МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

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 микропроцессоры»

Лабораторная работа №3

«ИСПОЛЬЗОВАНИЕ МАТЕМАТИЧЕСКОГО СОПРОЦЕССОРА»

Вариант №3

|  |  |  |
| --- | --- | --- |
| Выполнил: студент гр. ИКТб-3301-04-00 | \_\_\_\_\_\_\_\_\_\_\_\_ | К.Д. Важенин |
| Проверил: доцент кафедры РЭС | \_\_\_\_\_\_\_\_\_\_\_\_ | М.А. Земцов |

#### Киров 2023

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

**Ход работы:**

**1**.Исходные данные представлены в таблице 1.

Таблица 1 – Исходные данные

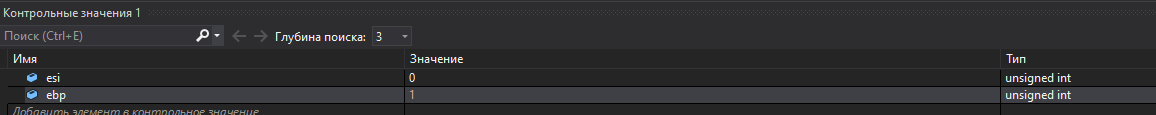
Изображение выглядит как текст

Автоматически созданное описание

**2**. Первым действием запишем счетчик внутри массива и номер строки с наименьшим элементом (здесь по умолчанию 1):

mov esi,0

mov ebp, 1



Теперь записываем инициализация сопроцессора. Потом запишем счетчик строки и начало алгоритма L1. Далее записывается первый элемент массива в вершину стека:

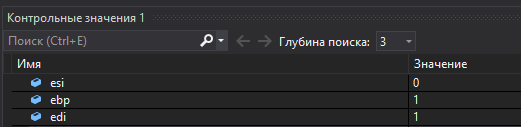
finit

mov edi, 1

fld tmp

L1:

fld mass[4\*esi]



Потом начинается цикл по элементам строки и сложение элементов в каждой строке:

mov cl, 5

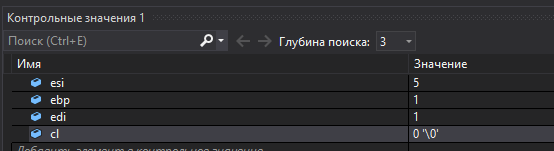
L2:

INC esi

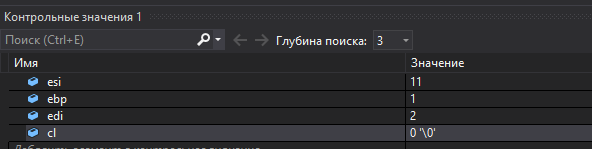
fadd mass[4\*esi]

DEC cl

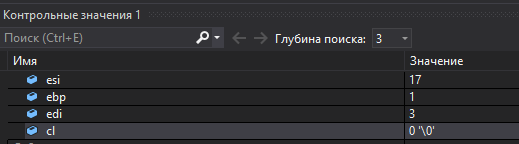
JNZ L2



Результат суммы элементов первой строки



Результат суммы элементов второй строки



Результат суммы элементов третьей строки

Далее сравнение суммы элементов между строк:

fcom st(1)

fstsw ax

sahf

ja more

jb less

je equal

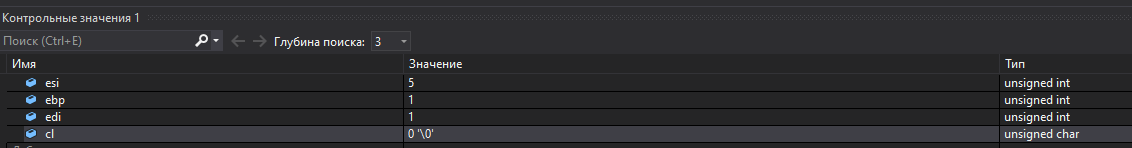
more:

fstp st(0)

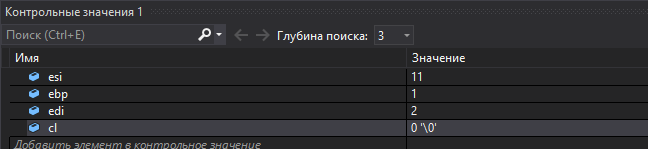
jmp end\_algorithm

less:

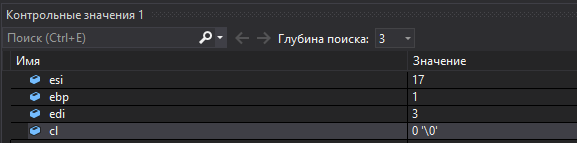
fstp st(1)



