УО БГУИР
Кафедра ЭВМ

Отчет по лабораторной работе
Тема: "Команды MMX/XMM"

Выполнил:
студент группы 950503
Полховский А.Ф.

Проверил:
к.т.н., доцент Одинец Д.Н.

Минск 2021

1. Постановка задачи

Вариант 12: Сформировать массив средних значений (результат[i] = avg(массив1[i], массив2[i])) *pavgb, pavgw.

Цель работы: Изучить расширение системы команд MMX процессоров Intel.

Создать консольное приложение, которое выполняет вычитание матриц тремя способами:

1) с использованием команд MMX

2) на ассемблере, без использования команд MMX

3) на языке Си

4)с использованием команд XMM (SSE)

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

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

2. Алгоритм решения задачи

Для того, чтобы реализовать поставленные задачи необходимо:

1. Создать консольное меню взаимодействия с пользователем
2. Реализовать проверку введённых значений на соответствие допустимым
3. Реализовать вычисления на языках Си, Ассемблера, Ассемблера с использованием MMX
4. Произвести замер производительности вычислений с помощью реализованных функций

3. Листинг программы

```
#include <iostream>
#include <random>
#include <time.h>
#include <iomanip>
#include <list>
#define s 16

const int COUNT_CYCLE = 10000000;
const int SIZE = 16;
const int RAND = 10000;
using namespace std;

void output(int ms[SIZE][SIZE])
{
	for (int i = 0; i < SIZE; i++)
	{
		for (int j = 0; j < SIZE; j++)
		{
			cout << ms[i][j] << ' ';
```

```
                }
                cout << endl;
        }
}

void null(int ms[SIZE][SIZE])
{
        for (int i = 0; i < SIZE; i++)
        {
                for (int j = 0; j < SIZE; j++)
                {
                        ms[i][j] = 0;
                }
        }
}

void rand(int ms[SIZE][SIZE])
{
        for (int i = 0; i < SIZE; i++)
        {
                for (int j = 0; j < SIZE; j++)
                {
                        ms[i][j] = rand() % RAND;
                }
        }
}

void rand(float ms[SIZE][SIZE])
{
        for (int i = 0; i < SIZE; i++)
        {
                for (int j = 0; j < SIZE; j++)
                {
                        ms[i][j] = rand() % RAND;
                }
        }
}

void output(float ms[SIZE][SIZE])
{
        for (int i = 0; i < SIZE; i++)
        {
                for (int j = 0; j < SIZE; j++)
                {
                        cout << ms[i][j] << ' ';
                }
                cout << endl;
        }
}

void toFloat(float ms1[SIZE][SIZE], int ms2[SIZE][SIZE])
{
        for (int i = 0; i < SIZE; i++)
        {
                for (int j = 0; j < SIZE; j++)
                {
                        ms1[i][j] = ms2[i][j];
                }
        }
}

int main()
{
        srand(time(NULL));
        clock_t start;
        clock_t end;
        int ms1[SIZE][SIZE];
        int ms2[SIZE][SIZE];
        int result_ms[SIZE][SIZE] = { 0 };
```

```
list<double> time;

rand(ms1);
rand(ms2);

/*Pure C*/
start = clock();
for (int i = 0; i < COUNT_CYCLE; i++)
{
	for (int i = 0; i < SIZE; i++)
	{
		for (int j = 0; j < SIZE; j++)
		{
			result_ms[i][j] = ms1[i][j] - ms2[i][j];
		}
	}
}
end = clock();
output(result_ms);
time.push_back((double)(end - start) / CLOCKS_PER_SEC);
cout << endl << "Pure C: " << (double)(end - start) / CLOCKS_PER_SEC << endl << endl;

/*Asm*/
null(result_ms);
start = clock();
for (int i = 0; i < COUNT_CYCLE; i++)
{
	_asm {
		xor esi, esi
		xor eax, eax
		mov ecx, s * s

		cycleAsm:
			mov eax, ms1[esi]
			sub eax, ms2[esi]
			mov result_ms[esi], eax
			add esi, 4
			loop cycleAsm
	}
}
end = clock();
output(result_ms);
time.push_back((double)(end - start) / CLOCKS_PER_SEC);
cout << endl << "ASM: " << (double)(end - start) / CLOCKS_PER_SEC << endl << endl;

/*MMX*/
null(result_ms);
start = clock();
for (int i = 0; i < COUNT_CYCLE; i++)
{
	_asm {
		pusha
		xor esi, esi
		mov ecx, s*s/2

		cycleMMX:
		movq MM0, ms1[esi]
			psubq MM0, ms2[esi]
			movq result_ms[esi], MM0
			add esi, 8
			loop cycleMMX
			emms
			popa
	}
}
end = clock();
output(result_ms);
time.push_back((double)(end - start) / CLOCKS_PER_SEC);
cout << endl << "MMX: " << (double)(end - start) / CLOCKS_PER_SEC << endl << endl;
```

```
	/*XMM*/
	float arr1[SIZE][SIZE];
	float arr2[SIZE][SIZE];
	float arr3[SIZE][SIZE];
	toFloat(arr1, ms1);
	toFloat(arr2, ms2);

	null(result_ms);
	start = clock();
	for (int i = 0; i < COUNT_CYCLE; i++)
	{
		_asm {
			pusha
			xor ecx,ecx
			xor esi, esi
			mov ecx, s * s / 4

			cycleXMM:
				movups XMM0, arr1[esi]
				subps XMM0, arr2[esi]
				movups arr3[esi], XMM0
				add esi, 16
				loop cycleXMM
				emms
				popa
		}
	}
	end = clock();
	output(arr3);
	time.push_back((double)(end - start) / CLOCKS_PER_SEC);
	cout << endl << "XMM: " << (double)(end - start) / CLOCKS_PER_SEC << endl << endl;

	cout << "|" << setw(5) << "C++ " << "|" << setw(5) << "ASM " << "|" << setw(5) << "MMX
" << "|" << setw(5) << "XMM " << "|" << endl;
	for (size_t i = 0; i < 4; i++)
	{
		cout << "|" << setw(5) << time.front();
		time.pop_front();
	}
	cout << "|" << endl;

	system("pause");
	return 0;
}
```

4. Тестовые пример

C++	ASM	MMX	XMM
1.104	5.208	1.947	0.981

5. Заключение

В ходе выполнения лабораторной работы были получены результаты выполнения вычитания матриц, реализованной разными способами: средствами языка Си, ассемблера и ассемблера с использованием MMX. Время, затраченное на выполнение вычислений функцией, реализованной с помощью команд MMX меньше аналогичных, реализованных на языке Си и ассемблере.