МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ

БЮДЖЕТНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНВЕРСИТЕТ

Факультет автоматики и вычислительной техники

Кафедра радиоэлектронных средств

Отчет по дисциплине

«Цифровые устройства и микропроцессоры»

Лабораторная работа №4

«МОДУЛЬНОЕ ПРОГРАММИРОВАНИЕ»

Вариант №15

Выполнил: студент группы ИНБс-3301 К.А. Юрлов

Проверил: доцент кафедры РЭС М.А. Земцов

Киров 2024

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

**Ход работы:**

Задание по варианту представлено на рисунке 1.

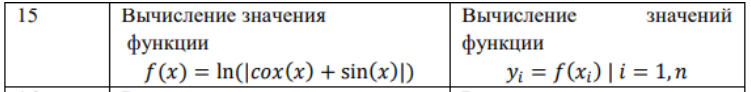


Рисунок 1 – задание по варианту

Проверка расчетов с помощью калькулятора:

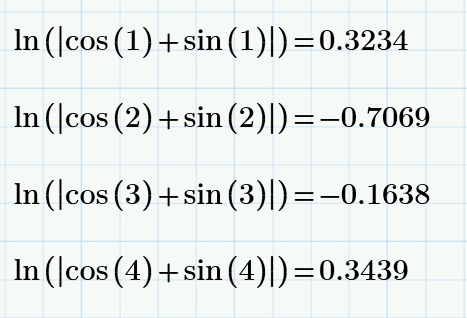


Рисунок 2 – проверка расчетов

Ввод данных в программу представлен на рисунках 3-4:

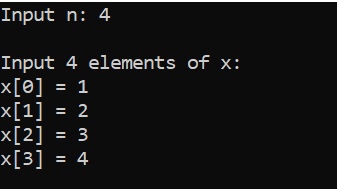


Рисунок 3 – ввод данных (терминал)

