ФГБОУ ВО «ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет автоматики и вычислительной техники

Кафедра радиоэлектронных средств

«ИСПОЛЬЗОВАНИЕ МАТЕМАТИЧЕСКОГО СОПРОЦЕССОРА»

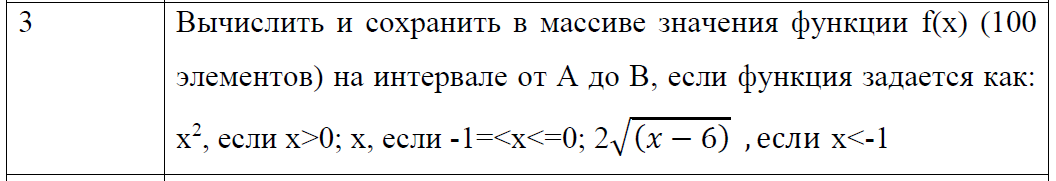
Лабораторная работа № 3

Вариант 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Выполнил: | студент группы ИНБс-33 | \_\_\_\_\_\_\_\_\_\_\_ |  | А.Ю. Бердников |
|  |  |  |  |  |
| Проверил: | доцент кафедры РЭС | \_\_\_\_\_\_\_\_\_\_\_ |  | М.А. Земцов |

Киров 2025

1. **Задание**



1. **Ход работы**

Значения x были взяты в диапазоне от -49 до 50.

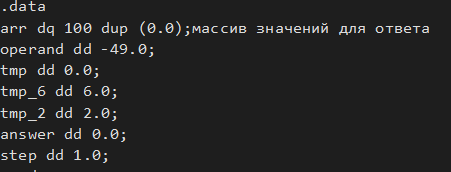


Рисунок 1 – Инициализация переменных и массива в секции данных

В начале выполнения программы помещаем 0 в регистр ecx (счётчик цикла) и инициализируем сопроцессор.

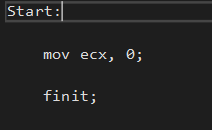


Рисунок 2 – начало выполнения программы

Помещаем в ebx -49, и прибавляем текущее значение счётчика ecx, получая число для проверки условий. Помещаем на вершину стека сначала answer (=0.0), а затем operand, который по мере выполнения программы будет изменяться от -49 до 50 (то есть наш “x”). Производим сравнение ebx.

Если ebx > 0, то переходим в метку L2;

Если ebx < -1, то переходим в метку L3;

Иначе (Если -1 <= ebx <= 0), то переходим в метку L4.

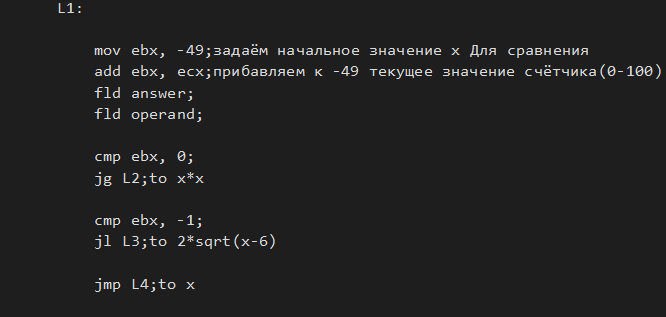
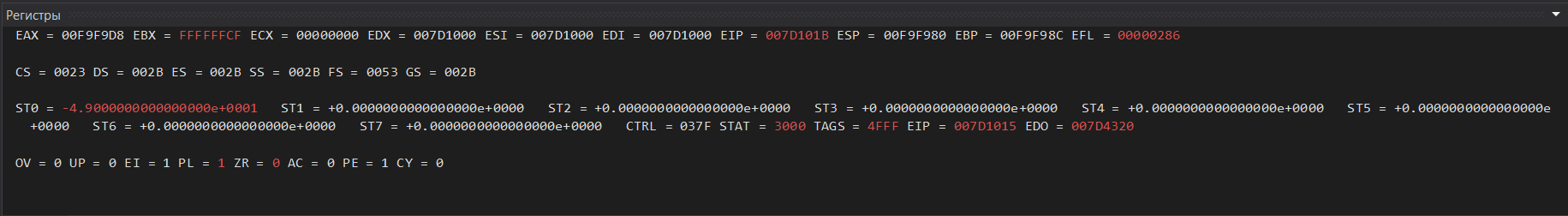
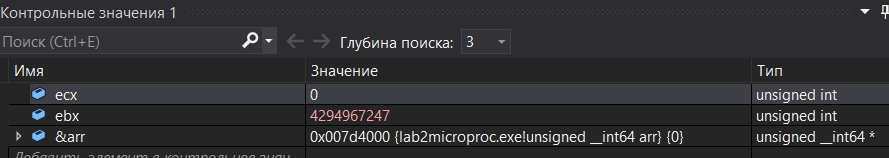


