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 лабораторной работе №3**

по дисциплине «Цифровые устройства и микропроцессоры»

на тему

**«Использование математического сопроцессора»**

Вариант 5

Выполнил: студент ИНБб-3301-02-00 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Н. С. Карпичев /

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

Киров 2024

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

**Задание:**



**Код программы:**

.686

.model flat,stdcall

option casemap :none

.stack 100h

.data

M1 dd 11

sum dd 0

array dd 121,11,140,22,26,33,1474,10,1474,11

N equ ($ - array) / 4

.code

ExitProcess PROTO STDCALL :DWORD

swap PROC

finit

fild array[esi]

fild array[esi + 4]

fcomip st(0), st(1) ;

jnb @@end\_compare ;

fild array [esi]

fild array [esi + 4]

; Меняем местами элементы

fxch

fistp array [esi + 4]

fistp array [esi]

@@end\_compare:

ret

swap ENDP

sum\_elements PROC

finit

fild M1[0] ; Загружаем константу 11 в стек сопроцессора

fild array[esi] ; Загружаем текущее число в стек сопроцессора

fprem ; получаем остаток от деления

ftst

fstsw ax

sahf

sete al ;

movzx eax, al ;

cmp eax,0

JE @@next1

fild array[esi]

fiadd sum

fistp sum

ret

sum\_elements ENDP

Start:

mov cx, N\*(N-1) ; Количество элементов в массиве

dec cx ; Нужно для вычитания единицы, т.к. внешний цикл должен проходить на один элемент меньше

mov ax, cx

@@outer\_loop:

mov esi, 0

@@inner\_loop:

; Сравнение элементов и обмен, если нужно

invoke swap

cmp esi,4\*(N-2)

JE @@outer\_loop

add esi, 4 ; Смещение на один элемент (4 байта) вперед

dec ax

cmp ax,0

JE @@next

jnz @@inner\_loop

loop @@outer\_loop

;

@@next:

mov esi, 0

@@sum\_loop:

invoke sum\_elements

@@next1:

add esi, 4 ; Смещение на один элемент (4 байта) вперед

cmp esi,4\*N

JE @@outer\_loo

loop @@sum\_loop

mov eax,sum

@@outer\_loo:

mov edi,[array+4]

mov eax,sum

exit:

Invoke ExitProcess,N

End Start

**Верификация программы:**

**Код программы на C++**

#include <iostream>

using namespace std;

int main()

{

int sum = 0;

int dd[] = { 121,11,140,22,26,33,1474,10,1474,11 };

int i, j;

int length = sizeof(dd) / sizeof(dd[0]);

for (i = 0; i < length - 1; i++) {

for (j = 0; j < length - i - 1; j++)

{

if (dd[j] > dd[j + 1])

{

// swap dd[j] and dd[j+1]

int temp = dd[j];

dd[j] = dd[j + 1];

dd[j + 1] = temp;

}

}

}

// print the sorted array

for (i = 0; i < length; i++) {

cout << dd[i] << " ";

}

cout << endl;

for (i = 0; i < length; i++)

{

if (dd[i] % 11 == 0)

{

sum += dd[i];

}

}

cout <<"СУММА="<< sum;

return 0;

