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 №15

|  |  |  |
| --- | --- | --- |
| Выполнил: студент гр. ИНБс-3301 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | А.С. Кокшарова |
| Проверил: доцент кафедры РЭС | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | М.А. Земцов |

#### Киров 2022

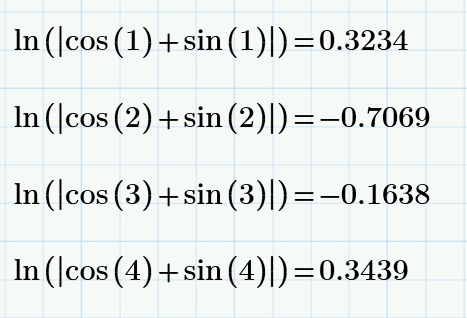
***Цель работы****: знакомство с технологией применения языка ассемблера при разработке программного обеспечения на языках высокого уровня.*

1. **Задание**

|  |  |  |
| --- | --- | --- |
| Вариант | Подпрограмма на C | Подпрограмма ассемблера |
|  | | |

1. **Ход работы**

***Расчет с помощью калькулятора:***



***Программное исполнение:***

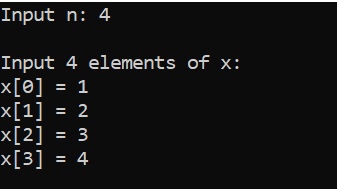
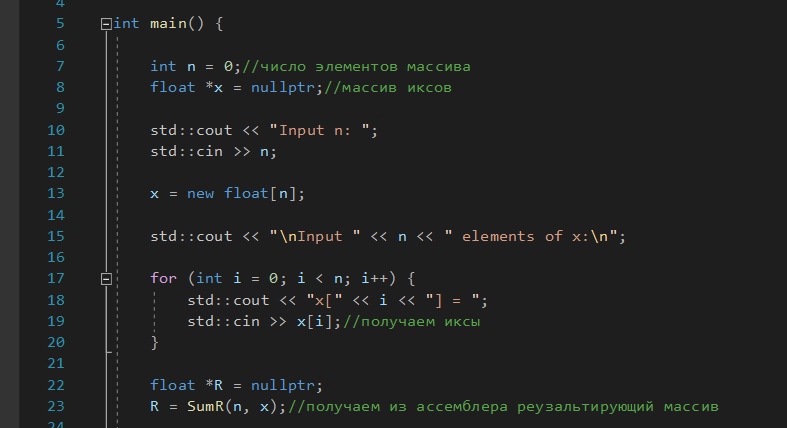
 