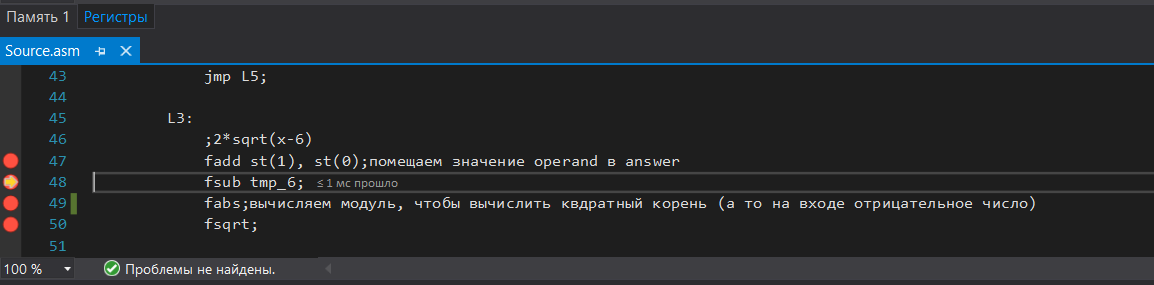
Рисунок 3 – сравнение “x” и переход к нужной метке



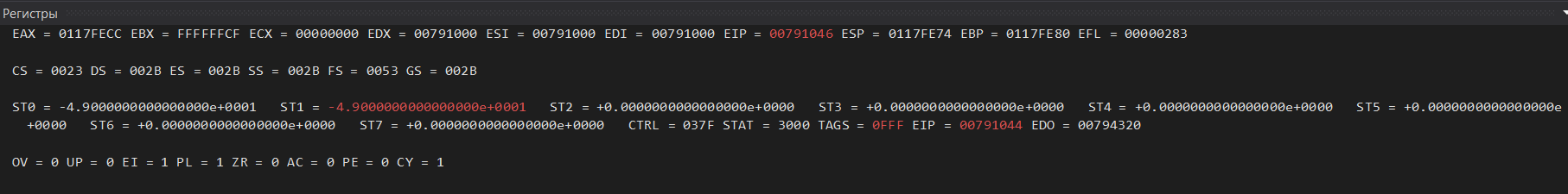


Далее разберём пошагово выполнение всех трёх условий.

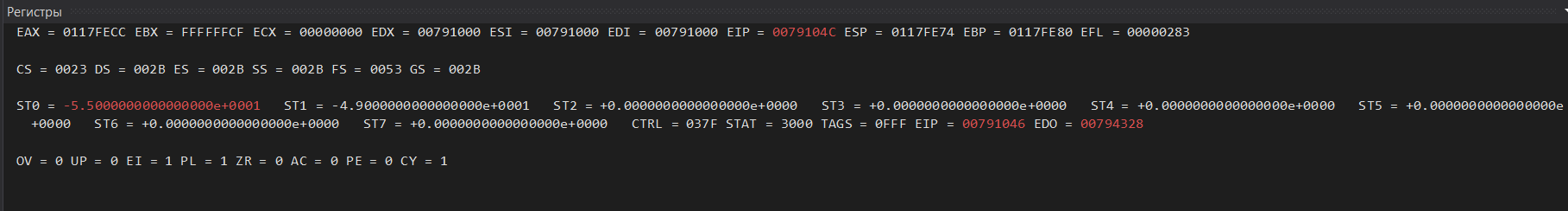
1) -49. Так как < -1. То мы переходим в метку L3.