}

**2.Результат работы программы на C++**

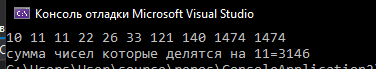


Рисунок 1-Результат выполнения работы на С++

Схема алгоритма представлена на рисунке 2

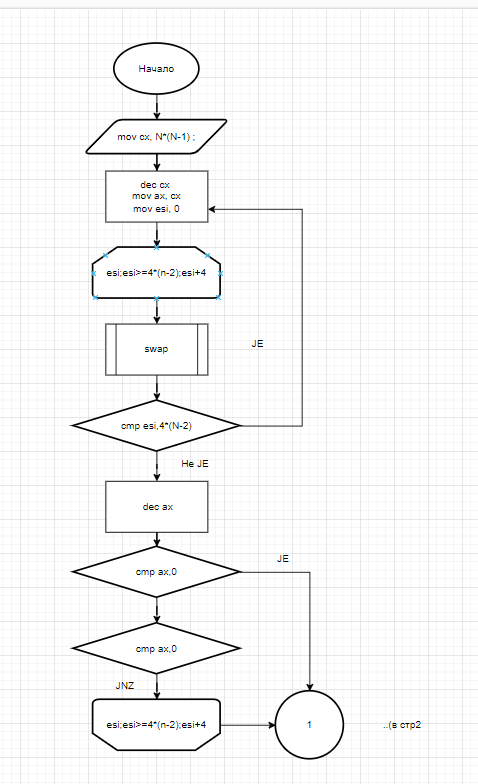
****

Рисунок 2 –Схема алгоритма на Assembler

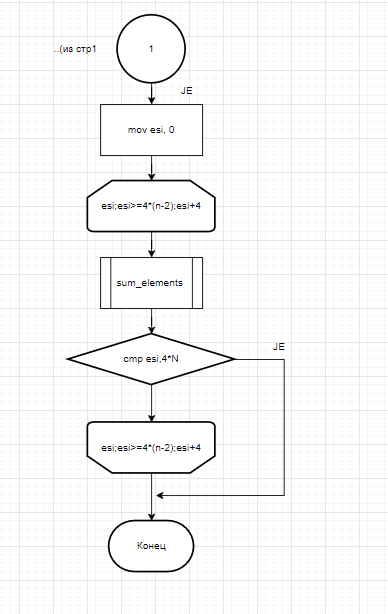
****

Рисунок 3 –Схема алгоритма на Assembler

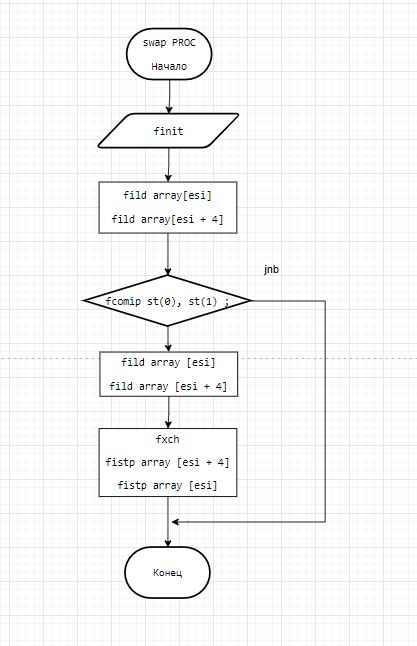
****

Рисунок 4 –Схема алгоритма на Assembler

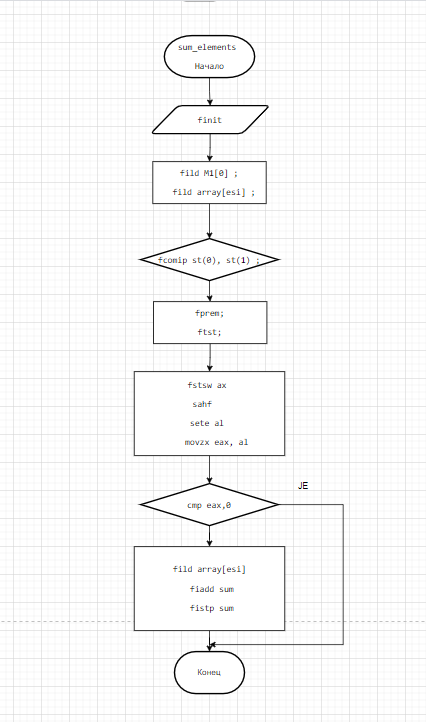
****

Рисунок5–Схема алгоритма на Assembler

