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 №2

по дисциплине «Цифровые устройства и микропроцессоры»

“ ПРИНЦИПЫ ВЫПОЛНЕНИЯ КОМАНД ВЕТВЛЕНИЯ, ОРГАНИЗАЦИЯ

ЦИКЛОВ И ПОДПРОГРАММ”

Вариант 13

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

г. Киров

2025

**Цель работы:** изучение принципов выполнения команд ветвления, организации циклов и подпрограмм микропроцессоров с архитектурой x86.

**Выполнение индивидуального задания**

Дано: X=16E6 Y=D45B Z=C9C9 (расположены в памяти один за другим)

**ЗАДАНИЕ № 13**

В цикле увеличить на три старшие байты X, Y, Z (результат X',Y',Z')

Вычислить M=X'-Y'+Z' Если биты 2, 9, 10, 6 = 0, то переход к п/п 1 (R=(M[8,10]=1)), иначе переход к п/п 2 (R=M\*2). Если R != 4341, то переход к АДР1 (R or ~Z') иначе переход к АДР2 (R + 66) {R - число со знаком}

**Текст программы:**

.686

.model flat, stdcall

.stack 100h

.data

AR DW 16E6h, 54363, 51657 ; Определяем массив с шестнадцатеричными значениями

M dw ?

R dw 0

num\_values equ ( ($ - AR) / 2 ) ; Количество элементов в массиве

.code

ExitProcess PROTO STDCALL :DWORD

Start:

mov ecx, num\_values

mov esi, OFFSET AR

loop\_start:

mov ax, [esi]

add ah, 3

mov [esi], ax

add esi, 2

loop loop\_start

mov bx, [AR] ; X

mov cx, [AR+2] ; Y

mov dx, [AR+4] ; Z

mov M, bx ; M = X'

sub M, cx ; M = X' - Y'

add M, dx ; M = M + Z'

mov ax, M

test ax, 4 ; Проверяем бит 2

jnz punkt2 ; Переход к пункту 2, если бит 2 = 1

test ax, 40h ; Проверяем бит 6

jnz punkt2 ; Переход к пункту 2, если бит 6 = 1

test ax, 200h ; Проверяем бит 9

jnz punkt2 ; Переход к пункту 2, если бит 9 = 1

test ax, 400h ; Проверяем бит 10

jnz punkt2 ; Переход к пункту 2, если бит 10 = 1

mov ax, 0 ; Обнуляем AX

or ax, 1024 ; Устанавливаем 10-й бит (1024)

or ax, 256 ; Устанавливаем 8-й бит (256)

mov [R], ax ; Записываем в R (R = M[8,10]=1)

jmp check\_R ; Переходим к проверке R

punkt2:

mov ax, M ; Загружаем M снова для умножения

shl ax, 1 ; Умножаем M на 2

mov [R], ax ; Сохраняем результат в R

check\_R:

cmp [R], 4341 ; Сравниваем R с 4341

jne adr1 ; Если R != 4341, переход к ADR1

add [R], 66 ; R = R + 66

jmp done ; Переходим к завершению

adr1:

; Выполняем R or ~Z'

mov ax, [AR+4] ; Загружаем Z

not ax ; Инвертируем Z

or ax, [R] ; Выполняем R or ~Z'

mov [R], ax ; Сохраняем результат в R

done:

; Успешный выход

invoke ExitProcess, 0

End Start

Шаги программы при выполнении:

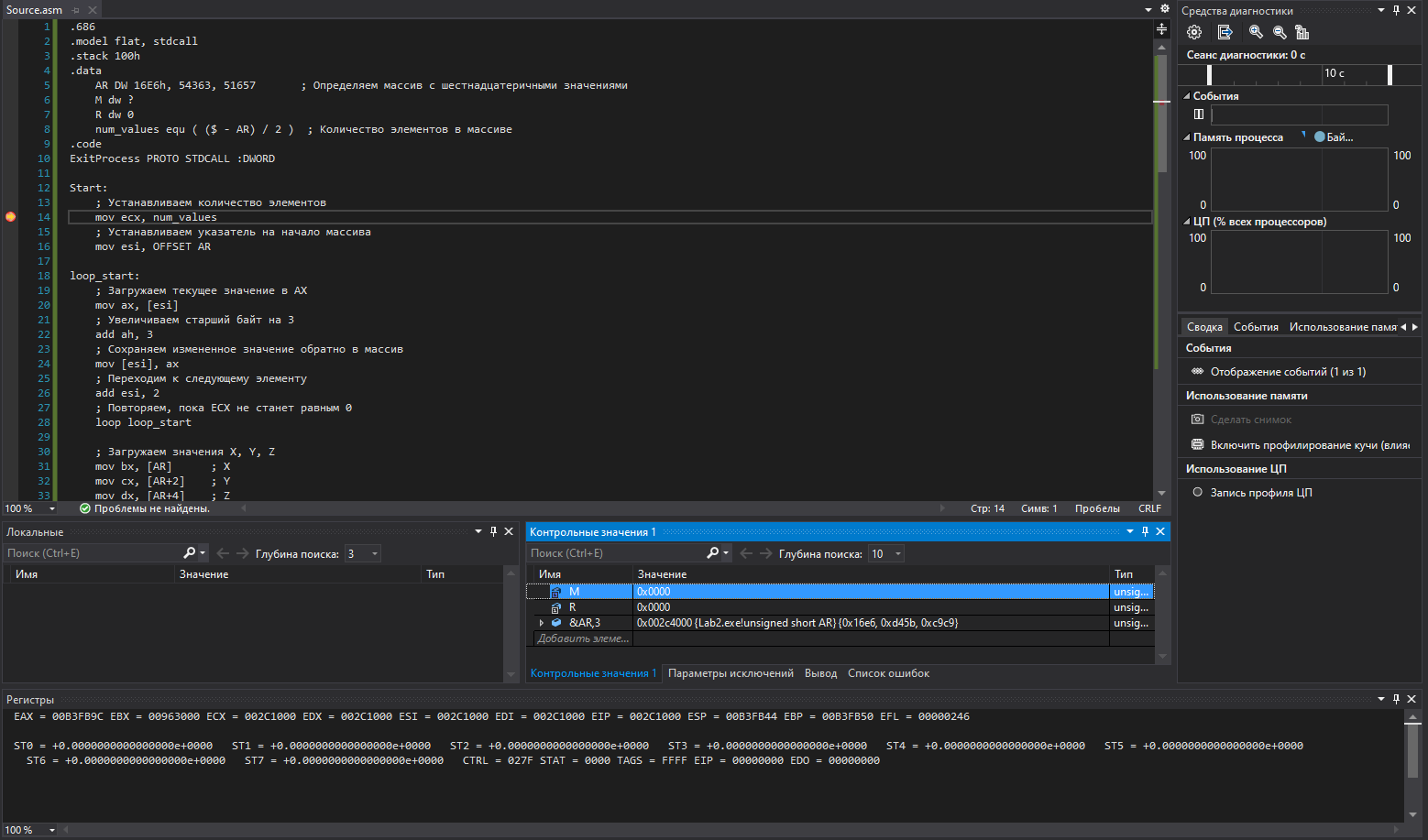


Рисунок 1 – Устанавливаем количество элементов

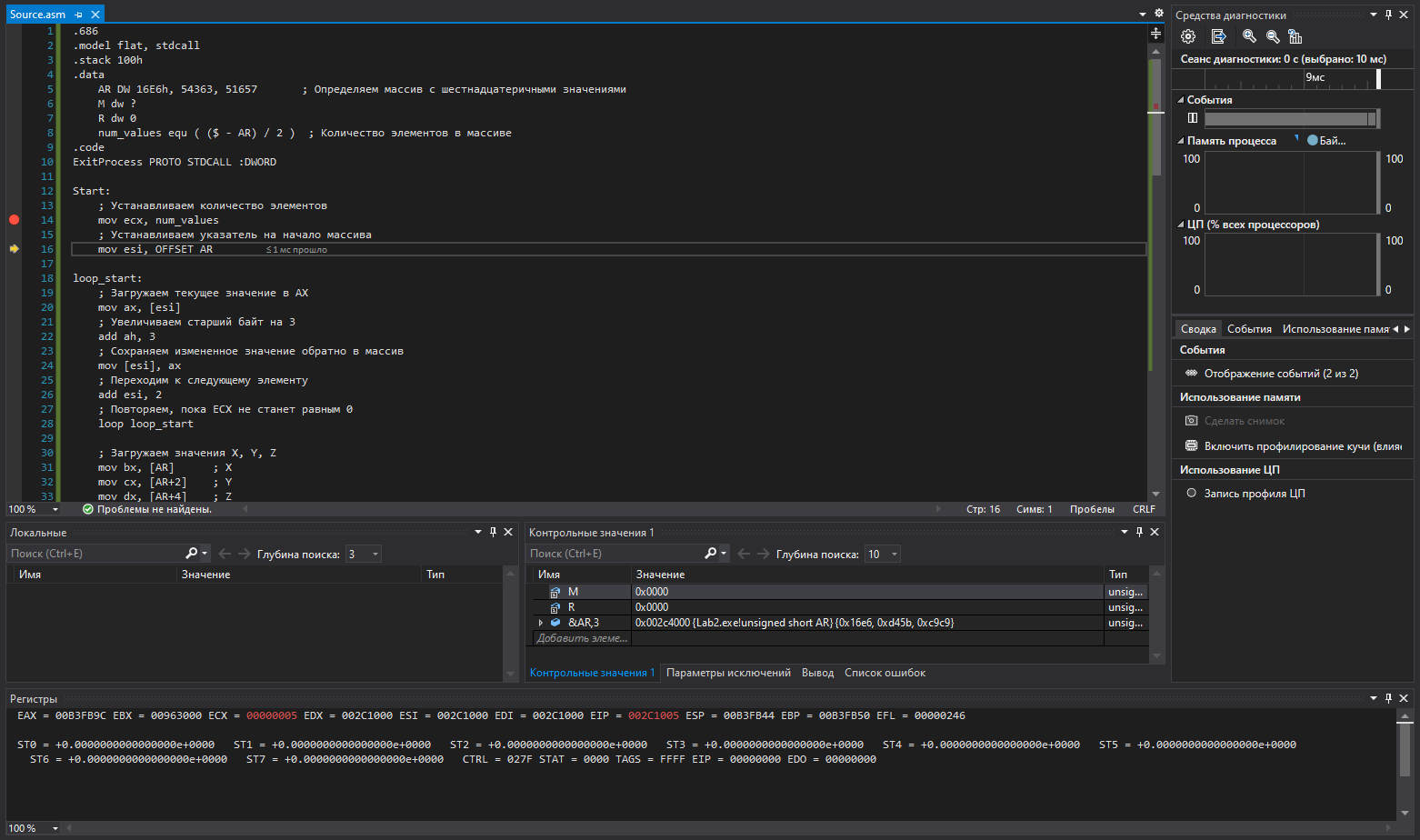


Рисунок 2 – Устанавливаем указатель на начало массива

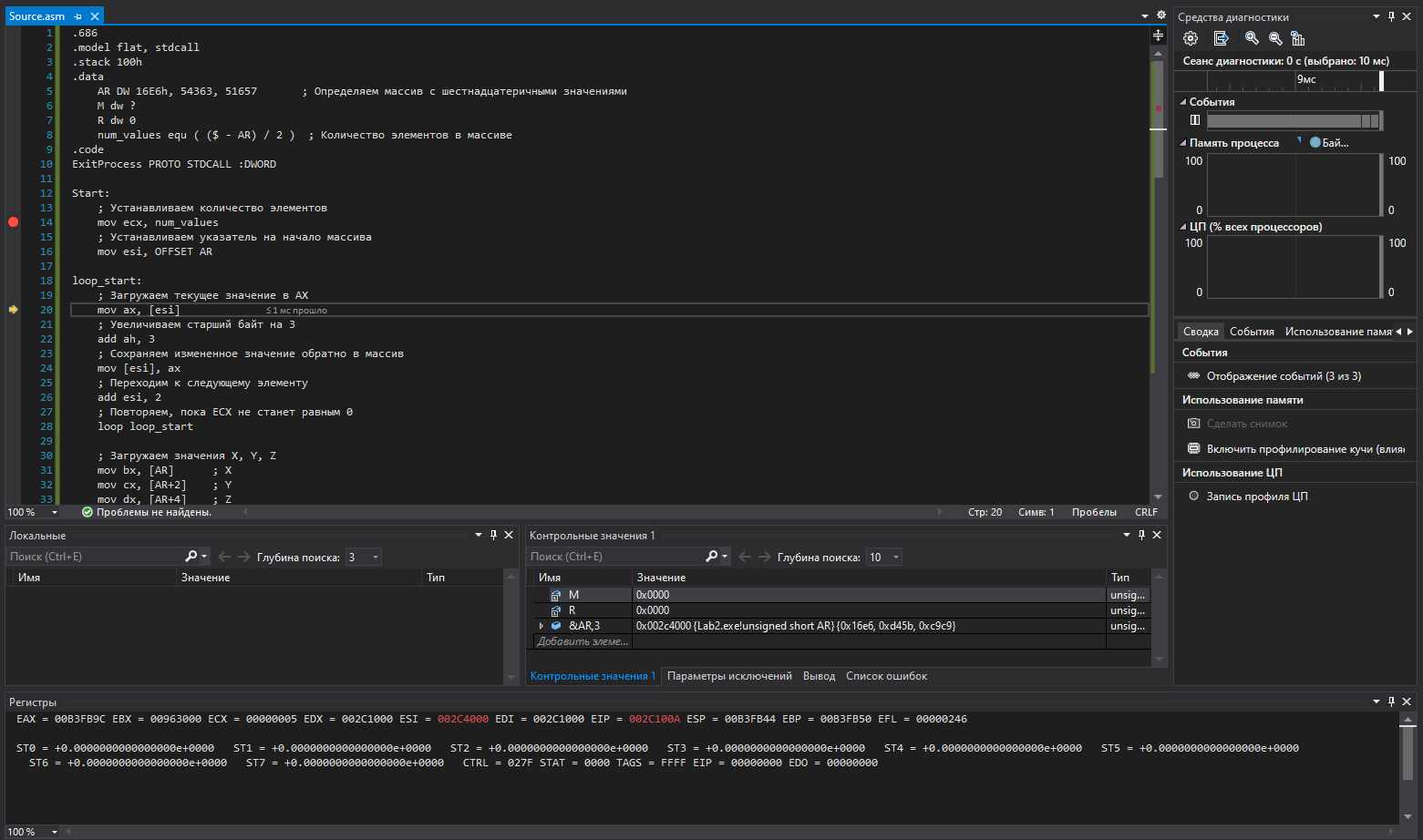


Рисунок 3 – Загружаем текущее значение в AX

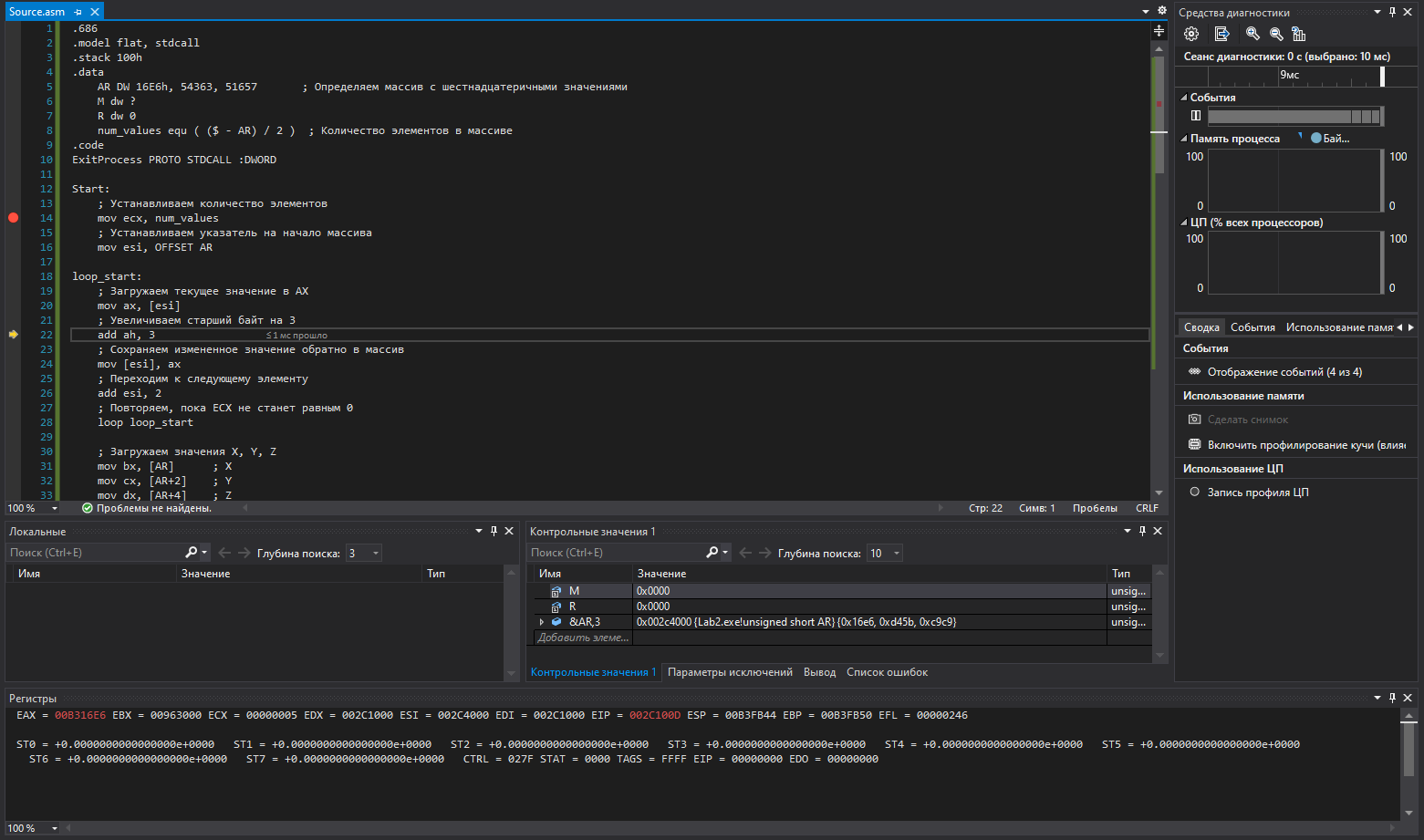


Рисунок 4 – Увеличиваем старший байт на 3

