МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ АВТОМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

КАФЕДРА РАДИОЭЛЕКТРОННЫХ СРЕДСТВ

Отчет по дисциплине

«Цифровые устройства и микропроцессоры»

Лабораторная работа №3

«Использование математического сопроцессора»

Вариант №6

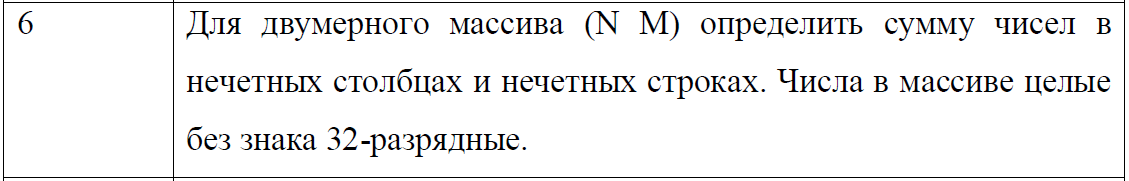
Выполнил: студент группы ИКТб– 31\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М.А. Дубровин

Проверил: доцент кафедры РЭС \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М.А. Земцов

#### Киров 2022

Цель работы: изучение принципов выполнения арифметических команд с помощью математического сопроцессора FPU микропроцессоров с архитектурой x86.

Задание:



Код программы:

.model flat,stdcall

.stack 100h

.data

array dd 17.0, 20.0, 30.0

dd 11.0, 24.0, 1.0

dd 2.0, 5.0, 6.7

dd 56.0, 27.0, 3.5

n dd 4;число строк

m dd 3;число столбцов

i dd 0;счётчик строк

j dd 0;счётчик столбцов

mysize dd 4;

odin dd 1;

SUM\_row dd 0.0;сумма чисел в нечётных строках

SUM\_col dd 0.0;сумма чисел в нечётных столбцах

.code

ExitProcess PROTO STDCALL :DWORD

Start:

finit;

;внешний цикл

L1:

mov j, 0;

xor eax, eax;

xor ebx, ebx;

xor ecx, ecx;

L2:

mov eax, i;

and eax,odin;

cmp eax, odin;

je L3;

jne L4;

L3:

xor eax, eax;

;address calculate

mov eax, m;

mul i;

add eax, j;

mul mysize;

;use eax

fld array[eax];

fld SUM\_row;

fadd;

fstp SUM\_row;

L4:

mov eax, j;

and eax, odin;

cmp eax, odin;

je L5;

jne L6;

L5:

xor eax, eax;

;address calculate

mov eax, m;

mul i;

add eax, j;

mul mysize;

;use eax

fld array[eax];

fld SUM\_col;

fadd;

fstp SUM\_col;

L6:

inc j;

mov ecx, j;

mov ebx, m;

cmp ecx, ebx;

jne L2;

inc i;

mov ecx, i;

mov ebx, n;

cmp ecx, ebx;

jne L1;

fld SUM\_row;

fld SUM\_col;

fstp SUM\_col;

fstp SUM\_row;

exit:

Invoke ExitProcess,1

End Start

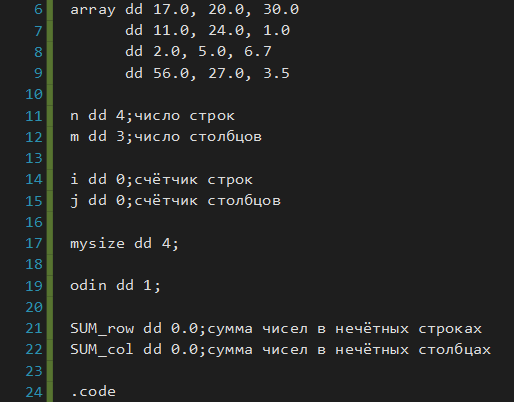
Входная матрица:

|  |  |  |
| --- | --- | --- |
| 17.0 | 20.0 | 30.0 |
| 11.0 | 24.0 | 1.0 |
| 2.0 | 5.0 | 6.7 |
| 56.0 | 27.0 | 3.5 |

