МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

#### ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ

БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет автоматики и вычислительной техники

Кафедра радиоэлектронных средств

Отчет по дисциплине

«Цифровые устройства и микропроцессоры»

Лабораторная работа №4

«МОДУЛЬНОЕ ПРОГРАММИРОВАНИЕ»

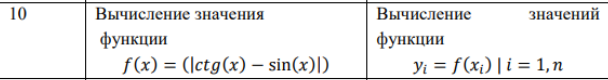
Вариант №10

|  |  |  |
| --- | --- | --- |
| Выполнил: студент группы ИНБс– 3301 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | А.С. Пупышев |
|  |  |  |
| Проверил: к.т.н. доцент кафедры РЭС | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | М.А. Земцов |

#### Киров 2022

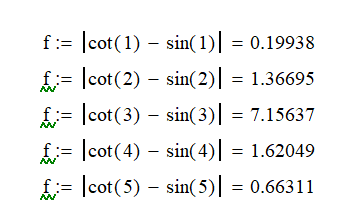
**Цель работы**: знакомство с технологией применения языка ассемблера при разработке программного обеспечения на языках высокого уровня.

1. **Задание**



1. **Ход работы**

***Расчет с помощью калькулятора:***



***Программное исполнение:***

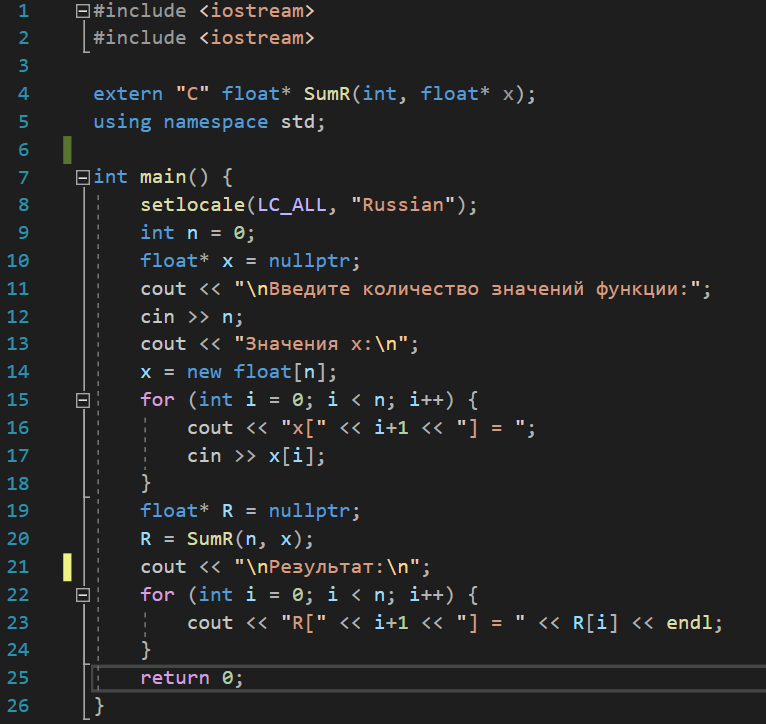
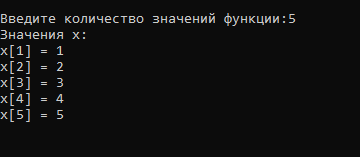


