Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Томский государственный университет систем управления и радиоэлектроники»

Факультет систем управления (ФСУ)

Кафедра автоматизированных систем управления (АСУ)

Отчет по лабораторной работе №4 по дисциплине

«Вычислительная техника»

Обучающийся гр. 431-3

_____Андреев.Д.П.

« __»_____ 2022 г.

Проверил: доцент кафедры АСУ

_____Алфёров.С.М,

«__ »_____ 2022

Томск 2022

Оглавление

1.Цель работы

Научиться обрабатывать массивы данных на языке Ассемблер. Познакомиться с векторными операциями процессора.

2.Задание

Задание состоит из трёх частей. В первой части нужно выполнить задание на языке c++. Во второй части требуется написать программу обработки массива согласно заданию, используя скалярные команды обработки данных. Во второй части требуется написать такую же программу, но используя векторные операции. Для каждой программы засечь время выполнения, провести не менее 100 замеров. Вычислить среднее время выполнения для каждой программы и сделать вывод об эффективности векторных операций. На выполнение всего задания выделяется 8 часов аудиторного времени.

Вариант№1:

Увеличить яркость красной составляющей верхней половины картинки.

3.Текст программы

```
#include <iostream>
#include <stdlib.h>
#include <conio.h>
#include <locale.h>
using namespace std;

int main()
{
	setlocale(LC_ALL, "Ru");
	int time_cpp = 0;
	int time_scal = 0;
	int time_vec = 0;
	time_t start;
	FILE* in, * out;
	unsigned __int8* buffer;
	unsigned __int32 wight, height;
	//Блок C++
	for (int i = 0; i < 100; i++)
	{
		cout << "Loadeing 0: " << i << "%" <<
endl;
		start = clock();//Старт замера
		fopen_s(&in, "in.bmp", "rb");
		buffer = (unsigned __int8*)malloc(54);
		fread(buffer, 1, 54, in);
		for (int i = 0; i <= 3; i++)
		{
			wight = buffer[21 - i] | wight <<
8;
			height = buffer[25 - i] | height <<
8;
		}
		fopen_s(&out, "OUT_cpp.bmp", "wb");
		fwrite(buffer, 1, 54, out);
		free(buffer);

		//Редактирование картинки
		buffer = (unsigned __int8*)malloc(wight *
height * 4);
		fread(buffer, 1, wight * height * 4, in);
		fclose(in);

		for (int i = 0; i < wight; i += 128)
		{
			for (int j = height; j > height / 2; j-
-)
			{
				for (int k = i; k < wight;
k++)
				{
					buffer[wight * 3
* j + k * 3 + 2] = 255;
					buffer[wight * 3
* j + k * 3 + 1] = 0;
					buffer[wight * 3
* j + k * 3 + 0] = 0;
				}
			}
		}
		fwrite(buffer, 1, wight * height * 4, out);
		fclose(out);
		free(buffer);
		time_cpp += clock() - start;//Конц замера
	}
	cout << endl;
	//Блок ASM_SCAL----------------------------------
--------------------------------
	in = NULL;
	out = NULL;
	buffer = 0;
	wight = 0;
	height = 0;
	for (int i = 0; i < 100; i++)
	{
		cout << "Loadeing 1: " << i << "%" <<
endl;
		start = clock();
		fopen_s(&in, "in.bmp", "rb");
		buffer = (unsigned __int8*)malloc(54);
		fread(buffer, 1, 54, in);
		__asm
		{
			mov ESI, buffer;
		//Указатель на массив(регистр индекса источни-
ка)
			mov ECX, 3;
			//Начинаем извлекать высоту и
ширину
			mov EBX, 18;
			mov EAX, [ESI + EBX];
			mov wight, EAX;
			mov EAX, [ESI + EBX + 4];
			mov height, EAX;
		}
		fopen_s(&out, "OUT_scal.bmp", "wb");
		fwrite(buffer, 1, 54, out);
		free(buffer);

		buffer = (unsigned __int8*)malloc(wight *
height * 4);
		fread(buffer, 1, wight * height * 4, in);
		fclose(in);

		__asm
		{
			mov ESI, buffer;//Получаем ад-
ресс из буфера
			mov ECX, wight;//Делаем счёт-
чик для 1 цикла
		l1:
			push ECX;//Убираем счётчик в
стек
			mov ECX, 300;//Делаем новый
счётчик для 2 цикла
		l2:
			mov EAX, 3;//Подготавливаем
часть индекса, который используется в 3 цикле
(wight*3*j)
			mul wight;
			mov EBX, height;
			sub EBX, ECX;
			mul EBX;
			mov EBX, EAX;
```

```
                        //Подготавливаем остальную часть индекса(+i*3). Тк там используется значения из 1 цикла, делаем махинации с стеком
                        mov EAX, ECX; //Сохраняем текущее значение счётчика
                        pop ECX;
        //Достаём значения счётчика с прошлого цикла
                        push ECX;                       //Тк нам понадобится eax, сохраняем значения текущего счётчика в стеке

                        mov EAX, wight;
                        sub EAX, ECX;
                        mov EDX, 3;
                        mul EDX;
                        add EBX, EAX;
                        mov EAX, 0;
                        //Возвращаем значения счётчика 1 цикла в стек и достаём значения текущего
                        pop EAX;
                        push ECX;
                        mov ECX, EAX;
                        mov EAX, 0;

                        push ECX;
                        mov ECX, wight;
                l3:
                        add EBX, 2;
                        mov[ESI + EBX], 255;//Освещаем пиксель
                        inc EBX;
                        loop l3;

