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»

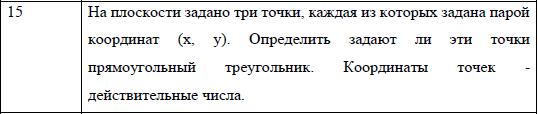
Вариант №15

|  |  |  |
| --- | --- | --- |
| Выполнил: студент гр. ИКТб-3301-04-00 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Я. Н. Чугунов |
| 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. Написать исполняемый код на C++, затем упростить и реализовать в assembler, согласно варианту задания.*

1. **Задание**



1. **Ход работы**

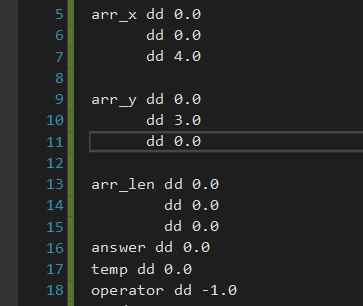
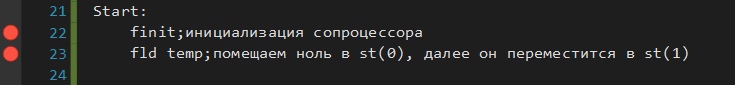


Рисунок 1 – Инициализация данных



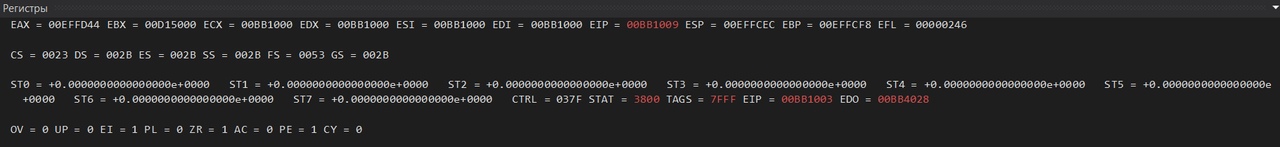


Рисунок 2 – Инициализация сопроцессора и помещение 0 на вершину стека

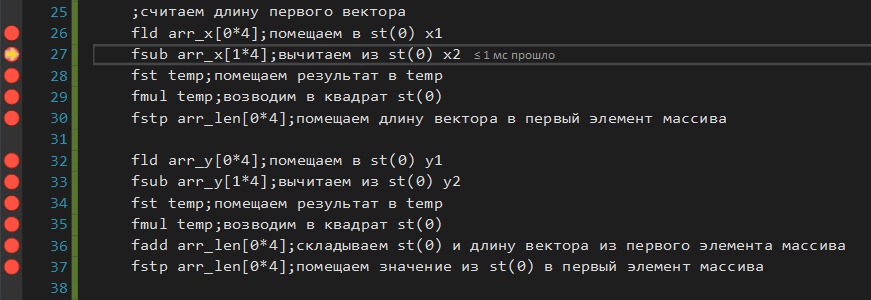


Рисунок 3 – Подсчет длины первого вектора

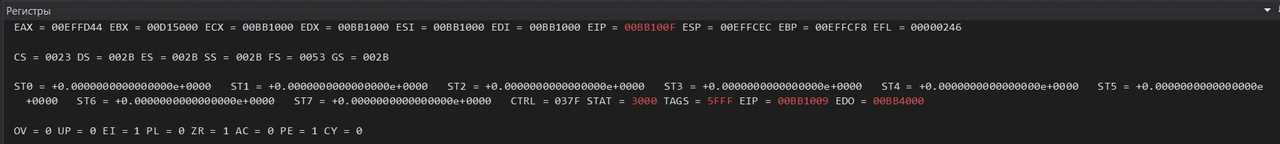


Рисунок 4 – Поместили х1 в st(0)

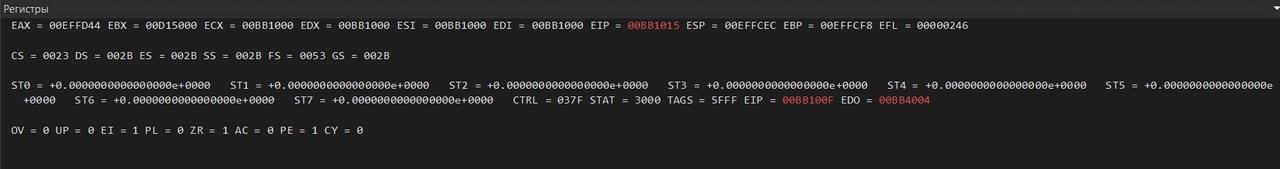


Рисунок 5 - Вычли x2 из st(0)

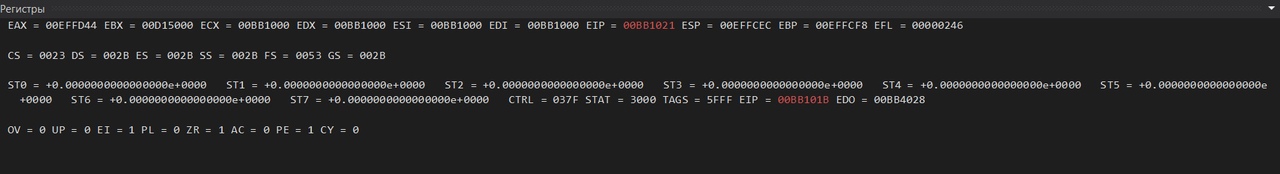


Рисунок 6 - Возвели в квадрат st(0) и затем поместили st(0) в первый элемент arr\_len

