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 8

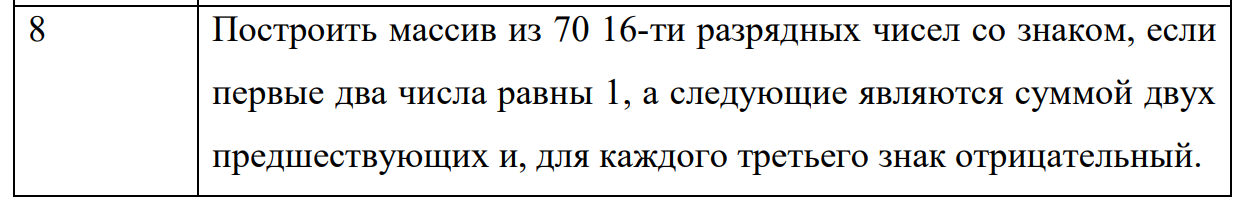
Выполнила: студентка группы ИНБб– 3301\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ Д.А. Логиновская /

Проверил: преподаватель 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++

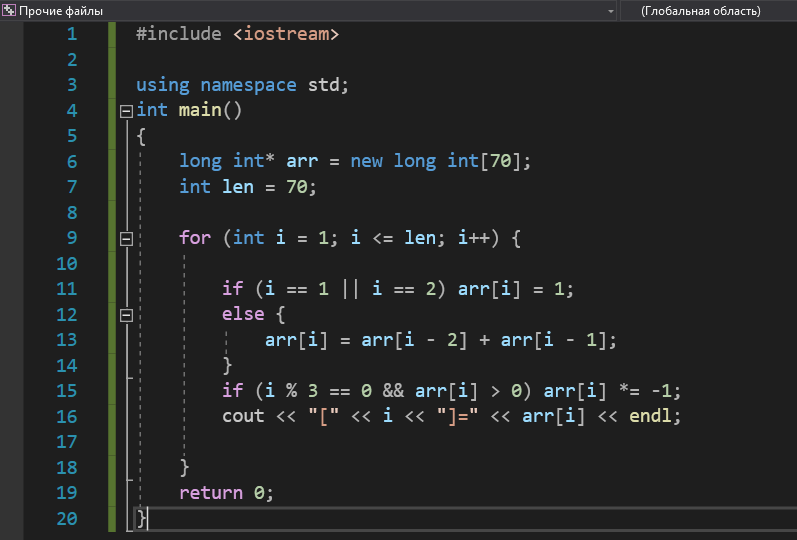


Рисунок 1 – Проверочная программа

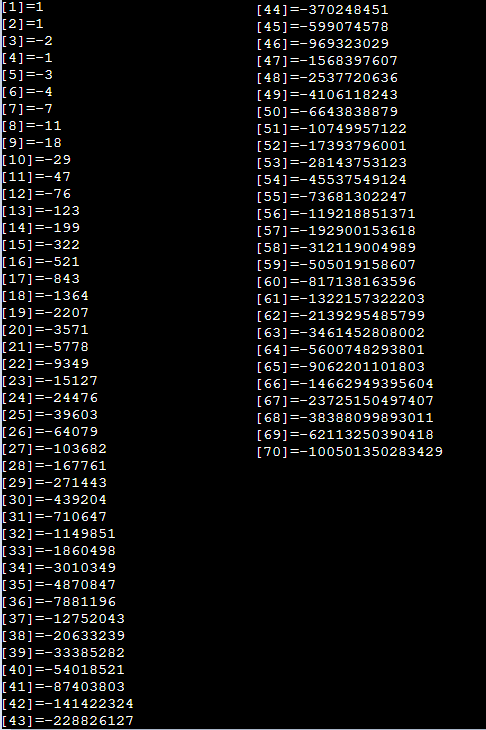


Рисунок 2 – Искомый массив

**Верификация программы:**

Ниже представлены скриншоты с выводом определенных чисел в массиве.

