МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ

БЮДЖЕТНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНВЕРСИТЕТ

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 группы ИНБс-3301 К.А. Юрлов

Проверил: доцент кафедры РЭС М.А. Земцов

Киров 2024

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

**Ход работы:**

Задание по варианту представлено на рисунке 1.

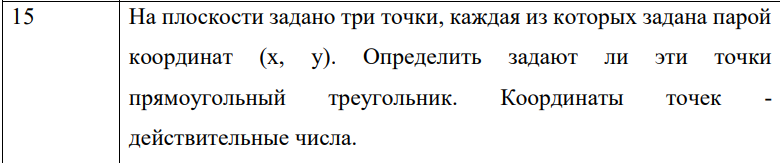


Рисунок 1 – задание по варианту

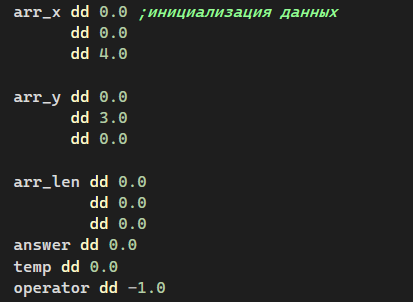


Рисунок 2 – инициализация данных

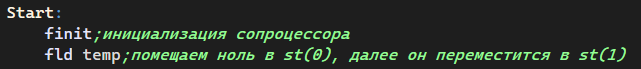


Рисунок 3 – инициализация сопроцессора и помещение 0 на вершину стека

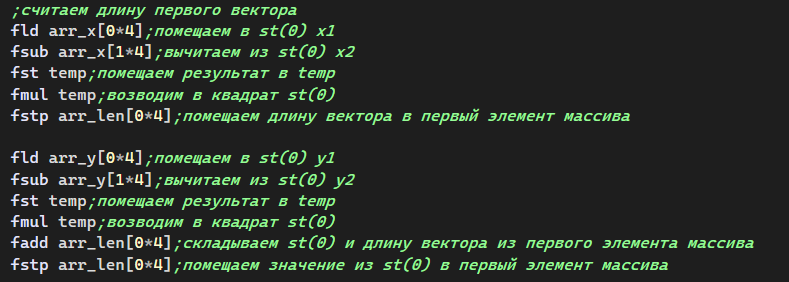


Рисунок 4 – подсчет длины первого вектора

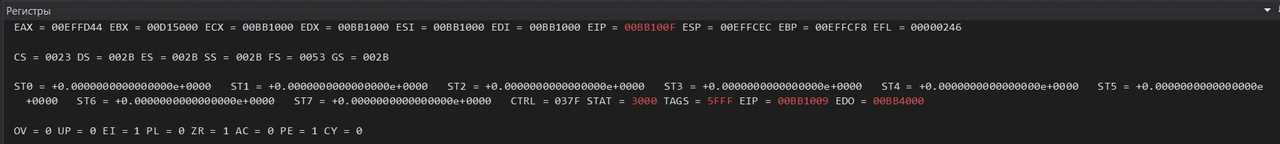


Рисунок 5 – помещаем x1 в st(0)

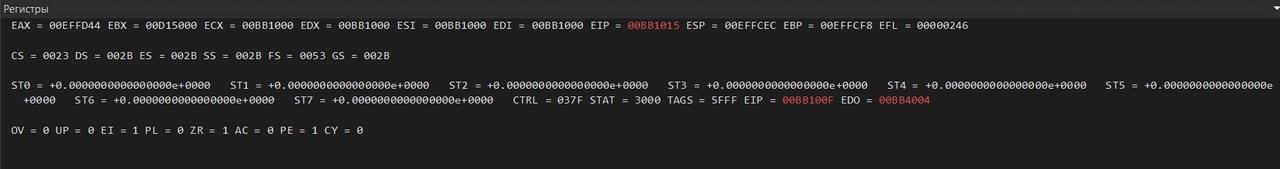


Рисунок 6 – вычитаем x2 из st(0)

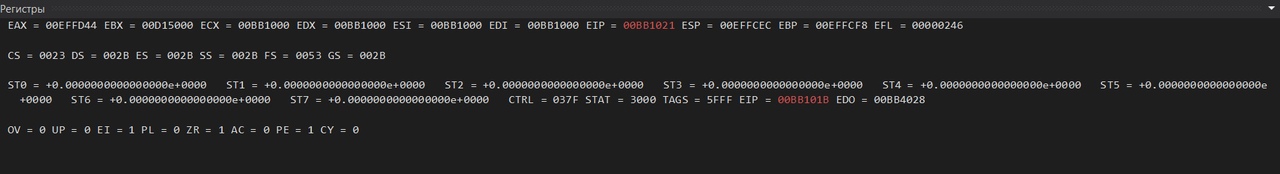


Рисунок 7 - Возвели в квадрат st(0) и затем поместили st(0) в первый элемент arr\_len