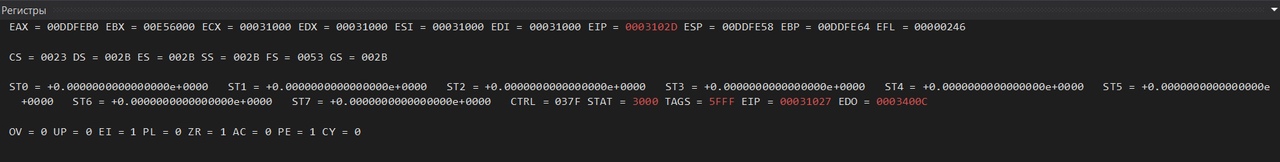


Рисунок 7 - Поместили y1 на вершину стека

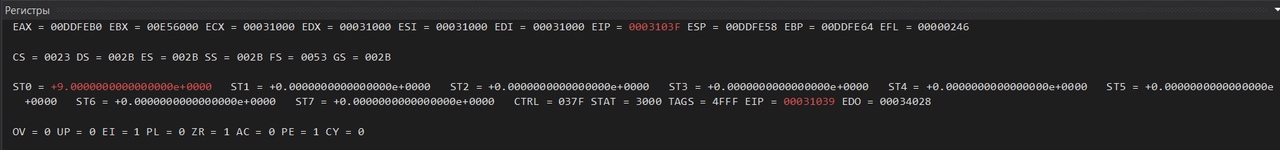


Рисунок 8 - Вычли y2 из st(0)

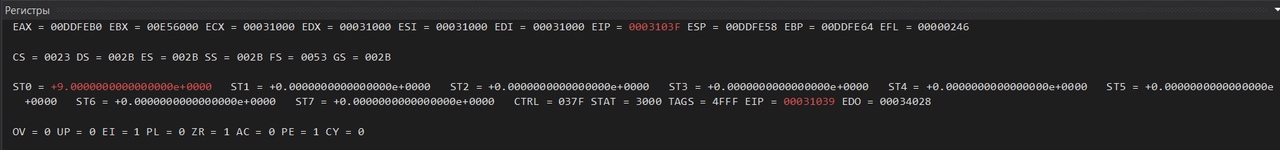


Рисунок 9 – Копируем значение в temp  
и возводим в квадрат st(0)

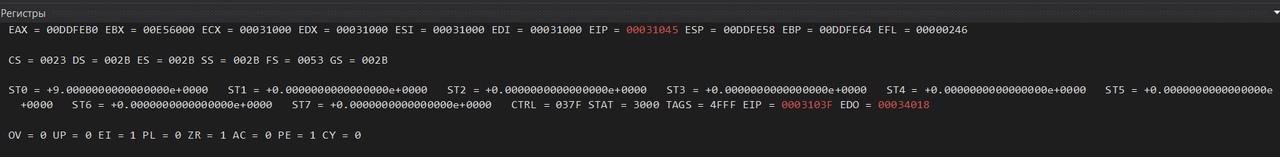


Рисунок 10 – Складываем st(0) и значение первого элемента массива arr\_len

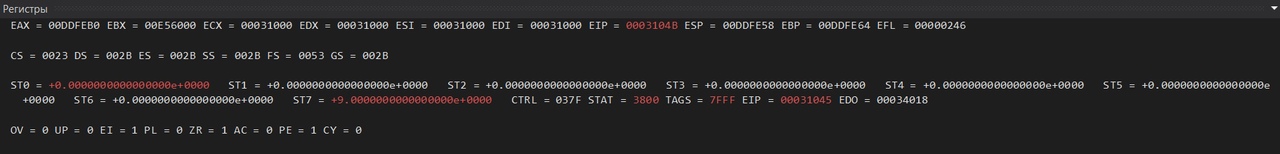


Рисунок 11 – Поместили значение из st(0) в первый элемент массива arr\_len. получили длину первого вектора

