Отчет по лабораторной работе №2

«Технологии векторного программирования SIMD»

по дисциплине  
«Архитектура ЭВМ»

|  |  |
| --- | --- |
| Выполнили:  студенты гр. 3530904/80004 | Захарова А. А.  Федоров Д. В.  Селезнев В. А. |
| Преподаватель | А. В. Петров |
|  |  |

«\_\_\_»\_\_\_\_\_\_\_\_\_\_2020 г.

1. Напишите программу «Изучение команд ММХ-SSE-AVX», аналогично примеру, но с новыми командами (необходимо использовать минимум две “особенные” команды и команду к AVX-12 регистрам). Объясните и покажите в отладчике visual C++ выполнение команд и изменение содержимого регистров.

“Особенными” командами можно считать команды

* с насыщением,
* сравнения,
* перестановок,
* упаковки/ распаковки,
* SSE3 и др.

2. Введите в программу три функции с форматами данных \_\_m64, \_\_m128, \_\_m256 из библиотек Cи (#include <xmmintrin.h>, <mmintrin.h>, <immintrin.h>), которые выполняют операции над описанными ранее массивами типа char и float. Покажите в Disassembler-е на регистрах выполнение команд.

Пример функции:\_\_m64 \_mm\_add\_pi8 (\_\_m64 m1 , \_\_m64 m2);

Функции у студентов должны отличаться.

3. Напишите программу работы с массивами с использованием их векторного описания и функций библиотеки **dvec**. С применением Disassembler-а определите количество машинных команд, затраченных на реализацию программы.

**Код программы**

#include <stdio.h>

#include <conio.h>

#include <xmmintrin.h>

int main(void) {

char qw1[8] = { 1, 1, 1, 1, 1, 1, 1, 1 };

char qw2[8] = { 2, 2, 2, 2, 2, 2, 2, 2 };

char qw3[8] = { 1, 1, 1, 1, 1, 1, 1, 7 };

int a = 1; int b = 2;

float c[4] = { 1, 2, 3, 4 };

float d[4] = { 5, 6, 7, 8 };

double f[2] = { 16, 4 };

char a128[16] = { 1, 18, 3, 19, 5, 21, 7, 23, 9, 25, 11, 27, 13, 29, 15, 31 };

char b128[16] = { 17, 2, 19, 4, 21, 6, 23, 8, 25, 10, 27, 12, 29, 14, 31, 16 };

\_asm { //mmx

movq mm0, qw1

movq mm1, qw2

//pcmpeqb mm0, mm1

paddb mm0, mm1 //штука сверху из изначального кода не захотела работать

movq qw1, mm0

}

printf("%s\n", "Summing elements of vectors qw1 + qw2 :");

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

{

printf("%d ", qw1[i]);

}

printf("\n");

\_\_m64 first = \*(\_\_m64\*)qw1;

\_\_m64 second = \*(\_\_m64\*)qw2;

\_\_m64 result = \_mm\_add\_pi8(first, second);

char\* resCheck = (char\*)&result;

printf("%s\n", "Summing elements of vectors qw1 + qw2 2:");

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

{

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

}

float res[4];

printf("\n%s\n", "\* \* \* \* \* SSE3 \* \* \* \* \*");

\_asm {

movups xmm0, c

movups xmm1, d

addsubps xmm0, xmm1 //рандомная штучка с сайта. складывает значения на нечетных позициях (если считать с нуля) и вычитает на четных

movups res, xmm0

}

\_mm\_empty(); //это штучка без которой не работал код из лабораторного практикума

//может можно было оживить его как-то по-другому, но у меня заработало так и я успокоилась

for (int i = 0; i < 4; i++)

{

printf("%f ", res[i]);

}

printf("\n");

printf("\n%s\n", "\* \* \* \* \* COMPARISON \* \* \* \* \*");

\_asm {

movups xmm0, c //1 2 3 4

movups xmm1, d //5 6 7 8

cmpss xmm0, xmm1, 0 //сравниваем первую цифирьку и если они разные, то в res на месте первого элемента 0

movups res, xmm0

}

\_mm\_empty();

for (int i = 0; i < 4; i++)

{

printf("%f ", res[i]);

}

printf("\n");