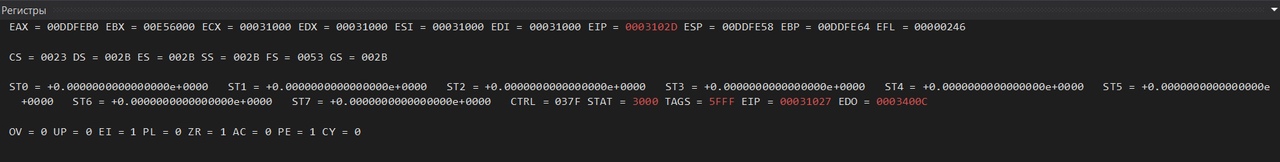


Рисунок 8 - Поместили y1 на вершину стека

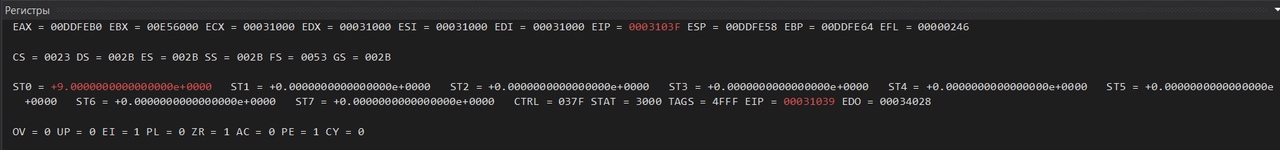


Рисунок 9 - Вычли y2 из st(0)

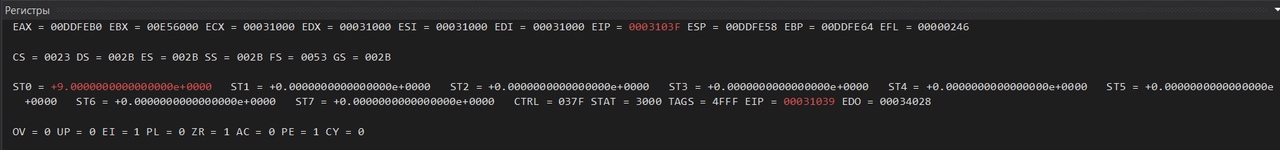


Рисунок 10 – Копируем значение в temp  
и возводим в квадрат st(0)

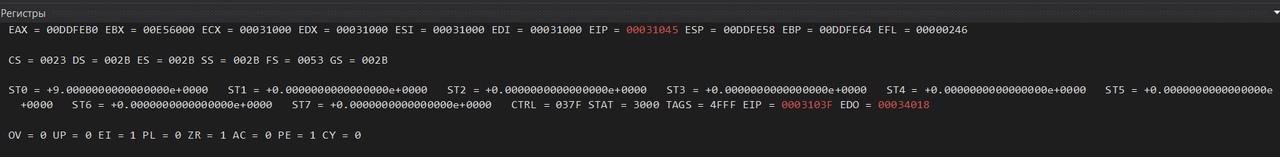


Рисунок 11 – Складываем st(0) и значение первого элемента массива arr\_len

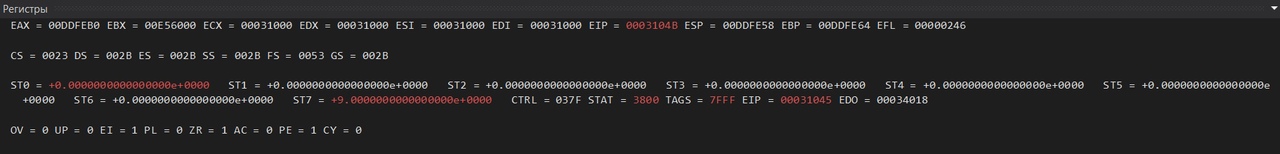


Рисунок 12 – Поместили значение из st(0) в первый элемент массива arr\_len. получили длину первого вектора