Рисунок 1 – Ввод данных

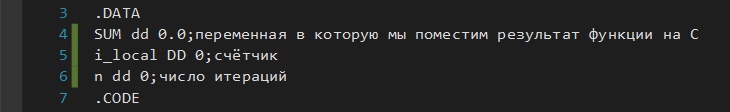


Рисунок 2 – Инициализация данных в ассемблере

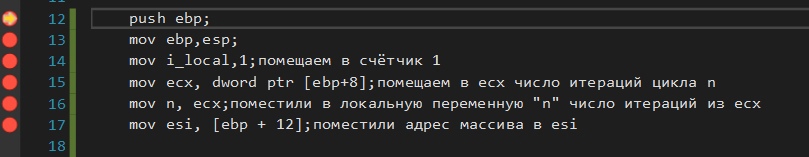


Рисунок 3 – Начало подпрограммы

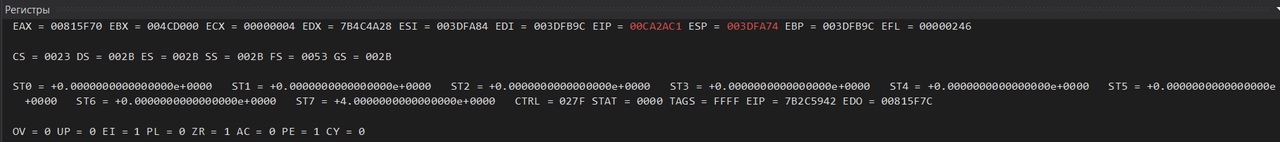


Рисунок 4 – push ebp

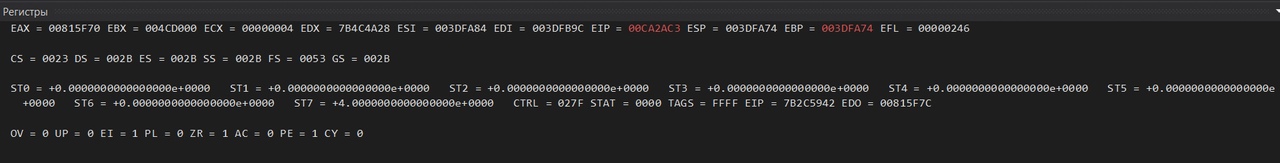
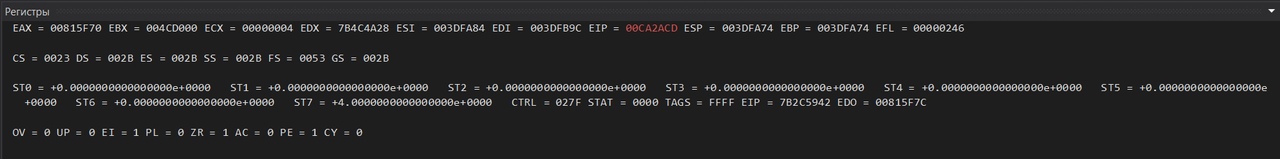


Рисунок 5 – mov ebp, esp



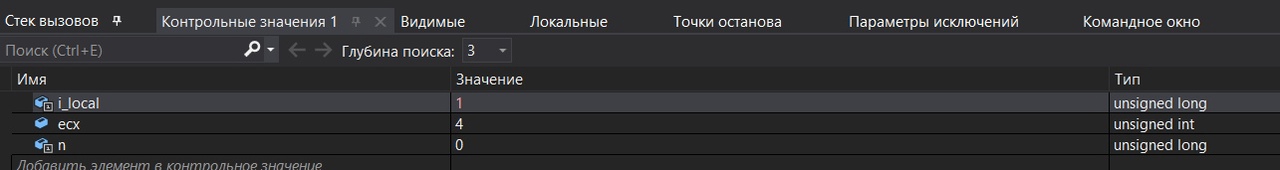
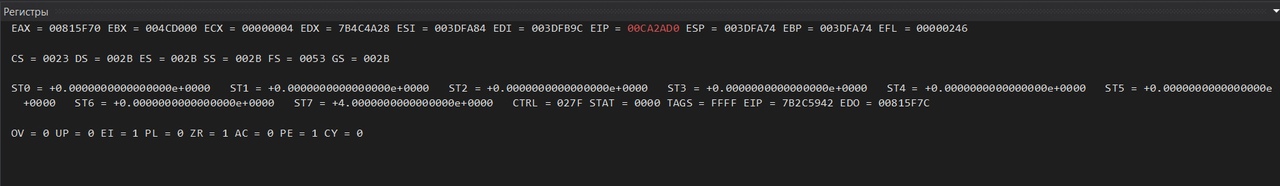


Рисунок 6 – Поместили 1 в счётчик



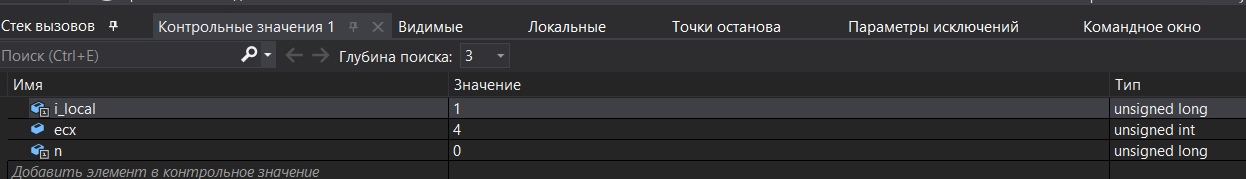
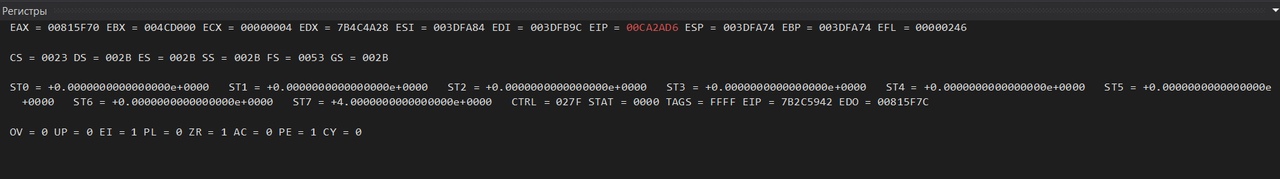


