МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования «Вятский государственный университет»

Институт математики и информационных систем

Факультет автоматики и вычислительной техники

Кафедра радиоэлектронных средст

Отчет по лабораторной работе №4

по дисциплине «Цифровые устройства и микропроцессоры»

“ МОДУЛЬНОЕ ПРОГРАММИРОВАНИЕ”

Вариант 13

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Выполнил: | студент группы  ИНБб-3301-02-00 |  |  |  | М.И. Солодянкин | |
|  |  |  |  |  |  |
| Проверил: | Педагог Колледж ВятГУ |  |  |  | М.А. Земцов |
|  |  |  |  |  |  |

г. Киров

2025

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

**Выполнение индивидуального задания**

1. Сформировать проект консольного приложения с использованием

модуля на основе языка ассемблера.

2. Разработать основной модуль приложения. Основной модуль обеспечивает ввод данных, вызов подпрограммы ассемблера и вывод результата.

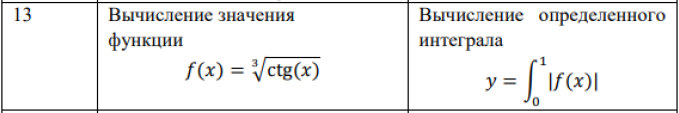
3. Добавить в проект модуль на языке ассемблера. Разработать подпрограмму на языке ассемблера в соответствии с вариантом.

4. Добавить в проект модуль на языке Си. Разработать функцию на языке высокого уровня в соответствии с вариантом.

5. Обеспечить вызов подпрограммы ассемблера из основного модуля на языке С++. Основной модуль обеспечивает ввод данных и вывод результата.

6. Осуществить вызов функции языка Си из модуля ассемблера.

7. Проверить работу приложения в режиме отладки. Записать содержимое стека перед вызовом подпрограммы ассемблера и функции Си.



**Текст программы:**

**Source.cpp**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <cmath>

#include <iostream>

using namespace std;

extern "C" double integr(double x);

int main(int argc, char\*\* argv)

{

double x;

cout << "Input x= ";

cin >> x;

double R = 0.0;

R = integr(x);

cout << "Result= " << R << endl;

return 0;

}

extern "C" double cotan(double z)

{

double f, cot;

cot = cos(z) / sin(z);

f = cbrt(cot);

return f;

}

**Asmblr.asm**

.586

.MODEL flat,C

.DATA

SUM DD 0.0

i\_local DD 0

.CODE

extern cotan:near ; объявление внешней функции cotan

public integr

integr proc C

push ebp

mov ebp,esp

mov ecx, dword ptr [ebp+8]

push dword ptr [ebp + 12]

push i\_local

call cotan

fabs

fld SUM

add esp,8

fadd

pop ecx

inc i\_local

fstp SUM

fld SUM

mov esp,ebp

pop ebp

ret

integr endp

End

**Верификация программы**

Проверка при x = 5

f(x) =

y =

Шаги программы при выполнении:

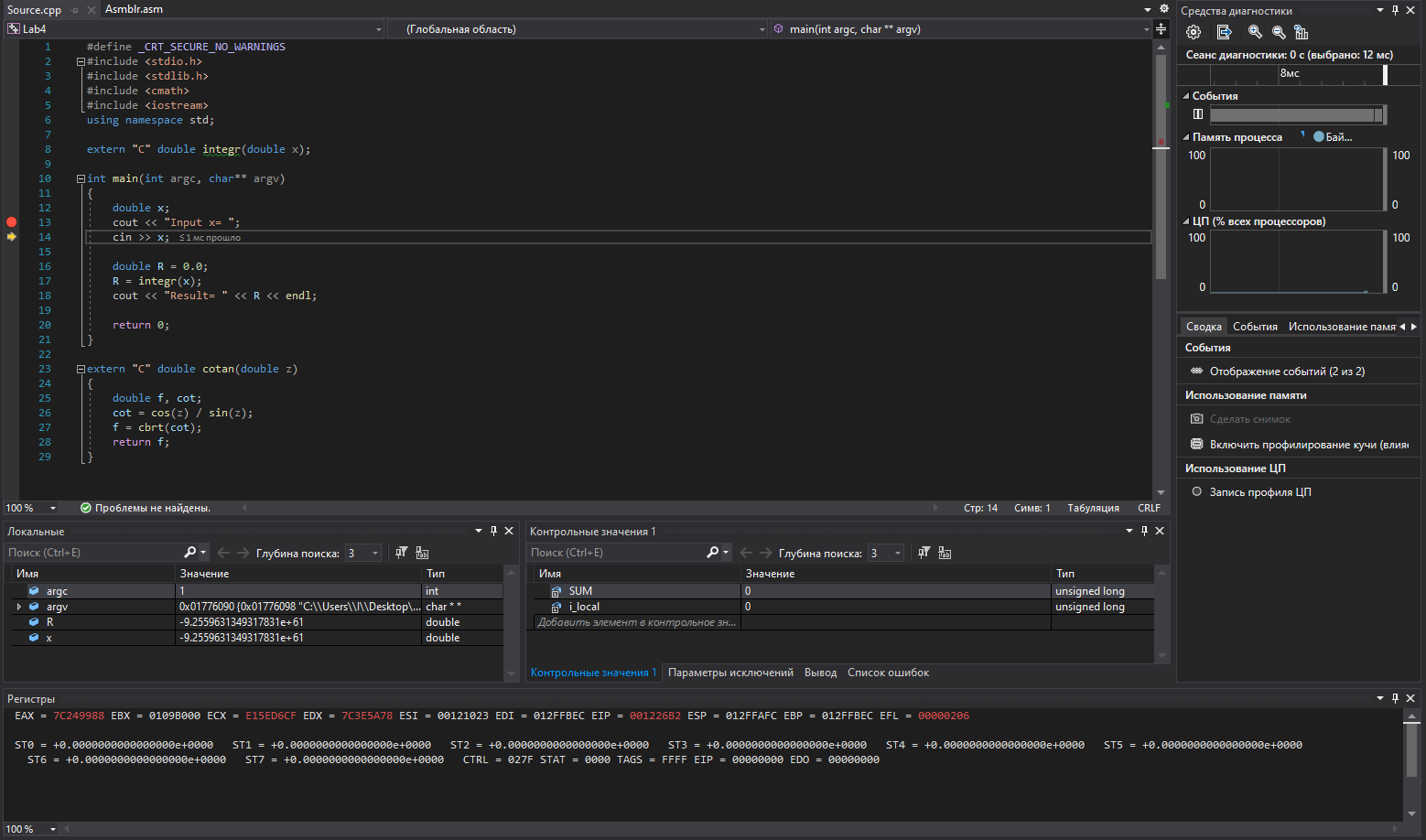


Рисунок 1 – Считывание значения x с клавиатуры

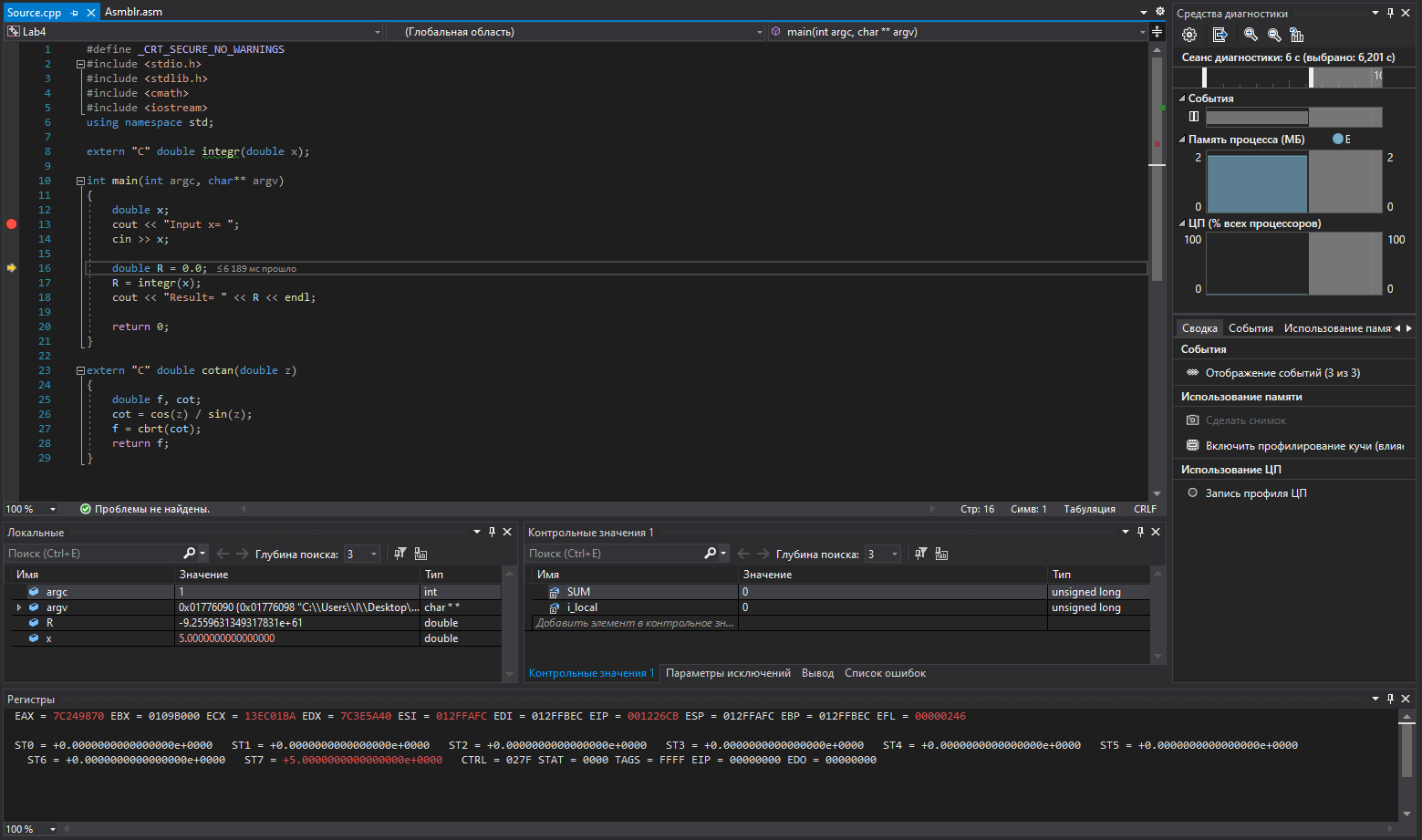


Рисунок 2 – Инициализация переменной R для хранения результата

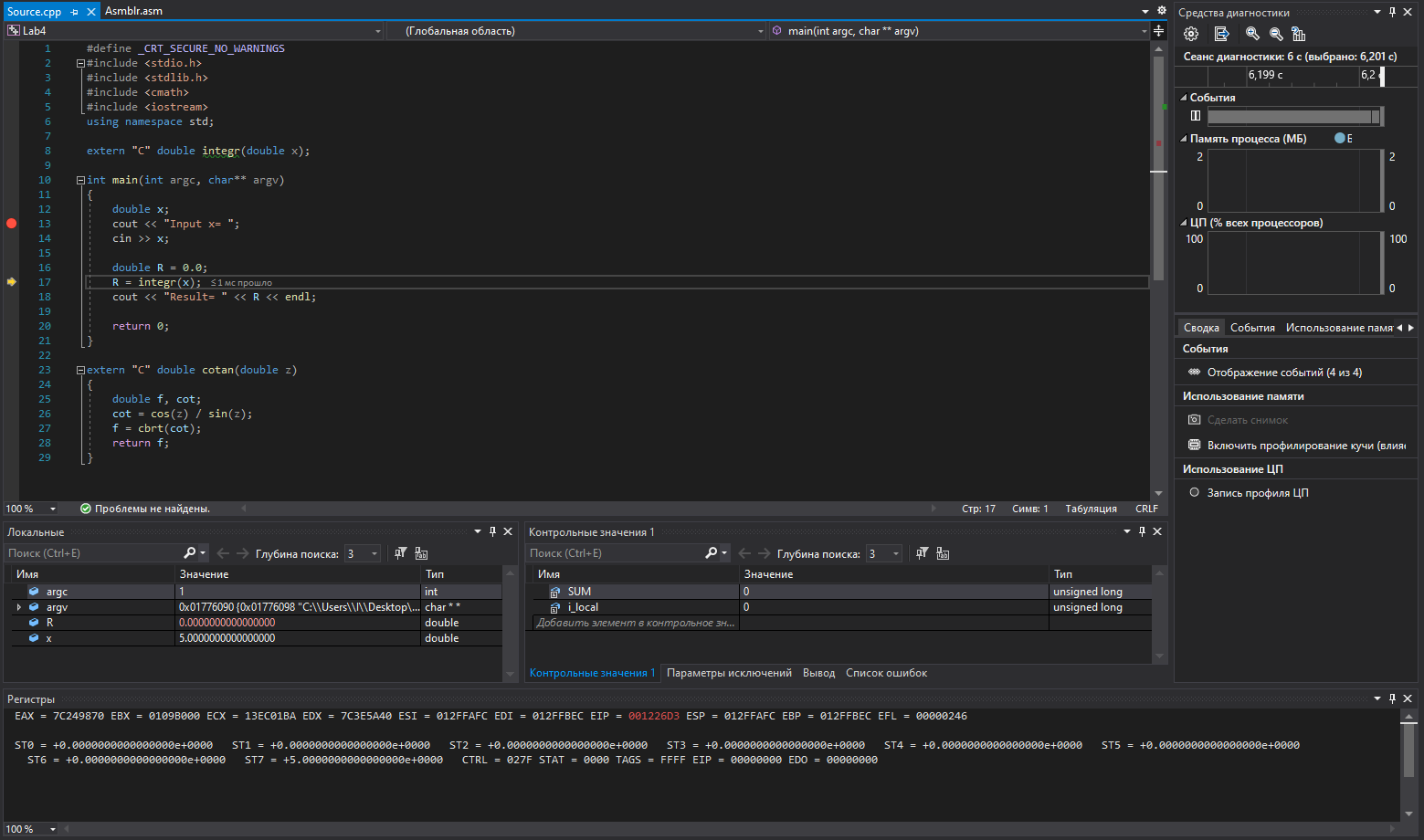


Рисунок 3 – Вызов функции integr с аргументом x и сохранение результата в R

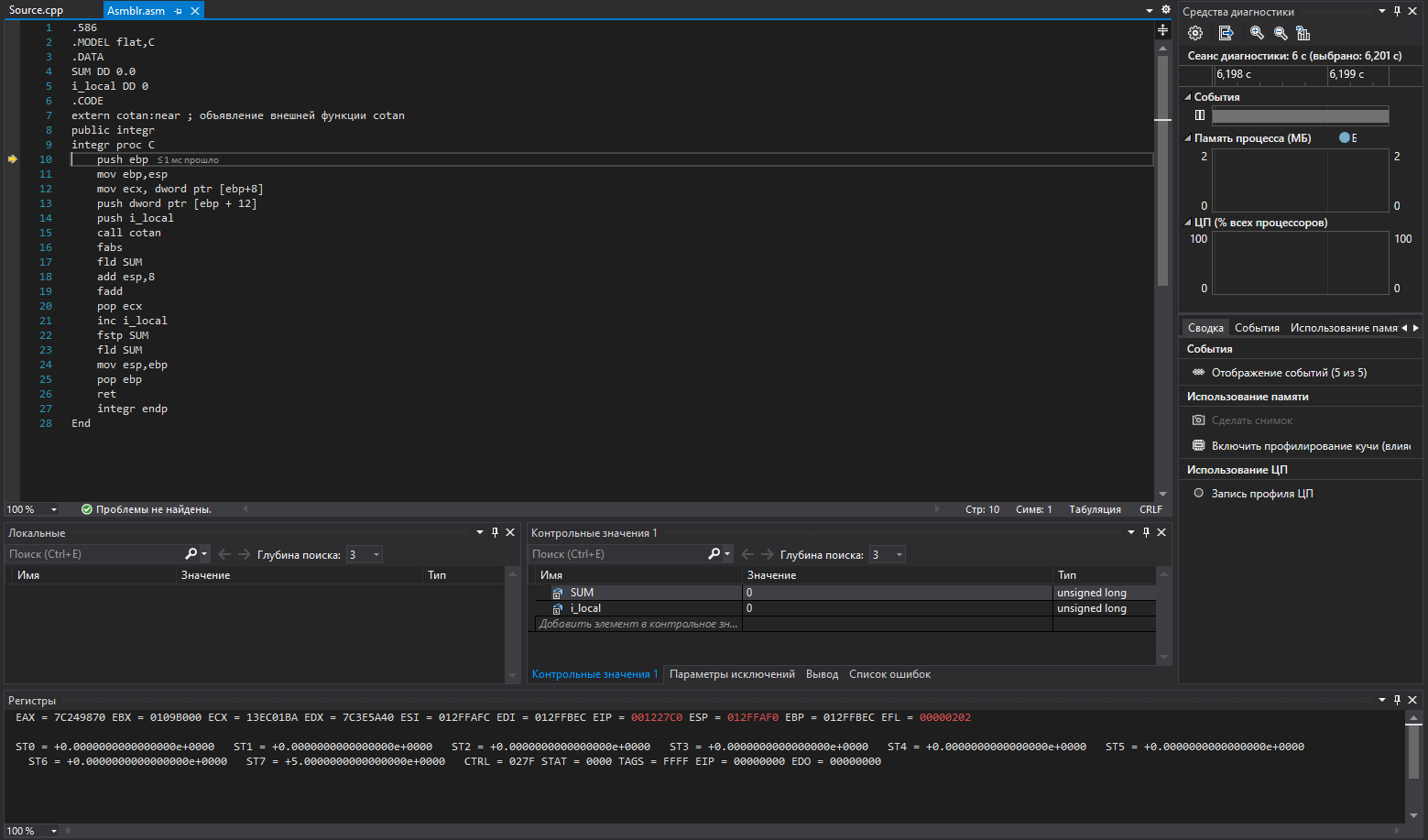