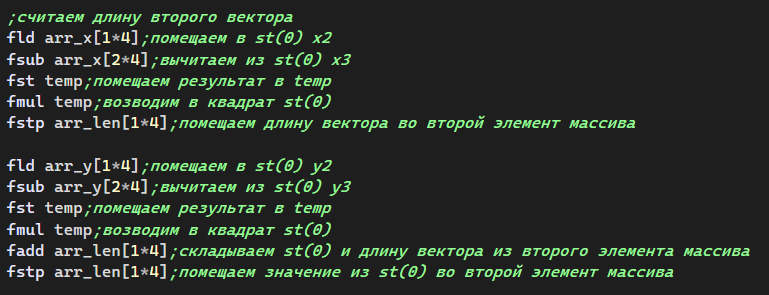


Рисунок 13 – Считаем длину второго вектора

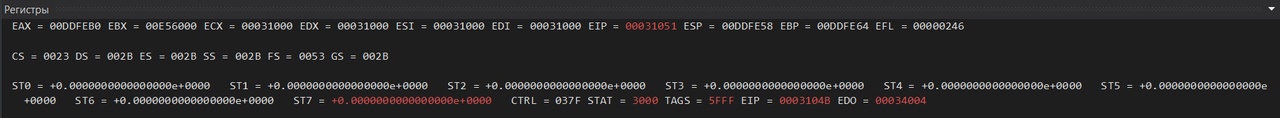


Рисунок 14 – Поместили в st(0) x2

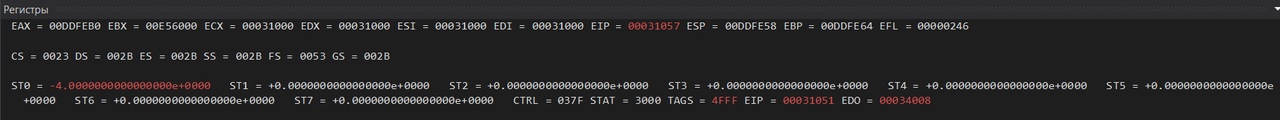


Рисунок 15 – Вычли x3 из st(0)

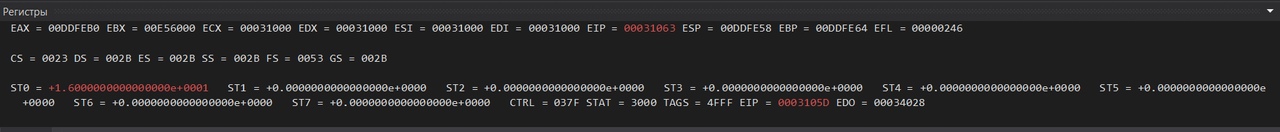


Рисунок 16 – Возвели в квадрат st(0)

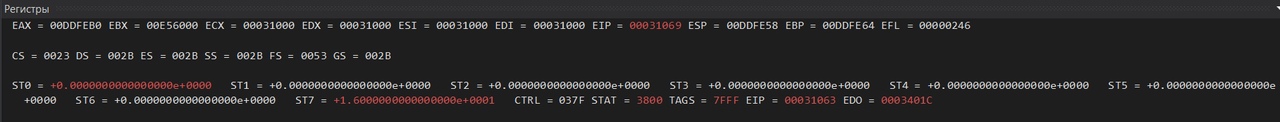


Рисунок 17 – Поместили значение из st(0) во второй элемент массива arr\_len

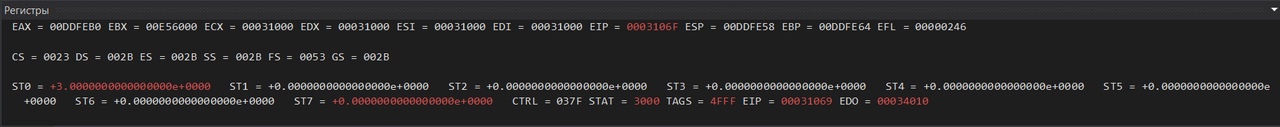


Рисунок 18 – Поместили в st(0) y2

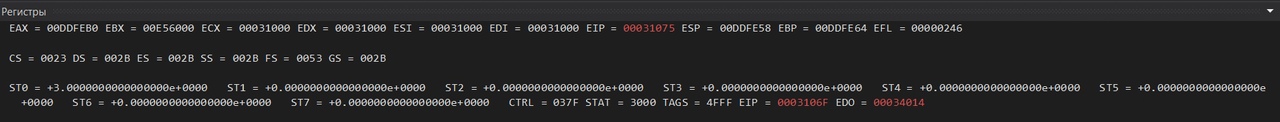


Рисунок 19 – Вычли из st(0) y3



Рисунок 20 – Возвели в квадрат st(0)

