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 №3

по дисциплине «Цифровые устройства и микропроцессоры»

“ИСПОЛЬЗОВАНИЕ МАТЕМАТИЧЕСКОГО СОПРОЦЕССОРА”

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 арифметических команд с помощью математического сопроцессора FPU микропроцессоров с архитектурой x86.

**Выполнение индивидуального задания**

Напишите программу на ассемблере, реализующую решение задачи в соответствие с вариантом. В вычислениях использовать команды математического сопроцессора.

Для матрицы N x M найти номер строки, сумма элементов которой является минимальной. Элементы массива 16-разрядные числа со знаком.

**Текст программы:**

.686

.model flat,stdcall

.stack 100h

.data

numbers dd -2.1, 7.1, 0.1, 10.1

dd 5.1, -3.1, 8.2, -1.2

dd -10.2, 9.2, -2.2, -6.2

elemSize = 4 ; размер одного элемента - dword

rows = 3 ; кол-во строк

cols = 4 ; кол-во столбцов

;rowIndex dword 1 ; обращаемся к 2-й строке

;colIndex dword 2 ; обращаемся к 3-му столбцу

NUM dd ?

MIN\_SUM dd 100.0

Rcol dd 0

.code

ExitProcess PROTO STDCALL :DWORD

Start:

mov ecx, 0

mov dx, 0

outerLoop:

cmp ecx, rows

je done

mov ebx, 0

mov NUM, 0

fld NUM

innerLoop:

mov eax,ebx

mov edx, ecx ; Копируем индекс строки в eax

imul edx, 4 ; Умножаем на количество столбцов (4)

add edx, ebx ; Добавляем индекс столбца

shl edx, 2 ; Умножаем на 2 (размер элемента в байтах)

fld [numbers + edx] ; Загружаем значение в st

fadd

add eax, ebx

cmp ebx, cols-1

je innerLoopDone

inc ebx

jmp innerLoop

innerLoopDone:

inc ecx

fstp NUM

mov eax, NUM

cmp eax, MIN\_SUM

jl swap

jmp outerLoop

swap:

mov MIN\_SUM, eax

mov Rcol, ecx

jmp outerLoop

done:

exit:

Invoke ExitProcess,1

End Start

**Верификация программы**

Для примера будет взята матрица numbers:

-2.1, 7.1, 0.1, 10.1

5.1, -3.1, 8.2, -1.2

-10.2, 9.2, -2.2, -6.2

В данной матрице суммы чисел в строках будут такие:

1 строка = 15.2

2 строка = 9

3 строка = -9.4

В итоге получается, что строка с наименьшей суммой под номером 3.

Шаги программы при выполнении:

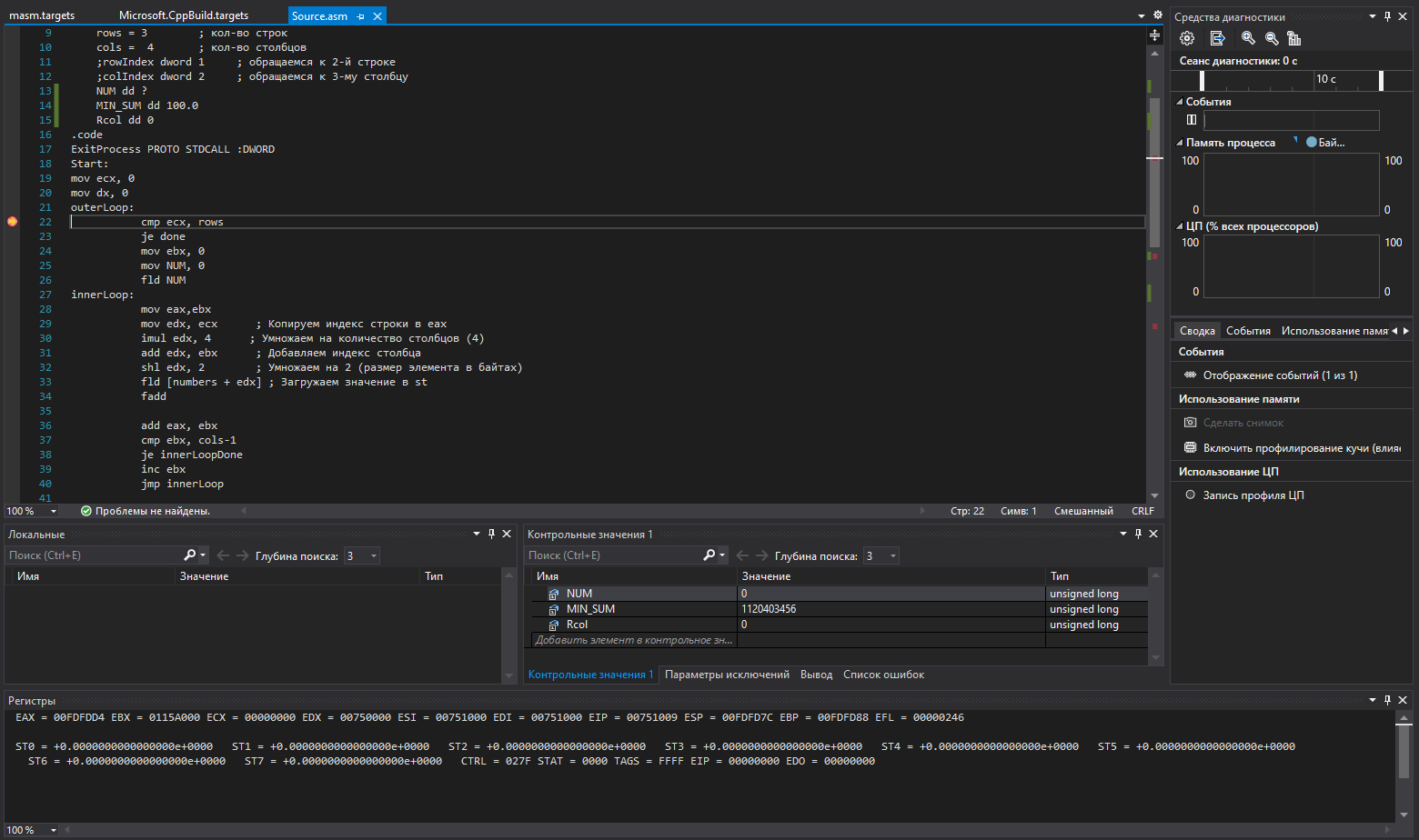


Рисунок 1 – Сравниваем текущий индекс строки с количеством строк

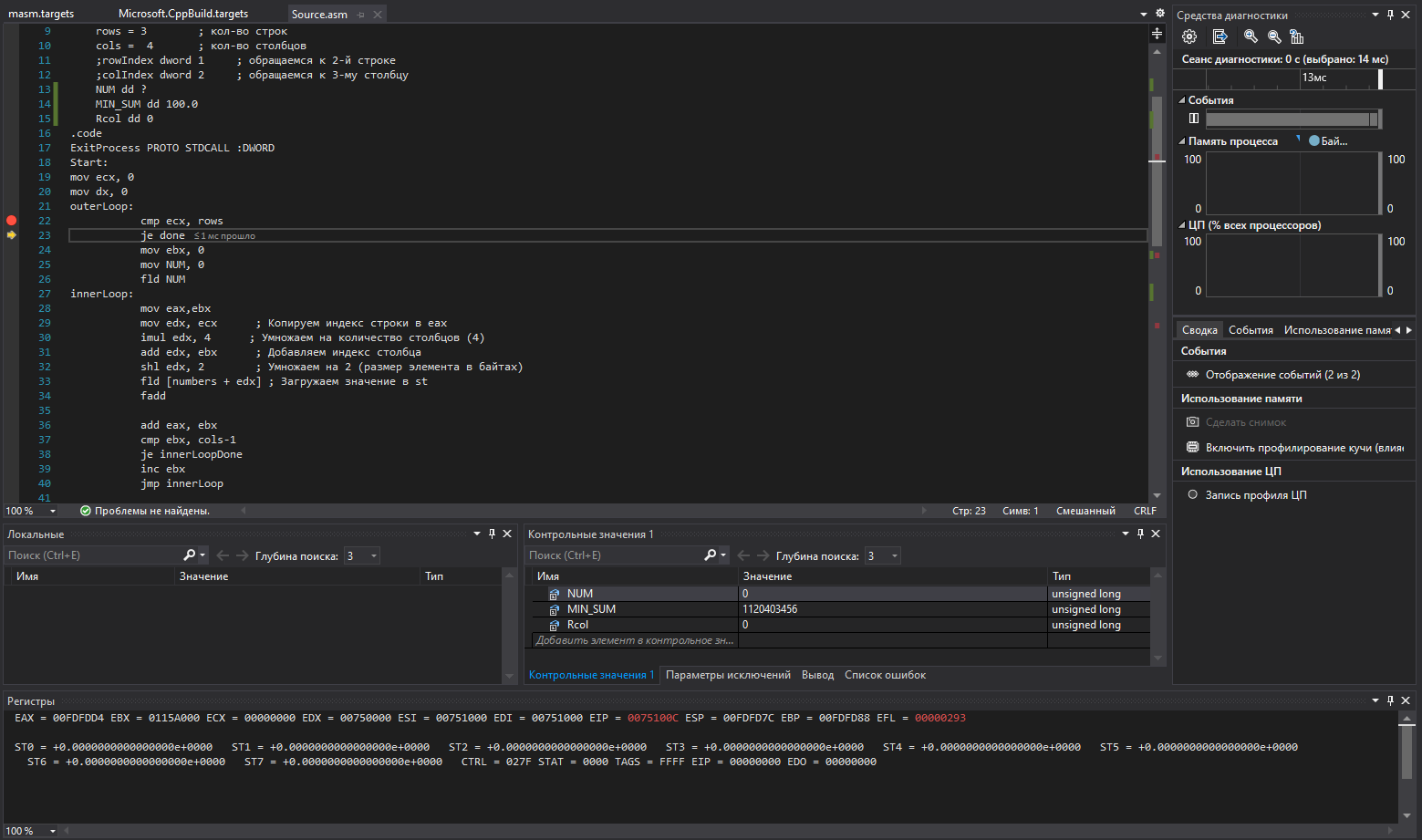


Рисунок 2 – Если индекс равен количеству строк, перейти к завершению

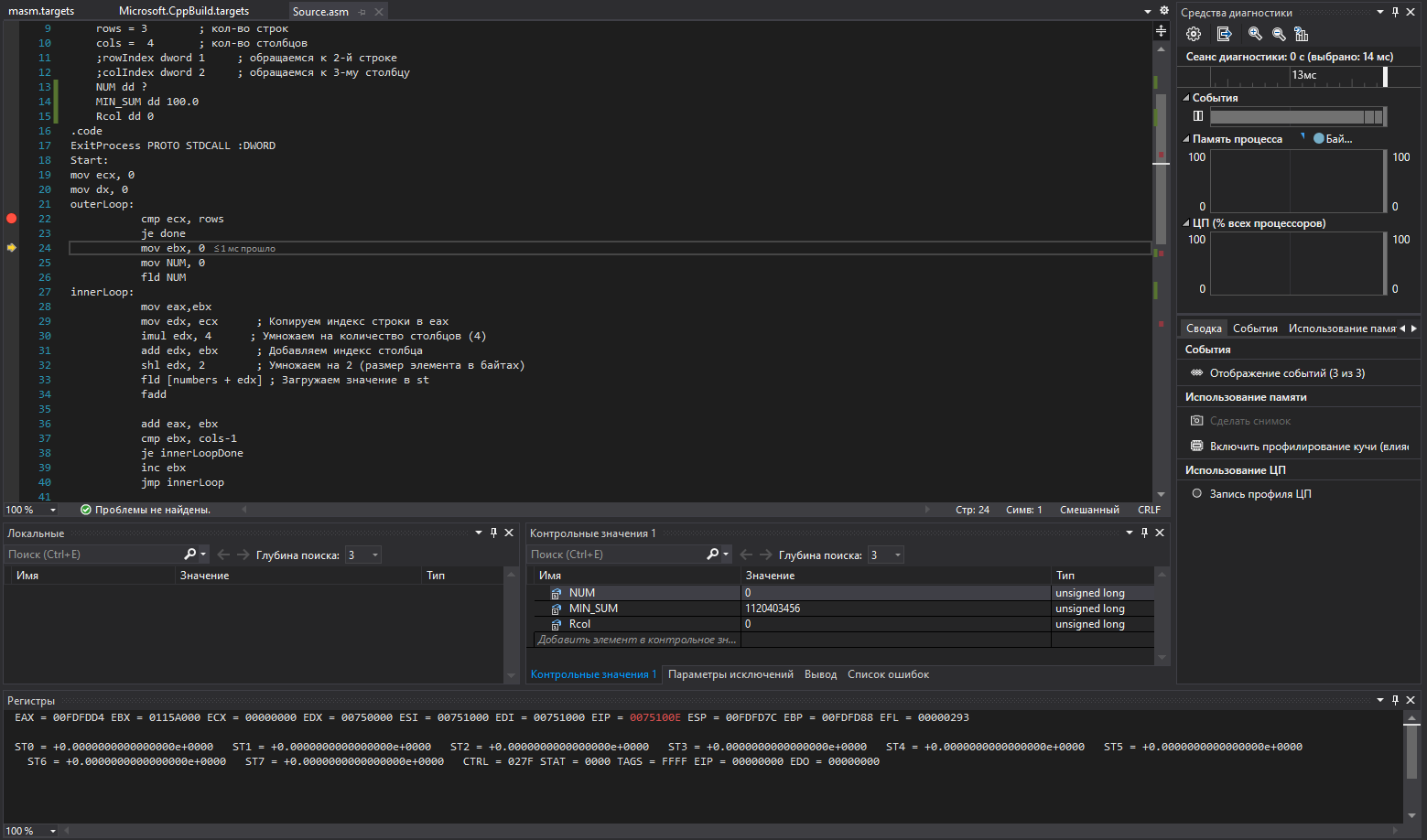


Рисунок 3 – Инициализация счетчика столбцов (ebx = 0)

