МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО

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 МП»

Лабораторной работе №3

«Использование математического сопроцесса»

Выполнил: студенты гр. ИНБб – 3301                                              /И.А.Капустин/  
  
Проверил: педагог кафедры РЭС\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/М.А.Земцов/

Киров 2025

**Цель работы:**

Изучение принципов выполнения арифметических команд с помощью математического сопроцессора FPU микропроцессоров с архитектурой x86.

**Задание:**

Создайте новый проект в Visual Studio (См. мет. указания к лабораторной работе №1). Для просмотра содержимого регистров сопроцессора в окне «Регистры» дополнительно установите галочку «С плавающей запятой».   
Напишите программу на ассемблере, реализующую решение задачи в соответствие с вариантом. В вычислениях использовать команды математического сопроцессора.  
  
Вычислить и сохранить в массиве значения функции f(x) (100 элементов) на интервале от А до В, если функция задается как: x 2 , если х>0; х, если -1=<x<=0; 2√(𝑥 + 6) , если х<-1

**Текст программы с комментариями:**

.686

.model flat,stdcall

.stack 100h

.data

A REAL8 -5.0 ; Начало интервала

B REAL8 5.0 ; Конец интервала (не используется, так как шаг фиксирован)

step REAL8 1.0 ; Фиксированный шаг изменения x

x REAL8 ? ; Текущее значение x

minus1 REAL8 -1.0 ; Константа -1.0 для сравнения

zero REAL8 0.0 ; Константа 0.0 для сравнения

six REAL8 6.0 ; Константа 6.0 для вычислений

two REAL8 2.0 ; Константа 2.0 для вычислений

array REAL8 10 dup(?) ; Массив из 10 элементов для хранения значений функции

tmp REAL8 ? ; Временная переменная для хранения x из FPU в памяти

tmp\_int dd ? ; Временная переменная для целочисленного значения x

.code

ExitProcess PROTO STDCALL :DWORD

Start:

finit ; Инициализация математического сопроцессора

fld A ; Загружаем начало интервала A в стек FPU

fstp x ; Сохраняем A в переменную x (текущее значение)

mov ecx, 10 ; Устанавливаем счетчик цикла на 10 элементов

lea edi, array ; Загружаем адрес начала массива в edi

LoopStart:

fld x ; Загружаем текущее x в стек FPU

fstp tmp ; Сохраняем его во временную переменную tmp в памяти

fld tmp ; Загружаем tmp обратно в стек FPU

fistp dword ptr tmp\_int ; Преобразуем значение x в целое (округление вниз) и сохраняем в tmp\_int

mov eax, tmp\_int ; Загружаем целочисленное значение x в eax для сравнения

cmp eax, 0

jg X\_gt\_0 ; Если x > 0, переходим к вычислению x^2

cmp eax, -1

jl X\_lt\_minus1 ; Если x < -1, переходим к вычислению 2\*sqrt(x + 6)

; Иначе -1 <= x <= 0, f(x) = x

fld tmp ; Загружаем x для сохранения в массив

jmp StoreValue ; Переход к сохранению значения

X\_gt\_0:

fld tmp ; Загружаем x в стек

fld tmp ; Загружаем x в стек (повторно для умножения)

fmul ; Вычисляем x \* x = x^2

jmp StoreValue ; Переход к сохранению значения

X\_lt\_minus1:

fld tmp ; Загружаем x в стек

fld six ; Загружаем 6 в стек

fadd ; Вычисляем x + 6

fsqrt ; Берем квадратный корень sqrt(x + 6)

fld two ; Загружаем 2 в стек

fmul ; Умножаем на 2, получаем 2\*sqrt(x + 6)

StoreValue:

fstp qword ptr [edi] ; Сохраняем вычисленное значение функции в массив по адресу edi

fld x ; Загружаем текущее значение x

fld step ; Загружаем шаг

fadd ; Прибавляем шаг к x

fstp x ; Сохраняем обновленное значение x

add edi, 8 ; Переходим к следующему элементу массива (8 байт — размер REAL8)

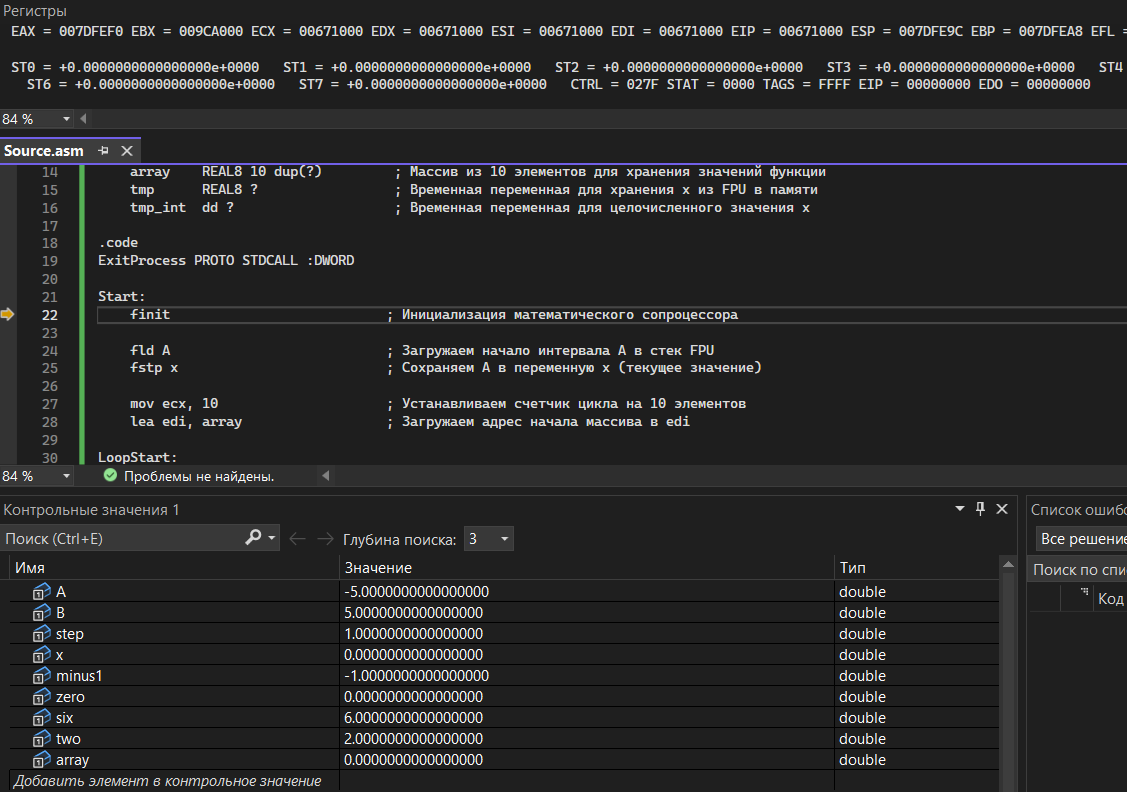
loop LoopStart ; Уменьшаем ecx и, если не 0, повторяем цикл

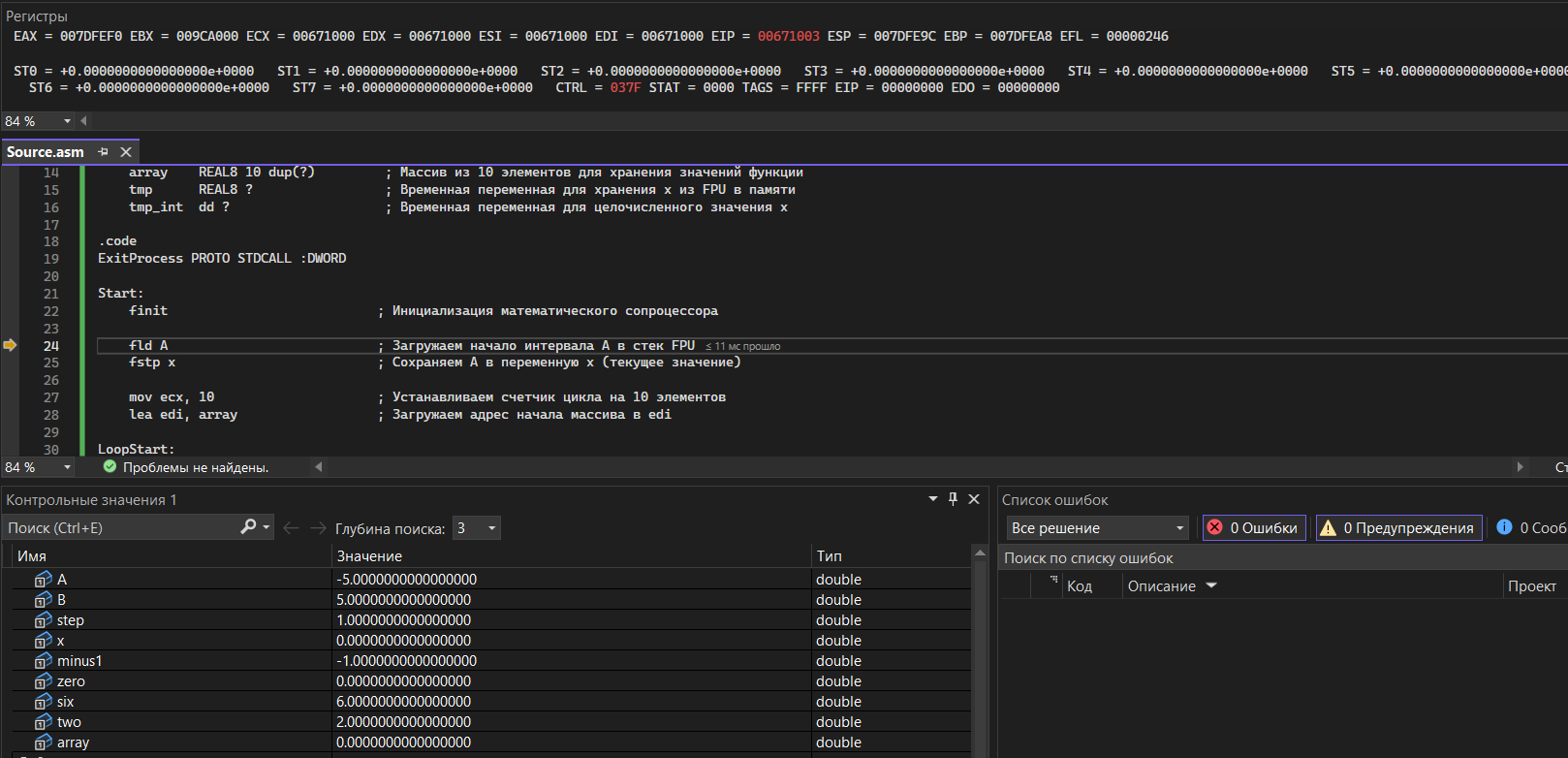
invoke ExitProcess, 0 ; Завершаем программу

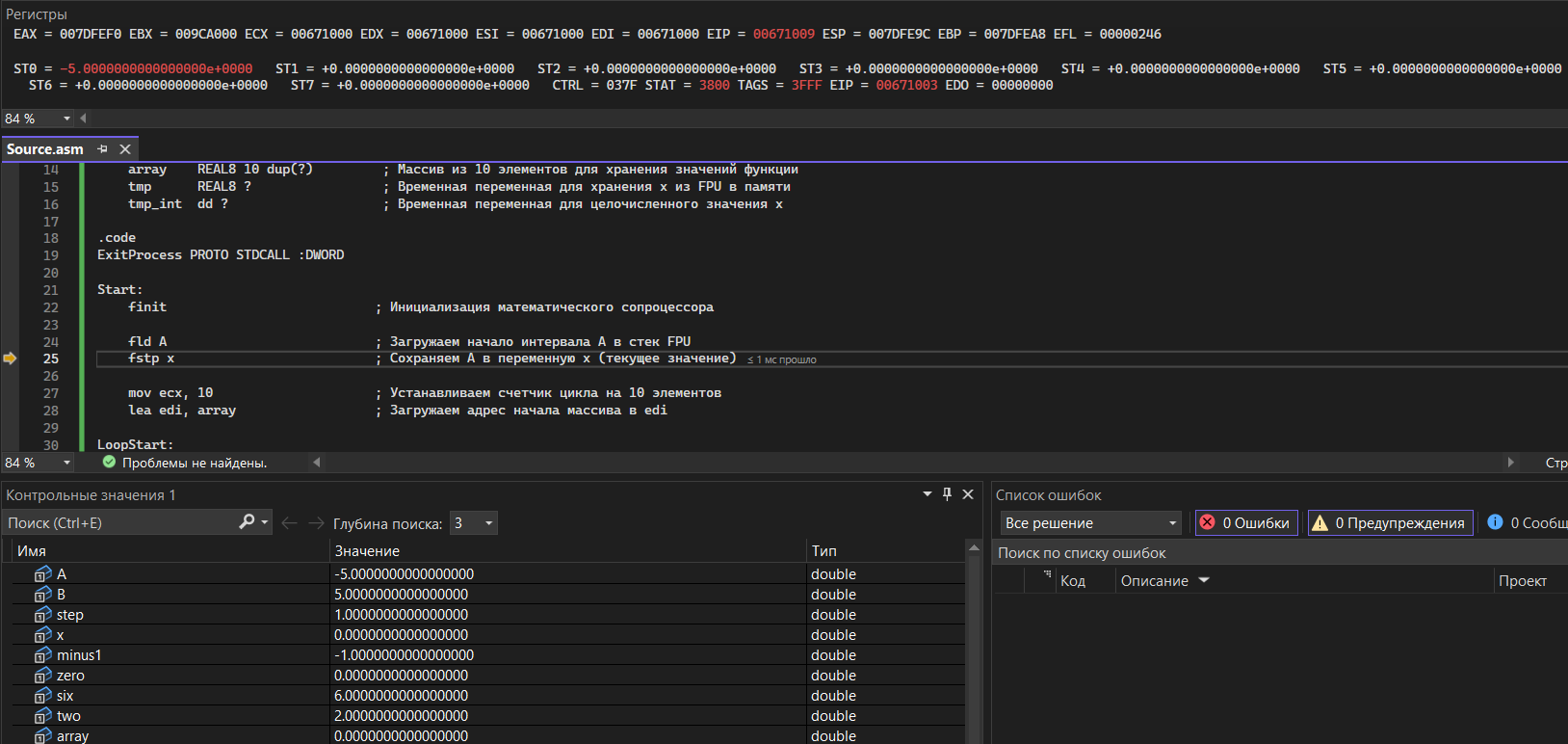
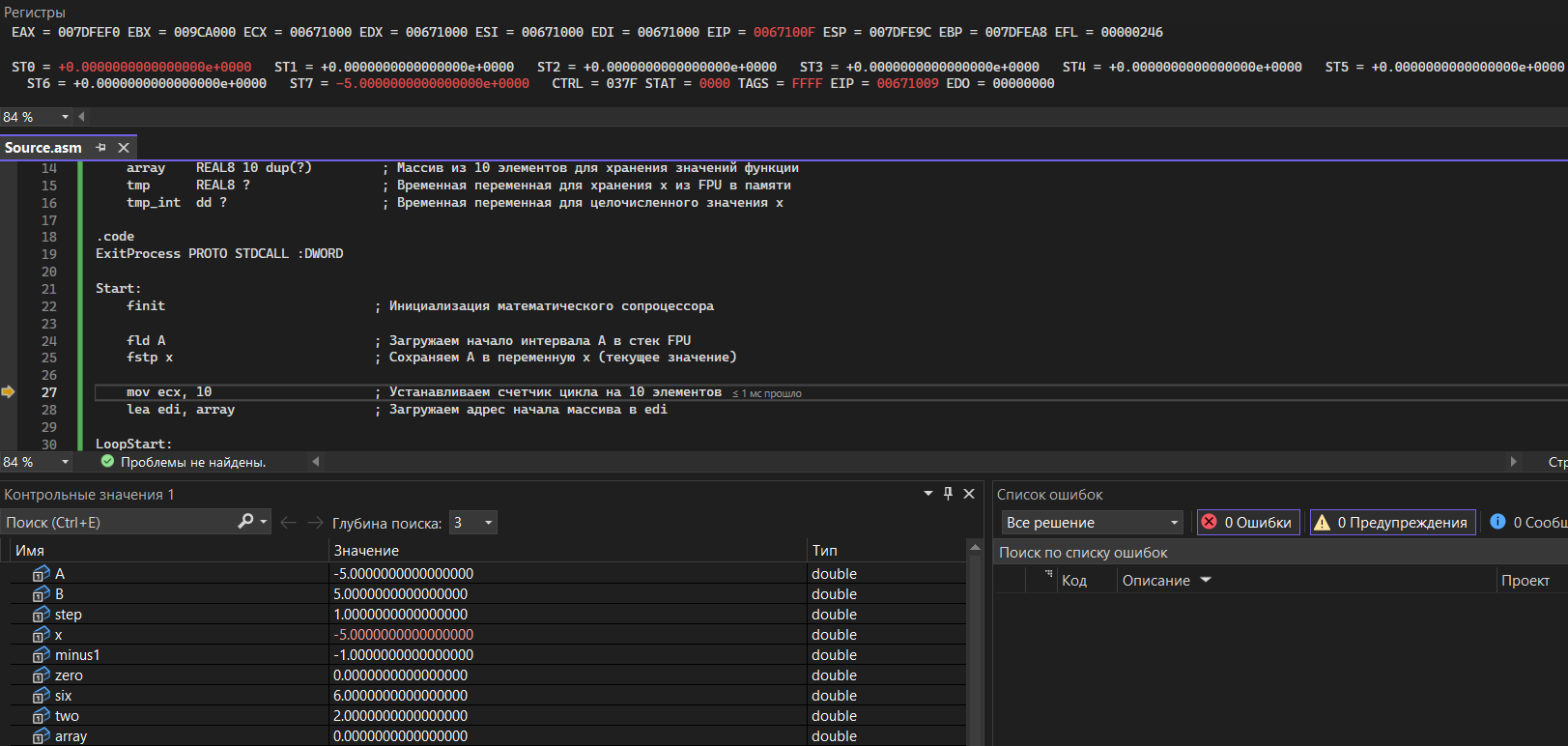
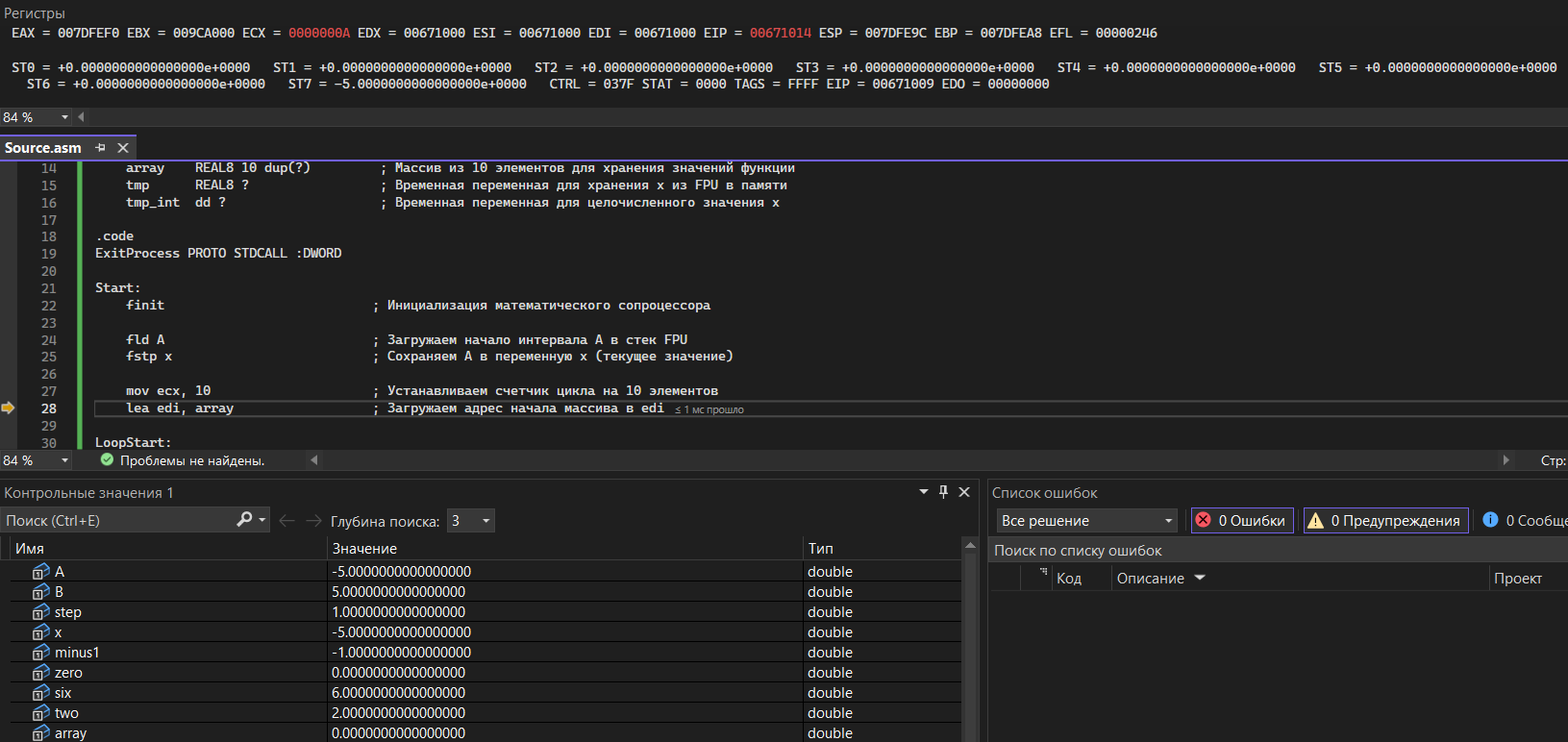
End Start