Рисунок 1 – Ввод данных

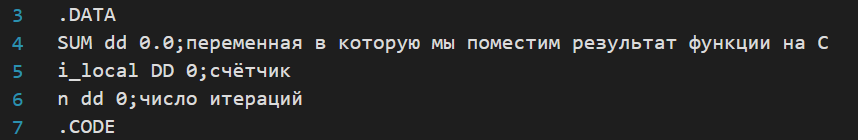


Рисунок 2 – Инициализация данных в ассемблере

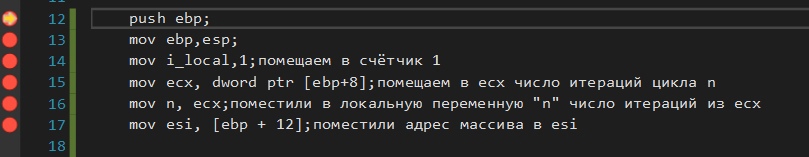
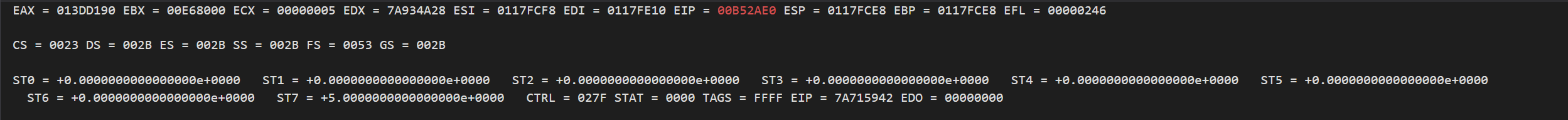


Рисунок 3 – Начало подпрограммы



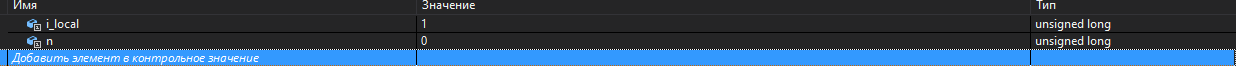
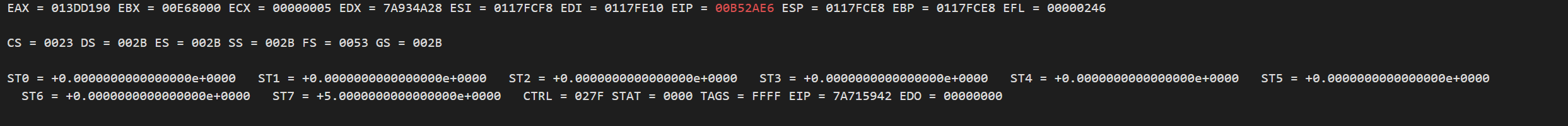


