Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Кафедра электронных вычислительных машин

Отчёт

по лабораторной работе № 2

на тему

«Команды MMX/XMM»

|  |  |
| --- | --- |
| Студент группы № 950503 | Гуринович А. В. |
| Преподаватель | Одинец Д. Н. |

Минск 2021

1. Постановка задачи.

Создать приложение, которое выполняет заданные вычисления (в соответствии с вариантом) тремя способами:

* 1. с использованием команд MMX;
  2. с использованием команд XMM (SSE);
  3. на Ассемблере, без использования команд MMX, XMM;
  4. на языке C.

После вычислений должны быть выведены время выполнения (интервал от начала выполнения вычислений до их завершения) и результат вычислений (конечные значения) 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 ММХ(XMM).

Вариант: 5.

Задание: перенос значений одной матрицы в другую.

1. 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, Ассемблера с использованием XMM.
4. Произвести замер производительности вычислений с помощью реализованных функций.
5. Листинг программы

include <chrono>

**class** Timer {

**private**:

**using** clock\_t = std::chrono::high\_resolution\_clock;

**using** second\_t = std::chrono::duration<**double**, std::ratio<1> >;

std::chrono::time\_point<clock\_t> m\_beg;

**public**:

Timer() : m\_beg(clock\_t::now()) {}

**void** reset(){

m\_beg = clock\_t::now();

}

**double** elapsed() **const** {

**return** std::chrono::duration\_cast<second\_t>(clock\_t::now() - m\_beg).count();

}

};

#include <iostream>

#include <iomanip>

#include "Timer.h"

**using** **namespace** std;

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

**const** **int** matrixHeight = 16;

**const** **int** matrixWidth = 16;

**const** **int** numberOfOperations = 100000;

cout << "Test values: matrixHeight = " << matrixHeight << endl

<< " matrixWidth = " << matrixWidth << endl

<< " numberOfOperations = " << numberOfOperations << endl;

cout << "| C++ | Assembley | MMX | SSE |" << endl;

**int** matrix[matrixHeight][matrixWidth];

**int** newMatrix[matrixHeight][matrixWidth];

**for**(**int** i = 0; i < matrixHeight; i++){

**for**(**int** j = 0; j < matrixWidth; j++){

matrix[i][j] = rand();

}

}

Timer timer;

**double** resultTime;

**do** {

int32\_t temp32a;

**int**\* pointerTemp32a = (**int**\*)&temp32a;

int32\_t temp32b;

**int**\* pointerTemp32b = (**int**\*)&temp32b;

resultTime = 0;

//C++

**for**(**int** d = 0; d < numberOfOperations; d++){

**for**(**int** i = 0; i < matrixHeight; i++){

**for**(**int** j = 0; j < matrixWidth; j++){

timer.reset();

\*pointerTemp32a = matrix[i][j];

temp32b = temp32a;

newMatrix[i][j] = \*pointerTemp32b;

resultTime += (timer.elapsed());

}

}

}

cout << '|' << fixed << setprecision(9) << resultTime << '|';

//Assembly

resultTime = 0;

**for**(**int** d = 0; d < numberOfOperations; d++){

**for**(**int** i = 0; i < matrixHeight; i++){

**for**(**int** j = 0; j < matrixWidth; j++) {

\*pointerTemp32a = (matrix[i][j]);

timer.reset();

**\_\_asm** {

mov eax, temp32a

mov temp32b, eax

}

resultTime +=timer.elapsed();

newMatrix[i][j] = \*pointerTemp32b;

}

}

}

cout << fixed << setprecision(9) << resultTime << '|';

//MMX

resultTime = 0;

int64\_t temp64a;

**int**\* pointerTemp64a = (**int**\*)&temp64a;

int64\_t temp64b;

**int**\* pointerTemp64b = (**int**\*)&temp64b;

**for**(**int** d = 0; d < numberOfOperations; d++){

**for**(**int** i = 0; i < matrixHeight; i++) {

**for**(**int** j = 0; j < matrixWidth; j++) {

pointerTemp64a[0] = (matrix[i][j]);

pointerTemp64a[1] = (matrix[i][j + 1]);

timer.reset();

**\_\_asm** {

movq mm0, temp64a

movq temp64b, mm0

EMMS

}

resultTime +=timer.elapsed();

newMatrix[i][j++] = \*(pointerTemp64b);

newMatrix[i][j] = \*((pointerTemp64b) + 1);

}

}

}

cout << fixed << setprecision(9) << resultTime << '|';

//SSE

resultTime = 0;

\_\_int128 temp128a;

**int**\* pointerTemp128a = (**int**\*)&temp128a;

\_\_int128 temp128b;

**int**\* pointerTemp128b = (**int**\*)&temp128b;

**for**(**int** d = 0; d < numberOfOperations; d++){

**for**(**int** i = 0; i < matrixHeight; i++) {

**for**(**int** j = 0; j < matrixWidth; j++) {

pointerTemp128a[0] = (matrix[i][j]);

pointerTemp128a[1] = (matrix[i][j + 1]);

pointerTemp128a[2] = (matrix[i][j + 2]);

pointerTemp128a[3] = (matrix[i][j + 3]);

timer.reset();