Рисунок 7 – Поместили в ecx число итераций n



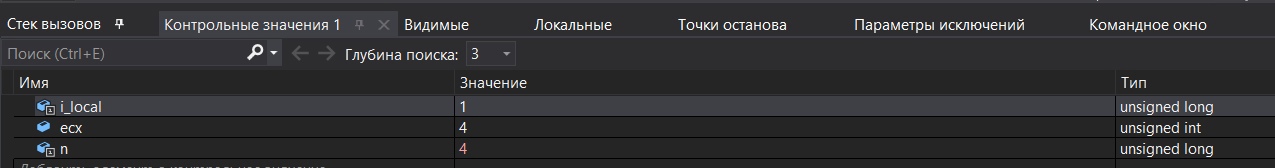


Рисунок 8 – Поместили в локальную переменную "n" число итераций из ecx

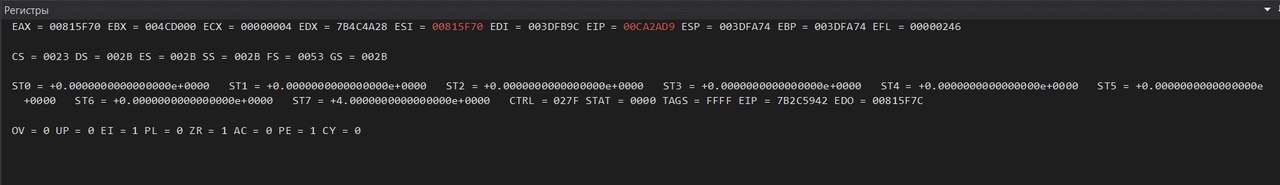


Рисунок 9 – Поместили адрес массива (второй полученный аргумент) в esi

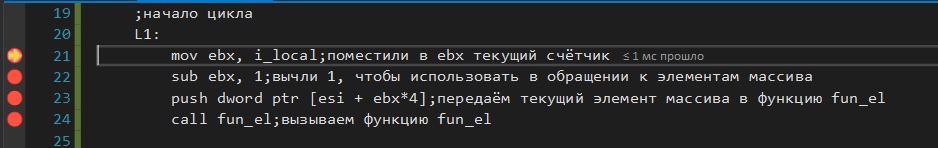
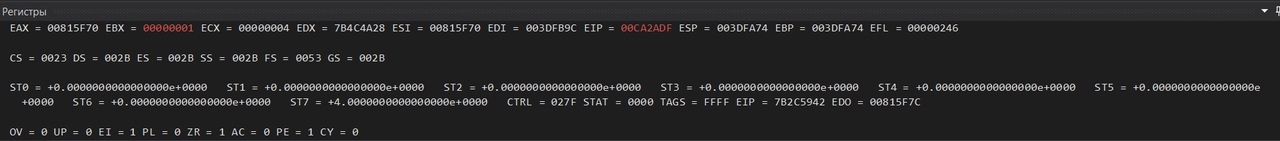


Рисунок 10 – Начало цикла



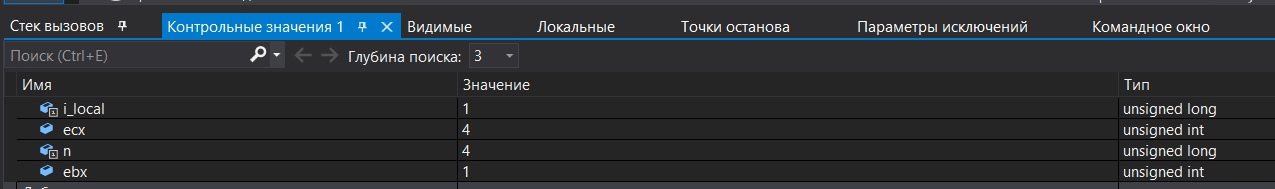
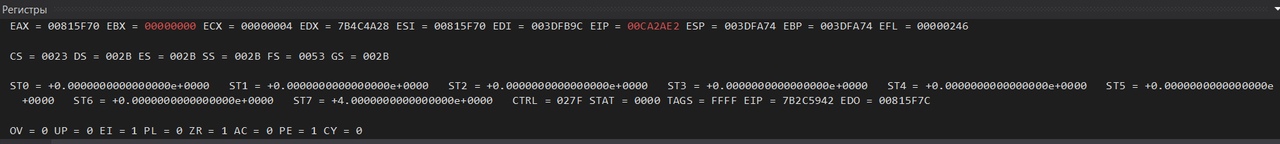