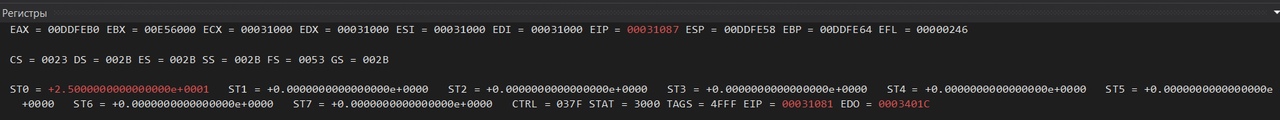


Рисунок 21 – Сложили st(0) и значение из второго элемента массива arr\_len



Рисунок 22 – Поместили значение st(0) во второй элемент массива arr\_len; посчитана длина второго вектора

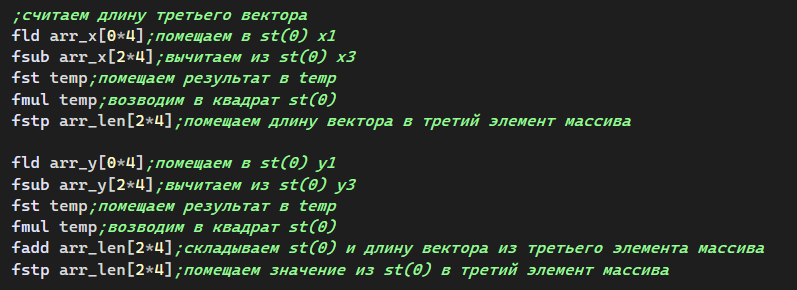


Рисунок 23 – Считаем длину третьего вектора



Рисунок 24 – Поместили x1 в st(0)

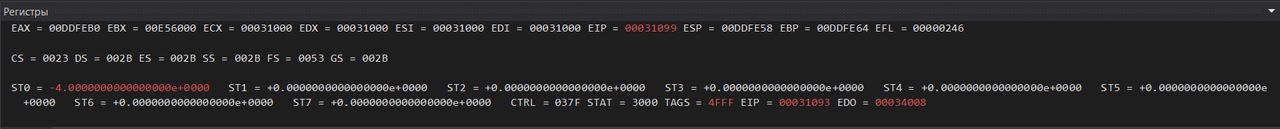


Рисунок 25 – Вычли x3 из st(0)



Рисунок 26 – Возводим в квадрат st(0)



Рисунок 27 – Помещаем st(0) в третий элемент массива arr\_len

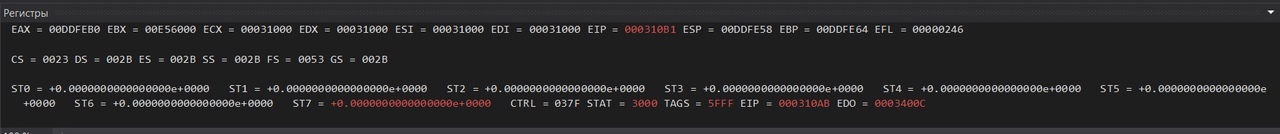


Рисунок 28 – Поместили в st(0) y1

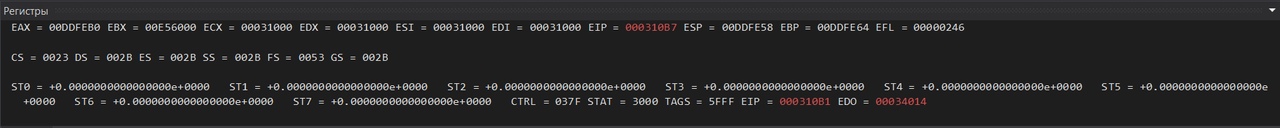


Рисунок 29 – Вычли y2 из st(0)

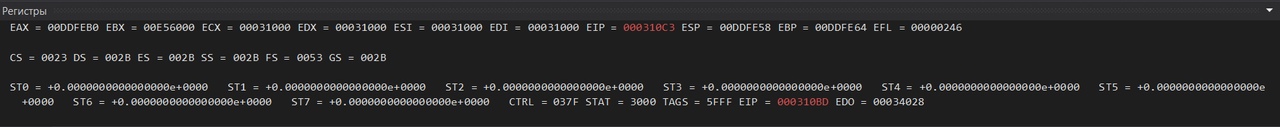


Рисунок 30 – Возвели в квадрат st(0)