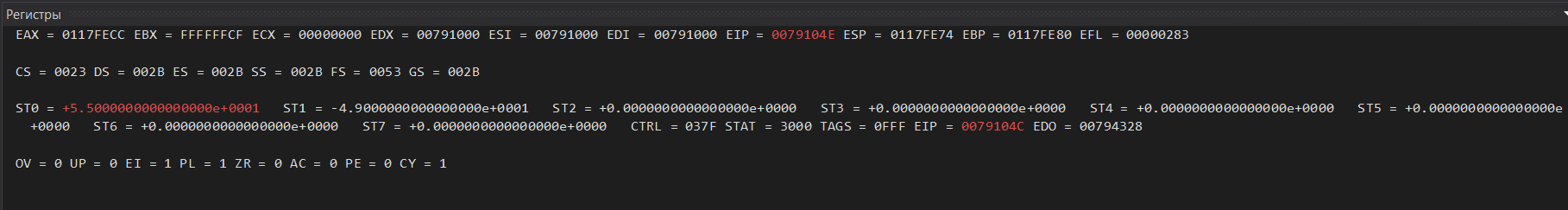
Скопировали значение из st(0), в st(1).



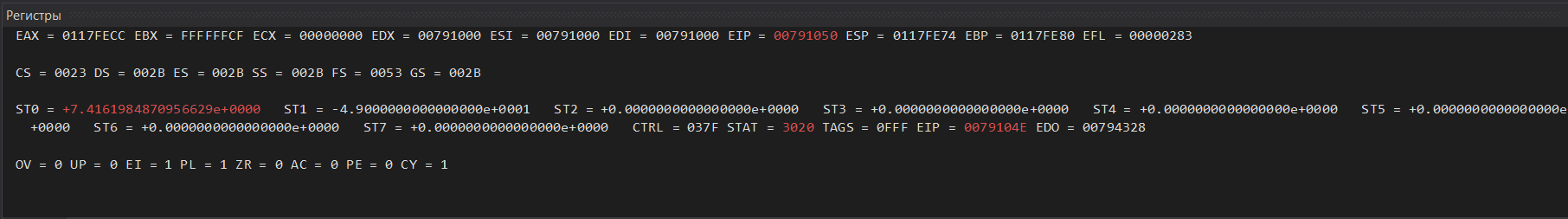
Вычли из st(0) 6 = -49 -6 = -55.



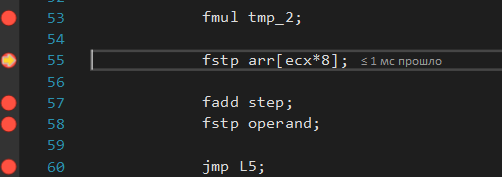
Так как все числа, попадающие в эту метку меньше -1, то есть они отрицательные, то чтобы вычислить квадратный корень нужно взять число по модулю.



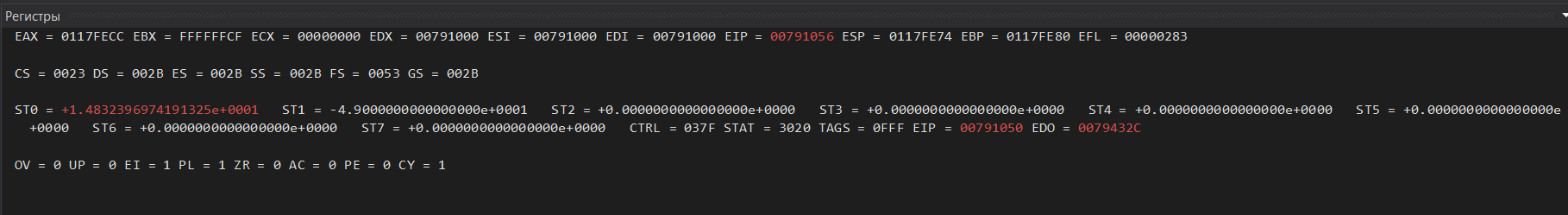
Вычисляем квадратный корень из 55 = 7.41619. Сходится с результатом программы.



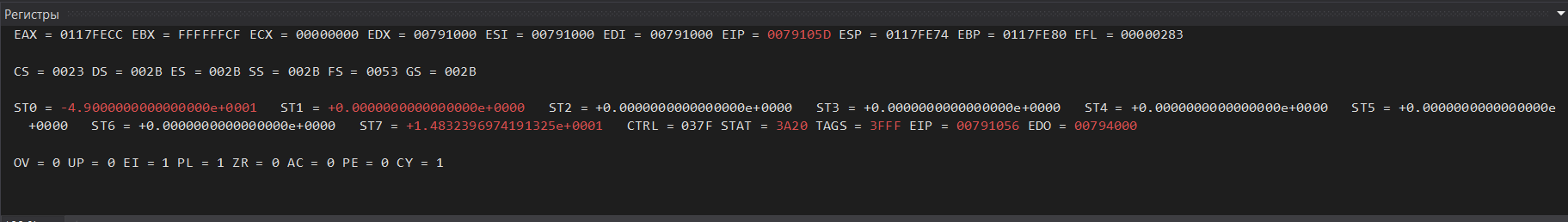
Продолжение кода метки L3.