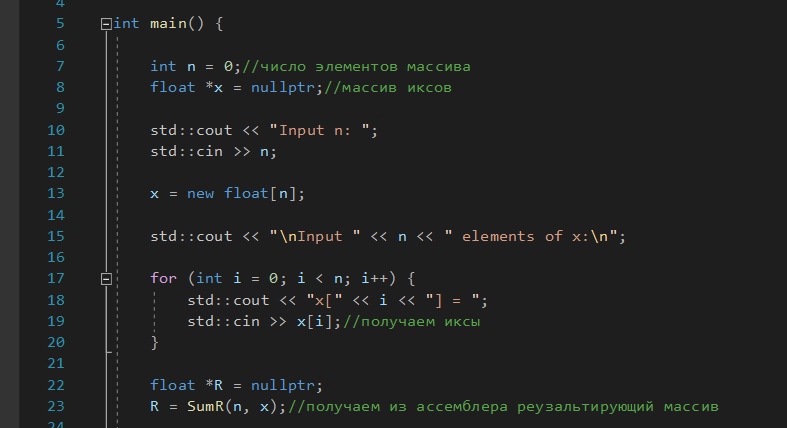


Рисунок 4 – ввод данных

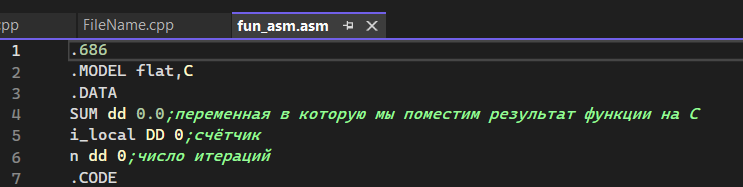


Рисунок 5 – инициализация данных в ассемблере

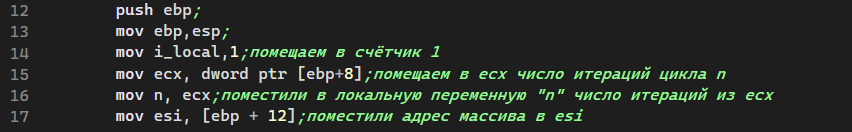


Рисунок 6 – подпрограмма

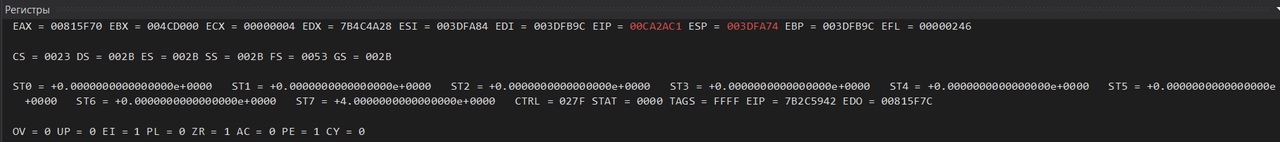


Рисунок 7 – push ebp

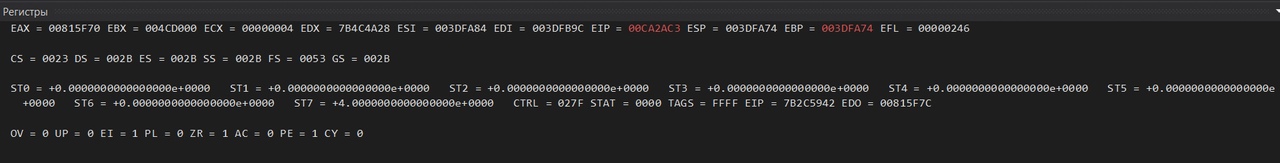
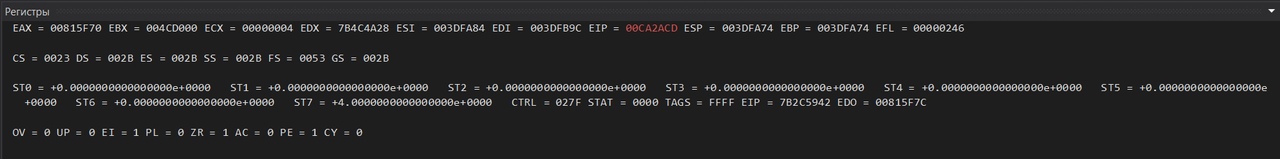


Рисунок 8 – mov ebp, esp



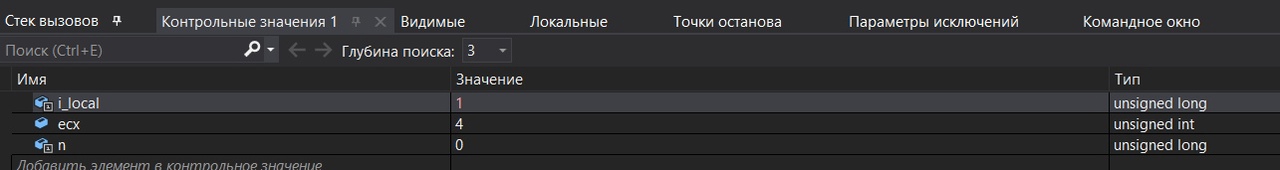


Рисунок 9 – помещаем 1 в счетчик

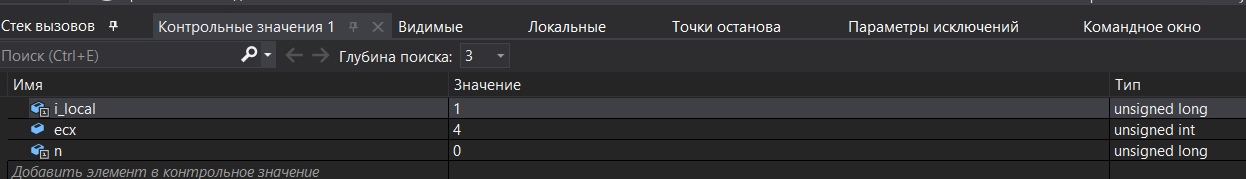
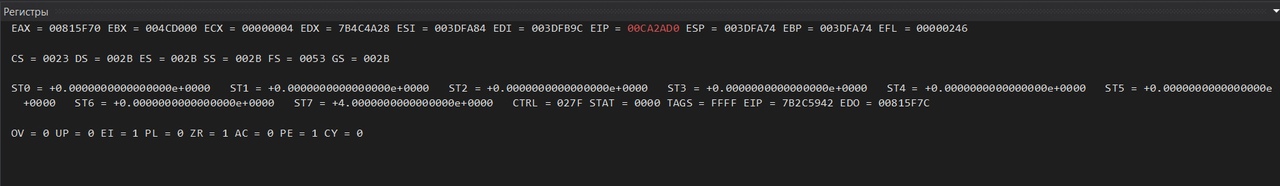