**\_\_asm** {

movups xmm0, temp128a

movups temp128b, xmm0

}

resultTime +=timer.elapsed();

newMatrix[i][j++] = \*(pointerTemp128b);

newMatrix[i][j++] = \*((pointerTemp128b) + 1);

newMatrix[i][j++] = \*((pointerTemp128b) + 2);

newMatrix[i][j] = \*((pointerTemp128b) + 3);

}

}

}

cout << fixed << setprecision(9) << resultTime << '|';

**if**(getchar() == 'Q'){

**break**;

}

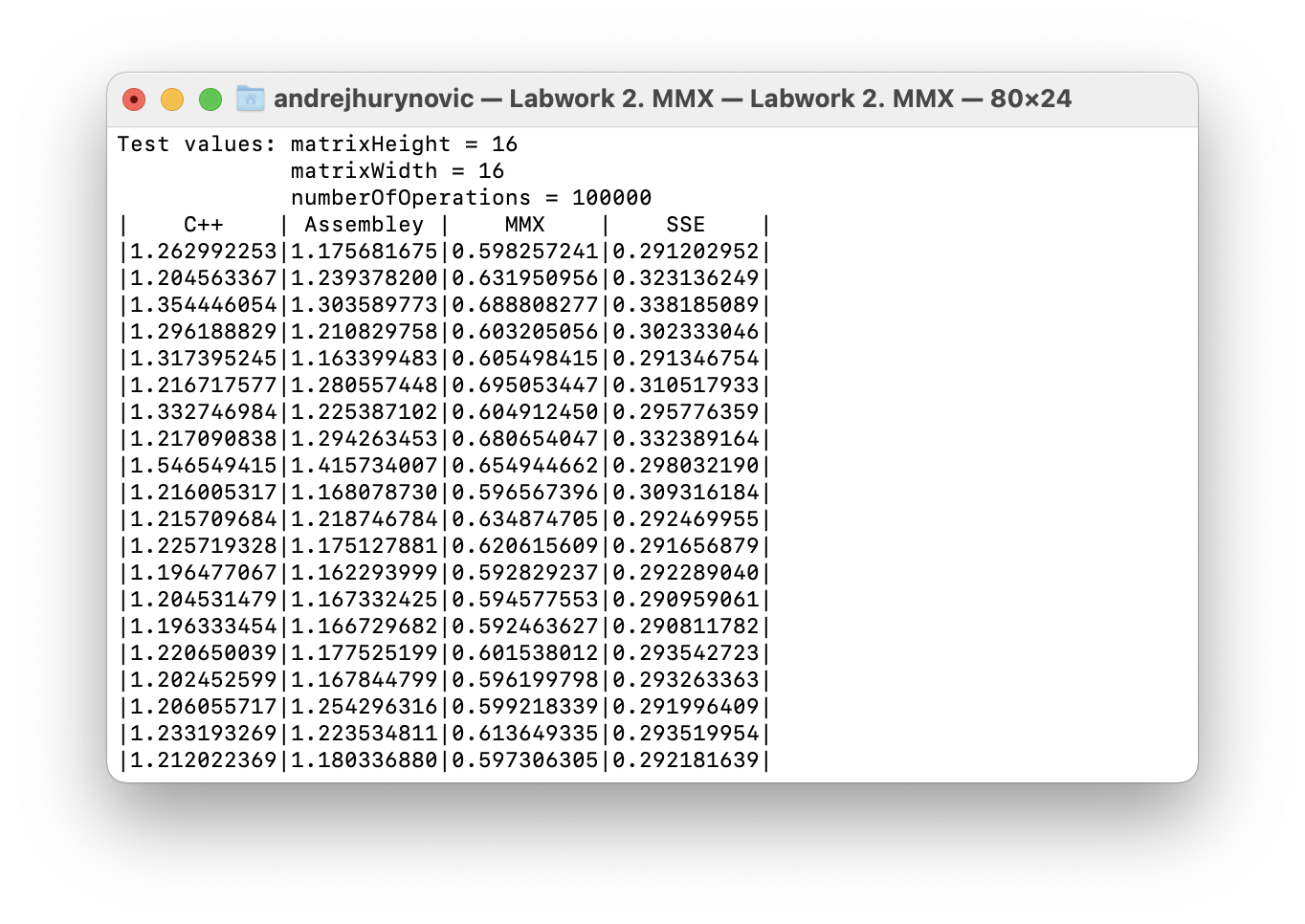
}**while**(**true**);

**return** 0;

}

1. Тестовые пример

Выводятся тестовые значения: размер матрицы, количество операций



1. Заключение

В ходе выполнения лабораторной работы были получены результаты выполнения вычитания матриц, реализованной разными способами: средствами языка Си, ассемблера, ассемблера с использованием MMX и ассемблера с использованием XMM.