                        pop ECX;
        //Забираем счётчик 2 цикла из стека
                        loop l2;

                        pop ECX;
        //Забираем счётчик 3 цикла из стека
                        sub ECX, 127;    //Смещаемся не на 1 пиксель, а на 64*2-1, тк взаимодействие с 64 нужными было в 3 цикле
                        cmp ECX, 0;                     //Если наше смещение привело к отрицательному(или равному нулую) ecx - выходим из 1 цикла
                        jle ex;
                        loop l1;
                ex:
                }
                fwrite(buffer, 1, wight * height * 4, out);
                fclose(out);
                free(buffer);
                time_scal += clock() - start;
        }
        cout << endl;
                //Блок ASM_VEC------------------------------------------------------------------------------


                in = NULL;
        out = NULL;
        buffer = NULL;
        wight = 0;
        height = 0;
        for (int i = 0; i < 100; i++)
        {
                cout << "Loadeing 2: " << i << "%" << endl;
                start = clock();
                in = NULL;
                out = NULL;
                buffer = NULL;
                wight = 0;
                height = 0;

                fopen_s(&in, "in.bmp", "rb");
                buffer = (unsigned __int8*)malloc(54);
                fread(buffer, 1, 54, in);

                __asm
                {
                        mov ESI, buffer;
        //Указатель на массив
                        mov ECX, 3;
                        //Начинаем извлекать высоту и ширину
                        mov EBX, 18;
                        mov EAX, [ESI + EBX];
                        mov wight, EAX;
                        mov EAX, [ESI + EBX + 4];
                        mov height, EAX;
                }
                fopen_s(&out, "OUT_vec.bmp", "wb");
                fwrite(buffer, 1, 54, out);
                free(buffer);

                buffer = (unsigned __int8*)malloc(wight * height * 4);
                fread(buffer, 1, wight * height * 4, in);
                fclose(in);

                __asm
                {
                        mov ESI, buffer;//Получаем адресс из буфера
                        mov ECX, wight;//Делаем счётчик для 1 цикла
                l11:
                        push ECX;//Убираем счётчик в стек
                        mov ECX, 300;//Делаем новый счётчик для 2 цикла
                l12:
                        mov EAX, 3;//Подготавливаем часть индекса, который используется в 3 цикле (wight*3*j)
                        mul wight;
                        mov EBX, height;
                        sub EBX, ECX;
                        mul EBX;
                        mov EBX, EAX;
                        //Подготавливаем остальную часть индекса(+i*3). Тк там используется значения из 1 цикла, делаем махинации с стеком
                        mov EAX, ECX; //Сохраняем текущее значение счётчика
                        pop ECX;
        //Достаём значения счётчика с прошлого цикла
```

```
                push EAX;                //Тк нам понадобится EAX, сохраняем значения текущего счётчика в стеке

                mov EAX, wight;
                sub EAX, ECX;
                mov EDX, 3;
                mul EDX;
                add EBX, EAX;
                mov EAX, 0;
                //Возвращаем значения счётчика 1 цикла в стек и достаём значения текущего
                pop EAX;
                push ECX;
                mov ECX, EAX;
                mov EAX, 0;

                push ECX;
                mov ECX, wight;
        l13:
                //Освещаем пиксель
                add EBX, 3;
                movd mm0, [ESI + EBX];//запись в регистр MMX
                mov EAX, 0x00FF0000;
                movd mm1, EAX;//запись в регистр MMX
                paddusb mm0, mm1;//Сложение с насыщением беззнаковых упакованных байт
                movd[ESI + EBX], mm0;
                //Освещаем пиксель
                loop l13;

                pop ECX;
        //Забираем счётчик 2 цикла из стека
                loop l12;

                pop ECX;
        //Забираем счётчик 1 цикла из стека
                sub ECX, 127;    //
                cmp ECX, 0;              //Если наше смещение привело к отрицательному(или равному нулую) ecx - выходим из 1 цикла
                jle ex1;
                loop l11;
        ex1:
        }
        fwrite(buffer, 1, wight * height * 4, out);
        fclose(out);
        free(buffer);
        time_vec += clock() - start;
    }
    cout << "Время работы алгоритма на C++:" << time_cpp << endl;
    cout << "Время работы алгоритма на ASM с использованием скаляров:" << time_scal << endl;
    cout << "Время работы алгоритма на ASM с использованием векторов:" << time_vec << endl;
}
```