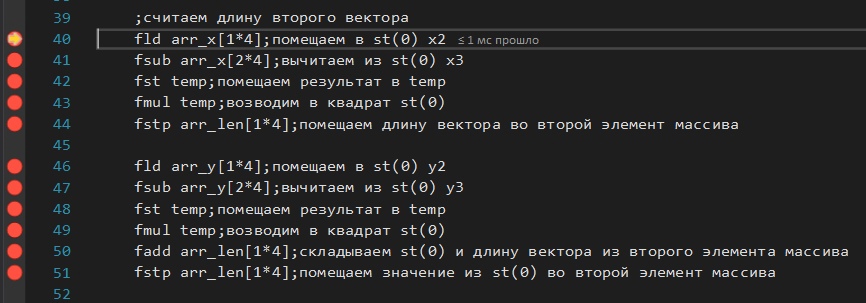


Рисунок 12 – Считаем длину второго вектора

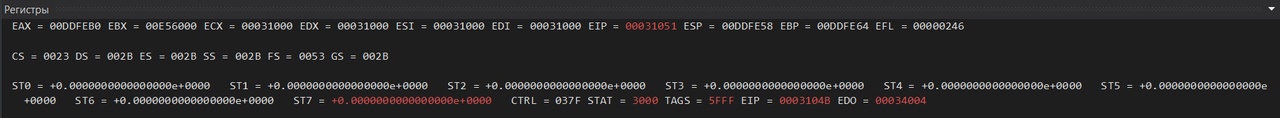


Рисунок 13 – Поместили в st(0) x2

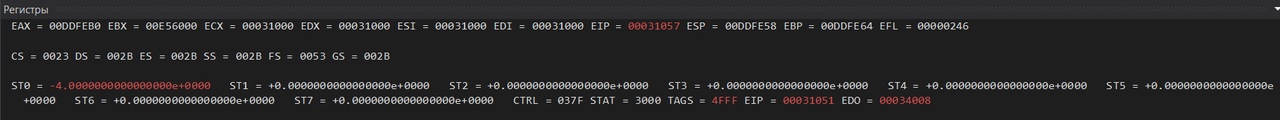


Рисунок 14 – Вычли x3 из st(0)

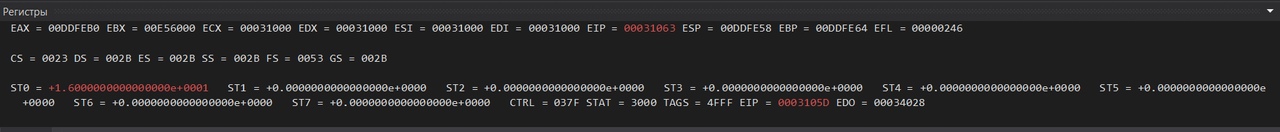


Рисунок 15 – Возвели в квадрат st(0)

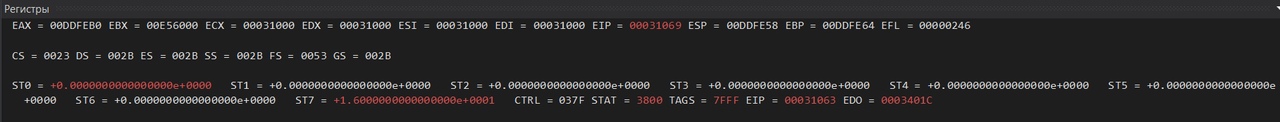


Рисунок 16 – Поместили значение из st(0) во второй элемент массива arr\_len

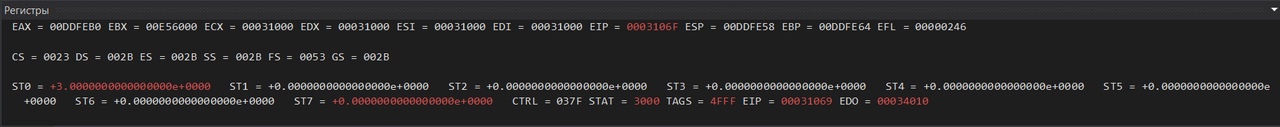


Рисунок 17 – Поместили в st(0) y2

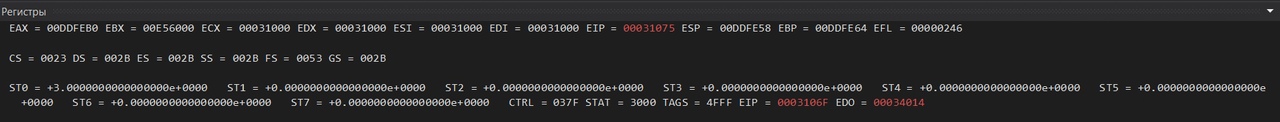


Рисунок 18 – Вычли из st(0) y3



Рисунок 19 – Возвели в квадрат st(0)

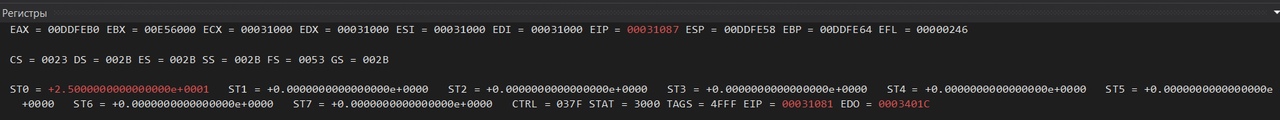


Рисунок 20 – Сложили st(0) и значение из второго элемента массива arr\_len



Рисунок 21 – Поместили значение st(0) во второй элемент массива arr\_len; посчитана длина второго вектора