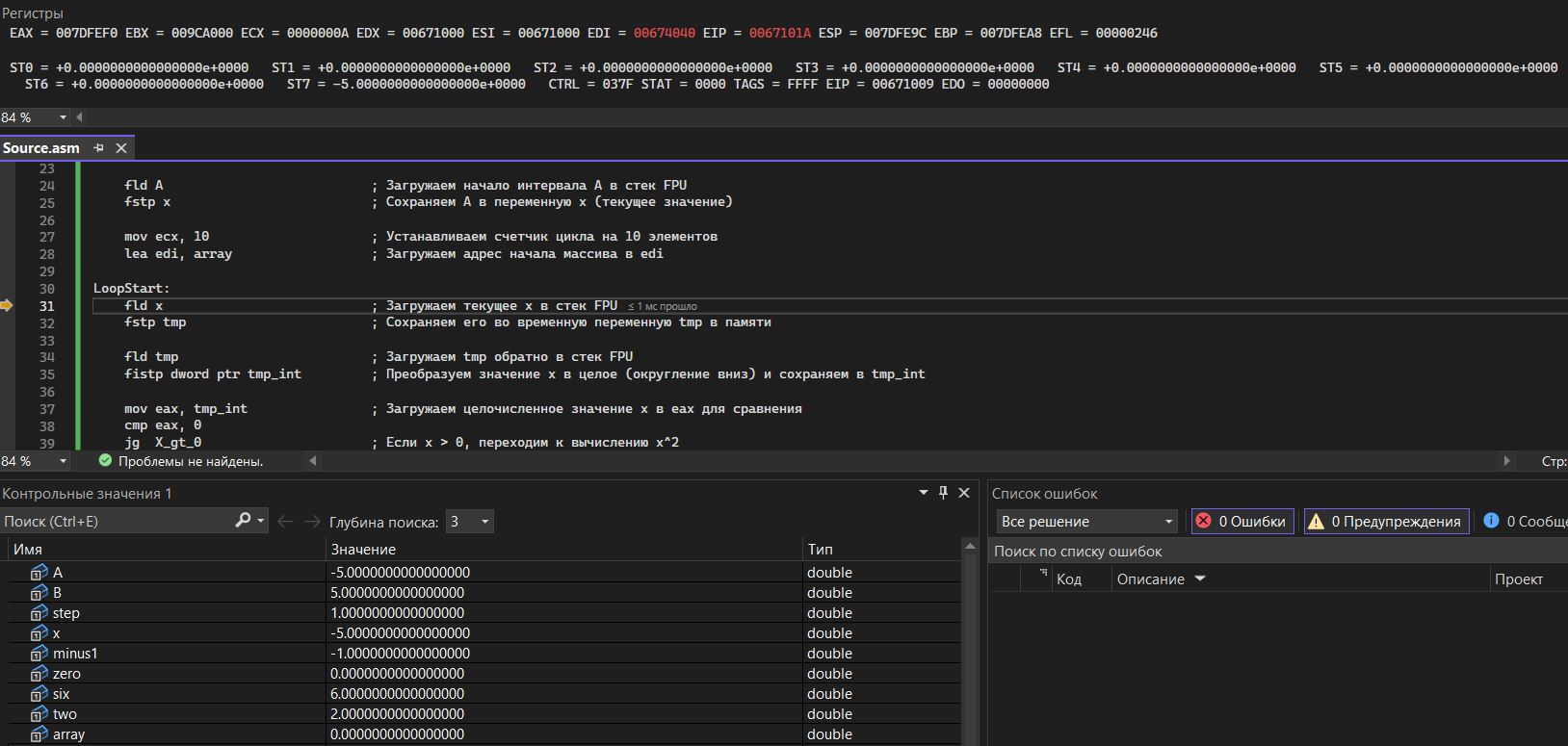
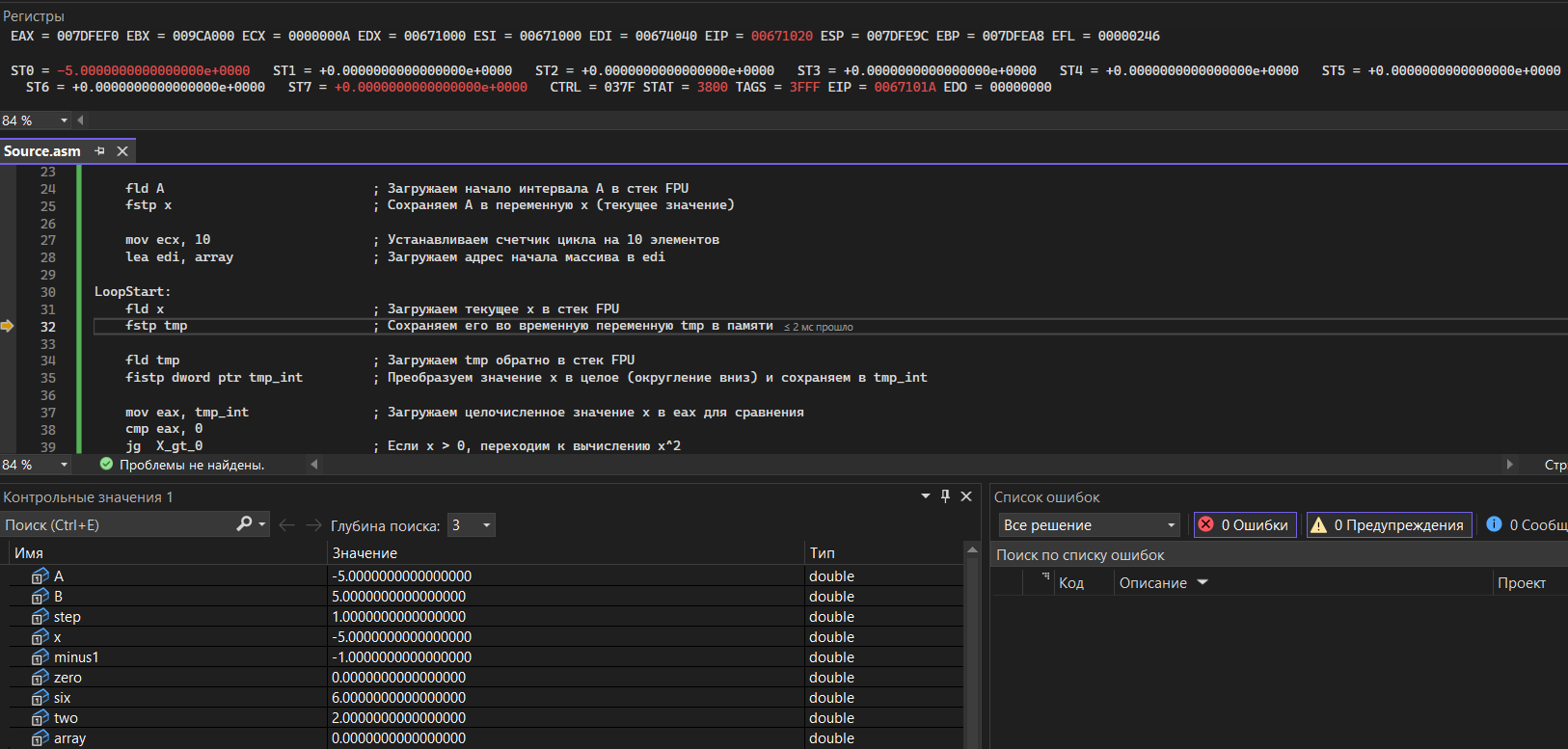
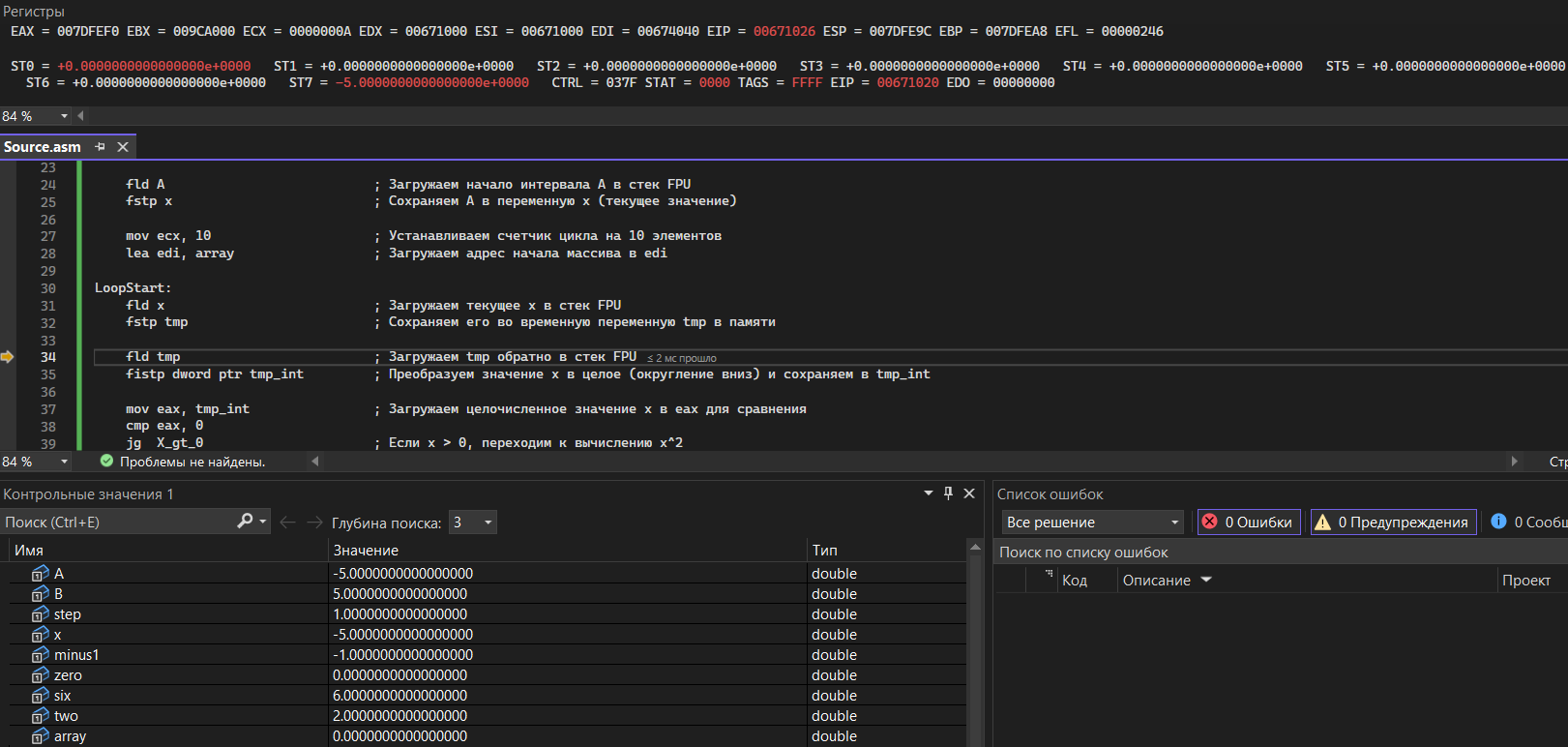
**Верификация программы: результаты расчета заданного выражения, скриншоты, показывающие содержимое регистров и значения переменных после каждого действия программы.**

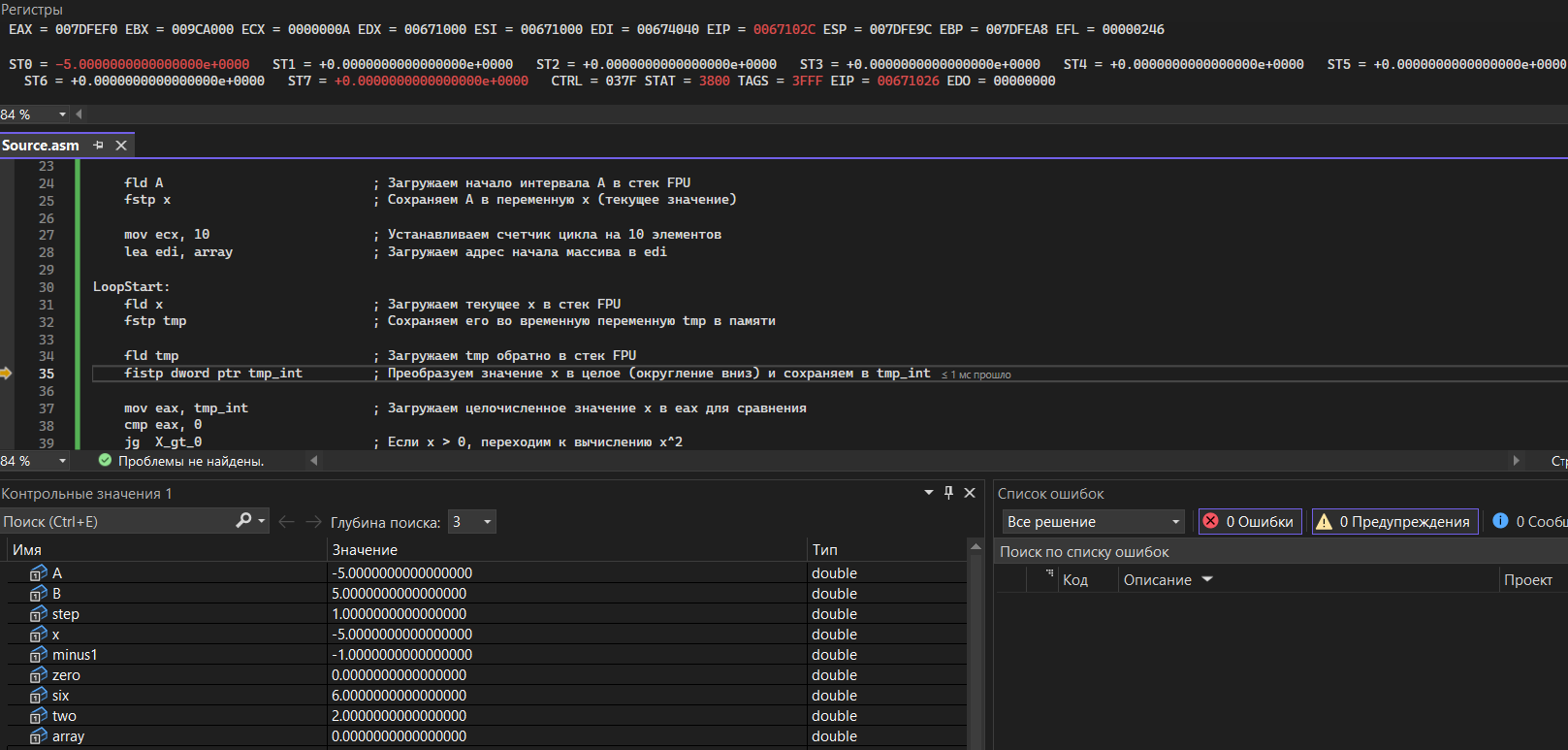
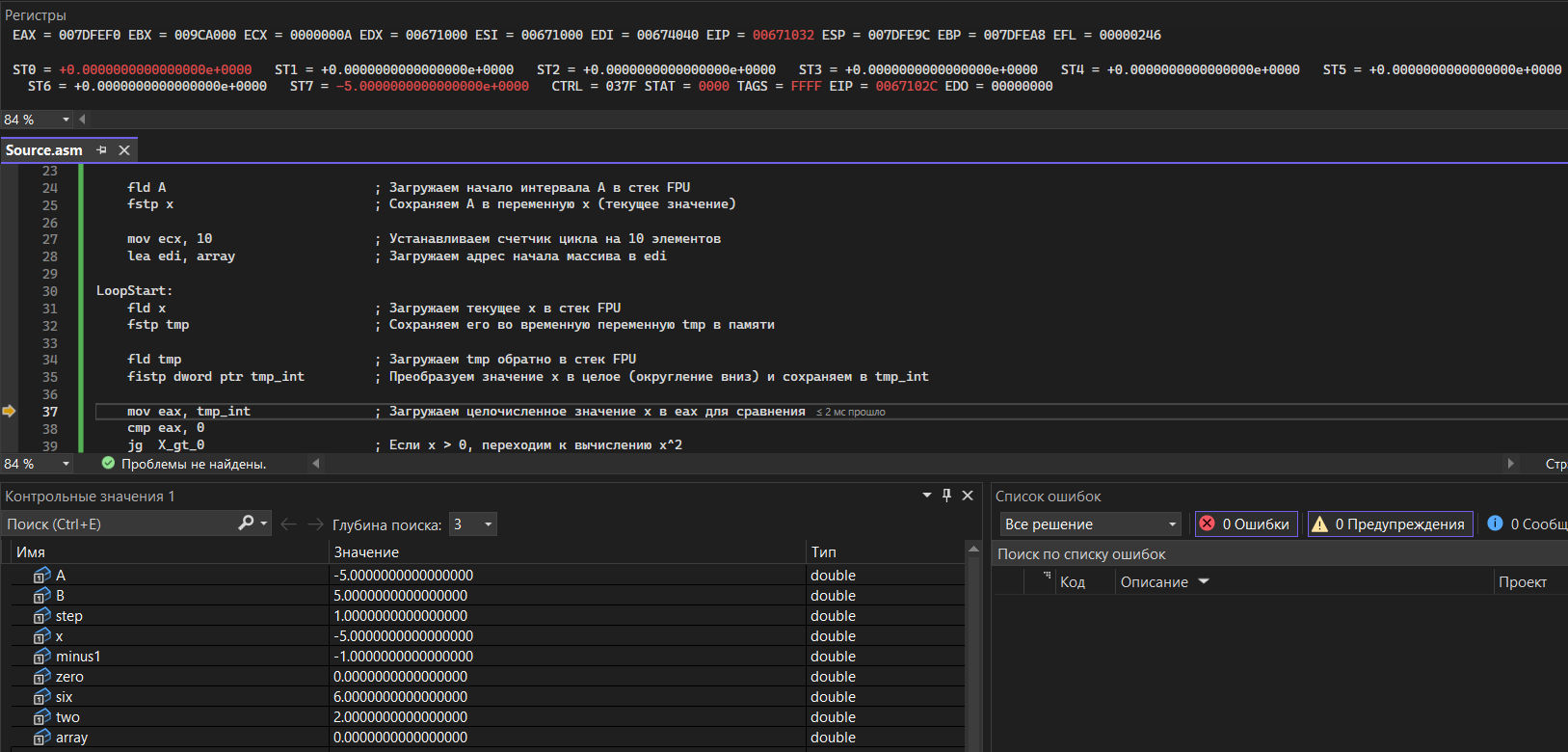
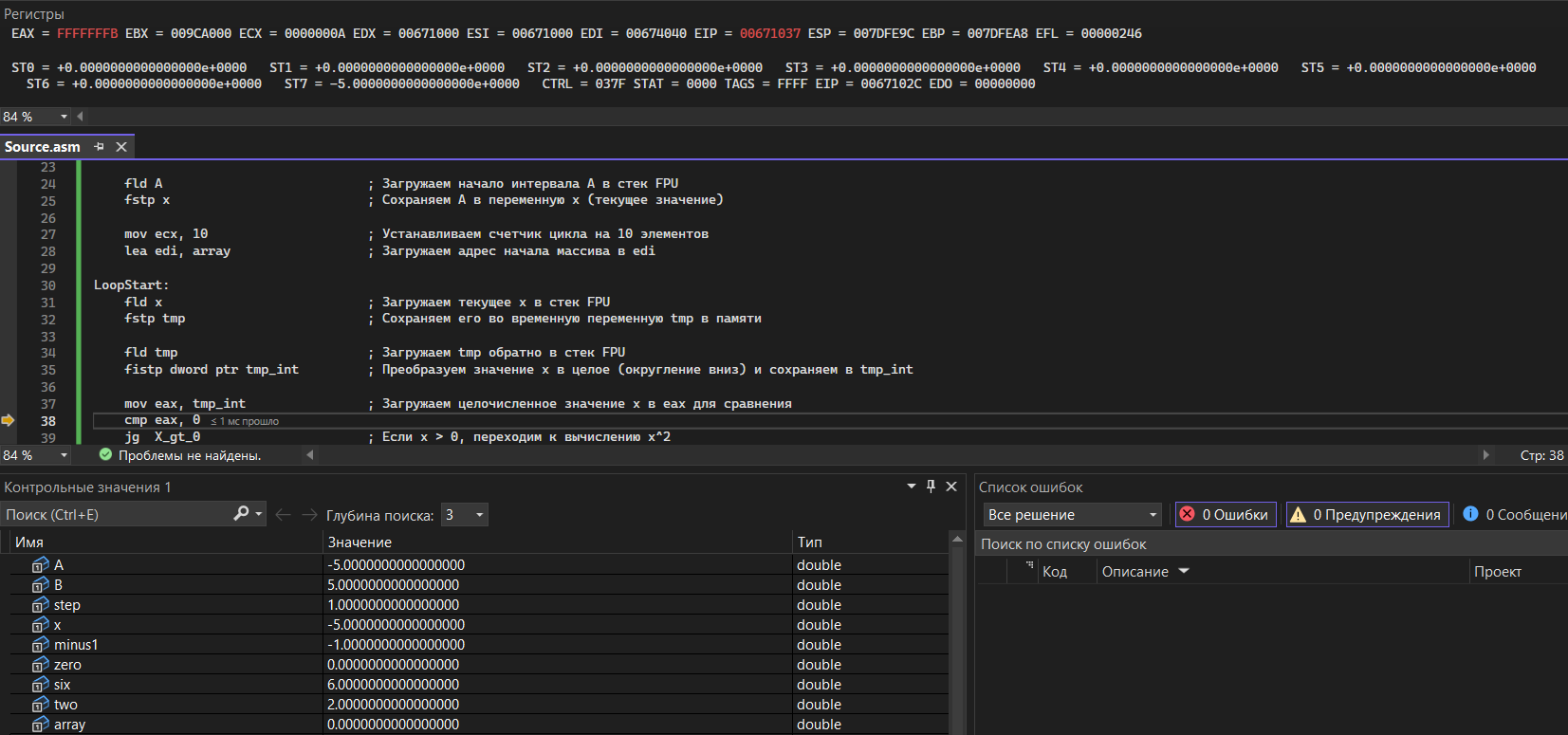
**1) Выполнение программы при условии, что x=-5 и вычисление идет по формуле - 2√(𝑥 + 6)**