Рисунок 11 – Поместили в ebx текущий счётчик



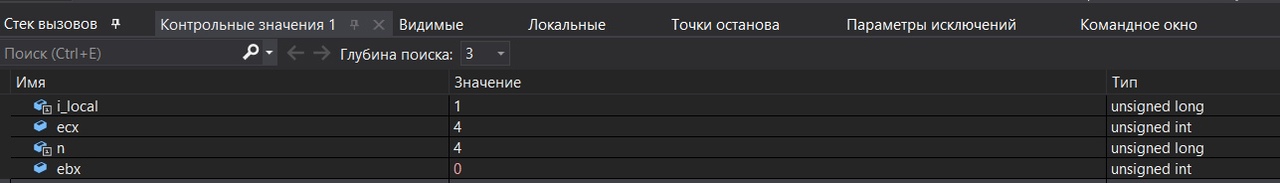


Рисунок 12 – Вычитаем из ebx 1

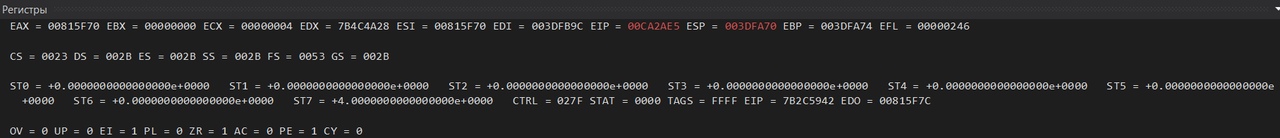


Рисунок 13 – Передаём в функцию fun\_el текущий элемент массива

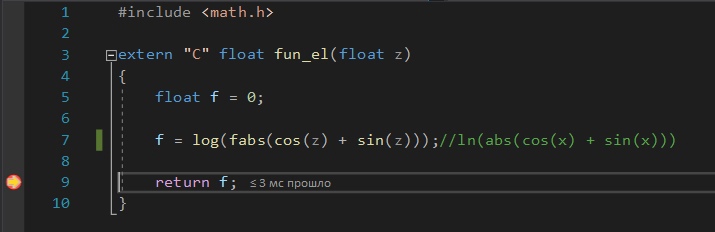
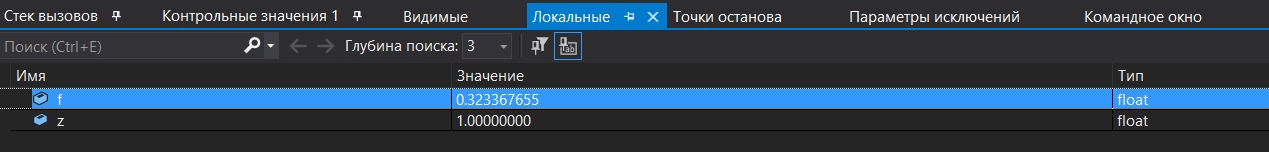