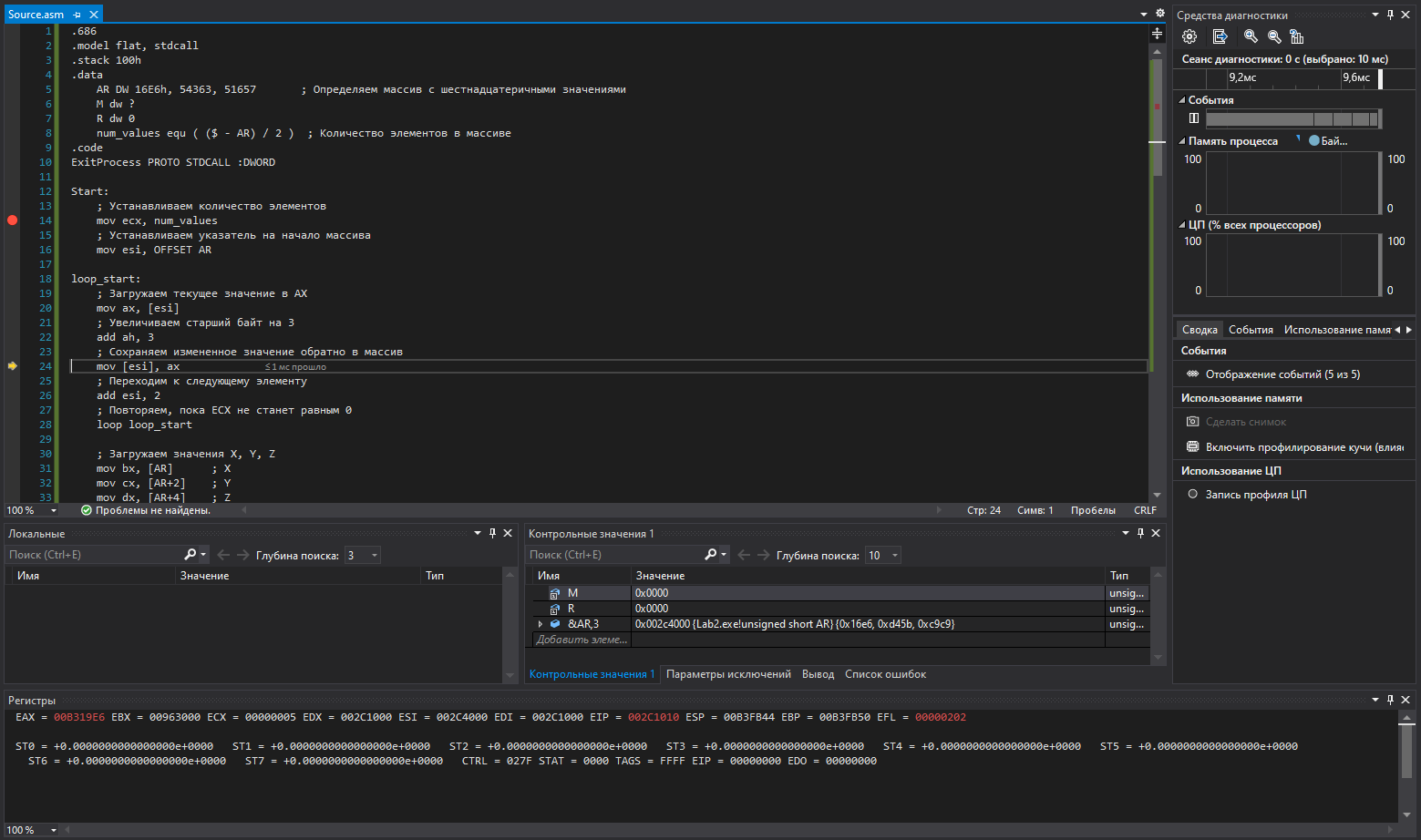


Рисунок 5 – Сохраняем измененное значение обратно в массив

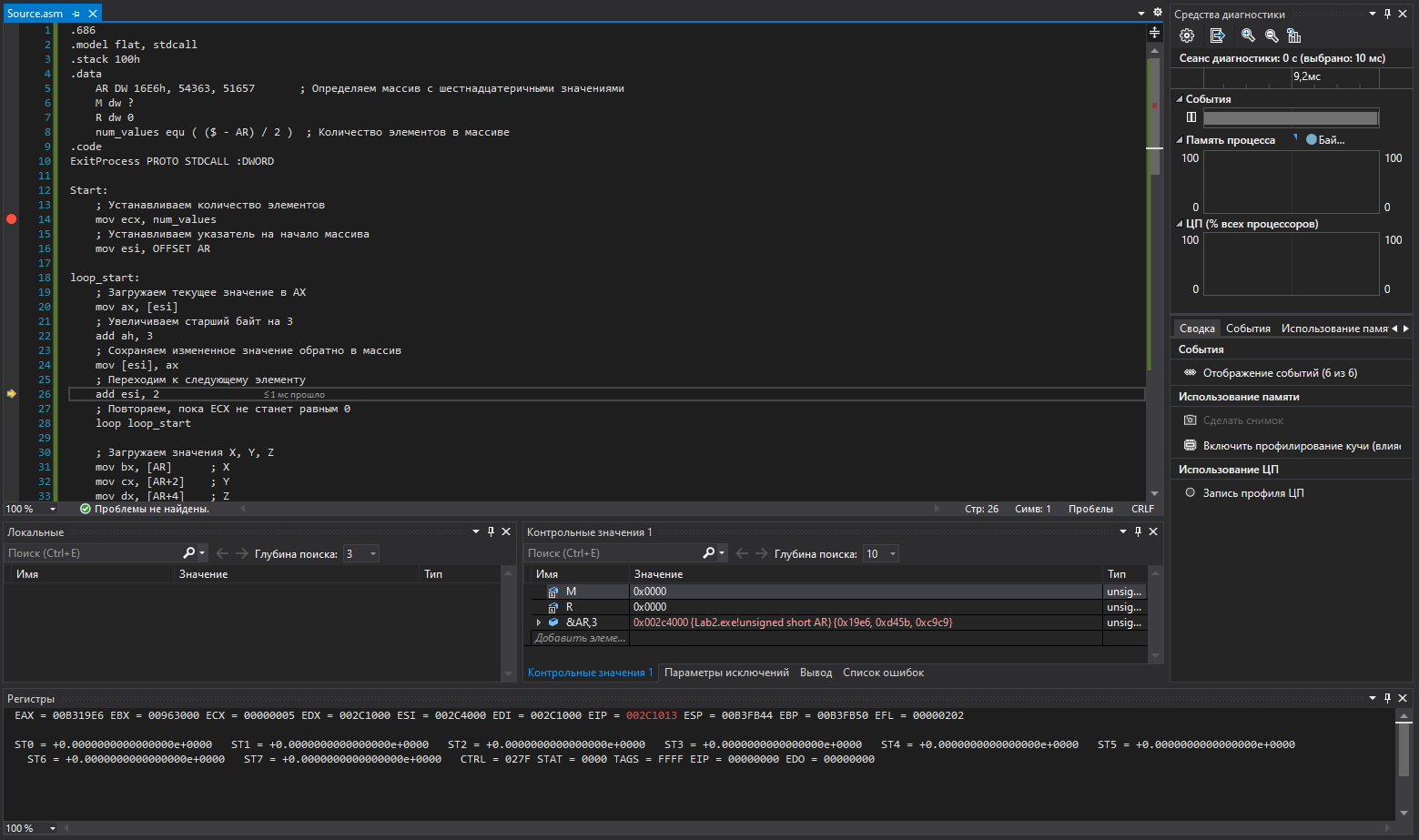


Рисунок 6 – Переходим к следующему элементу

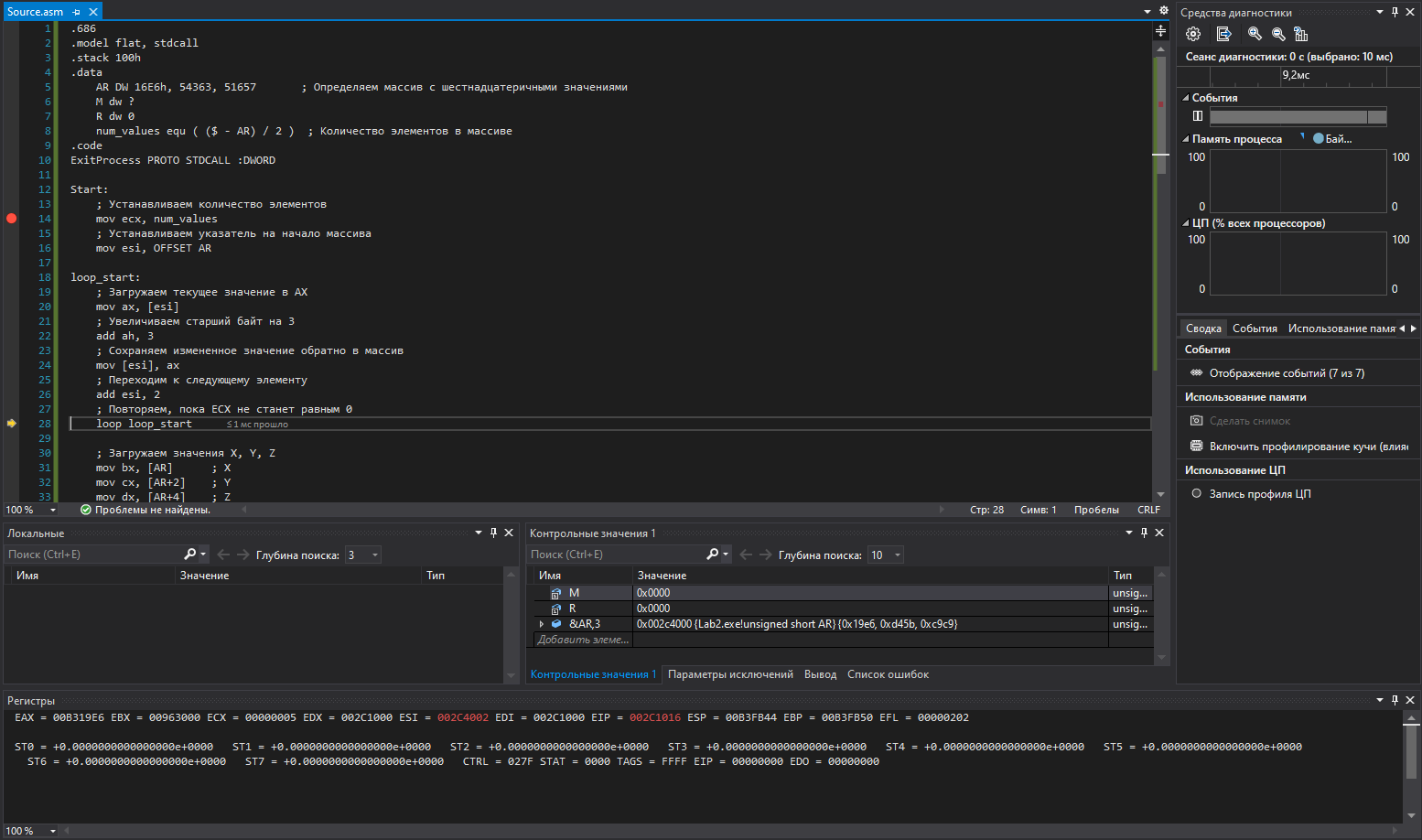


Рисунок 7 – Повторяем, пока ECX не станет равным 0

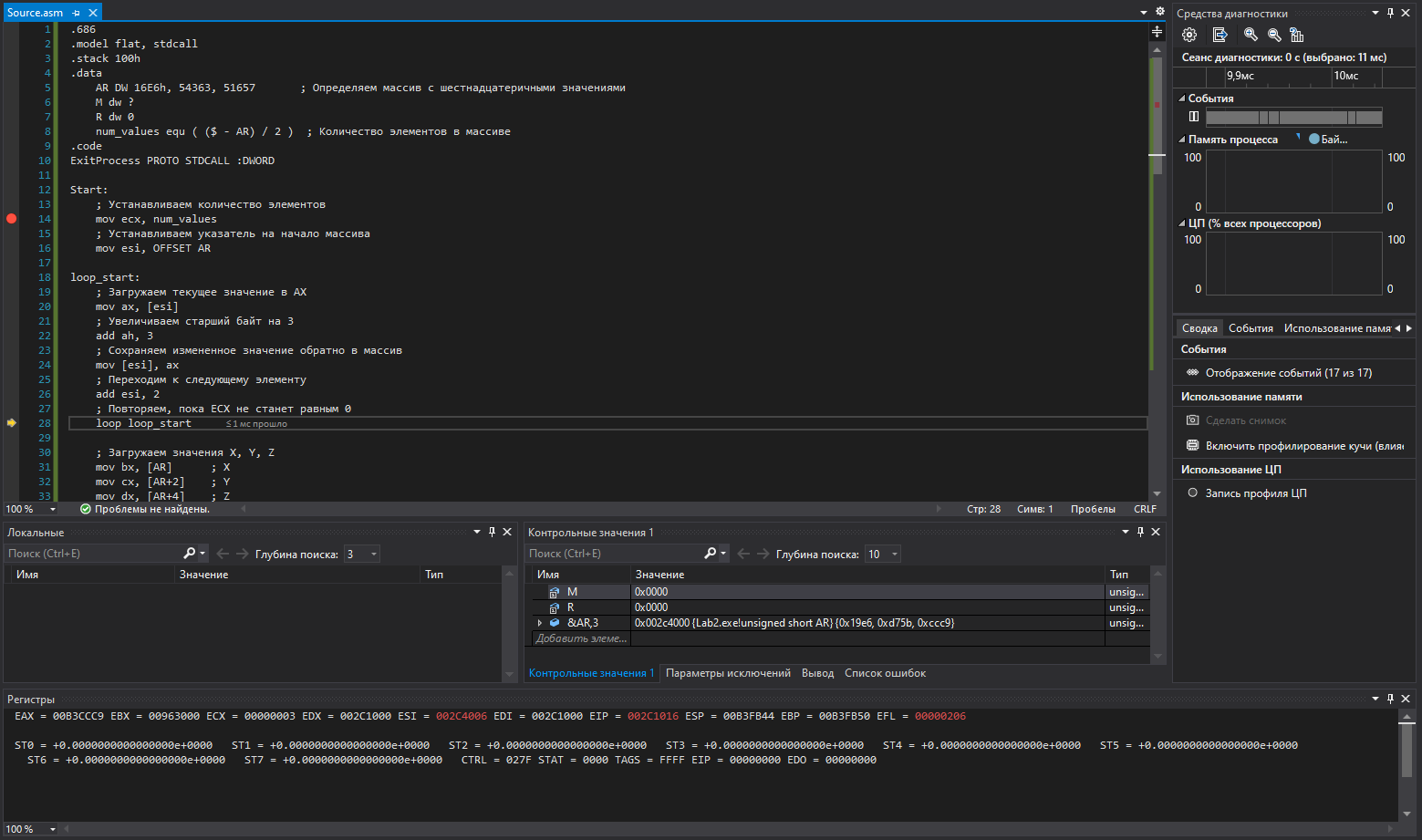