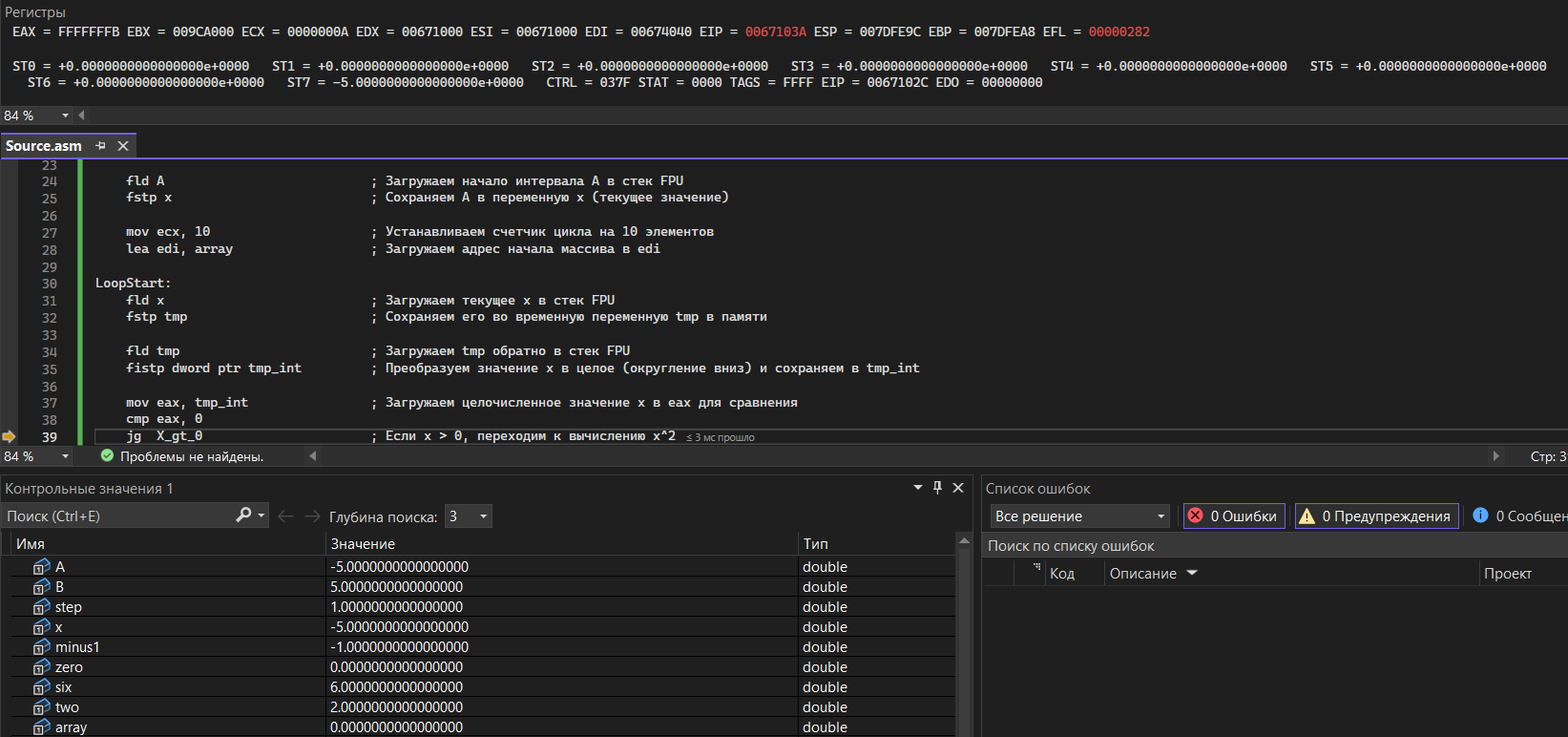
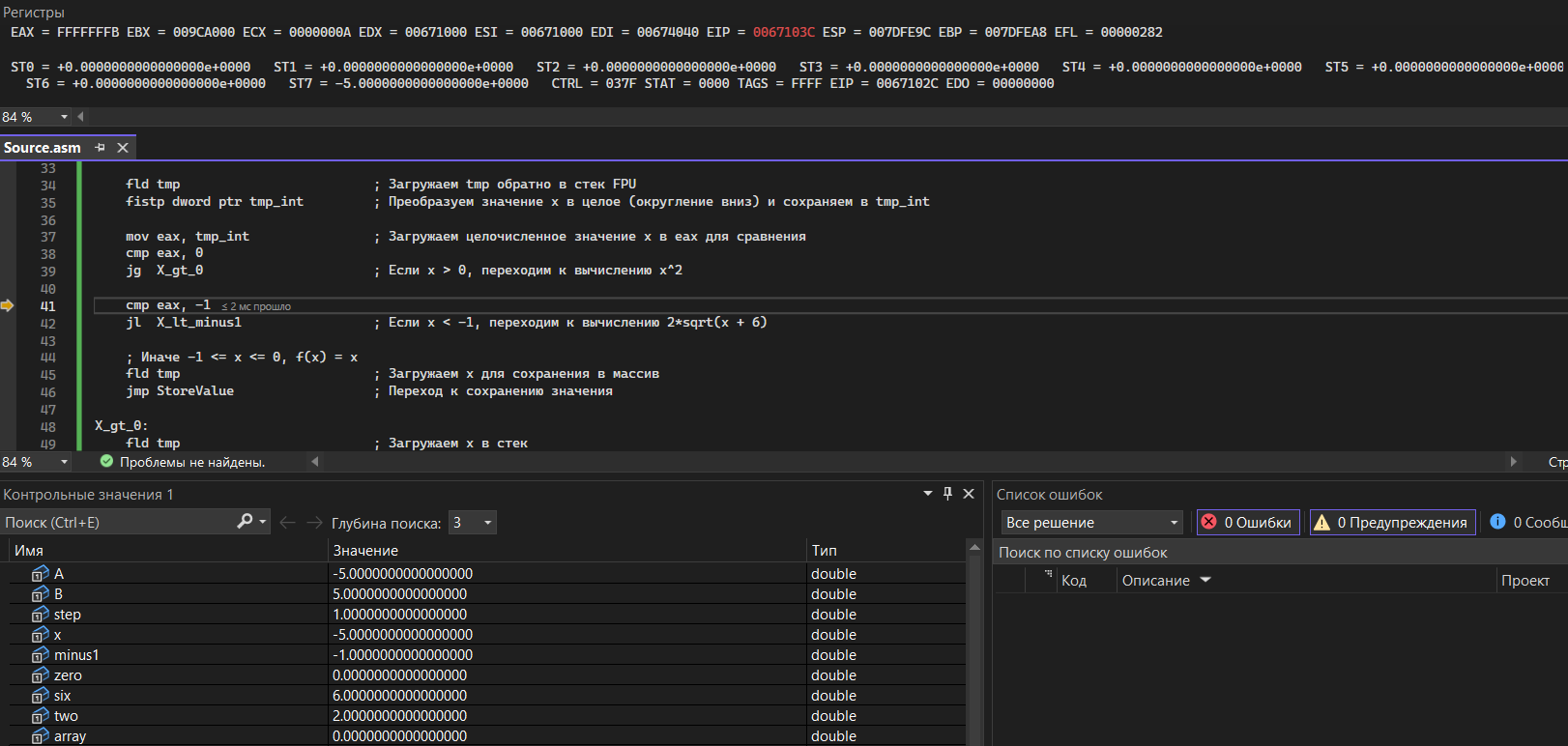
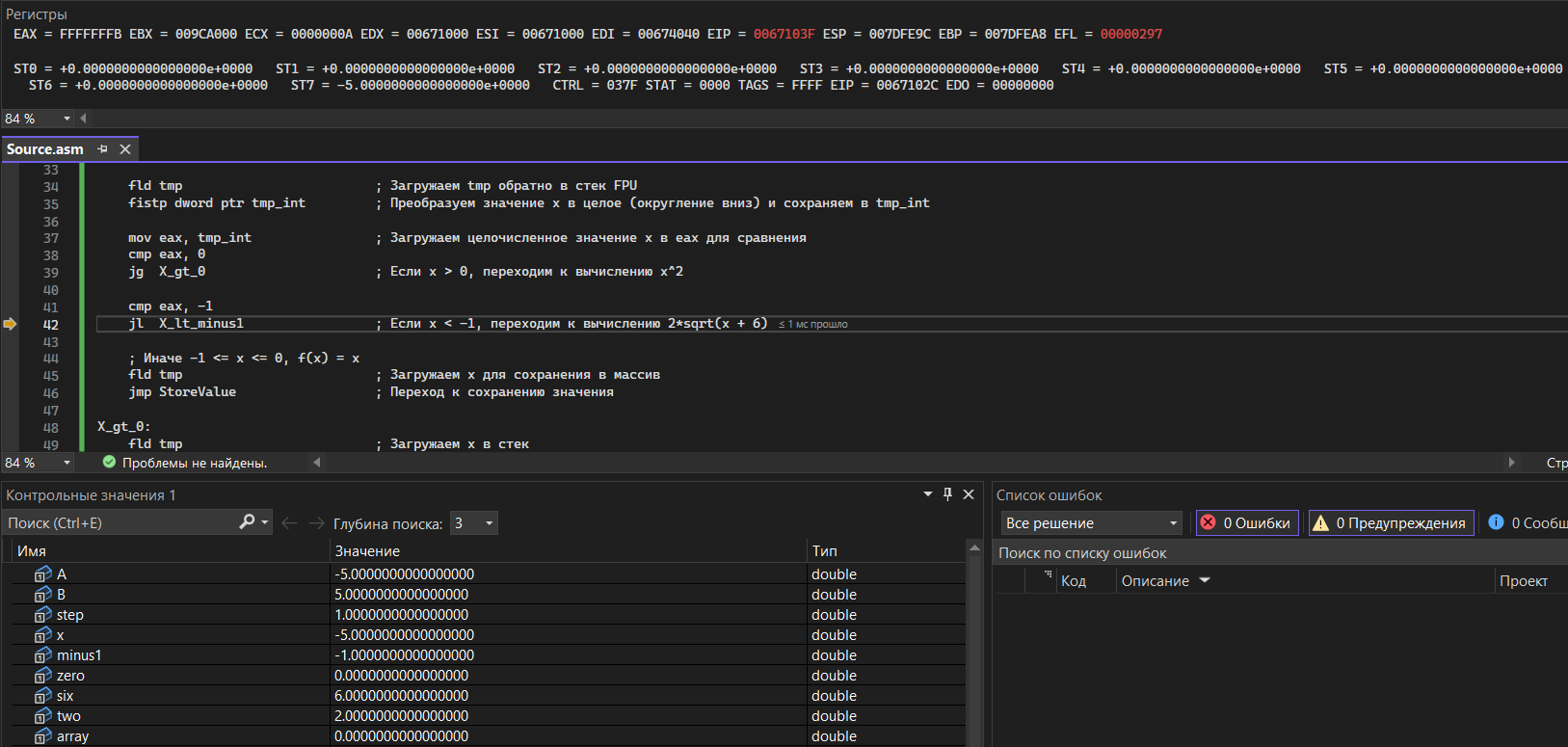
Инициализация математического сопроцессора ****

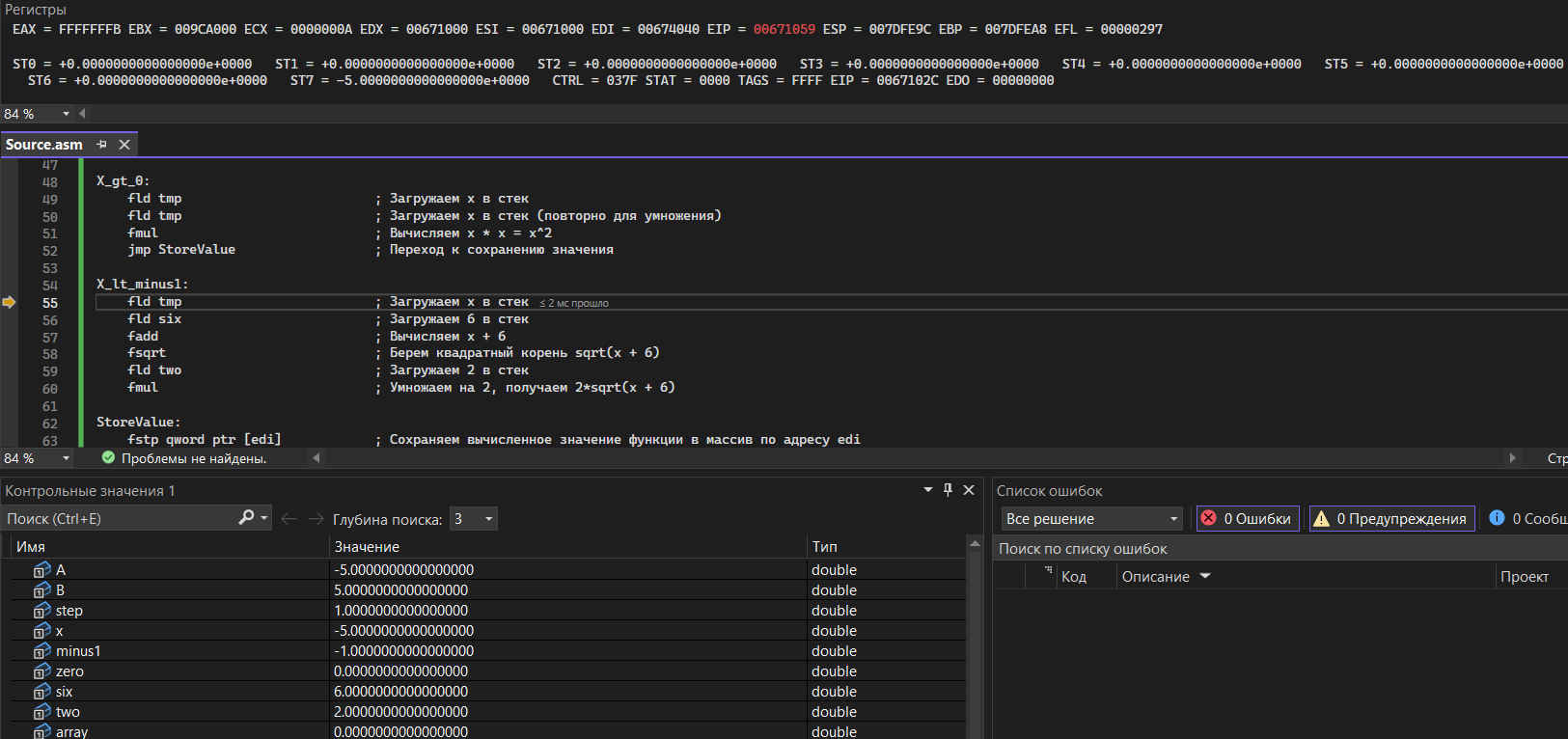
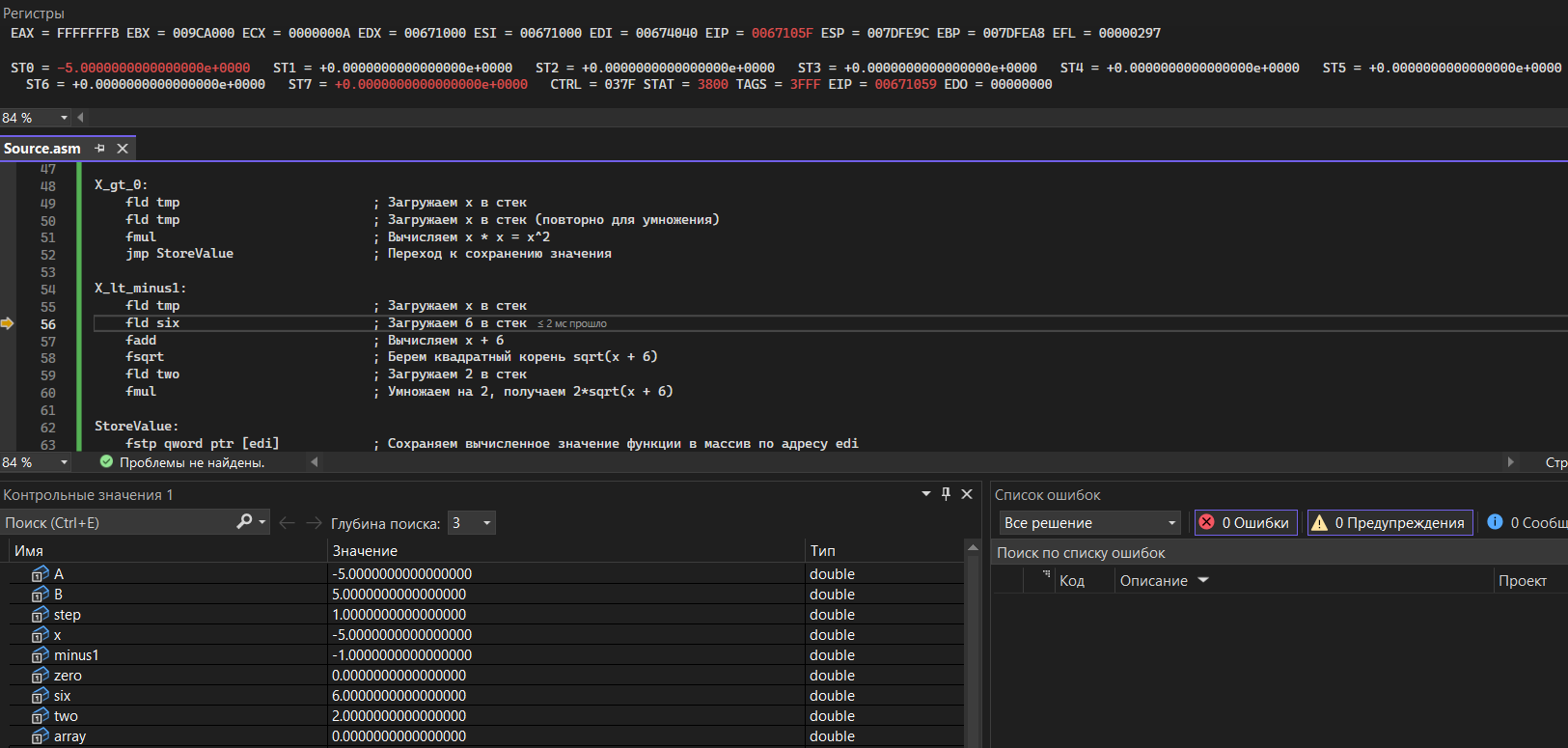
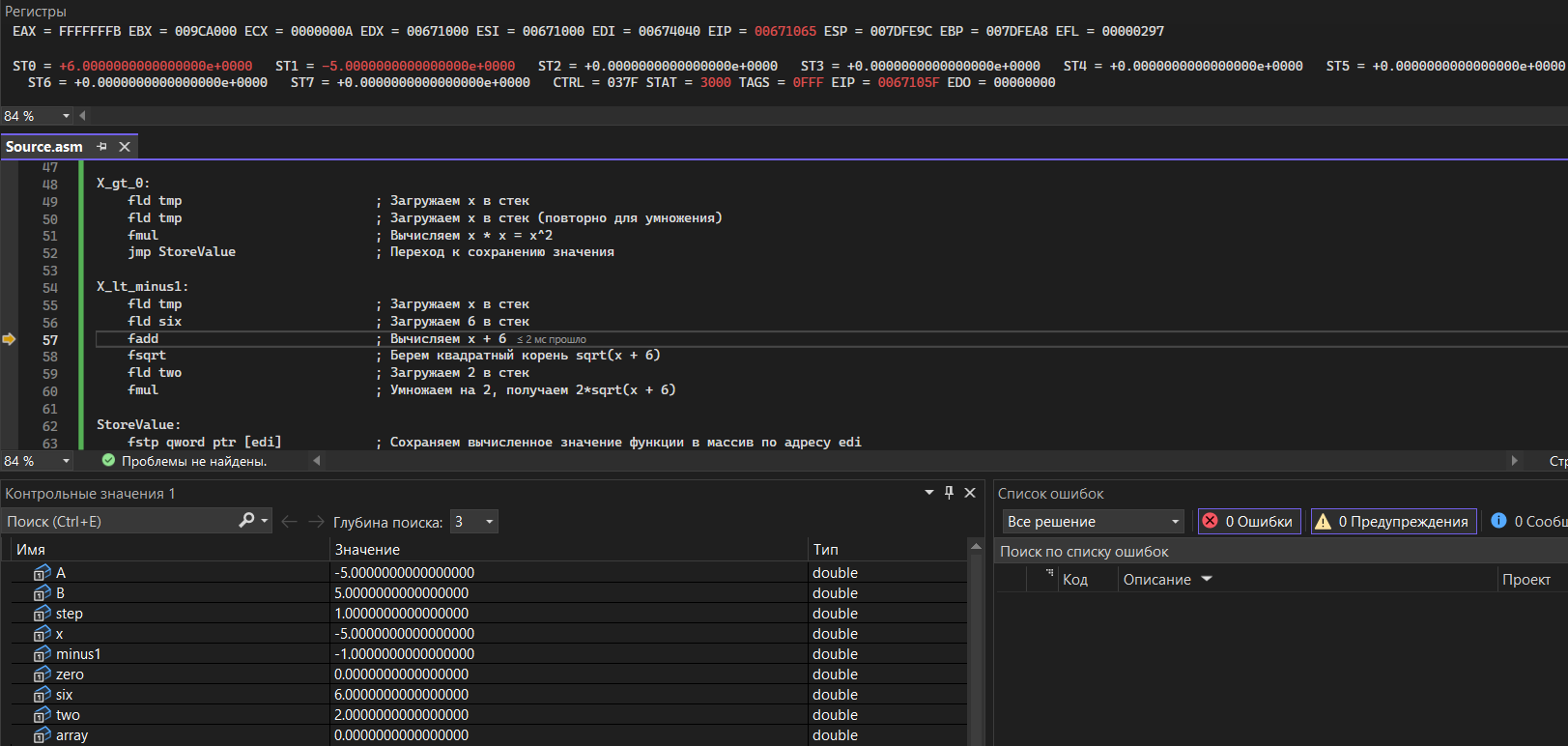
Загружаем начало интервала A в стек FPU ****

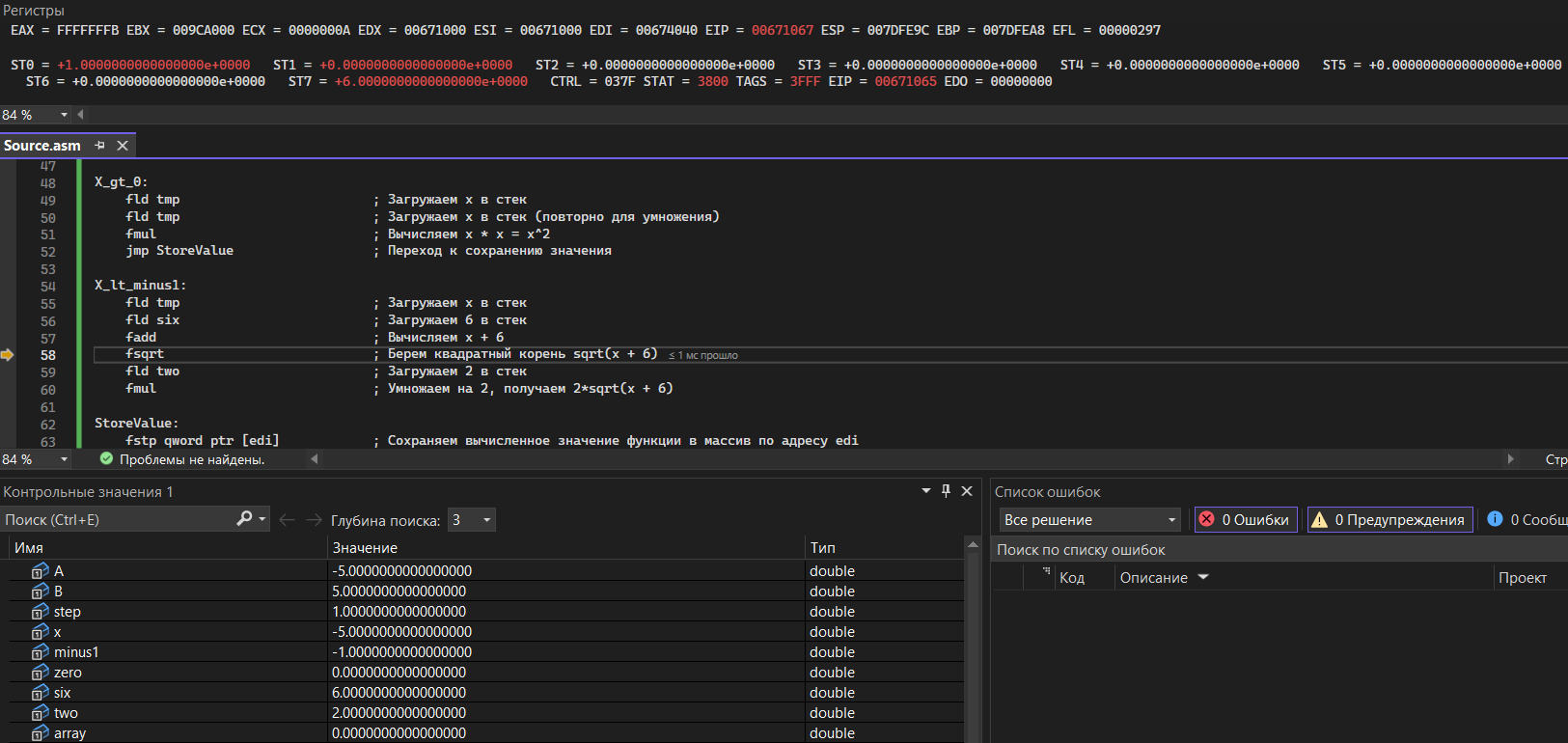
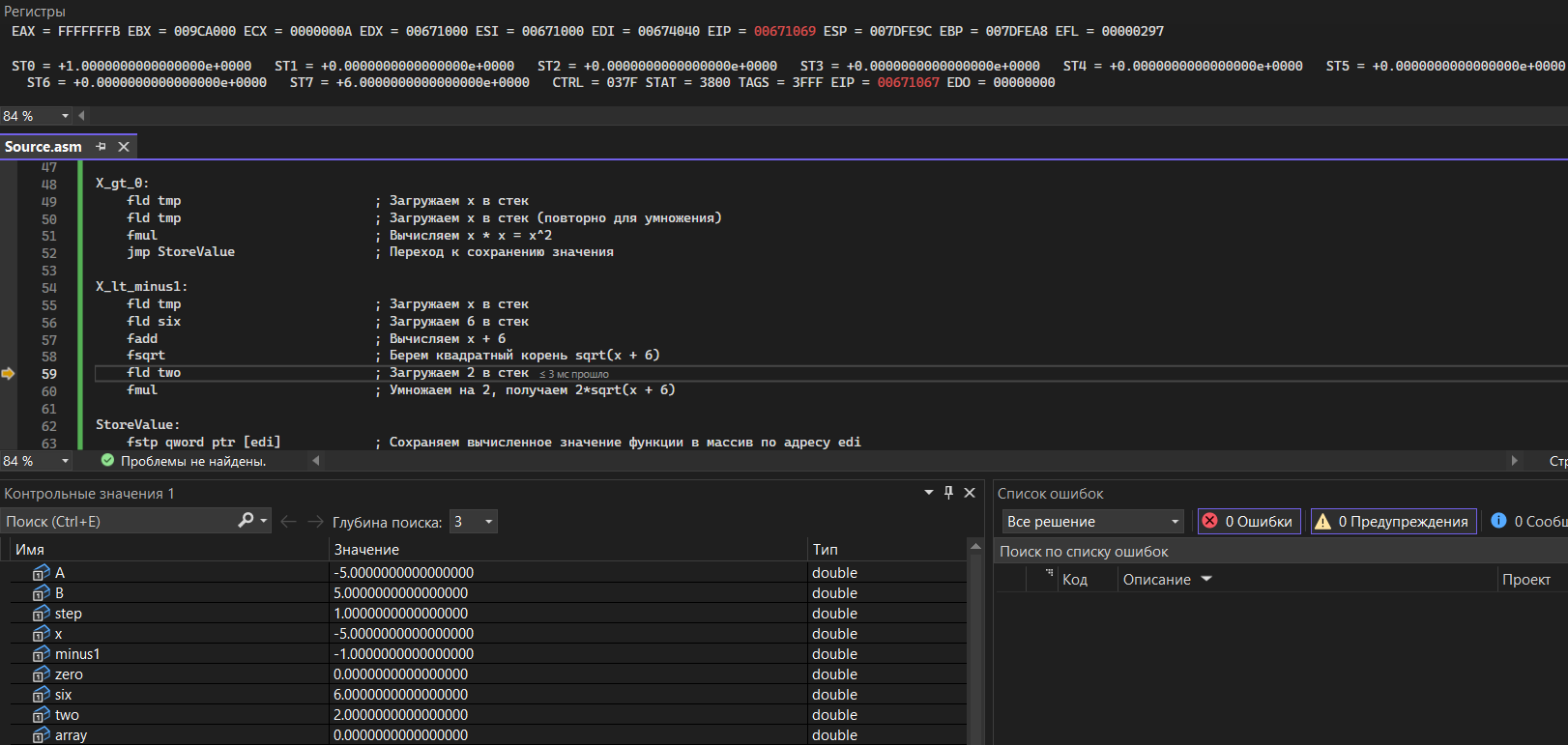
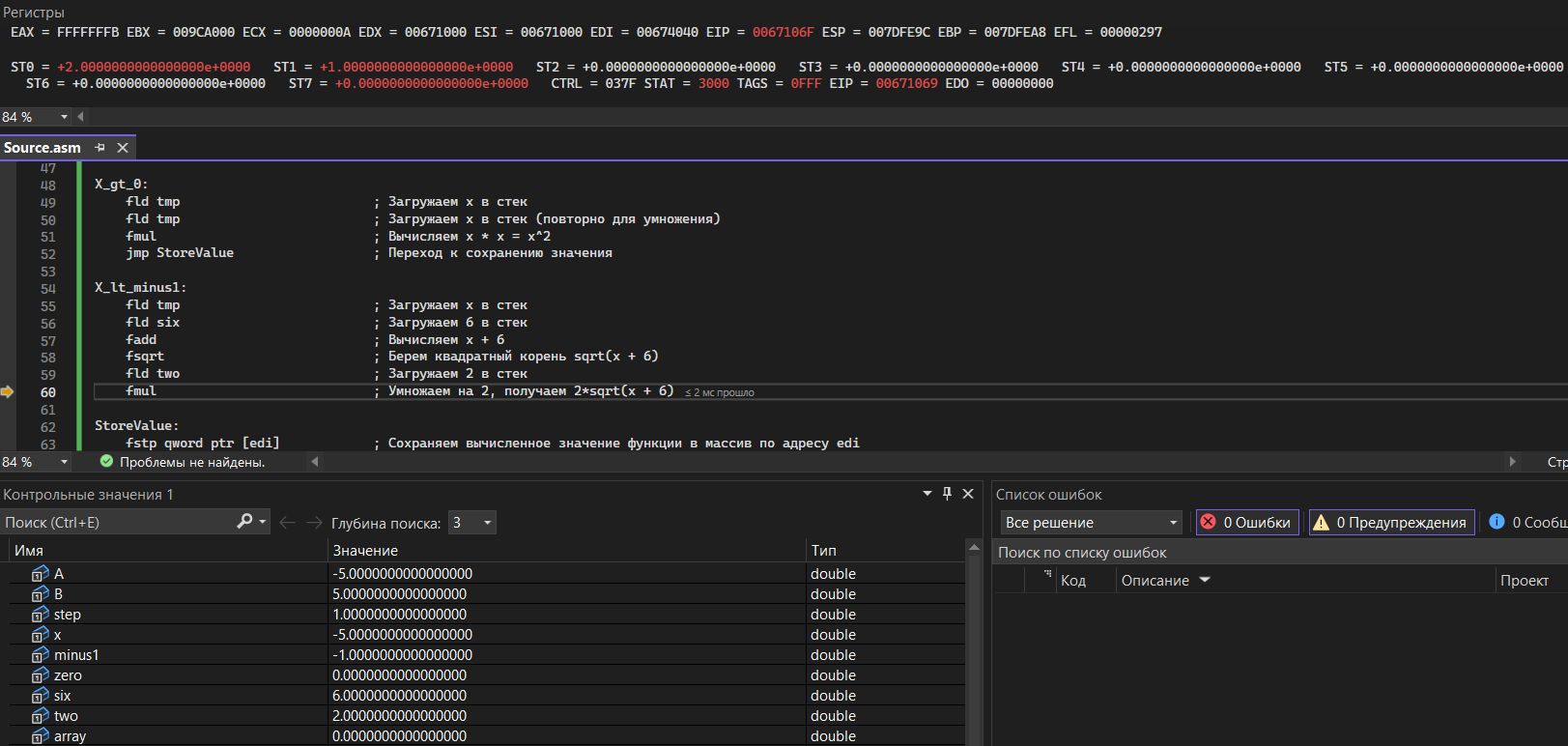
Сохраняем A в переменную x (текущее значение) ****Устанавливаем счетчик цикла на 10 элементов ****Загружаем адрес начала массива в edi ****