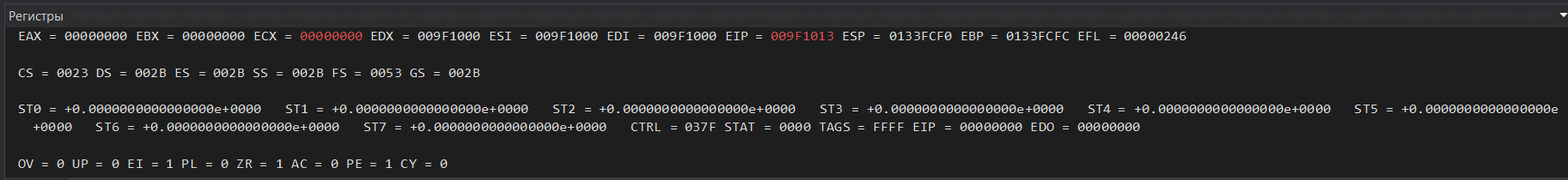
Сумма чисел в нечётных строках: 11.0 + 24.0 + 1.0 + 56.0 + 27.0 + 3.5 = 122.5

Сумма чисел в нечётных столбцах: 20.0 + 24.0 + 5.0 + 27.0 = 76

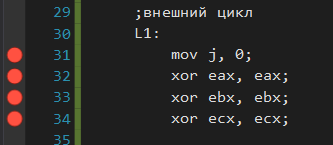
Работа программы:



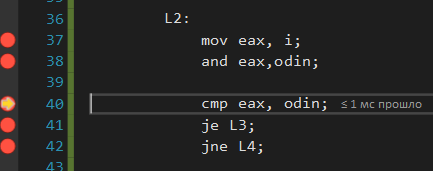
Инициализация данных



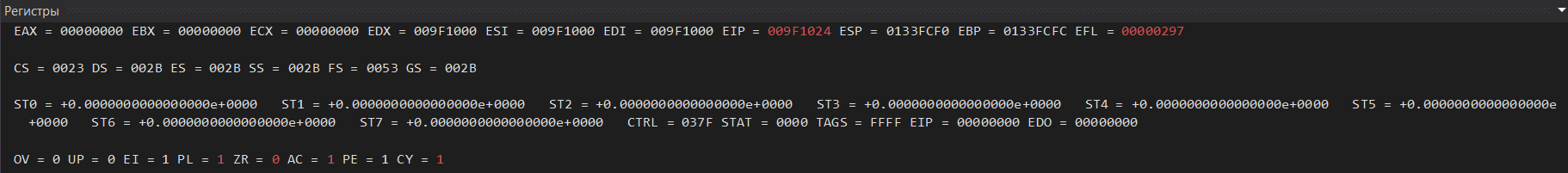
Обнуляем регистры eax, ebx, ecx



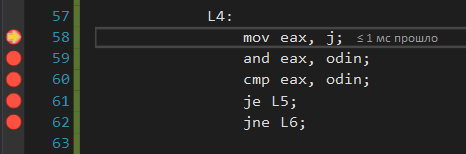
Начало внешнего цикла

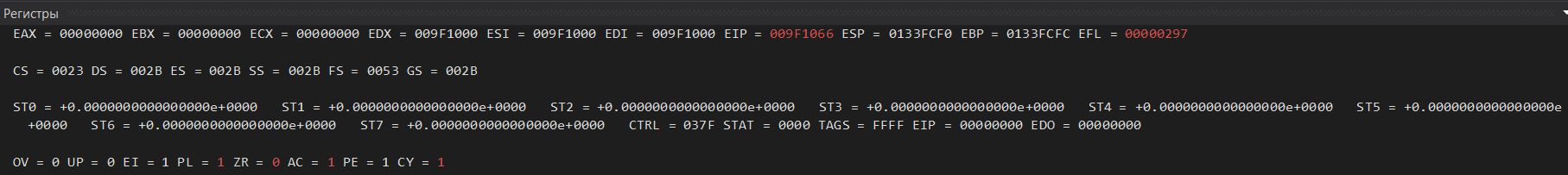


Проверяем индекс строки на чётность



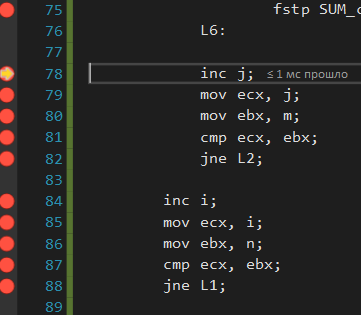
Так как индекс 0 – чётный, то переход к метке L4