Рисунок 8 – Успешная обработка массива

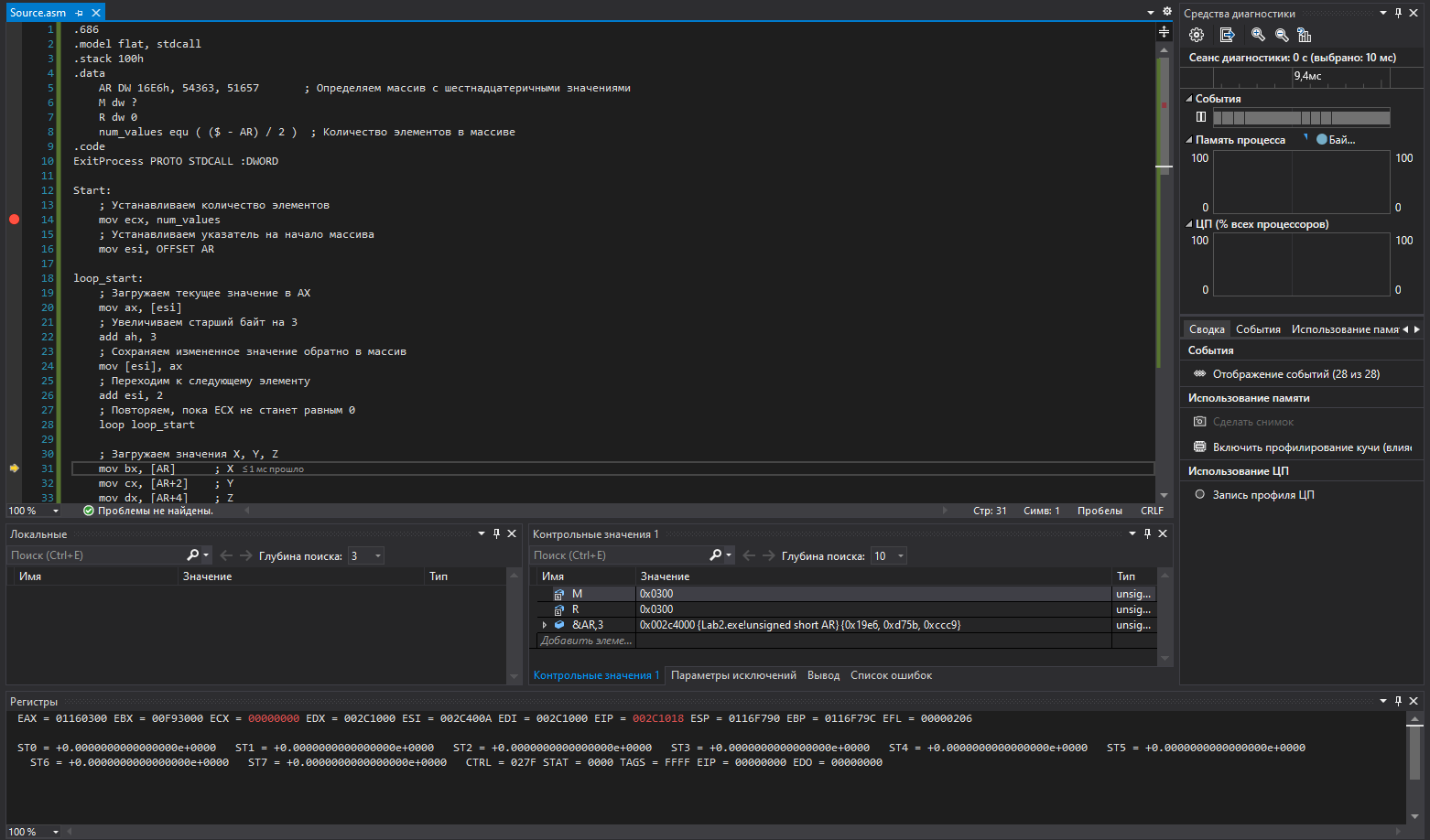


Рисунок 9 – Загрузка X в bx

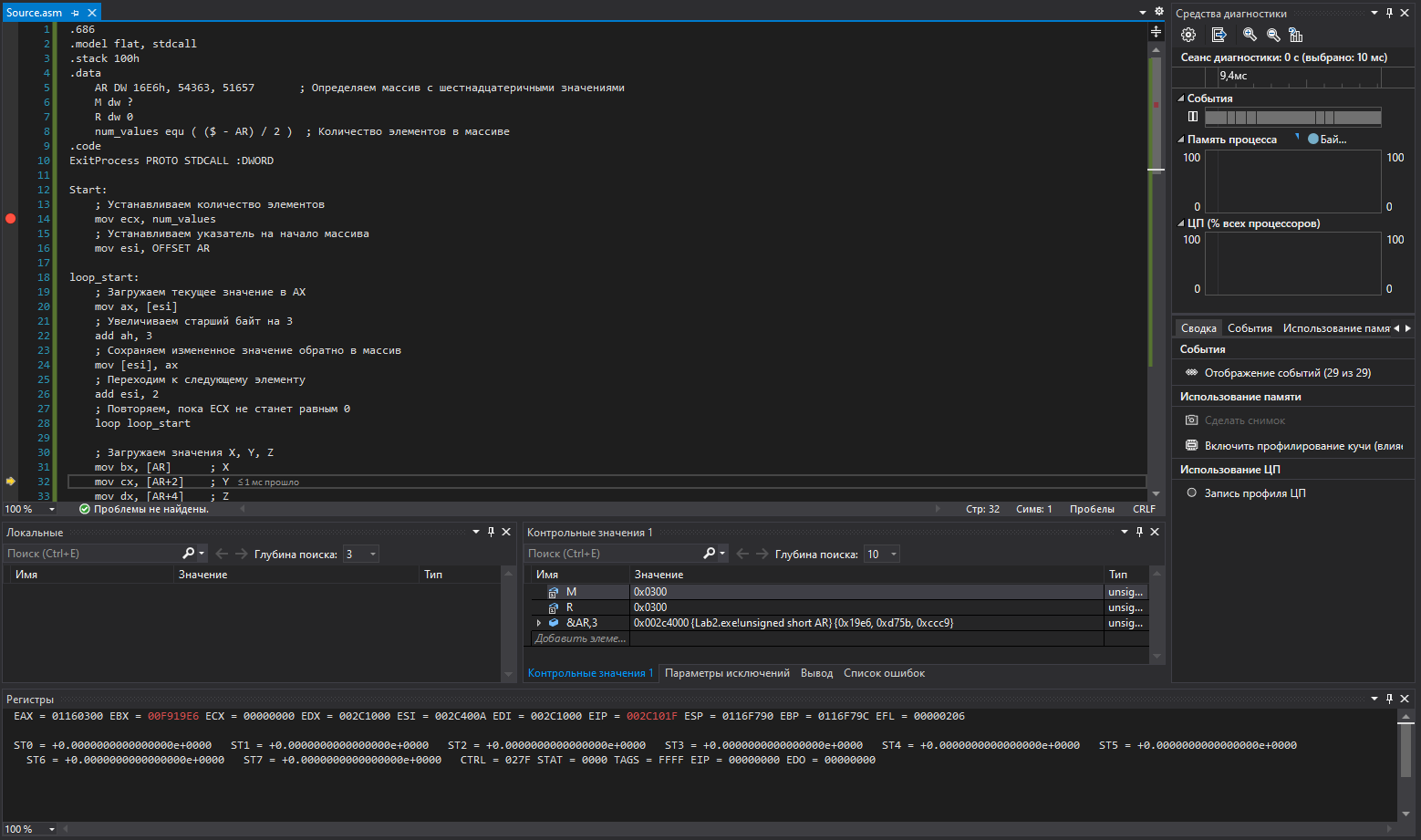


Рисунок 10 – Загрузка Y в cx

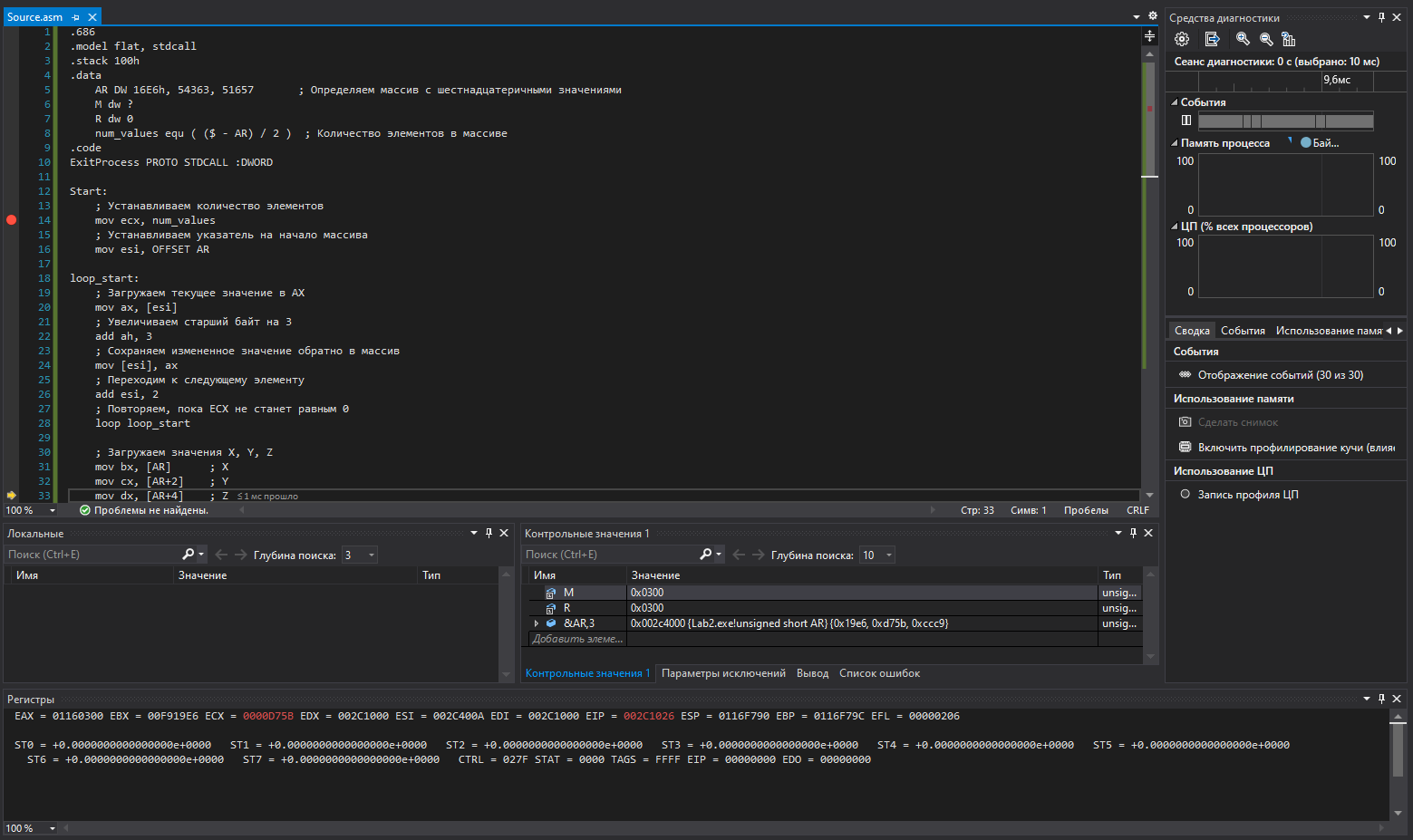


Рисунок 11 – Загрузка Z в dx

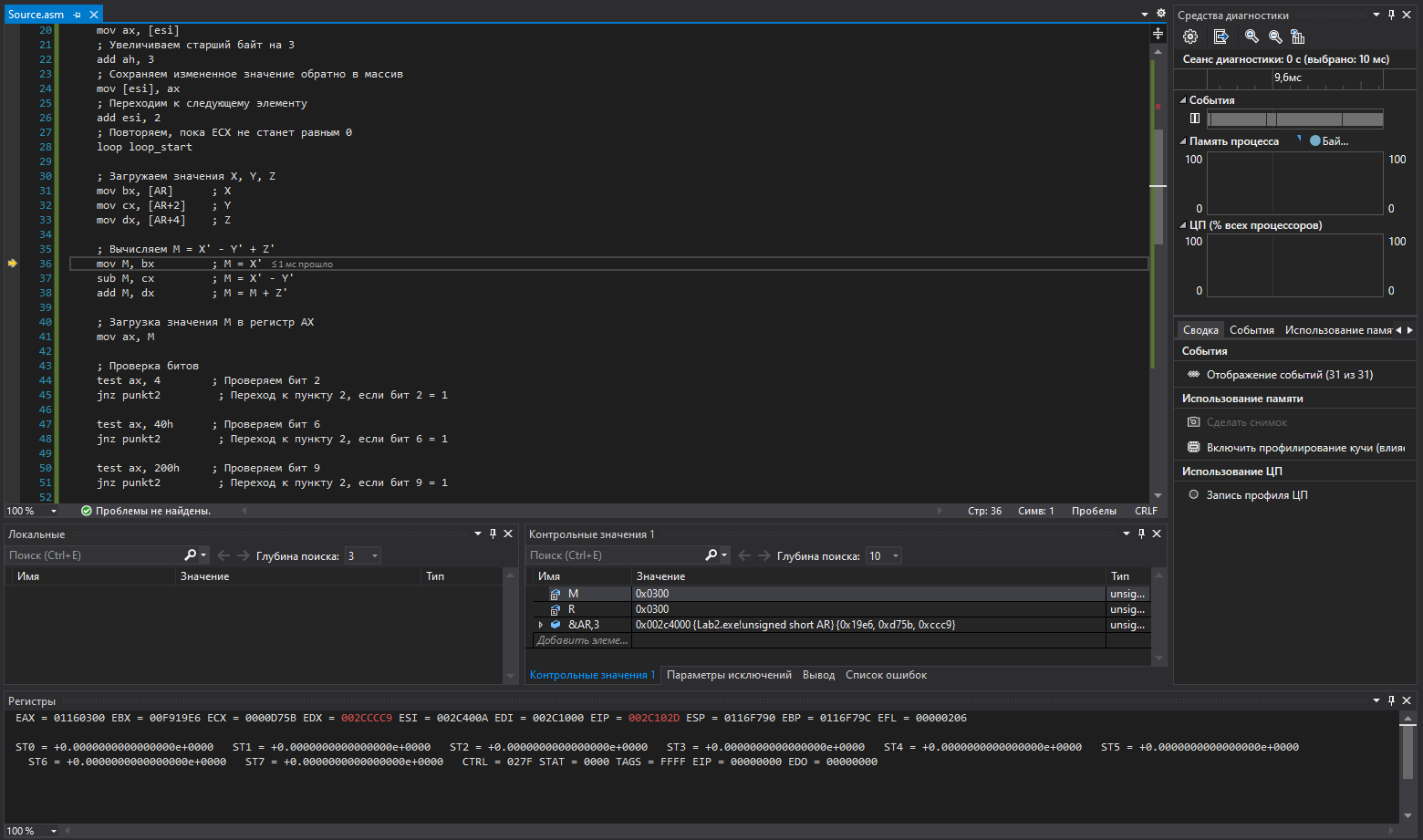


Рисунок 12 – Перенос в M значения X’