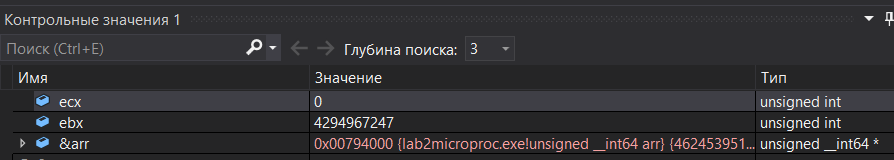
Умножаем 7.41619 на 2 = 14.832. Сходится с результатом программы.



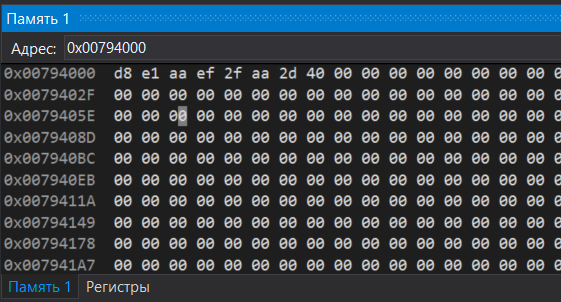
Помещаем значение st(0) в массив и снимаем его с вершины стека.



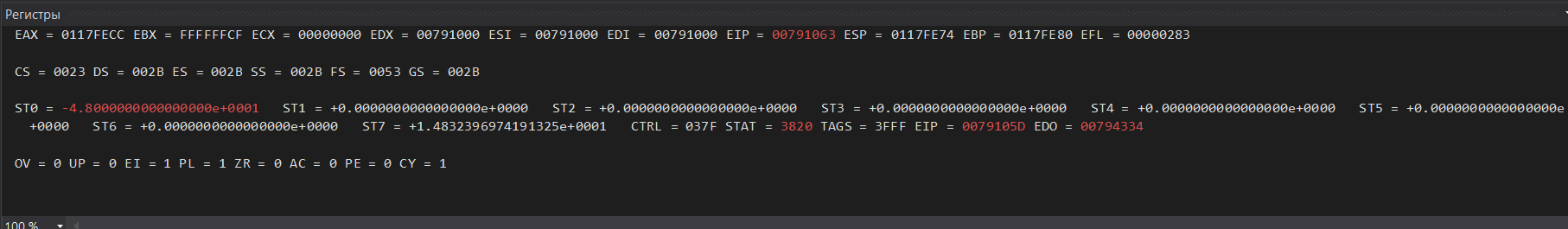
Адрес массива (&arr = 0x00794000)

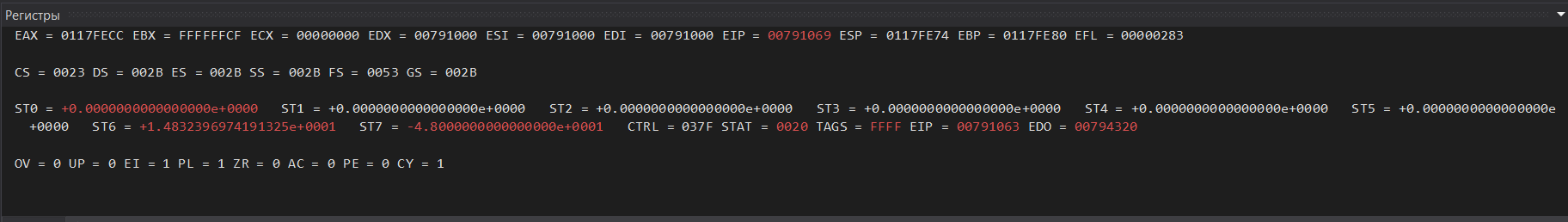


Как можно увидеть в массив записалось 8-ми байтное число d8 e1 aa ef 2f aa 2d 40.