Рисунок 10 – помещаем в ecx число итераций n

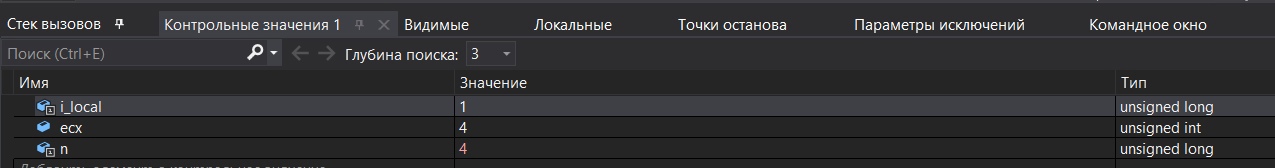
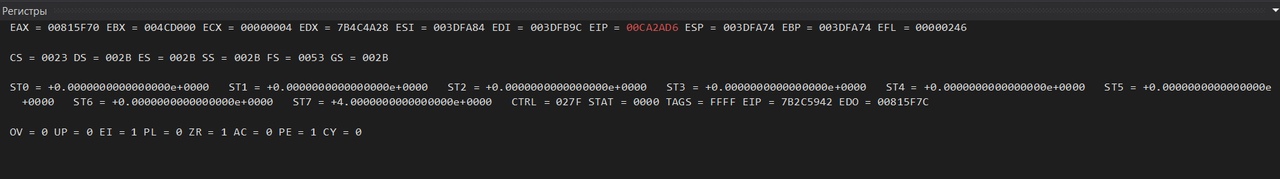


Рисунок 11 – поместили в локальную переменную "n" число итераций из ecx

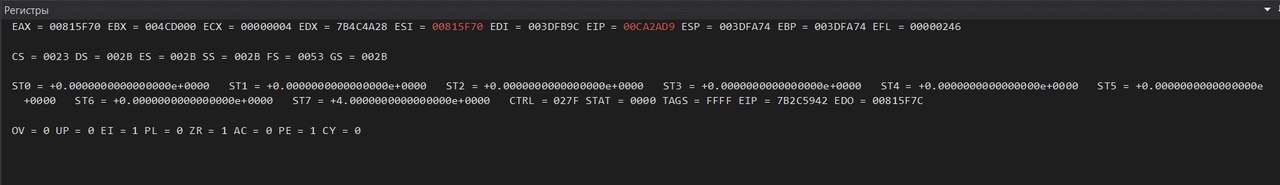


Рисунок 12 – помещаем адрес массива в esi

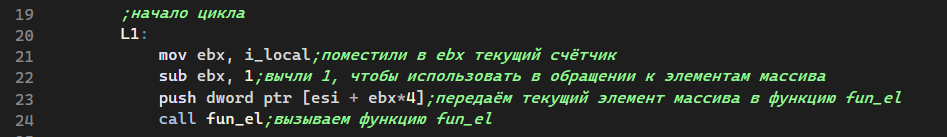


Рисунок 13 – начало цикла

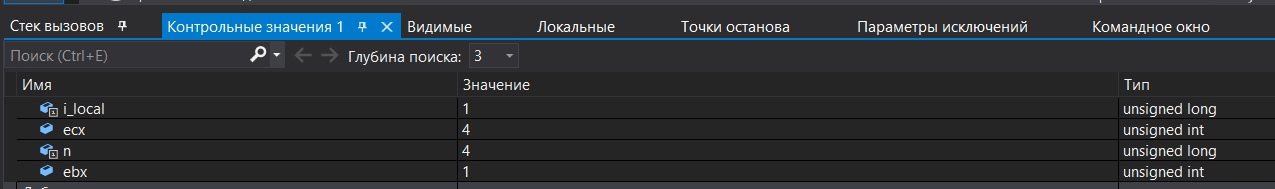
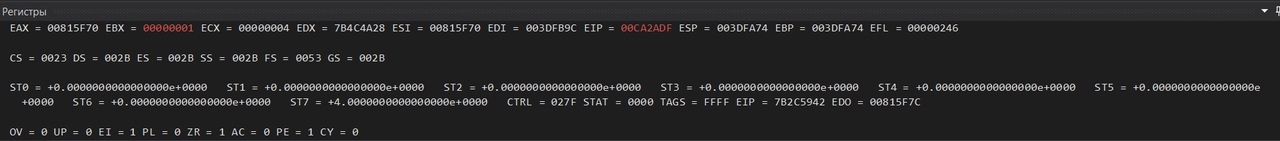