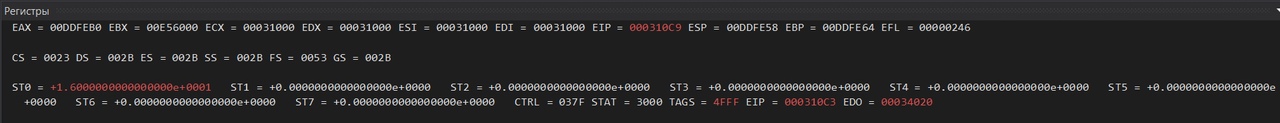


Рисунок 31 – Сложили st(0) и третий элемент массива arr\_len

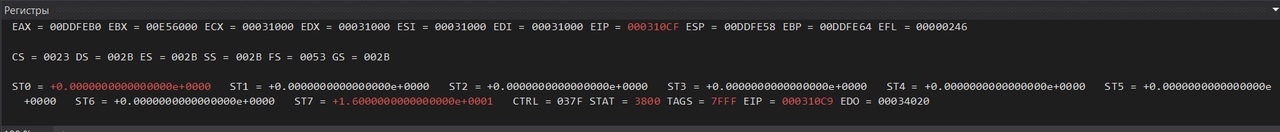


Рисунок 32 – Поместили st(0) в третий элемент массива arr\_len; длина третьего вектора посчитана