Рисунок 4 – Сохранение базового указателя текущего стека

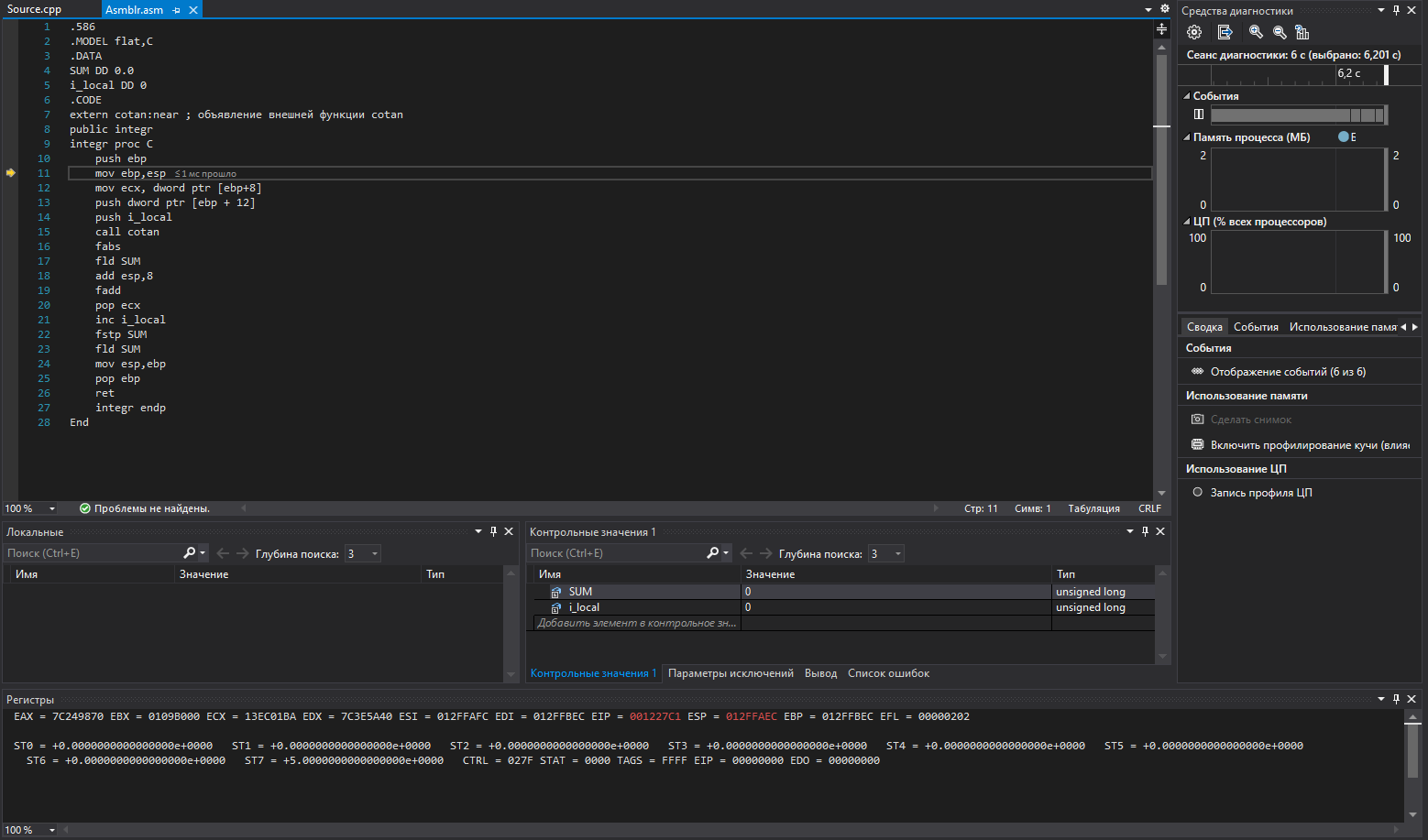


Рисунок 5 – Установка нового базового указателя на вершину стека

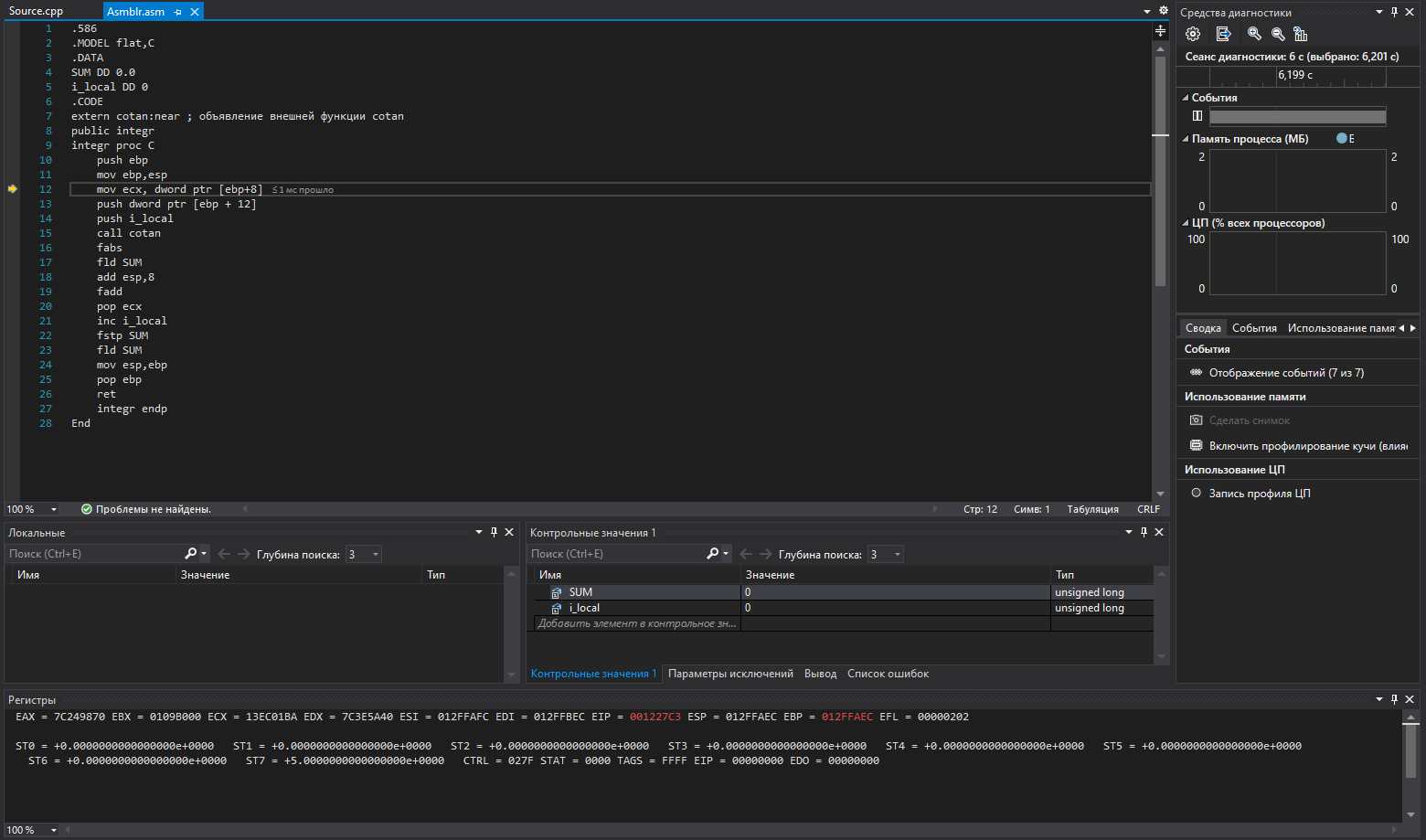


Рисунок 6 – Получение первого аргумента (x) из стека

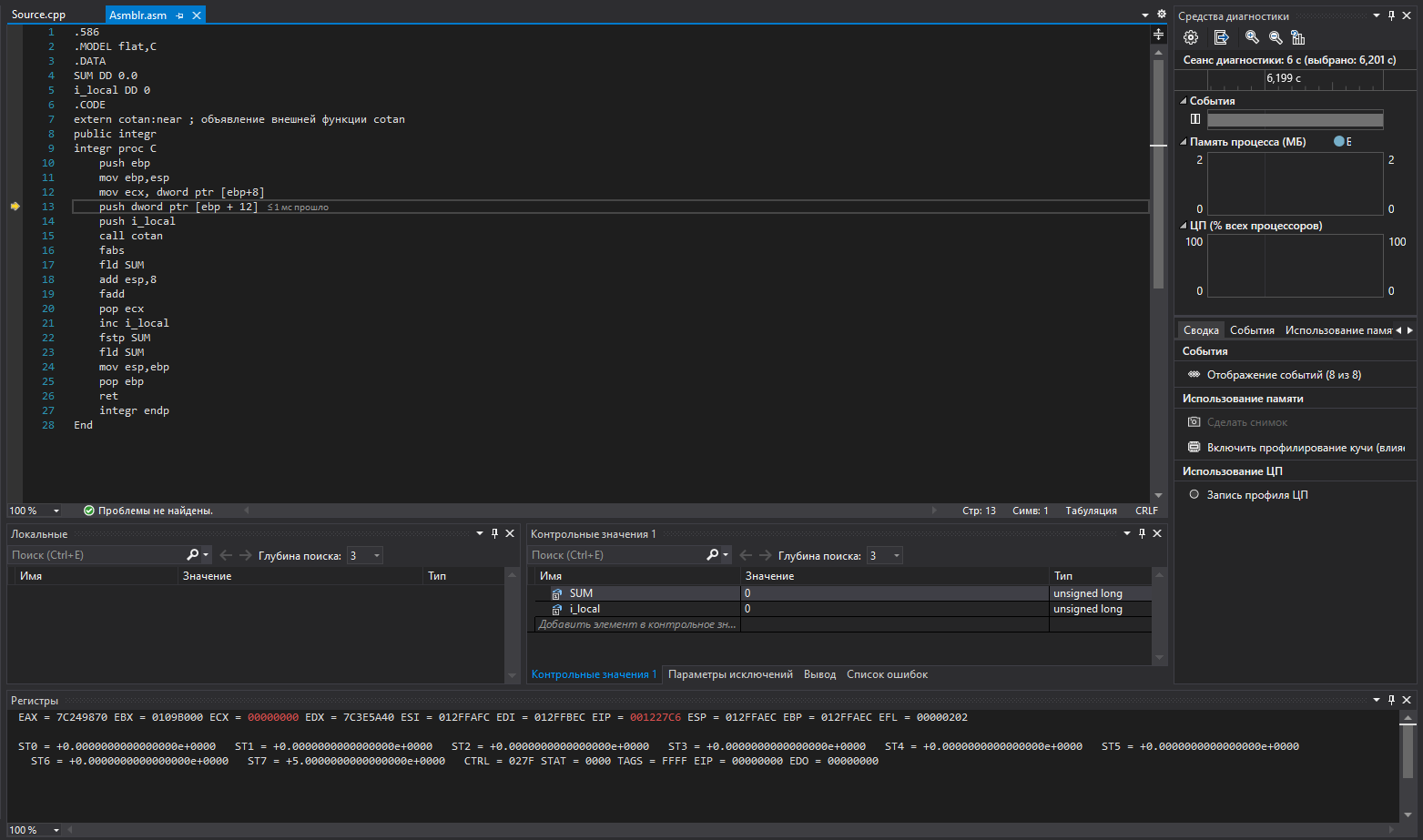


Рисунок 7 – Передача второго аргумента

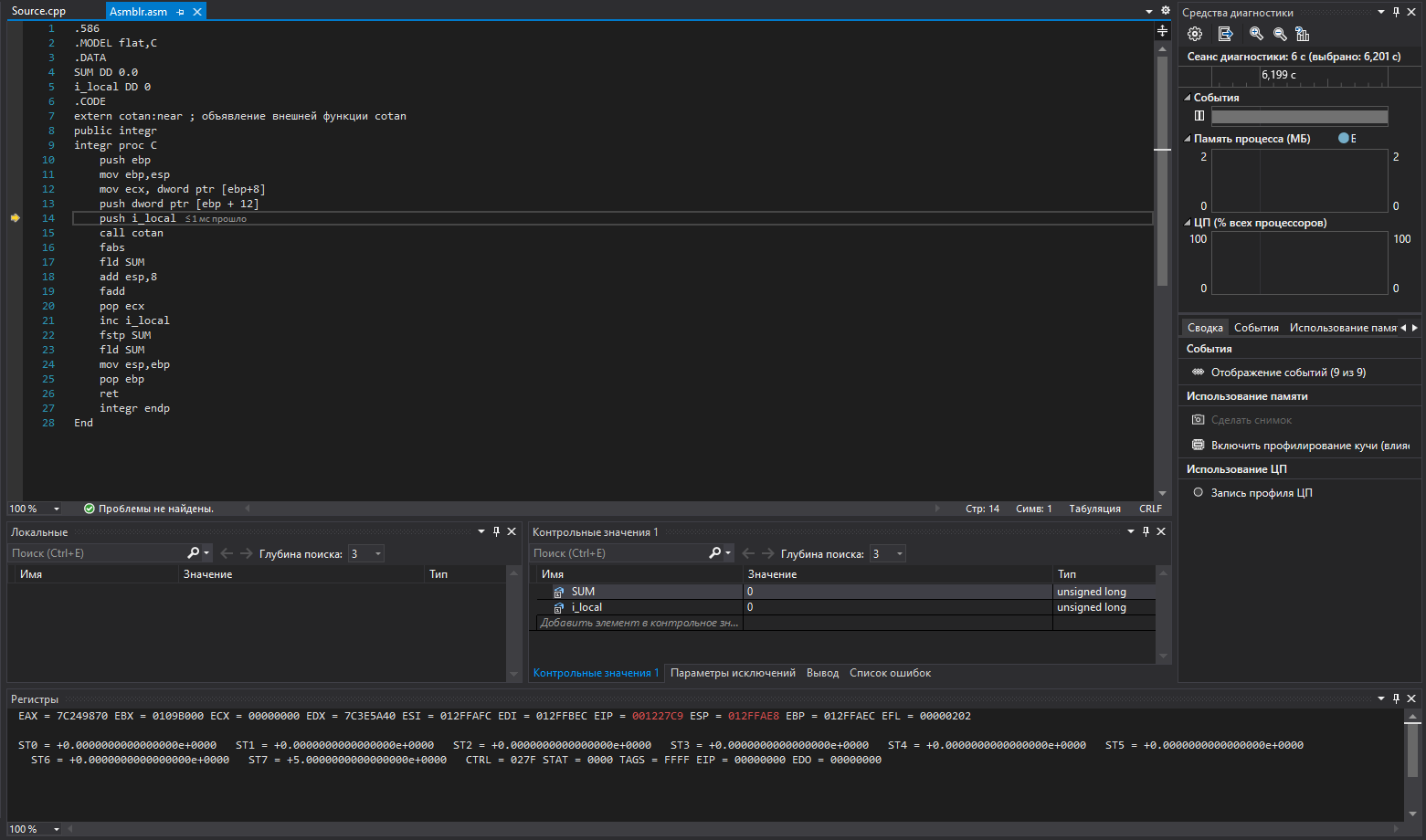