Рисунок 14 – помещаем в ebx текущий счетчик

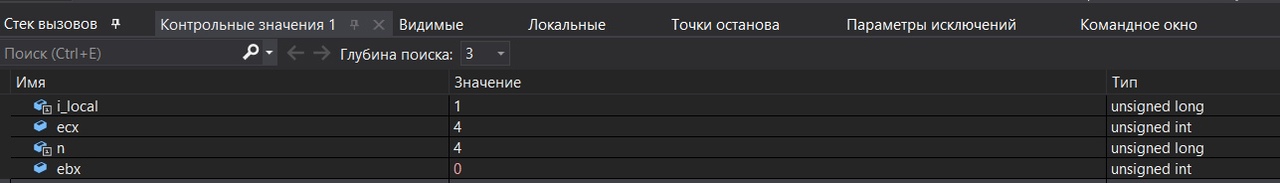
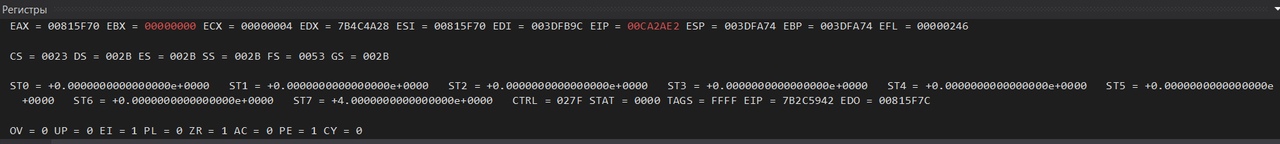


Рисунок 15 – вычитаем из ebx 1

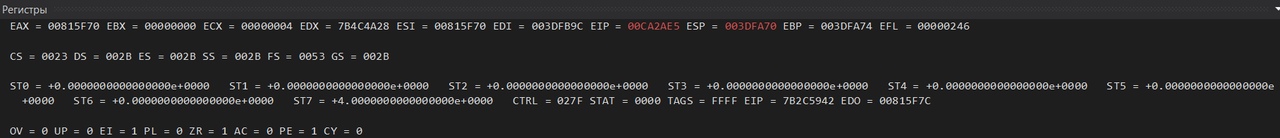


Рисунок 16 – передаем в функцию fun\_el текущий элемент массива

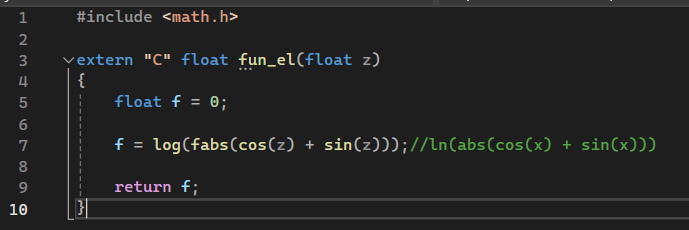


Рисунок 17 – функция fun\_el

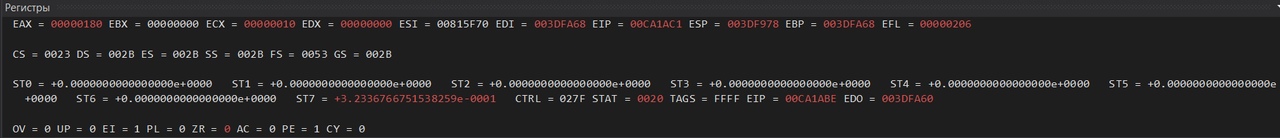
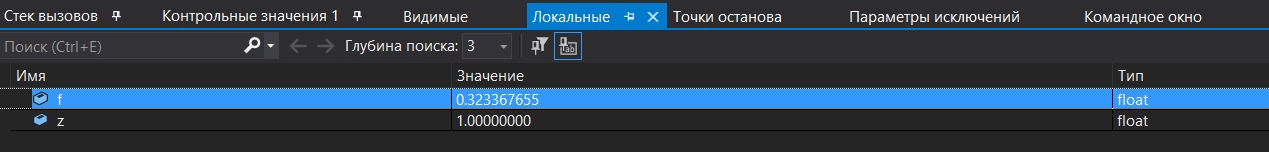


Рисунок 18 – вычисленные значения функции fun\_el

Полученное значение в st(0) – 0.3234, совпадает с теоретическим расчётом на калькуляторе. Значение st(0) представлено на рисунке 19:

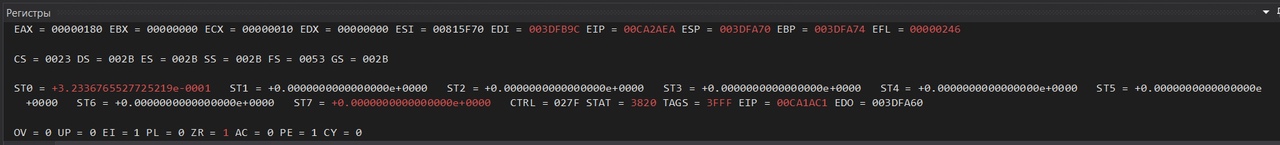


Рисунок 19 – сверяем значение

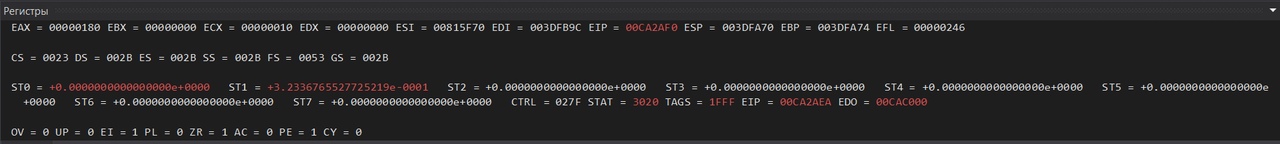


Рисунок 20 – помещаем SUM в st(0)

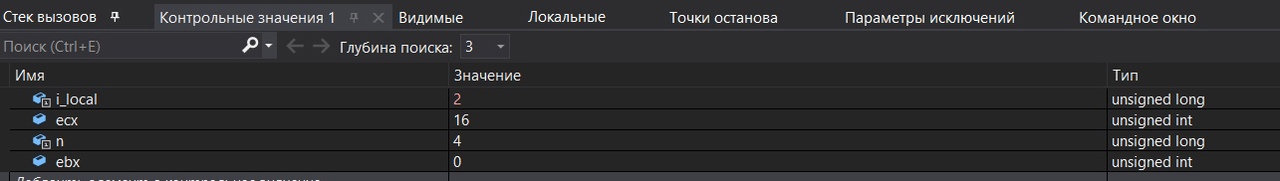
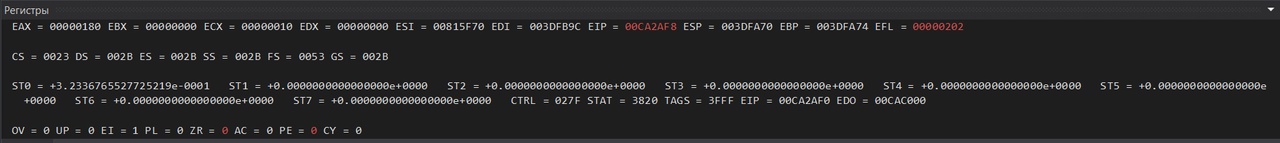


Рисунок 21 – увеличиваем счетчик на 1

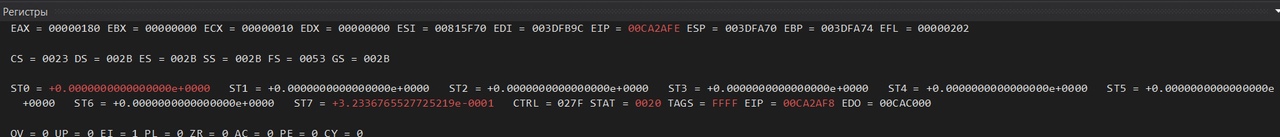


Рисунок 22 – помещаем st(0) в SUM

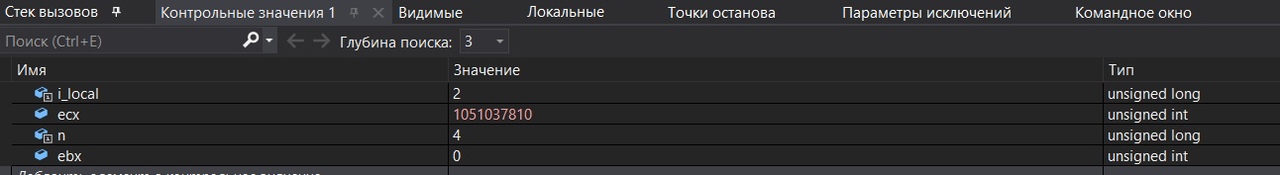
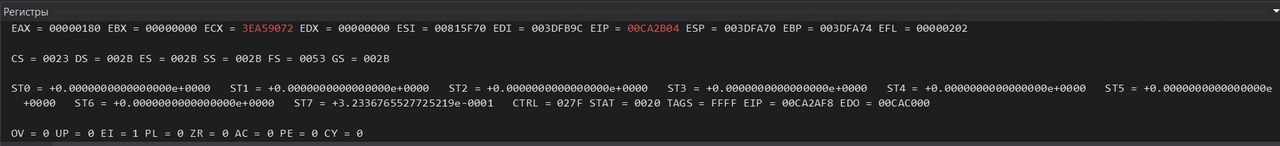