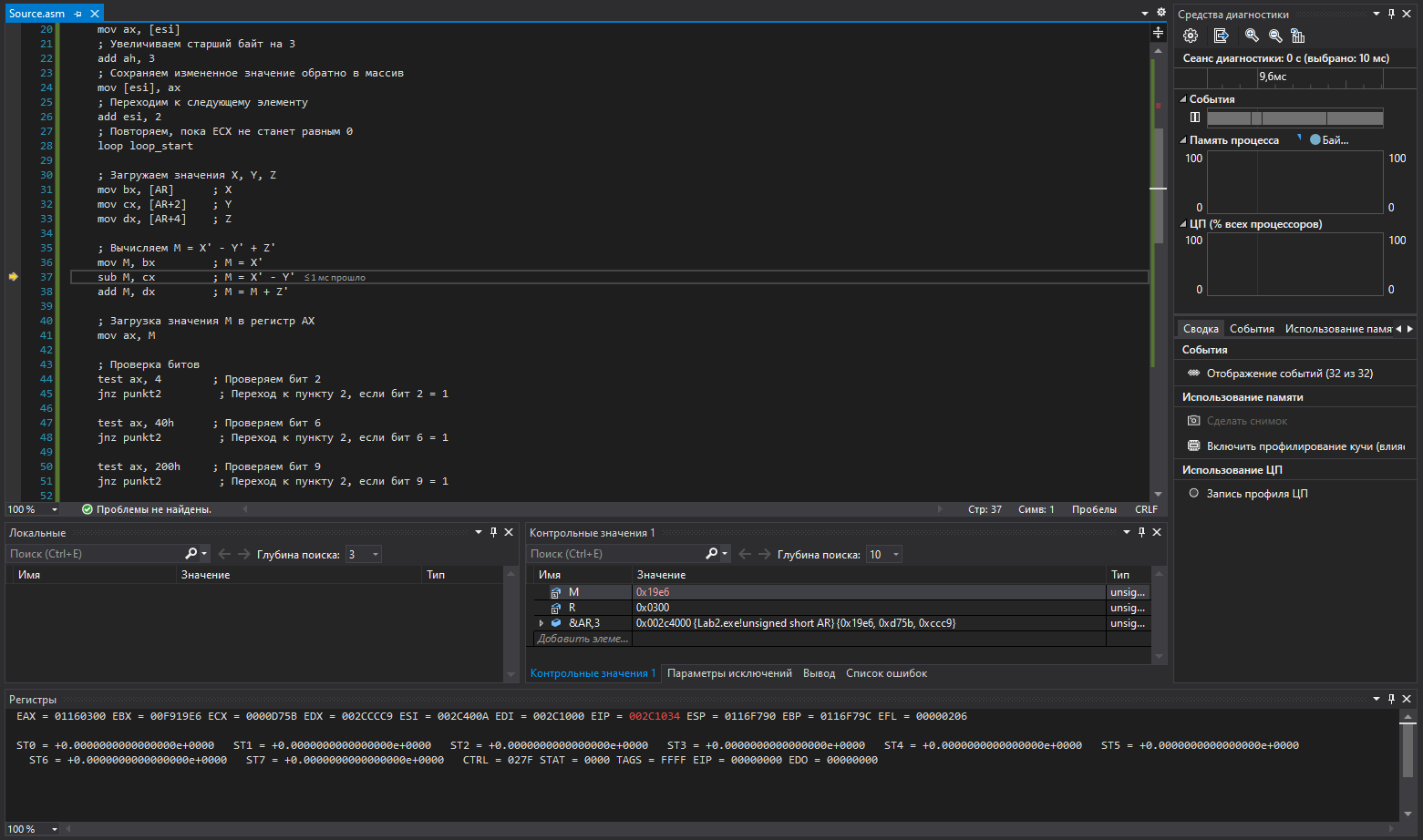


Рисунок 13 – Вычитание Y’ из M

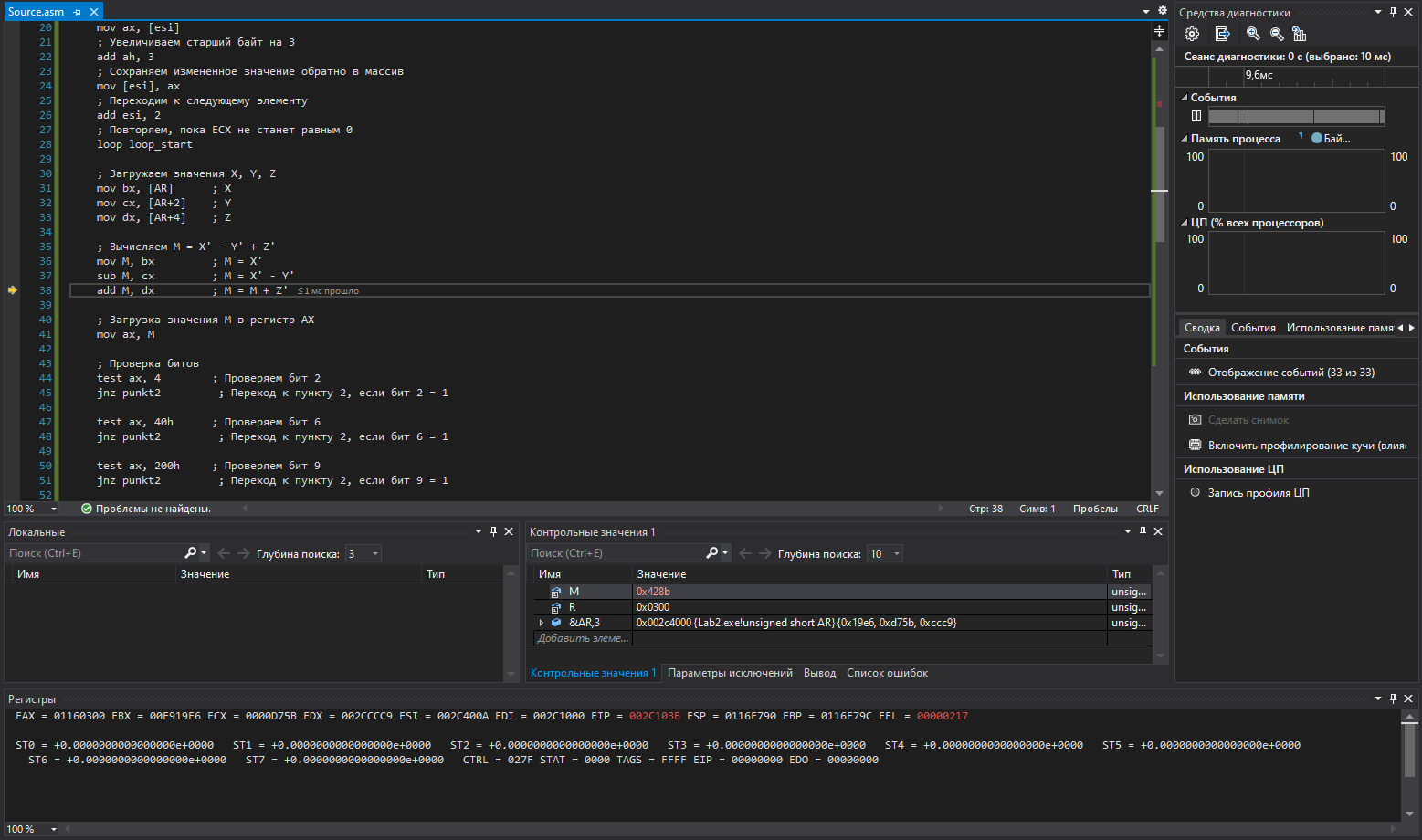


Рисунок 14 – Добавление Z’ к M

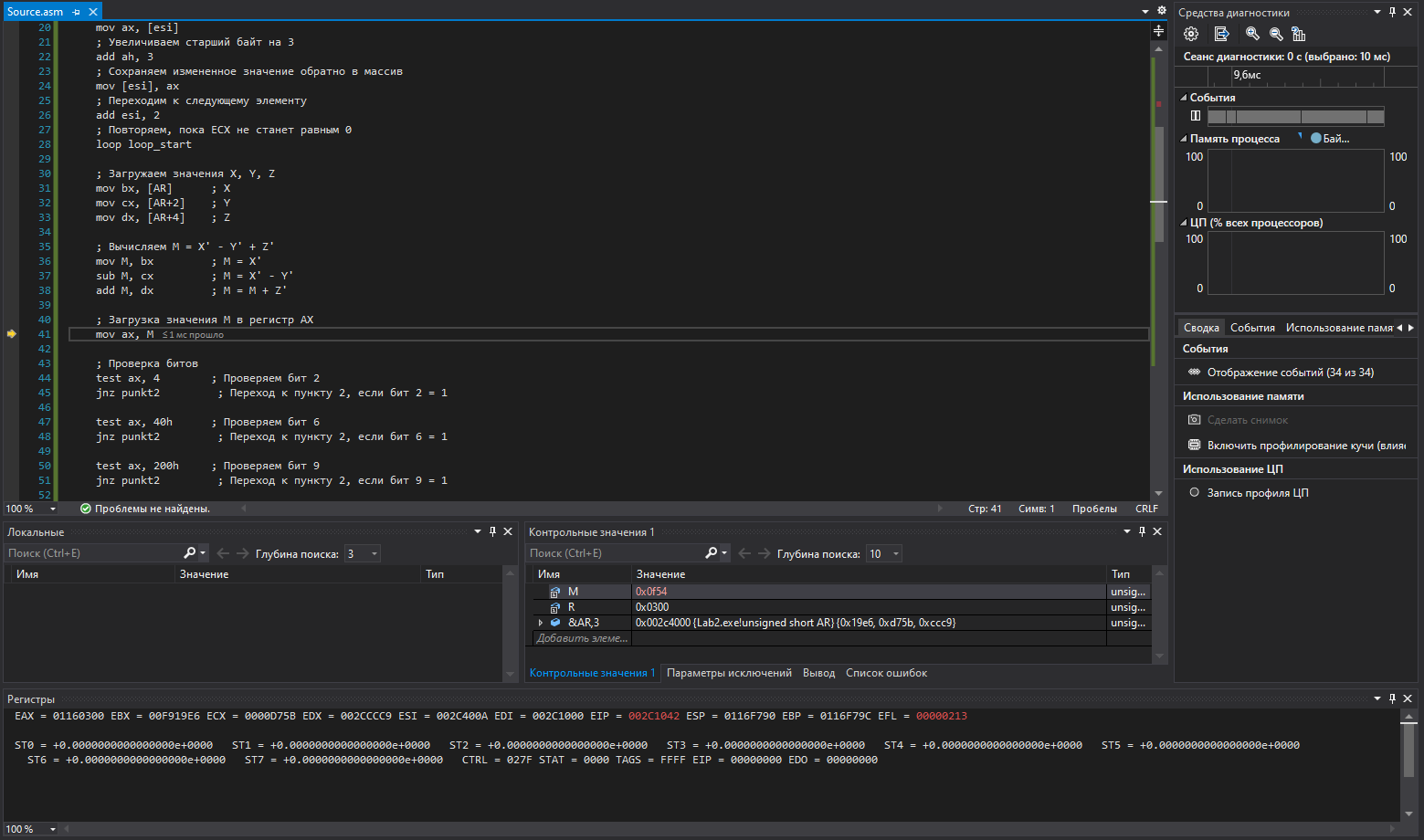


Рисунок 15 – Загрузка M в ax

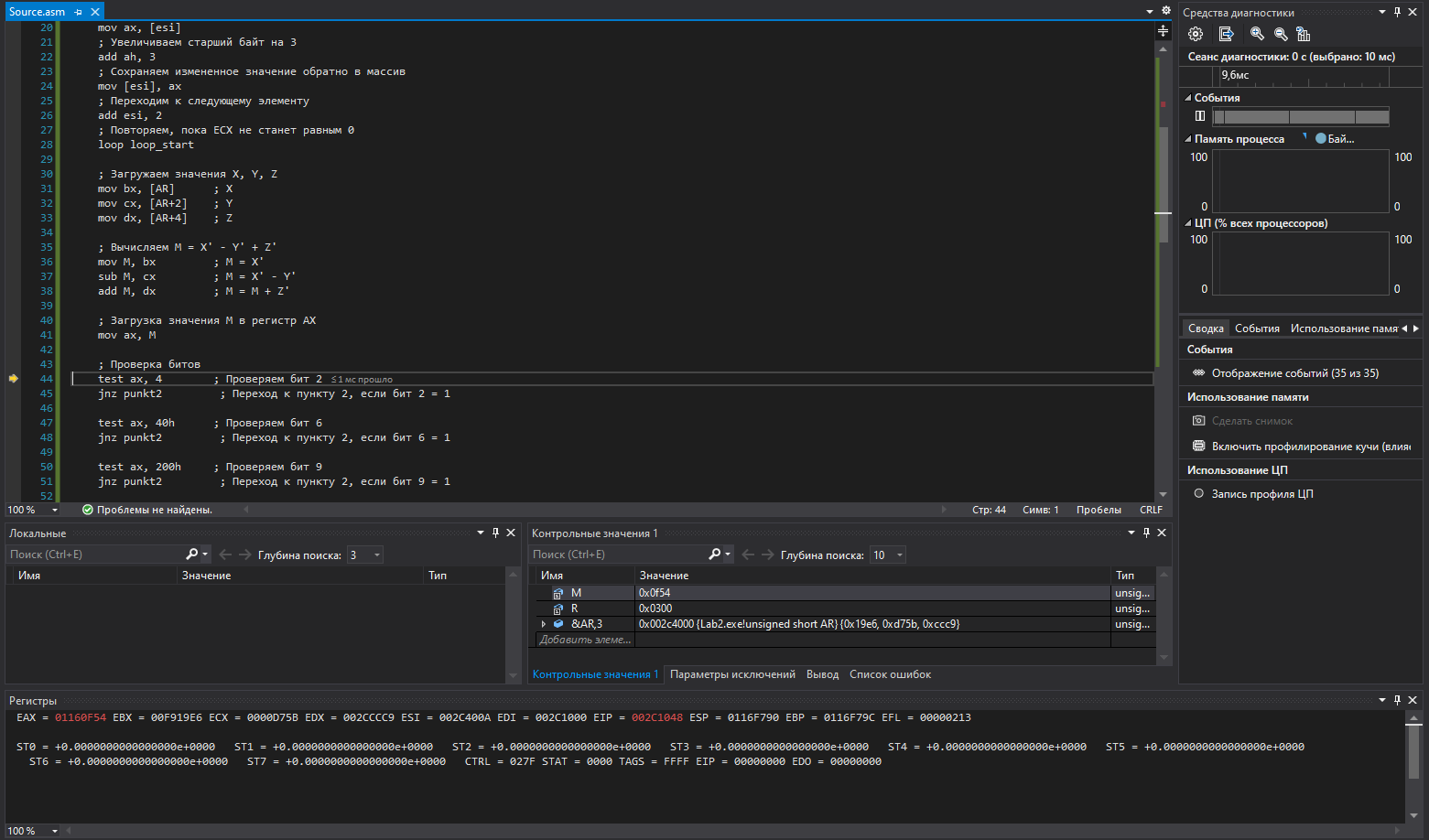


Рисунок 16 – Проверка 2 бита