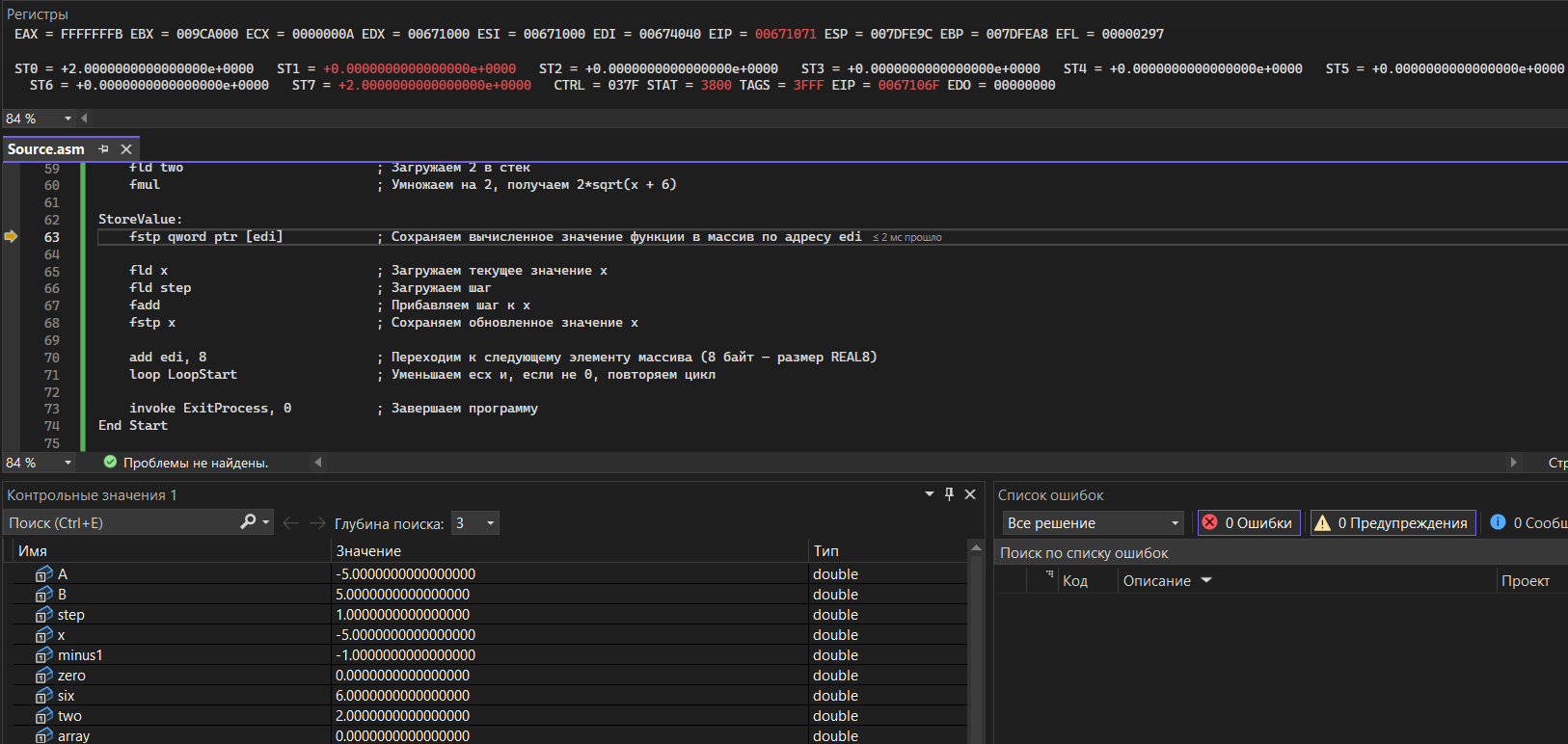
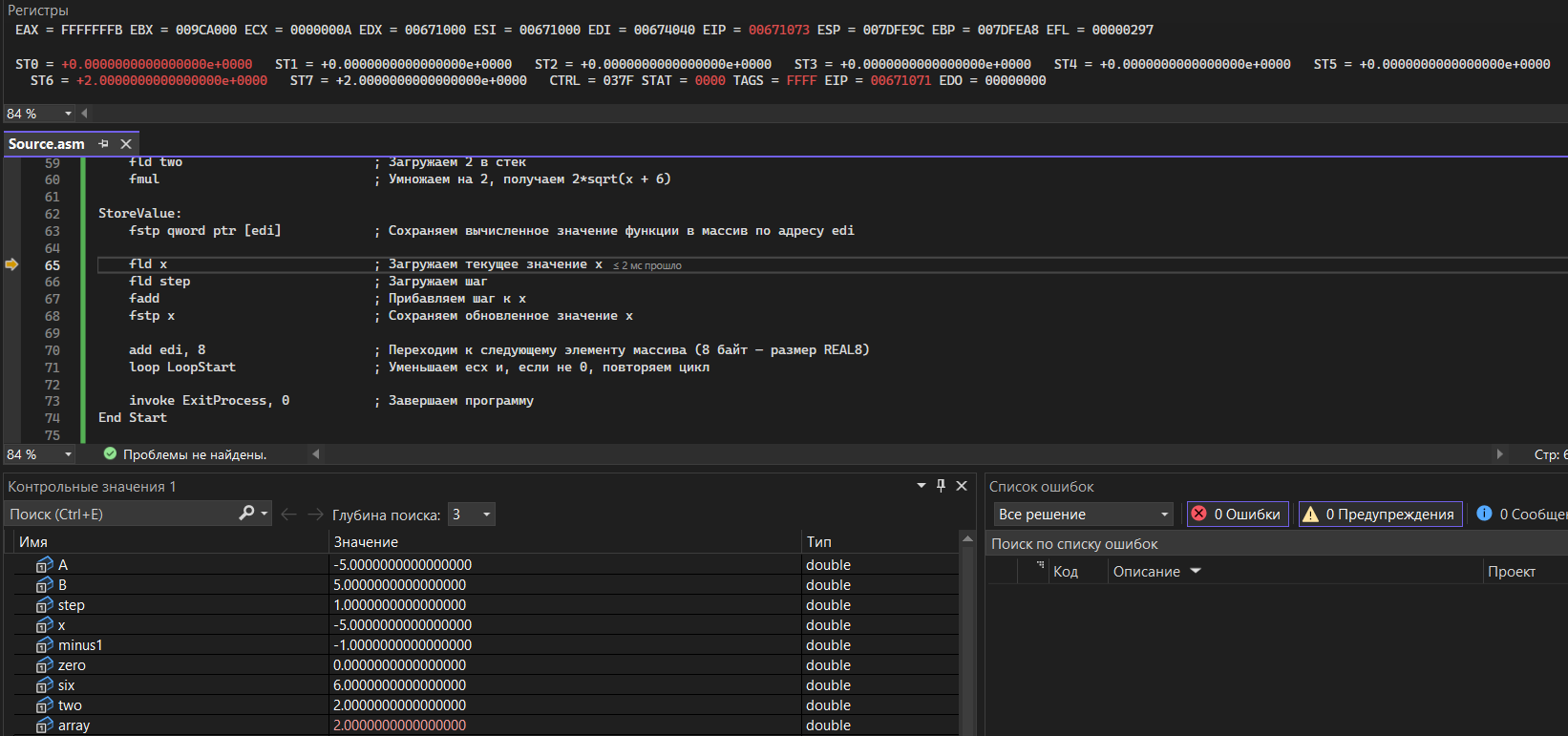
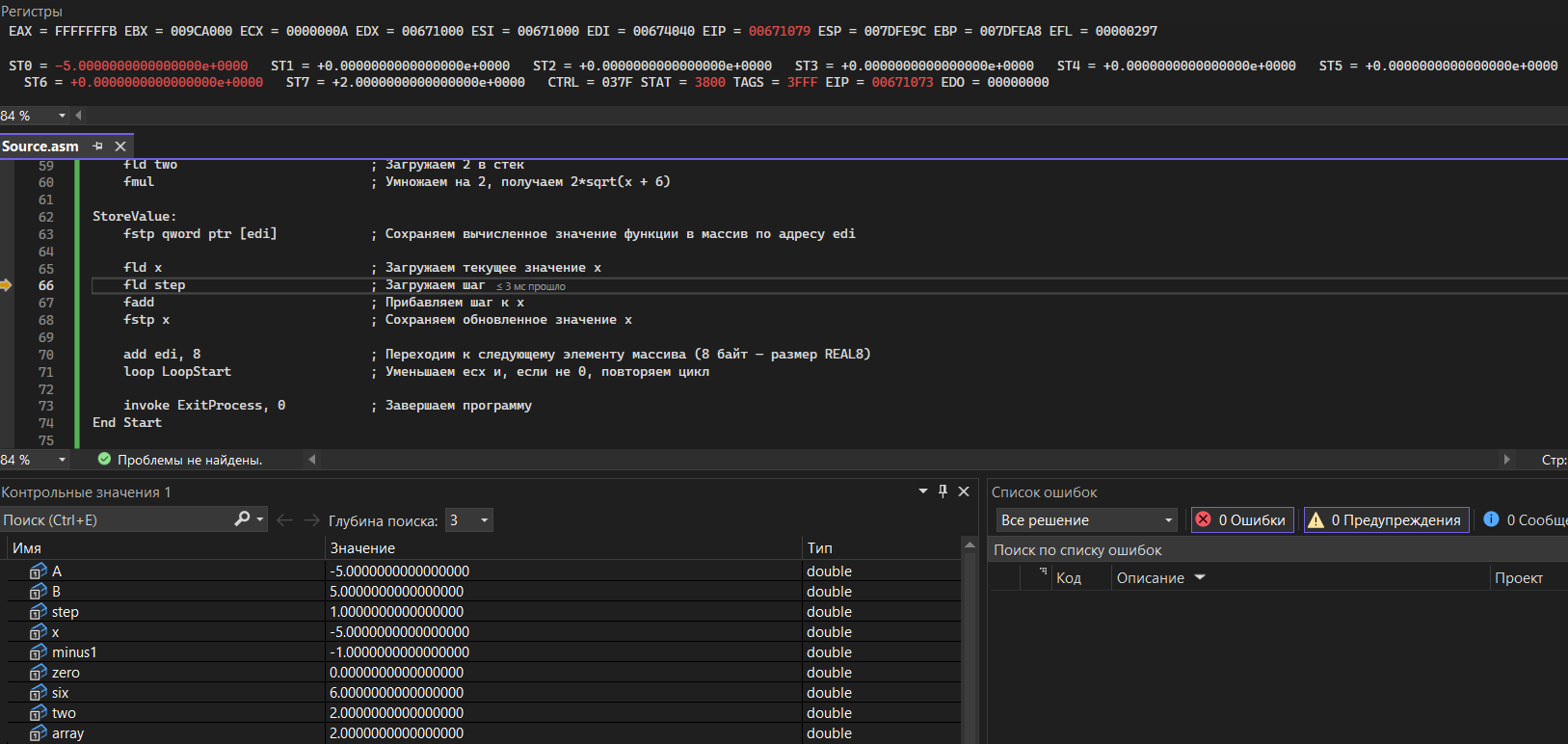
Загружаем текущее x в стек FPU ****Сохраняем его во временную переменную tmp в памяти ****Загружаем tmp обратно в стек FPU ****

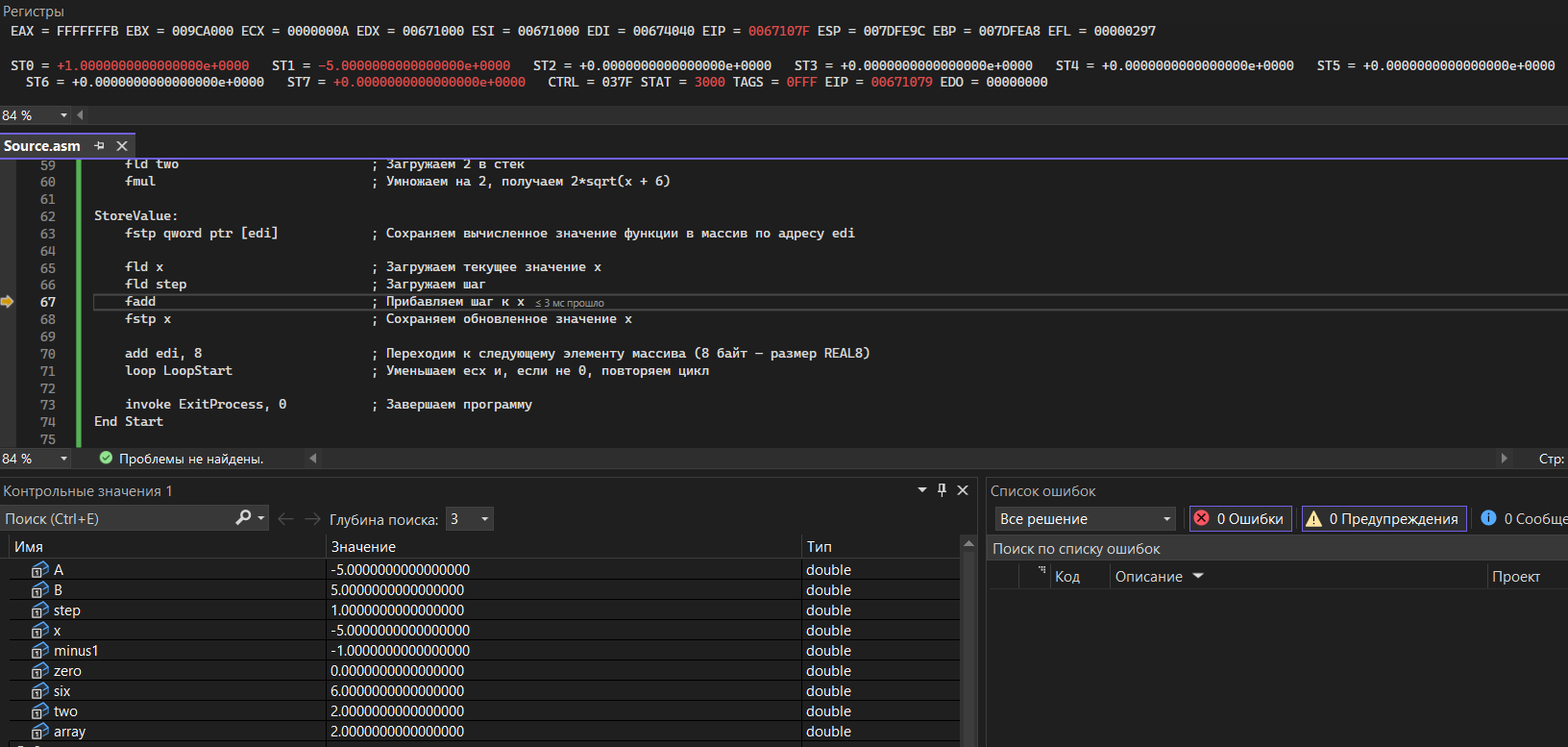
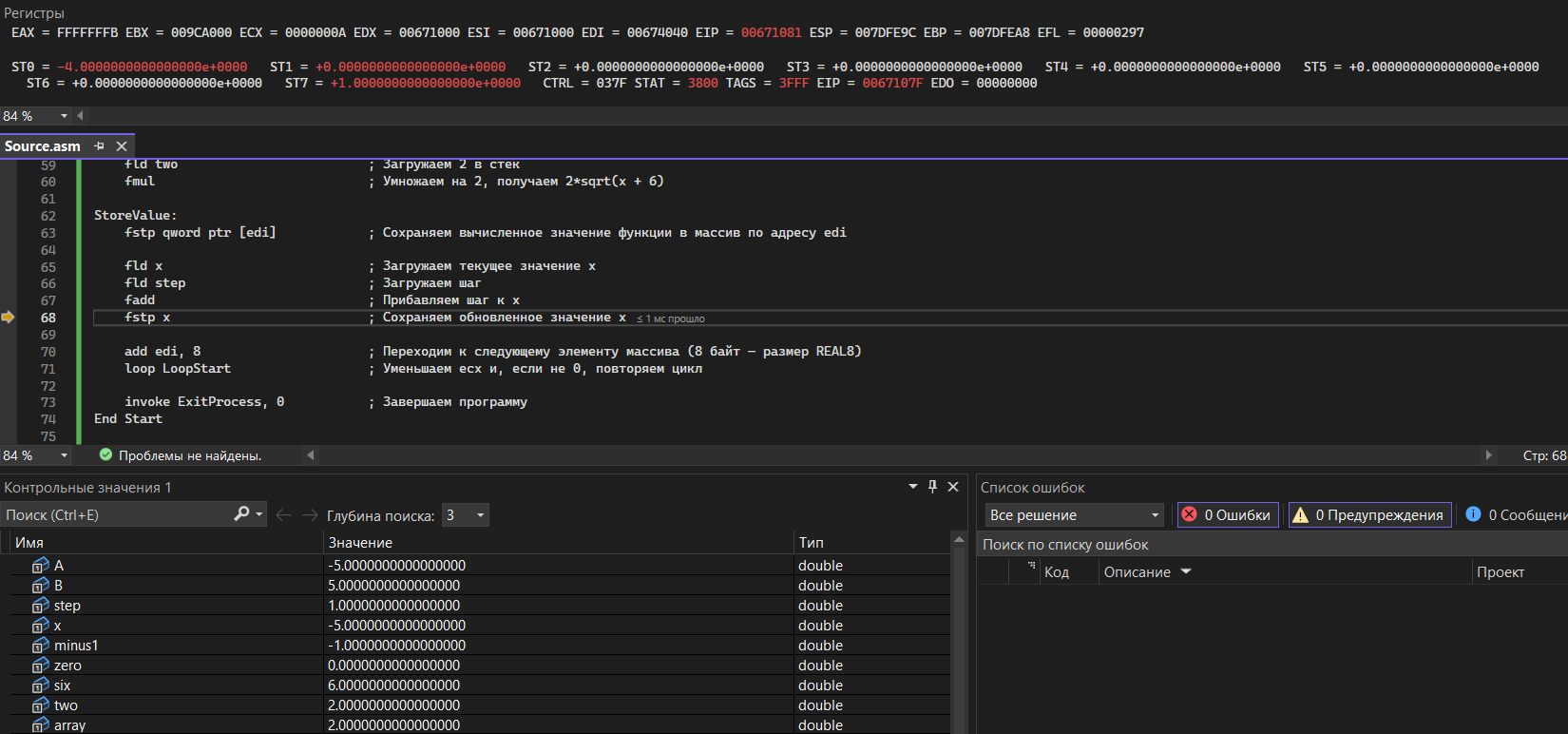
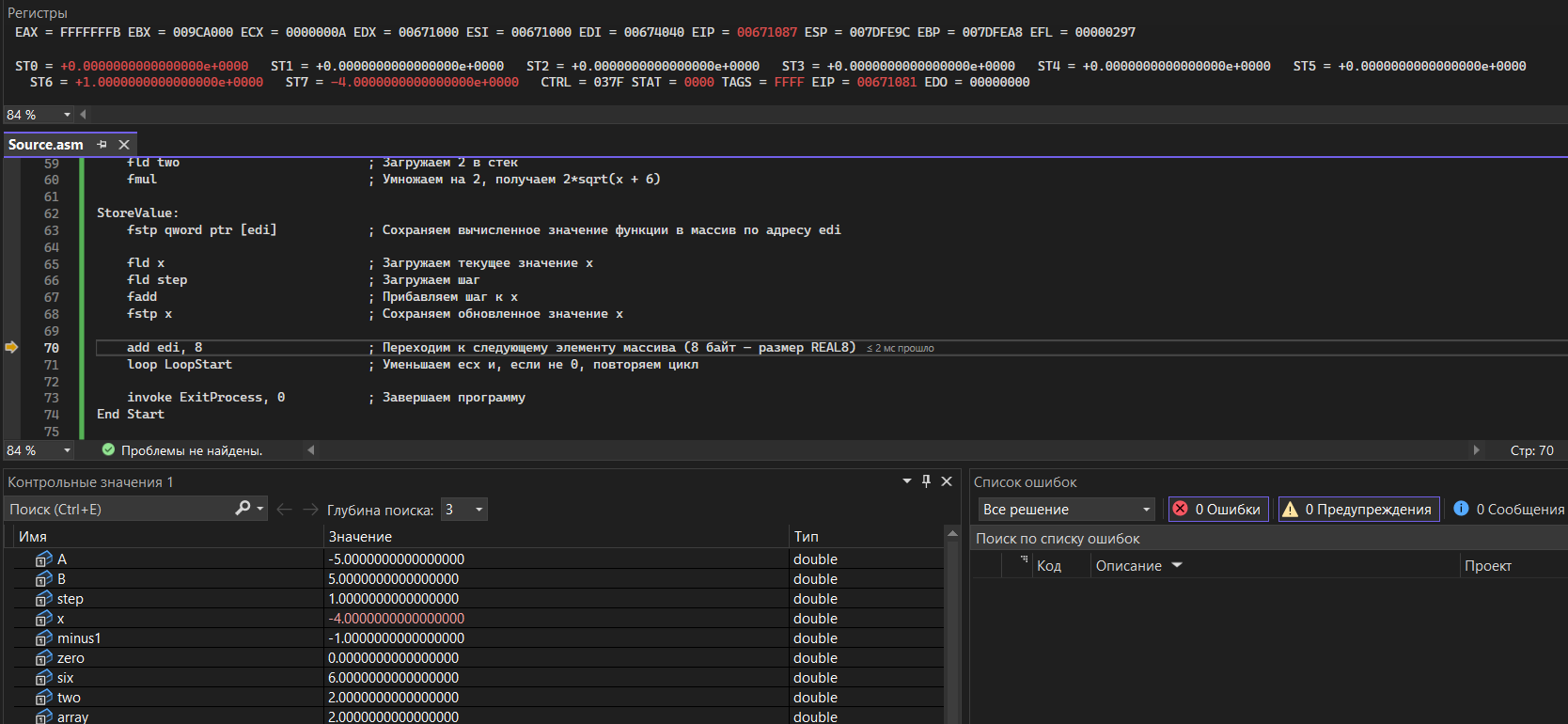
Преобразуем значение x в целое (округление вниз) и сохраняем в tmp\_int ****Загружаем целочисленное значение x в eax для сравнения **  
**