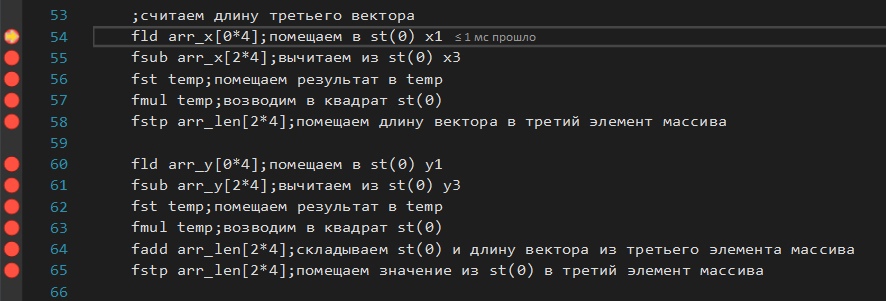


Рисунок 22 – Считаем длину третьего вектора



Рисунок 23 – Поместили x1 в st(0)

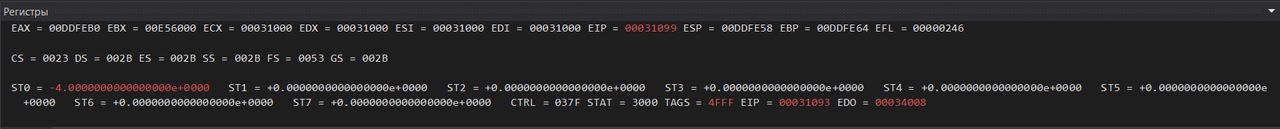


Рисунок 24 – Вычли x3 из st(0)



Рисунок 25 – Возводим в квадрат st(0)



Рисунок 26 – Помещаем st(0) в третий элемент массива arr\_len

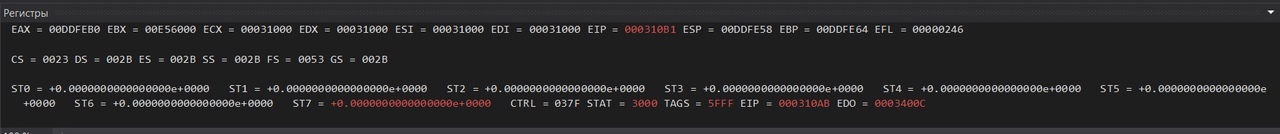


Рисунок 27 – Поместили в st(0) y1

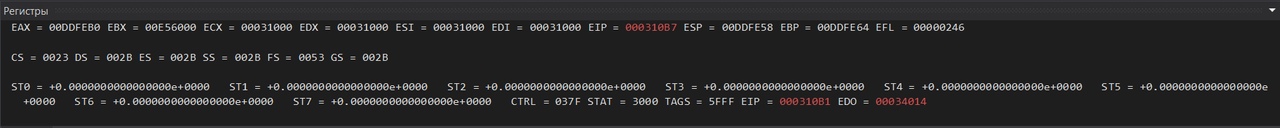


Рисунок 28 – Вычли y2 из st(0)

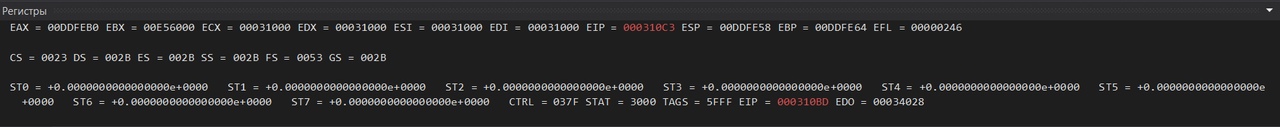


Рисунок 29 – Возвели в квадрат st(0)

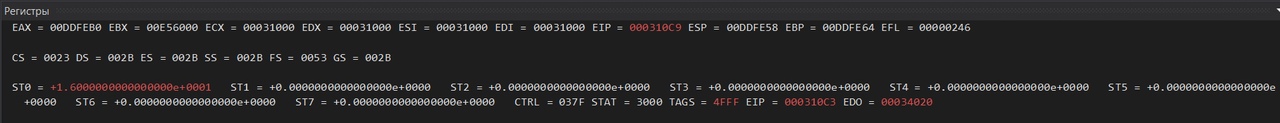


Рисунок 30 – Сложили st(0) и третий элемент массива arr\_len

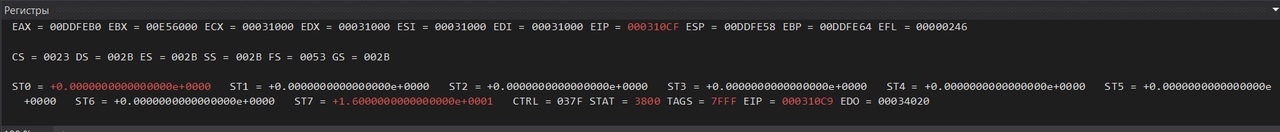


Рисунок 31 – Поместили st(0) в третий элемент массива arr\_len; длина третьего вектора посчитана