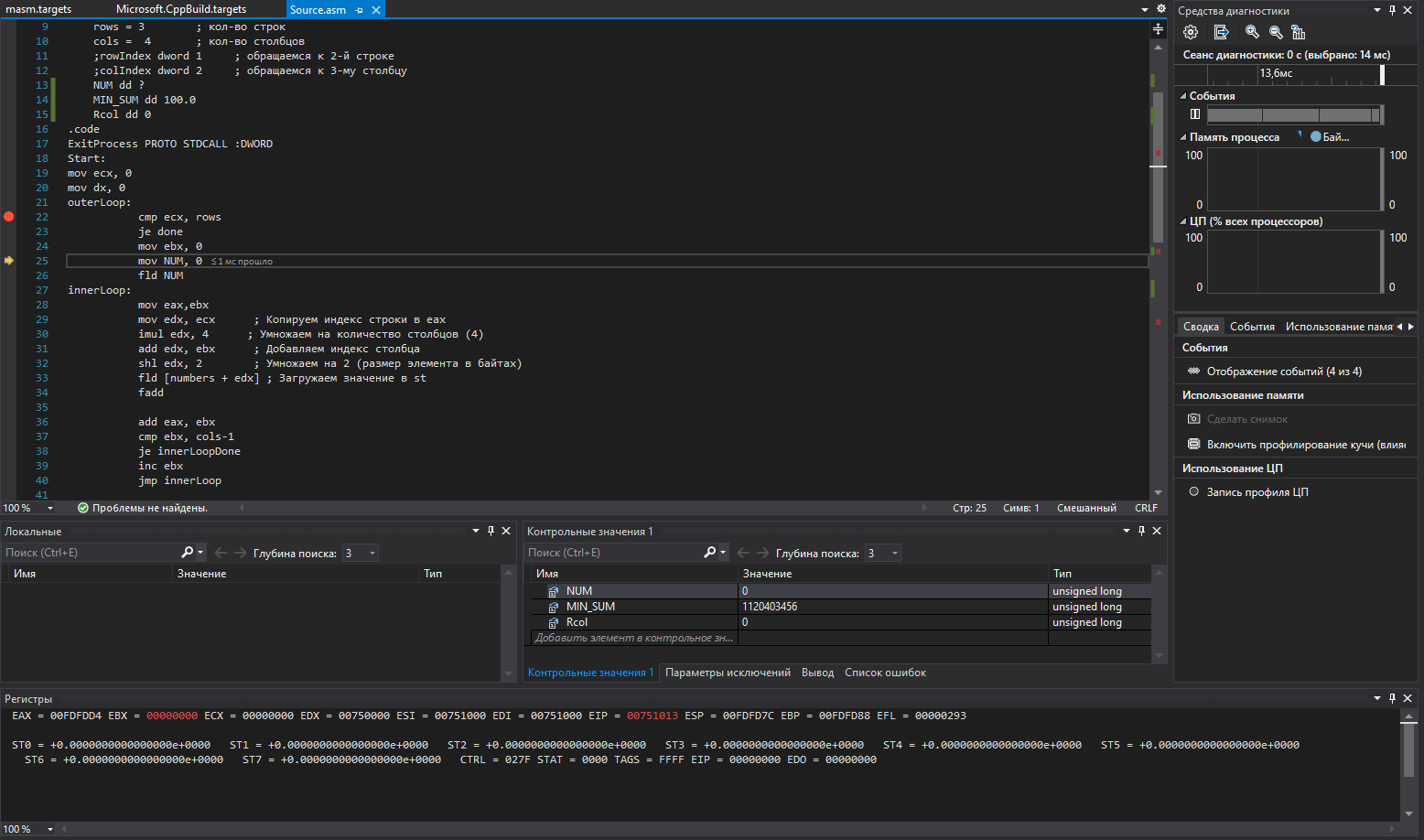


Рисунок 4 – Сброс суммы для текущей строки

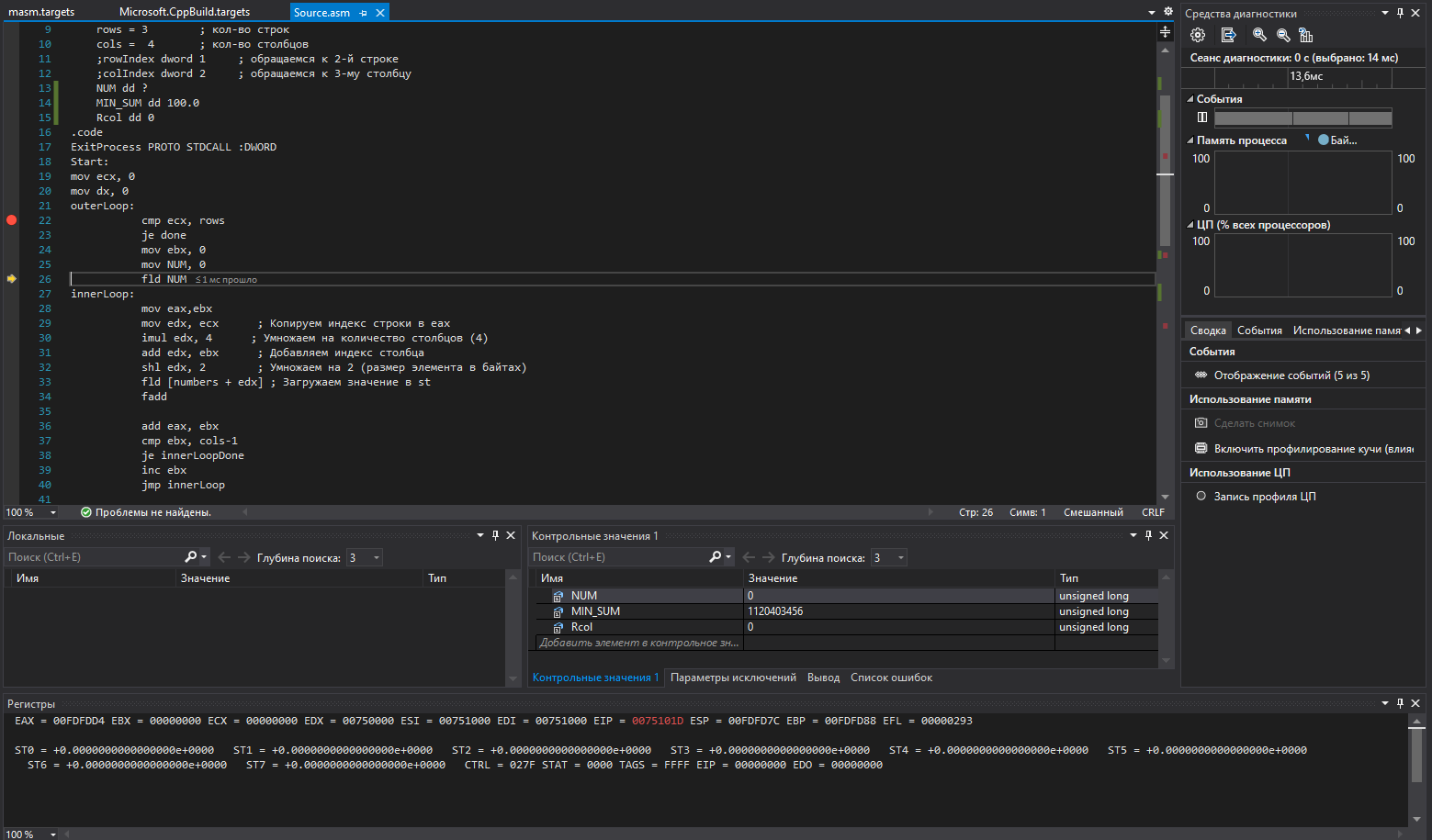


Рисунок 5 – Загружаем значение NUM в стек FPU

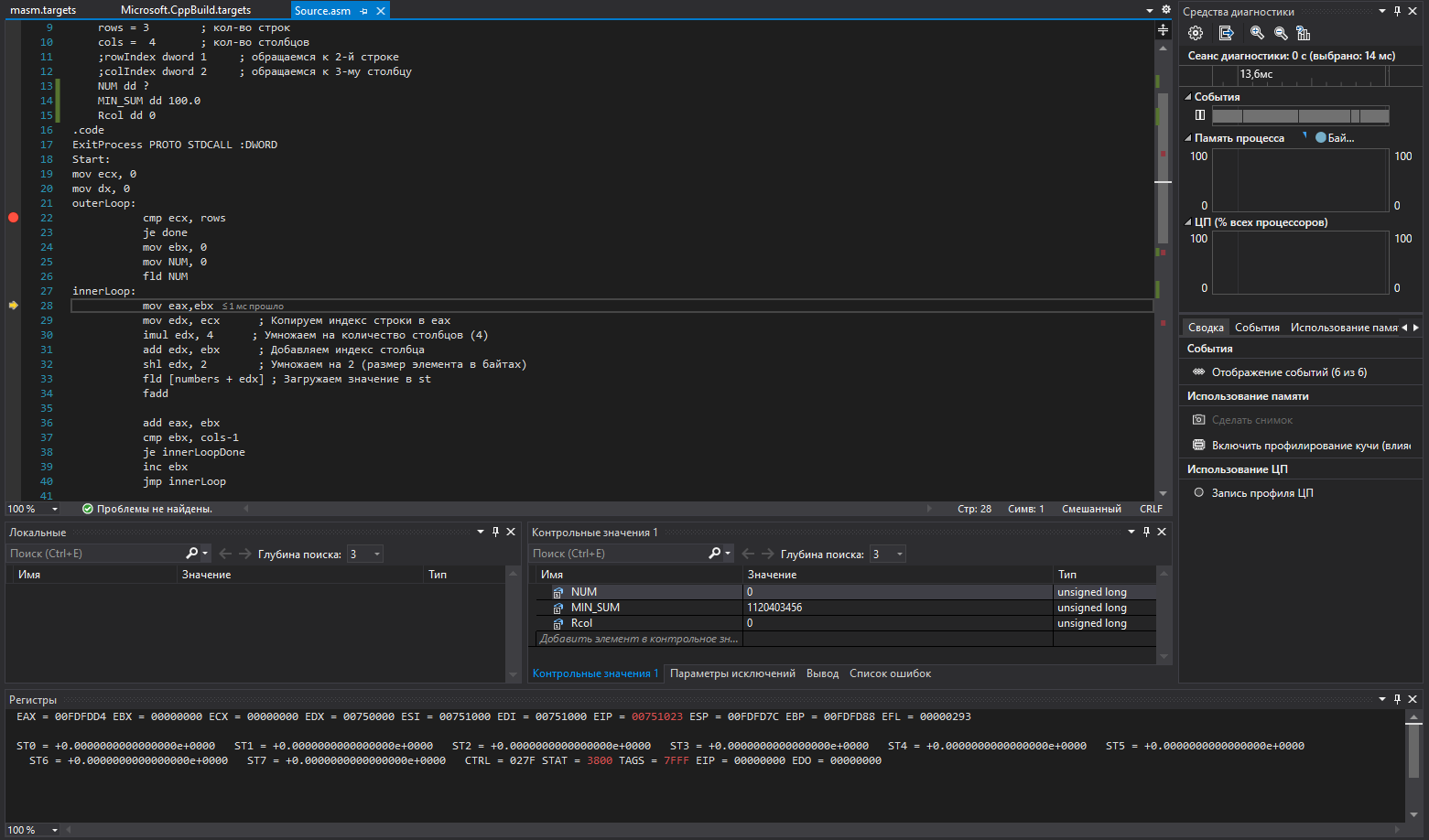


Рисунок 6 – Копируем индекс столбца в eax

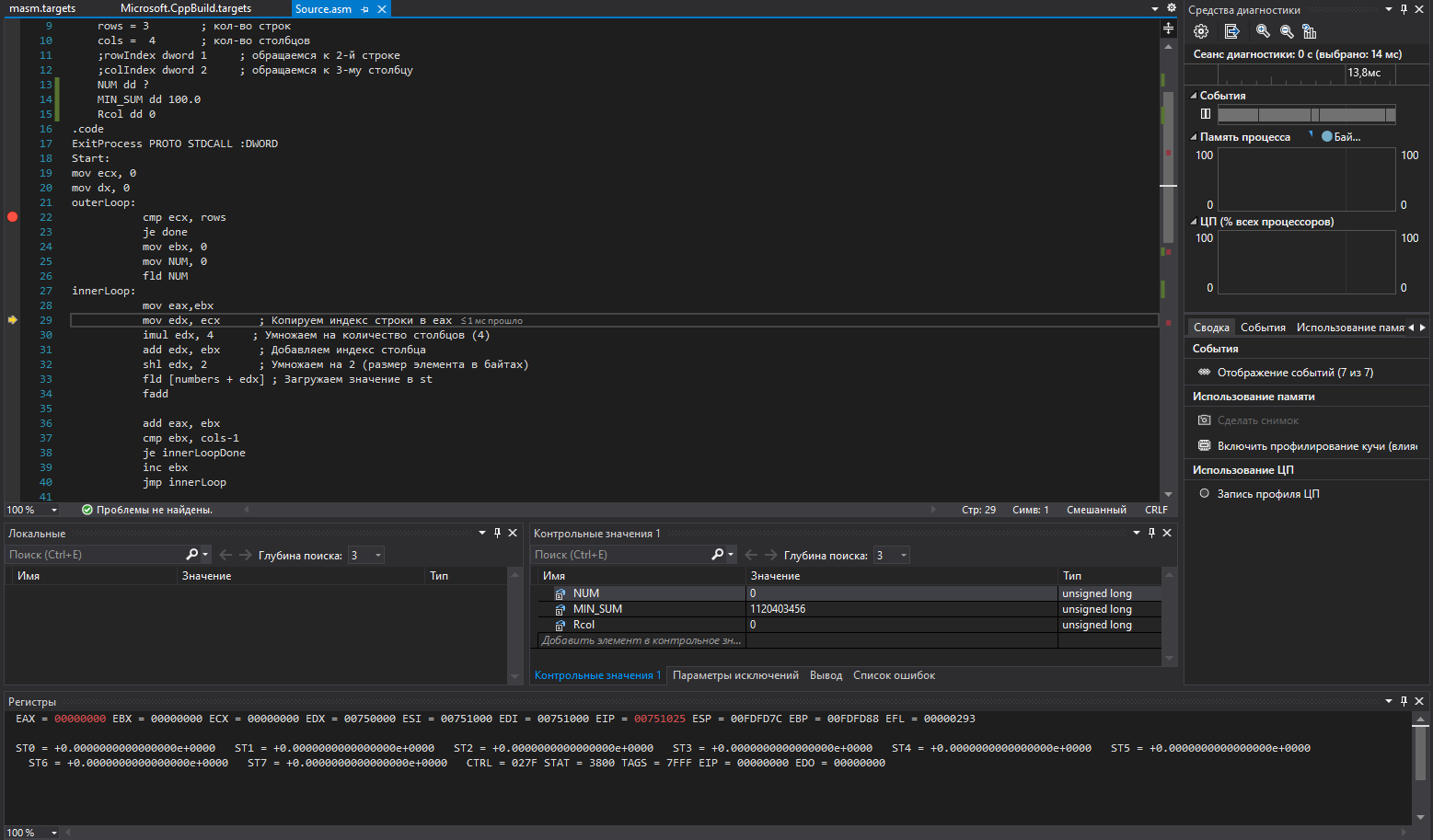