Рисунок 8 – Передача локальной переменной i\_local в стек

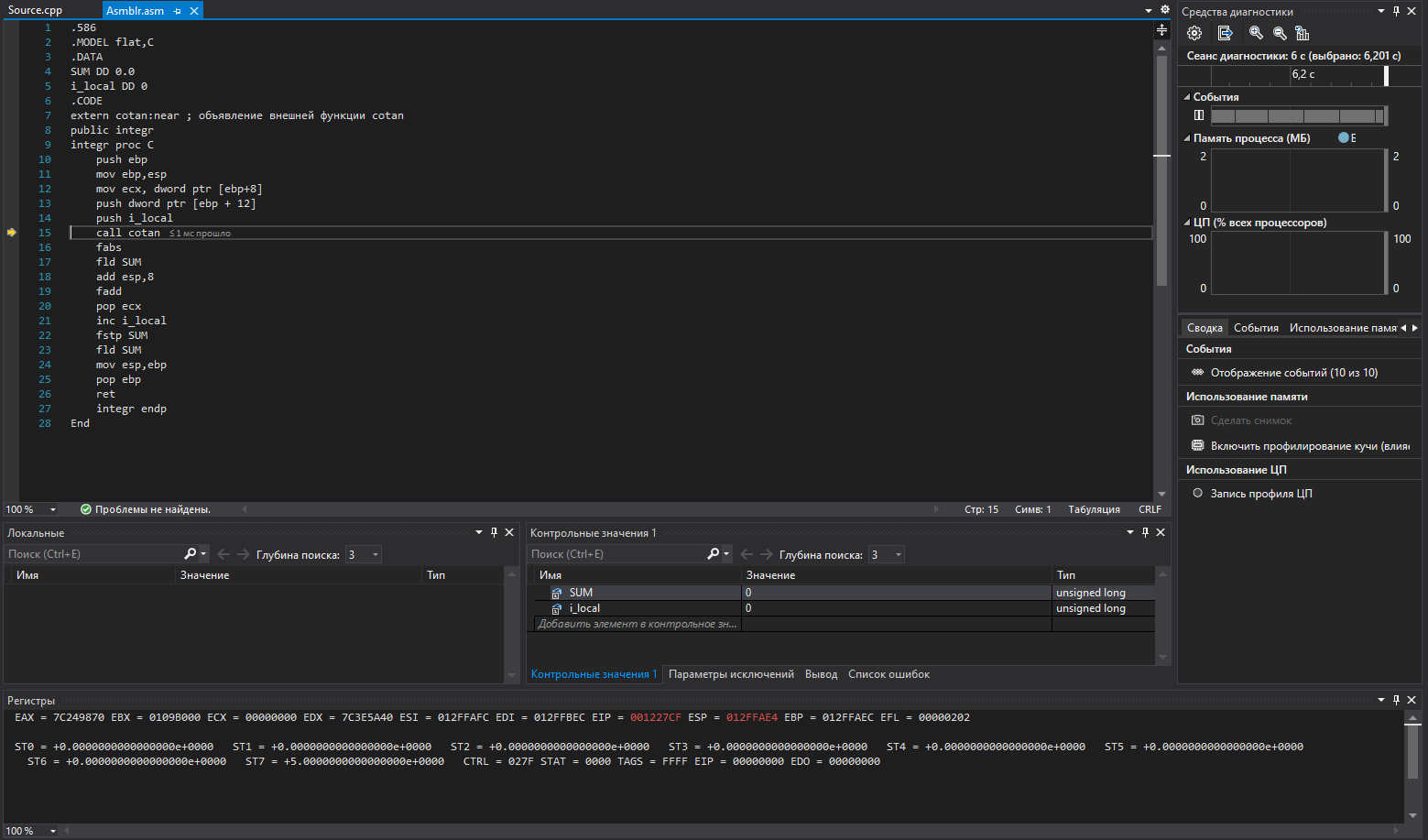


Рисунок 9 – Вызов функции cotan

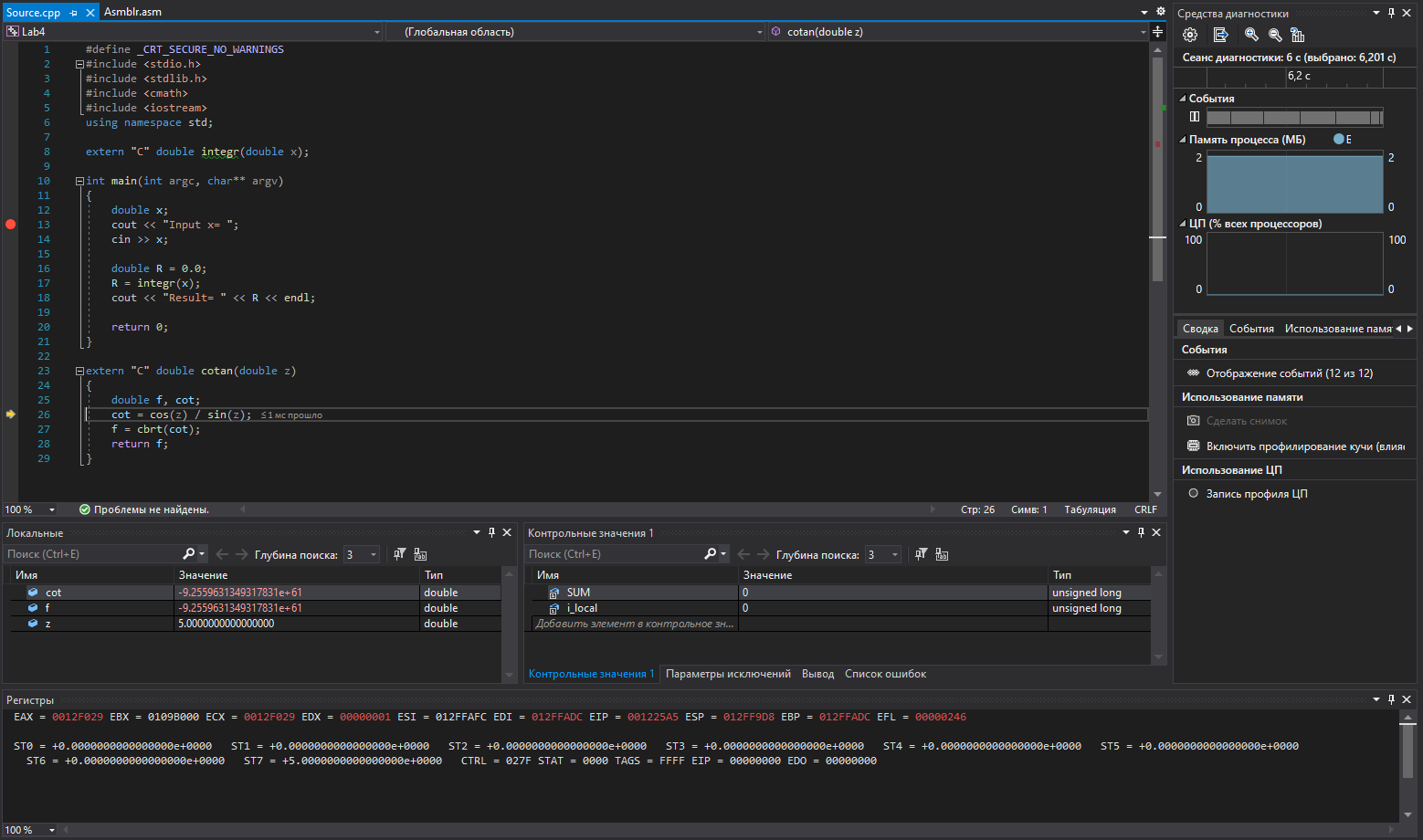


Рисунок 10 – Вычисление котангенса z

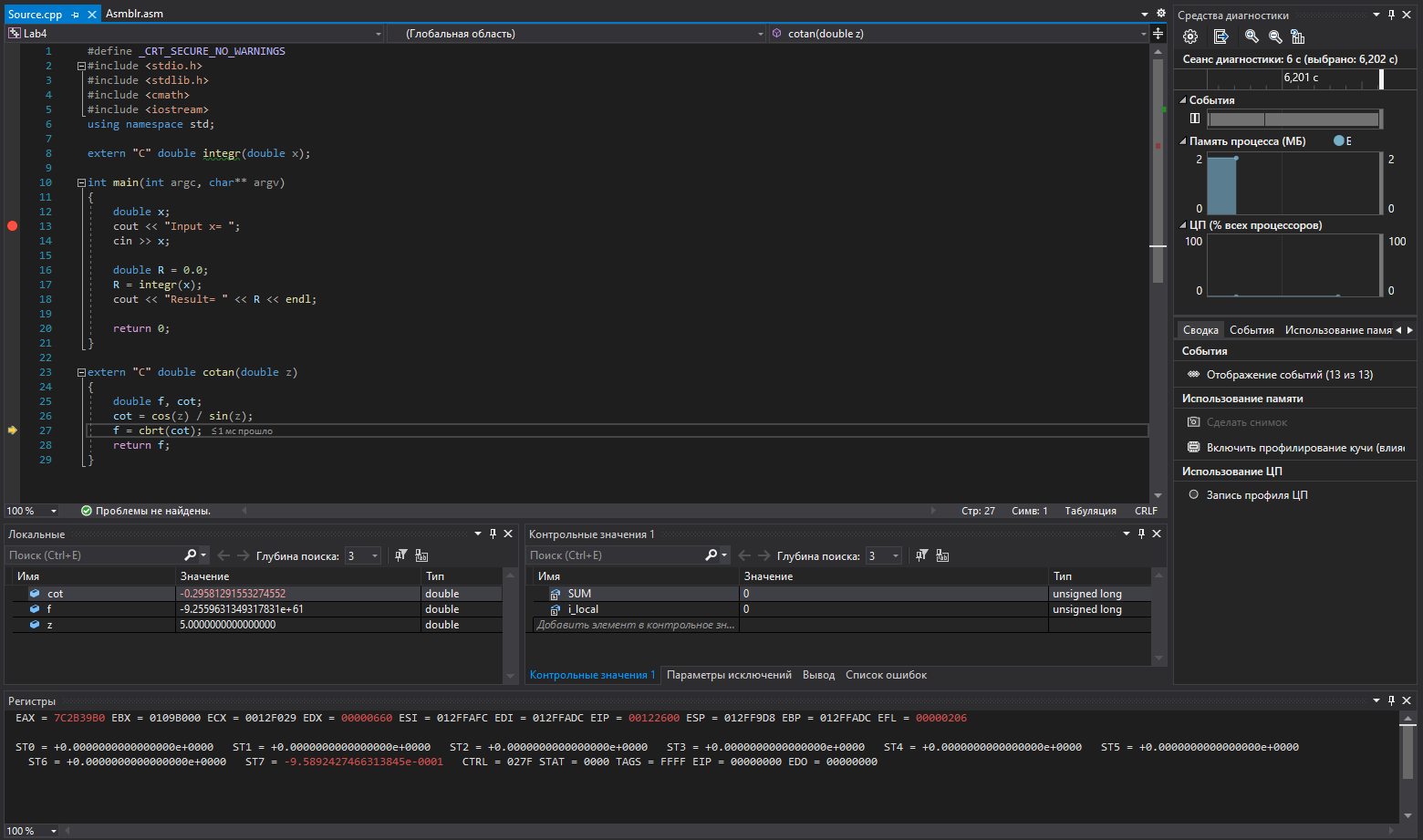


Рисунок 11 – Вычисление кубического корня из котангенса

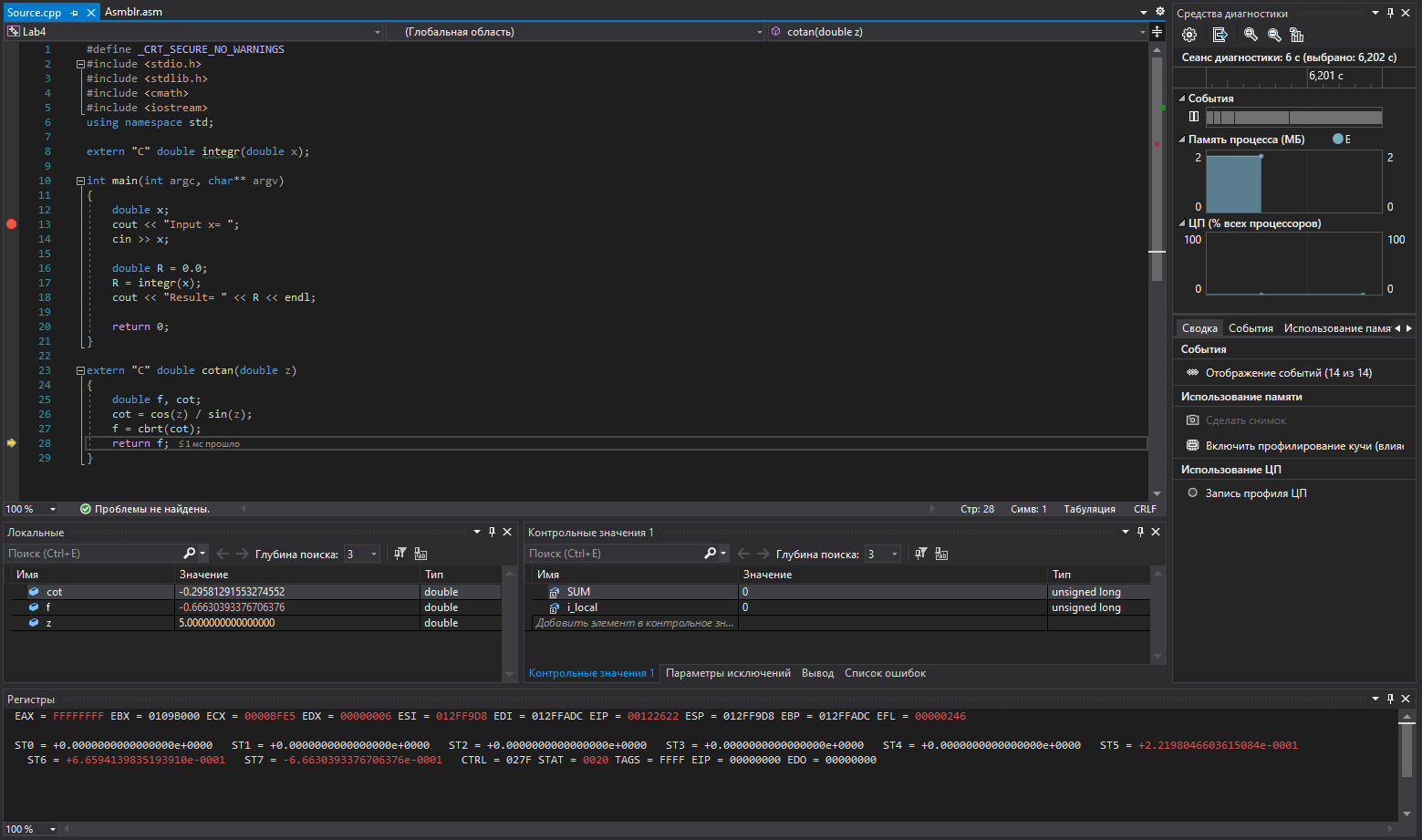