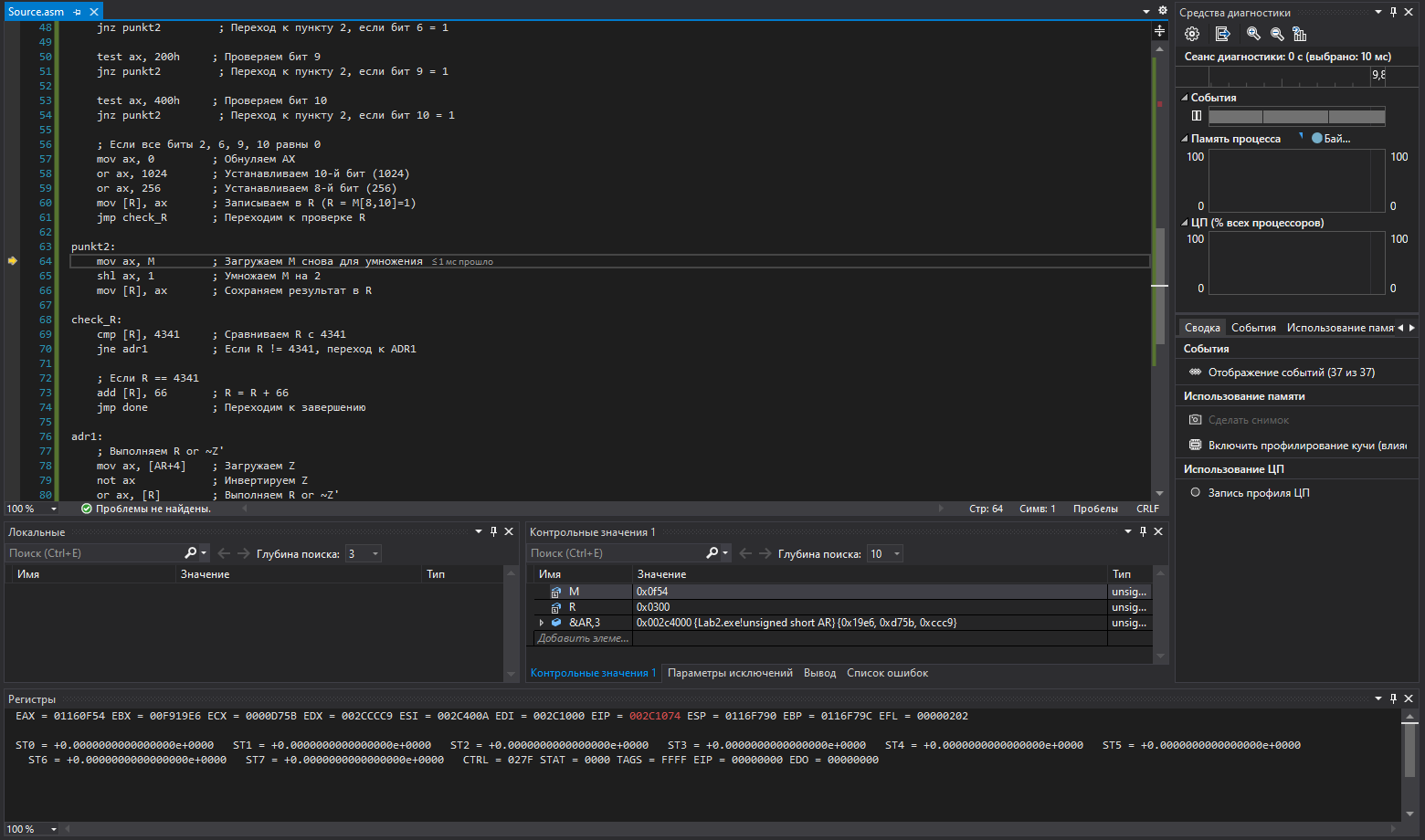


Рисунок 17 – Переход к punkt2, загружаем M снова для умножения

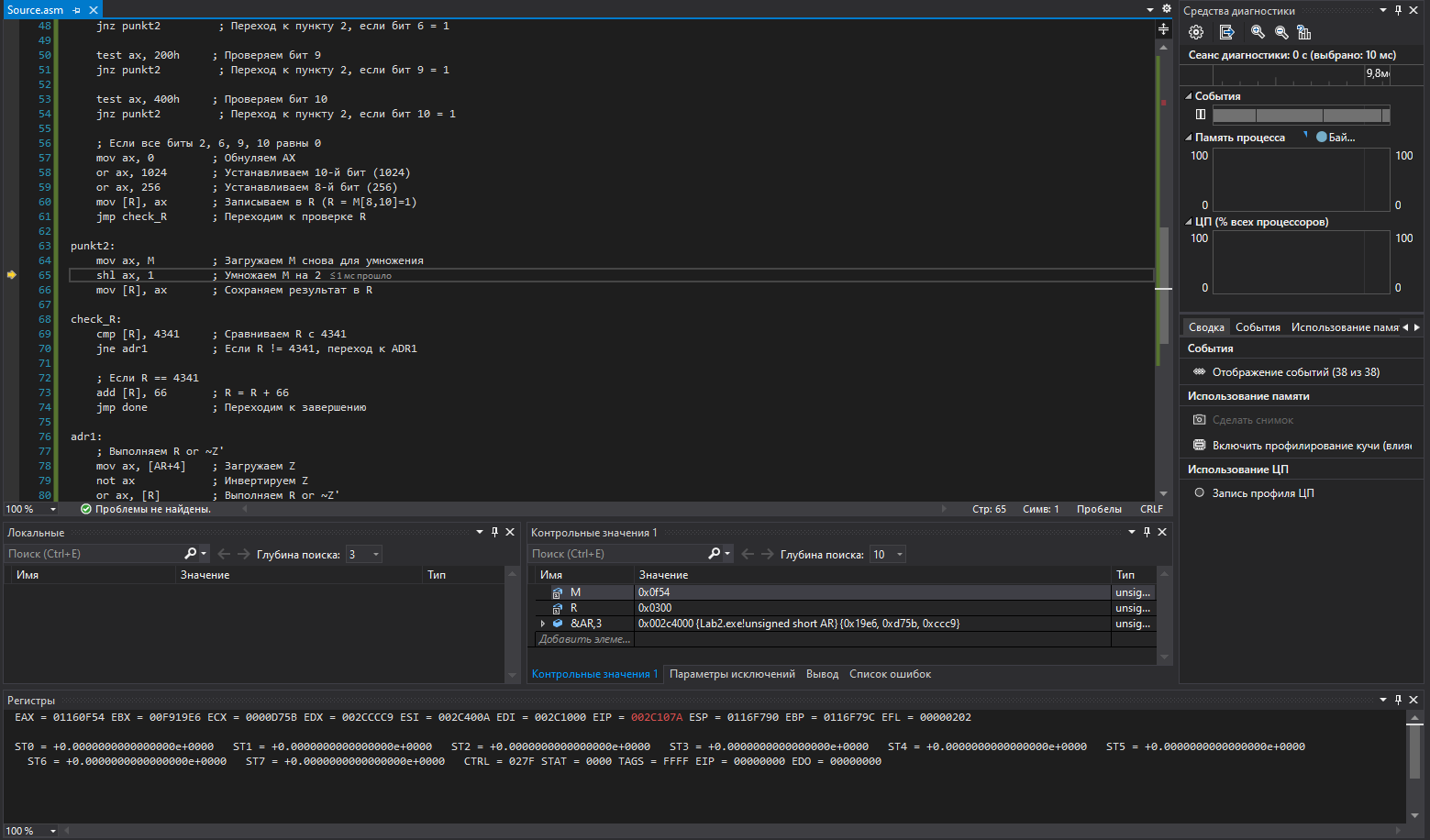


Рисунок 18 – Умножаем M на 2

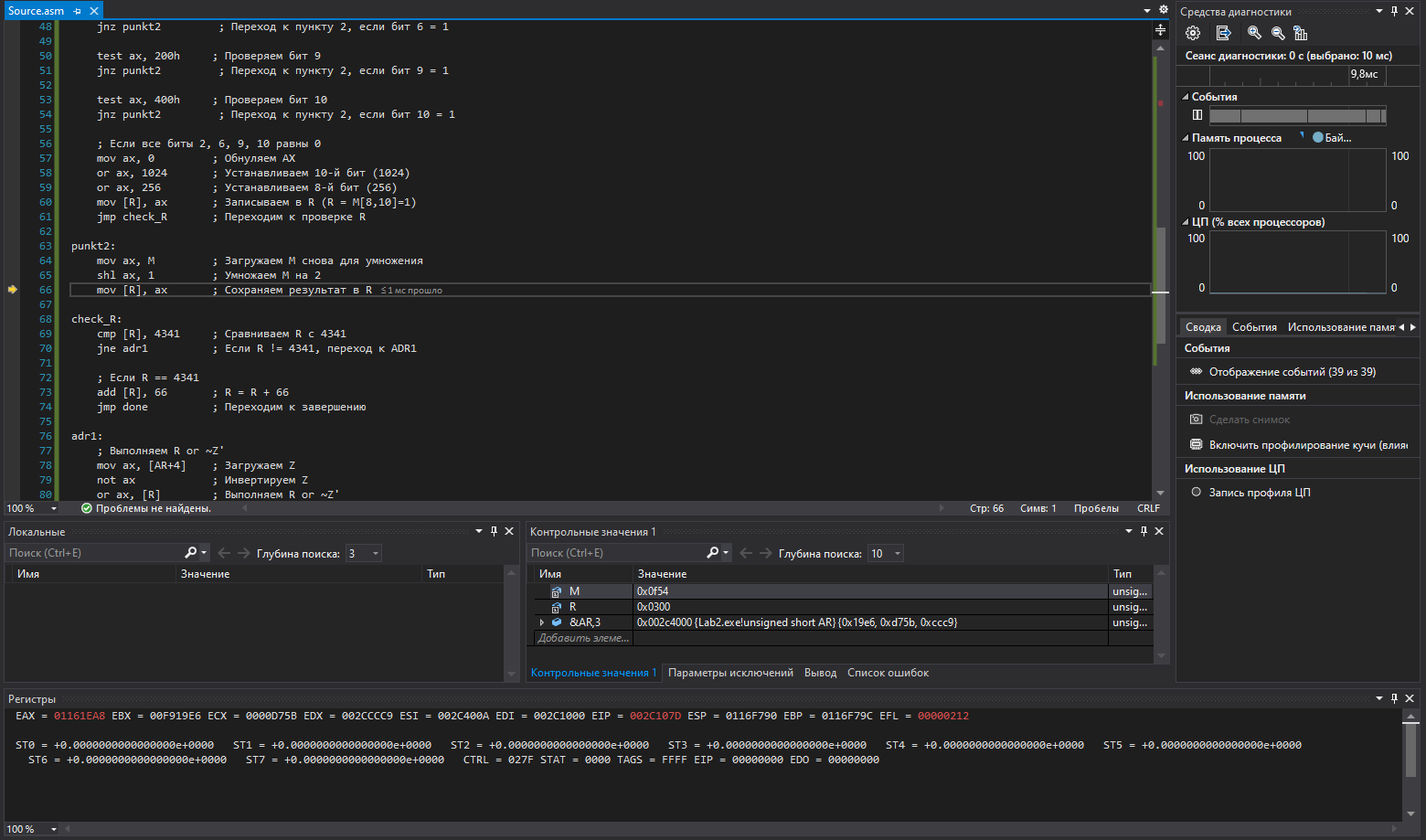


Рисунок 19 – Сохраняем результат в R

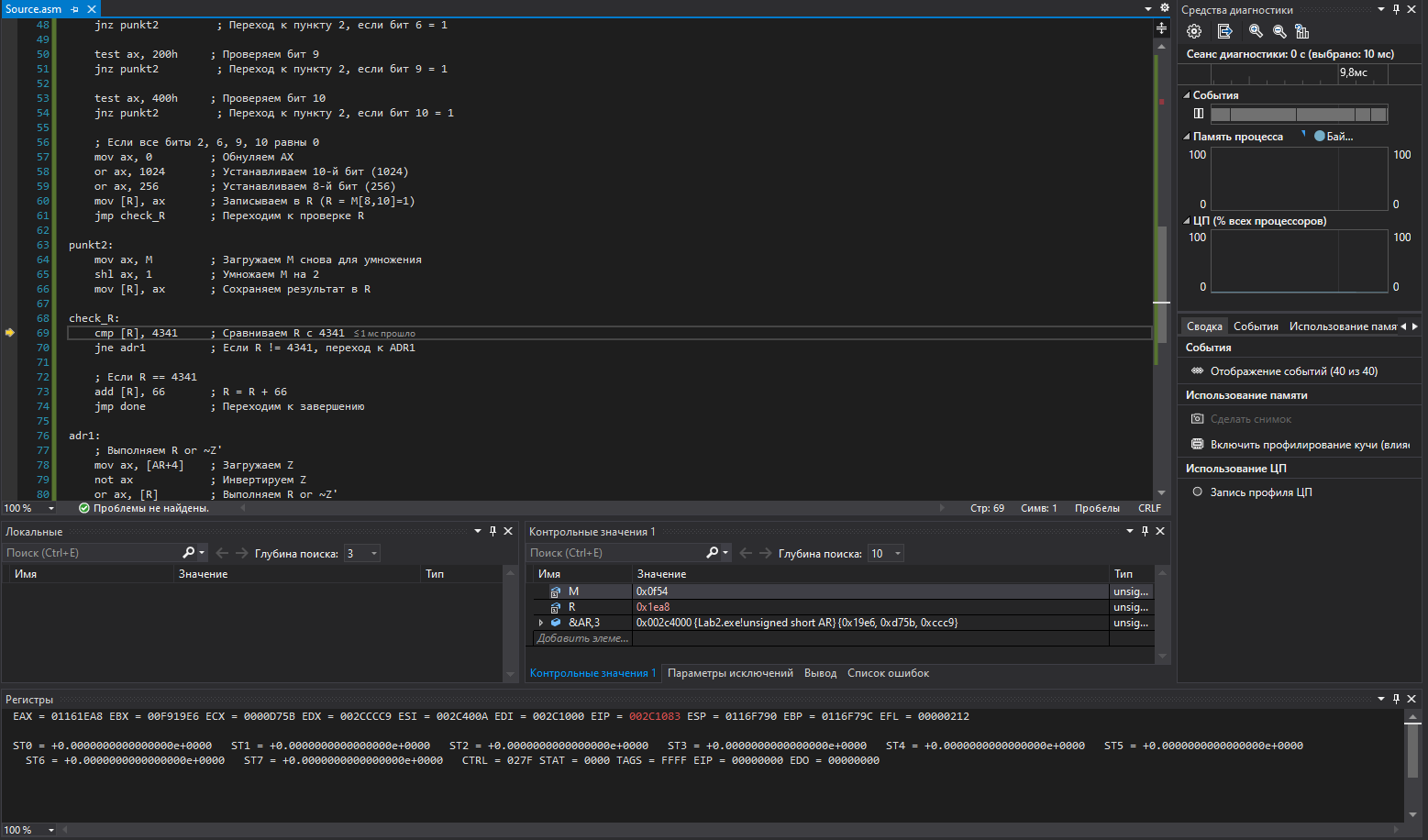