Далее считаем результат

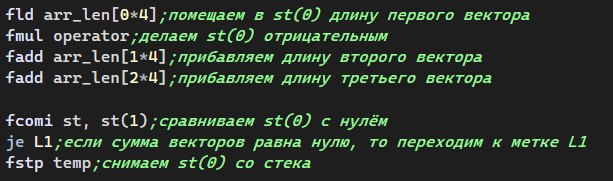


Рисунок 33 – Делаем отрицательным длину первого вектора и считаем сумму

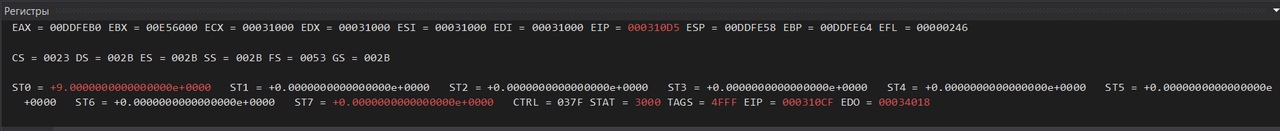


Рисунок 34 – Поместили в st(0) длину первого вектора

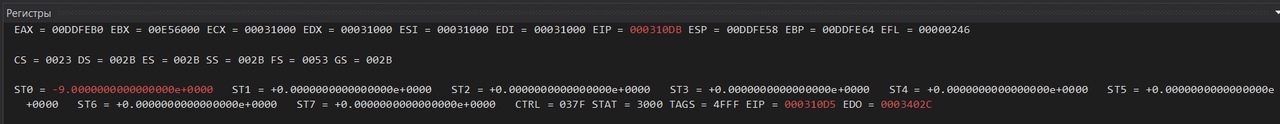


Рисунок 35 – Сделали его отрицательным

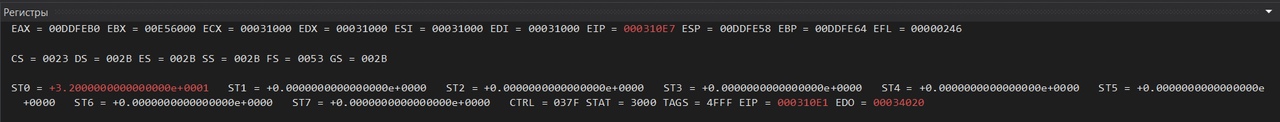


Рисунок 36 – Сложили со 2 и 3 длинами векторов

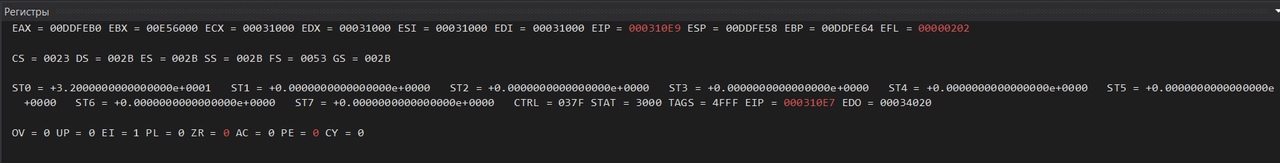


Рисунок 37 – Сравниваем с 0. 32 не равно нулю, значит идём дальше

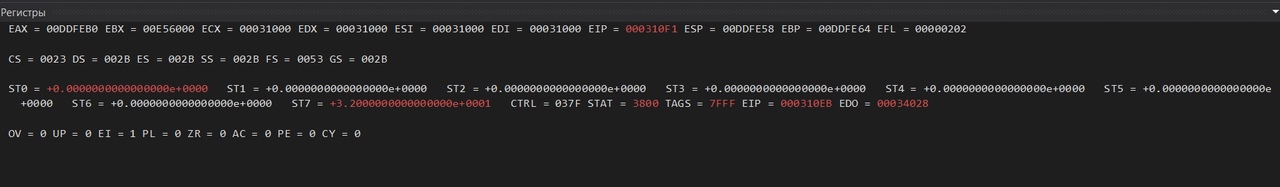


Рисунок 38 – Сняли st(0) с вершины стека

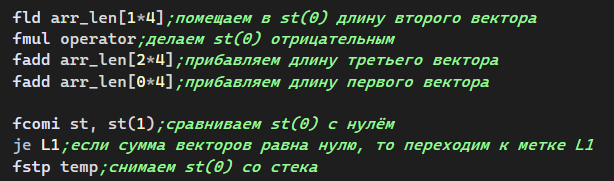


Рисунок 39 – Делаем отрицательной длину второго вектора и считаем сумму

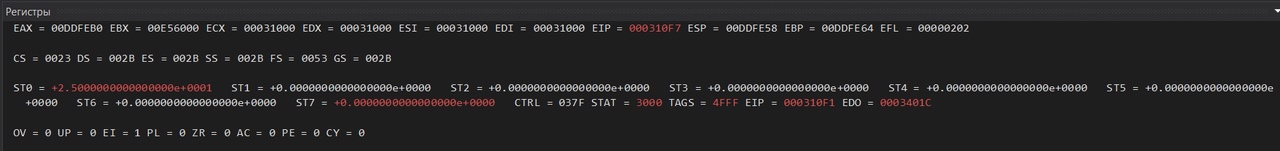


Рисунок 40 – Поместили в st(0) длину второго вектора

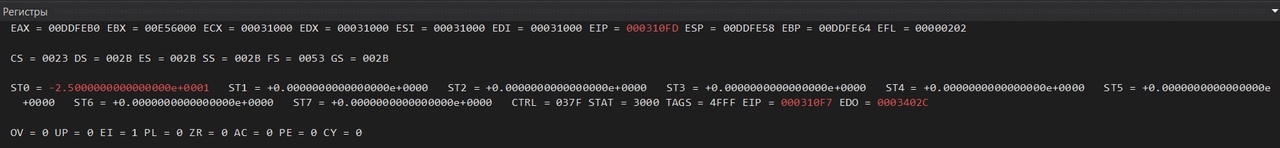


Рисунок 41 – Сделали его отрицательным

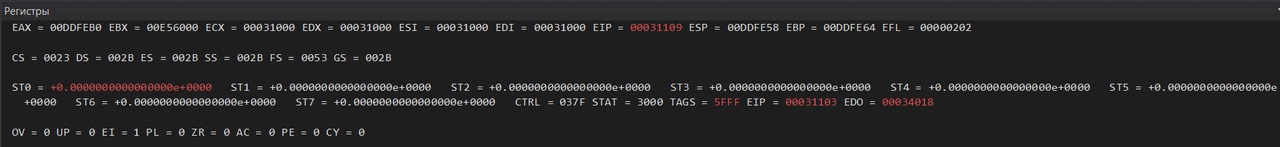


Рисунок 42 – Прибавили длину 1 и 3 вектора

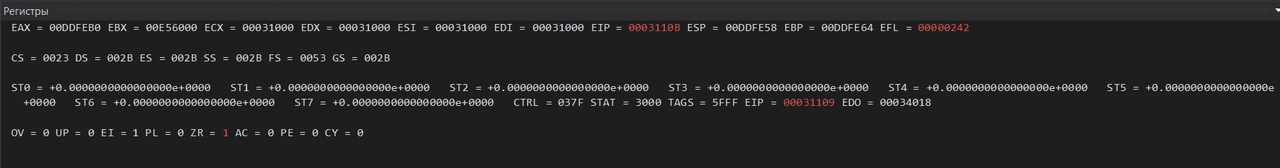


Рисунок 43 – сравниваем st(0) с 0. Сумма векторов = 0. значит треугольник прямоугольный.