Рисунок 12 – Возврат результата

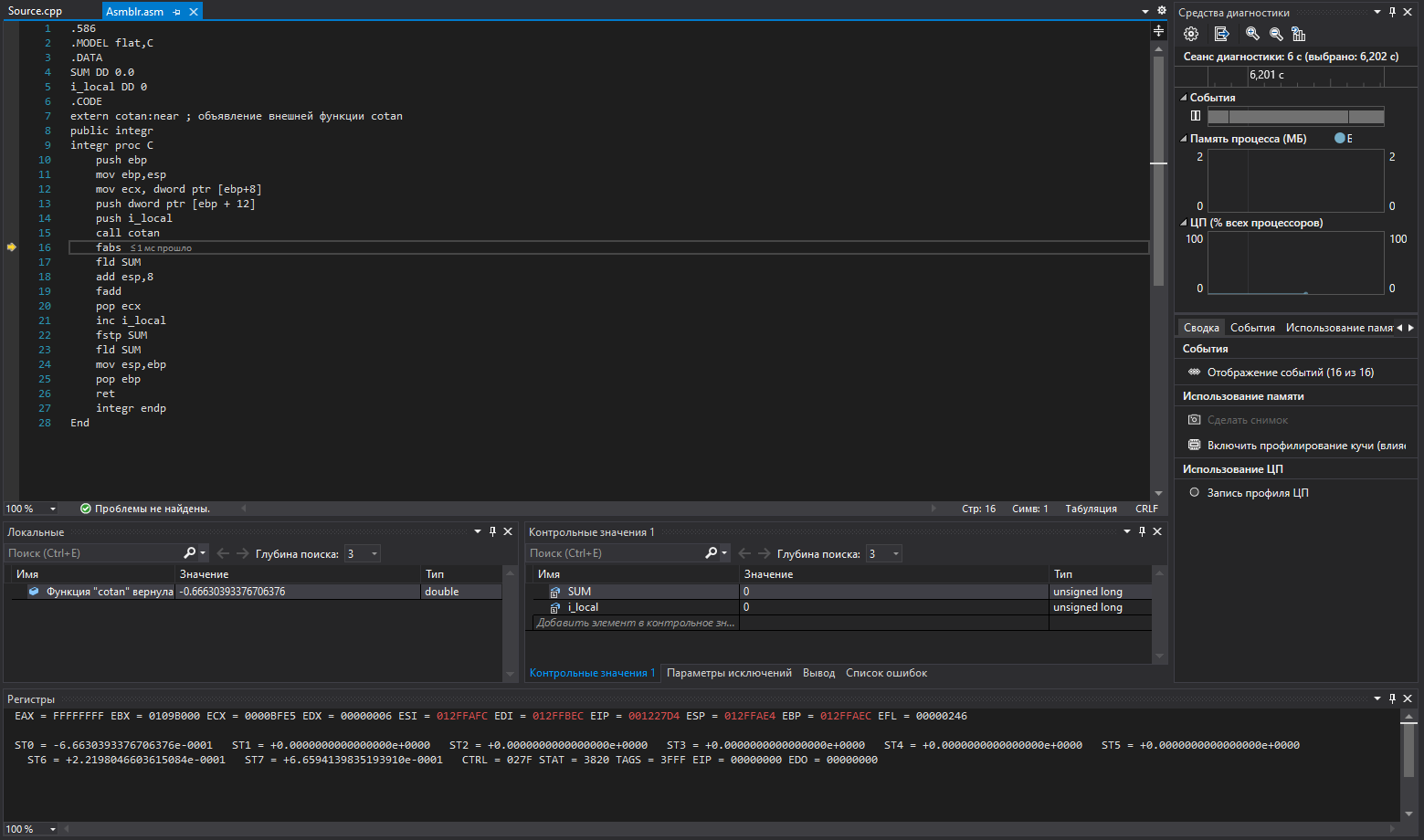


Рисунок 13 – Вычисление абсолютного значения результата (f(x) = -0,666304)

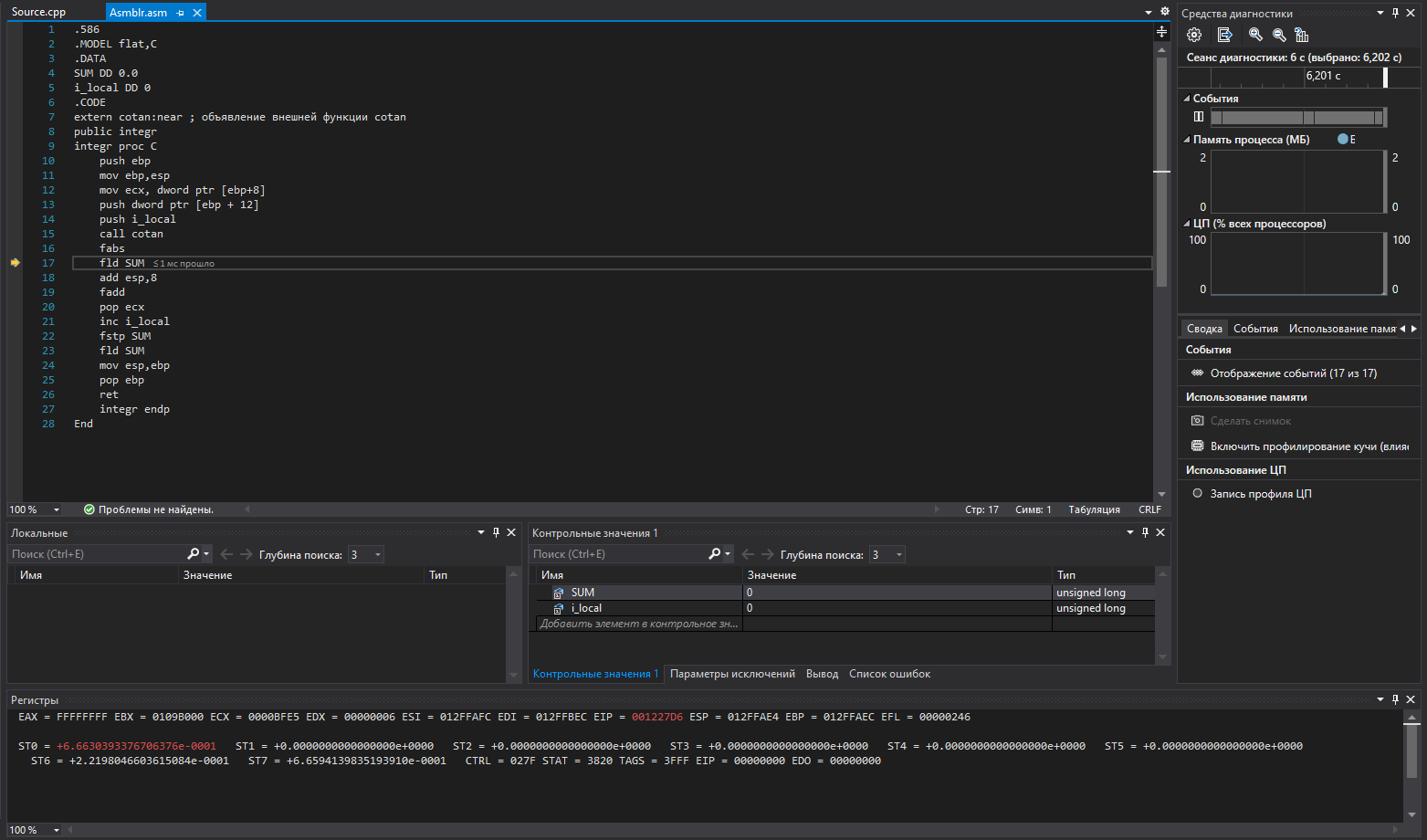


Рисунок 14 – Загрузка текущего значения SUM в стек

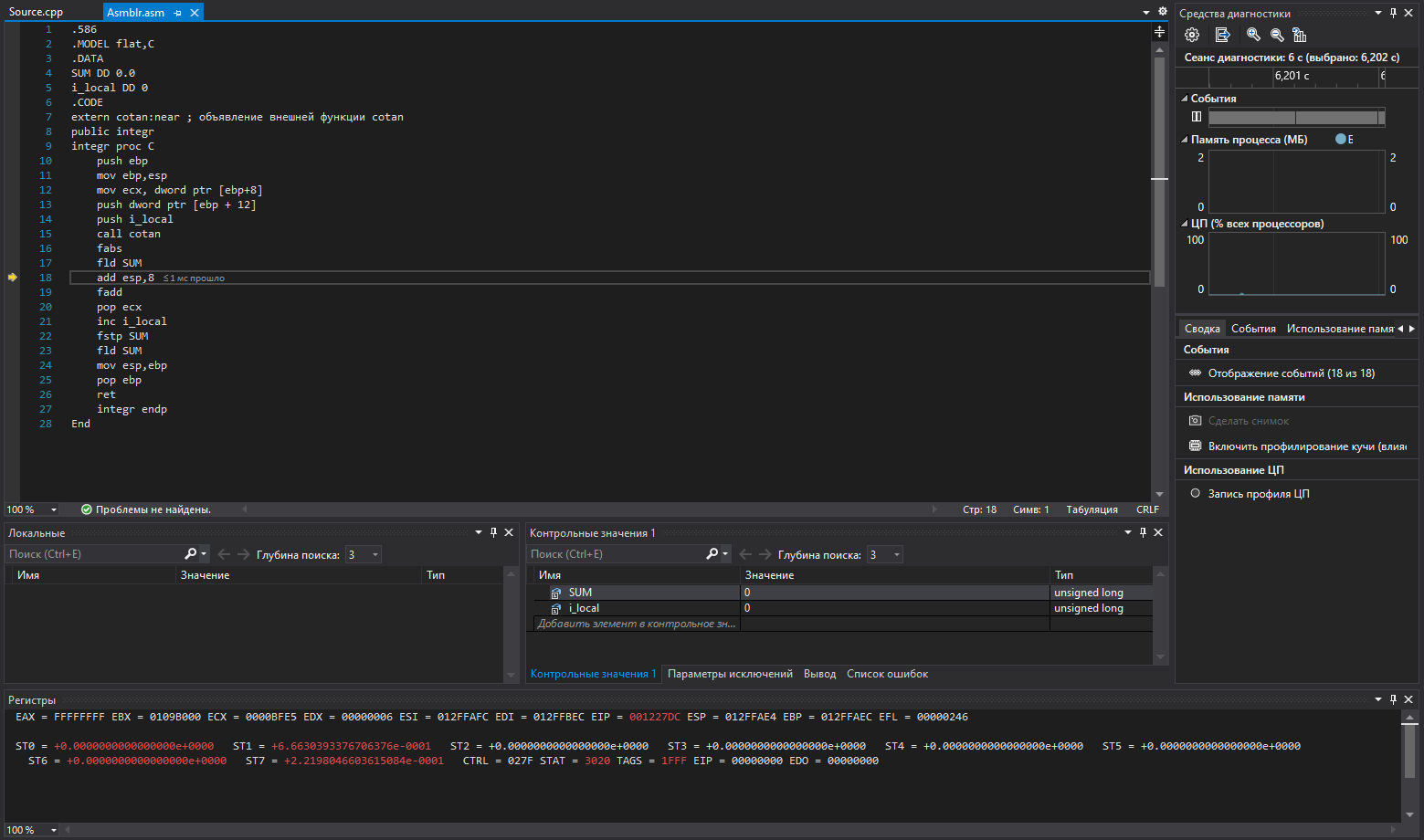


Рисунок 15 – Очистка стека от переданных аргументов (2 аргумента по 4 байта)

