#### 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

## Оглавление

1. Цель работы	3
2.Задание	4
3.Текст программы	5
4.Результат работы программы	8
Вывол	C

# 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 с++. 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

```
time_cpp += clock() - start;//Конц замера
#include <iostream>
#include <stdlib.h>
                                                                           cout << endl;
#include <conio.h>
                                                                           //Блок ASM SCAL-----
#include <locale.h>
using namespace std;
                                                                            in = NULL;
                                                                            out = NULL:
int main()
                                                                            buffer = 0;
                                                                            wight = 0;
        setlocale(LC_ALL, "Ru");
                                                                           height = 0;
        int time_cpp = 0;
                                                                            for (int i = 0; i < 100; i++)
        int time_scal = 0;
        int time_vec = 0;
                                                                                    cout << "Loadeing 1: " << i << "%" <<
        time_t start;
                                                                   endl;
        FILE* in, * out;
                                                                                    start = clock();
        unsigned __int8* buffer;
                                                                                    fopen_s(&in, "in.bmp", "rb");
        unsigned __int32 wight, height;
                                                                                    buffer = (unsigned __int8*)malloc(54);
        //Блок С++
                                                                                    fread(buffer, 1, 54, in);
        for (int i = 0; i < 100; i++)
                                                                                      _asm
                 cout << "Loadeing 0: " << i << "%" <<
                                                                                             mov ESI, buffer;
                                                                           //Указатель на массив(регистр индекса источни-
endl;
                 start = clock();//Старт замера
                                                                   ка)
                 fopen_s(&in, "in.bmp", "rb");
                                                                                             mov ECX, 3;
                 buffer = (unsigned __int8*)malloc(54);
                                                                                             //Начинаем извлекать высоту и
                 fread(buffer, 1, 54, in);
                                                                   ширину
                 for (int i = 0; i <= 3; i++)
                                                                                             mov EBX, 18;
                                                                                             mov EAX, [ESI + EBX];
                         wight = buffer[21 - i] | wight <<
                                                                                             mov wight, EAX;
                                                                                             mov EAX, [ESI + EBX + 4];
8;
                         height = buffer[25 - i] | height <<
                                                                                             mov height, EAX;
8;
                                                                                    fopen_s(&out, "OUT_scal.bmp", "wb");
                 fopen_s(&out, "OUT_cpp.bmp", "wb");
                                                                                    fwrite(buffer, 1, 54, out);
                 fwrite(buffer, 1, 54, out);
                                                                                    free(buffer);
                 free(buffer);
                                                                                    buffer = (unsigned __int8*)malloc(wight *
                                                                   height *4);
                 //Редактирование картинки
                 buffer = (unsigned __int8*)malloc(wight *
                                                                                    fread(buffer, 1, wight * height * 4, in);
height *4);
                                                                                    fclose(in);
                 fread(buffer, 1, wight * height * 4, in);
                 fclose(in);
                                                                                      asm
                 for (int i = 0; i < wight; i += 128)
                                                                                             mov ESI, buffer;//Получаем ад-
                                                                   ресс из буфера
                         for (int j = height; j > height / 2; j -
                                                                                             mov ECX, wight;//Делаем счёт-
                                                                   чик для 1 цикла
-)
                                                                                    11:
                                  for (int k = i; k < wight;
                                                                                             push ECX;//Убираем счётчик в
k++)
                                                                   стек
                                                                                             то ЕСХ, 300;//Делаем новый
                                  {
                                          buffer[wight * 3
                                                                   счётчик для 2 цикла
* j + k * 3 + 2] = 255;
                                                                                    12:
                                          buffer[wight * 3
                                                                                             то ЕАХ, 3;//Подготавливаем
*i + k * 3 + 1] = 0;
                                                                   часть индекса, который используется в 3 цикле
                                          buffer[wight * 3
                                                                   (wight*3*j)
*j + k * 3 + 0] = 0;
                                                                                             mul wight;
                                  }
                                                                                             mov EBX, height;
                                                                                             sub EBX, ECX;
                                                                                             mul EBX;
                 fwrite(buffer, 1, wight * height * 4, out);
                                                                                             mov EBX, EAX;
                 fclose(out);
                 free(buffer);
```

```
//Подготавливаем остальную
часть индекса(+i*3). Тк там используется значения из 1
                                                                               cout << "Loadeing 2: " << i << "%" <<
цикла, делаем махинации с стеком
                                                               endl:
                        mov EAX, ECX; //Сохраняем
                                                                               start = clock();
                                                                               in = NULL;
текущее значение счётчика
                                                                               out = NULL;
                       pop ECX;
                                                                               buffer = NULL;
       //Достаём значения счётчика с прошлого цикла
                       push ECX;
                                               //Tĸ
                                                                               wight = 0;
нам понадобится еах, сохраняем значения текущего
                                                                               height = 0;
счётчика в стеке
                                                                               fopen_s(&in, "in.bmp", "rb");
                       mov EAX, wight;
                                                                               buffer = (unsigned __int8*)malloc(54);
                        sub EAX, ECX;
                                                                               fread(buffer, 1, 54, in);
                       mov EDX, 3;
                       mul EDX;
                                                                                 asm
                       add EBX, EAX;
                                                                               {
                       mov EAX, 0;
                                                                                       mov ESI, buffer;
                                                                       //Указатель на массив
                       //Возвращаем значения счётчи-
ка 1 цикла в стек и достаём значения текущего
                                                                                       mov ECX, 3;