Рисунок 7 – Копируем индекс строки в edx

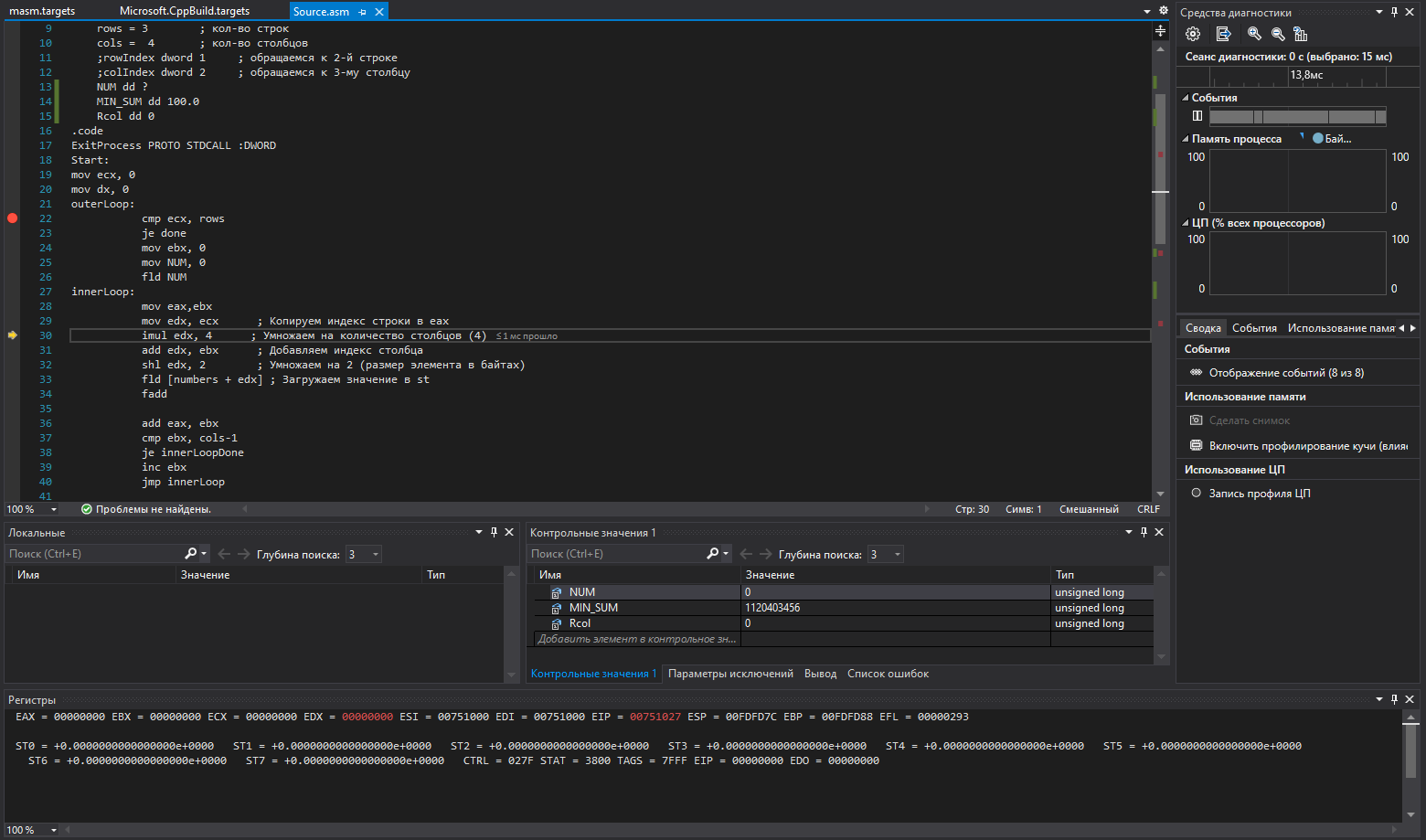


Рисунок 8 – Умножаем индекс строки на количество столбцов (4)

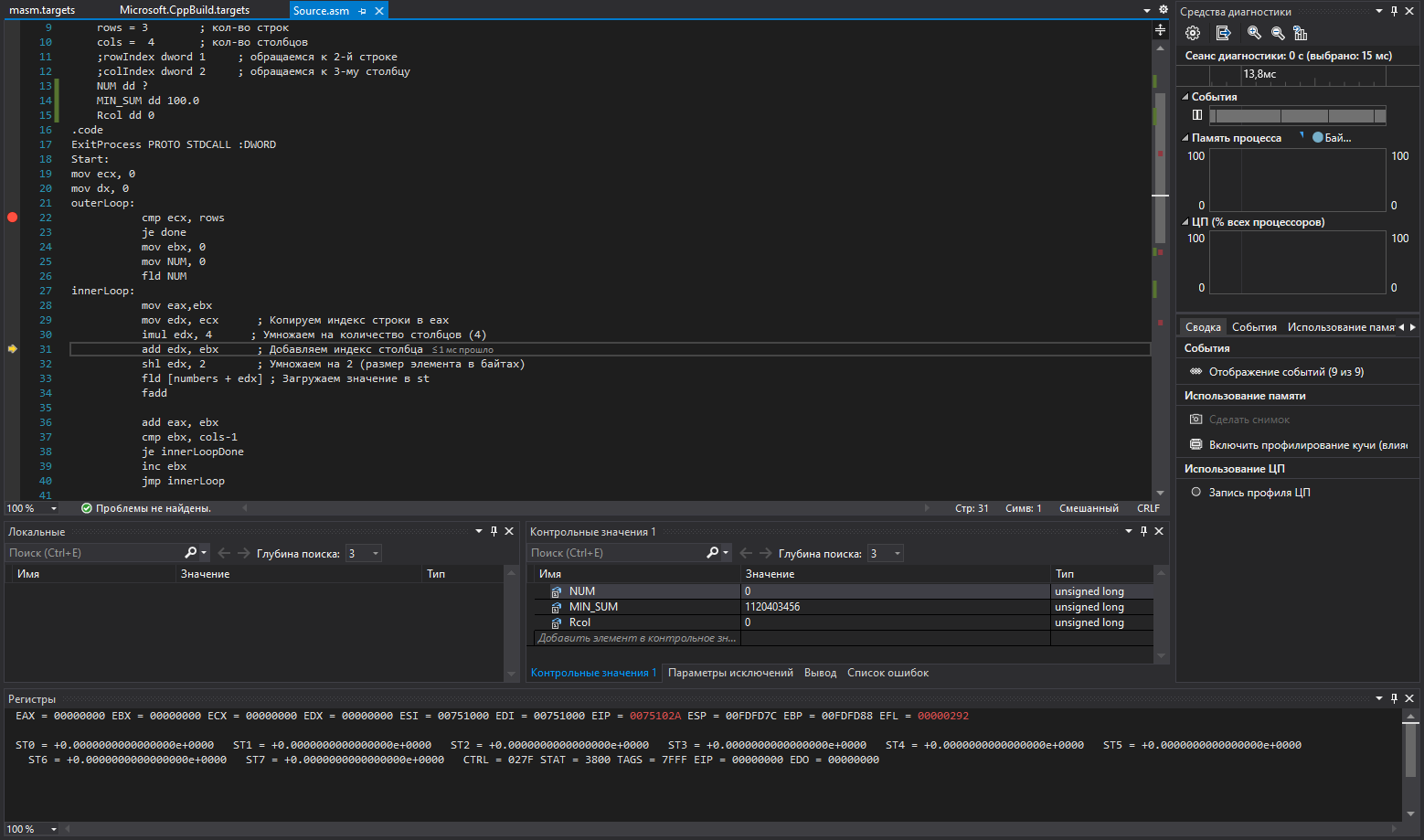


Рисунок 9 – Добавляем индекс столбца

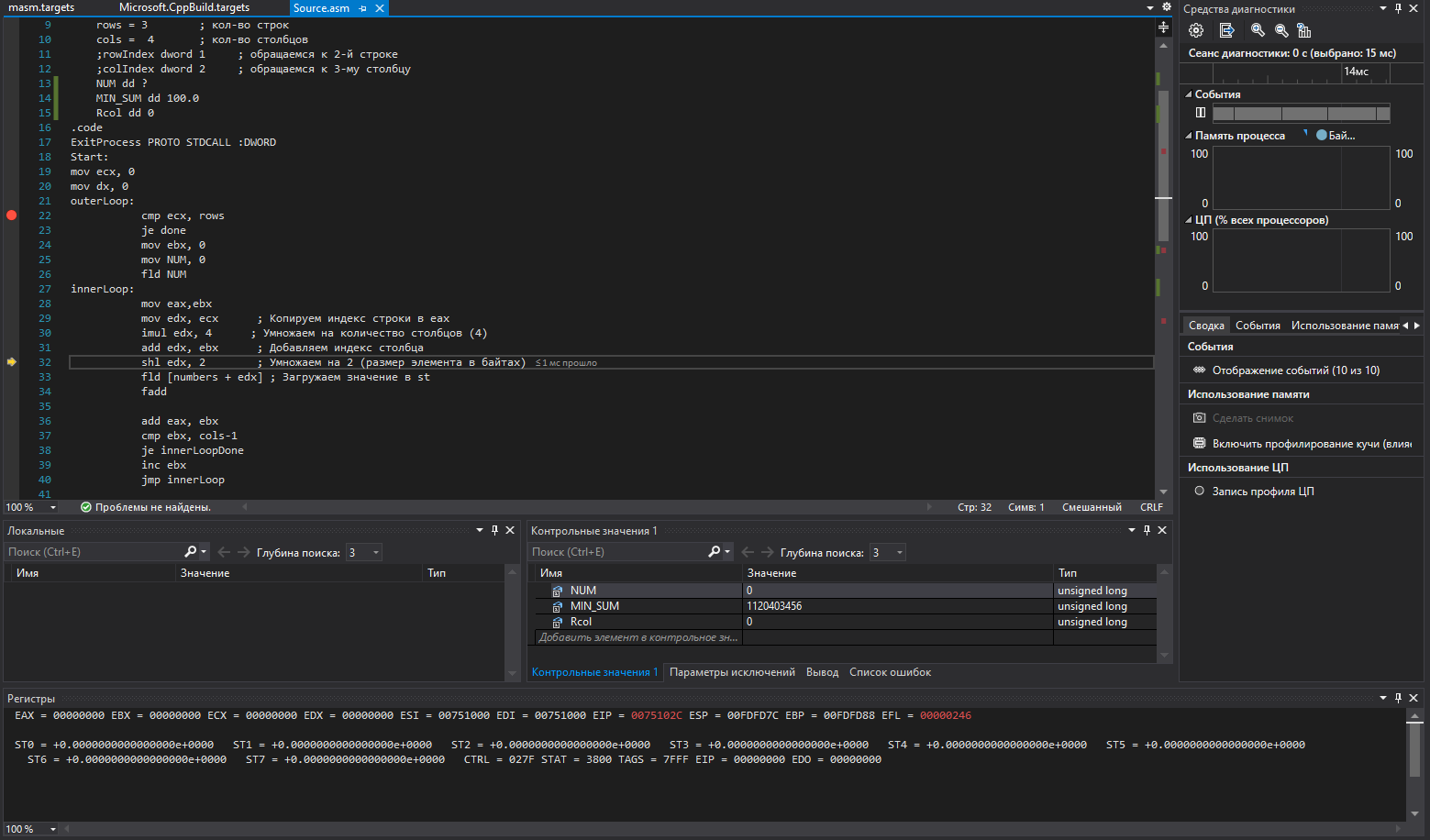


Рисунок 10 – Умножаем на 4 (размер элемента в байтах)