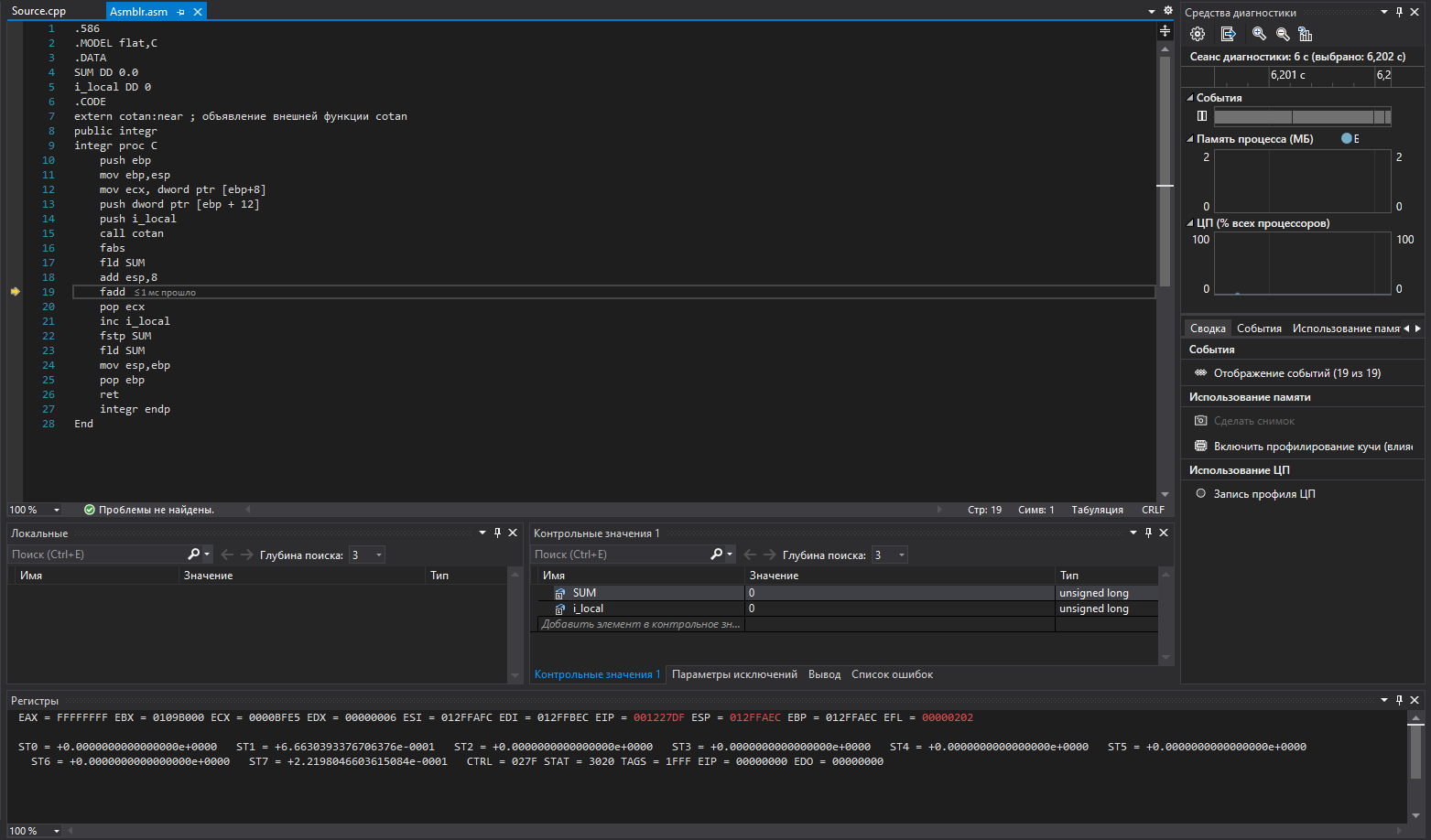


Рисунок 16 – Сложение верхнего значения стека с SUM

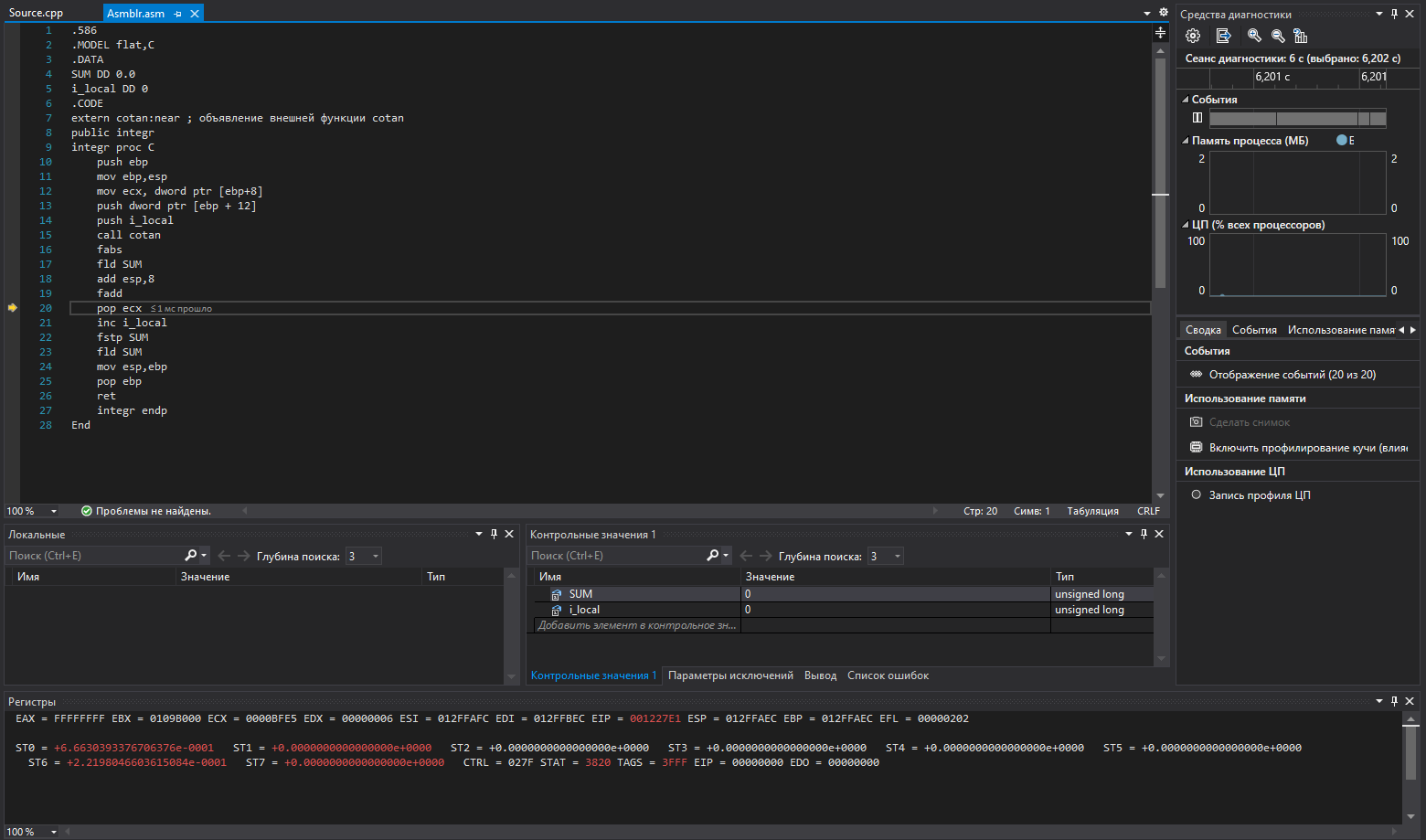


Рисунок 17 – Восстановление значения ecx из стека

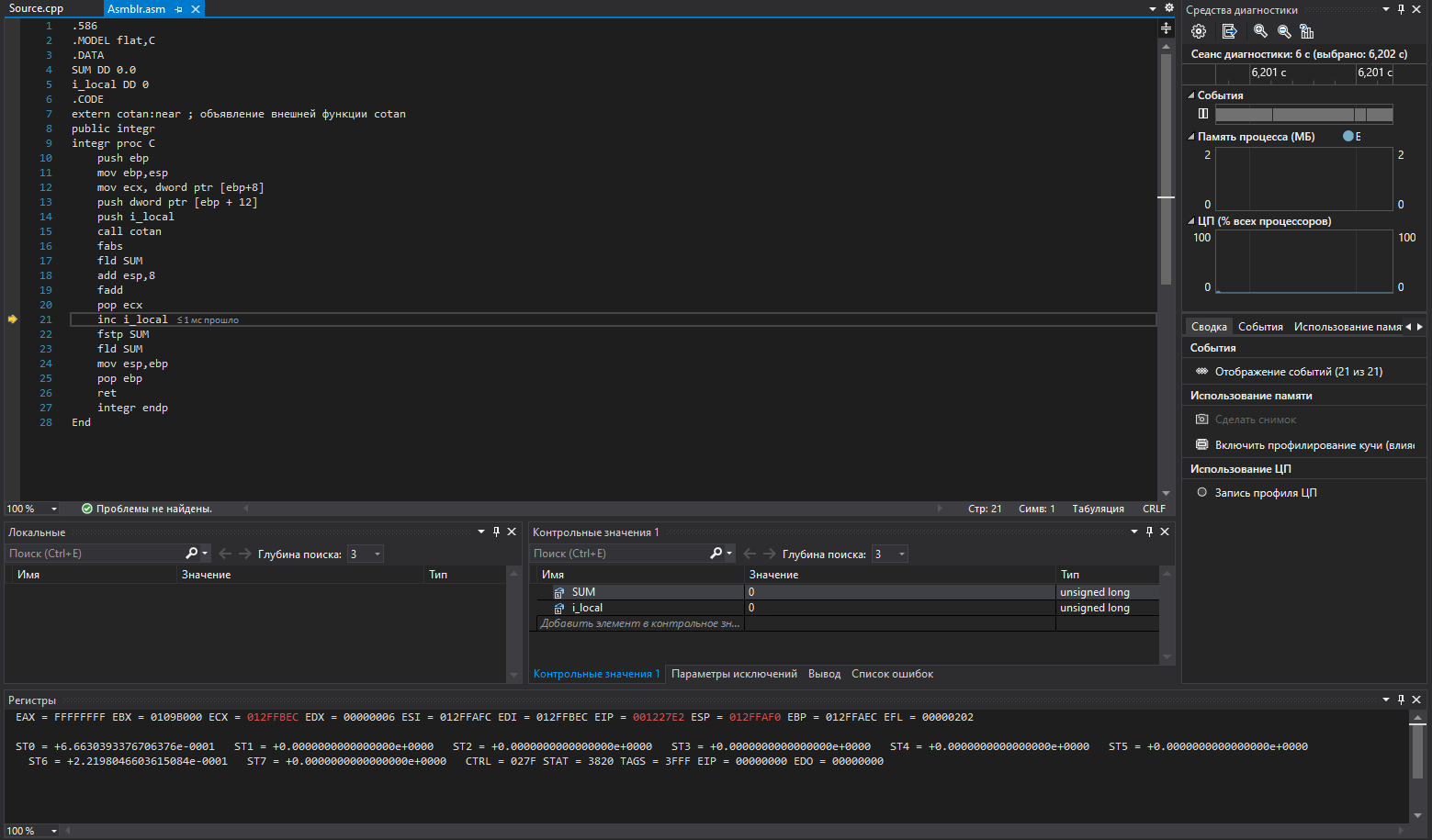


Рисунок 18 – Увеличение локального индекса i\_local на 1

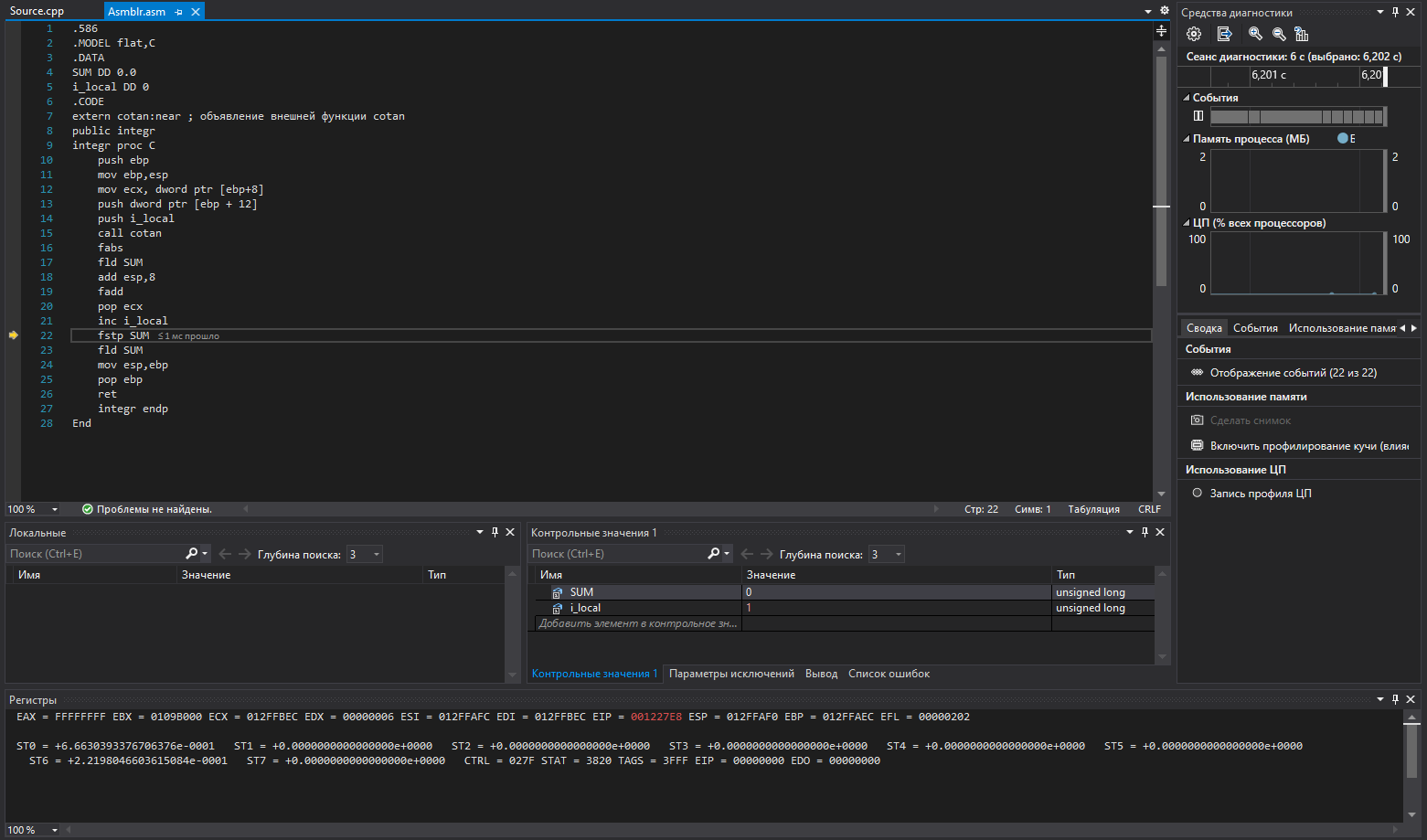


Рисунок 19 – Сохранение результата обратно в переменную SUM