                       pop EAX;
                                                                                       //Начинаем извлекать высоту и
                       push ECX;
                                                               ширину
                       mov ECX, EAX;
                                                                                       mov EBX, 18;
                                                                                       mov EAX, [ESI + EBX];
                       mov EAX, 0;
                                                                                       mov wight, EAX;
                       push ECX;
                                                                                       mov EAX, [ESI + EBX + 4];
                       mov ECX, wight;
                                                                                       mov height, EAX;
               13:
                                                                               fopen_s(&out, "OUT_vec.bmp", "wb");
                       add EBX, 2;
                                                                               fwrite(buffer, 1, 54, out);
                       mov[ESI + EBX],
255;//Освещаем пиксель
                                                                               free(buffer);
                        inc EBX;
                       loop 13;
                                                                               buffer = (unsigned __int8*)malloc(wight *
                                                               height *4);
                       pop ECX;
                                                                               fread(buffer, 1, wight * height * 4, in);
       //Забираем счётчик 2 цикла из стека
                                                                               fclose(in);
                       loop 12;
                                                                                 asm
                       pop ECX;
       //Забираем счётчик 3 цикла из стека
                                                                                       mov ESI, buffer;//Получаем ад-
                       sub ECX, 127; //Смещаемся
                                                               ресс из буфера
не на 1 пиксель, а на 64*2-1, тк взаимодействие с 64
                                                                                       mov ECX, wight;//Делаем счёт-
нужными было в 3 цикле
                                                               чик для 1 цикла
                       cmp ECX, 0;
                                               //Если
                                                                               111:
наше смещение привело к отрицательному(или равному
                                                                                       push ECX;//Убираем счётчик в
нулую) есх - выходим из 1 цикла
                                                               стек
                       jle ex;
                                                                                       то ЕСХ, 300;//Делаем новый
                       loop 11;
                                                               счётчик для 2 цикла
               ex:
                                                                               112:
                                                                                       тоу ЕАХ, 3;//Подготавливаем
               fwrite(buffer, 1, wight * height * 4, out);
                                                               часть индекса, который используется в 3 цикле
               fclose(out);
                                                               (wight*3*j)
               free(buffer);
                                                                                       mul wight;
                                                                                       mov EBX, height;
               time_scal += clock() - start;
                                                                                       sub EBX, ECX;
       cout << endl;
                                                                                       mul EBX:
               //Блок ASM_VEC-----
                                                                                       mov EBX, EAX;
                                                                                       //Подготавливаем остальную
                                                               часть индекса(+i*3). Тк там используется значения из 1
               in = NULL;
                                                               цикла, делаем махинации с стеком
       out = NULL;
                                                                                       mov EAX, ECX; //Сохраняем
       buffer = NULL;
                                                               текущее значение счётчика
        wight = 0;
                                                                                       pop ECX;
       height = 0;
                                                                       //Достаём значения счётчика с прошлого цикла
       for (int i = 0; i < 100; i++)
```

```
push EAX;
                                                                                     //Освещаем пиксель
нам понадобится ЕАХ, сохраняем значения текущего
                                                                                     loop 113;
счётчика в стеке
                                                                                     pop ECX;
                       mov EAX, wight;
                                                                     //Забираем счётчик 2 цикла из стека
                       sub EAX, ECX;
                                                                                     loop 112;
                       mov EDX, 3;
                       mul EDX;
                                                                                     pop ECX;
                       add EBX, EAX;
                                                                     //Забираем счётчик 1 цикла из стека
                       mov EAX, 0;
                                                                                     sub ECX, 127; //
                       //Возвращаем значения счётчи-
                                                                                     cmp ECX, 0;
                                                                                                            //Если
ка 1 цикла в стек и достаём значения текущего
                                                             наше смещение привело к отрицательному(или равному
                       pop EAX;
                                                             нулую) есх - выходим из 1 цикла
                       push ECX;
                                                                                     jle ex1;
                       mov ECX, EAX;
                                                                                     loop 111;
                       mov EAX, 0;
                                                                             ex1:
                                                                             fwrite(buffer, 1, wight * height * 4, out);
                       push ECX;
                       mov ECX, wight;
                                                                             fclose(out);
               113:
                                                                             free(buffer);
                       //Освещаем пиксель
                                                                             time_vec += clock() - start;
                       add EBX, 3;
                                                                     }
                       movd mm0, [ESI +
                                                                     cout << "Время работы алгоритма на С++:" <<
ЕВХ];//запись в регистр ММХ
                                                             time_cpp << endl;
                       mov EAX, 0x00FF0000;
                                                                     cout << "Время работы алгоритма на ASM с ис-
                                                             пользованием скаляров:" << time_scal << endl;
                       movd mm1, EAX;//запись в ре-
гистр ММХ
                                                                     cout << "Время работы алгоритма на ASM с ис-
                                                             пользованием векторов:" << time_vec << endl;
                       paddusb mm0, mm1;//Сложение
с насыщением беззнаковых упакованных байт
                       movd[ESI + EBX], mm0;
```

## 4. Результат работы программы

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

```
Morconb отладки Microsoft Visual Studio
Loadeing 2: 78%
Loadeing 2: 79%
Loadeing 2: 80%
Loadeing 2: 81%
Loadeing 2: 82%
Loadeing 2: 83%
Loadeing 2: 83%
Loadeing 2: 85%
Loadeing 2: 85%
Loadeing 2: 86%
Loadeing 2: 87%
Loadeing 2: 87%
Loadeing 2: 88%
Loadeing 2: 89%
Loadeing 2: 99%
Loadeing 2: 99%
Loadeing 2: 99%
Loadeing 2: 92%
Loadeing 2: 95%
Loadeing 2: 96%
Loadeing 2: 96%
Loadeing 2: 98%
Loadeing 2: 99%
Время работы алгоритма на C++:1097
Время работы алгоритма на ASM с использованием скаляров:1354
Время работы алгоритма на ASM с использованием векторов:1562
```

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.