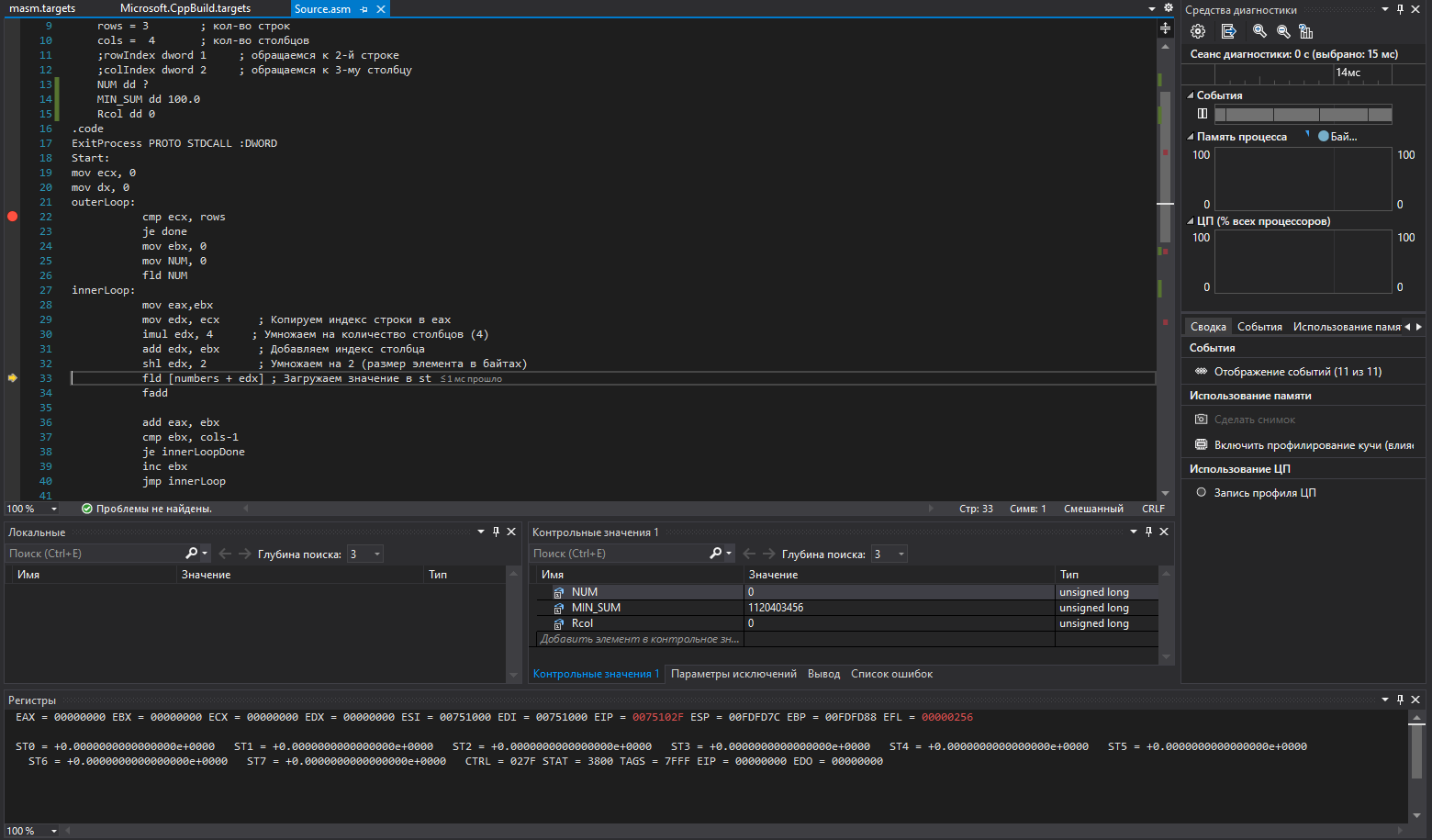


Рисунок 11 – Загружаем значение из массива numbers в стек FPU

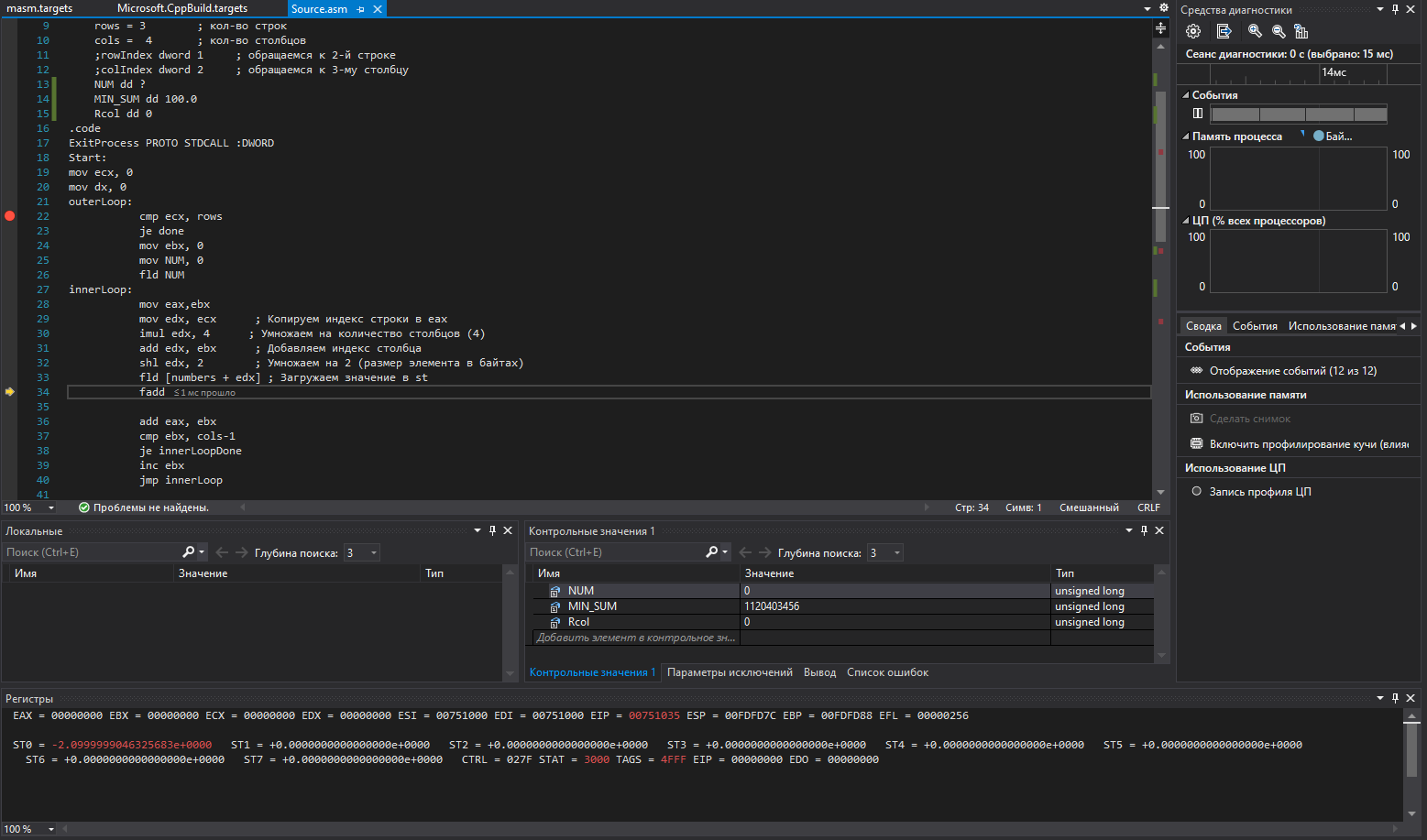


Рисунок 12 – Складываем загруженное значение с текущим значением в st(0)

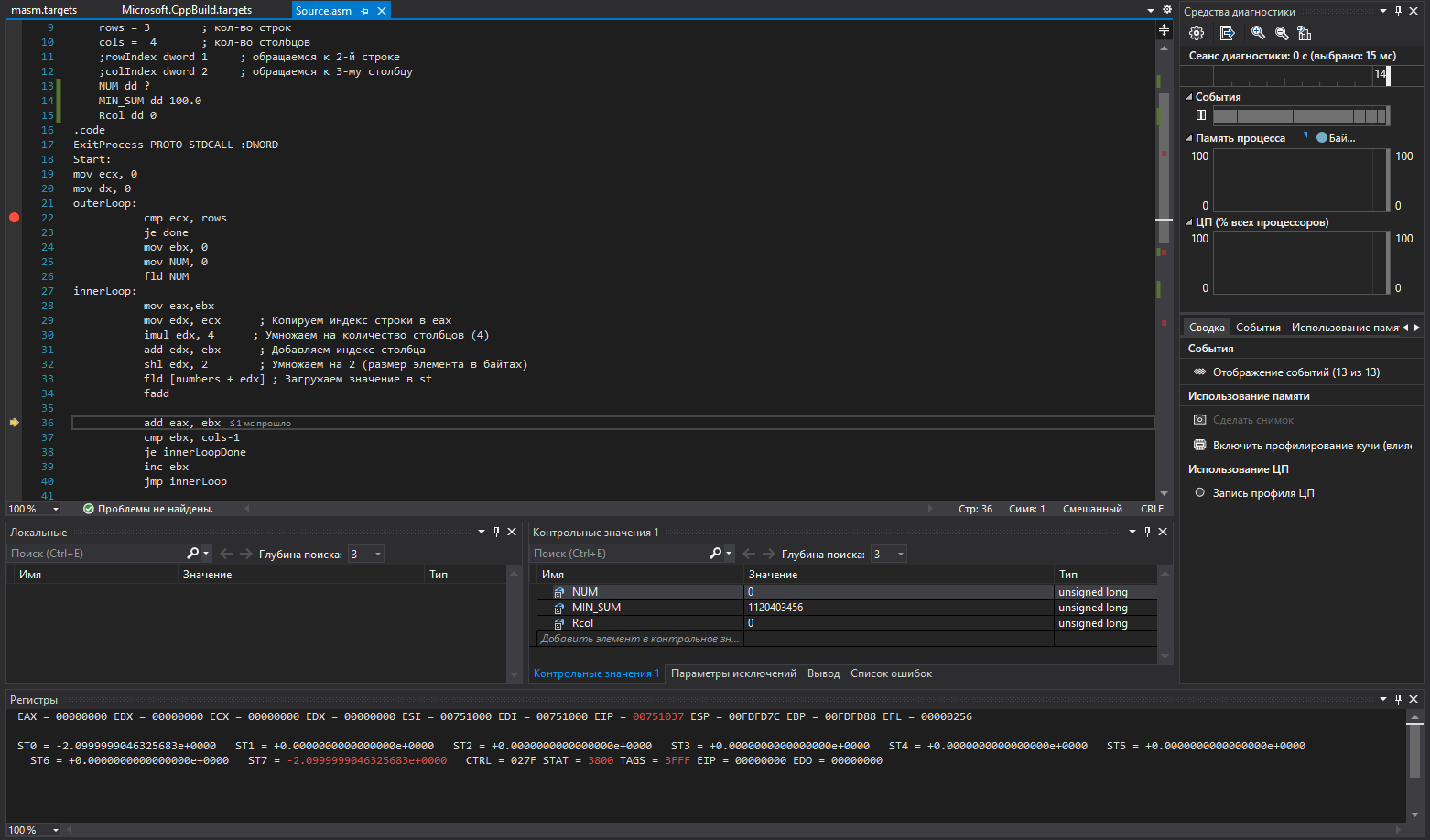


Рисунок 13 – Обновляем значение eax (индекс столбца)

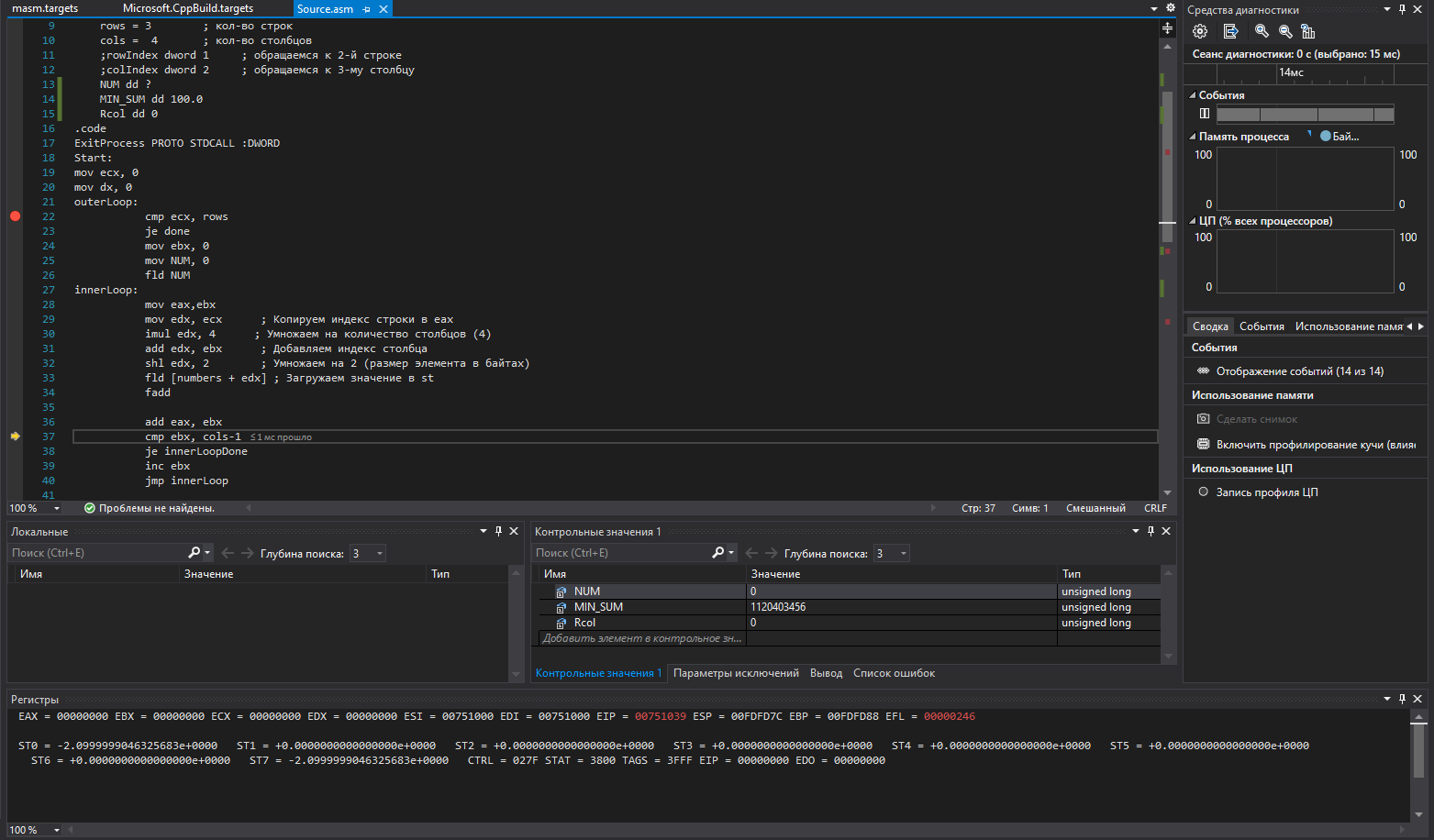


Рисунок 14 – Сравниваем индекс столбца с максимальным индексом

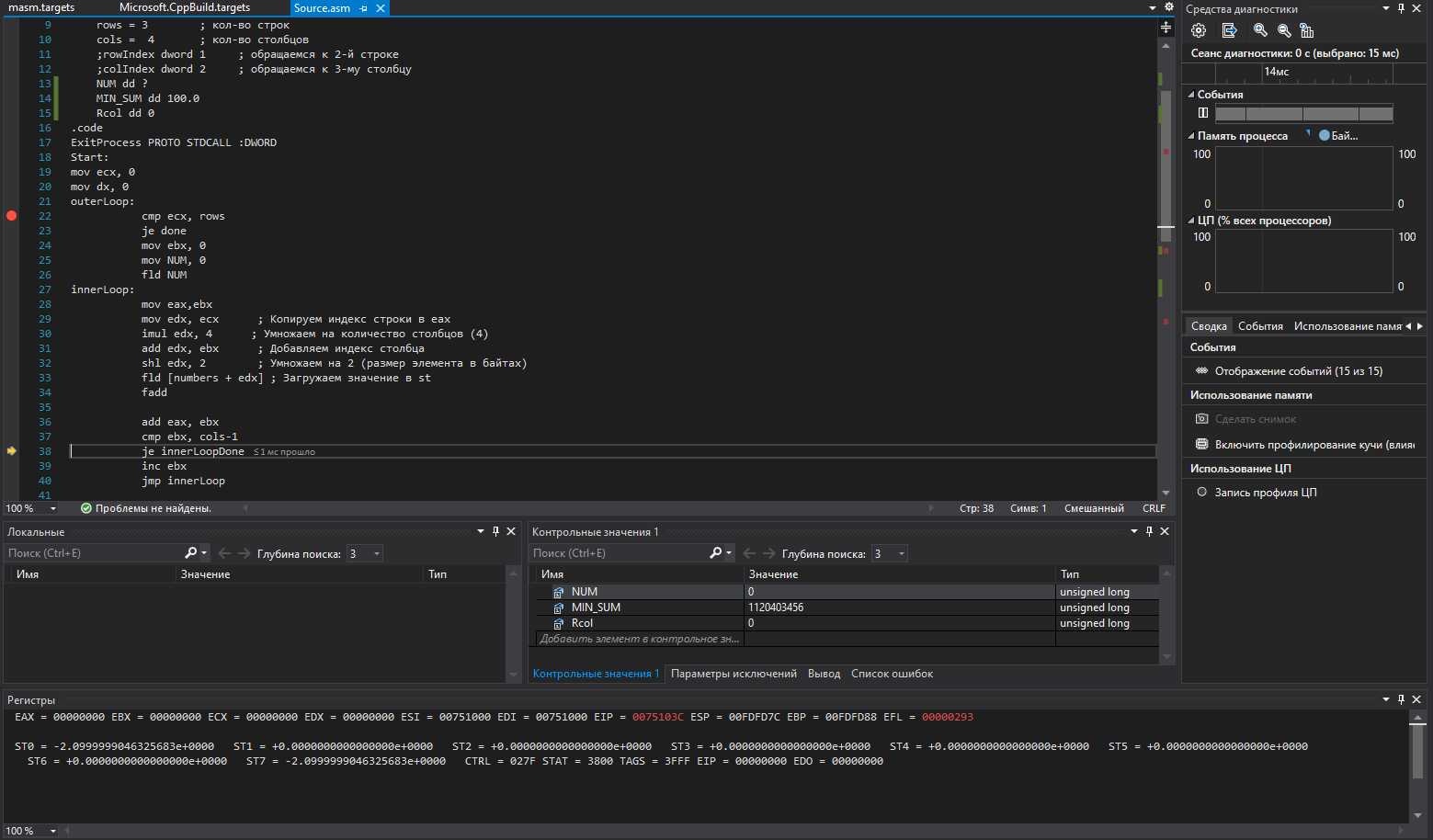


Рисунок 15 – Если индекс столбца последний, перейти к завершению внутреннего цикла