Далее считаем результат

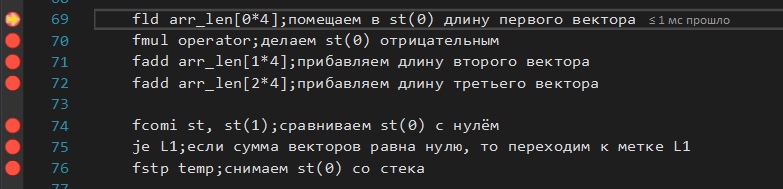


Рисунок 32 – Делаем отрицательным длину первого вектора и считаем сумму

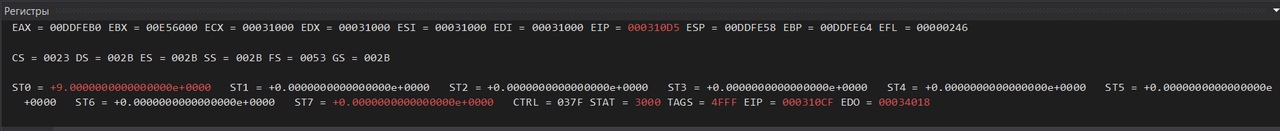


Рисунок 33 – Поместили в st(0) длину первого вектора

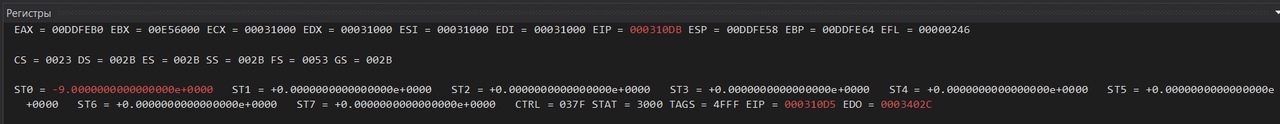


Рисунок 34 – Сделали его отрицательным

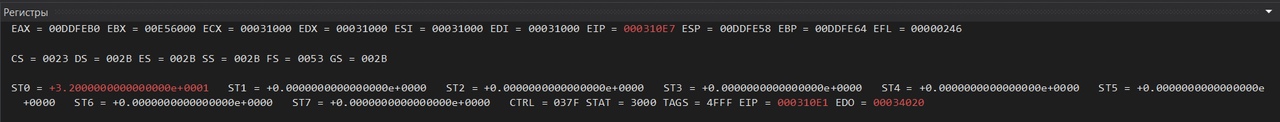


Рисунок 35 – Сложили со 2 и 3 длинами векторов

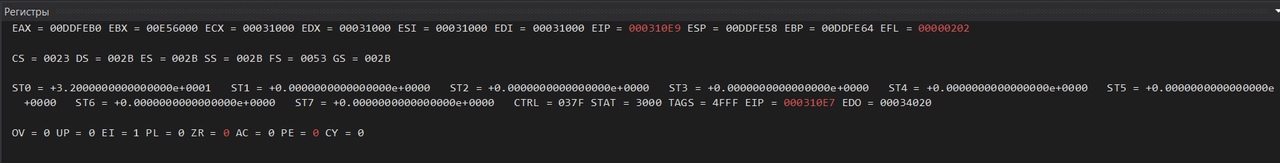


Рисунок 36 – Сравниваем с 0. 32 не равно нулю, значит идём дальше

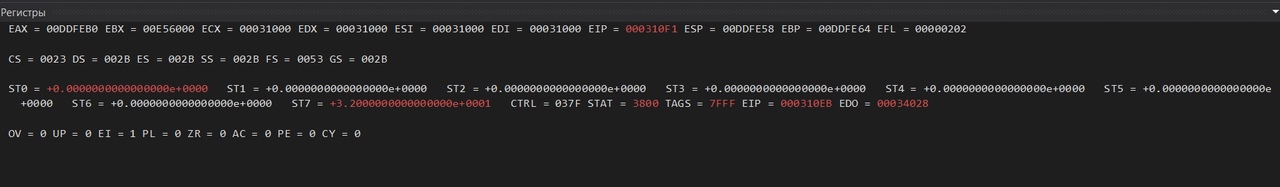


Рисунок 37 – Сняли st(0) с вершины стека

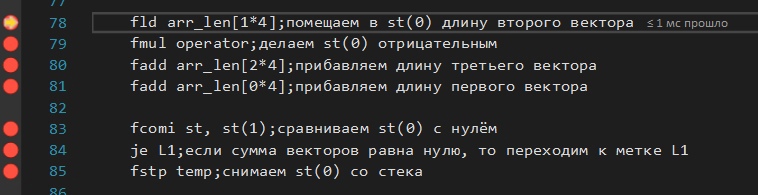


Рисунок 38 – Делаем отрицательной длину второго вектора и считаем сумму

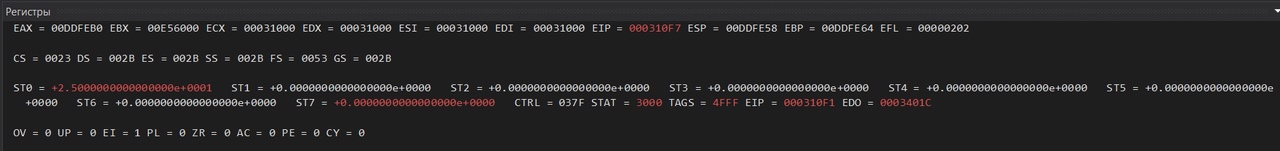


Рисунок 39 – Поместили в st(0) длину второго вектора