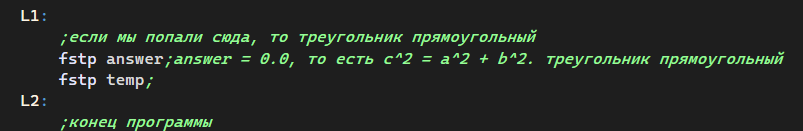


Рисунок 44 – переходим к метке L1

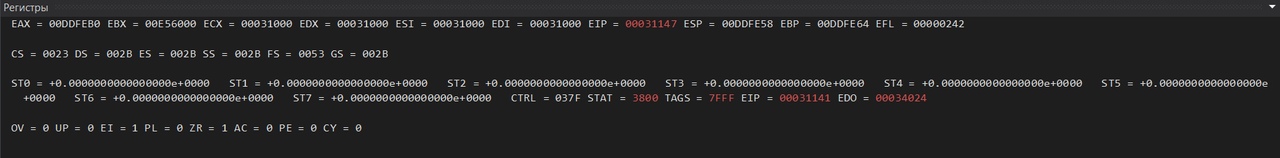


Рисунок 45 – Результат

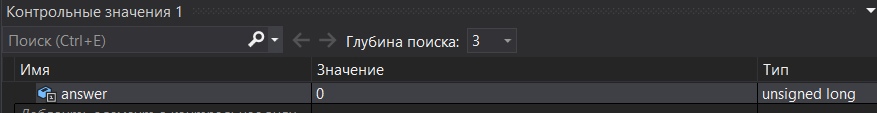


Рисунок 46 – answer = 0 - треугольник прямоугольный

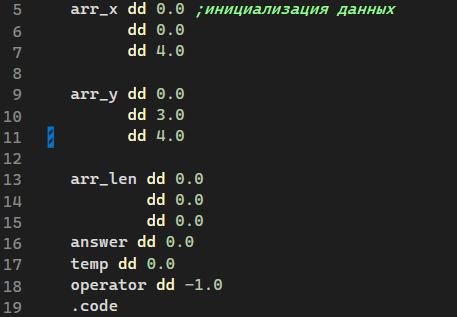


Рисунок 47 – Изменили начальные данные

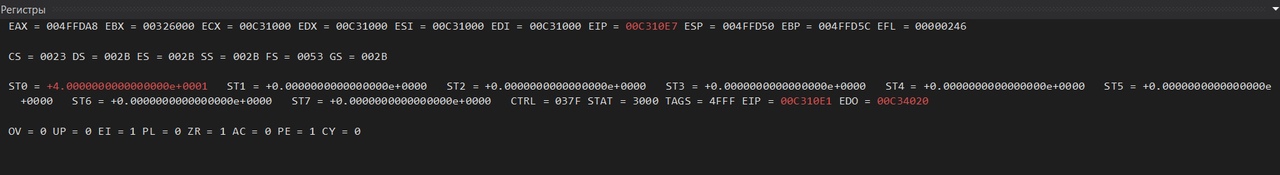


Рисунок 48 – Если первый элемент массива arr\_len сделать отрицательным, то сумма векторов будет равна 40

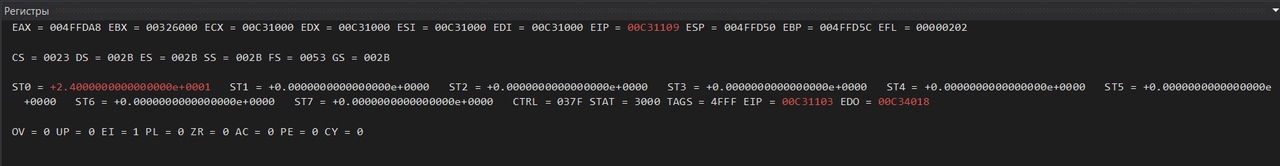
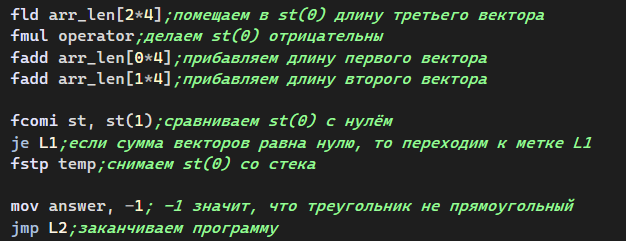


Рисунок 49 – Если второй элемент массива arr\_len сделать отрицательным, то сумма векторов будет равна 24



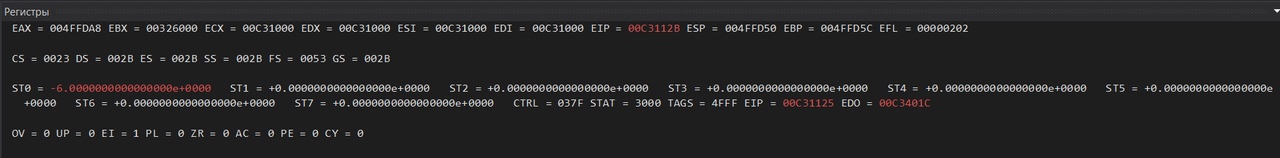
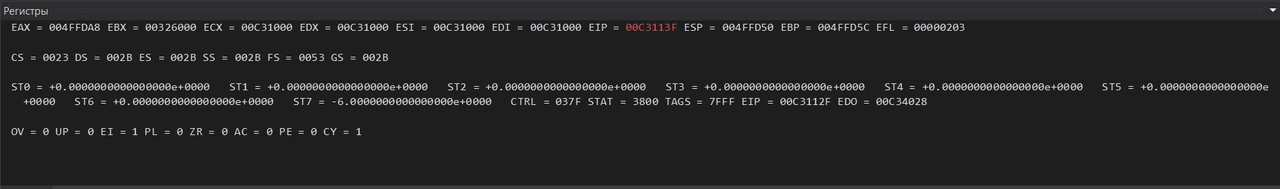