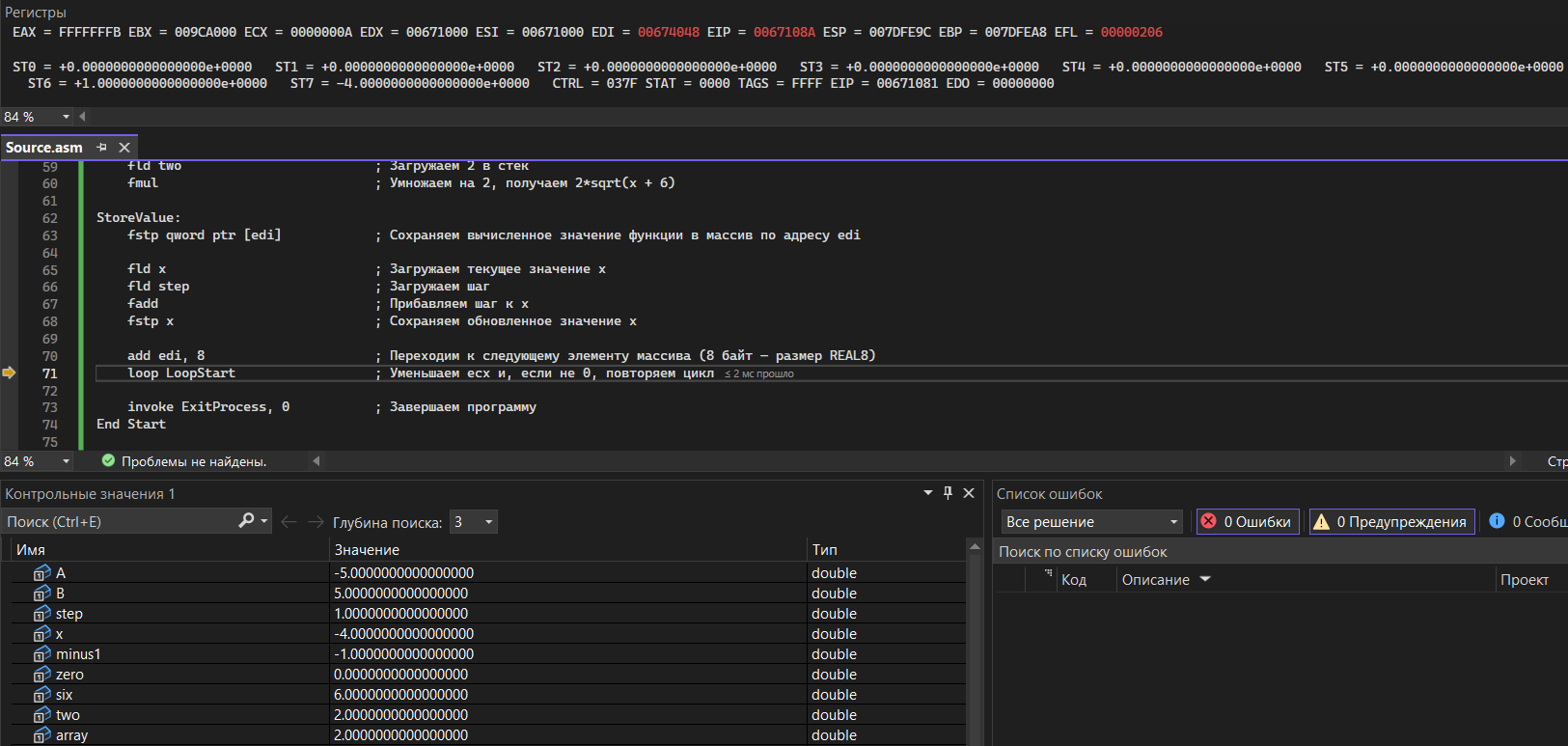
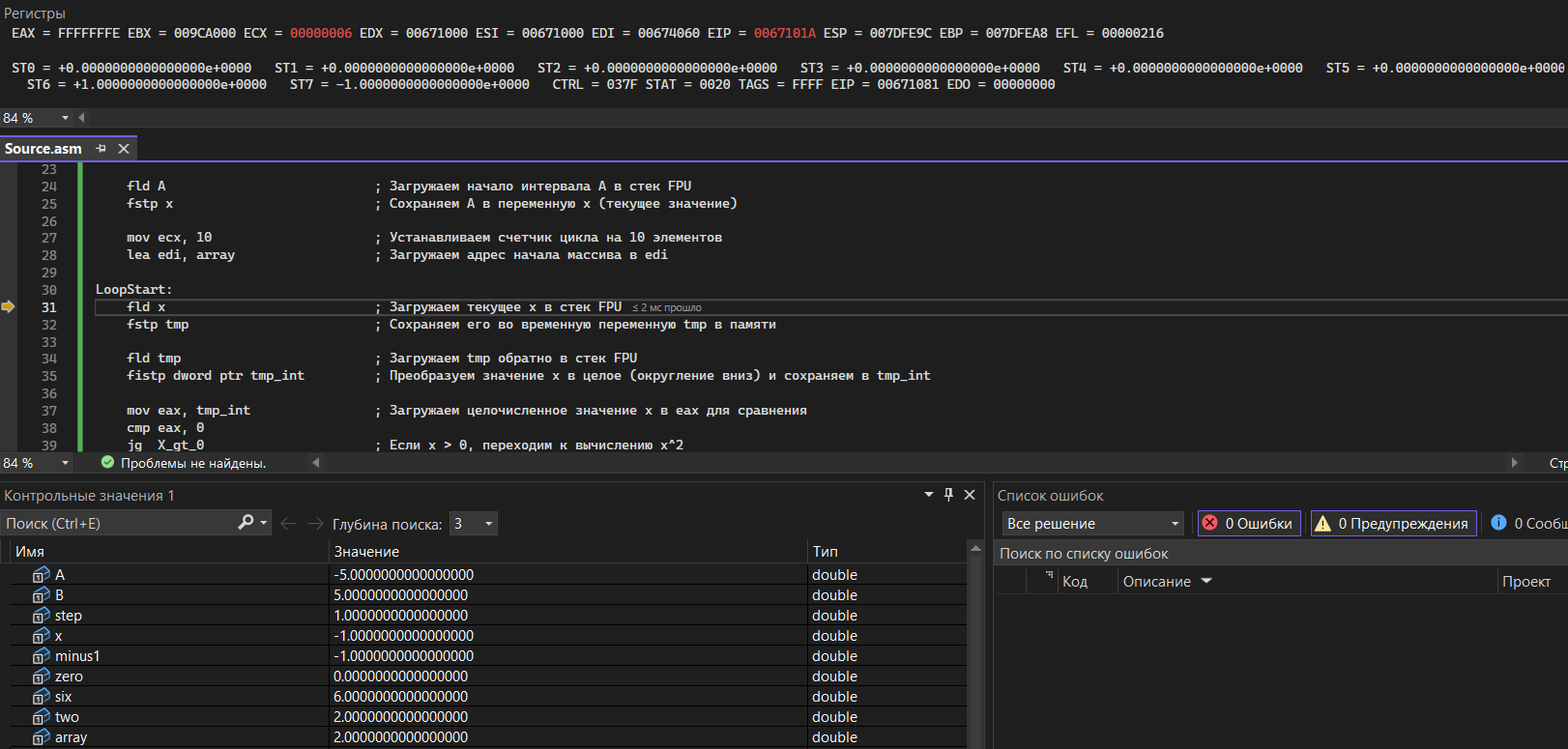
Если x > 0, переходим к вычислению x^2 **   
** Если x < -1, переходим к вычислению 2\*sqrt(x + 6) ****

Загружаем x в стек **** Загружаем 6 в стек **** Вычисляем x + 6 ****

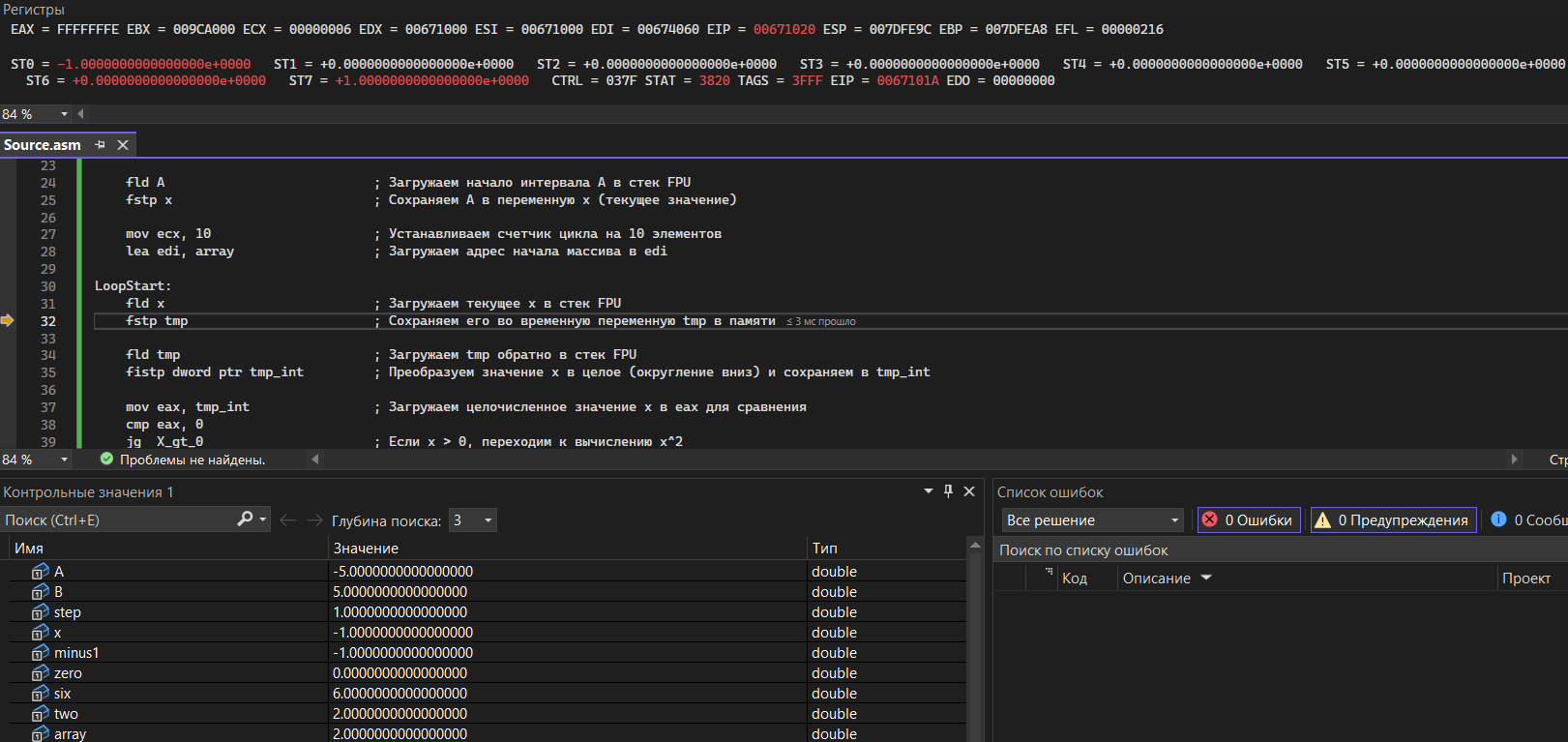
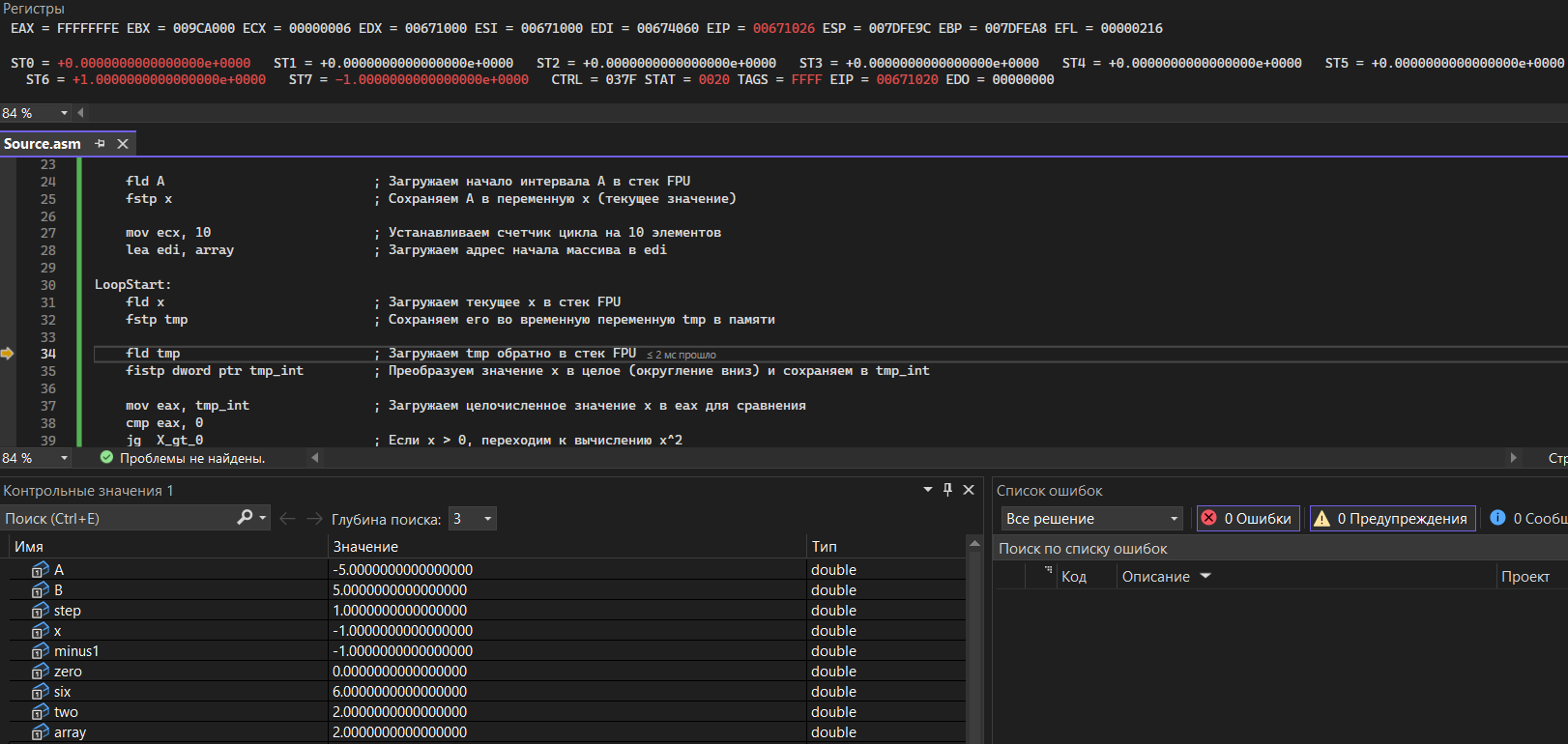
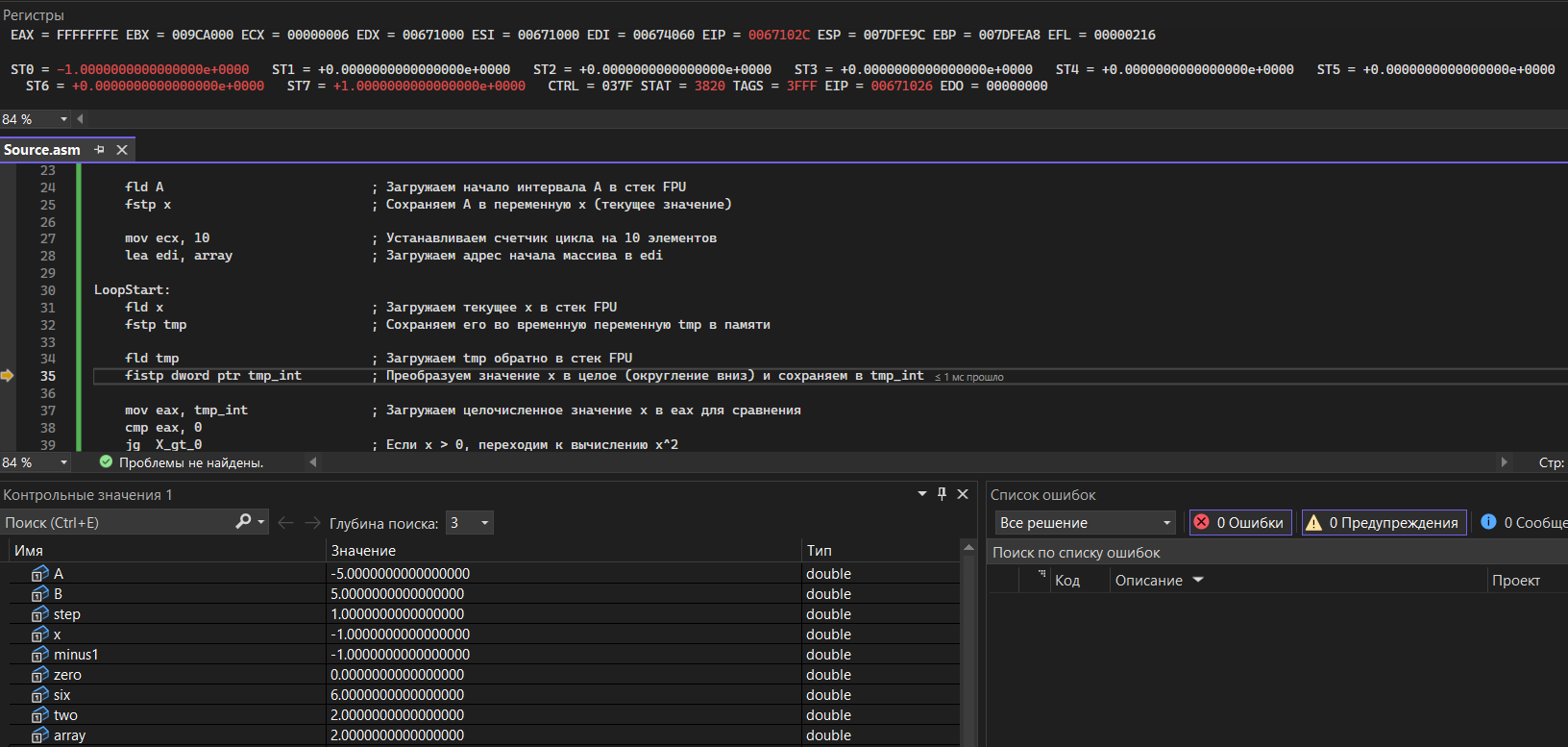
Берем квадратный корень sqrt(x + 6) **** Загружаем 2 в стек **** Умножаем на 2, получаем 2\*sqrt(x + 6) ****