Рисунок 3 – Поместили 1 в счётчик



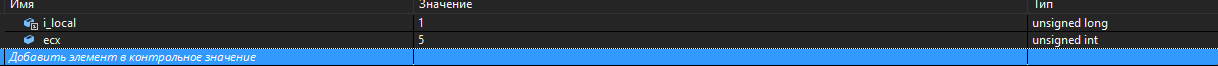


Рисунок 4 – Поместили в ecx число итераций n

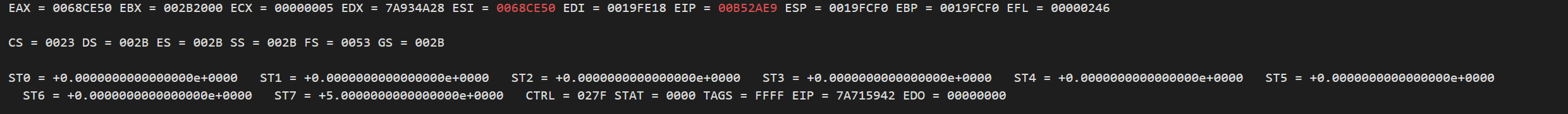


Рисунок 5 – Поместили адрес массива (второй полученный аргумент) в esi

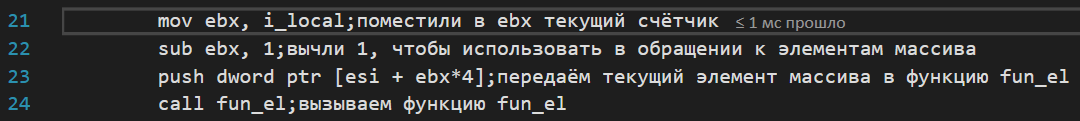


Рисунок 6 – Начало цикла

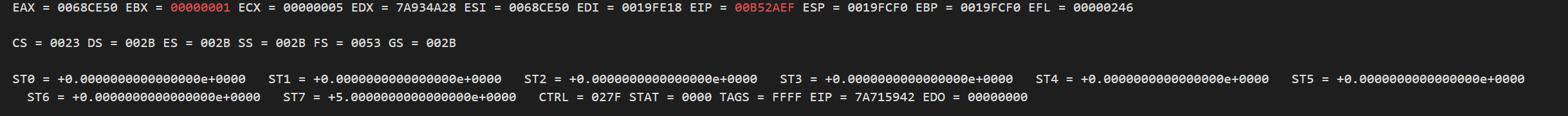


Рисунок 7 – Поместили текущий счётчик в ebx

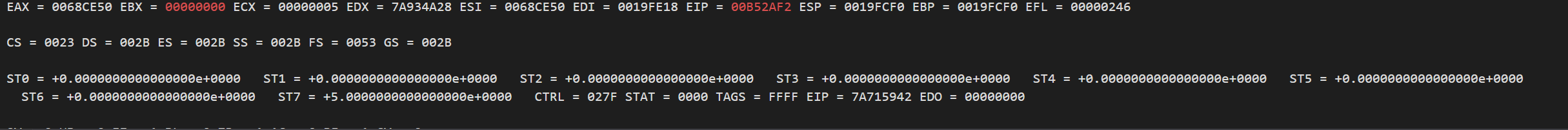


Рисунок 7 – Вычитаем 1 для того, чтобы использовать в обращении к элементам массива

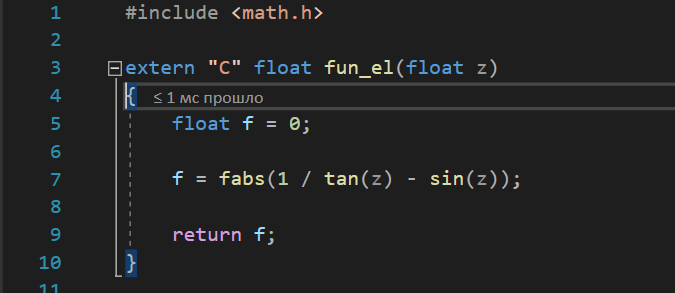


Рисунок 8 – Передаем в fun\_el текущий элемент массива



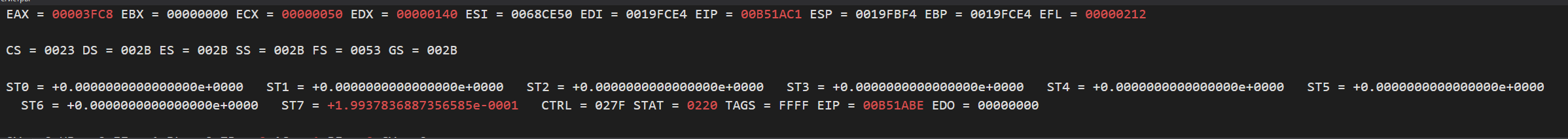


Рисунок 9 – Вычисленные значения функции fun\_el

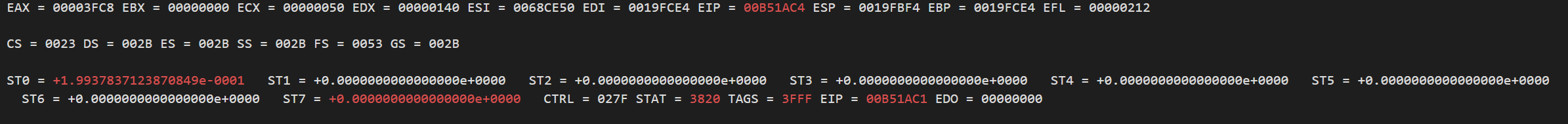
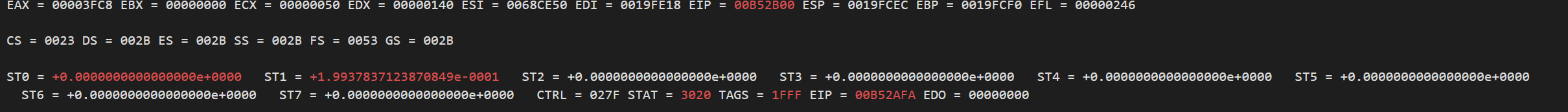


Рисунок 10 – Возврат в ассемблер. В st(0) записано значение 0.199378, что совпадает с теоретическим расчётом



