БГУИР

Кафедра ЭВМ

Отчет по лабораторной работе №2

Тема: «Команды MMX/XMM»

Выполнил:

студент группы 750501 Новицкий А.Д.

Проверил:

к.т.н., доцент Одинец Д.Н.

Минск

2019

**СОДЕРЖАНИЕ**

1. Постановка задач………………………………………………………………..3

2. Алгоритм решения задачи……………………………………………………...4

3. Листинг программы…………………………………………………………..5-7

4. Тестовые примеры……………………………………………………………...8

5. Заключение……………………………………………………………………...9

**ПОСТАНОВКА ЗАДАЧ**

Создать консольное приложение, которое выполняет операцию побитового И для каждой пары соответствующих элементов тремя способами:

1) с использованием команд MMX

2) на ассемблере, без использования команд MMX

3) на языке Си

После вычислений должны быть выведены время выполнения и результат для каждого случая.

Значения элементов матриц генерируются приложением (не вводятся с клавиатуры). Вычисления производятся несколько (1 млн) раз. Размер матриц (векторов) кратен количеству элементов в регистре ММХ.

**АЛГОРИТМ РЕШЕНИЯ ЗАДАЧИ**

1. Прибавить к полученному значению единицу
2. Вычислить косинус из полученного значения

**ЛИСТИНГ ПРОГРАММЫ**

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <time.h>

#include <conio.h>

#include <windows.h>

void main(int argc, char \* argv[])

{

long double a = 0, b = 0, step = 0, x, f;

int one = 1;

double t;

LARGE\_INTEGER timerFrequency, timerStart, timerStop;

while (true) {

printf("Input 'a': ");

scanf("%lf", &a); // считываем переменную а - начало интервала

printf("\nInput 'b': ");

scanf("%lf", &b); // считываем переменную b - конец интервала

printf("\nInput 'step': ");

scanf("%lf", &step); // считываем переменную step - шаг увеличения

if (a >= b)

printf("\nYou are stupid! "); // если конец интервала меньше чем начало, то пользователь

// слегка глупый

else

break;

}

x = a;

f = 0;

QueryPerformanceFrequency(&timerFrequency); // получение частоты счетчика в переменную timerFrequency

QueryPerformanceCounter(&timerStart); // получение занчения счетчика на момент старта в

// переменную timerStart

while(x <= b)

{

f += sin(cos(sqrt(x) + 1));

x += step;

};

QueryPerformanceCounter(&timerStop); // получение занчения счетчика на момент завершения в

// переменную timerStop

t = (double)(timerStop.QuadPart - timerStart.QuadPart) / (double)timerFrequency.QuadPart;

// вычисление прошедшего времени

printf("\nTime C is %lf seconds.\n", t);

printf("%lf\n", f);

x = a;

f = 0;

QueryPerformanceCounter(&timerStart);

// получение занчения счетчика на момент старта в

// переменную timerStart

\_asm {

finit //инициализация математическго сопроцессора

fld b //ST(0) = b

fld x //ST(0) = x

//ST(1) = b

loop\_start :

fcom // сравниваем ST(0) и ST(1) (x и b)

fstsw ax // регистр SR -> AX

and ah, 01000101b // if(x>b)[C3=1, C2 = 1, C1 =1] loop\_end

jz loop\_end

fld x //ST(0) = x

//ST(1) = x

//ST(1) = b

fsqrt //ST(0) = sqrt(x)

//ST(1) = x

//ST(1) = b

fiadd one // ST(0) = sqtr(x)+1

// ST(1) = x

// ST(2) = b

fcos // ST(0) = cos(sqrt(x)+1)

// ST(1) = x

// ST(2) = b

fsin // ST(0) = sin(cos(sqrt(x)+1))

// ST(1) = x

// ST(2) = b

fadd f // ST(0) = sin(cos(sqrt(x)+1)) + f

// ST(1) = x

// ST(2) = b

fstp f // F = (sin(x+1) / cos(x) + 2) + f

// ST(0) = x

// ST(1) = b

fadd step // ST(0) = x + step

// ST(1) = b

fst x // X = ST(0)

// ST(0) = x + step

// ST(1) = b

jmp loop\_start // goto loop\_start

loop\_end :

fwait // ожидает завершение команд FPU

}

QueryPerformanceCounter(&timerStop);

// получение занчения счетчика на момент завершения в // переменную timerStop

t = ((double)timerStop.QuadPart - timerStart.QuadPart) / (double)timerFrequency.QuadPart;

// вычисление прошедшего времени

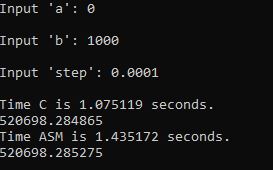
printf("Time ASM is %lf seconds.\n", t);

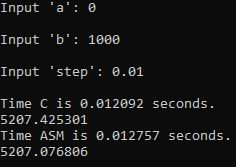
printf("%lf\n", f);

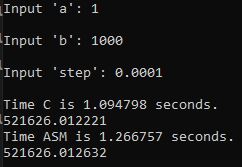
\_getch();

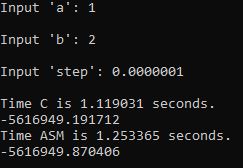
}

**ТЕСТОВЫЕ ПРИМЕРЫ**

****

****

****

****

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения лабораторной работы были получены базовые навыки работы с математическим сопроцессором.

Были получены знания для чего нужен сопроцессор, а именно для параллельных и более точных вычислений данных с плавающей запятой. Также узнал о необходимости синхронизации сопроцессора с процессором для того, чтобы процессор не получил неверные данные от сопроцессора, так как он работает параллельно, для этого используется команда FWAIT.

Были изучены такие команды, как: FLD (загрузка значения в стек), FST (получение команды из стека без выгрузки), FADD (добавить значение к ST(0)), FSUB (вычитание из ST(0)), FMUL (умножение ST(0)), FDIV (деление ST(0)), FSQRT (вычисление квадратного корня из ST(0), FABS (получение модуля ST(0)), FCHS (поменять знак ST(0)), FCOM (сравнить ST(0) и ST(1)), FSTSW (выгрузить значение регистра SW), FPTAN (вычисление тангенса).

Были изучены регистры: состояния (SW), управления (CW), числовые (ST(0) - ST(7)), регистр тэгов (TAG(0) – TAG(7)), указателя команд (FID), указателя операнда (FDP).

SW – выполняет функции в чем-то аналогичные функциям регистра флагов (E)FLAGS, но только для операций с плавающей точкой.

CW – предназначен для управления режимами работы сопроцессора. Он содержит ряд полей, значения которых влияют на точность результата, поведение процессора при встрече некорректных операндов и результатов и т.п.

Числовые регистры – служат для хранения данных над которыми будут выполняться математические операции.

Регистры тэгов - описывает содержимое всех числовых регистров.

FID – указывает на текущую исполняемую команду.

FDP – хранит адрес операнда.