Рисунок 23 – помещаем SUM в ecx

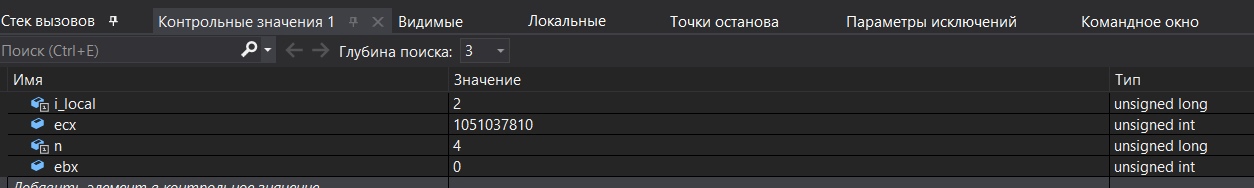
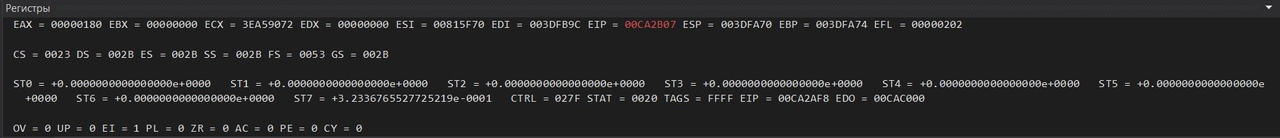


Рисунок 24 – помещаем содержимое ecx в текущий элемент массива по адресу esi + ebx\*4. Обнуляем SUM