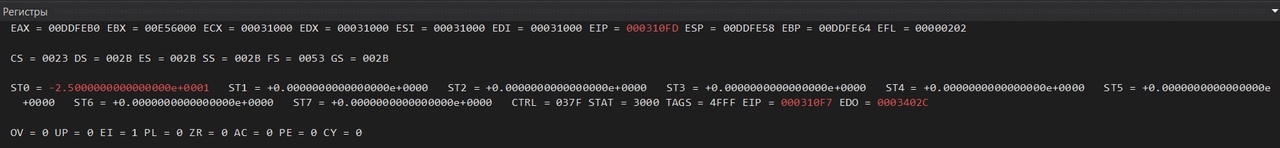


Рисунок 40 – Сделали его отрицательным

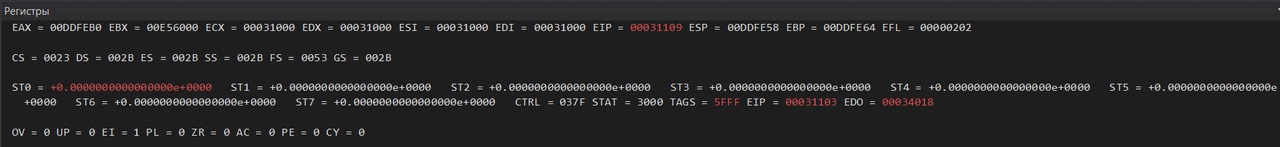


Рисунок 41 – Прибавили длину 1 и 3 вектора

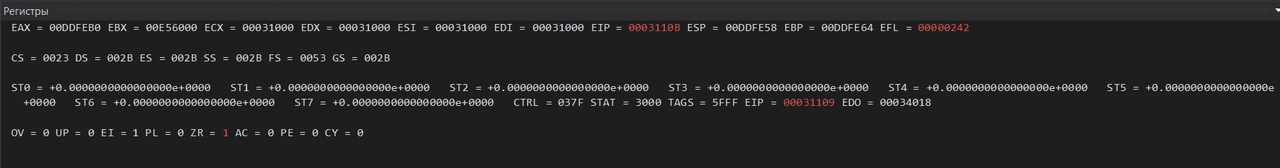
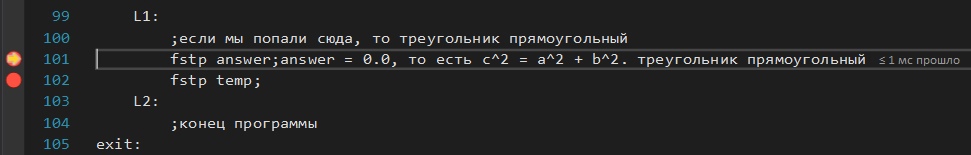


Рисунок 42 – сравниваем st(0) с 0. Сумма векторов = 0. значит треугольник прямоугольный.

Переходим к метке L1



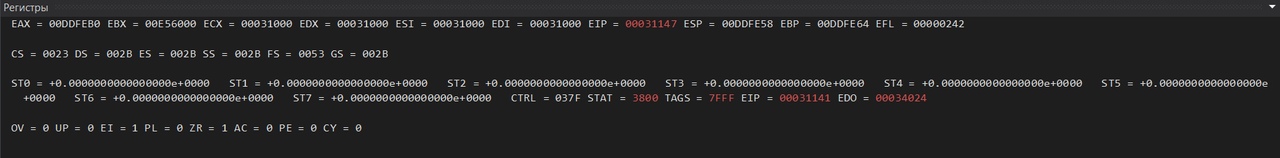


Рисунок 43 – Результат

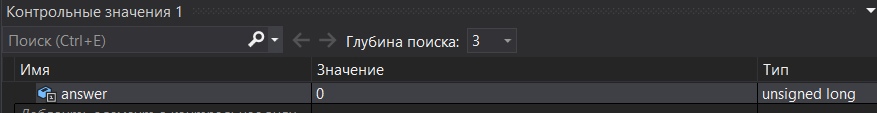


Рисунок 44 – answer = 0; значит треугольник прямоугольный. Конец работы программы

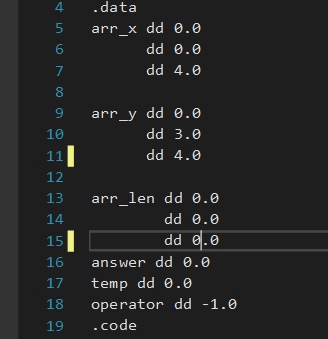


Рисунок 45 – Изменили начальные данные

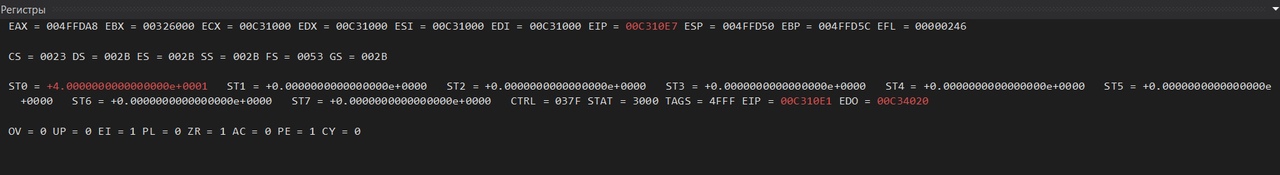


Рисунок 46 – Если первый элемент массива arr\_len сделать отрицательным, то сумма векторов будет равна 40

