten;

float result1 = 0.0;

float result = 0.0;

char TextBuf[100];

MyDotProduct\_SSE(&result1, A, B, NUM);

GetLocalTime(&st);

tst = 60000 \* (long)st.wMinute + 1000 \* (long)st.wSecond + (long)st.wMilliseconds;

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

{

MyDotProduct\_SSE(&result, A, B, NUM);

}

GetLocalTime(&st);

ten = 60000 \* (long)st.wMinute + 1000 \* (long)st.wSecond + (long)st.wMilliseconds - tst;

sprintf\_s(TextBuf, "Result = %f, time = %d ", result1, ten);

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

}

void Work\_FPU(HWND hWnd)

{

SYSTEMTIME st;

long tst, ten;

//float result = 0.0;

char TextBuf[100];

GetLocalTime(&st);

tst = 60000 \* (long)st.wMinute + 1000 \* (long)st.wSecond + (long)st.wMilliseconds;

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

{

MyDotProduct\_FPU(&Result, A, B, NUM);

}

GetLocalTime(&st);

ten = 60000 \* (long)st.wMinute + 1000 \* (long)st.wSecond + (long)st.wMilliseconds - tst;

sprintf\_s(TextBuf, "Result = %f, time = %d ", Result, ten);

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

}

void Work\_CPP(HWND hWnd)

{

SYSTEMTIME st;

long tst, ten;

float result = 0.0;

char TextBuf[100];

GetLocalTime(&st);

tst = 60000 \* (long)st.wMinute + 1000 \* (long)st.wSecond + (long)st.wMilliseconds;

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

{

result = MyDotProduct(A, B, NUM);

}

GetLocalTime(&st);

ten = 60000 \* (long)st.wMinute + 1000 \* (long)st.wSecond + (long)st.wMilliseconds - tst;

sprintf\_s(TextBuf, "Result = %f, time = %d ", result, ten);

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

}

float MyDotProduct(float\* A, float\* B, long N)

{

float result = 0.0;

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

{

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

}

return result;

}

**vectsse.asm**

.686

.xmm

.model flat, C

.code

MyDotProduct\_SSE proc result:DWORD, A:DWORD, B:DWORD, NUM:DWORD

mov edx, result

mov eax, A

mov ebx, B

mov ecx, NUM

shl ecx, 2

sub ecx, 16

@cycle:

movups xmm0, xmmword ptr[eax + ecx]

movups xmm1, xmmword ptr[ebx + ecx]

mulps xmm0, xmm1

haddps xmm0, xmm0

haddps xmm0, xmm0

addss xmm2, xmm0

sub ecx, 16

cmp ecx, 0

jge @cycle

movss dword ptr[edx], xmm2

ret

MyDotProduct\_SSE endp

end

**vectfpu.asm**

.686

.xmm

.model flat, C

.code

MyDotProduct\_FPU proc result:DWORD, A:DWORD, B:DWORD, NUM:DWORD

mov eax, dword ptr[result]

mov ebx, dword ptr[A]

mov ecx, dword ptr[B]

mov edx, dword ptr[NUM]

fld dword ptr[ebx+4\*edx-2]

fmul dword ptr[ecx+4\*edx-2]

dec edx

@cycle:

fld dword ptr[ebx+4\*edx]

fmul dword ptr[ecx+4\*edx]

fadd

dec edx

cmp edx, 0

jge @cycle

fstp dword ptr[eax]

ret

MyDotProduct\_FPU endp

end

|  |  |
| --- | --- |
| **Результати: (5.5\*8.5\*1000=46 750)** | **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 2,5 рази швидше за формат FPU та у 6,8 рази 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.