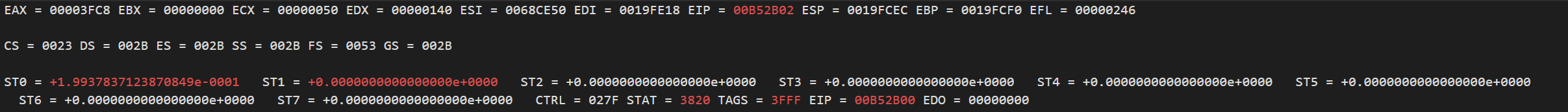


Рисунок 11 – Поместили SUM в st(0) и сложили с st(1)



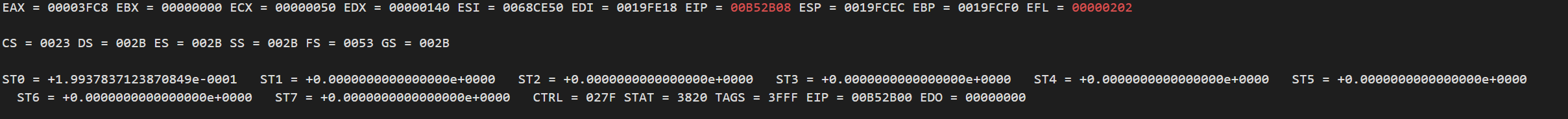
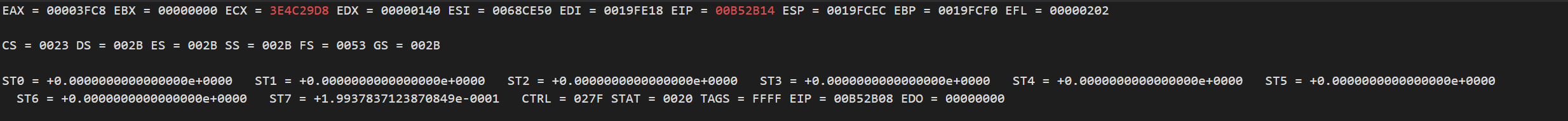


Рисунок 12 – Увеличили счетчик на 1



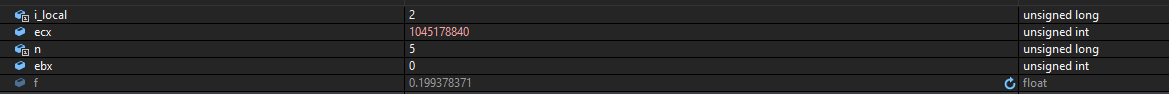


Рисунок 13 – Поместили SUM в ecx

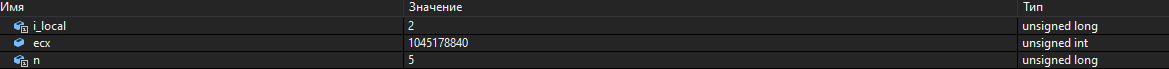


Рисунок 14 – Поместили содержимое ecx в текущий элемент массива по адресу esi + ebx\*4.

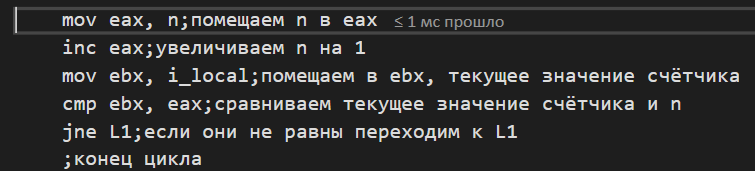
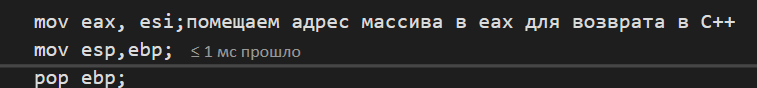


Рисунок 15 – Проверка на условие, что ebx и eax

Так как в качестве примера я задал n=5, то цикл не закончится до тех пор, пока eax не станет равным ebx.