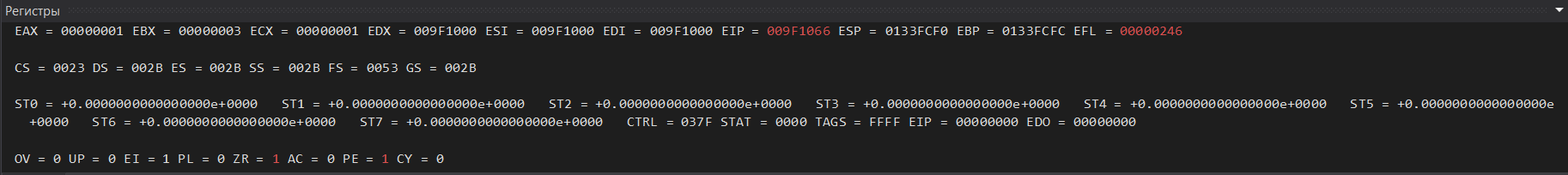


Проверили индекс столбца на чётность

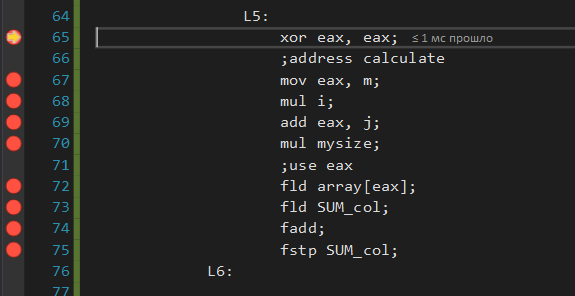
Так как индекс 0 – чётный то переходим к метке L6

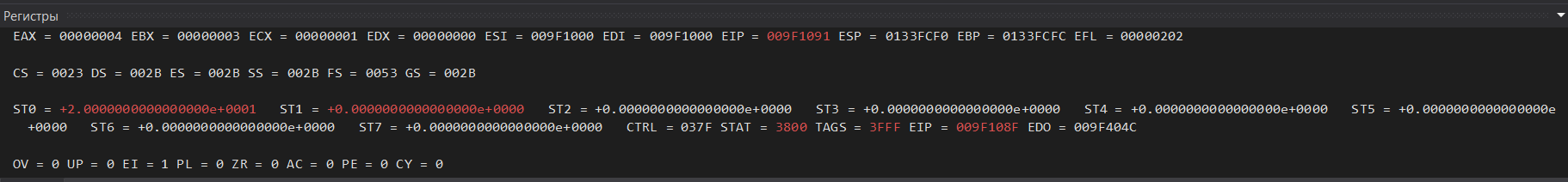


Увеличиваем счётчик столбца на 1, проверяем условие 1 < 3, поэтому продолжаем внутренний цикл

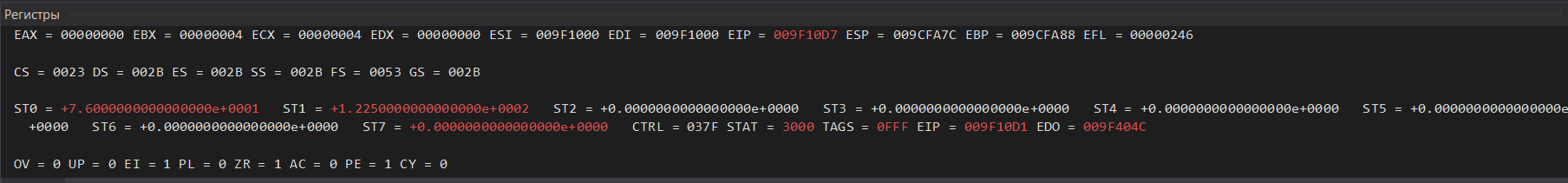


На второй итерации внутреннего цикла j = 1, eax = 1; нечётно. Поэтому складываем SUM\_col с текущим элементом массива