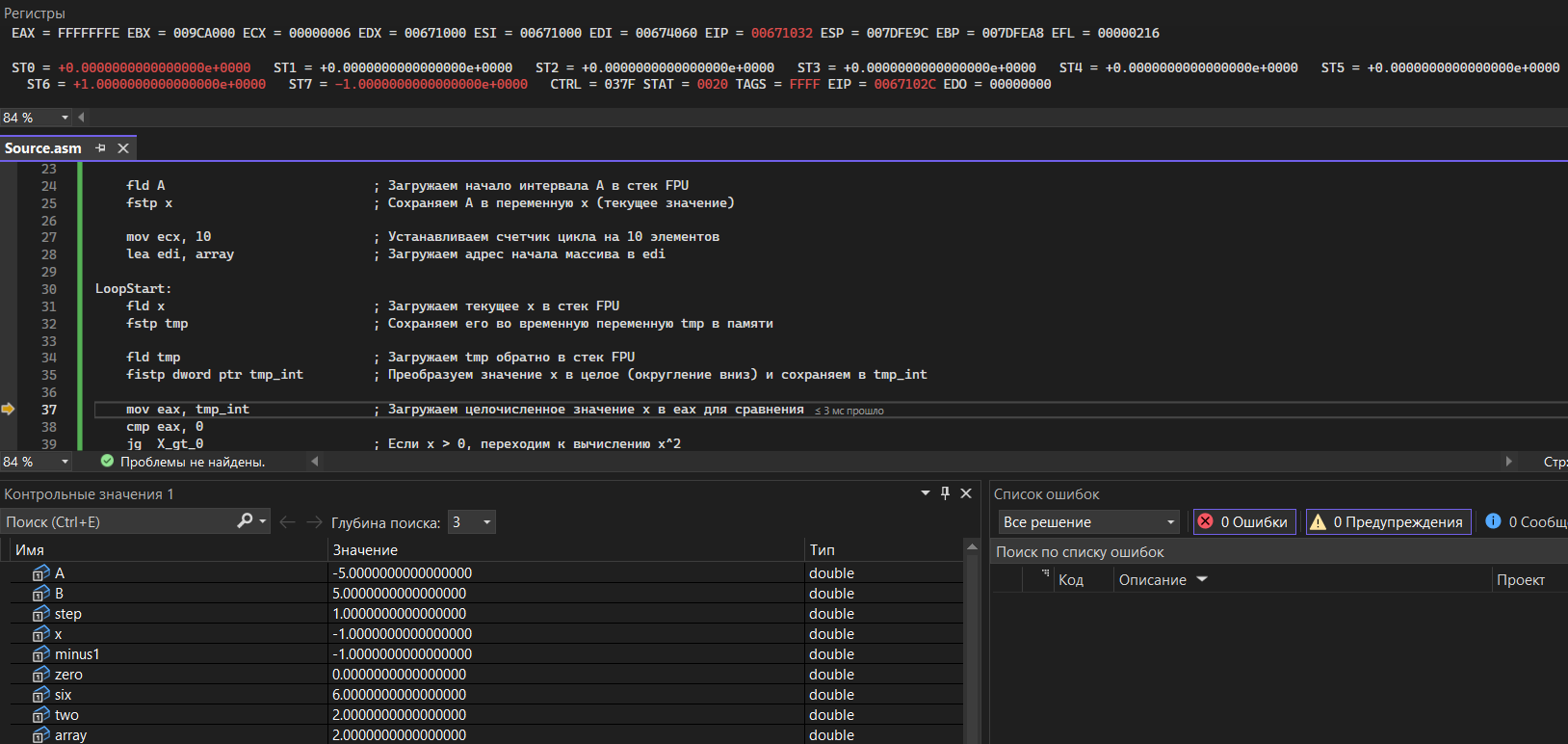
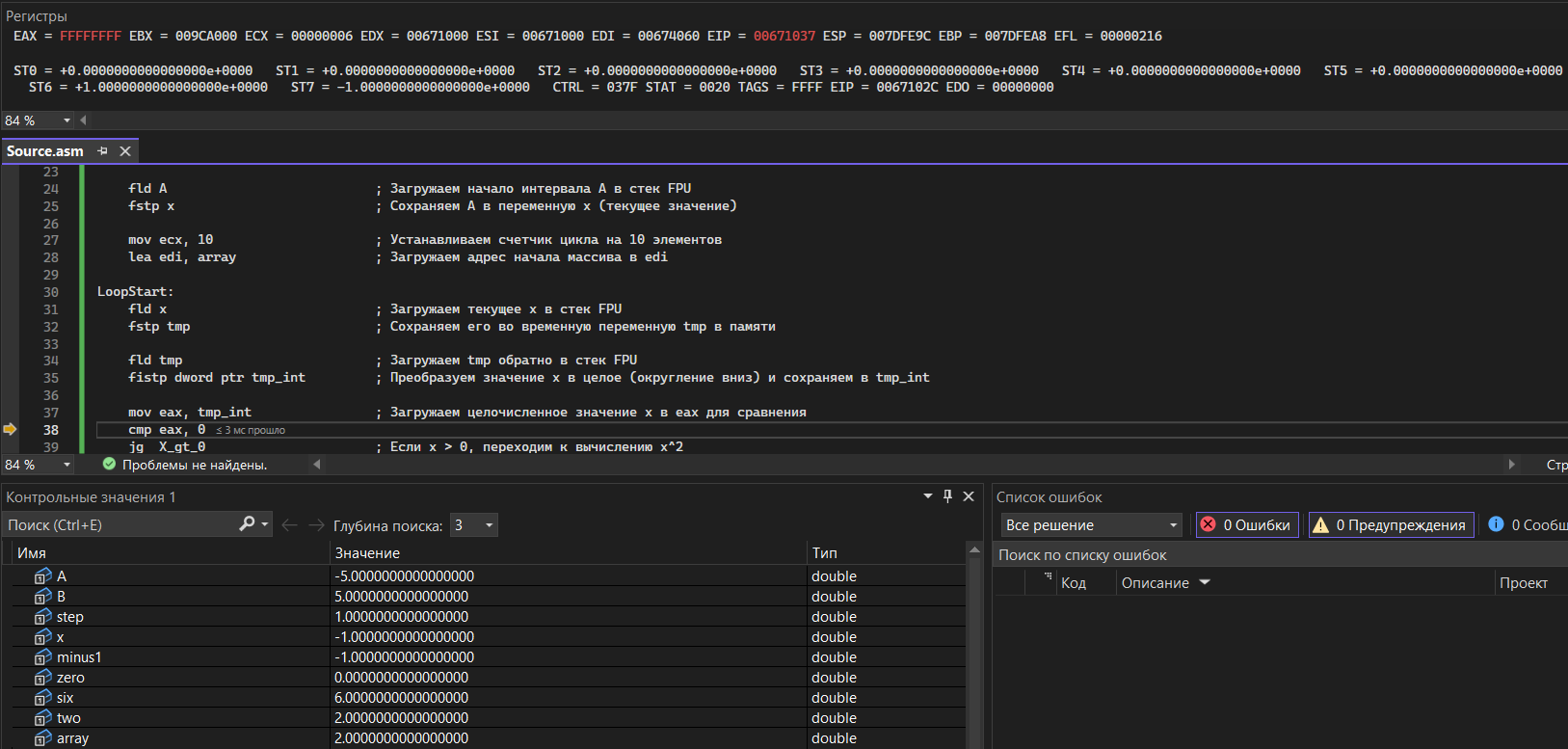
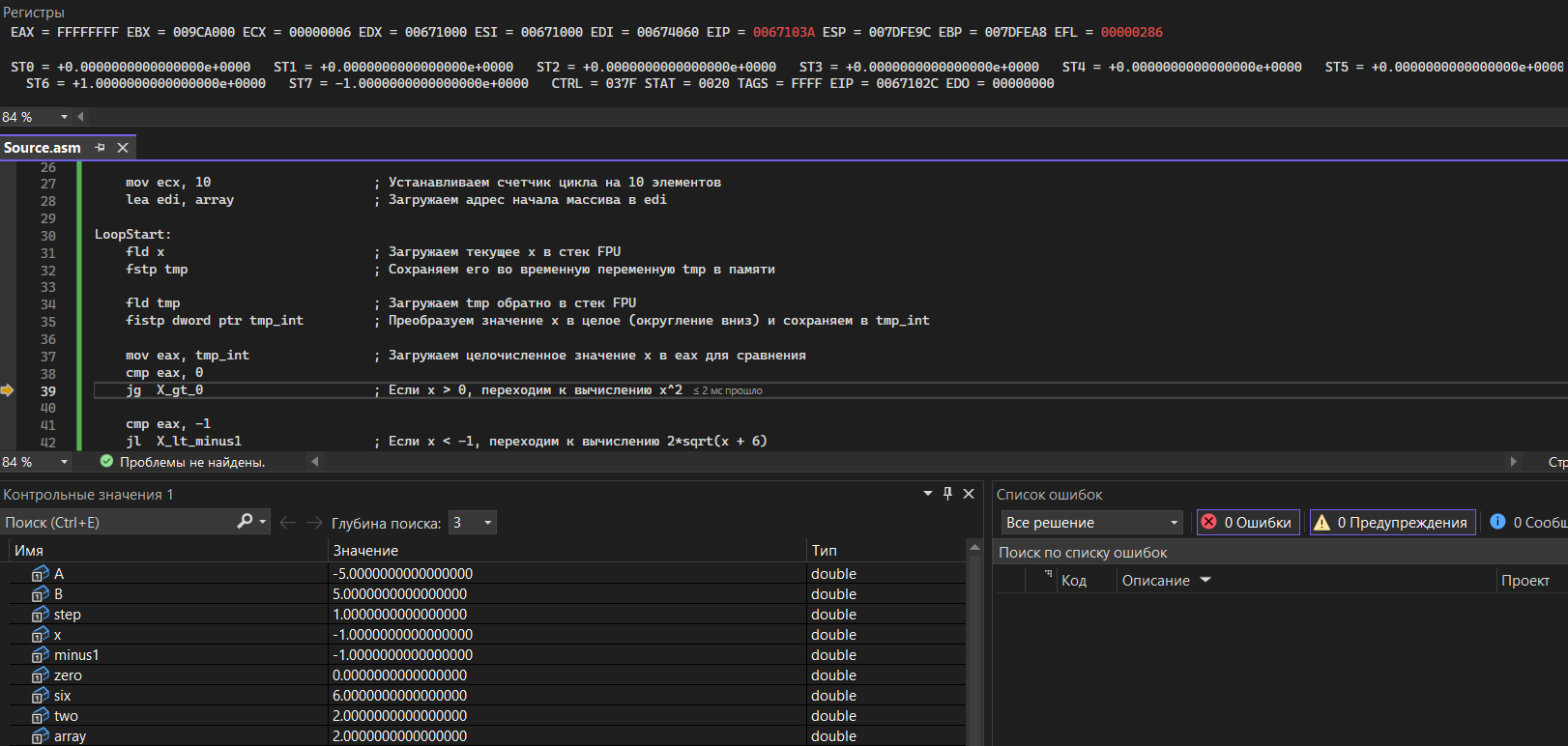
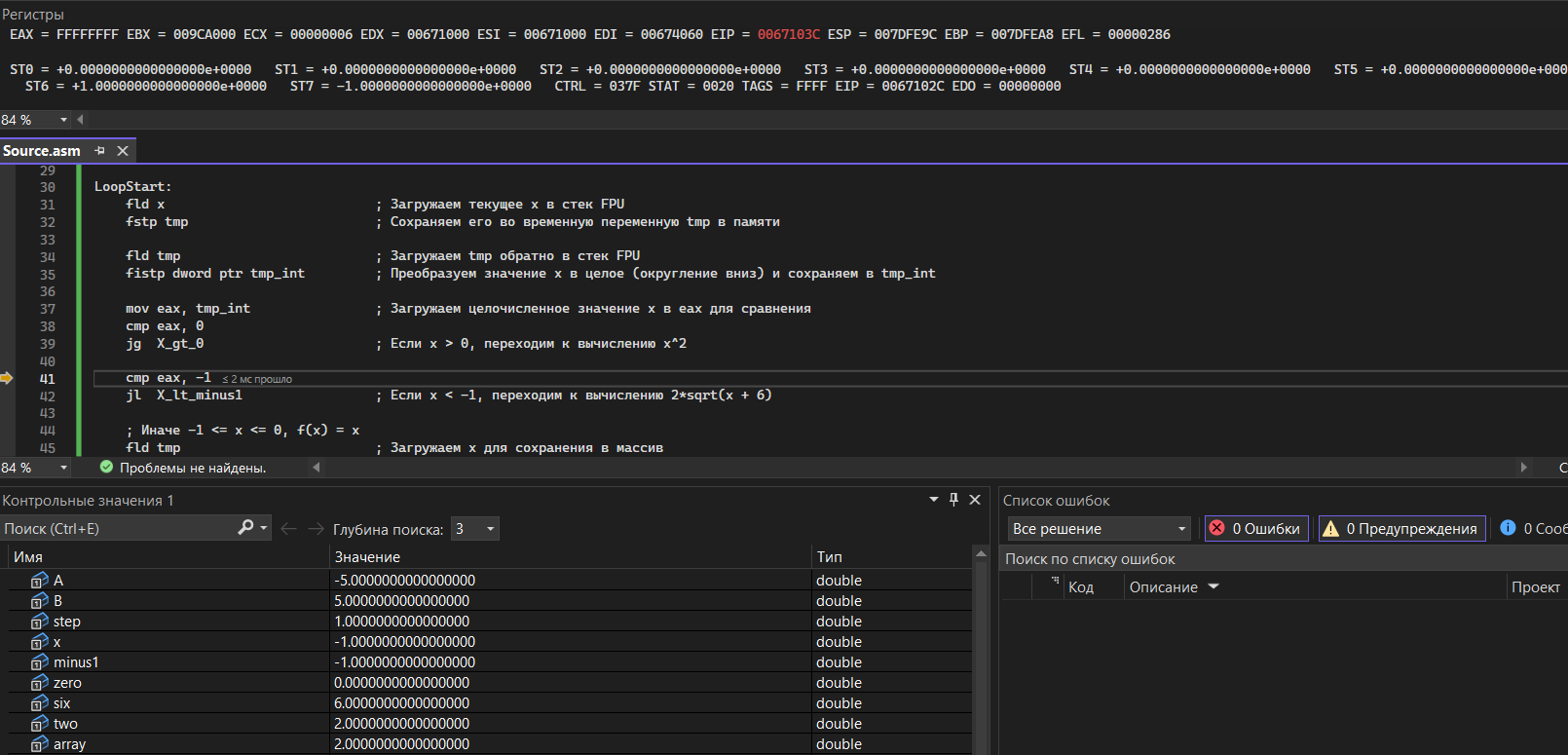
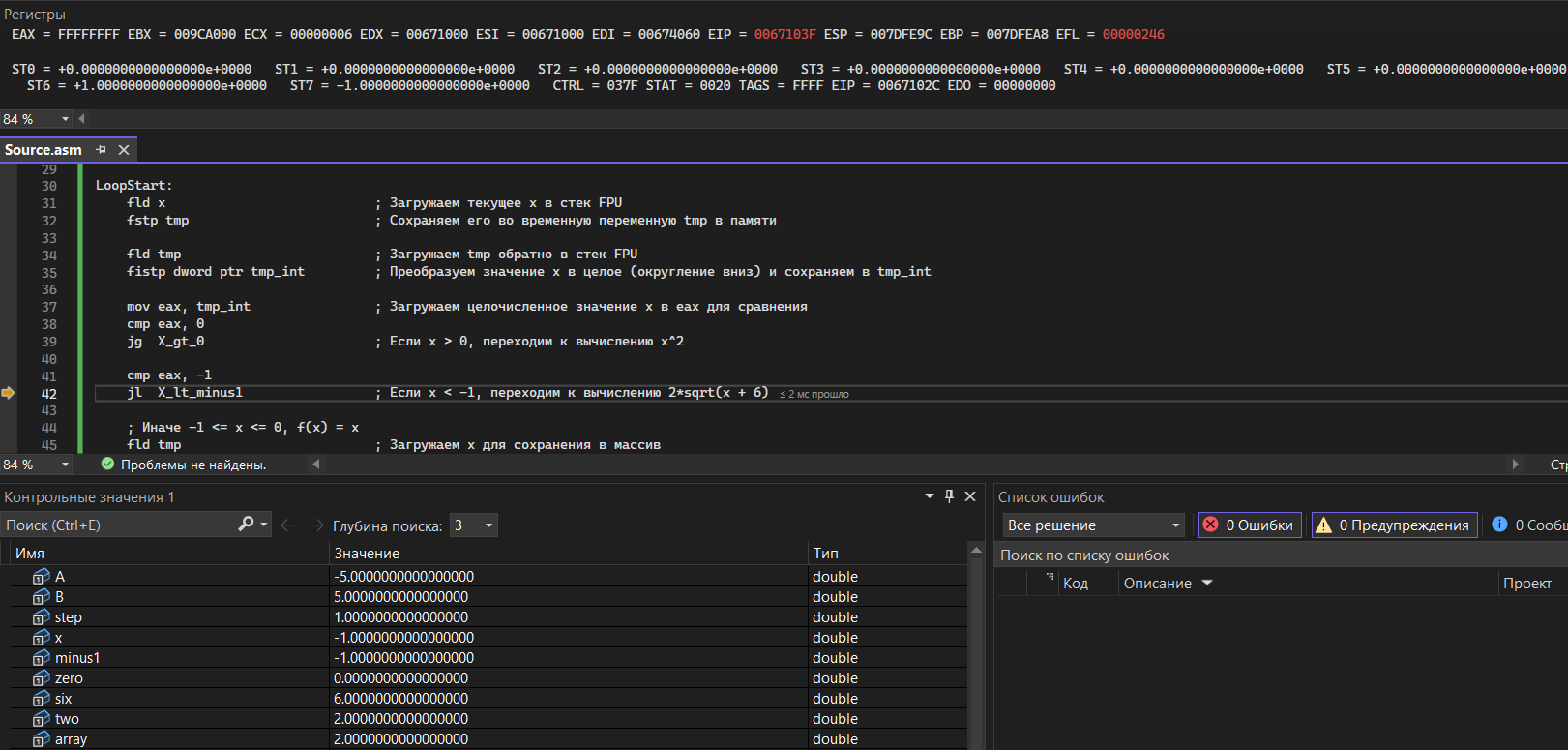
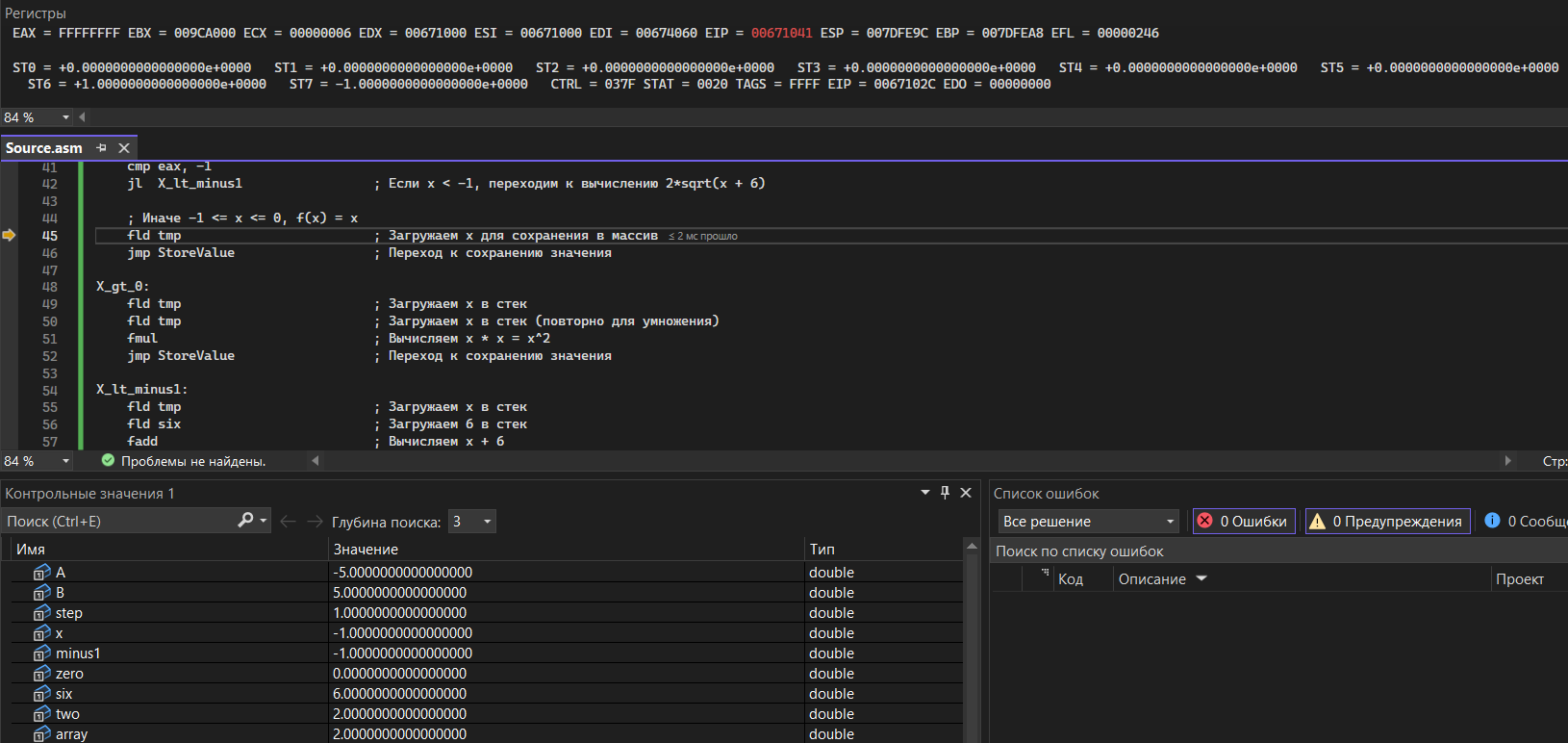
Сохраняем вычисленное значение функции в массив по адресу edi **** Загружаем текущее значение x **** Загружаем шаг****

