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 12

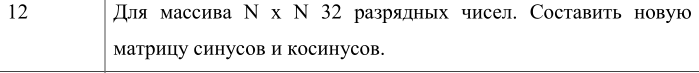
Выполнил: студент группы ИКТб-33 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Д.А. Огородов/

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.

**Индивидуальное задание:**



Для проверки результата составим проверочную программу на языке С++

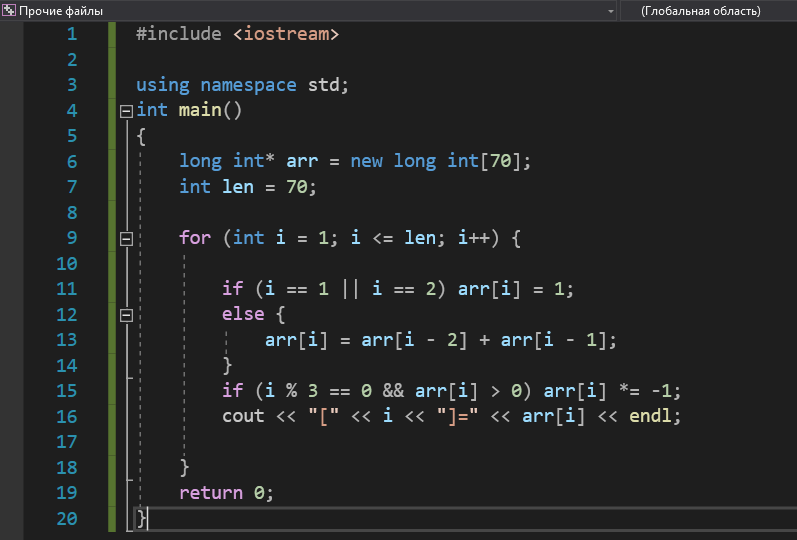


Рисунок 1 – Проверочная программа

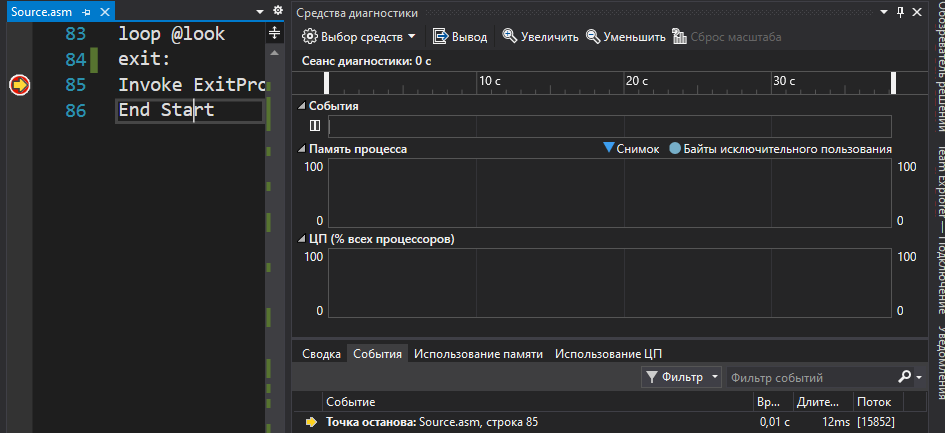


Рисунок 3 – Время работы программы на ассемблере

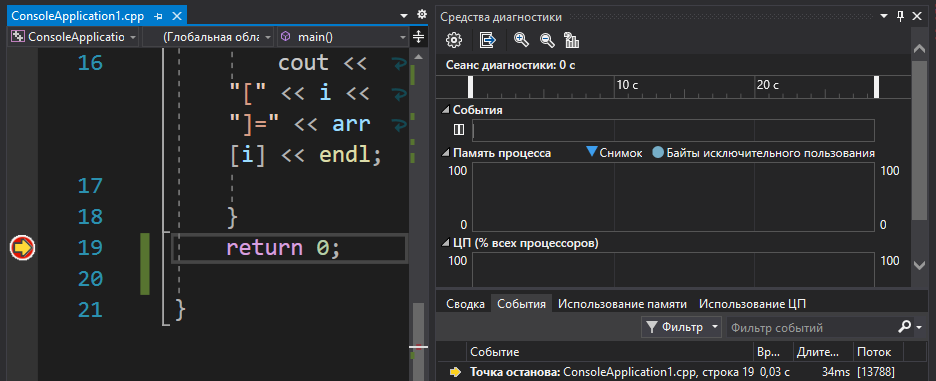


Рисунок 4 – Время работы программы на C++

**Вывод:** Программа работает на ассемблере в 3 раза быстрее чем на C++ 12мс и 34мс соответственно. Так происходит из-за использования математического сопроцессора в ассемблере. Также компилятор в c++ переводит код в машинный, а в ассемблере код изначально пишется на машинном языке. Ассемблер работает быстрее, потому что программист не пишет ничего лишнего, а чтобы обеспечить универсальность применения языковых конструкций, все машинные коды, созданные трансляторами, избыточны.

**Текст программы:**

.686

.model flat,stdcall

.stack 100h

.data

res\_arr dq 70 dup (?)

len equ ($-res\_arr)/8

zero dd 0;

one dq 1.0 ;

three dd 3

cur\_i\_float dd 3

cur\_i\_reg dd 1

tmp dq 0

tt dq 70 dup (?)

.code

ExitProcess PROTO STDCALL :DWORD

Start:

finit ;

xor eax,eax

xor ebx,ebx

xor edx,edx

xor ecx,ecx

mov ecx,0

xor ebx,ebx

mov ebx,cur\_i\_reg ;т

fld one ;

fstp res\_arr[ecx+ebx\*8]

inc cur\_i\_reg ;

mov ebx,cur\_i\_reg

fld one

fstp res\_arr[ecx+ebx\*8]

inc cur\_i\_reg ;i = 3

@@st:

mov ecx,0

mov ebx,cur\_i\_reg

sub ebx,2; i = i-2

fld res\_arr[ecx+ebx\*8]

inc ebx

fadd res\_arr[ecx+ebx\*8]

mov ebx,cur\_i\_reg

@@cmp\_thr:

fld three ;

fld cur\_i\_float ;

fprem ;

fld zero

fcompp ;

fstsw ax

sahf ;

fxch;

je @@th

jne @@add\_to\_arr ;

@@th: ;знак минус у числа

fst tmp ; загрузили из st (0) в esp

fld tmp

fcomp zero;

fstsw ax ;

sahf;

ja @@change ;

jbe @@add\_to\_arr;

@@change:

fchs; замена знака

@@add\_to\_arr:

mov ebx, cur\_i\_reg ;

fstp res\_arr[ecx+ebx\*8] ;

cmp ebx,len ;

je @res;

jne @@inc ;

@@inc:

fld cur\_i\_float ;

fadd one ;

fstp cur\_i\_float

inc cur\_i\_reg ;

jmp @@st ;

@res: ;

xor eax,eax

mov ebx,1

mov ecx,len

@look:

fld res\_arr[eax+ebx\*8];

inc ebx ;

fstp tt[eax+ebx\*8];

loop @look

exit:

Invoke ExitProcess,1

End Start