Рисунок 14 – Вызываем функцию fun\_el



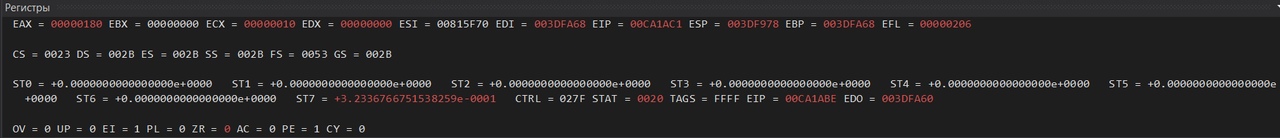


Рисунок 15 – Вычисленные значения функции fun\_el

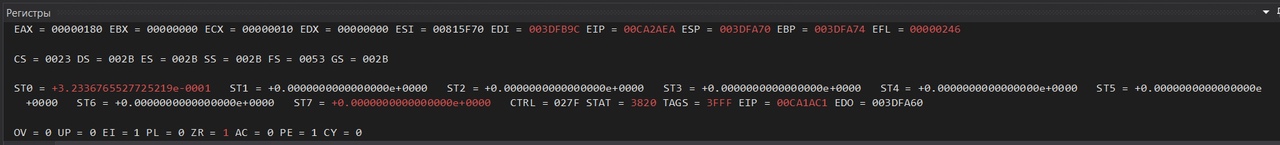


Рисунок 16 – Возврат в ассемблер. В st(0) записано значение 0.3234, что совпадает с теоретическим расчётом

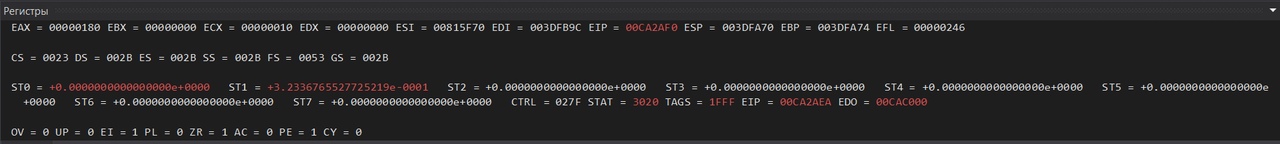
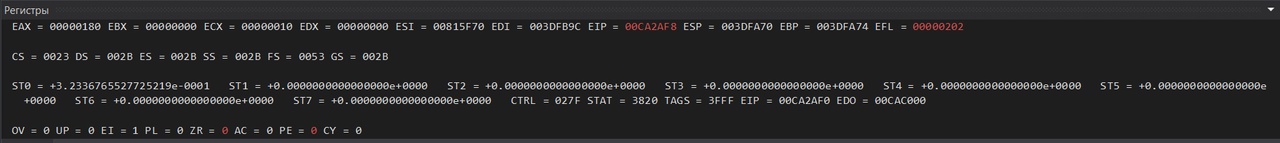


Рисунок 17 – Поместили SUM в st(0)



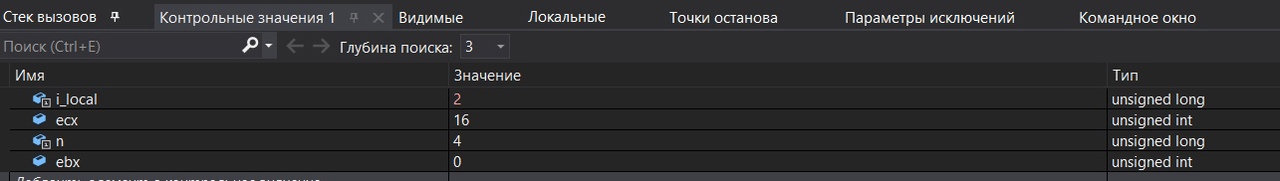


Рисунок 18 – Увеличили счётчик на 1

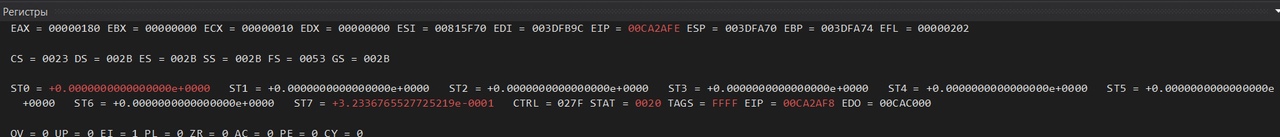
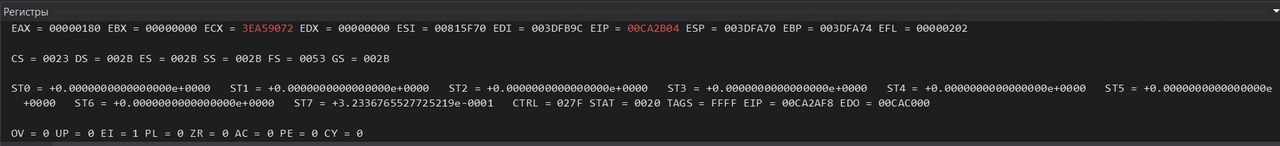