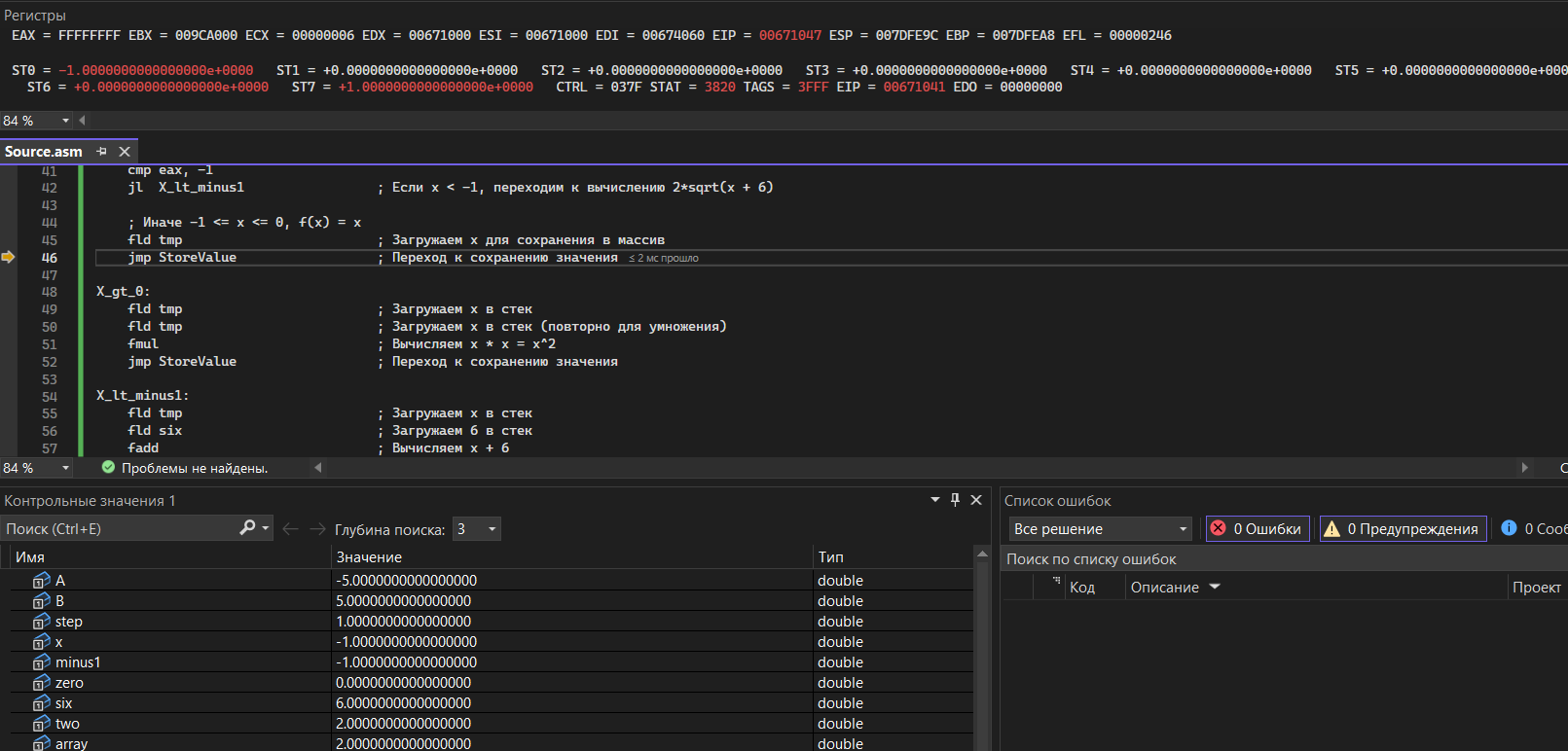
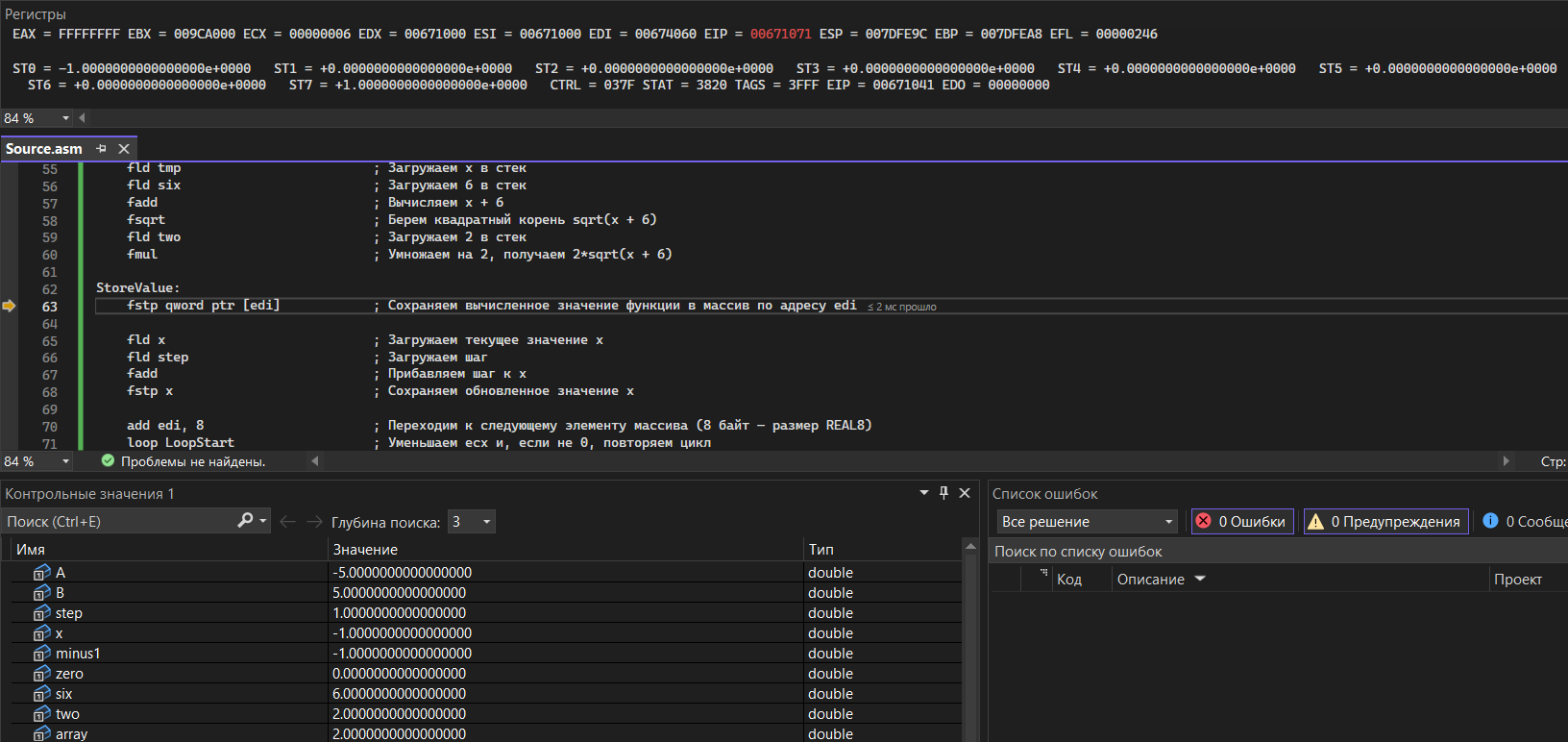
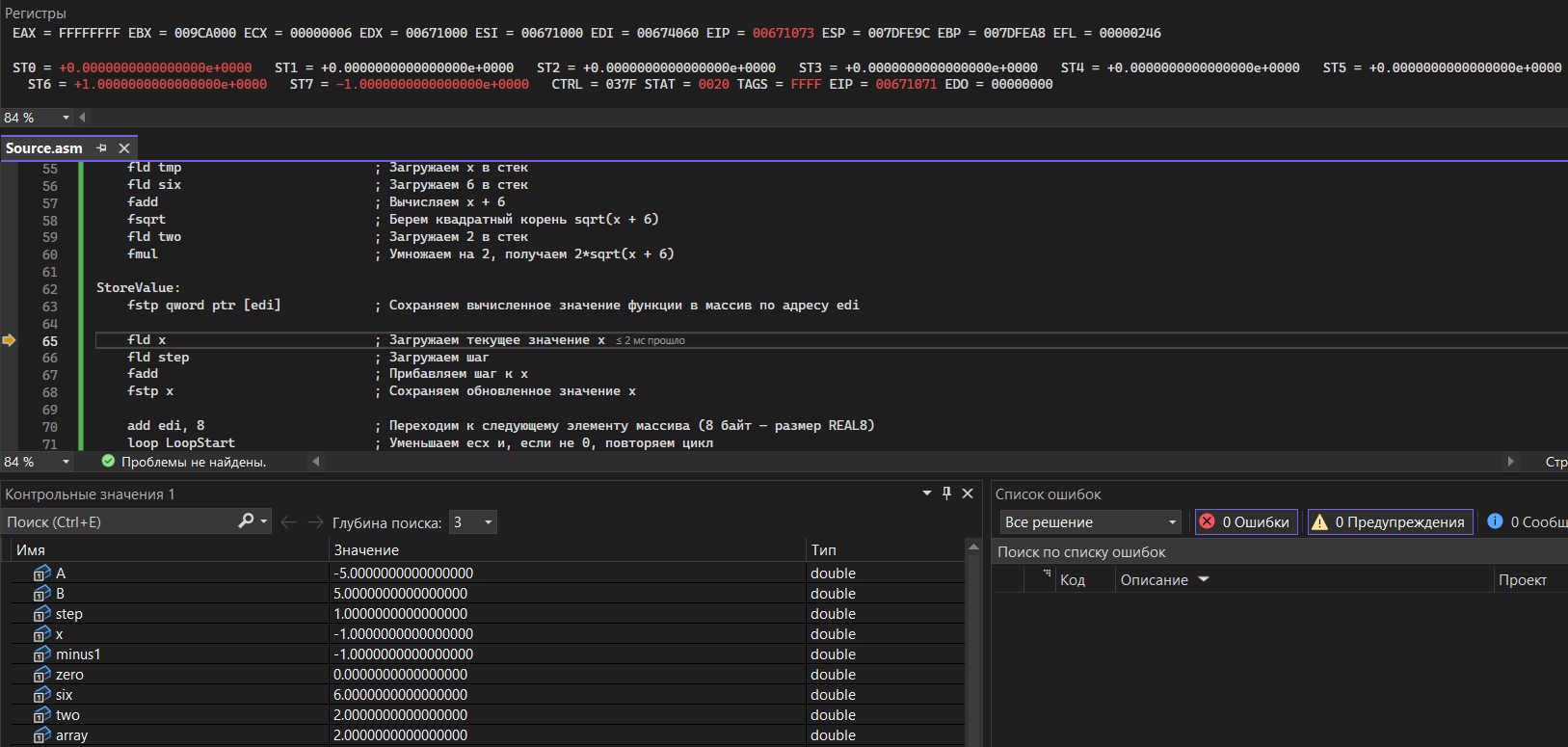
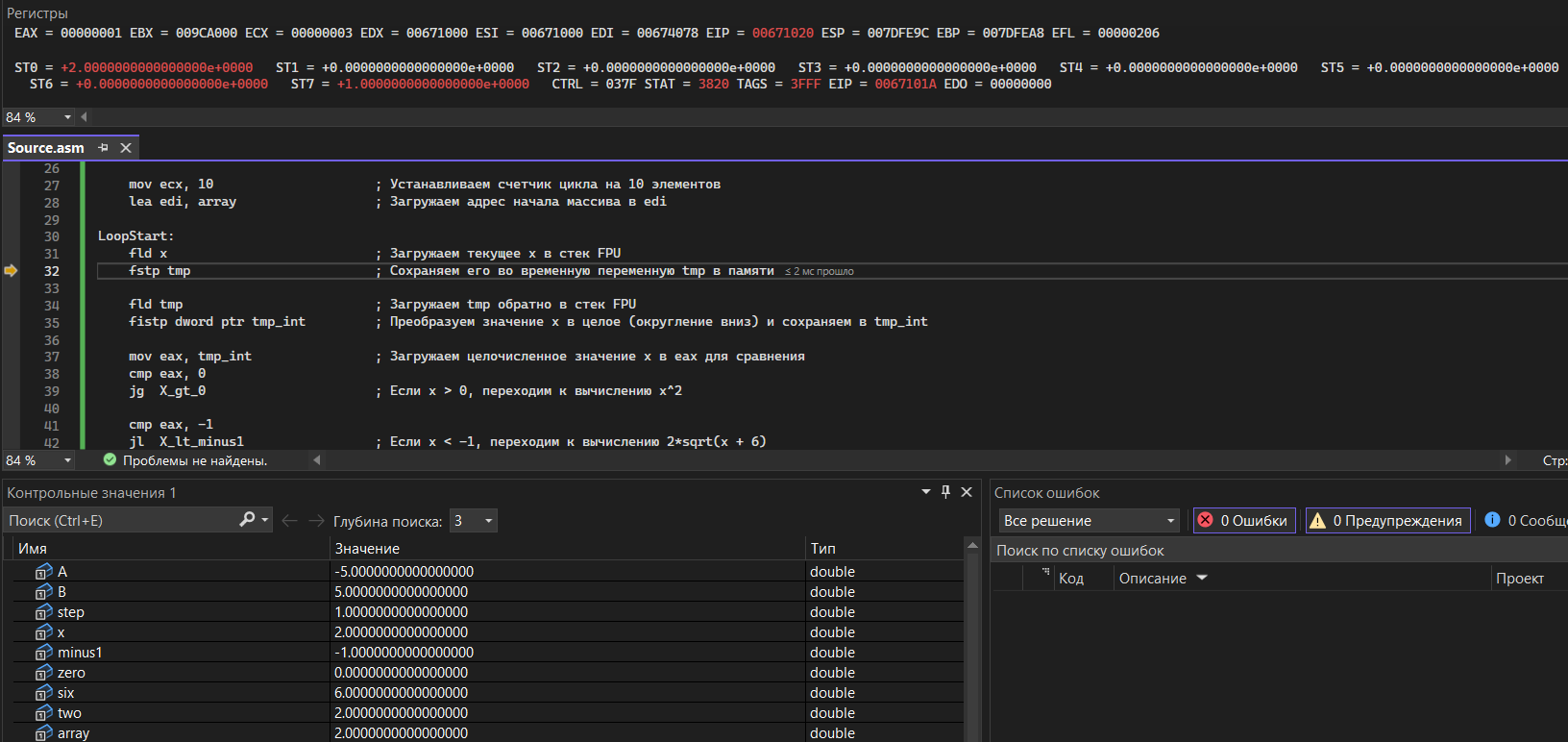
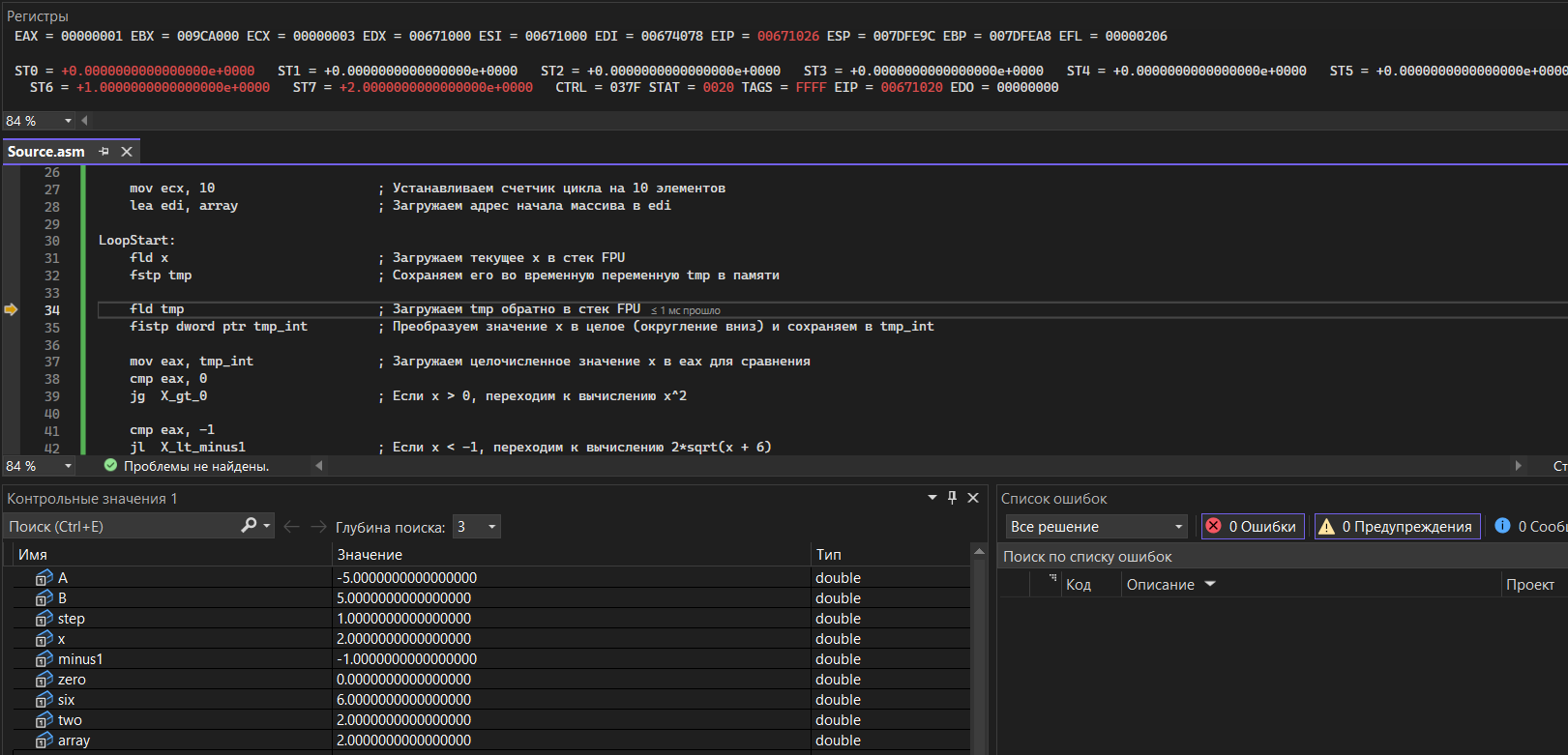
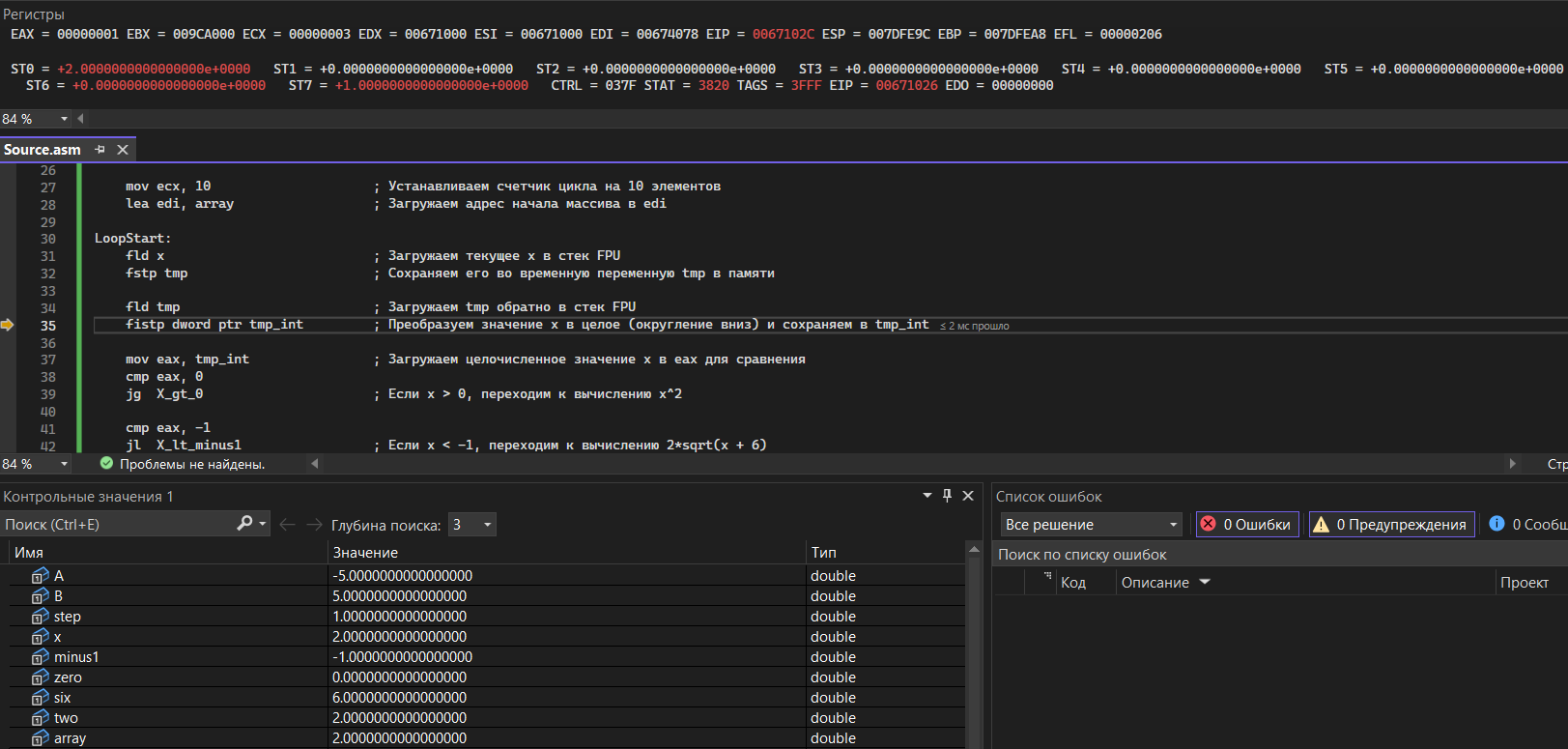
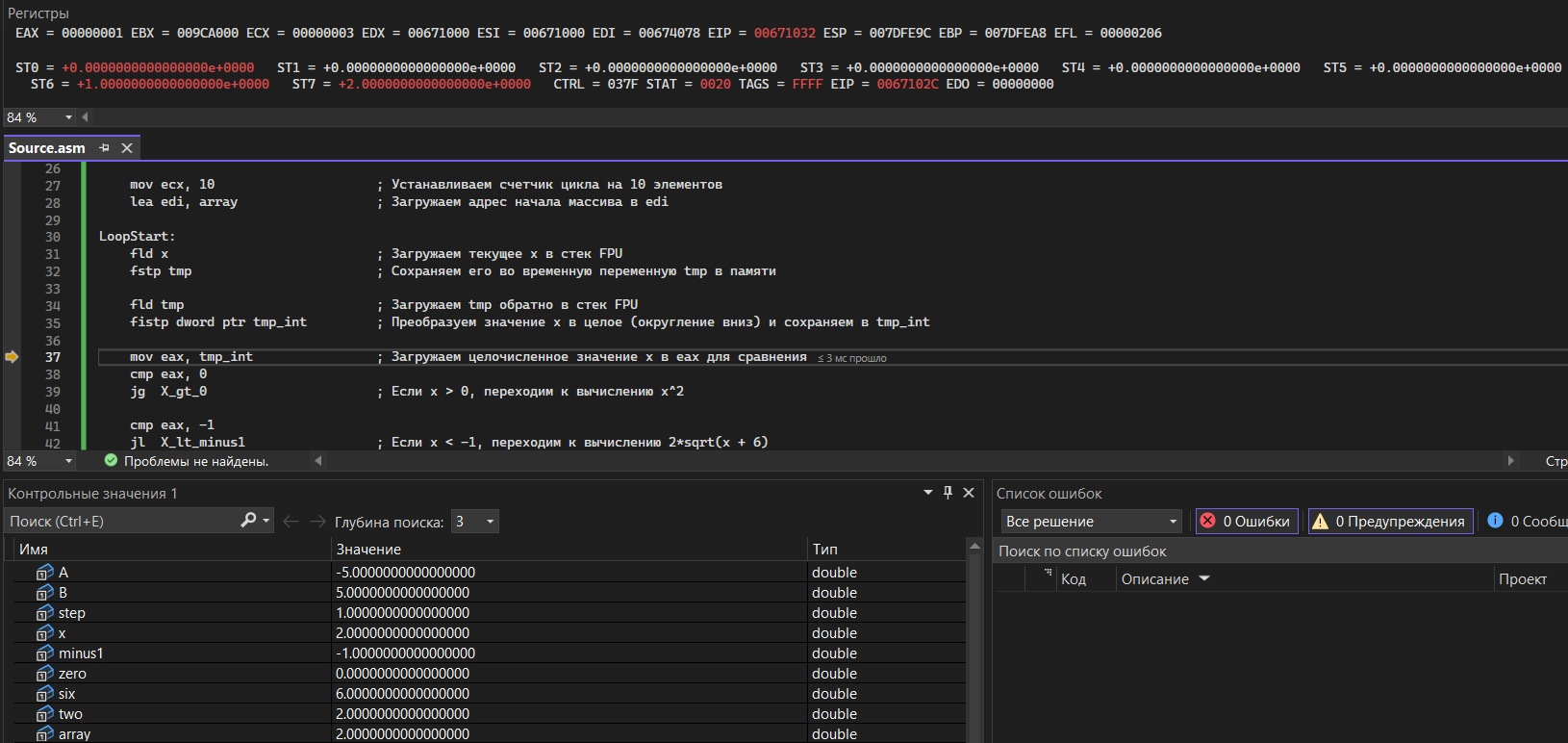
Прибавляем шаг к x **** Сохраняем обновленное значение x **** Переходим к следующему элементу массива (8 байт — размер REAL8) ****