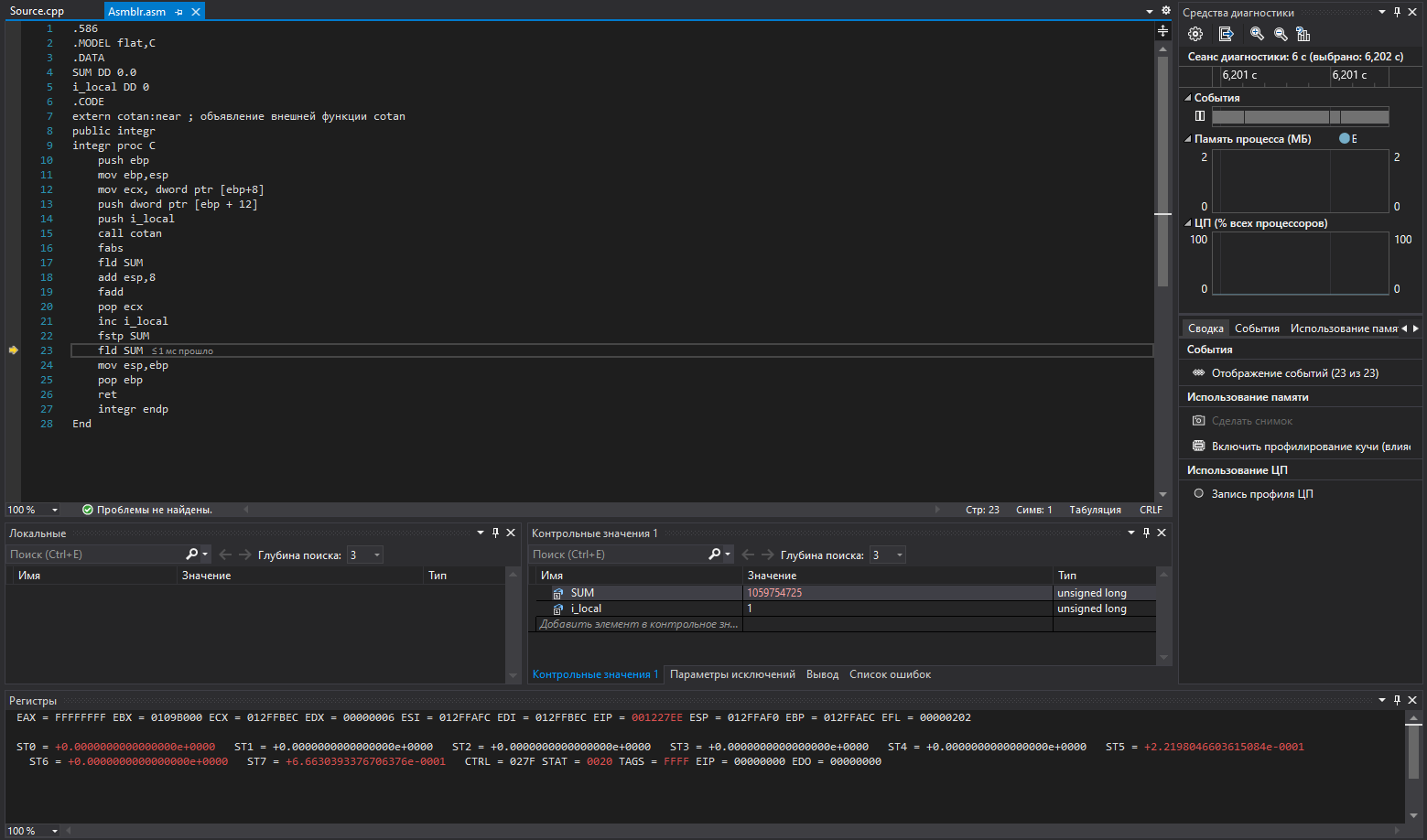


Рисунок 20 – Загрузка обновленного значения SUM в стек

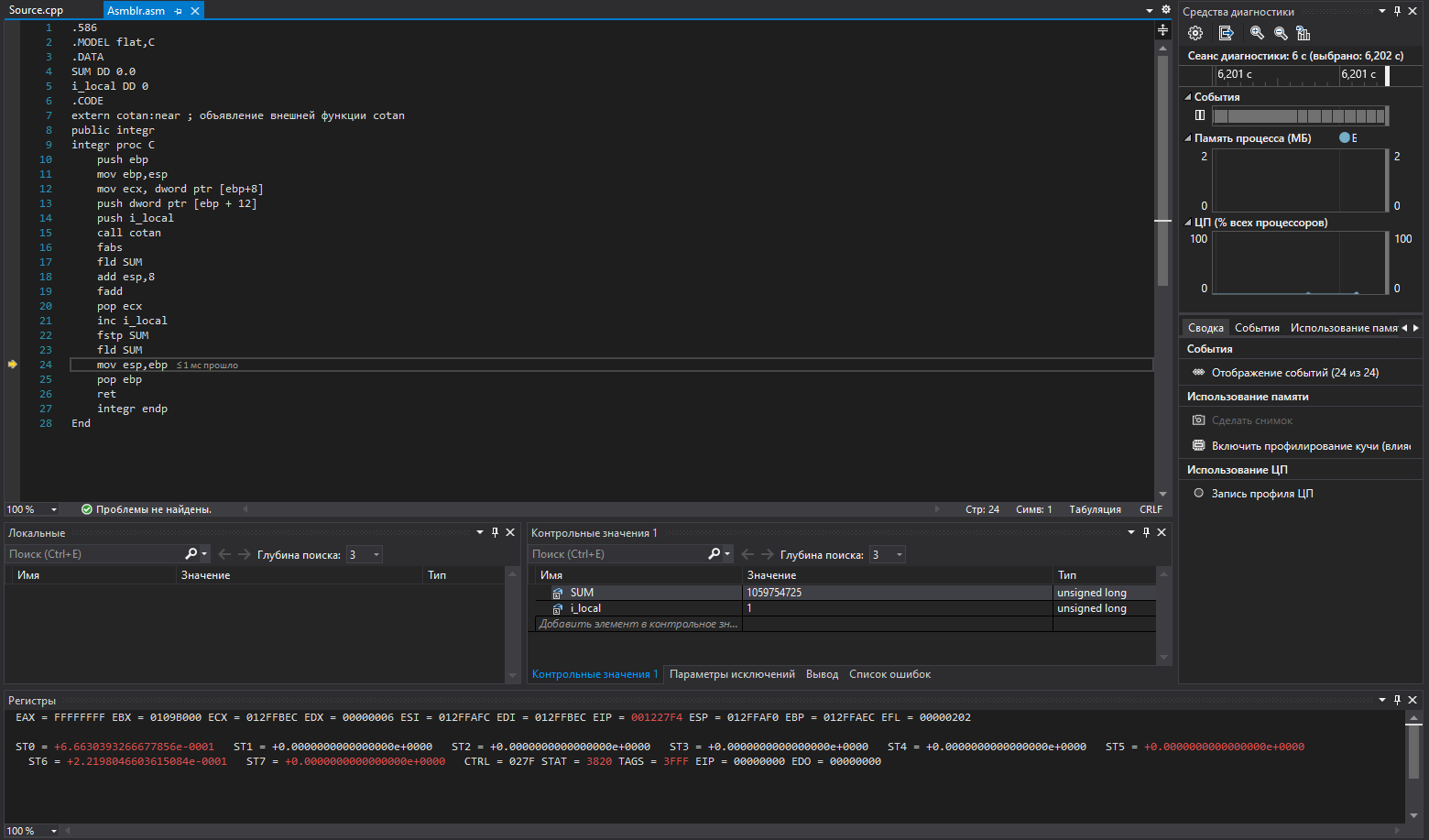


Рисунок 21 – Восстановление указателя стека перед выходом из функции

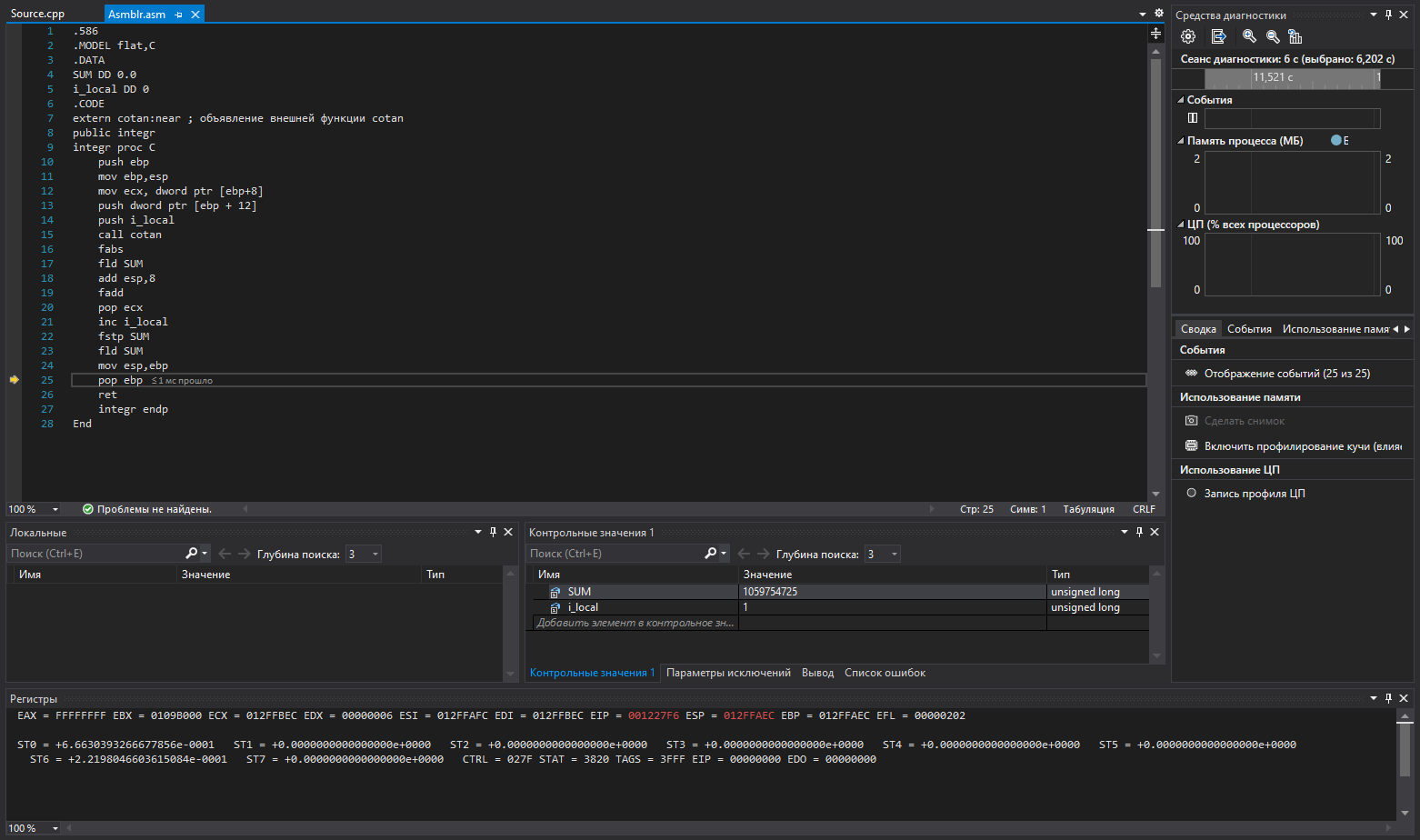


Рисунок 22 – Восстановление базового указателя

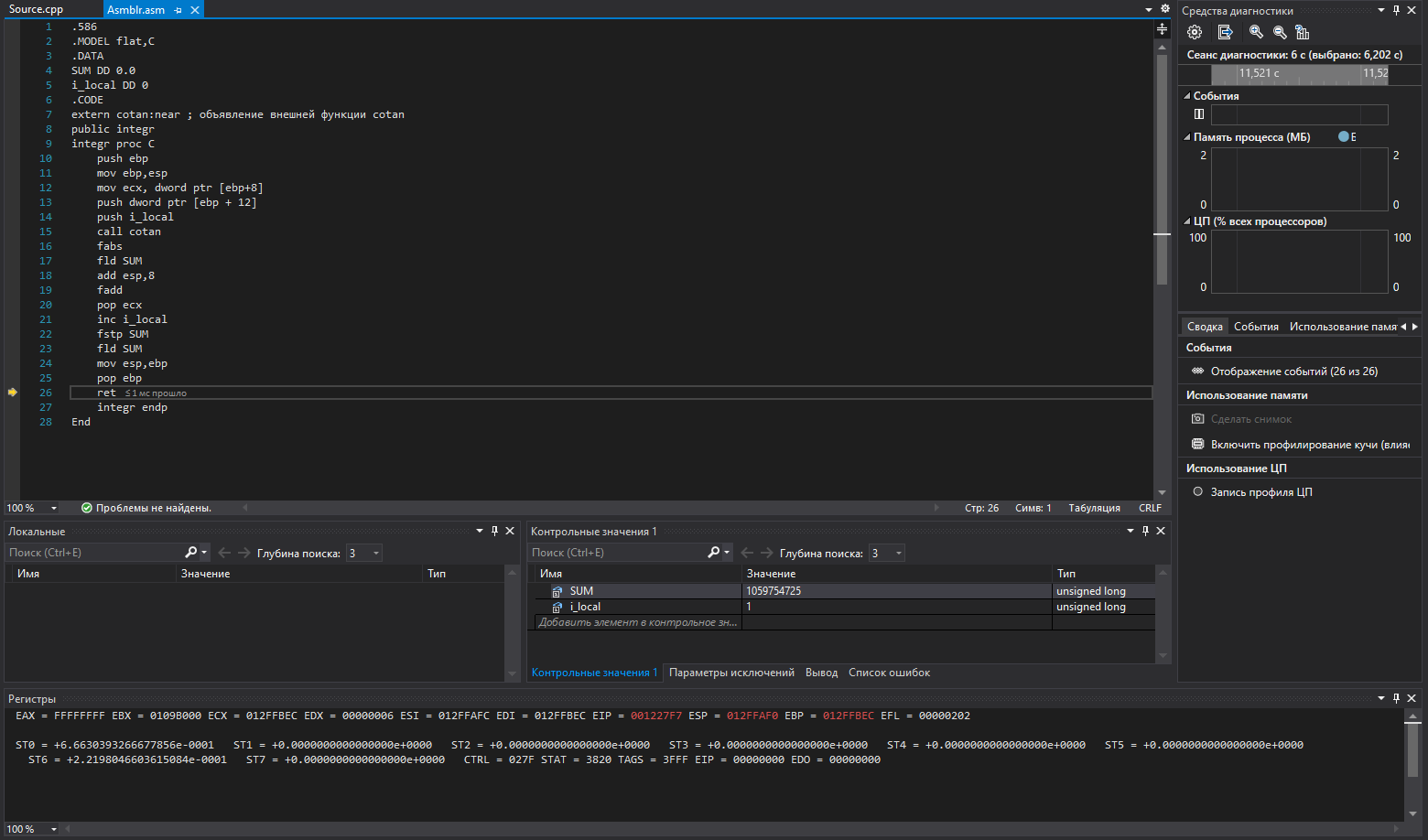


Рисунок 23 – Возврат из функции