Складываем и помещаем результат в SUM\_col



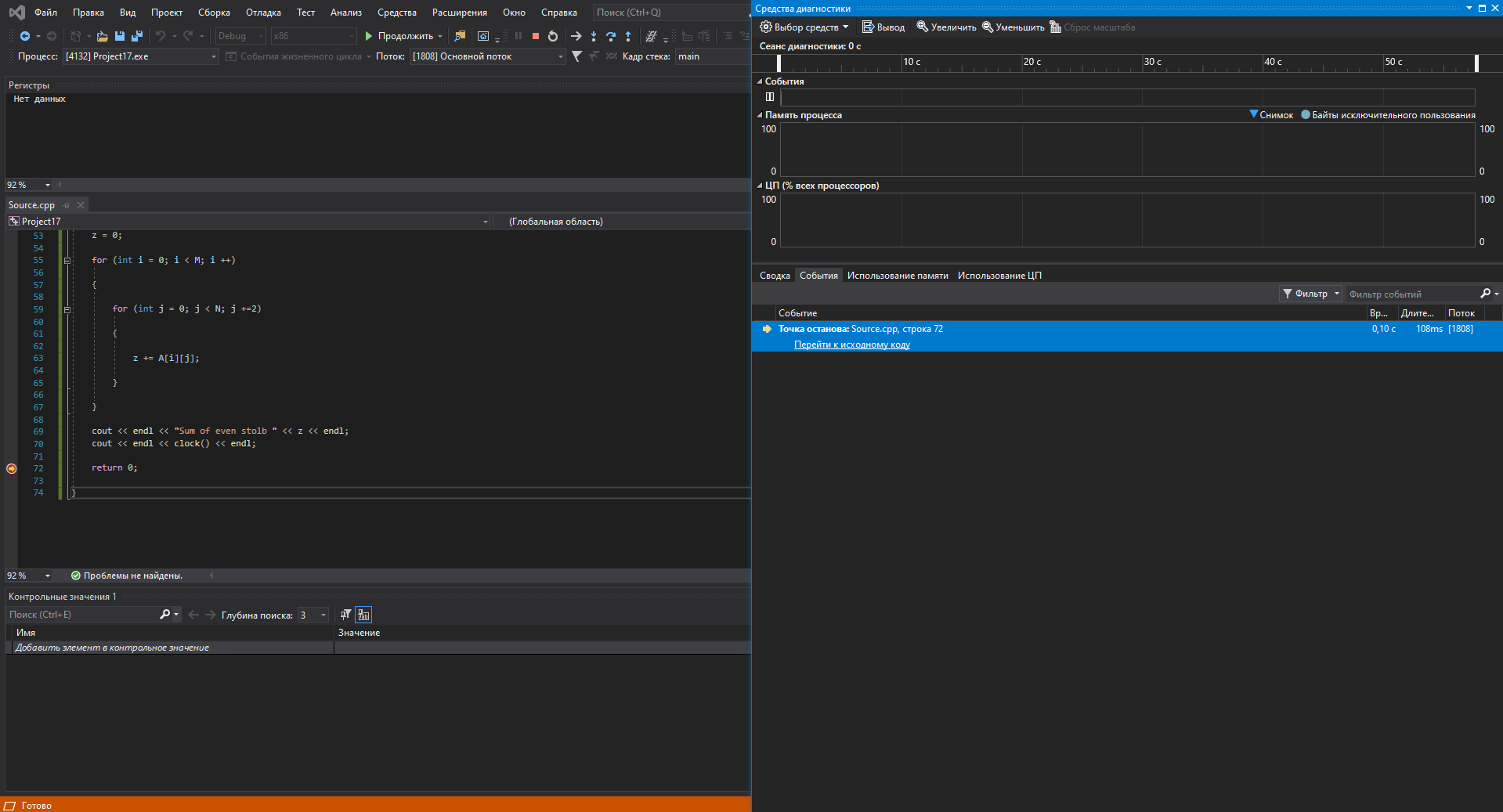
После выполнения всех итераций цикла

В st(1) поместили SUM\_row и в st(0) SUM\_col для наглядности.

Как можно увидеть значение в st(0) = 76 – совпадает с рассчитанной вручную суммой в нечётных столбцах и в st(1) = 122.5 – совпадает с рассчитанной вручную суммой в нечётных строках.

**Вывод:** в ходе лабораторной работы были изучены операции, которые можно выполнять с помощью математического сопроцессора. Программа прошла верификацию ручным расчетом, а это значит, что она выполнена правильно.

Защита:



Код:

#include <iostream>

#include <ctime>

using namespace std;

int main()

{

srand(time(NULL));

unsigned long S = 0;

const int N = 4; //Для изменения массива изменять это число

const int M = 3;

int A[N][M];

for (int i = 0; i < M; i++)

{

for (int j = 0; j < N; j++)

{

A[i][j] = rand() % 100;

cout << A[i][j] << " ";

}

cout << endl;

}

for (int i = 0; i < M; i += 2)

{

for (int j = 0; j < N; j++)

{

S += A[i][j];

}

}

cout << endl << "Sum of even lines " << S << endl;

int z;

z = 0;

for (int i = 0; i < M; i ++)

{

for (int j = 0; j < N; j +=2)

{

z += A[i][j];