Сравнение ST0 и ST1



Сравнение суммы элементов первой и второй строк

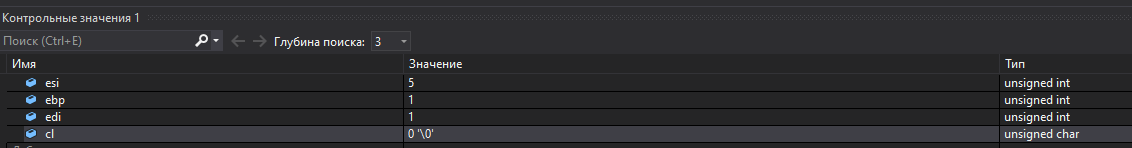


Результат сравнения регистров ST0 и ST1

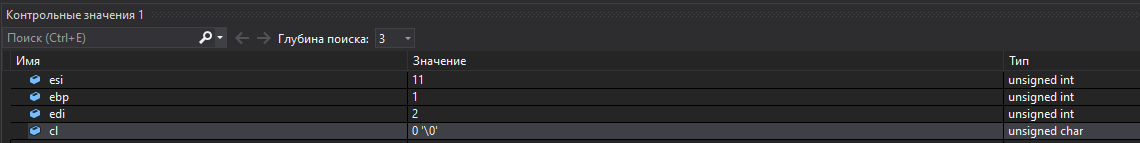
Далее определение номера строки с наименьшей суммой элементов:

mov ebp, edi

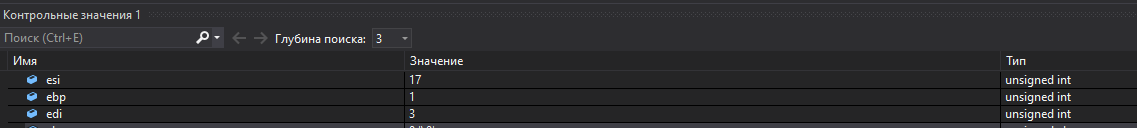
jmp end\_algorithm



Определение номера строки с наименьшей суммой элементов (1ая строка)



При сравнение первой и второй строкой, первая строка остается с наименьшей суммой элементов



Определение номера строки с наименьшей суммой элементов (1ая строка)

Далее запишем конец итерации и цикла по строкам:

equal:

fstp st(0)

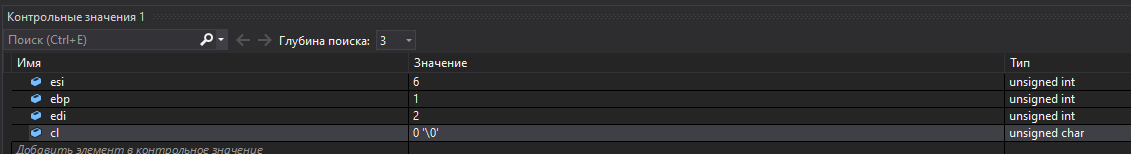
end\_algorithm:

INC esi

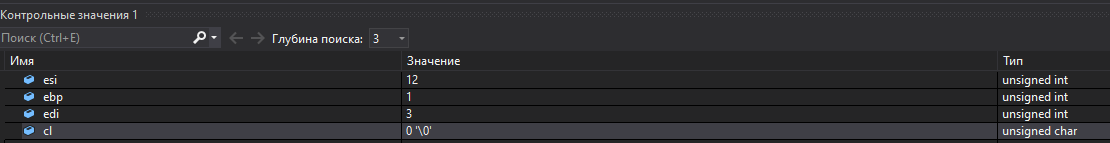
inc edi

cmp edi, 4

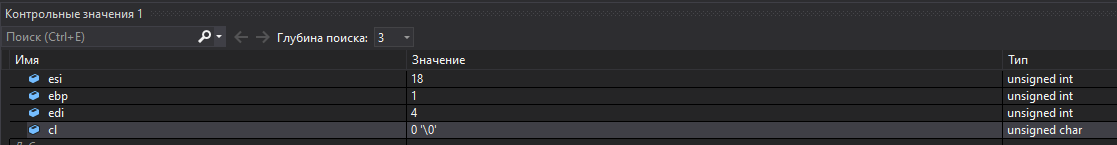
JNZ L1



Конец первой итерации алгоритма



Конец второй итерации алгоритма

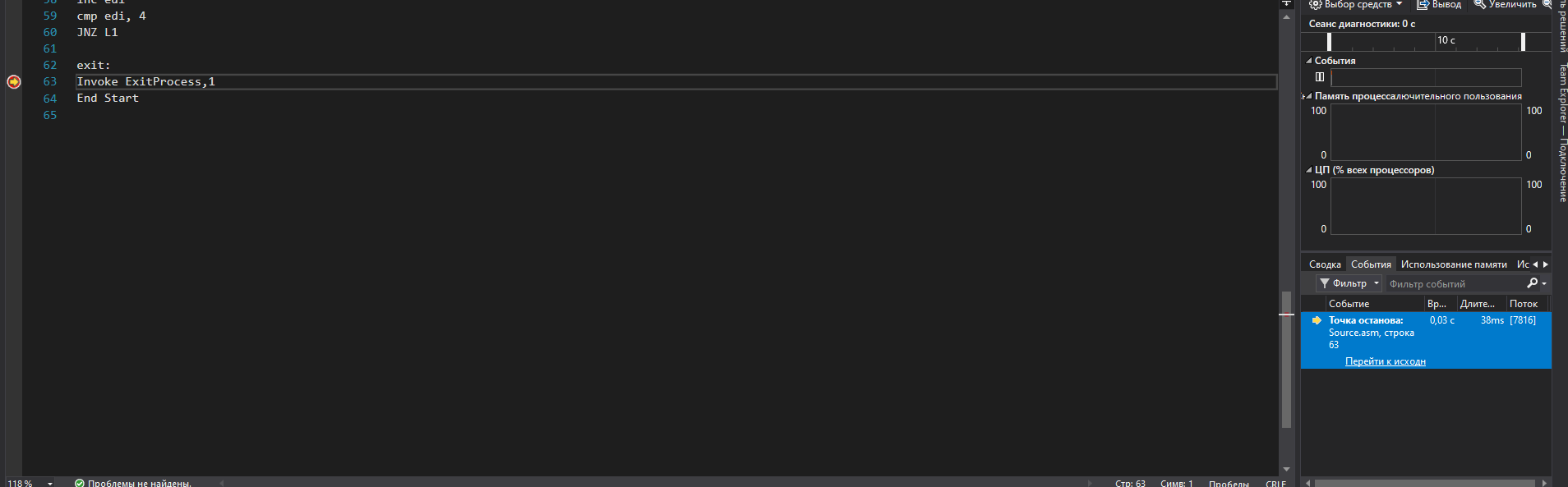


Конец цикла

***Вывод:*** в ходе лабораторной работы была изучена принципы выполнения команд с помощью математического сопроцессора FPU, такие команды как: finit, fld, fadd, fcom, fstsw, fstp.

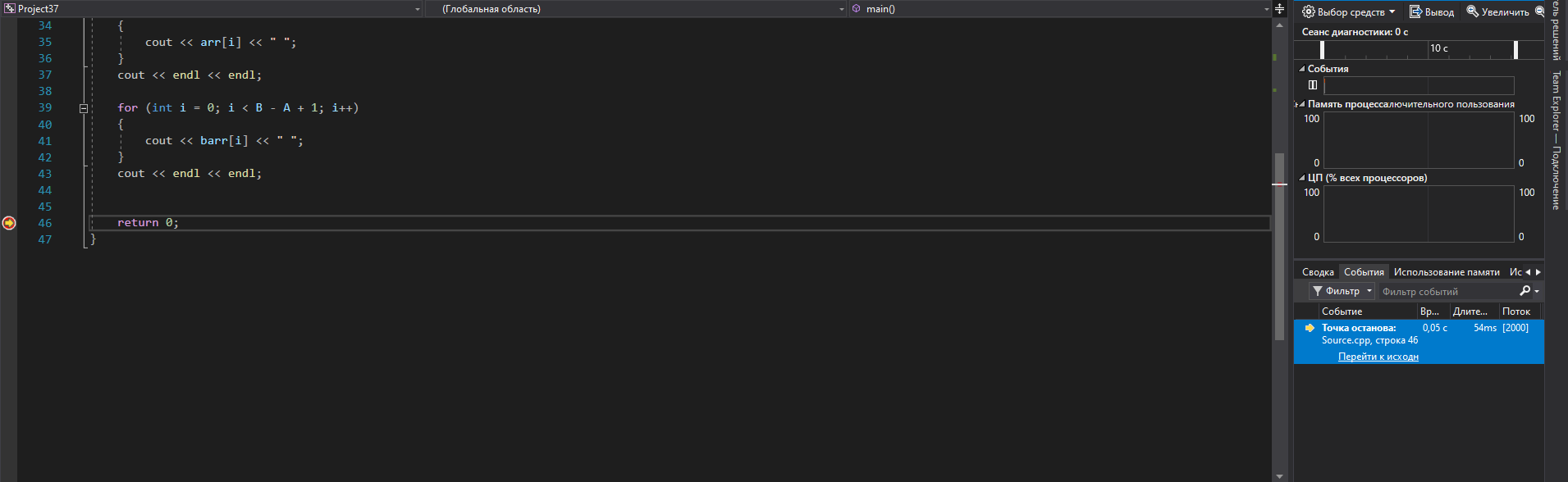
**2.1 Результаты работы программы на языке С**

1. Измерение скорости работы на языке Assambler



Скорость работы равна 38 мс

1. Измерение скорости работы на языке C



Скорость работы равна 54 мс

Вывод:

Таким образом время работы программы на Ассемблере в 1,42 раза быстрее, чем время работы на языке С.