[6] = -4

[17] (11 в 16-ой система счисления) = -843

[39] (27 в 16-ой система счисления) = -33385282

[47] (2F в 16-ой система счисления) = -1568397607

[65] (41 в 16-ой система счисления) = -9062201101803

[70] (46 в 16-ой система счисления) = -100501350283429

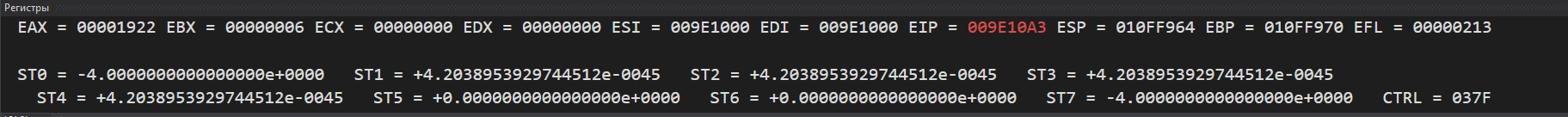


Рисунок 3 – 6-ой элемент массива

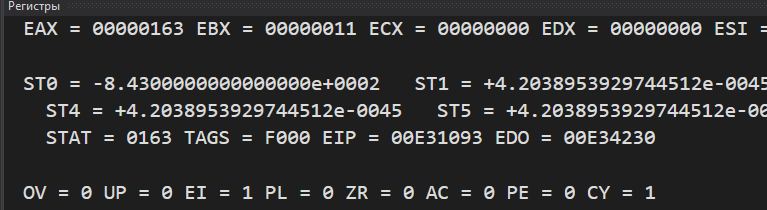


Рисунок 4 – 17-ый элемент массива

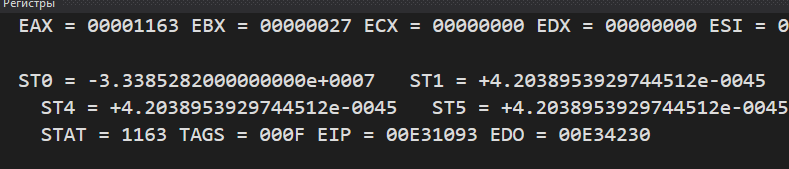


Рисунок 5 – 39-ый элемент массива

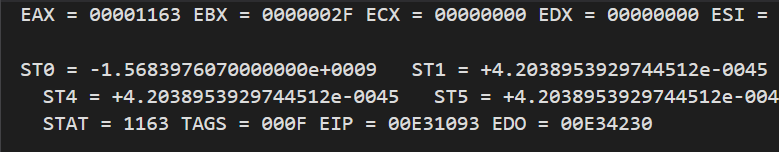


Рисунок 6 – 47-ой элемент массива

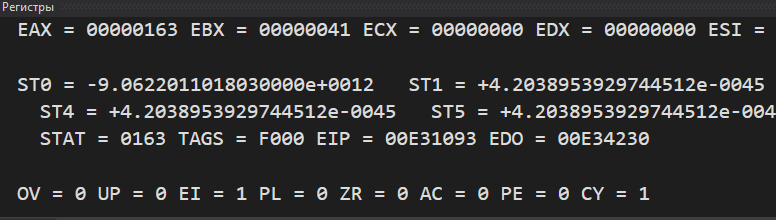


Рисунок 7 – 65-ый элемент массива

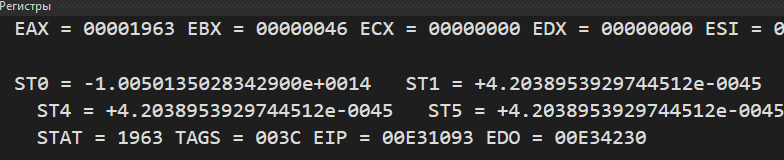


Рисунок 8 – 70-ый элемент массива

**Текст программы:**

.686

.model flat,stdcall

.stack 100h

.data

res\_arr dq 70 dup (?) ;массив из 70 элементов

len equ ($-res\_arr)/8 ;длина массива ( директива equ присваивает значение константе len) dq- директив(4 слова)

zero dd 0; (определяет переменную размером в 4 байта)

one dq 1.0 ;цифра 1 для первых двух значений в 8 байтной форме

three dd 3 ;константа 3 в 4-ех байтной форме

cur\_i\_float dd 3 ;в сопроцессоре начинаем с 3го элемента, т.к. первые два уже заполнили

cur\_i\_reg dd 1 ;в процессоре начинаем с 1 элемента

tmp dq 0 ;для переменного значения ряда (определяет переменную размером в 8 байт)

tt dq 70 dup (?) ; если необходимо выделить память, но не инициализировать ее

.code

ExitProcess PROTO STDCALL :DWORD

Start:

finit ;освобождает все регистры сопроцессора

xor eax,eax

xor ebx,ebx

xor edx,edx

xor ecx,ecx

mov ecx,0

xor ebx,ebx

mov ebx,cur\_i\_reg ;текущий индекс = 1

fld one ; загружает единицу в стек

fstp res\_arr[ecx+ebx\*8] ; Сохраняет значение с извлечением из стека

inc cur\_i\_reg ; Увеличение на 1

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] ;1 слагаемое

inc ebx; прибовляет 1 к индексу

fadd res\_arr[ecx+ebx\*8] ;2 слагаемое

mov ebx,cur\_i\_reg ; счет в ebx

@@cmp\_thr:

fld three ; загрузили 3

fld cur\_i\_float ; загружаем текущий индекс

fprem ; получаем остаток от деления

fld zero

fcompp ; проверяем равен ли остаток нулю и извлекает их из стека

fstsw ax ; загружаем все флаги в регистр АХ

sahf ; перенос из регистра ah в регистр флагов

fxch; меняет местами st(0) с st(1)

je @@th ; если третье в редяу- переход на метку для изменения знака (результат равный)

jne @@add\_to\_arr ; иначе - добавляем сумму в массив (переход, если не равно)

@@th: ;знак минус у числа

fst tmp ; загрузили из st (0) в esp

fld tmp

fcomp zero; сравнение вещественного числа в ST(0) с операндом с последующим выталкиванием ST(0) из стека

fstsw ax ; загружаем все флаги в регистр АХ

sahf; перенос из регистра ah в регистр флагов

ja @@change ;если tmp > 0, то переходим по адресу замены знака

jbe @@add\_to\_arr; перейти, если меньше или равно

@@change:

fchs; замена знака

@@add\_to\_arr:

mov ebx, cur\_i\_reg ; текущий индекс

fstp res\_arr[ecx+ebx\*8] ; загрузили из st (0) в esp с извлечением из стека

cmp ebx,len ; i<70 = условие выхода (сравнение двух чисел)

je @res; Переход если равны

jne @@inc ; Переход, если не равно

@@inc:

fld cur\_i\_float ; загрузили текущий индекс

fadd one ; увеличили его на 1

fstp cur\_i\_float ;вывели значение в переменную

inc cur\_i\_reg ; повысили на 1 для регистра

jmp @@st ;цикл

@res: ;получить значение из массива

xor eax,eax

mov ebx,1

mov ecx,len

@look:

fld res\_arr[eax+ebx\*8]; загрузили число в стек

inc ebx ; увеличили на 1

fstp tt[eax+ebx\*8]; Сохраняет значение с извлечением из стека

loop @look

exit:

Invoke ExitProcess,1

End Start

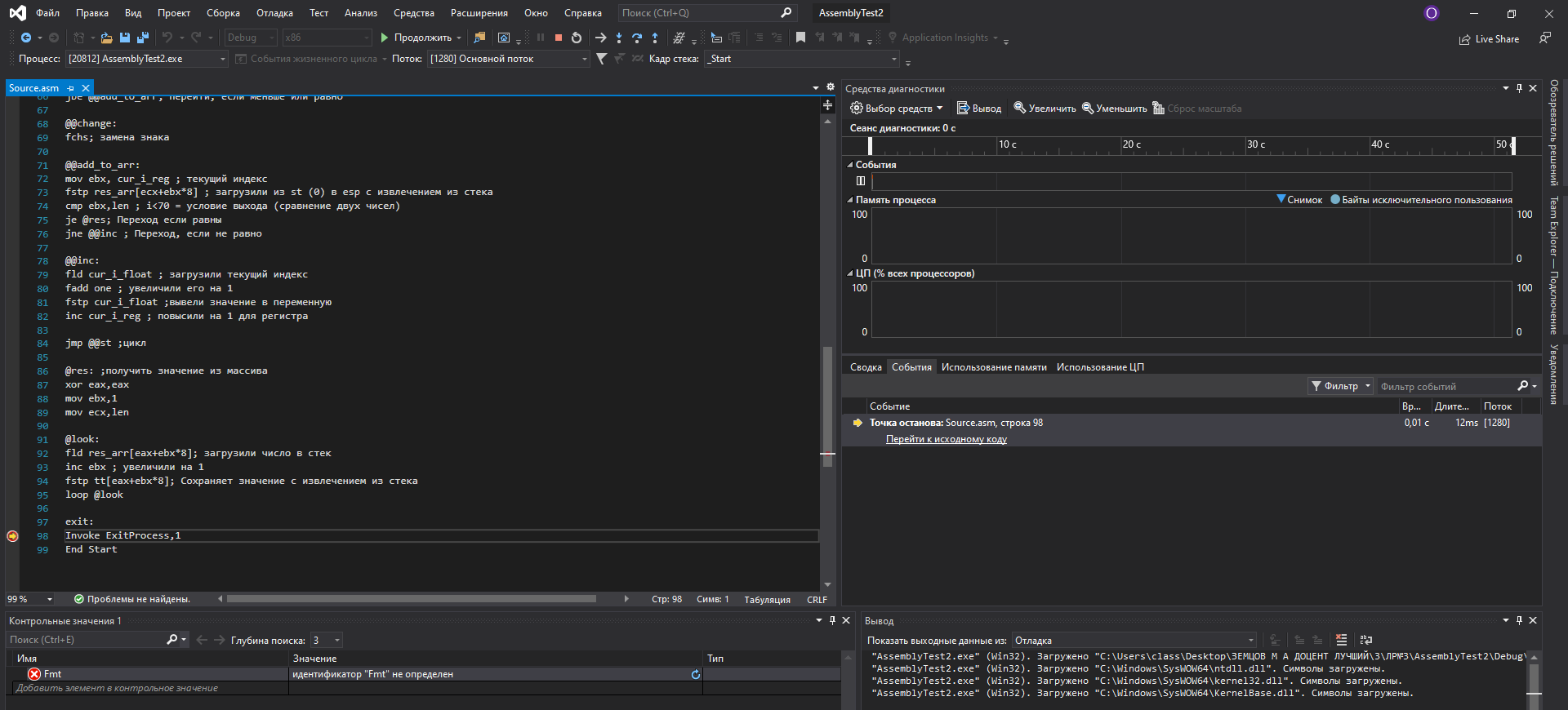


Рисунок 9 – Профилировка производительности программы на ассемблере. Время = 12мс.

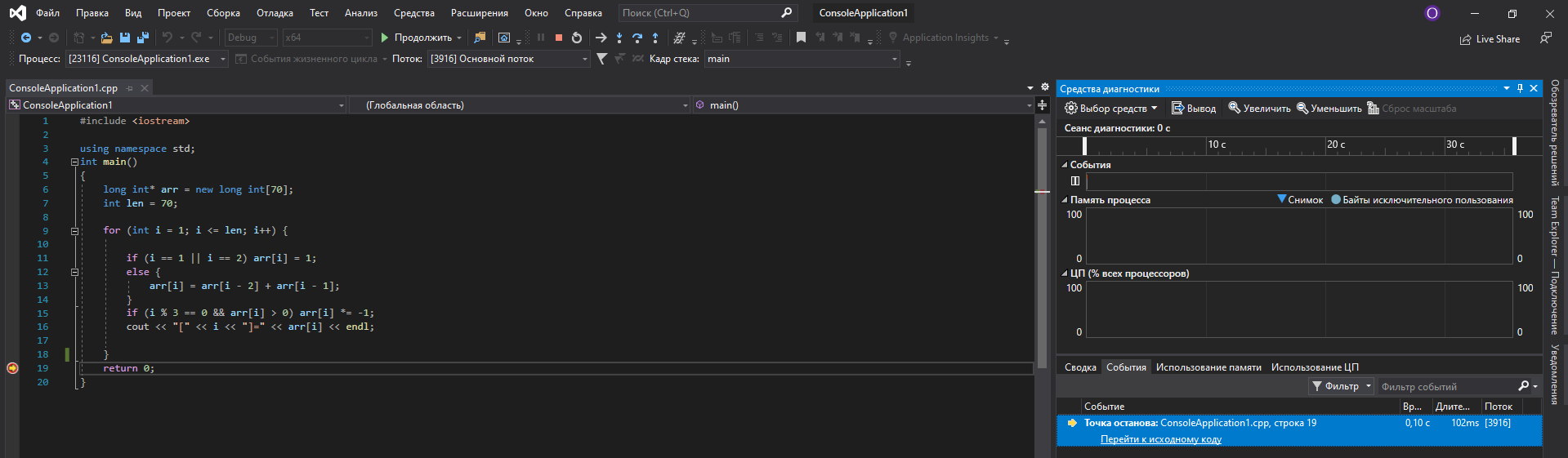


Рисунок 10 – Профилировка производительности программы на ассемблере. Время = 102мс.

**Вывод:** в ходе лабораторной работы были изучены принципы выполнения арифметических команд с помощью математического сопроцессора FPU микропроцессоров с архитектурой x86, а также были изучены следующие команды: помещение числа в стек, проверка остатка от деления на равенство с нулем, обмен значениями между стеками, изменение знака. Значения в массиве и в результатах проверочной программы одинаковые, что говорит о правильности выполнения программы.

На ассемблере аналогичная программа выполняется быстрее, чем на C++, потому что в программе на ассемблере используется математический сопроцессор, в дизассемблированном коде C++ нет команд вызова для работы с математическим сопроцессором. Операции с использованием математического сопроцессора дают выигрыш в производительности при работе с числами с плавающей точкой.