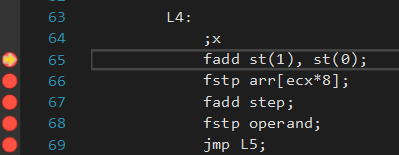
Увеличиваем на 1 значение st(0) и записываем в переменную operand, снимая значение со стека.

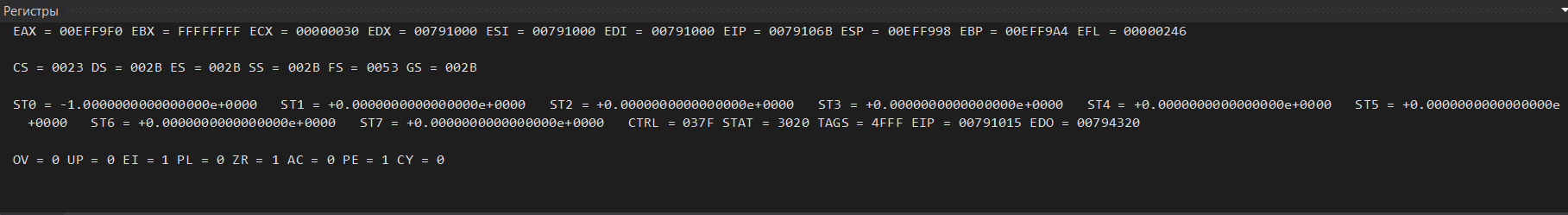


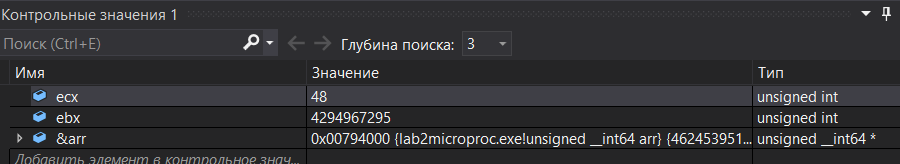


И переходим к метке L5 (код метки идёт далее).

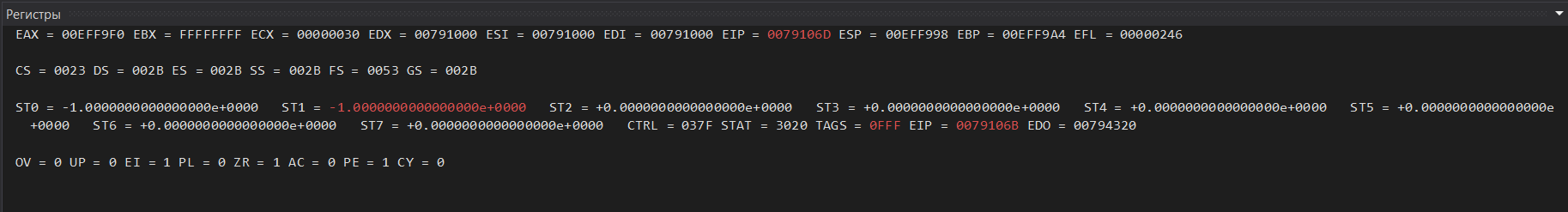
2) ebx = -1. Заходим в метку L4. st(0) = -1. ecx = 48.



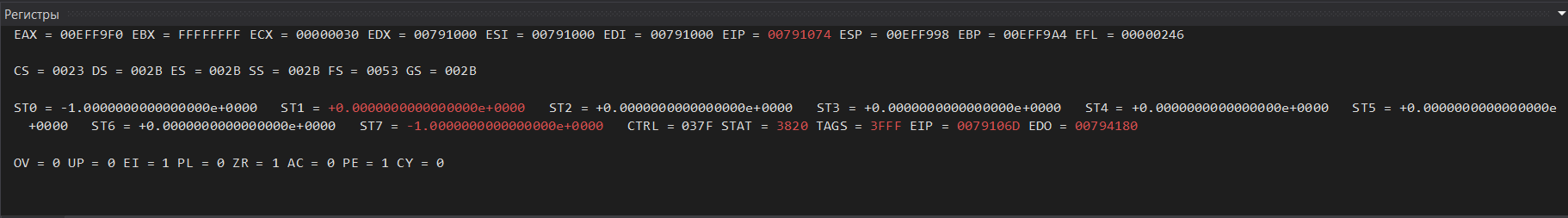




Также копируем значение из st(0) в st(1).