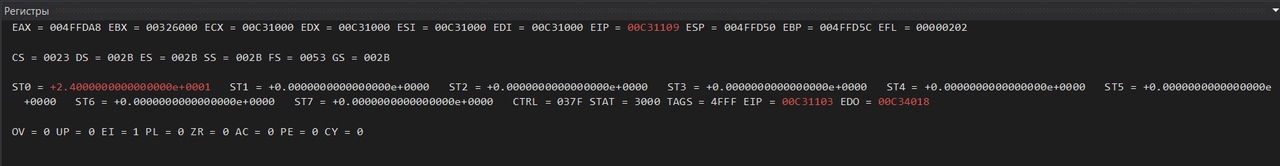
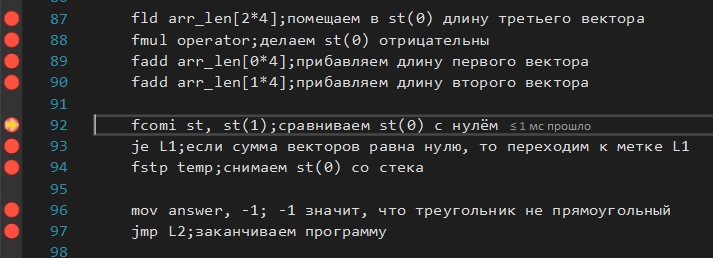


Рисунок 47 – Если второй элемент массива arr\_len сделать отрицательным, то сумма векторов будет равно 24



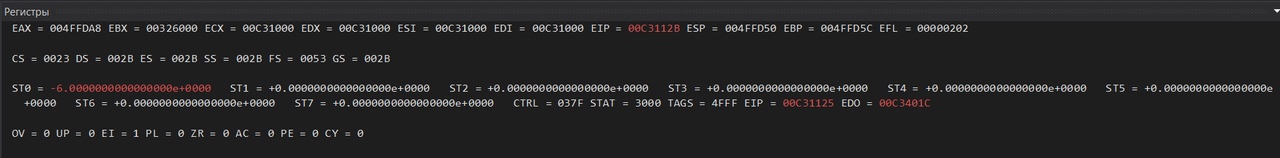
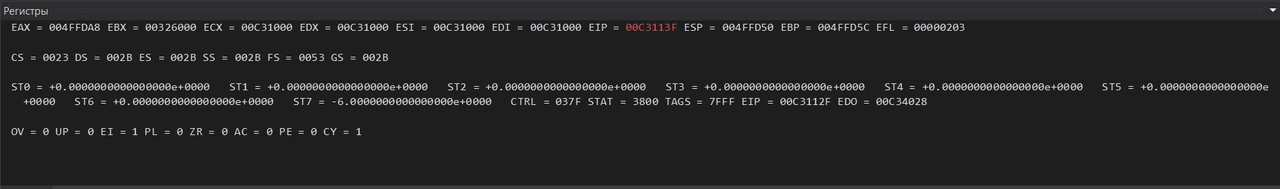


Рисунок 48 – Если третий элемент массива arr\_len сделать отрицательным, то сумма векторов будет равна -6



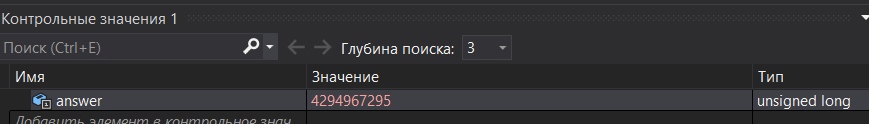


Рисунок 49 – Сумма векторов не равна нулю ни при одном из этих трёх случаев, соответственно треугольник не является прямоугольным. Об этом свидетельствует answer = -1

***Вывод:*** программа корректно выполняет действия, подсчеты в сопроцессоре правильные, условия задачи выполняются. Так же изучили принципы выполнения арифметических команд с помощью математического сопроцессора FPU микропроцессоров с архитектурой x86.