}

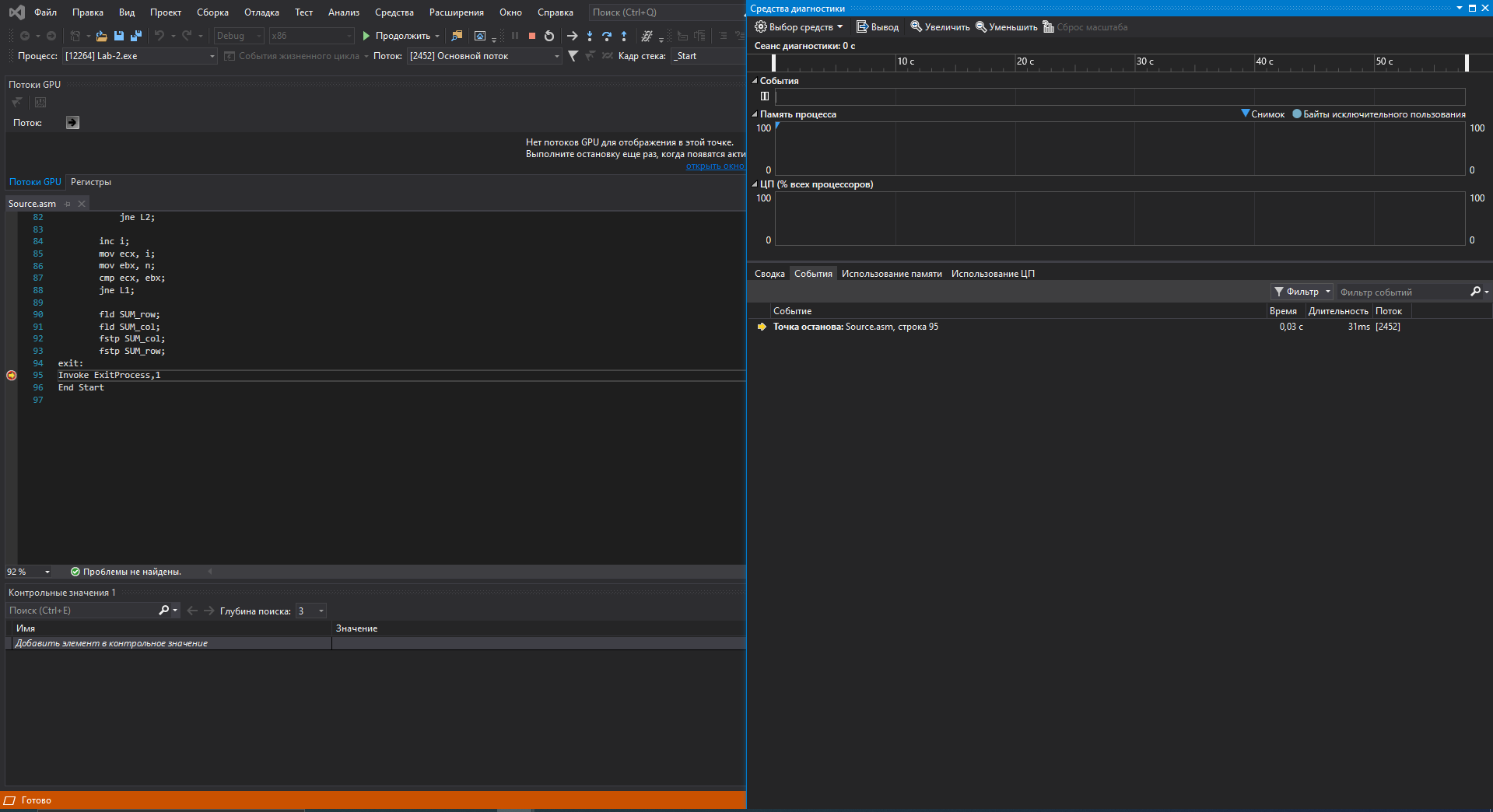
}

cout << endl << "Sum of even stolb " << z << endl;

cout << endl << clock() << endl;

return 0;

}



Время ассемблера

Вывод:

Языки программирования имеют избыточность по отношению к ассемблеру, компилятору приходится выполнять преобразования, в то время как ассемблер наиболее близок к двоичному исполняемому коду, который непосредственно исполняется в процессоре. соотвественно есть огромная куча возможностей как сделать сделать идеально подходящий к данному процессору код и соответственно самый быстрый. Но и ошибки, допускаемые при программировании на ассемблере, могут оказаться более критичными.

компилятор более высокого чем ассемблер языка выбирает для некоторого выражения на языке высокого уровня подходящий набор команд процессора и вставляет в конечный исполняемый файл. и этот подбор может быть не всегда идеальный.