Помещаем значение из st(0) в массив, снимая с вершины стека.



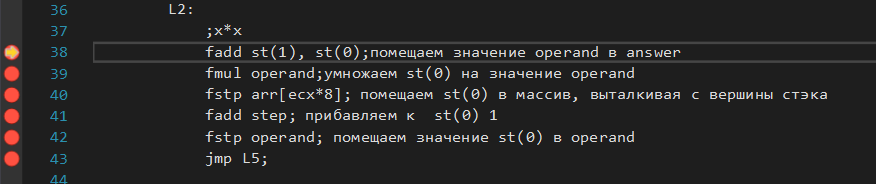


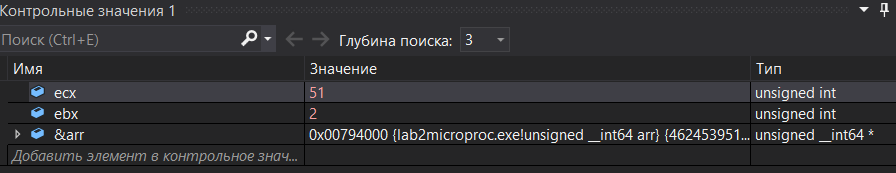
Значение 00 00 00 00 00 00 f0 bf поместилось в массив.

Увеличиваем на 1 значение st(0) и записываем в переменную operand, снимая значение со стека.

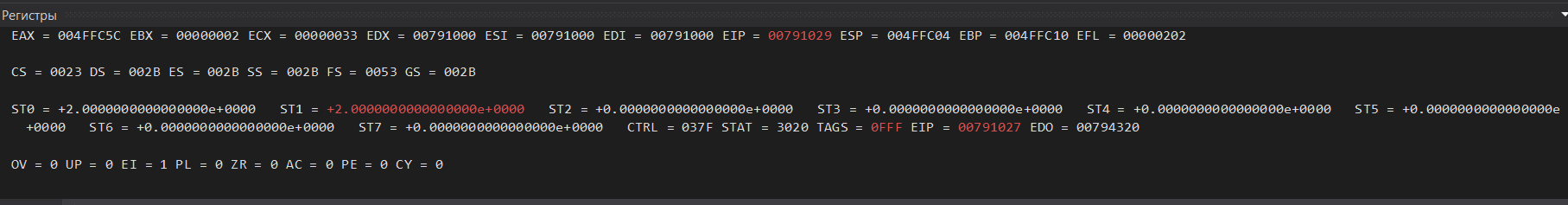
И переходим к метке L5 (код метки идёт далее).

3) ebx = 2. Заходим в метку L2. st(0) = 2. ecx = 51.

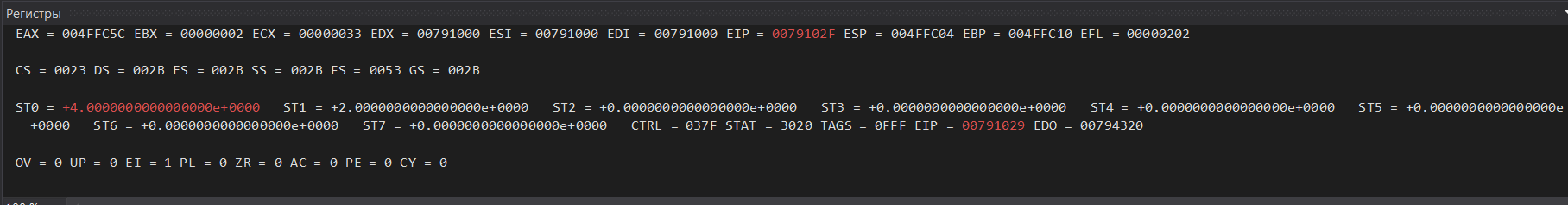




Также копируем значение из st(0) в st(1).