4.Результат работы программы

На рисунке 4.1-4.2 представлен результат работы программы на картинке.

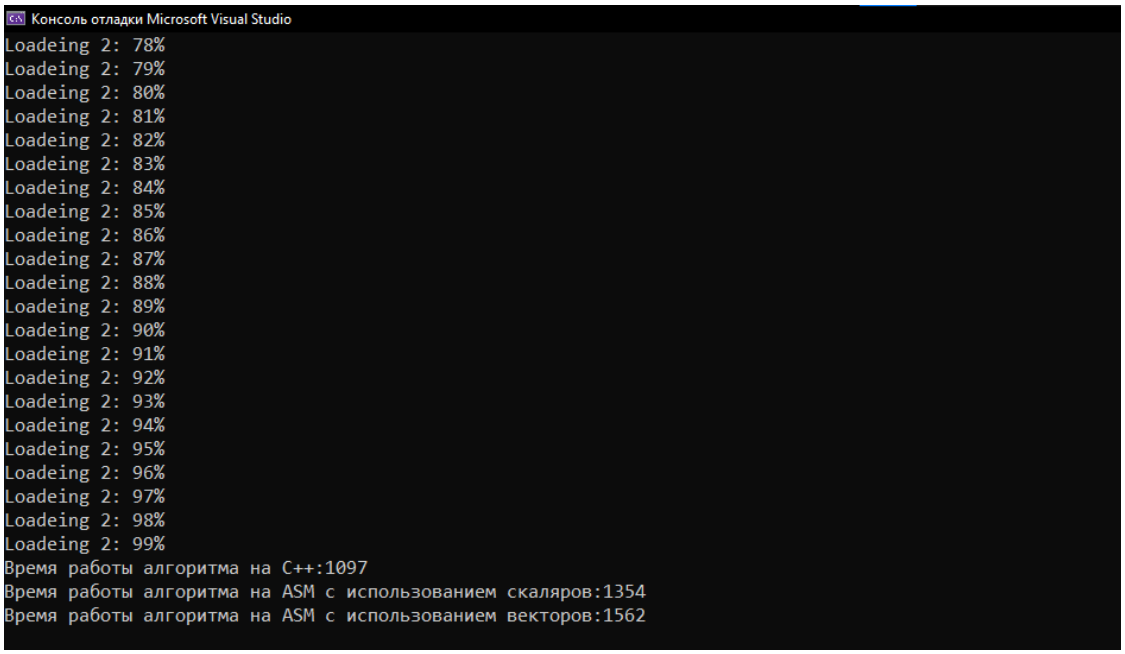

Рисунок 4.1- Результат работы программы

Рисунок 4.2- Итоговые изображения

Вывод

Векторные операции позволяют работать с массивами данных эффективнее за счёт команд, которые не получится использовать при работе с скалярными операциями, а также, за счёт возможности работы сразу с несколькими элементами массива.