Рисунок 20 – Сравниваем R с 4341

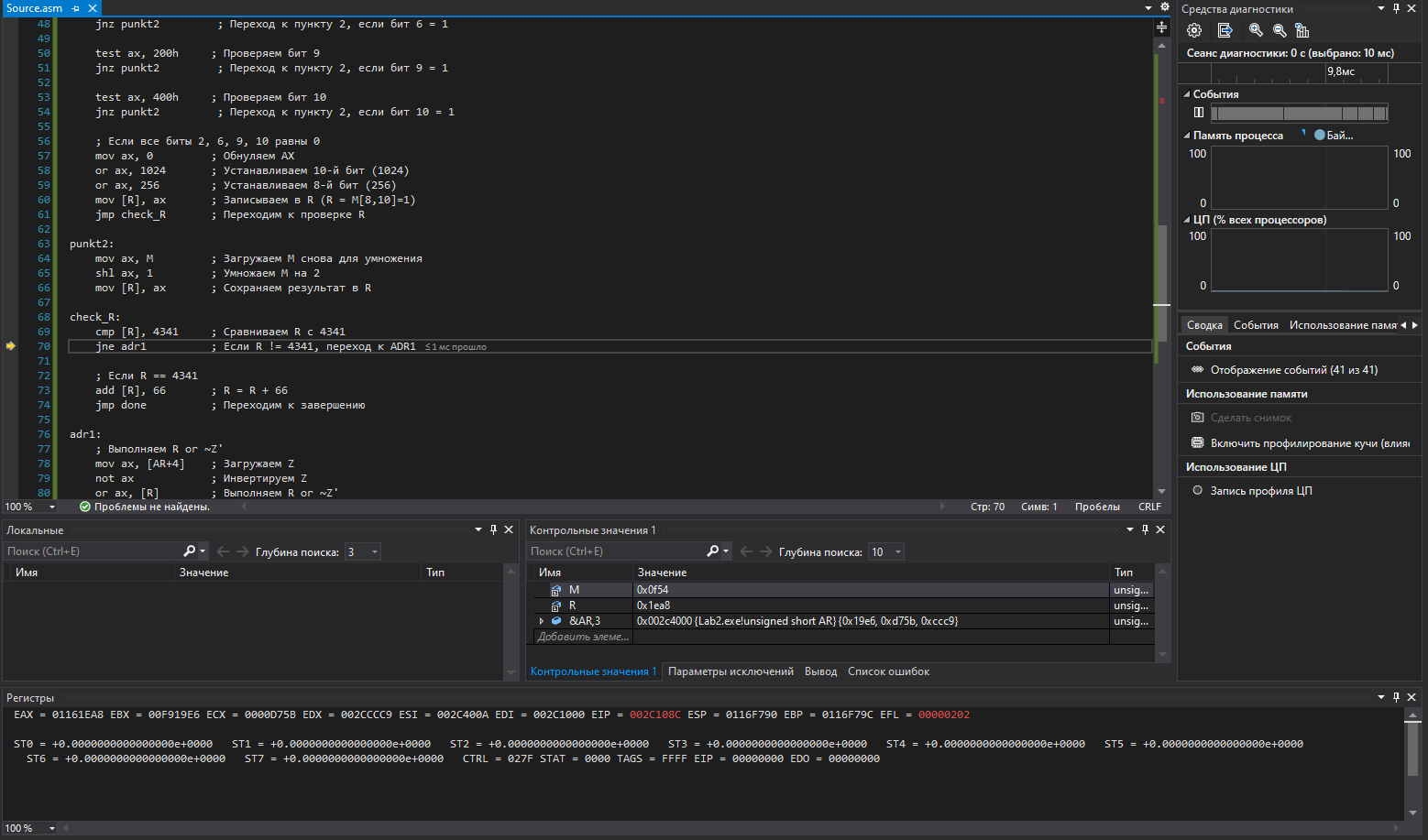


Рисунок 21 – Если R != 4341, переход к ADR1

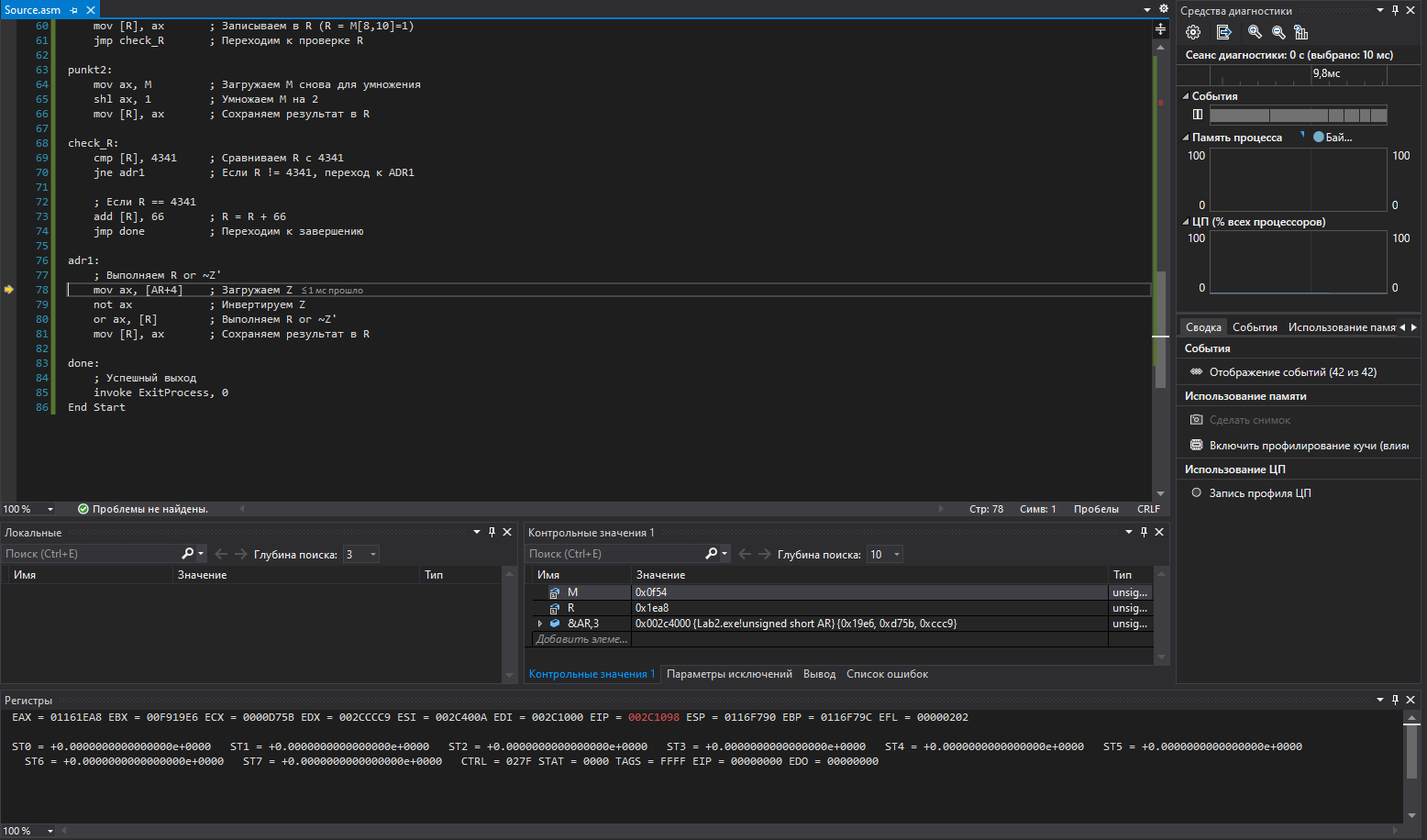


Рисунок 22 – Загружаем Z’ в ax

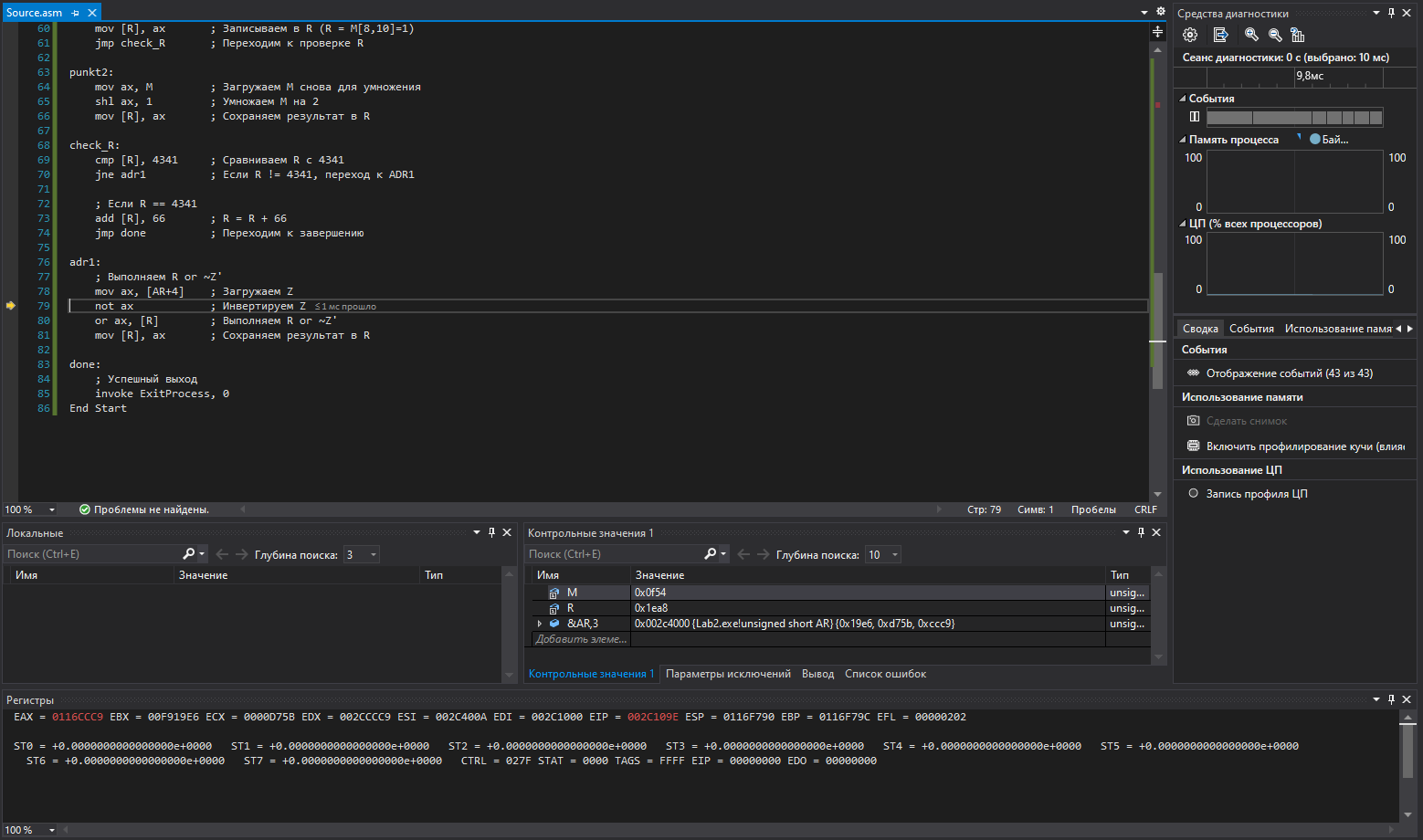


Рисунок 23 – Инвертируем Z’