1. **Код программы:**

.686

.model flat,stdcall

.stack 100h

.data

arr\_x dd 0.0

dd 0.0

dd 4.0

arr\_y dd 0.0

dd 3.0

dd 0.0

arr\_len dd 0.0

dd 0.0

dd 0.0

answer dd 0.0

temp dd 0.0

operator dd -1.0

.code

ExitProcess PROTO STDCALL :DWORD

Start:

finit;инициализация сопроцессора

fld temp;помещаем ноль в st(0), далее он переместится в st(1)

;считаем длину первого вектора

fld arr\_x[0\*4];помещаем в st(0) x1

fsub arr\_x[1\*4];вычитаем из st(0) x2

fst temp;помещаем результат в temp

fmul temp;возводим в квадрат st(0)

fstp arr\_len[0\*4];помещаем длину вектора в первый элемент массива

fld arr\_y[0\*4];помещаем в st(0) y1

fsub arr\_y[1\*4];вычитаем из st(0) y2

fst temp;помещаем результат в temp

fmul temp;возводим в квадрат st(0)

fadd arr\_len[0\*4];складываем st(0) и длину вектора из первого элемента массива

fstp arr\_len[0\*4];помещаем значение из st(0) в первый элемент массива

;считаем длину второго вектора

fld arr\_x[1\*4];помещаем в st(0) x2

fsub arr\_x[2\*4];вычитаем из st(0) x3

fst temp;помещаем результат в temp

fmul temp;возводим в квадрат st(0)

fstp arr\_len[1\*4];помещаем длину вектора во второй элемент массива

fld arr\_y[1\*4];помещаем в st(0) y2

fsub arr\_y[2\*4];вычитаем из st(0) y3

fst temp;помещаем результат в temp

fmul temp;возводим в квадрат st(0)

fadd arr\_len[1\*4];складываем st(0) и длину вектора из второго элемента массива

fstp arr\_len[1\*4];помещаем значение из st(0) во второй элемент массива

;считаем длину третьего вектора

fld arr\_x[0\*4];помещаем в st(0) x1

fsub arr\_x[2\*4];вычитаем из st(0) x3

fst temp;помещаем результат в temp

fmul temp;возводим в квадрат st(0)

fstp arr\_len[2\*4];помещаем длину вектора в третий элемент массива

fld arr\_y[0\*4];помещаем в st(0) y1

fsub arr\_y[2\*4];вычитаем из st(0) y3

fst temp;помещаем результат в temp

fmul temp;возводим в квадрат st(0)

fadd arr\_len[2\*4];складываем st(0) и длину вектора из третьего элемента массива

fstp arr\_len[2\*4];помещаем значение из st(0) в третий элемент массива

;считаем результат

fld arr\_len[0\*4];помещаем в st(0) длину первого вектора

fmul operator;делаем st(0) отрицательным

fadd arr\_len[1\*4];прибавляем длину второго вектора

fadd arr\_len[2\*4];прибавляем длину третьего вектора

fcomi st, st(1);сравниваем st(0) с нулём

je L1;если сумма векторов равна нулю, то переходим к метке L1

fstp temp;снимаем st(0) со стека

fld arr\_len[1\*4];помещаем в st(0) длину второго вектора

fmul operator;делаем st(0) отрицательным

fadd arr\_len[2\*4];прибавляем длину третьего вектора

fadd arr\_len[0\*4];прибавляем длину первого вектора

fcomi st, st(1);сравниваем st(0) с нулём

je L1;если сумма векторов равна нулю, то переходим к метке L1

fstp temp;снимаем st(0) со стека

fld arr\_len[2\*4];помещаем в st(0) длину третьего вектора

fmul operator;делаем st(0) отрицательны

fadd arr\_len[0\*4];прибавляем длину первого вектора

fadd arr\_len[1\*4];прибавляем длину второго вектора

fcomi st, st(1);сравниваем st(0) с нулём

je L1;если сумма векторов равна нулю, то переходим к метке L1

fstp temp;снимаем st(0) со стека

mov answer, -1; -1 значит, что треугольник не прямоугольный

jmp L2;заканчиваем программу

L1:

;если мы попали сюда, то треугольник прямоугольный

fstp answer;answer = 0.0, то есть c^2 = a^2 + b^2. треугольник прямоугольный

fstp temp;

L2:

;конец программы

exit:

Invoke ExitProcess,1

End Start

1. **Код программы, реализованный на C++:**

#include <iostream>

#include <cmath>

typedef struct

{

float x, y;

} Point;

template <class T> T sqr(T a) { return a \* a; }

float dist(Point a, Point b) { return sqrt(sqr(a.x - b.x) + sqr(a.y - b.y)); }

bool func(Point a, Point b, Point c)

{

if (dist(a, b) + dist(a, c) <= dist(c, b))

return 0;

if (dist(a, b) + dist(c, b) <= dist(a, c))

return 0;

if (dist(c, b) + dist(c, a) <= dist(a, b))

return 0;

return 1;

}

int main()

{

Point a, b, c;

std::cout << "Enter A(x,y) B(x,y) C(x,y)" << std::endl;

std::cout << "A(x): ";

std::cin >> a.x;

std::cout << "A(y): ";

std::cin >> a.y;

std::cout << "A: (" << a.x << ";" << a.y << ")" << std::endl;

std::cout << "B(x): ";

std::cin >> b.x;

std::cout << "B(y): ";

std::cin >> b.y;

std::cout << "B: (" << b.x << ";" << b.y << ")" << std::endl;

std::cout << "C(x): ";

std::cin >> c.x;

std::cout << "C(y): ";

std::cin >> c.y;

std::cout << "C: (" << c.x << ";" << c.y << ")" << std::endl;

std::cout << (func(a, b, c) ? "Yes, It's right triangle ABC" : "No, It is not right triangle ABC") << std::endl;

return 0;

}