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

Крылов А.В. Одинец Д.Н.

Минск, 2021

1. **Постановка задачи**
2. Создать приложение, которое выполняет вычисления (Вариант 14: Выполнить операцию побитового исключающего ИЛИ для каждой пары соответствующих элементов) тремя способами:
3. с использованием команд MMX
4. на ассемблере, без использования команд MMX
5. 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 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 матриц

Выполнение операции побитового ИЛИ для элементов матриц

Вывод результата

Вывод времени, затраченного на выполнение

1. **Листинг программы**

Далее приведен листинг программы, реализующей различные способы выполнения вычислений.

#include <time.h>

#include <iostream>

using namespace std;

#define SIZE 4

#define COUNTER 1000000 //количество вычислений

void printArray(int array[SIZE][SIZE]); //вывод массива

int main()

{

srand(time(NULL));

int array1[SIZE][SIZE], array2[SIZE][SIZE], array3[SIZE][SIZE];

for (int i = 0; i < SIZE; i++) //ввод матрицы

{

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

{

//заполнение матриц рандомными значениями

array1[i][j] = rand() % 100;

array2[i][j] = rand() % 100;

}

}

//вывод исходных матриц

printf("Array 1:\n");

printArray(array1);

printf("Array 2:\n");

printArray(array2);

// рассчет на языке си

clock\_t begin\_c = clock(); //начинаем отсчет времени

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

{

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

{

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

{

array3[i][j] = array1[i][j] | array2[i][j]; //применяем операцию побитового или |

}

}

}

clock\_t end\_c = clock();

//выводим результат

printf("Result using C:\n");

printArray(array3);

printf("\n");

// рассчет с использованием MMX

clock\_t begin\_mmx = clock();

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

{

int cnt = 8;

\_asm

{

MOV EBX, cnt // цикл - 8 итерация(т.к. в массиве 16 элементов размером

// 4 байта = 64, а MMX позволяет за одну инструкцию обработать 8 байт)

XOR ESI, ESI // сброс флагов

STARTM :

MOVQ MM0, array1[ESI] // запись данных в MMX регистр

MOVQ MM1, array2[ESI] // запись данных в MMX регистр

POR MM0, MM1 // операция логического ИЛИ

MOVQ array3[ESI], MM0 //записываем результат в 3 массив

ADD ESI,8 // передвигаем указатель на 4 байта

DEC EBX //уменьшаем количество проходов

CMP EBX,0 // сравниваем с нулем

JNE STARTM

EMMS // очищаем регистры

}

}

clock\_t end\_mmx = clock();

//вывод результата

printf("Result using MMX:\n");

printArray(array3);

printf("\n");

// рассчет на ассемблере

clock\_t begin\_asm = clock();

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

{

int cnt = 16;

\_\_asm

{

XOR ESI, ESI// сброс флагов, индекс источника

MOV EBX, cnt

START :

MOV EAX, array1[ESI] // EAX = array1[ESI], аккумулятор

MOV EDX, array2[ESI] // EDX = array2[ESI], регистр данных

OR EDX, EAX // операция логического ИЛИ

MOV array3[ESI], EDX // array3[ESI] = DX - запись результата

ADD ESI, 4

DEC EBX

CMP EBX, 0

JNE START

}

}

clock\_t end\_asm = clock();

//вывод результата

printf("Result using ASM:\n");

printArray(array3);

printf("\n");

//вывод затраченного на выполнение времени

printf("Computing using C\n");

printf("time: %.6lf sec\n\n", (float)(end\_c - begin\_c) / CLOCKS\_PER\_SEC);

printf("Computing using ASM\n");

printf("time: %.6lf sec\n\n", (float)(end\_asm - begin\_asm) / CLOCKS\_PER\_SEC);

printf("Computing using MMX\n");

printf("time: %.6lf sec\n\n", (float)(end\_mmx - begin\_mmx) / CLOCKS\_PER\_SEC);

system("pause");

return 0;

}

void printArray(int array[SIZE][SIZE])

{

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

{

printf("\t");

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

{

printf("\t %d ", array[i][j]);

}

printf("\n");

}

printf("\n");

}

4

1. **Тестирование программ**

Было произведено 3 тестирования с разной нагрузкой процессора:

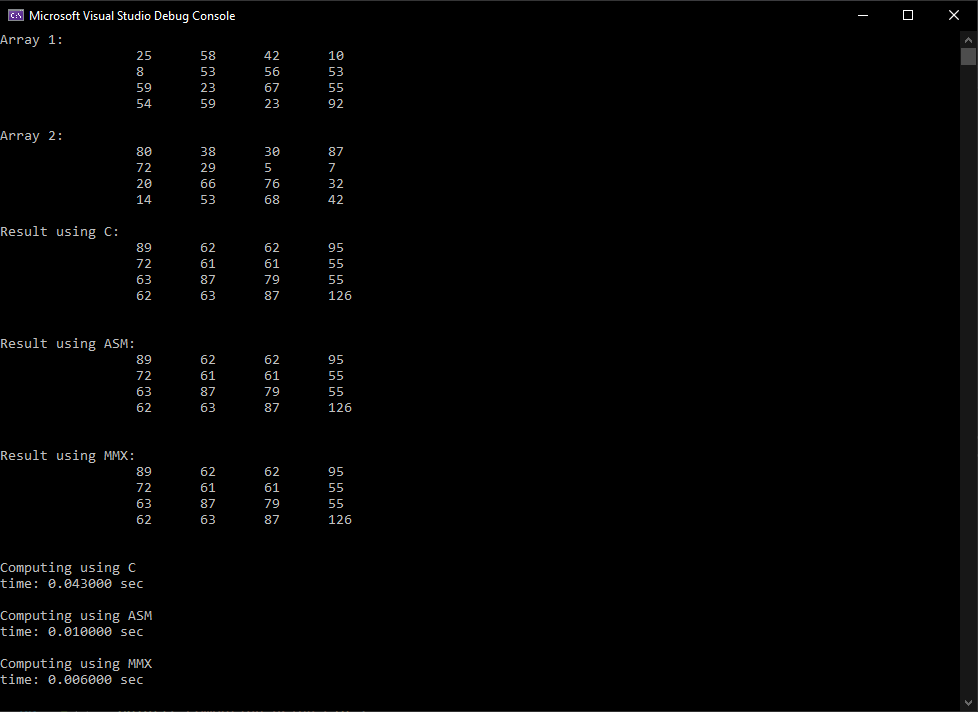


Рисунок 4.1 – Тест 1

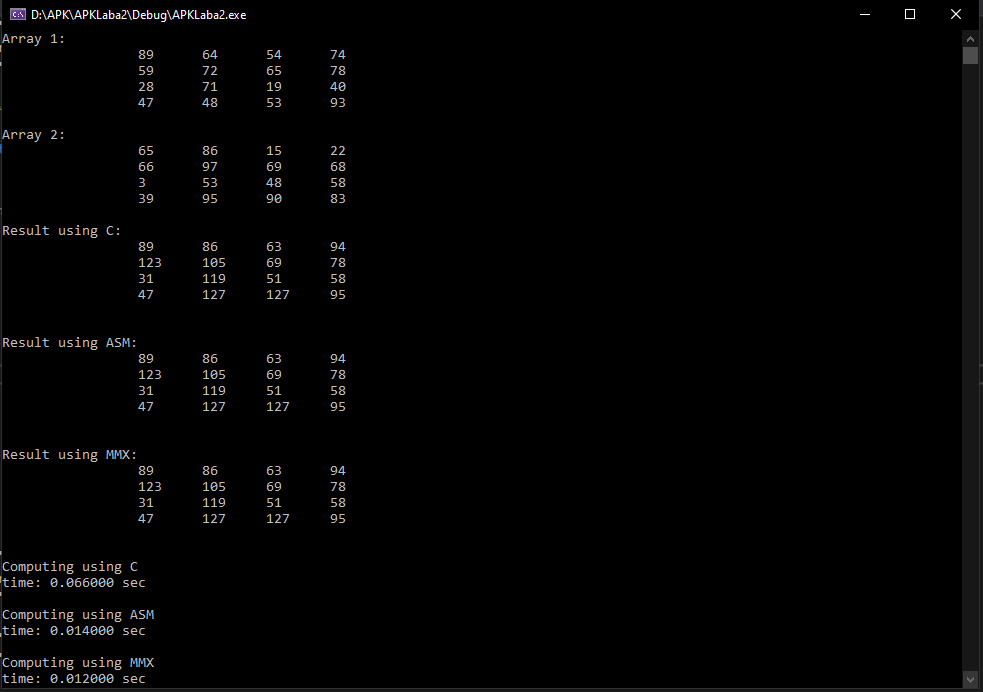


Рисунок 4.2 – Тест 2

5

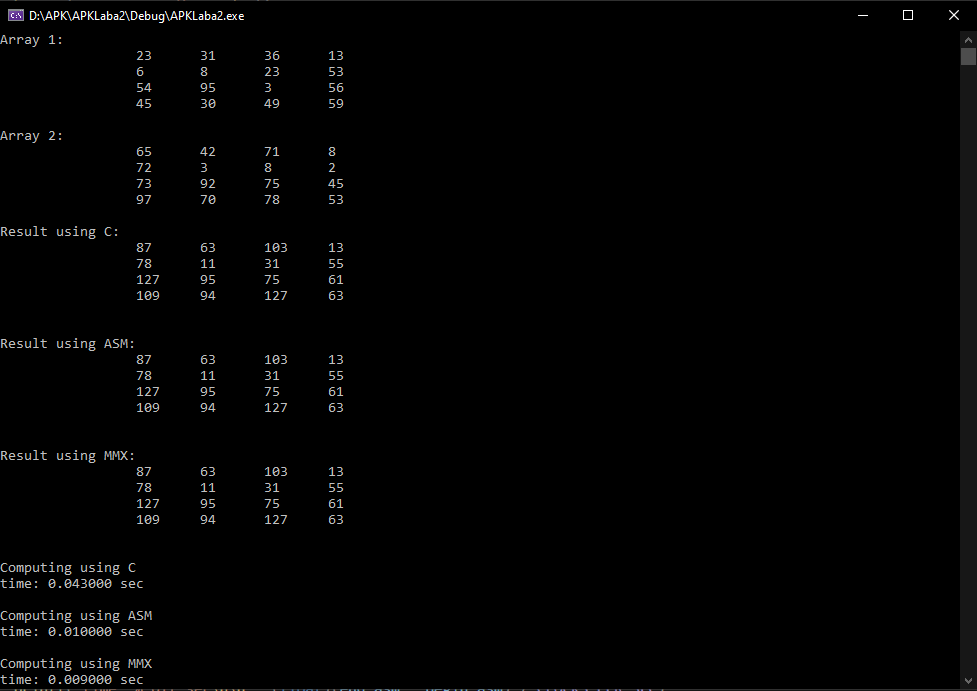


Рисунок 4.3 – Тест 3

* + 1. **Заключение**
  + ходе лабораторной работы удалось выполнить операцию побитового ИЛИ

для соответствующих элементов двумерных массивов.

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

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

6