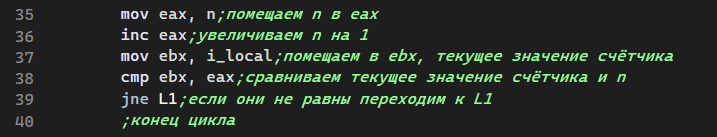


Рисунок 25 – проверяем условия на завершение цикла

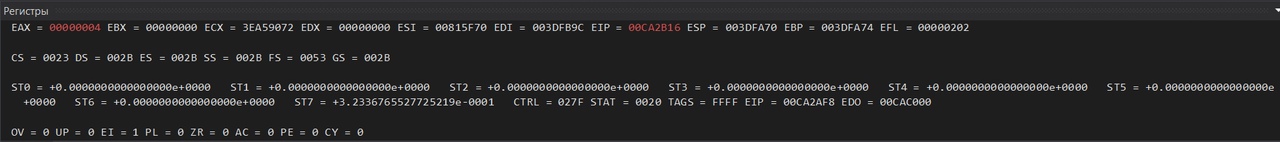


Рисунок 26 – помещаем n в eax

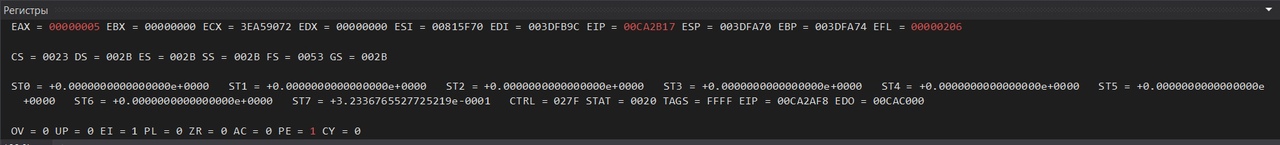


Рисунок 27 – увеличиваем n на 1

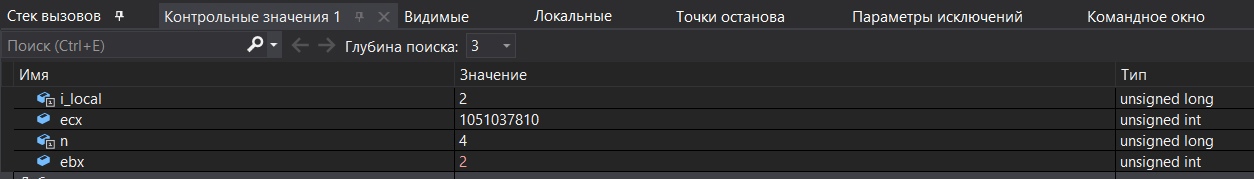
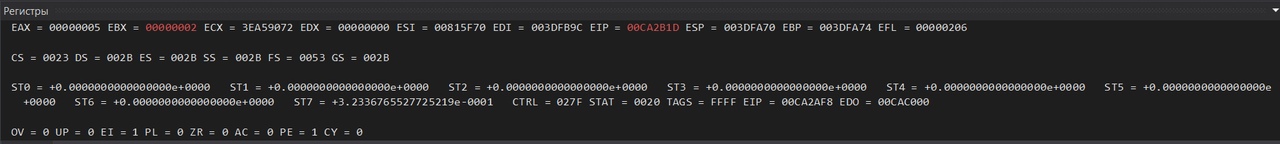


Рисунок 28 – помещаем в ebx текущее значение счётчика

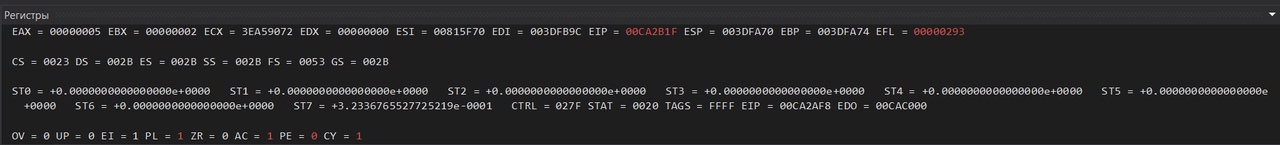


Рисунок 29 – сравниваем n и текущего значения счётчика. Т.к. n больше ebx, то продолжаем цикл

На второй итерации значение функции fun\_el равно -0.7069, что совпадает с теоретическим расчетом. Значение функции fun\_el представлено на рисунке 30:

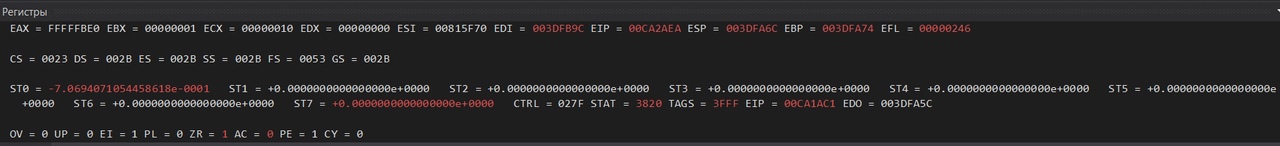


Рисунок 30 – значение функции fun\_el

На третьей итерации значении функции fun\_el равно -0.1638, что так же совпадает с теоретическим расчетом. Третья итерация представлена на рисунке 31:

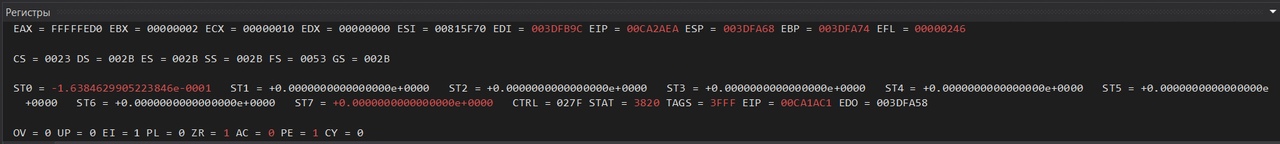


Рисунок 31 – третья итерация функции fun\_el

На четвертой итерации значении функции fun\_el равно 0.3439, что так же совпадает с теоретическим расчетом. Четвертая итерация завершает цикл и представлена на рисунке 32:

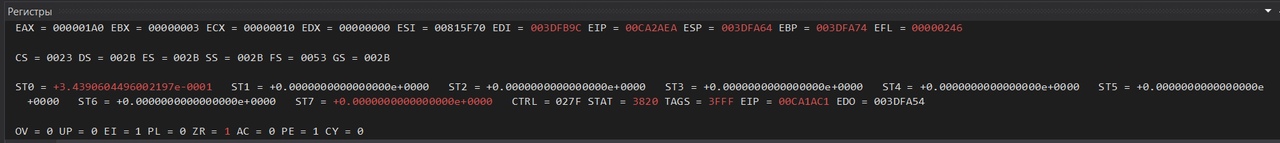


Рисунок 32 – четвертая итерация

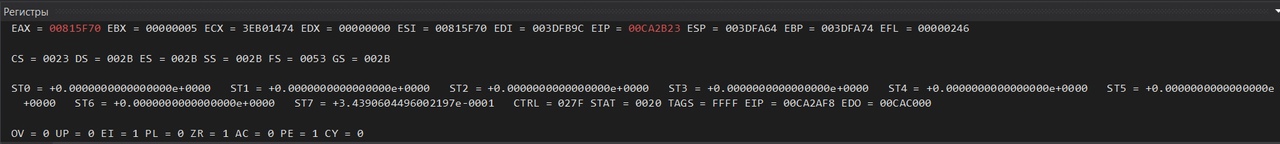
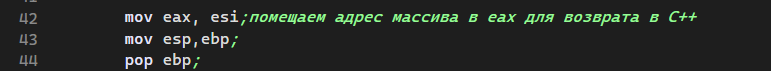


Рисунок 33 – помещаем в eax, esi (адрес массива)

Выходные данные, полученные по завершению четвертой итерации представлены на рисунке 34 и совпадают с теоретическими расчетами:

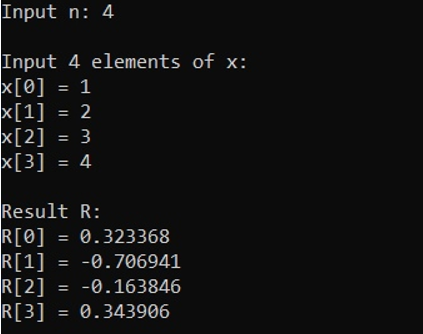


Рисунок 34 – выводимые данные

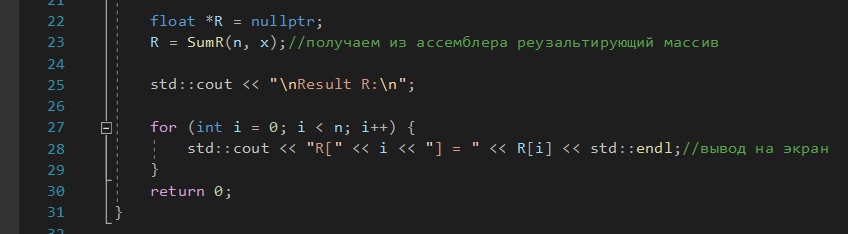


Рисунок 35 – реализованный кодом цикл вывода в терминал

**Вывод:** результат теоретических расчетов совпадает с результатом, выводимым программой.

**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;

}

**FileName.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\_asm.asm**

.686

.MODEL flat,C

.DATA

SUM *dd 0.0;переменная в которую мы поместим результат функци*и на C

i\_lo*cal DD 0*;счётч*ик*

*n dd 0;числ*о итераций

.CODE

exter*n fun\_el:near ; объявление внешней* функции fun\_el

public SumR

Su*m*R proc C

push ebp;

mo*v ebp,esp;*

*mov i\_lo*cal,1;помещаем в счётчик 1

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

*mov n, ecx;поместили в локальную переме*нную "n" число итера*ций из ecx*

*mov esi, [ebp + 1*2*];поместили а*дрес массива в esi

*;начало цикла*

*L1:*

*mov ebx,* i\_local;пом*естили в ebx текущий счётчик*

*sub ebx, 1;вычли 1, чтобы ис*пользовать в обращении к элеме*нтам массива*

*push dword ptr [esi + ebx\*4];перед*аём текущий э*лемент массива в функцию* fun\_el

*call fun\_el;вызывае*м функ*цию fun\_el*

*fld SUM;п*омещаем в st(*0) SUM*

*fadd;складываем* st(0) и s*t(1)*

*inc i\_local;увеличиваем* счётчик на 1

*fstp SUM;помещае*м значение из st(0) в SU*M*

*mov ecx, SUM;помещаем SUM в ecx*

*mov [esi +* ebx\*4], ecx;*помещаем в те*кущий элеме*нт массива значен*ие из ec*x*

*mov SUM, 0;обн*уляем SUM

mov *eax, n;помещаем n в eax*

*inc eax;увеличи*ваем n на 1

*mov ebx, i\_local;помещаем в ebx, текущее* значен*ие счётчика*

*cmp ebx, eax;сравни*в*аем текущее* значение счёт*чика и n*

*jne L1;если они не равны переходим к* L1

;конец *ц*икла

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

mov esp,ebp;

pop ebp;

ret

SumR endp

End