Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Кафедра электронных вычислительных машин

Лабораторная работа №2

«Команды MMX/XMM»

Выполнил: Проверил:

Студент группы 950501 Преподаватель

Деркач А.В. Одинец Д.Н.

Минск, 2021

1. Постановка задачи

Создать приложение, которое выполняет вычисления (Вариант 5: Копирование элементов одной матрицы в другую) четырьмя способами:

1. с использованием команд MMX
2. на ассемблере, без использования команд MMX
3. на языке Си
4. с использованием инструкций SSE

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

Значения элементов матриц генерируются приложением (не вводятся с клавиатуры). Вычисления производятся многократно (например, 1 млн раз). Размер матриц (векторов) кратен количеству элементов в регистре MMX.

1. Алгоритм

Программа состоит из нескольких подпрограмм (частей программы), представляющих собой некоторые функции. К ним относятся функции:

* Инициализация матрицы
* Копирование одной матрицы в другую
* Вывод результата
* Вывод времени, затраченного на выполнение алгоритма.

1. Листинг программы

Далее приведен листинг программы, реализующей различные способы выполнения вычислений.

#include "stdio.h"

#include "windows.h"

#define SIZE 4

int main() {

int ms1[SIZE][SIZE], ms2[SIZE][SIZE];

int time0, time1, time2, time3, time4;

long cnt = SIZE\*SIZE;

for (int i = 0; i < SIZE; i++) {

for (int j = 0; j < SIZE; j++) {

ms1[i][j] = i \* SIZE + j \* SIZE;

}

}

printf("Entered matrix: \n");

for (int i = 0; i < SIZE; i++) {

for (int j = 0; j < SIZE; j++) {

printf("%d ", ms1[i][j]);

}

printf("\n");

}

time0 = GetTickCount();

printf("\nMMX:\n");

for (int i = 0; i < 10000000; i++)

{

cnt = 16;

\_\_asm

{

push esi

push edi

mov ecx, [cnt]

lea esi, [ms1 + ecx \* 2]

lea edi, [ms2 + ecx \* 2]

neg ecx

loop1:

movq mm0, [esi + ecx \* 2]

movq [edi + ecx \* 2], mm0

inc ecx

jnz loop1

emms

pop edi

pop esi

}

}

printf("Matrix was copied...\n");

time1 = GetTickCount() - time0;

printf("Time = %d ms\n", time1);

printf("\nAssembly:\n");

for (int i = 0; i < 10000000; i++)

{

cnt = 16;

\_\_asm

{

push eax

push ecx

push esi

xor esi, esi

xor ecx, ecx

loop2:

movsx eax, ms1[esi]

mov ms2[esi], eax

add cx, ax

add esi, 2

sub cnt, 1

jnz loop2

pop esi

pop ecx

pop eax

}

}

printf("Matrix was copied...\n");

time2 = GetTickCount() - time0 - time1;

printf("Time = %d ms\n", time2);

printf("\nC:\n");

for (int i = 0; i < 10000000; i++)

{

for (int i = 0; i < SIZE; i++) {

for (int j = 0; j < SIZE; j++) {

ms2[i][j] = ms1[i][j];

}

}

}

printf("Matrix was copied...\n");

time3 = GetTickCount() - time0 - time1 - time2;

printf("Time = %d ms\n", time3);

printf("\nSSE:\n");

for (int i = 0; i < 10000000; i++)

{

cnt = 16;

\_\_asm

{

push esi

push edi

mov ecx, [cnt]

lea esi, [ms1 + ecx \* 2]

lea edi, [ms2 + ecx \* 2]

neg ecx

loop3:

movups xmm0, [esi + ecx \* 2]

movups [edi + ecx \* 2], xmm0

inc ecx

jnz loop3

emms

pop edi

pop esi

}

}

printf("Matrix was copied...\n");

time4 = GetTickCount() - time0 - time1 - time2-time3;

printf("Time = %d ms\n", time4);

return 0;

}

1. Тестирование программ

Было произведено 3 тестирования с разной нагрузкой процессора:



Рисунок 4.1 – Тест 1



Рисунок 4.2 – Тест 2



Рисунок 4.3 – Тест 3

1. Заключение

В ходе лабораторной работы удалось выполнить копирование одной матрицы в другую, используя разные способы выполнения вычислений.

Программа была запущена в среде разработки Visual Studio 2019 на 64-ех разрядной Windows 10.

По результатам лабораторной работы видно, что медленнее всего алгоритм выполняется на языке C, а быстрее всего, при использовании команд MMX.