**3. Результат работы программы на Assembler**

**Суммы:**



Рисунок 6-результат суммы полученный на **Assembler**

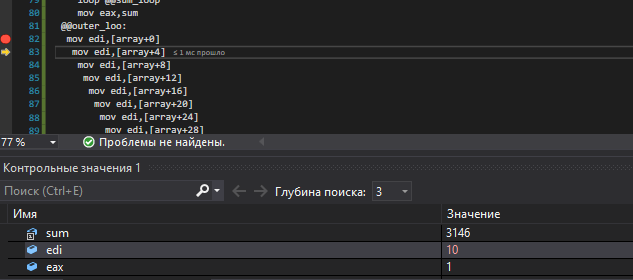


Рисунок 7-результат 1 элемента массива

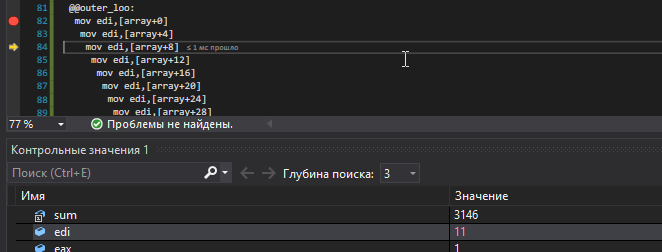


Рисунок 8-результат 2 элемента массива

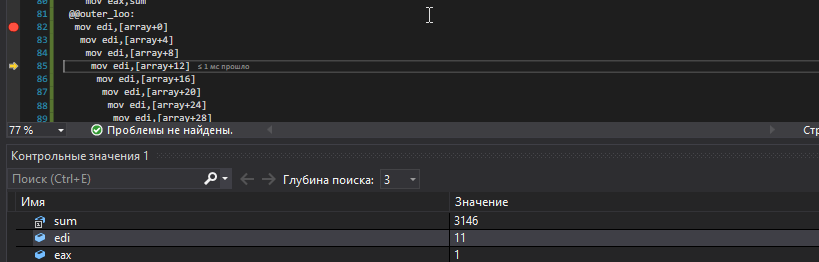


Рисунок 9-результат 3 элемента массива

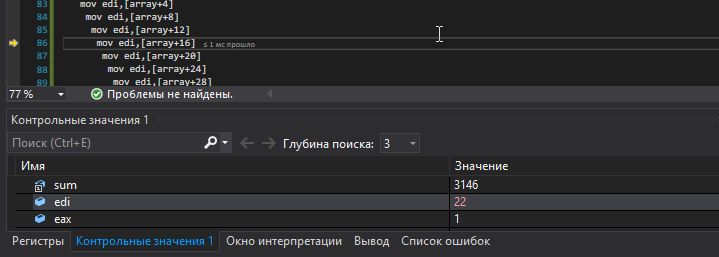


Рисунок 10-результат 4 элемента массива

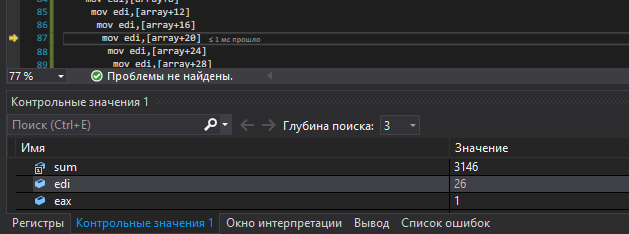


Рисунок 11-результат 5 элемента массива

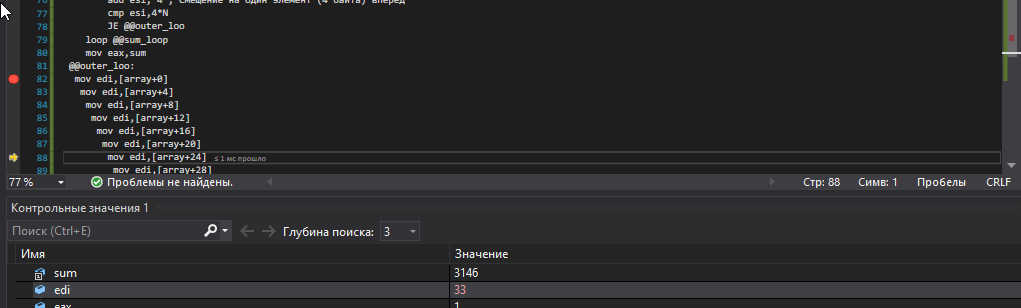


Рисунок 12-результат 6 элемента массива