Рисунок 19 – Поместили st(0) в SUM



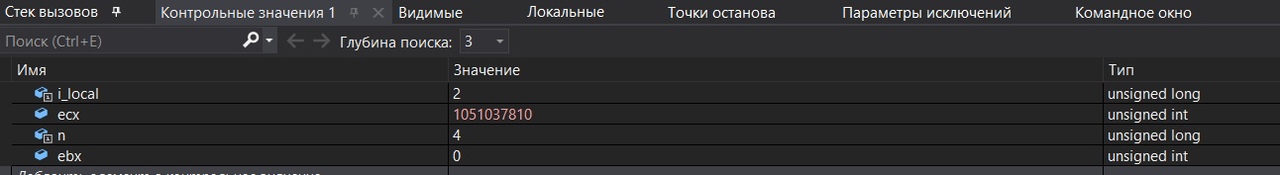
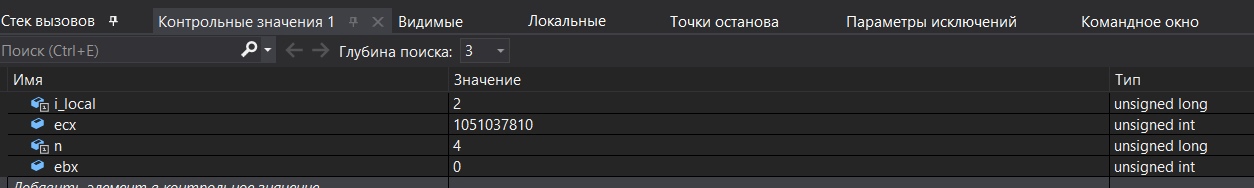


Рисунок 20 – Поместили SUM в ecx



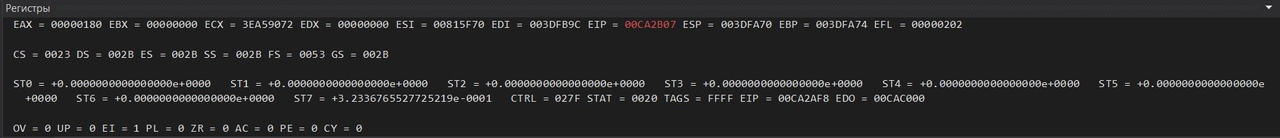


Рисунок 21 – Поместили содержимое ecx в текущий элемент массива по адресу esi + ebx\*4. Обнуляем SUM