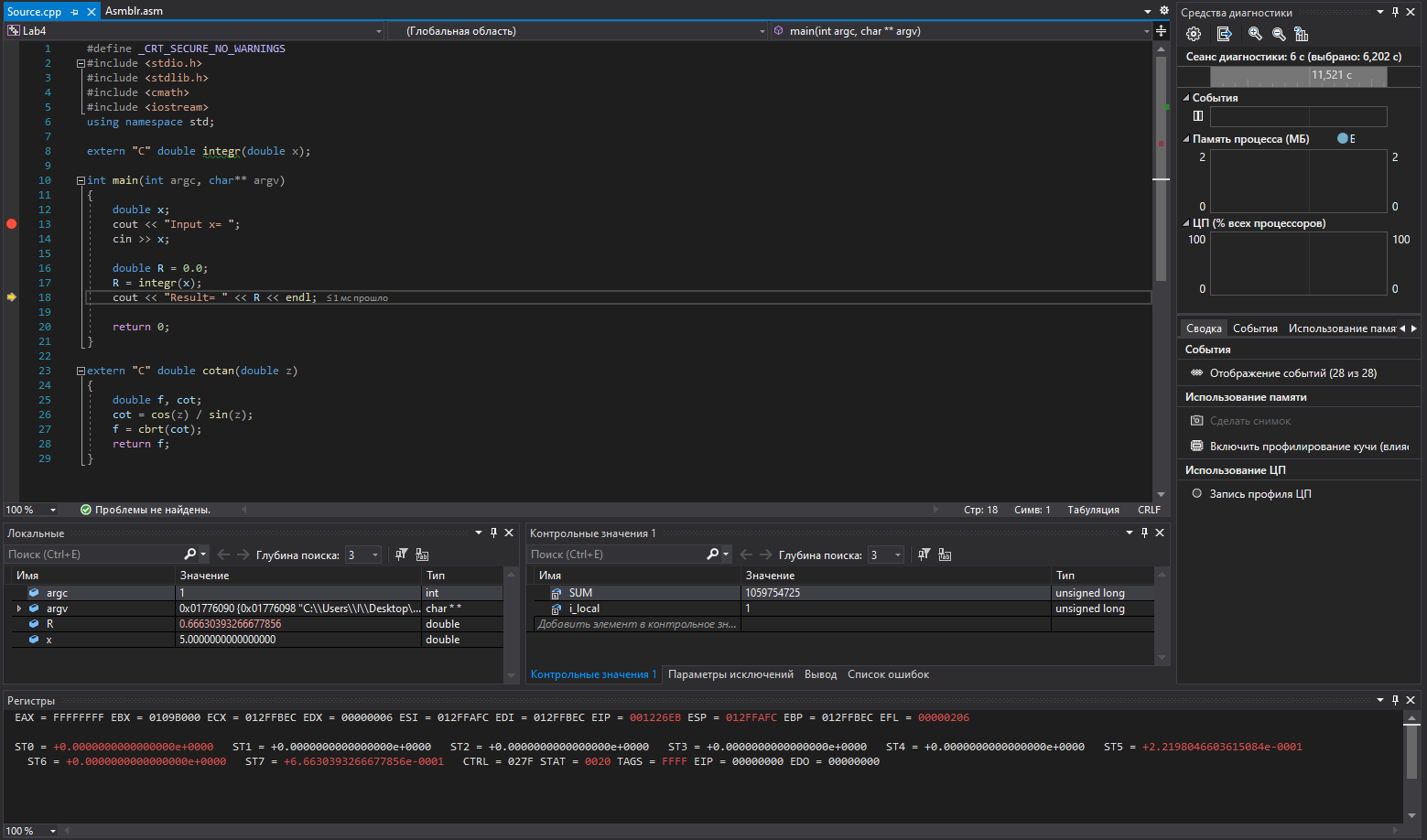


Рисунок 24 – Вывод результата на экран

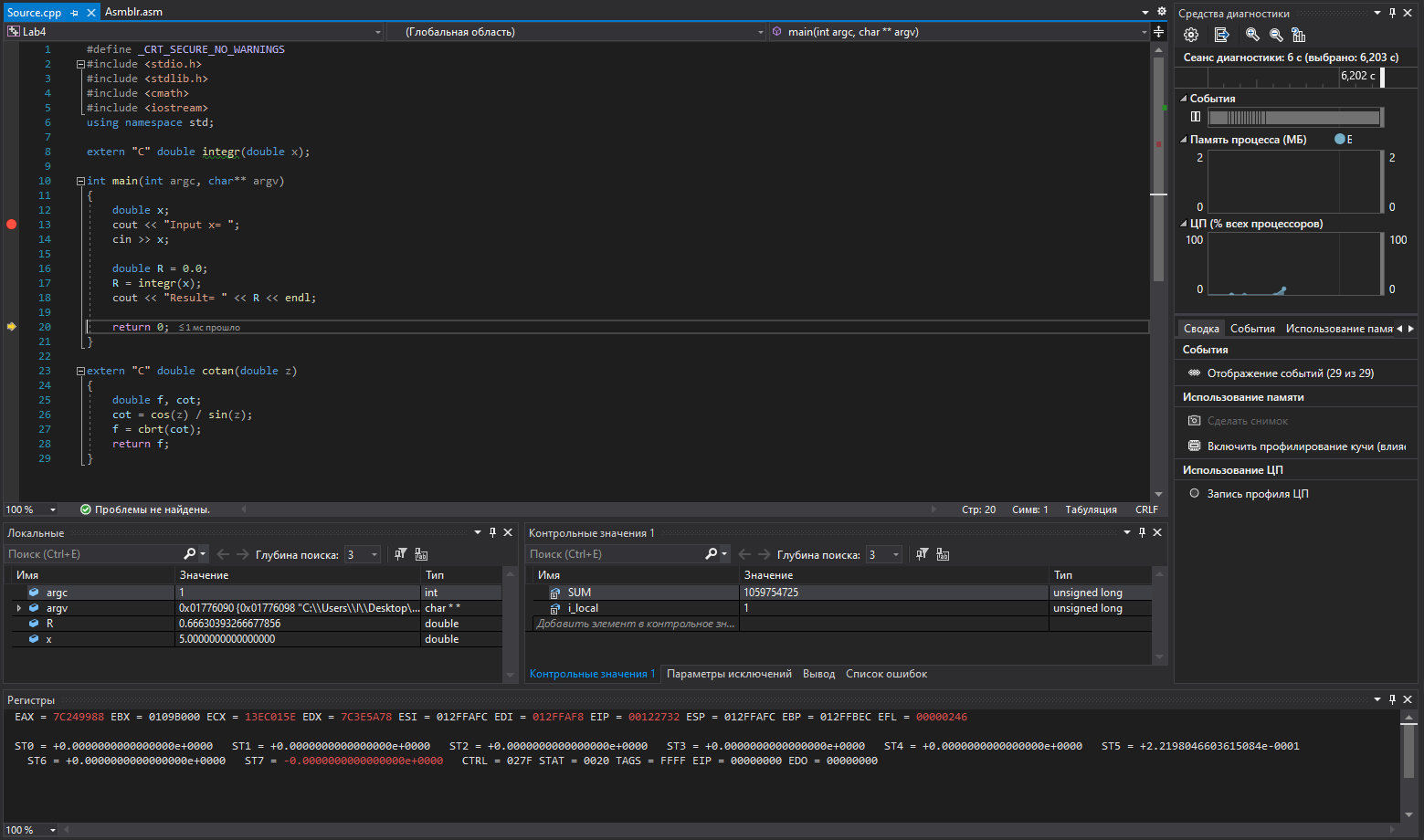


Рисунок 25 – Завершение программы



Рисунок 26 – Конечный вывод в терминал (y = 0,666304)

**Вывод:** было изучение взаимодействия между кодом на языке C++ и ассемблером, а также понимание того, как низкоуровневый код может быть интегрирован в высокоуровневые приложения.