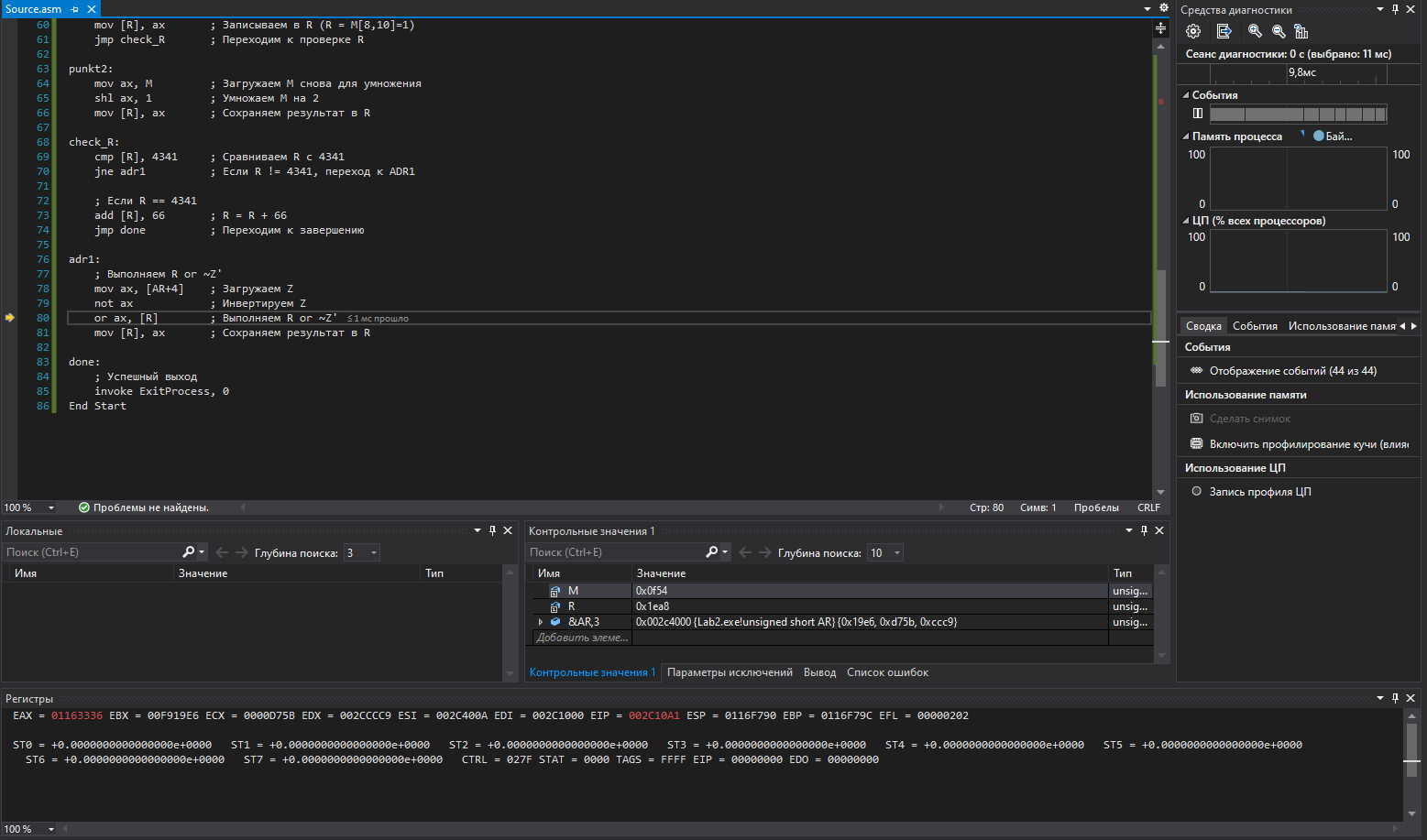


Рисунок 24 – Выполняем R or ~Z'

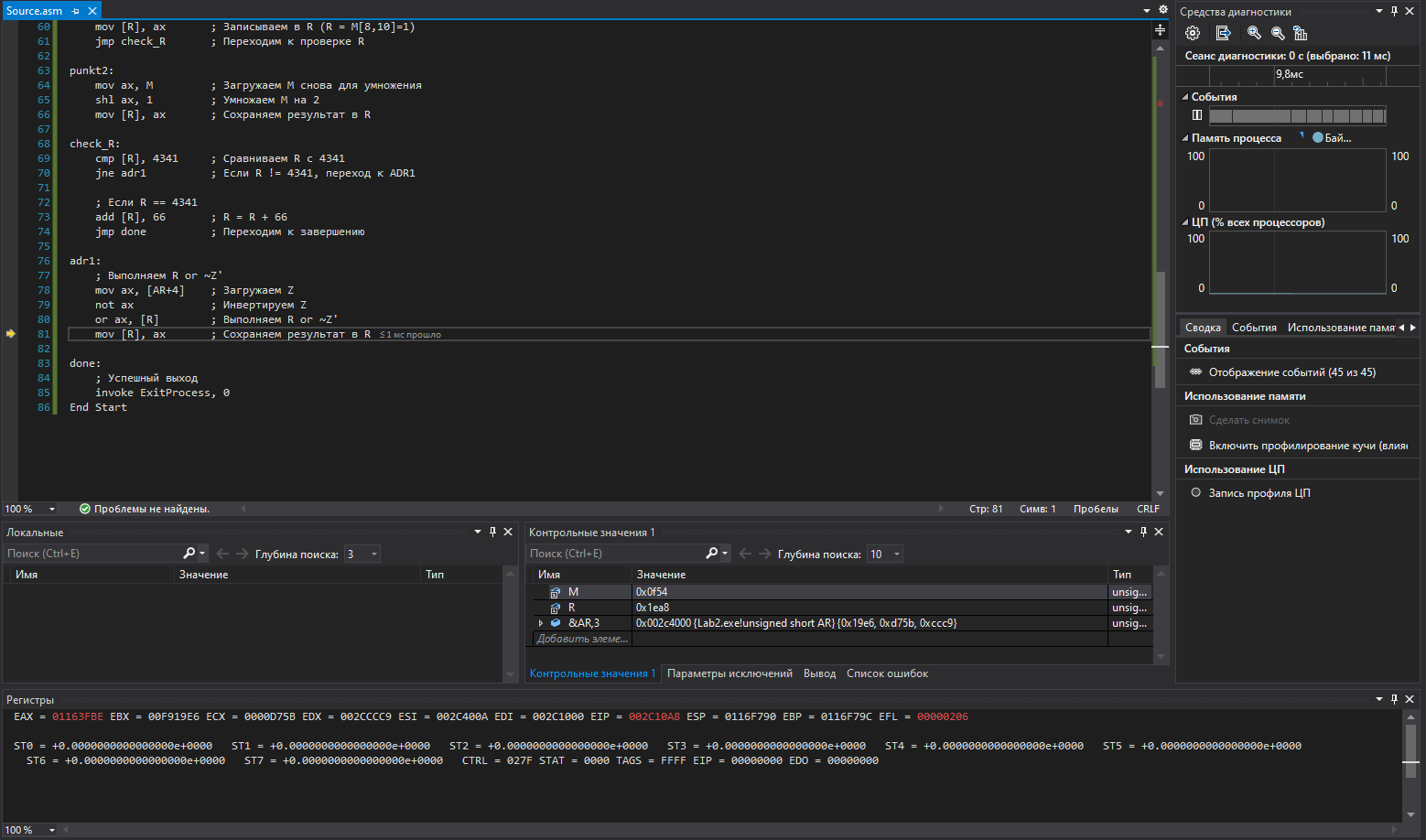


Рисунок 25 – Сохраняем результат в R

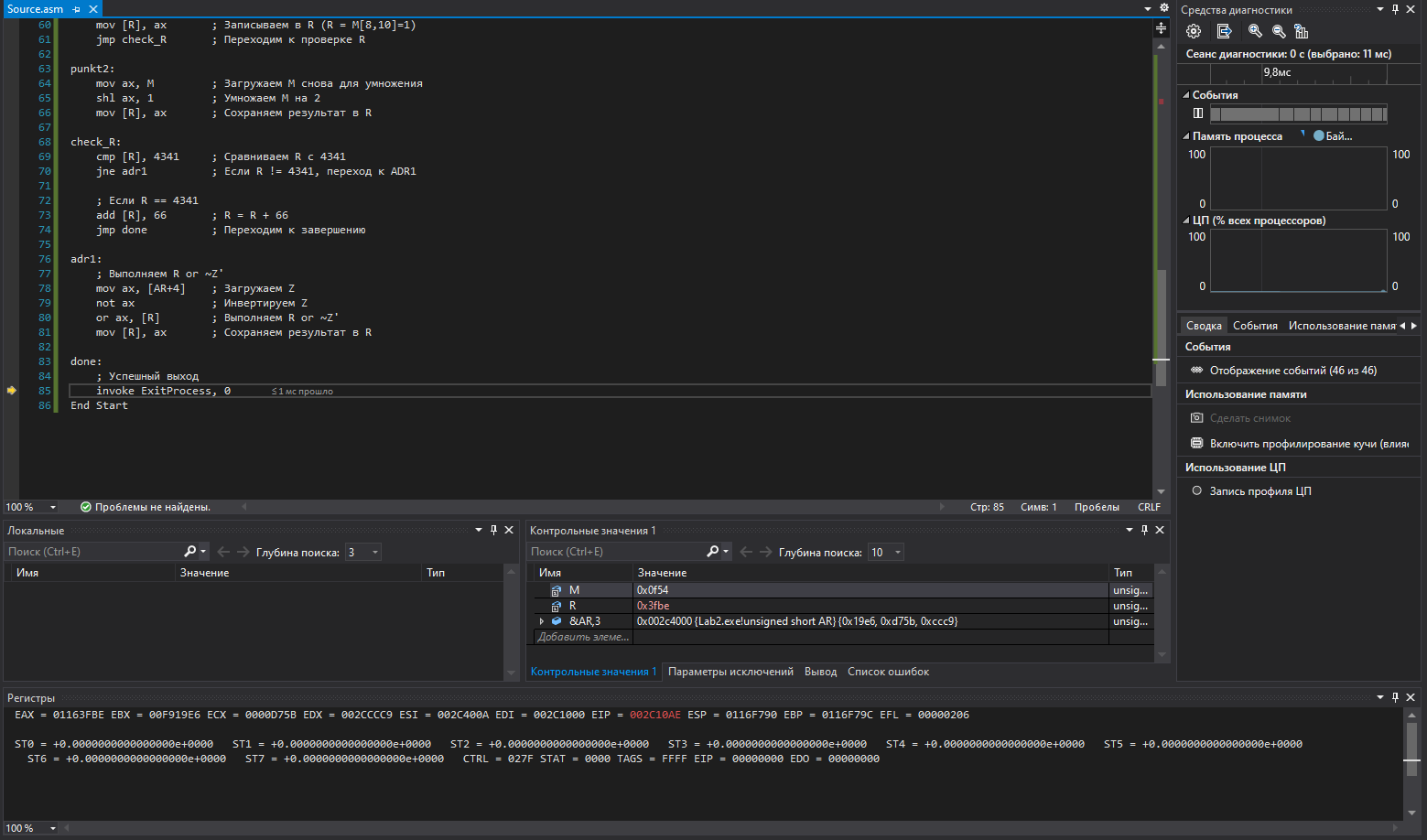


Рисунок 26 – Успешный выход

**Вывод:** в ходе выполнения лабораторной работы были получены навыки написания команд ветвления, организации циклов и подпрограмм в микропроцессорах с архитектурой x86.