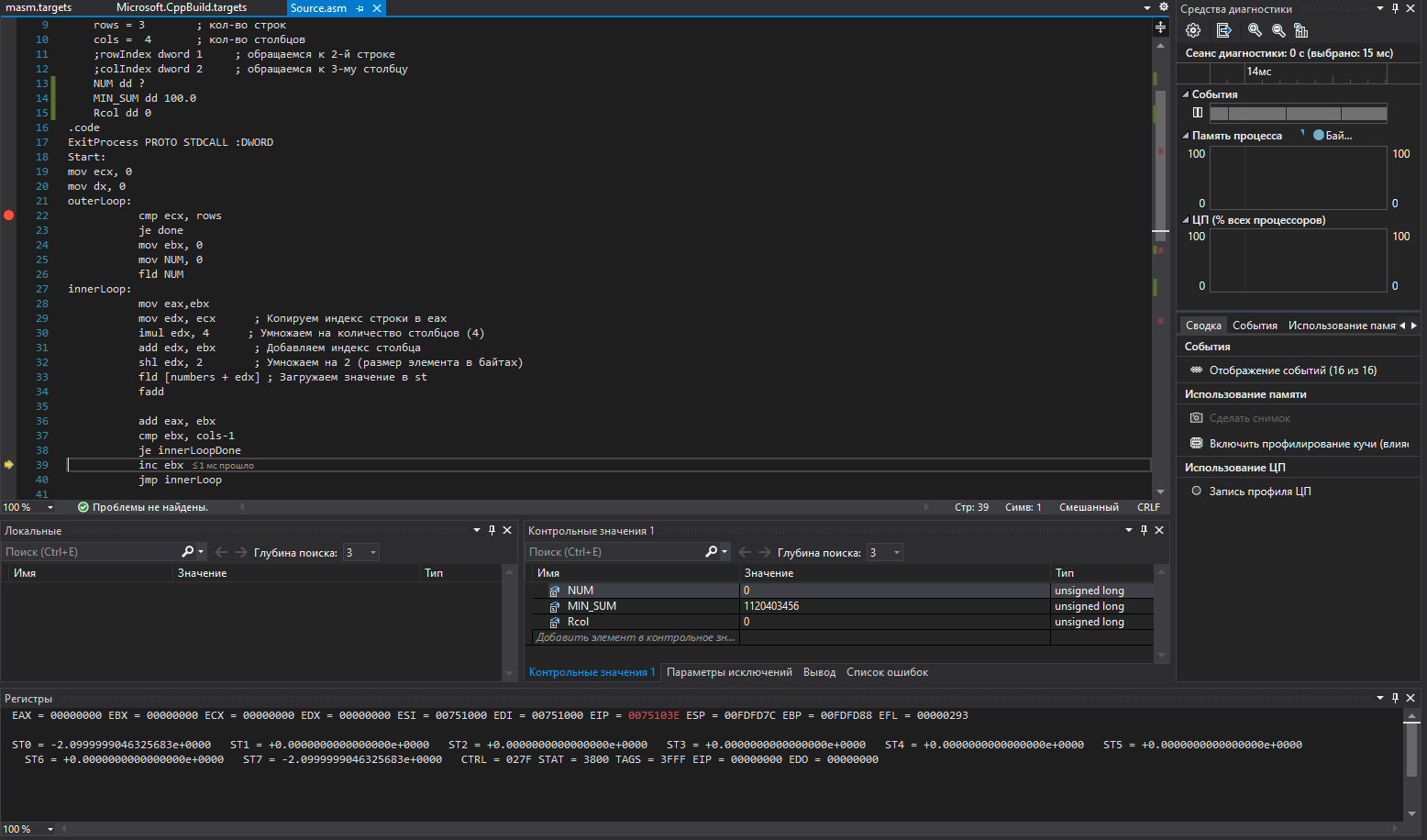


Рисунок 16 – Увеличиваем индекс столбца

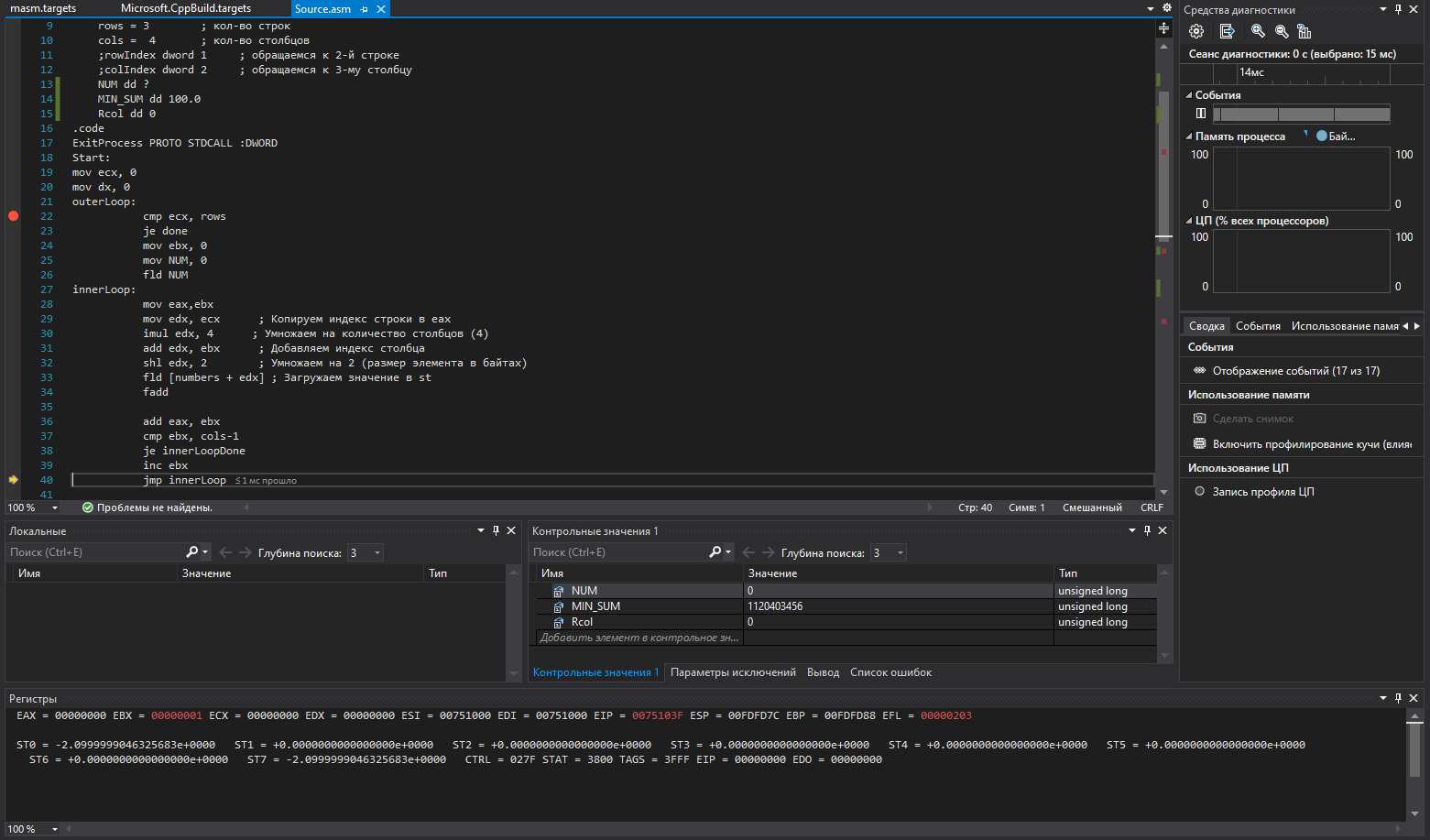


Рисунок 17 – Переход к началу внутреннего цикла

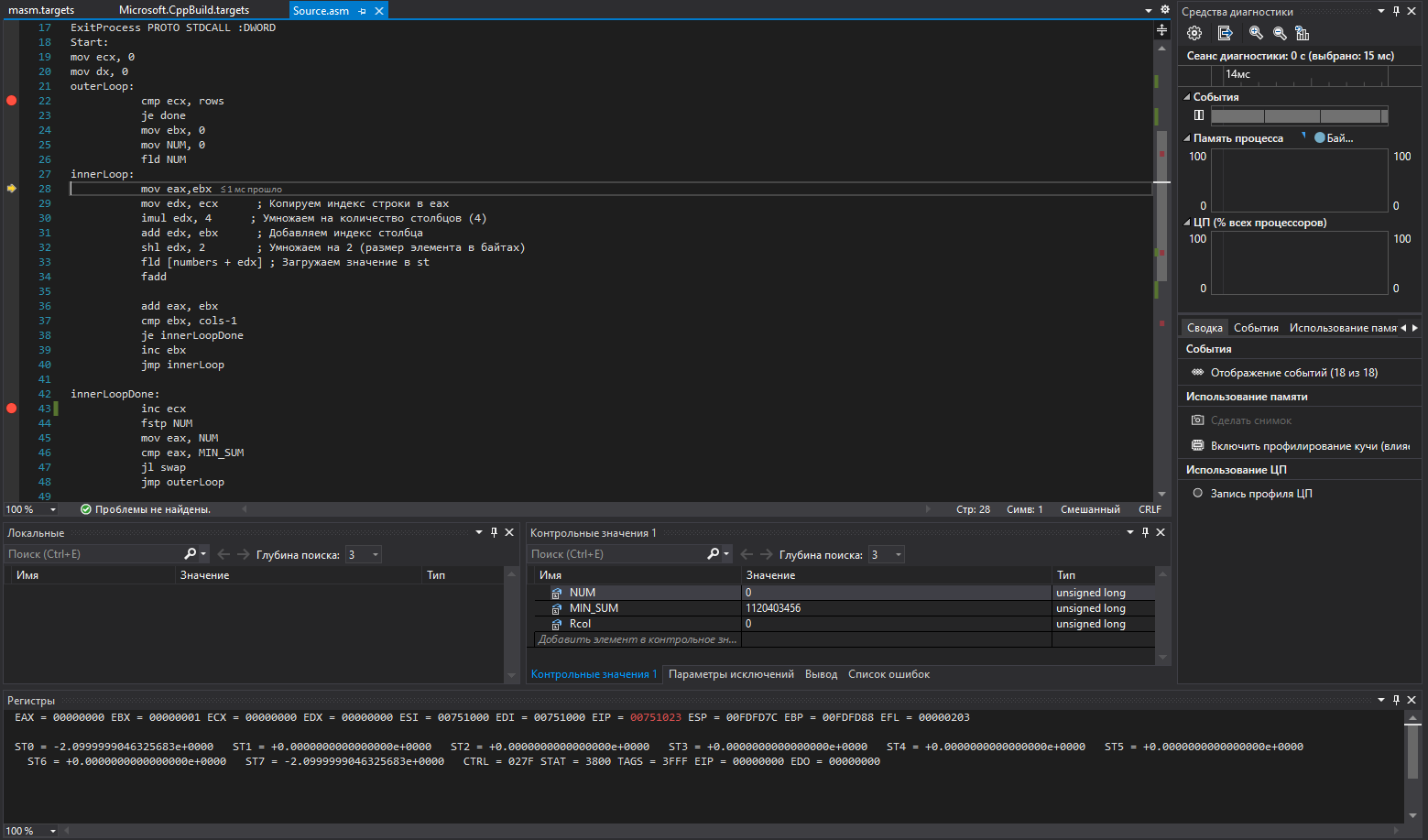


Рисунок 18 – Возврат к началу внутреннего цикла для дальнейшего сложения значений в строке