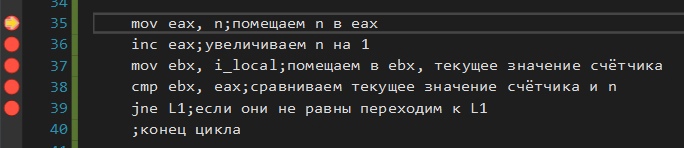


Рисунок 22 – Проверка условия на завершение цикла

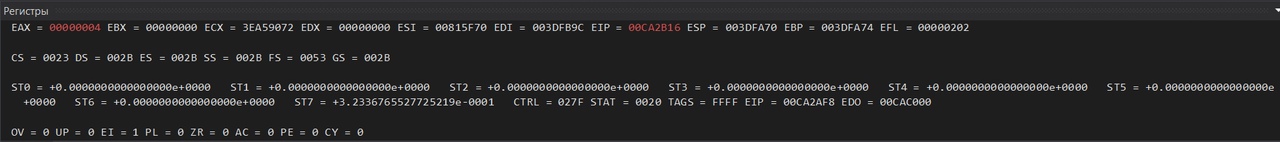


Рисунок 23 – Поместили n в eax

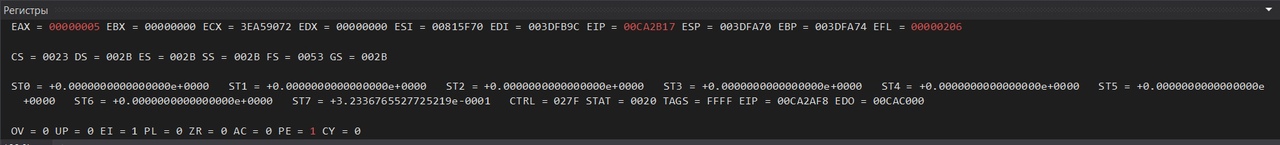
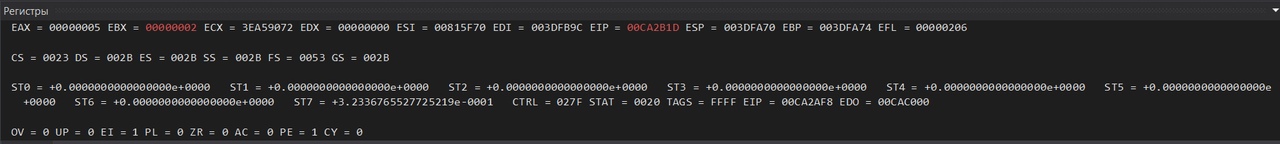


Рисунок 24 – Увеличиваем n на 1



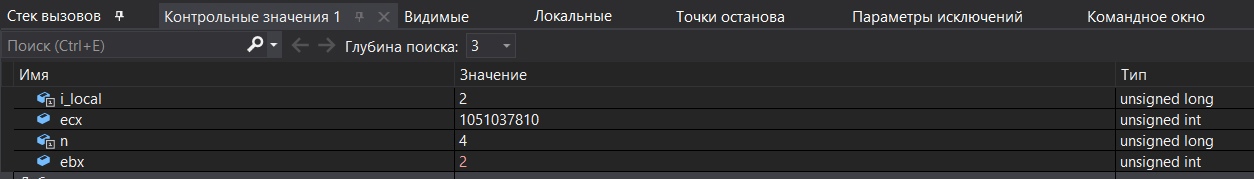


Рисунок 25 – Поместили в ebx текущее значение счётчика

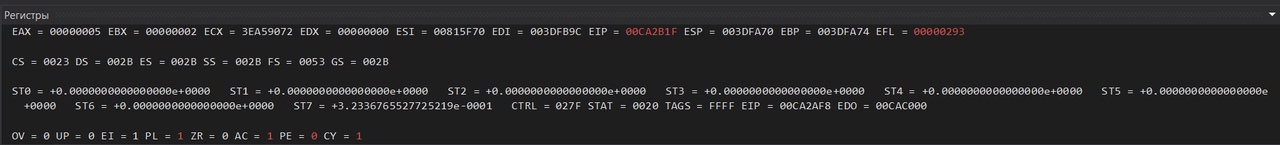


Рисунок 26 – Сравнение n и текущего значения счётчика

так как n больше ebx, то продолжаем цикл

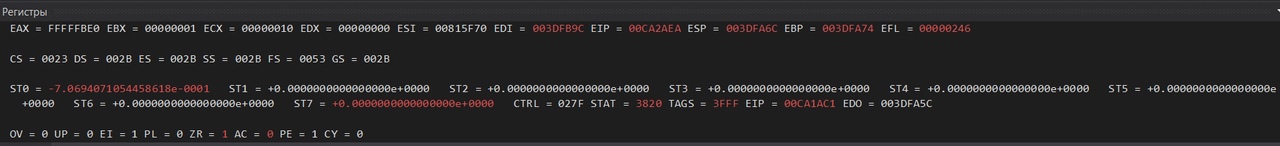


Рисунок 27 – На второй итерации значение функции fun\_el равно -0.7069, что совпадает с теоретическим расчётом

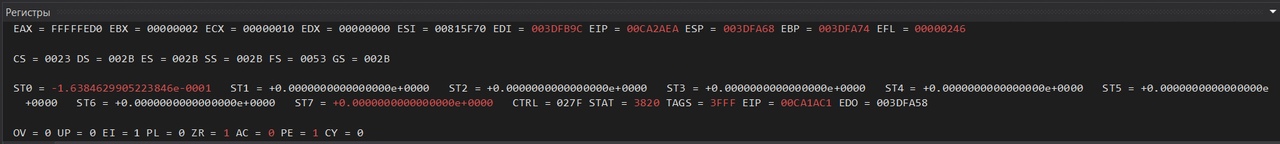


Рисунок 28 – На третьей итерации значение функции fun\_el равно -0.1638, что совпадает с теоретическим расчётом