\_mm\_empty();

printf("\n%s\n", "\* \* \* \* \* TRANSPOSITION \* \* \* \* \*");

\_asm {

movups xmm0, c //1 2 3 4

movups xmm1, d //5 6 7 8

movhlps xmm0, xmm1 //переместили последние два значения d на первые два значения в c

movups res, xmm0 //7 8 3 4

}

for (int i = 0; i < 4; i++)

{

printf("%f ", res[i]);

}

printf("\n");

\_\_m128 first2 = \*(\_\_m128\*)c;

\_\_m128 second2 = \*(\_\_m128\*)d;

\_\_m128 ress = \_mm\_movehl\_ps(second2, first2);

float\* resss = (float\*)& ress;

printf("\n%s\n", "\* \* \* \* \* TRANSPOSITION 2 \* \* \* \* \*");

for (int i = 0; i < 4; i++)

{

printf("%f ", resss[i]);

}

\_asm { //sse

movups xmm0, c

movups xmm1, d

addps xmm0, xmm1

movups c, xmm0

}

\_mm\_empty();

printf("%s", "\nSumming elements of vectors c + d :\n");

for (int i = 0; i < 4; i++)

{

printf("%f ", c[i]);

}

printf("\n");

\_asm { //sse2

movups xmm1, f

sqrtpd xmm0, xmm1

movups f, xmm0

}

\_mm\_empty();

printf("\n%s %f %s %f\n", "Square of ", f[0], "is", f[1]);

\_asm {

movups xmm0, a128

movups xmm1, b128

pminub xmm0, xmm1

movups a128, xmm0

}

printf("\n%s\n", "Comparing elements :");

for (int i = 0; i < 16; i++)

{

printf("( %d , %d) ; ", a128[i], b128[i]);

}

printf("\n");

printf("\n%s\n", "Minimum elements :");

for (int i = 0; i < 16; i++)

{

printf("%d ", a128[i]);

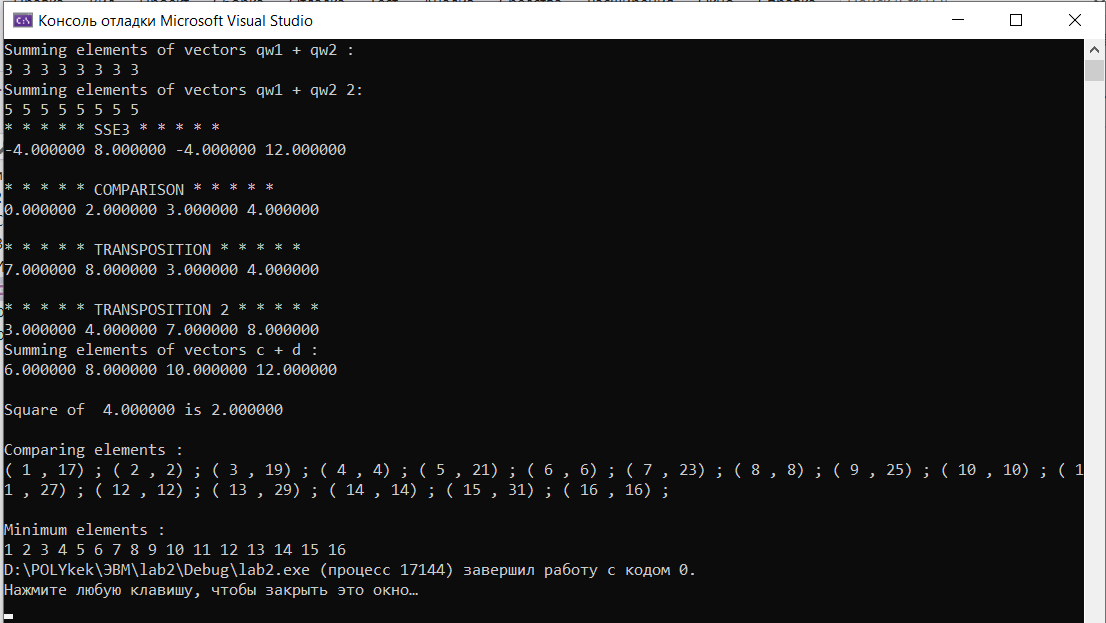
}

return 0;

\_getch();

}

**Результат работы программы**

****