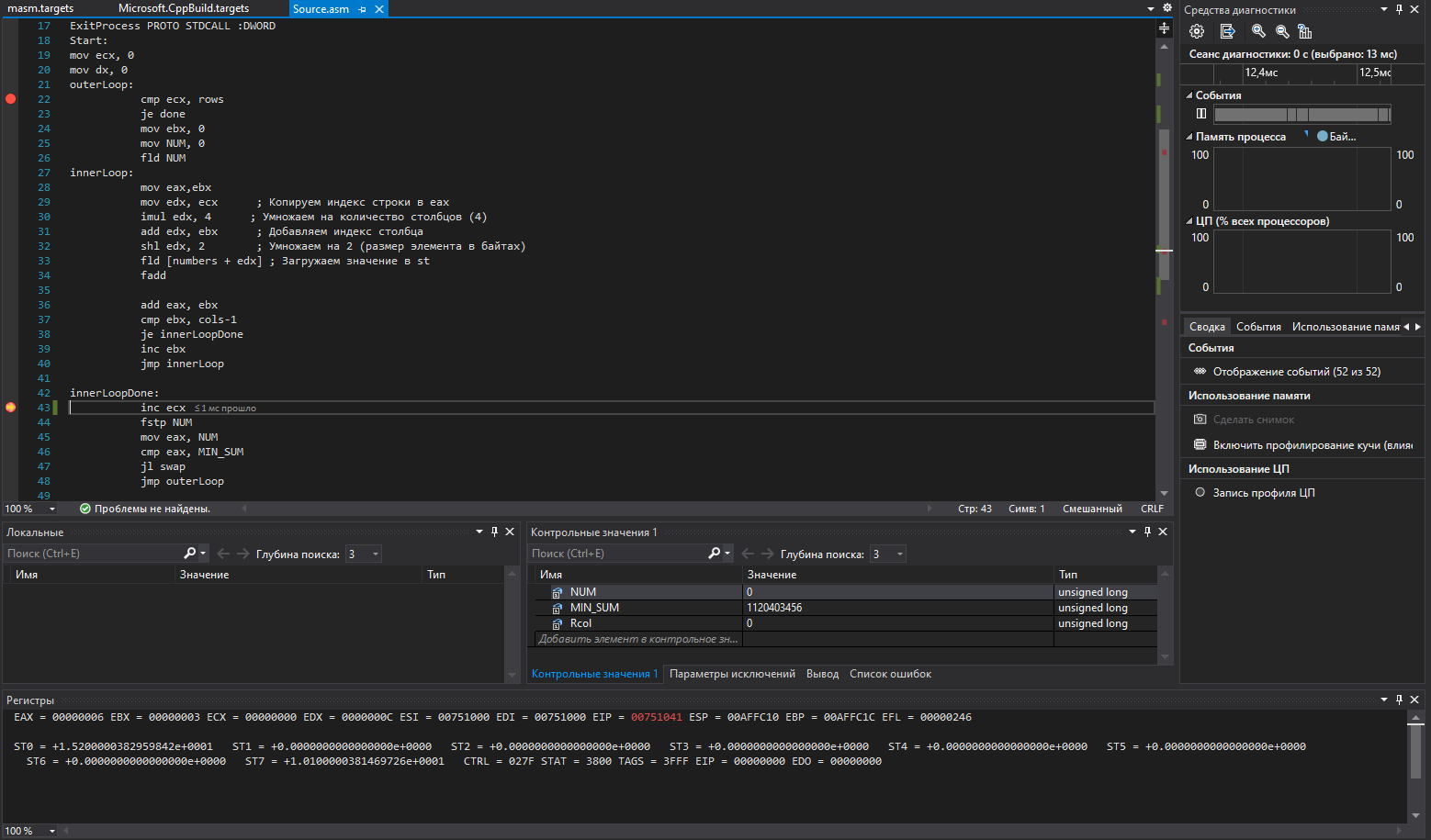


Рисунок 19 – Увеличиваем индекс строки

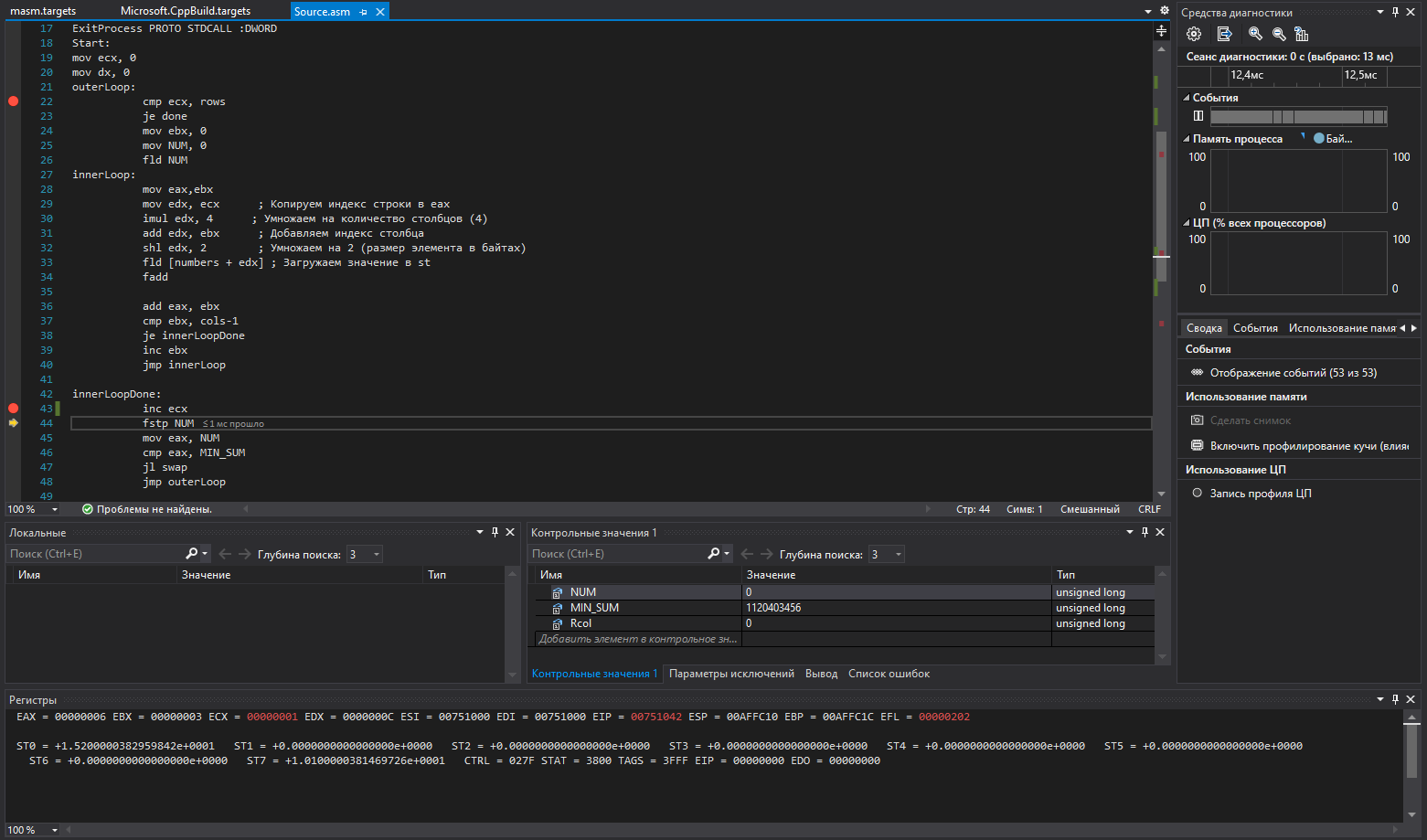


Рисунок 20 – Сохраняем сумму из стека FPU в переменную NUM

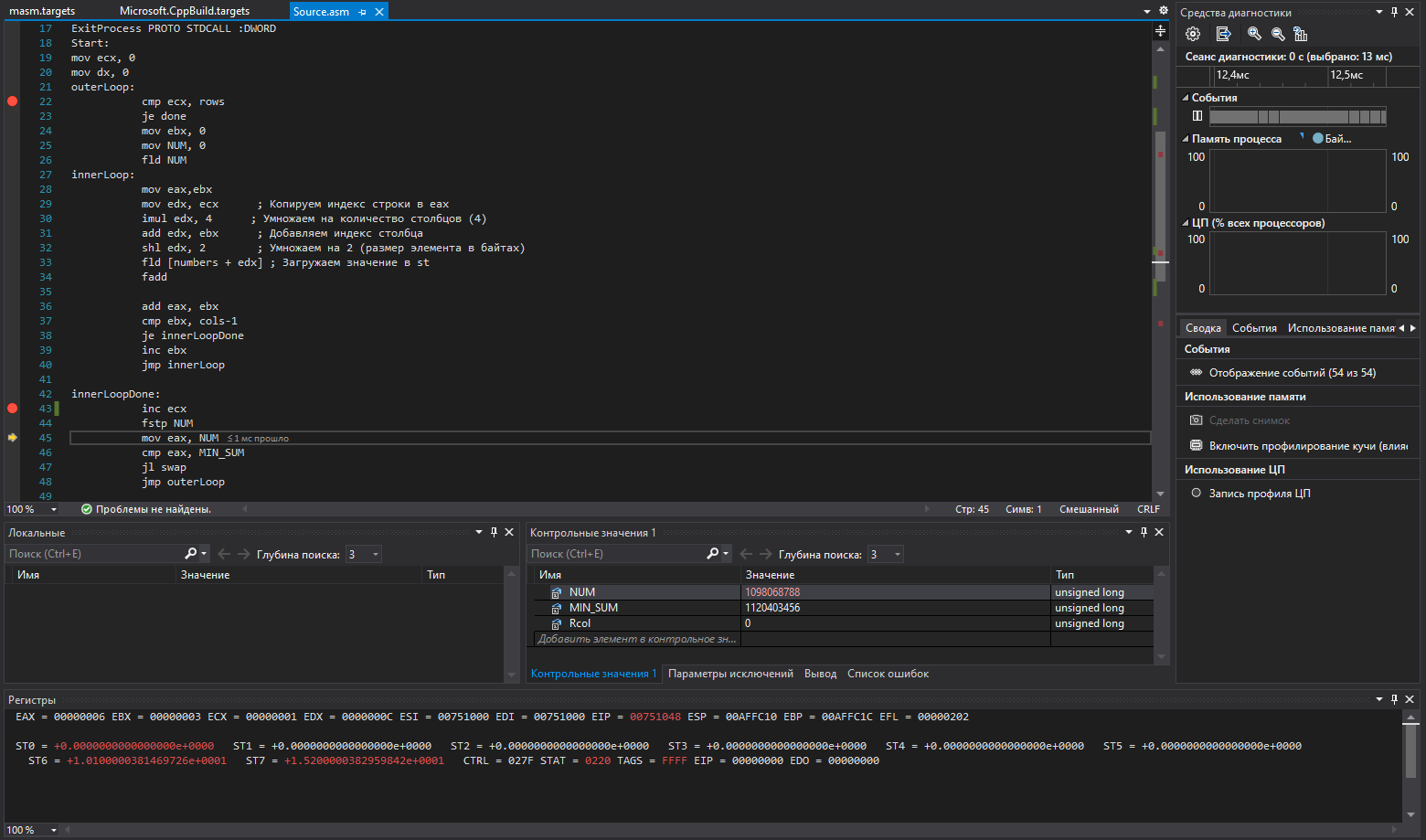


Рисунок 21 – Копируем сумму в eax для сравнения

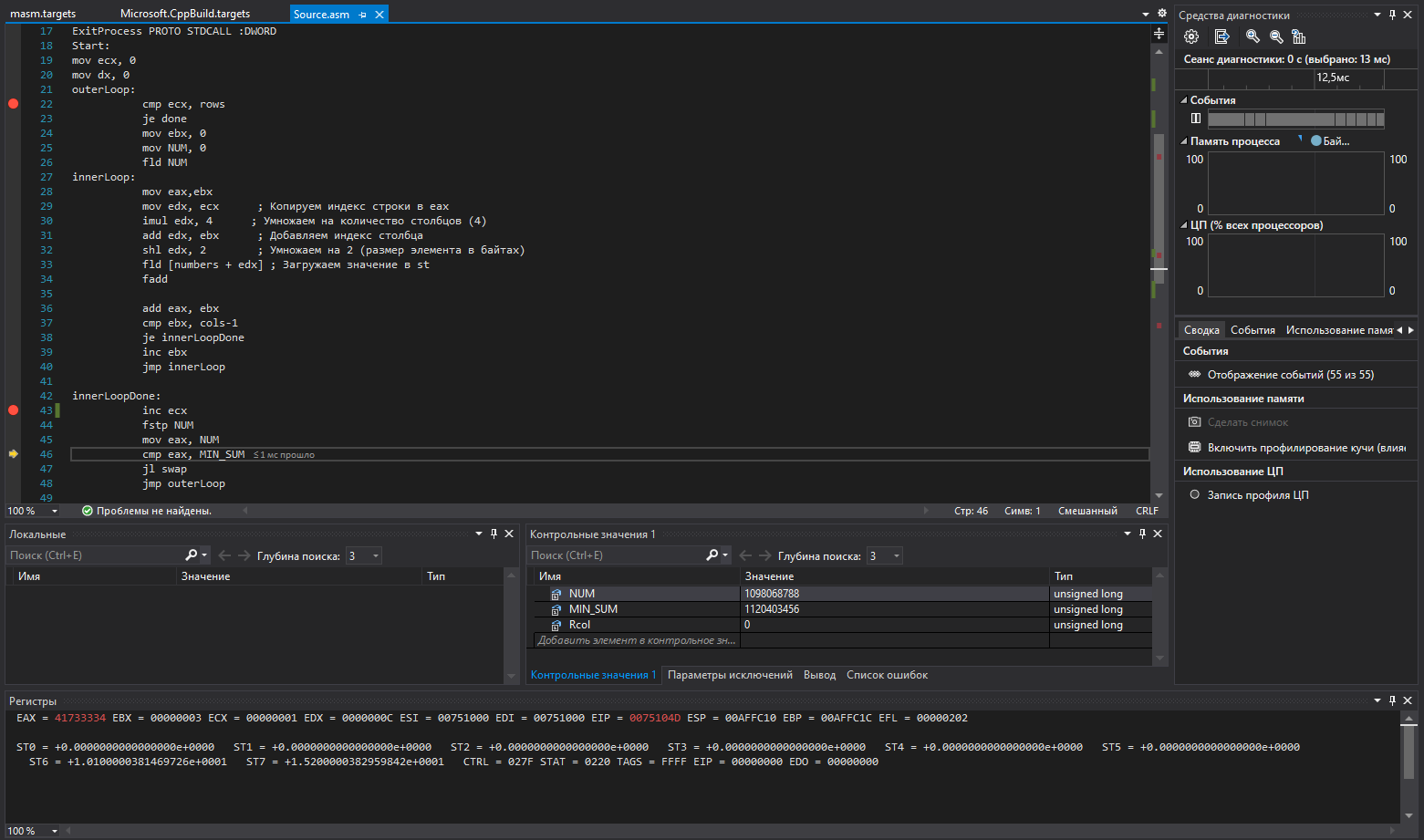


Рисунок 22 – Сравниваем сумму с минимальной суммой

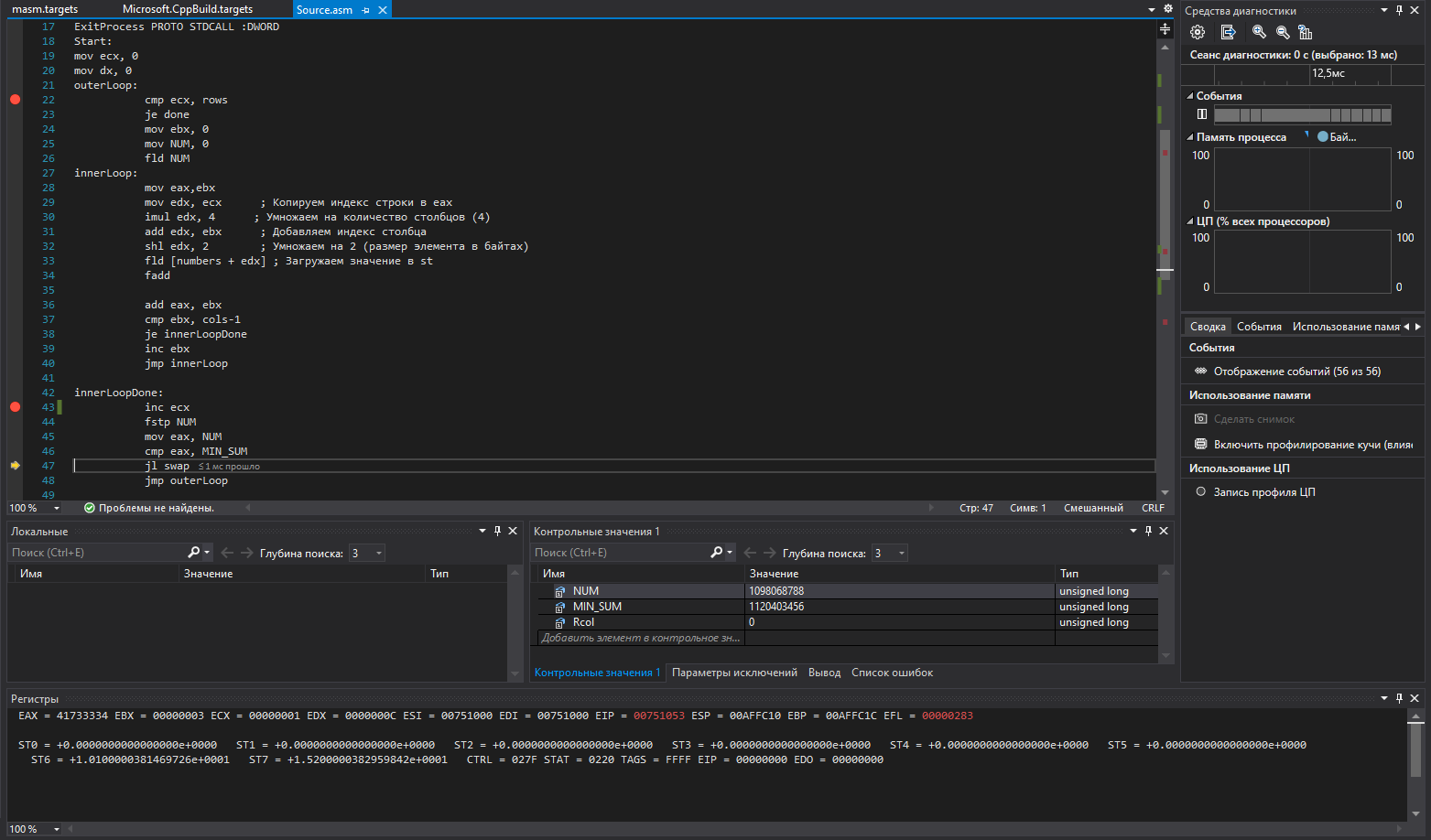


Рисунок 23 – Если сумма меньше, переходим к обмену

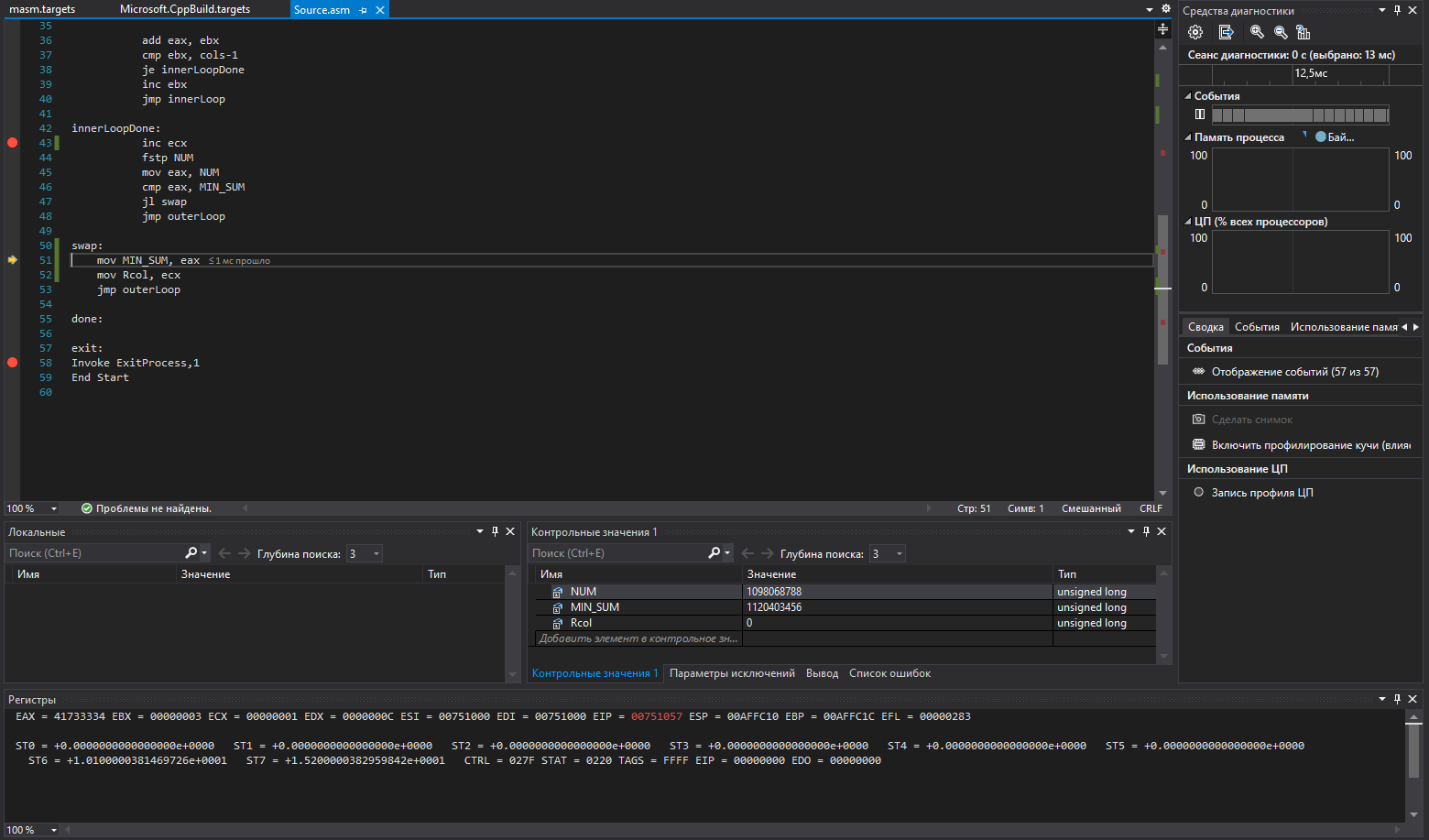


Рисунок 24 – Обновляем минимальную сумму

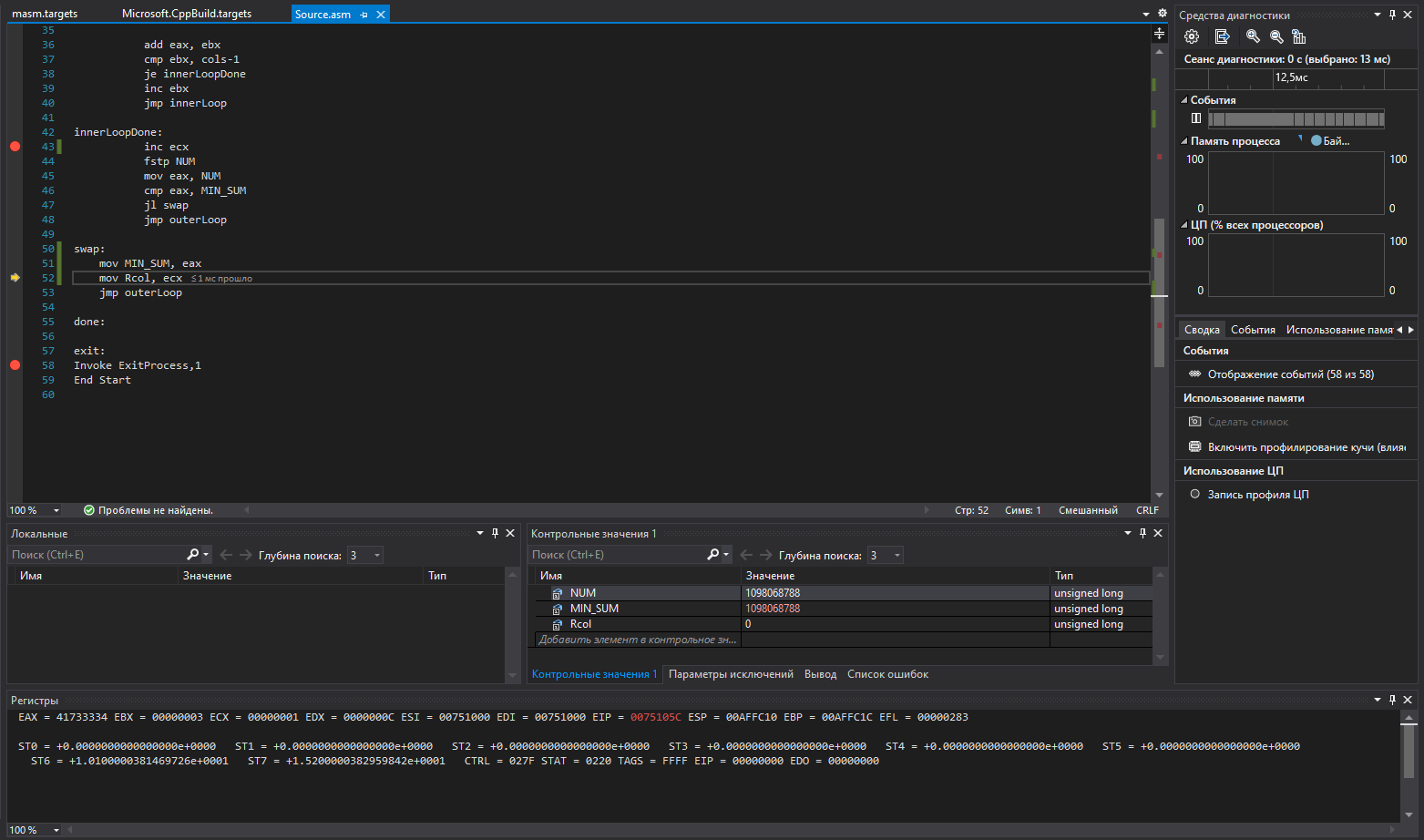


Рисунок 25 – Сохраняем индекс строки с минимальной суммой