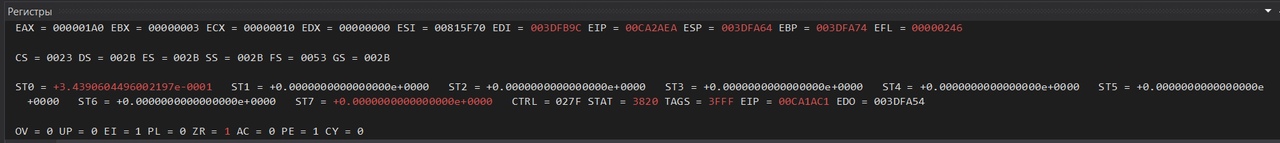
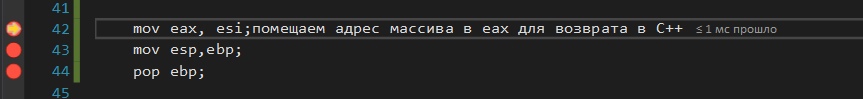


Рисунок 29 – На четвёртой итерации значение функции fun\_el равно 0.3439, что совпадает с теоретическим расчётом. цикл закончился



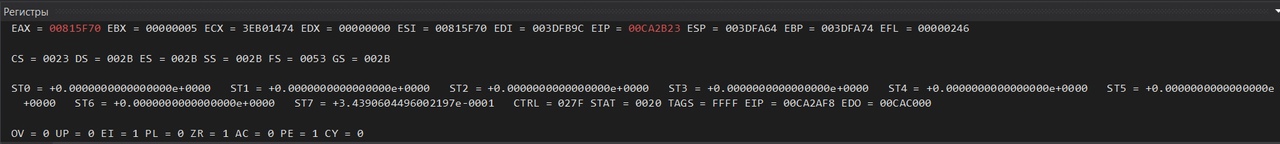


Рисунок 30 – Помещаем в eax, esi (адрес массива)

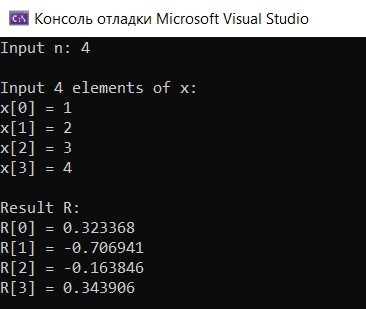
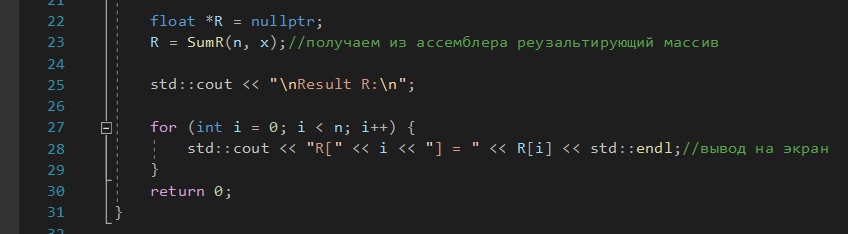
 

Рисунок 31 – Вернулись в главную функцию. вывелись результаты расчёта. Они совпадают с теоретическим расчётом.

Результаты ручного расчёта и результат программы совпадают.

***Вывод:*** программа выполняет задачу верно. Так же ознакомились с технологией применения языка ассемблера при разработке программного обеспечения на языках высокого уровня.

**3. Код программы:**

***lab4.cpp***

#include <iostream>

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

int main() {

int n = 0;//число элементов массива

float \*x = nullptr;//массив иксов

std::cout << "Input n: ";

std::cin >> n;

x = new float[n];

std::cout << "\nInput " << n << " elements of x:\n";

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

std::cout << "x[" << i << "] = ";

std::cin >> x[i];//получаем иксы

}

float \*R = nullptr;

R = SumR(n, x);//получаем из ассемблера реузальтирующий массив

std::cout << "\nResult R:\n";

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

std::cout << "R[" << i << "] = " << R[i] << std::endl;//вывод на экран

}

return 0;

}

***fun\_C.cpp***

#include <math.h>

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

{

float f = 0;

f = log(fabs(cos(z) + sin(z)));//ln(abs(cos(x) + sin(x)))

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