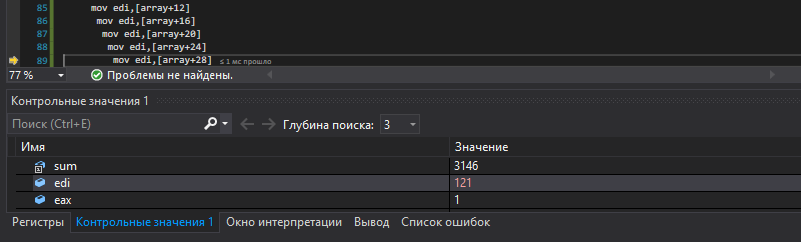


Рисунок 13-результат 7 элемента массива

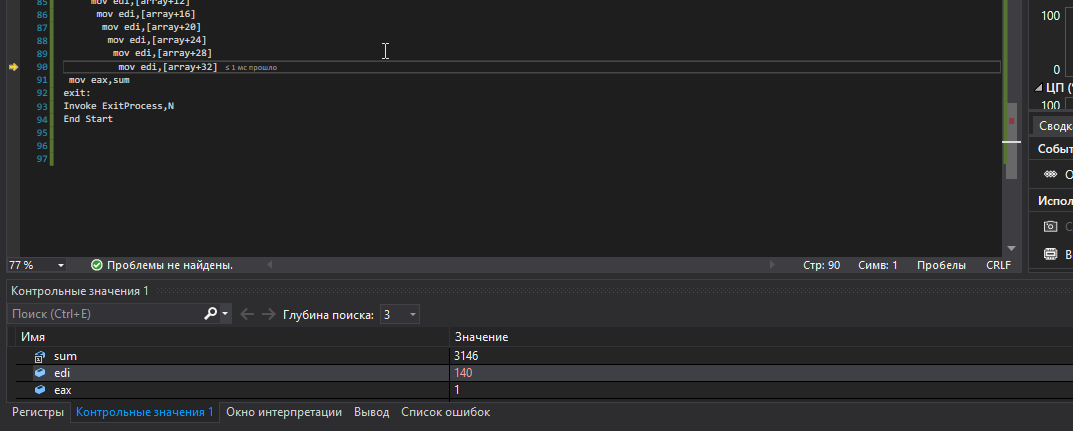


Рисунок 14-результат 8 элемента массива

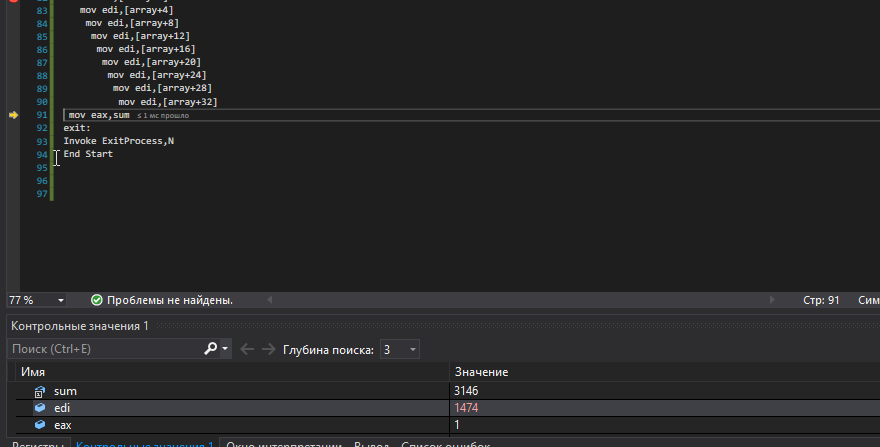


Рисунок 15-результат 9 элемента массива

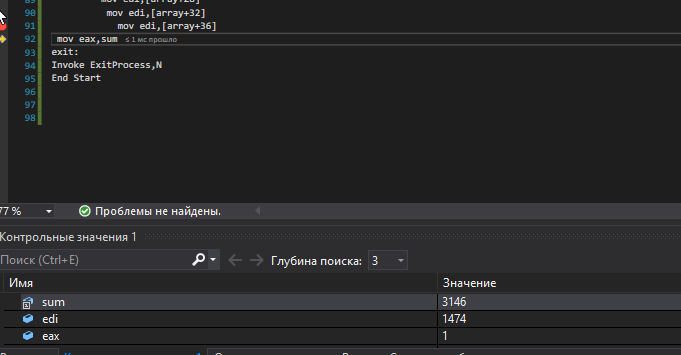


Рисунок 16-результат 10 элемента массива

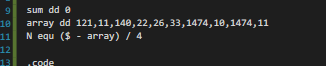


Рисунок 17- Исходный массив

**Вывод:** в ходе работы были изучены принципы выполнения арифметических команд с помощью математического сопроцессора FPU микропроцессоров с архитектурой x86.