**3. Код программы:**

.686

.model flat,stdcall

.stack 100h

.data

mass dd 5.532, 0.621 , 0.564 , -32.772 , -1.32, 0.511 ;

dd -0.4, 5.3,-8.2, -1.4189, 0.4444, 0.4914 ;

dd -0.3, 21.21, 0.3, 5.21, -7.32, 10.3 ;

tmp dd 1.7e38

.code

ExitProcess PROTO STDCALL :DWORD

Start:

mov esi,0

mov ebp, 1

finit

mov edi, 1

fld tmp

L1:

fld mass[4\*esi]

mov cl, 5

L2:

INC esi

fadd mass[4\*esi]

DEC cl

JNZ L2

fcom st(1)

fstsw ax

sahf

ja more

jb less

je equal

more:

fstp st(0)

jmp end\_algorithm

less:

fstp st(1)

mov ebp, edi

jmp end\_algorithm

equal:

fstp st(0)

end\_algorithm:

INC esi

inc edi

cmp edi, 4

JNZ L1

exit:

Invoke ExitProcess,1

End Start

*Код на С++ :*

#include <stdlib.h>

#include <time.h>

#include <iostream>

#include "windows.h"

using namespace std;

int main()

{

int A;

int B;

int D;

int arr[100];

srand(time(0));

for (int i = 0; i < 100; i++)

{

arr[i] = rand() % 101;

cout << arr[i] << " ";

}

cout << endl;

cout << "\nA = ";

cin >> A;

cout << "B = ";

cin >> B;

//-------------------------------------------

int\* barr = new int[ B - A + 1];

for (int i = 0; i < B - A + 1; i++)

{

int x = arr[A+i];

if (x > 0) barr[i] = x \* x;

else if (x >= -1) barr[i] = x;

else barr[i] = 2 \* sqrt(abs(x - 6));

}

//-------------------------------------------

for (int i = A; i <= B; i++)

{

cout << arr[i] << " ";

}

cout << endl << endl;

for (int i = 0; i < B - A + 1; i++)

{

cout << barr[i] << " ";

}

cout << endl << endl;

return 0;

}

***Дизассемблированный код:***

.686

.model flat,stdcall

.stack 100h

.data

mass dd 5.532, 0.621 , 0.564 , -32.772 , -1.32, 0.511 ;

dd -0.4, 5.3,-8.2, -1.4189, 0.4444, 0.4914 ;

dd -0.3, 21.21, 0.3, 5.21, -7.32, 10.3 ;

tmp dd 1.7e38

.code

ExitProcess PROTO STDCALL :DWORD

Start:

mov esi,0

00B51000 mov esi,0

mov ebp, 1

00B51005 mov ebp,1

finit

00B5100A wait

00B5100B fninit

mov edi, 1

00B5100D mov edi,1

fld tmp

00B51012 fld dword ptr [tmp (0B54048h)]

L1:

fld mass[4\*esi]

00B51018 fld dword ptr mass (0B54000h)[esi\*4]

mov cl, 5

00B5101F mov cl,5

L2:

INC esi

00B51021 inc esi

fadd mass[4\*esi]

00B51022 fadd dword ptr mass (0B54000h)[esi\*4]

DEC cl

00B51029 dec cl

JNZ L2

00B5102B jne \_Start+21h (0B51021h)

fcom st(1)

00B5102D fcom st(1)

fstsw ax

00B5102F wait

00B51030 fnstsw ax

sahf

00B51032 sahf

ja more

00B51033 ja \_Start+39h (0B51039h)

jb less

00B51035 jb \_Start+3Dh (0B5103Dh)

je equal

00B51037 je \_Start+43h (0B51043h)

more:

fstp st(0)

00B51039 fstp st(0)

jmp end\_algorithm

00B5103B jmp \_Start+45h (0B51045h)

less:

fstp st(1)

00B5103D fstp st(1)

mov ebp, edi

00B5103F mov ebp,edi

jmp end\_algorithm

00B51041 jmp \_Start+45h (0B51045h)

equal:

fstp st(0)

00B51043 fstp st(0)

end\_algorithm:

INC esi

00B51045 inc esi

inc edi

00B51046 inc edi

cmp edi, 4

00B51047 cmp edi,4

JNZ L1

00B5104A jne \_Start+18h (0B51018h)

exit:

Invoke ExitProcess,1

00B5104C push 1

00B5104E call \_ExitProcess@4 (0B51067h)