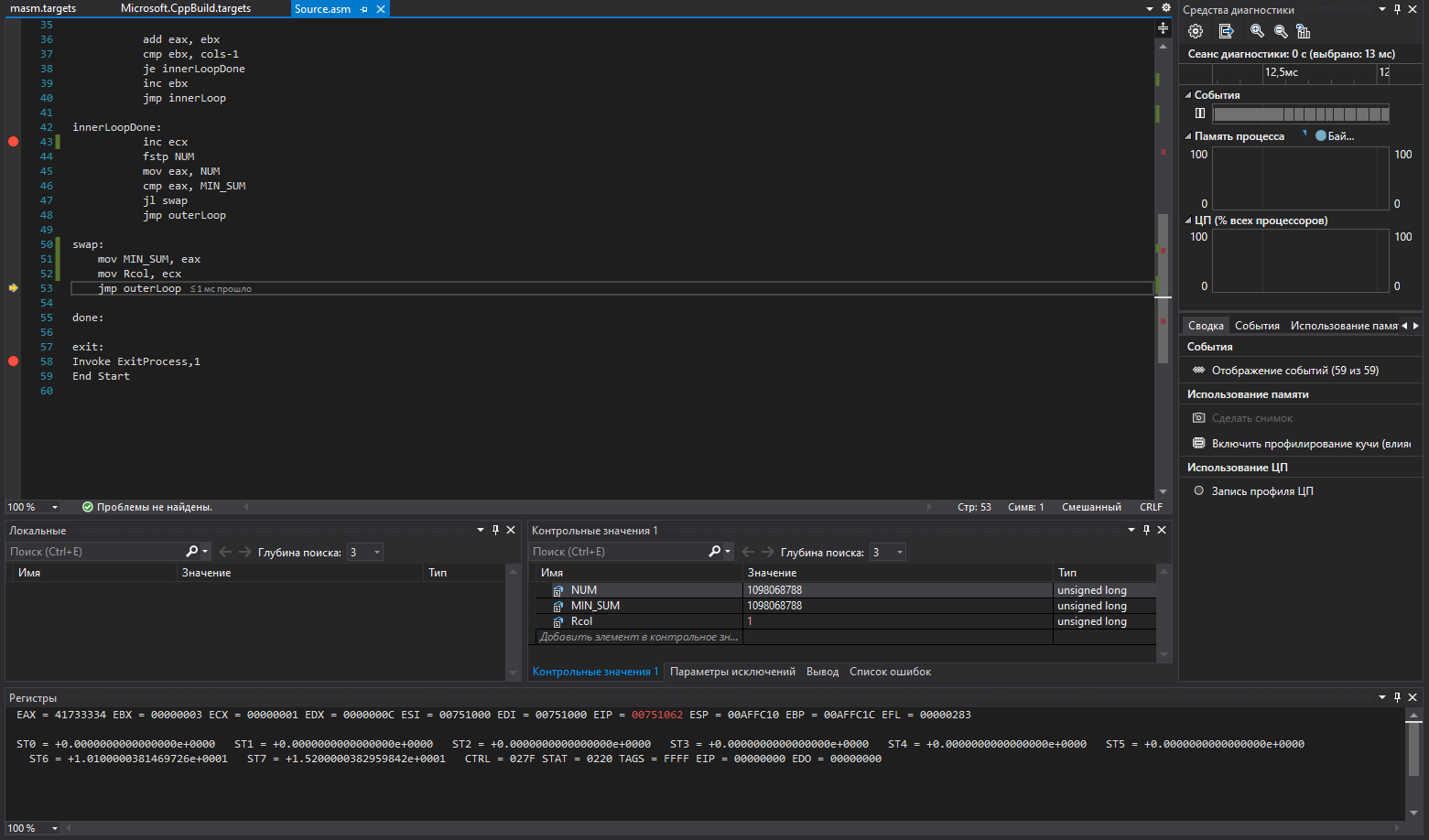


Рисунок 26 – Переход к следующей строке

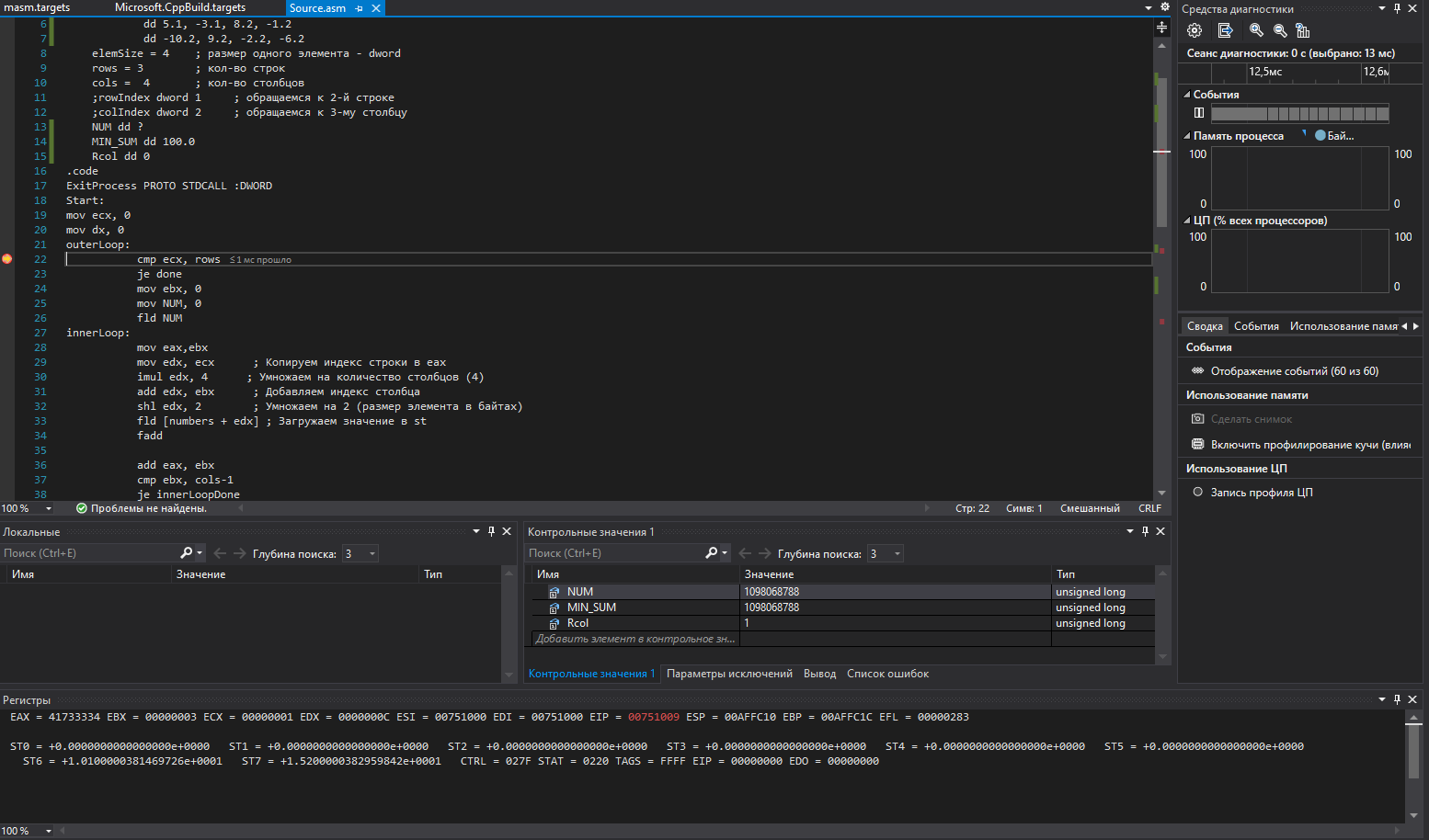


Рисунок 27 – Возврат к началу внешнего цикла для перехода к следующей строке и продолжению сложения значений

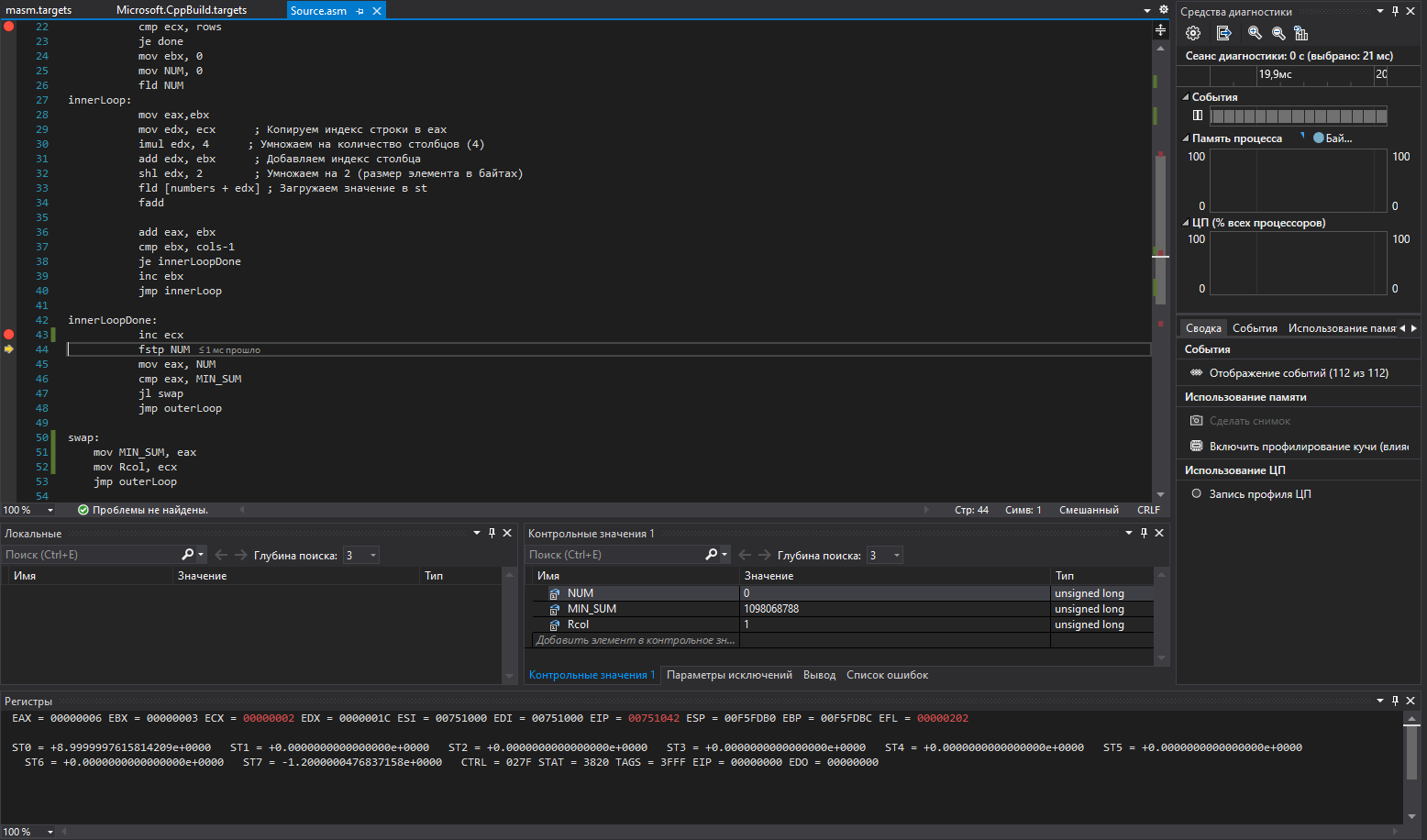


Рисунок 28 – Соответствие суммы значений во второй строке

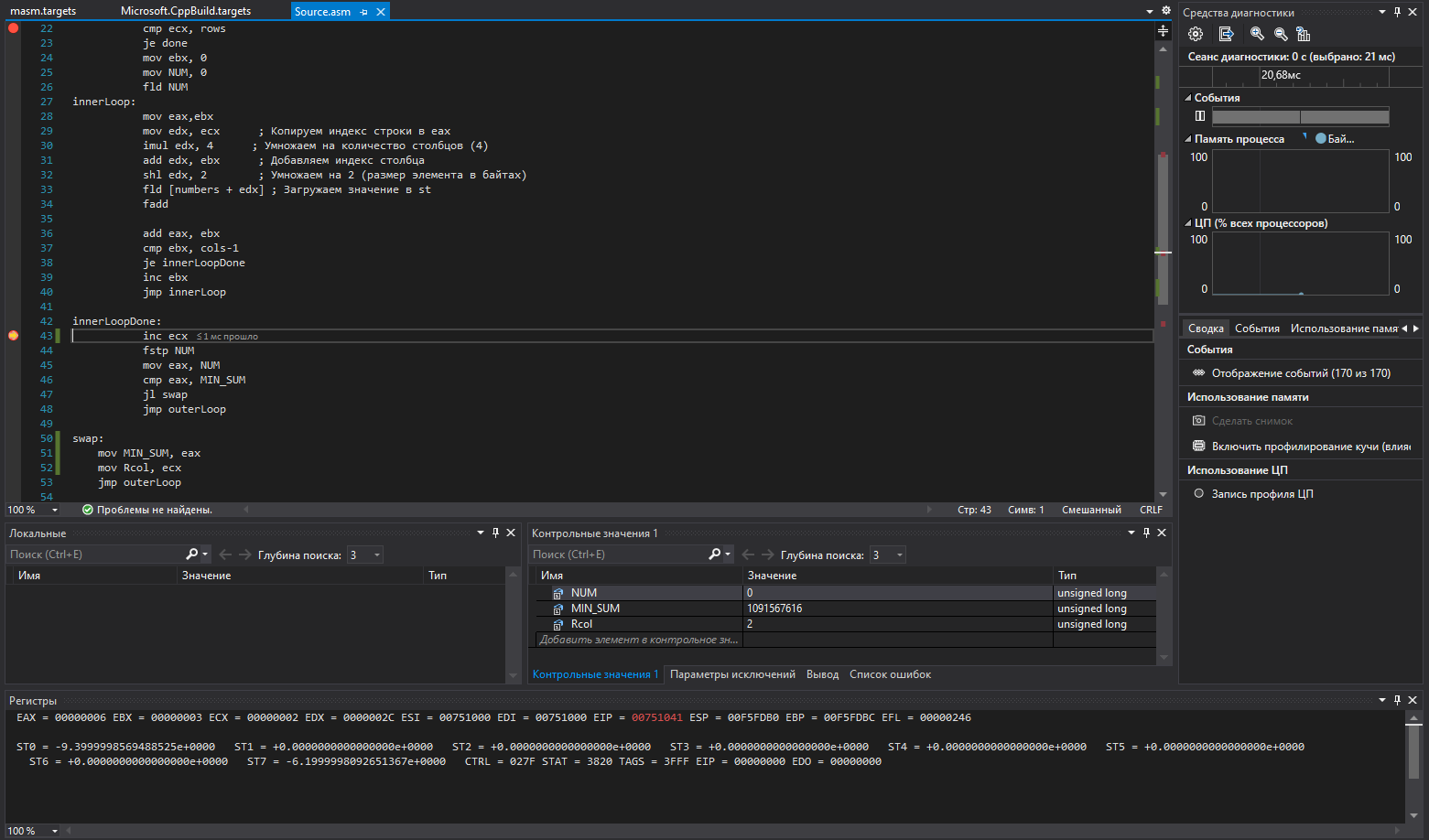


Рисунок 29 – Соответствие суммы значений в третьей строке

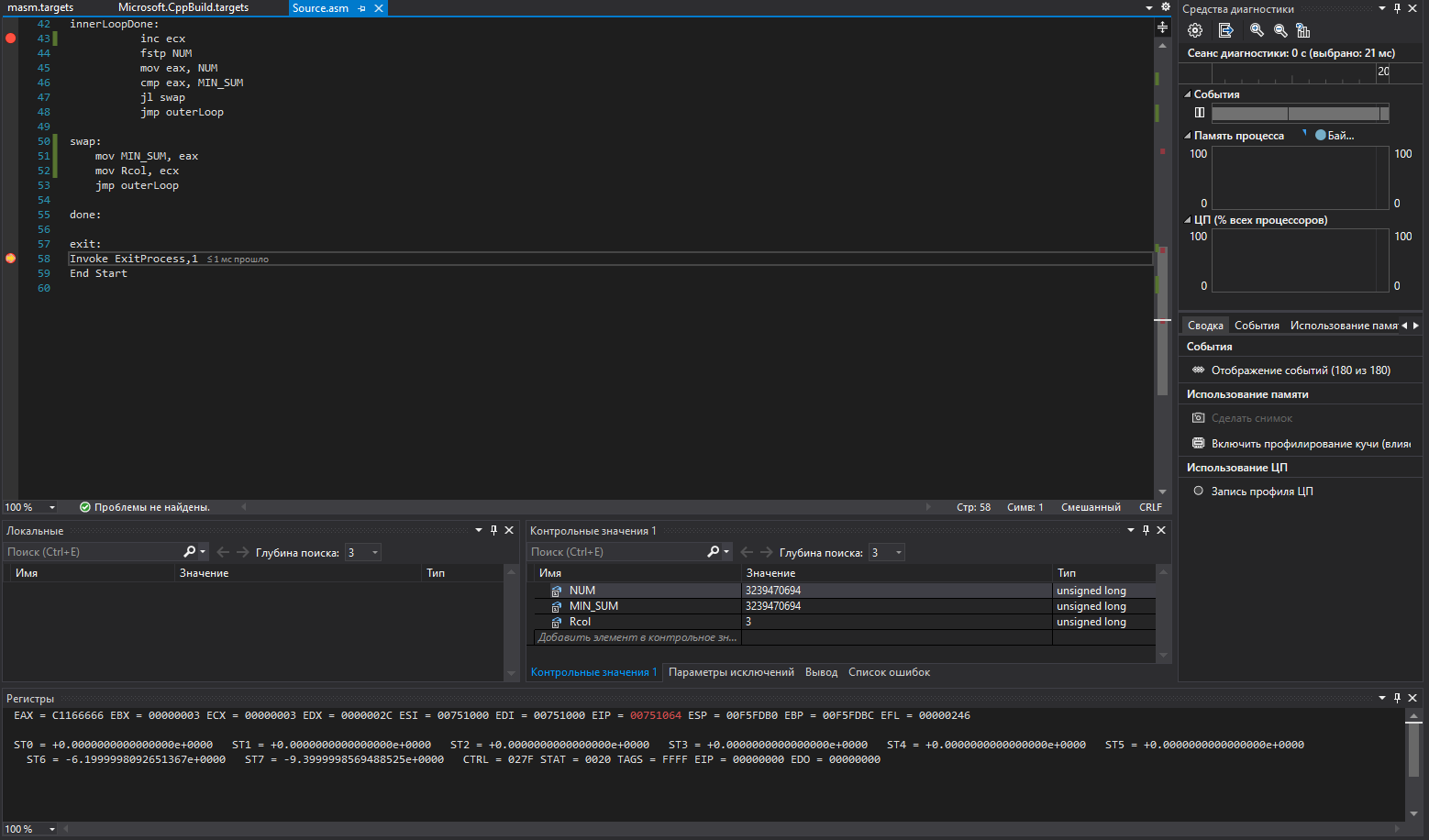


Рисунок 30 – Корректное определение номера строки с минимальной суммой при завершении программы (Rcol = 3)

**Вывод:** в ходе лабораторной работы была изучена реализация и выполнение арифметических команд с помощью математического сопроцессора FPU микропроцессоров архитектуры x86 на примере программы, вычисляющей сумму элементов строк матрицы и определяющей строку с минимальной суммой. В результате работы с кодом были усвоены следующие ключевые моменты:

Использование стековой архитектуры FPU для выполнения операций с плавающей точкой, что продемонстрировало особенности загрузки чисел в стек, их сложения и сохранения результата.

Работа с вещественными числами в памяти и их обработка с помощью инструкций FPU (fld, fadd, fstp), что позволило на практике понять, как реализуются арифметические операции с плавающей точкой на уровне ассемблера.