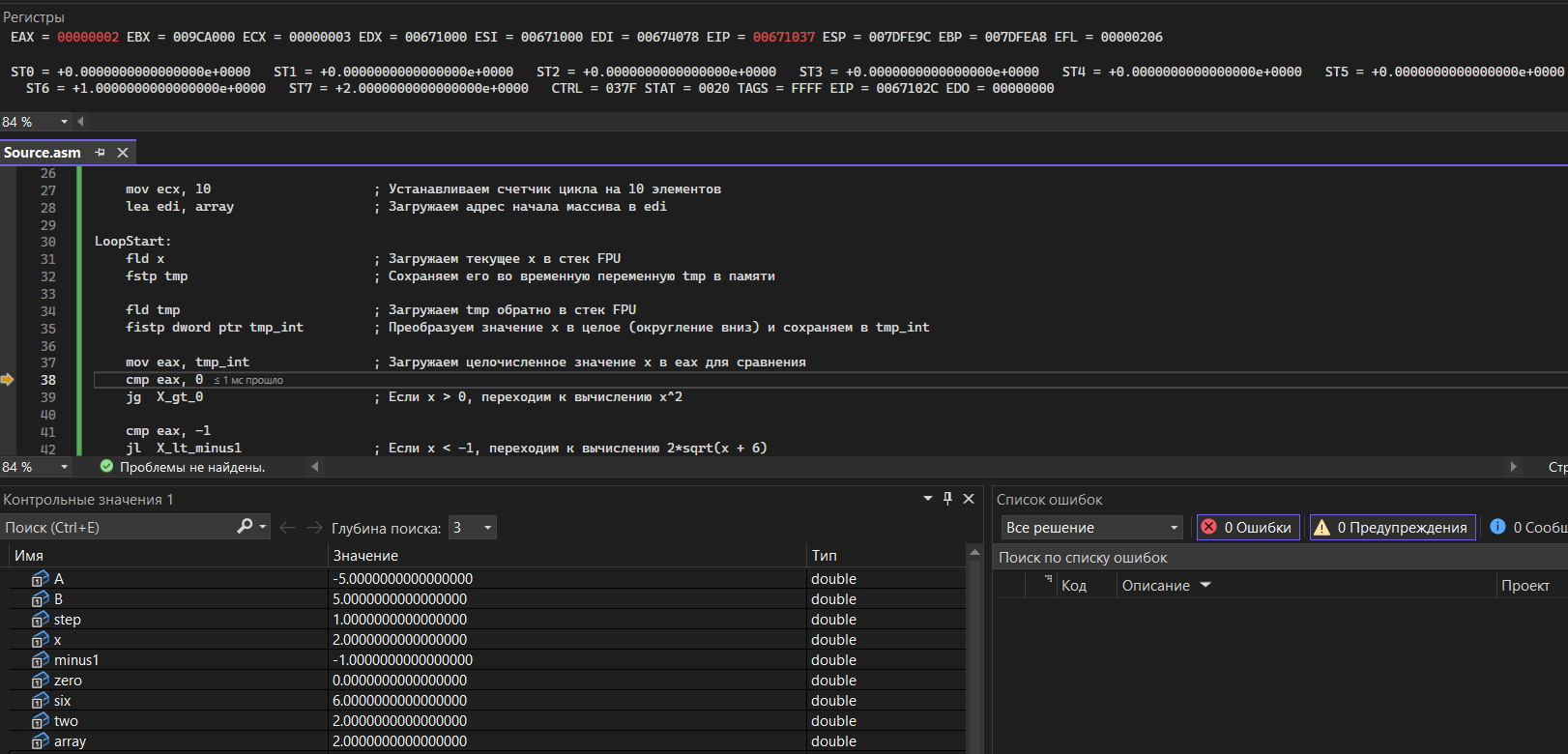
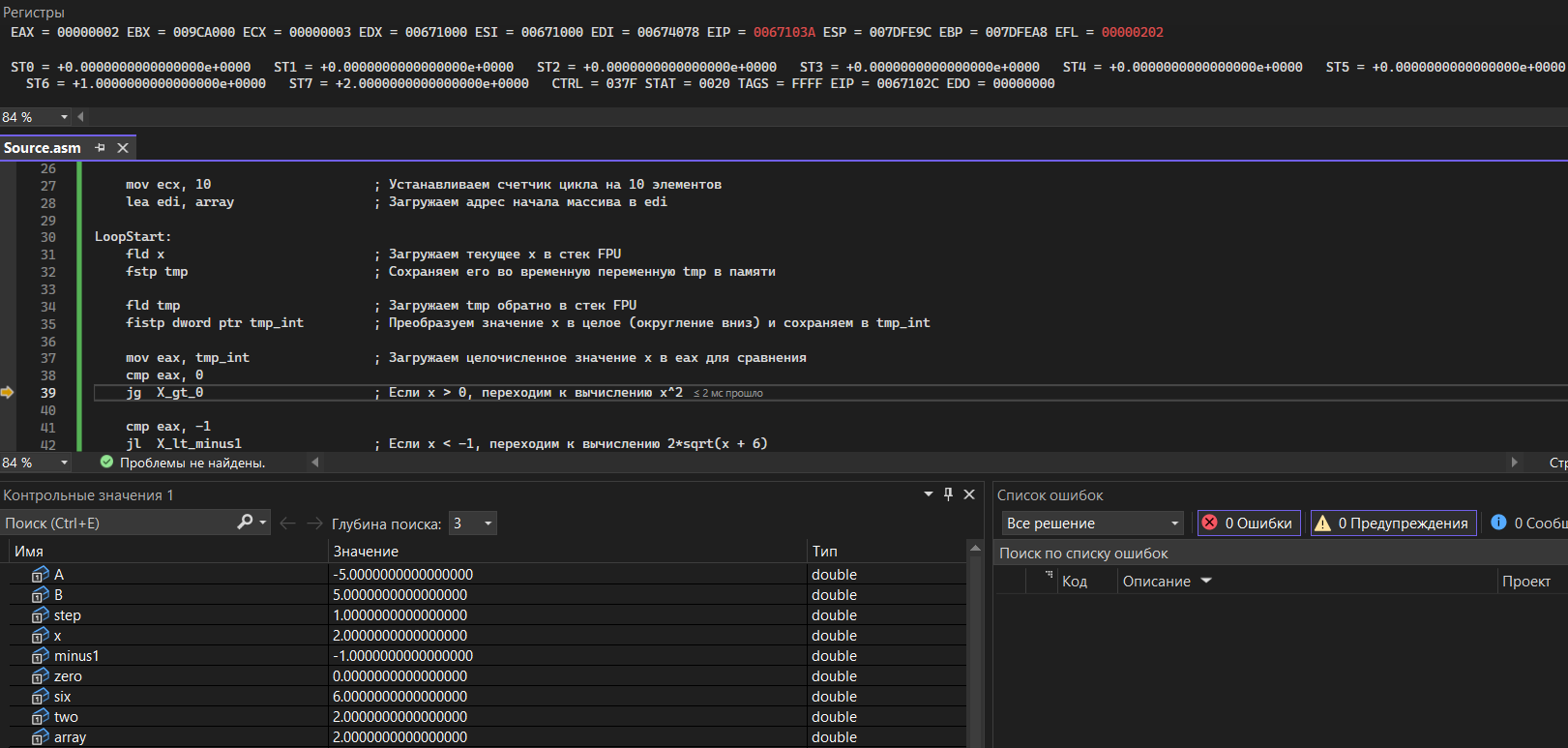
Уменьшаем ecx и, если не 0, повторяем цикл **  
2) Выполнение программы при условии, что x=-1 и вычисление идет по формуле - x**Загружаем текущее x в стек FPU ****

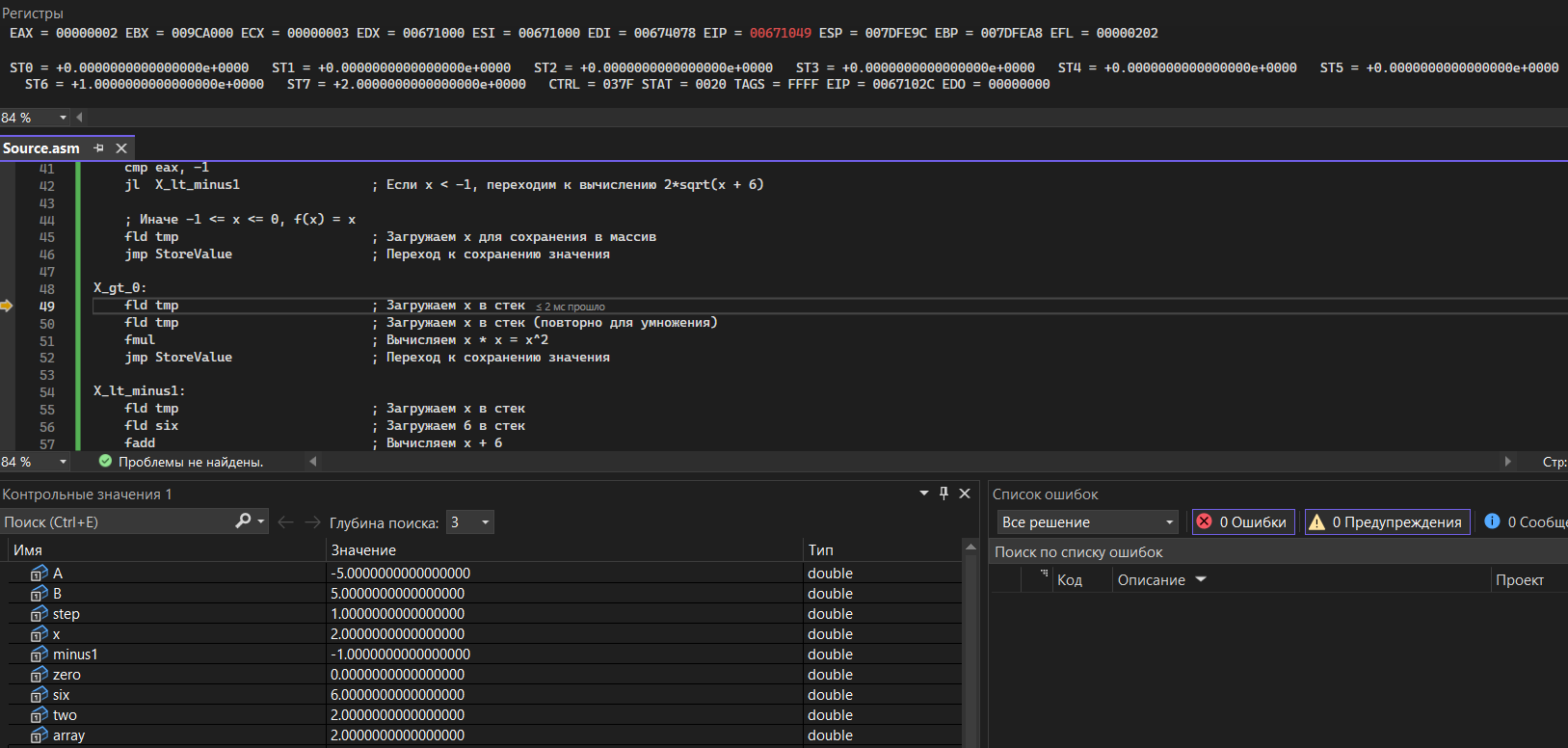
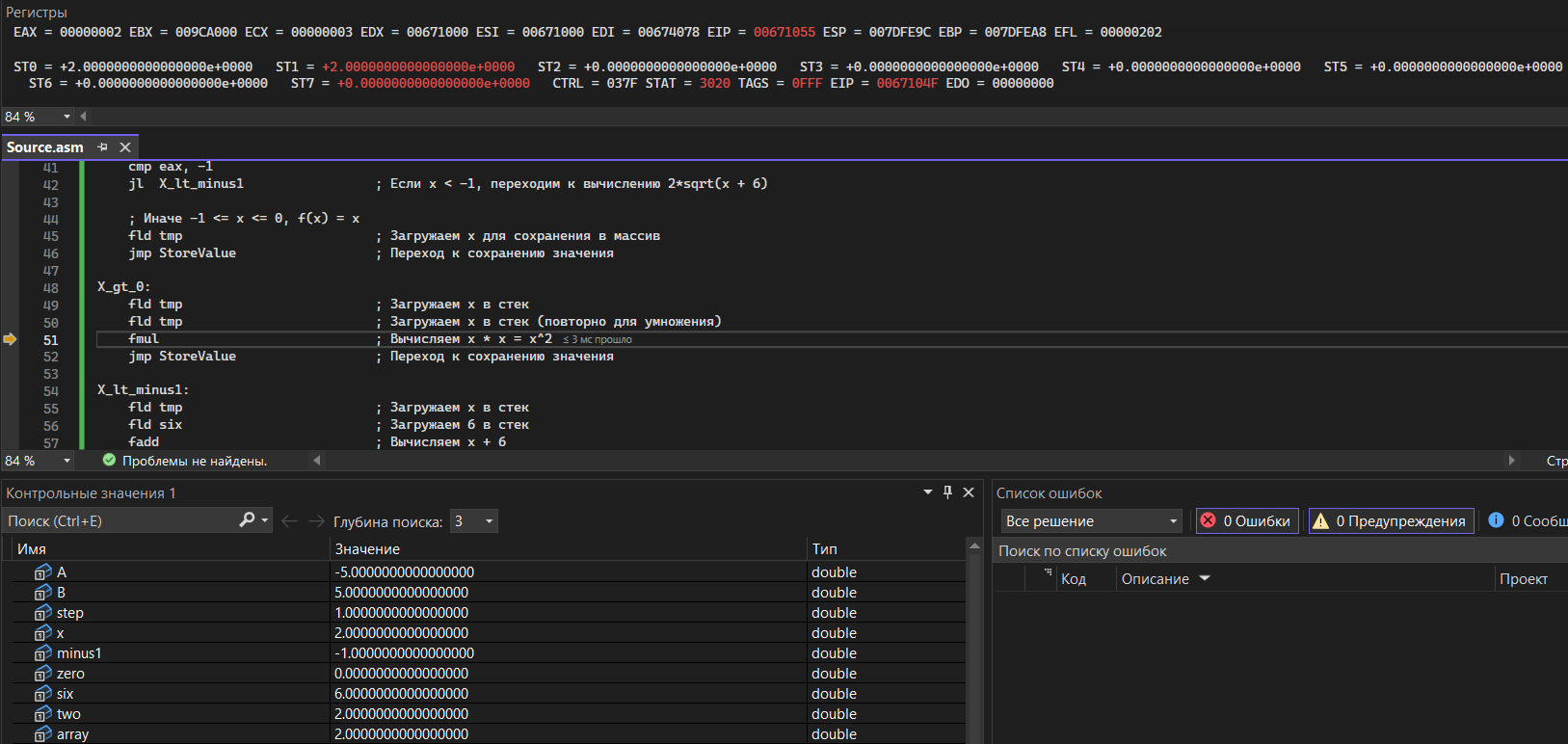
Сохраняем его во временную переменную tmp в памяти

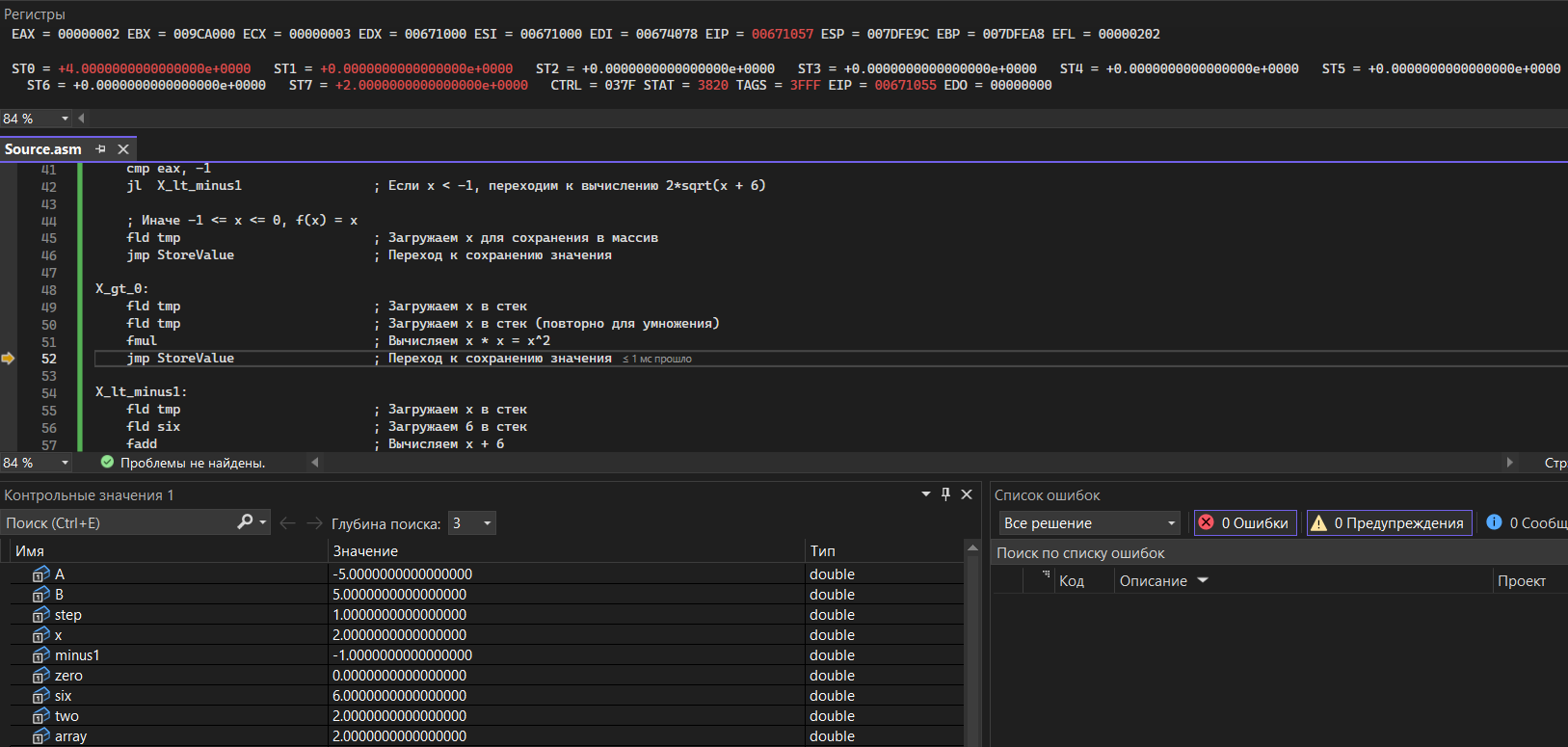
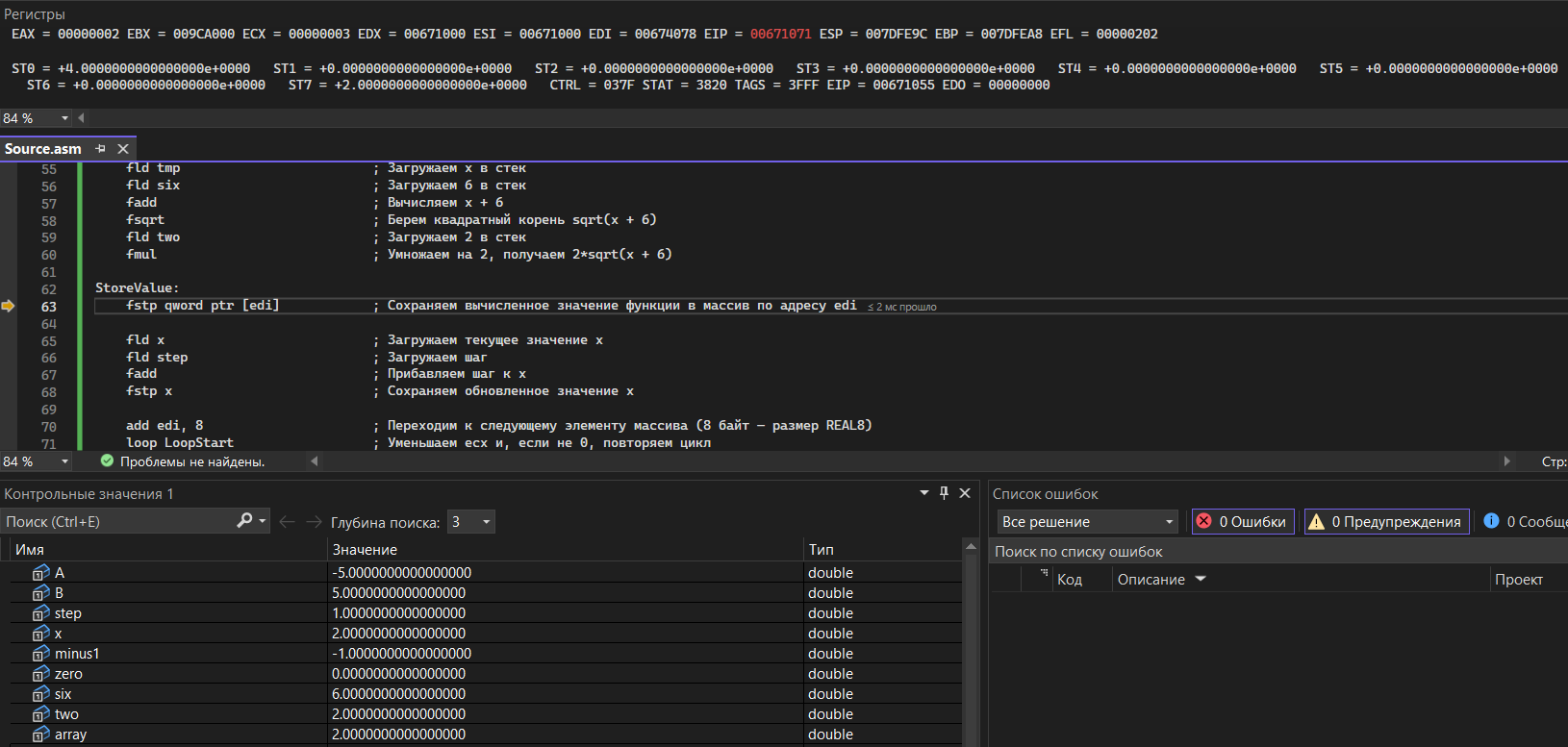
**** Загружаем tmp обратно в стек FPU **** Преобразуем значение x в целое (округление вниз) и сохраняем в tmp\_int ****

Загружаем целочисленное значение x в eax для сравнения **  
** Если x > 0, переходим к вычислению x^2 **  
** Если x < -1, переходим к вычислению 2\*sqrt(x + 6) **** Загружаем x для сохранения в массив ****

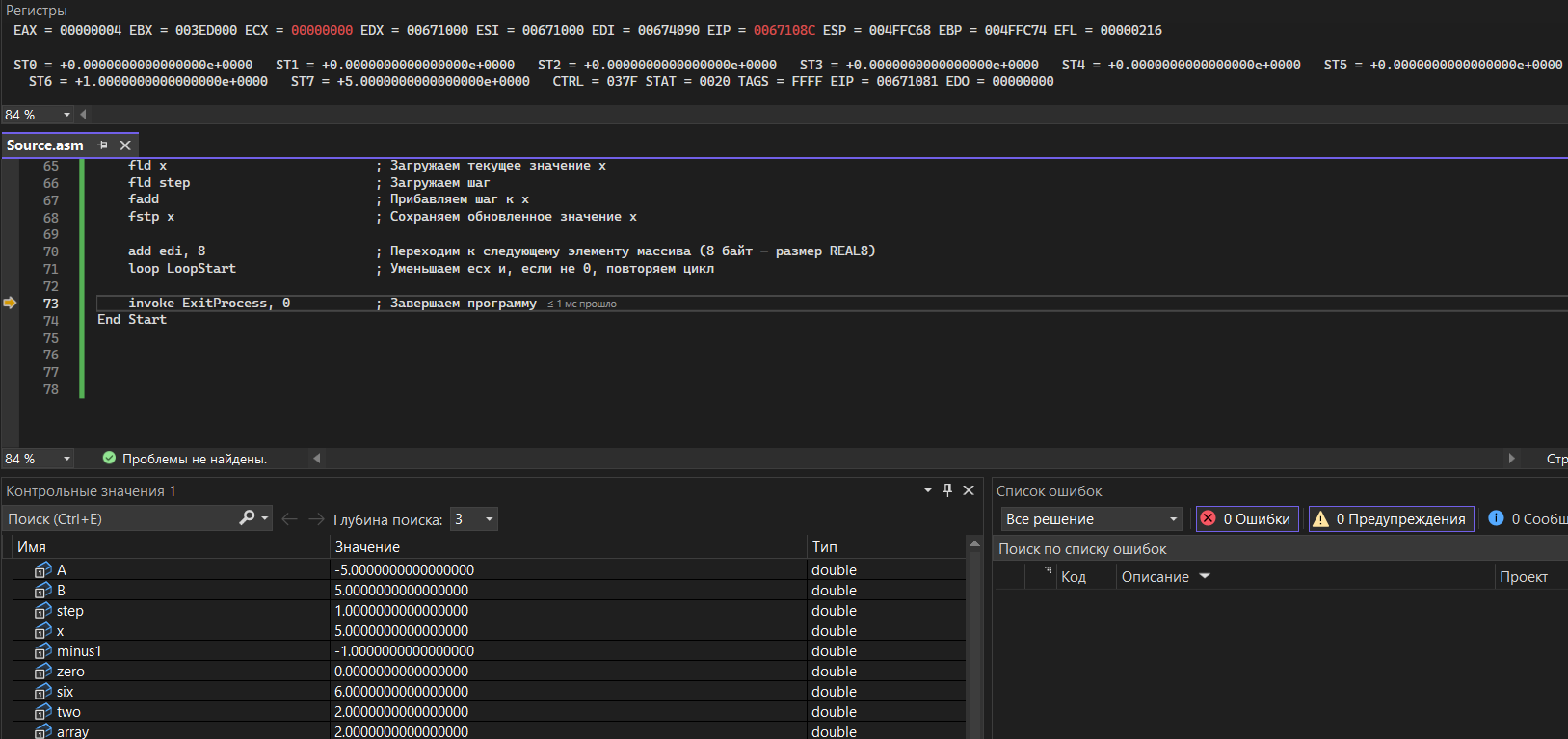
Переход к сохранению значения **** Сохраняем вычисленное значение функции в массив по адресу edi **** Загружаем текущее значение x **  
3) Выполнение программы при условии, что x=2 и вычисление идет по формуле – x^2**  
Сохраняем его во временную переменную tmp в памяти **** Загружаем tmp обратно в стек FPU **** Преобразуем значение x в целое (округление вниз) и сохраняем в tmp\_int **** Загружаем целочисленное значение x в eax для сравнения **** Загружаем целочисленное значение x в eax для сравнения

cmp eax, 0 **** Если x > 0, переходим к вычислению x^2 ****

Загружаем x в стек **** Загружаем x в стек (повторно для умножения) **** Вычисляем x \* x = x^2 ****

Переход к сохранению значения **** Сохраняем вычисленное значение функции в массив по адресу edi ****

Завершаем программу

****

**Вывод:**Я изучил принципы выполнения арифметических команд с помощью математического сопроцессора FPU микропроцессоров с архитектурой x86.