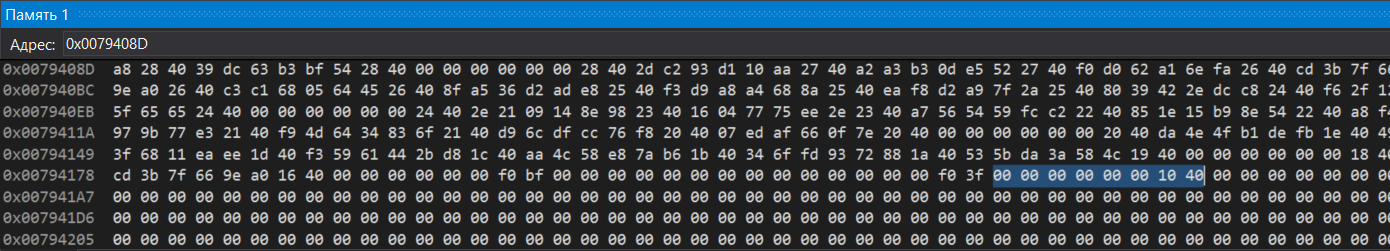


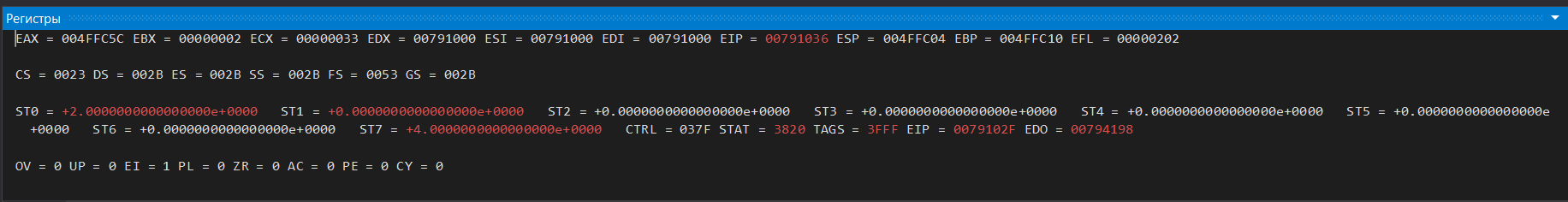
Умножаем на operand, а значения в st(0) и в operand равны, то есть возводим в квадрат. 2\*2 = 4.



Далее помещаем st(0) в массив и снимаем его со стека. Записалось число

00 00 00 00 00 00 00 10 40.

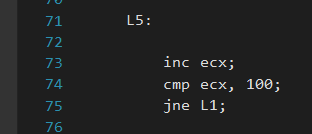




Увеличиваем на 1 значение st(0) и записываем в переменную operand, снимая значение со стека.

И переходим к метке L5 (код метки идёт далее).

Метка L5. Увеличиваем счётчик ecx на 1, сравниваем со значением 100. И если ecx == 100, то выходим из программы.



**Вывод:** все значения “x” от -49 до 50 были проверены; программа правильно делает переходы по условиям, в зависимости от “x”. Значения в сопроцессоре вычислялись верно.

**Программный код:**

.686

.model flat,stdcall

.stack 100h

.data

arr dq 100 dup (0.0);массив значений для ответа

operand dd -49.0;

tmp dd 0.0;

tmp\_6 dd 6.0;

tmp\_2 dd 2.0;

answer dd 0.0;

step dd 1.0;

.code

ExitProcess PROTO STDCALL :DWORD

Start:

mov ecx, 0;

finit;

L1:

mov ebx, -49;задаём начальное значение x Для сравнения

add ebx, ecx;прибавляем к -49 текущее значение счётчика(0-100)

fld answer;

fld operand;

cmp ebx, 0;

jg L2;to x\*x

cmp ebx, -1;

jl L3;to 2\*sqrt(x-6)

jmp L4;to x

L2:

;x\*x

fadd st(1), st(0);помещаем значение operand в answer

fmul operand;умножаем st(0) на значение operand

fstp arr[ecx\*8]; помещаем st(0) в массив, выталкивая с вершины стэка

fadd step; прибавляем к st(0) 1

fstp operand; помещаем значение st(0) в operand

jmp L5;

L3:

;2\*sqrt(x-6)

fadd st(1), st(0);помещаем значение operand в answer

fsub tmp\_6;

fabs;вычисляем модуль, чтобы вычислить квдратный корень (а то на входе отрицательное число)

fsqrt;

fmul tmp\_2;

fstp arr[ecx\*8];

fadd step;

fstp operand;

jmp L5;

L4:

;x

fadd st(1), st(0);

fstp arr[ecx\*8];

fadd step;

fstp operand;

jmp L5;

L5:

inc ecx;

cmp ecx, 100;

jne L1;

exit:

Invoke ExitProcess,1

End Start