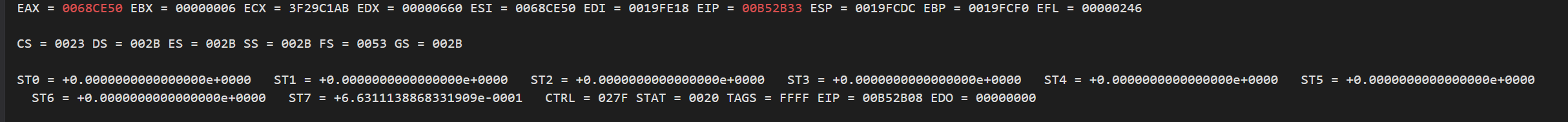


Рисунок 16 – Помещаем в eax, esi (адрес массива)

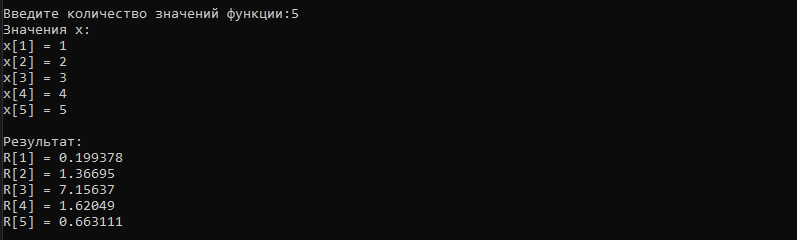


Рисунок 17 – Вернулись в главную функцию, в консоли вывелись результаты, которые совпадают ручным расчётом.

**Вывод:** в ходе данной лабораторной работы познакомились с модульным программированием, синтаксисом EXTERN, PUBLIC, а также с организацией вызова функций с использованием языка Си/С++ и ассемблера.

**3. Код программы:**

***laba4.cpp***

#include <iostream>

#include <iostream>

extern "C" float\* SumR(int, float\* x);

using namespace std;

int main() {

setlocale(LC\_ALL, "Russian");

int n = 0;

float\* x = nullptr;

cout << "\nВведите количество значений функции:";

cin >> n;

cout << "Значения x:\n";

x = new float[n];

for (int i = 0; i < n; i++) {

cout << "x[" << i+1 << "] = ";

cin >> x[i];

}

float\* R = nullptr;

R = SumR(n, x);

cout << "\nРезультат:\n";

for (int i = 0; i < n; i++) {

cout << "R[" << i+1 << "] = " << R[i] << endl;

}

return 0;

}

***fun\_C.cpp***

#include <math.h>

extern "C" float fun\_el(float z)

{

float f = 0;

f = fabs(1 / tan(z) - sin(z));

return f;

}

***fun\_as.asm***

.586

.MODEL flat,C

.DATA

SUM dd 0.0;переменная в которую мы поместим результат функции на C

i\_local DD 0;счётчик

n dd 0;число итераций

.CODE

extern fun\_el:near; объявление внешней функции fun\_el

public SumR

SumR proc C

push ebp;

mov ebp,esp;

mov i\_local,1;помещаем в счётчик 1

mov ecx, dword ptr [ebp+8];помещаем в ecx число итераций цикла n

mov n, ecx;поместили в локальную переменную "n" число итераций из ecx

mov esi, [ebp + 12];поместили адрес массива в esi

;начало цикла

L1:

mov ebx, i\_local;поместили в ebx текущий счётчик

sub ebx, 1;вычли 1, чтобы использовать в обращении к элементам массива

push dword ptr [esi + ebx\*4];передаём текущий элемент массива в функцию fun\_el

call fun\_el;вызываем функцию fun\_el

fld SUM;помещаем в st(0) SUM

fadd;складываем st(0) и st(1)

inc i\_local;увеличиваем счётчик на 1

fstp SUM;помещаем значение из st(0) в SUM

mov ecx, SUM;помещаем SUM в ecx

mov [esi + ebx\*4], ecx;помещаем в текущий элемент массива значение из ecx

mov SUM, 0;обнуляем SUM

mov eax, n;помещаем n в eax

inc eax;увеличиваем n на 1

mov ebx, i\_local;помещаем в ebx, текущее значение счётчика

cmp ebx, eax;сравниваем текущее значение счётчика и n

jne L1;если они не равны переходим к L1

;конец цикла

mov eax, esi;помещаем адрес массива в eax для возврата в C++

mov esp,ebp;

pop ebp;

ret

SumR endp

End