Рисунок 50 – Если третий элемент массива arr\_len сделать отрицательным, то сумма векторов будет равна -6



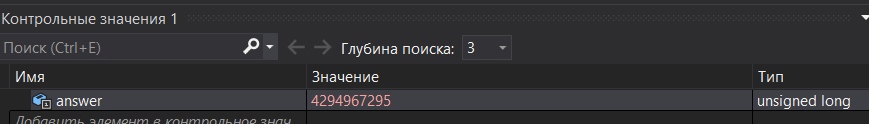


Рисунок 51 – Сумма векторов не равна нулю ни при одном из этих трёх случаев, соответственно треугольник не является прямоугольным. Об этом свидетельствует answer = -1

**Код программы**

.686

.model flat,stdcall

.stack 100h

.data

arr\_x *dd 0.0 ;инициализаци*я данных

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

ExitPr*ocess PROTO STDCALL :DWORD*

Start:

*finit;инициализация сопроцессора*

*fld temp;помещаем* *ноль в st(0), далее он перемес*тится в st(1)

*;считаем длину пе*рвого вектора

*fld arr\_x[0\*4];помеща*ем в st(0*) x1*

*fsub arr\_x[1\*4];выч*итаем из s*t(0) x2*

*fst temp;помеща*ем результат в tem*p*

*fmul temp;возводим в квадрат st(0)*

*fstp ar*r\_len[0\*4];поме*щаем длину вектора в* первый элемент *массива*

*fld arr\_y*[0\*4];пом*ещаем в st(0) y1*

*fsub ar*r\_y[1\*4];в*ычитаем из st(0) y2*

*fst* temp;помещаем рез*ультат в temp*

*fmul temp;возводим в квадрат st(0)*

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

*fstp arr\_len[0\*4];помещаем з*начение из st(0*) в первый элемент м*ассива

;счит*аем длину второго век*тора

fl*d arr\_x[1\*4];помещаем в st*(0) x2

f*sub arr\_x[2\*4];вычитаем и*з st(0) x3

fst t*emp;помещаем результат в temp*

*fmul temp;возводи*м в квадрат st(*0)*

*fstp arr\_len[1\**4];помещаем длин*у вектора во второй э*лемент ма*ссива*

*fld arr\_y[1\*4];п*омещаем в *st(0) y2*

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

*fst temp;помещаем результат в temp*

*fmul temp;возводи*м в квадрат st(0)

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

*fs*tp arr\_len[1\*4]*;помещаем значение и*з st(0) во второ*й 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 результ*ат в temp

fmu*l temp;возводим в кв*адрат st(0)

fs*tp arr\_len[2\*4];помещ*аем 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 tem*p;возводим в квадрат st(0)*

*fadd arr\_len[2\*4];склад*ы*ваем st(0) и длину* вектора из треть*его элемента массива*

*fstp arr\_len[2\*4*];помещаем зна*чение из st(0) в третий эле*мент массива

;*считаем результат*

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

*fmul* operator;делаем s*t(0) отрицательным*

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

*fadd arr\_len[*2\*4];приба*вляем длину третьего ве*ктора

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

*je* L1;если сумма *векторов равна нулю, то пе*реходим к метке L1

*fstp temp;снимаем st(0) со стек*а

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

fmu*l operator;делаем st(0) о*трицат*ельным*

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

fad*d arr\_len[0\*4];прибавля*ем длину первого *вектора*

*fcomi st, st(1);сравниваем s*t(0) с нулём

*je L1;если сумма векторов* равна нулю, то пе*реходим к метке L1*

*fstp temp;сн*имаем st(0) со сте*ка*

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

*fmul ope*rator;*делаем st(0) отрицательны*

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

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

*fcomi st, st(*1);срав*ниваем st(0) с нулём*

je L1*;если сумма векторов равна нулю, то переходим к ме*тке L1

fstp *temp;снимаем st(0) со стека*

*mov answer, -1; -1 значит, что т*реугольник *н*е прям*оугольный*

*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

https://github.com/LLIkoJIbHuk/mpp