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

«ПРИНЦИПЫ

«ИСПОЛЬЗОВАНИЕ МАТЕМАТИЧЕСКОГО СОПРОЦЕССОРА»

Вариант №15

|  |  |  |
| --- | --- | --- |
| Выполнил: студент гр. ИНБс-3301-01-00 | \_\_\_\_\_\_\_\_\_\_\_\_ | С.А. Симахин |
| Проверил: доцент кафедры РЭС | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | М. А. Земцов |

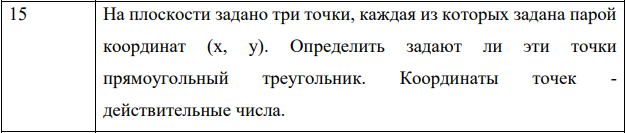
#### Киров 2025

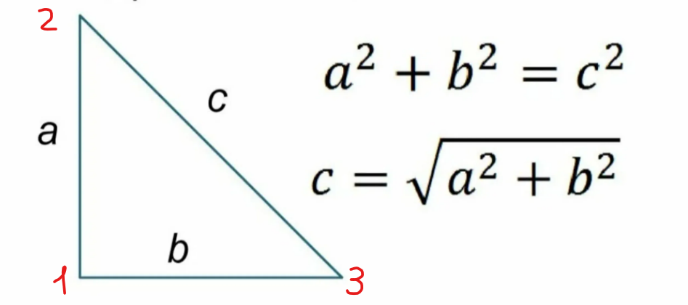
**Цель работы:** изучение принципов выполнения арифметических команд с помощью математического сопроцессора FPU микропроцессоров с архитектурой x86.

**Ход работы:**

Исходные данные представлены в таблице 1 и пояснении ниже.

Таблица 1 – Исходные данные



Для проверки, является ли фигура прямоугольным треугольником можно использовать теорему Пифагора:

a – между точками 2 и 1

b – между точками 3 и 1

c – между точками 3 и 2

Пример:

Для нахождения расстояния между точками 2 и 1 (длина а) можно использовать формулу:

Преобразуя теорему Пифагора в координатный вид, получим:

Далее упростим формулу, получим конечный вид проверки, которая используется в коде:

Далее начинаем разбор кода:

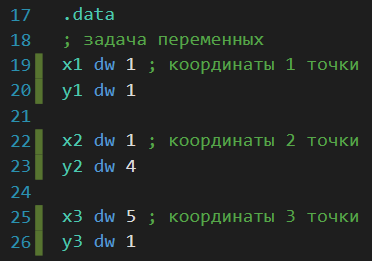


Рисунок 1 – Задаем координаты точек треугольника

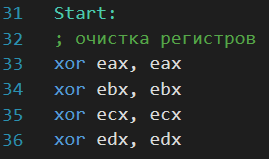
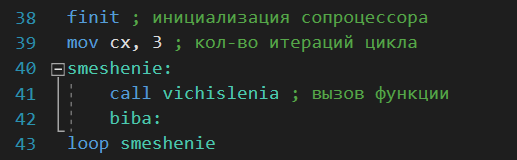


Рисунок 2 – очистка регистров



38 строка: командой **finit** производим инициализацию сопроцессора с плавающей запятой (FPU), также данная команда выполняет:  
 - сброс всех управляющих регистров   
 - сброс параметров точности вычислений и округлений  
 - очистка стека FPU

****39 строка: количество итераций для цикла **smeshenie**

40 строка: начало цикла **smeshenie**

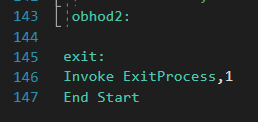
41 строка: вызов функции **vishislenia**

42 строка: выход точки **biba**

43 строка: конец цикла **smeshenie**

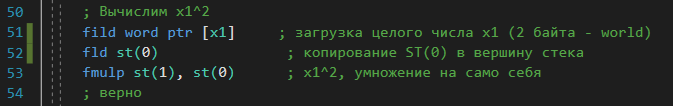
В случае прохождения всех 3-ех итераций цикла **smeshenie** идет переход в точку **obhod2** и выход из программы:





Далее все действия будут происходить в функции **vichislenia:**

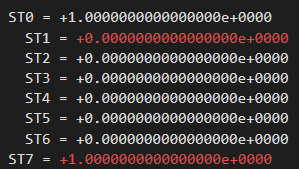




51 строка:

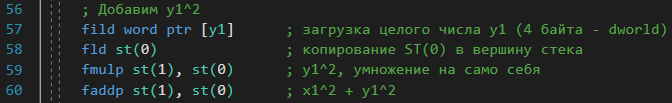
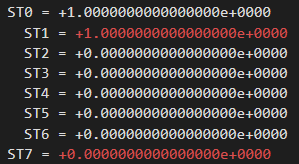


52 строка:

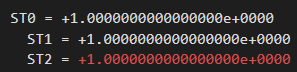


x1^2 = 1 \* 1 = 1 🡪 Верно

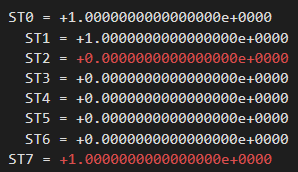
53 строка:



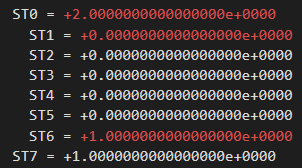
57 строка:



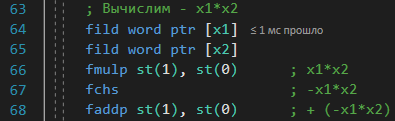
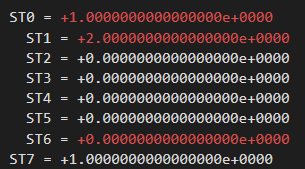
58 строка:

59 строка:

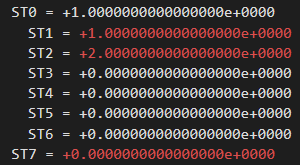
<- y1^2 = 1 \* 1 = 1 🡪 Верно

60 строка:

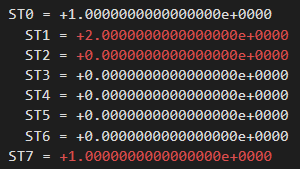
<- x1^2 + y1^2 = 1 + 1 = 2 🡪 Верно



64 строка:

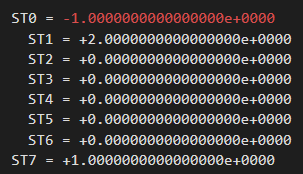


65 строка:

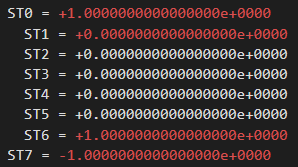


<- x1 \* x2 = 1 \* 1 = 1 🡪 Верно

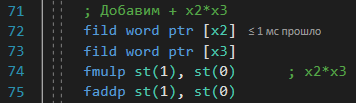
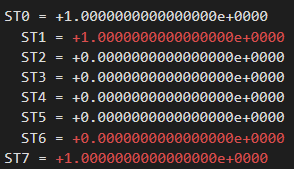
66 строка:

67 строка:

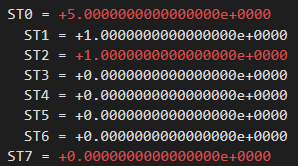
<- инверсия знака 1🡪 Верно

68 строка:

<- 2 – 1 = 1🡪 Верно

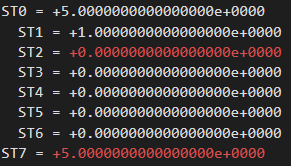


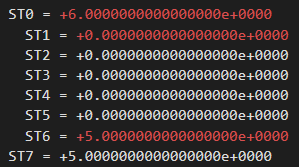
72 строка:



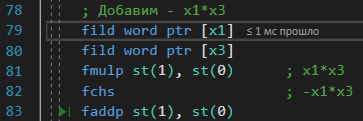
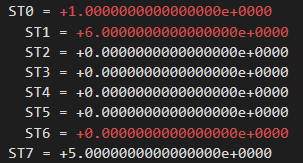
73 строка:

<- 1 \* 5 = 5🡪 Верно

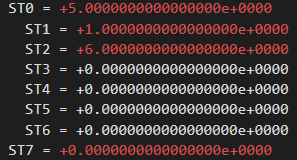
74 строка:

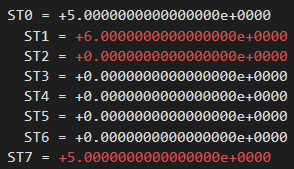
75 строка:

<- 1 + 5 = 6🡪 Верно



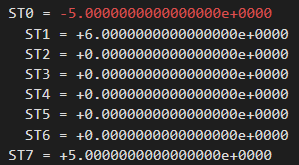
79 строка:

80 строка:

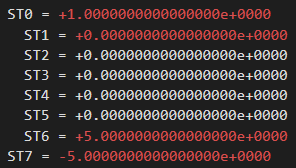


<- 1 \* 5 = 5 🡪 Верно

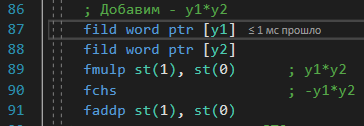
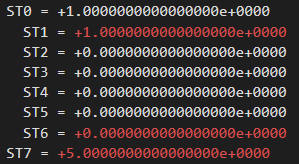
81 строка:

82 строка:

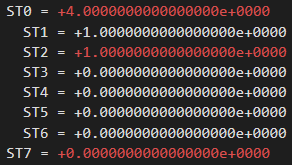
<- инверсия знака 5🡪 Верно



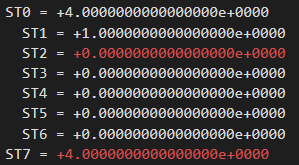
83 строка:



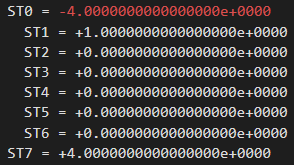
87 строка:



88 строка:

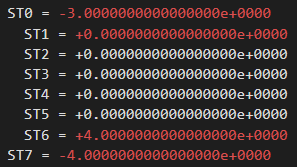
89 строка:

<- 1 \* 4 = 4🡪 Верно



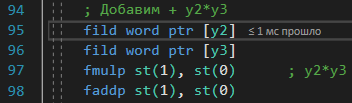
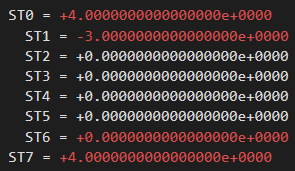
<- инверсия знака 4🡪 Верно

90 строка:

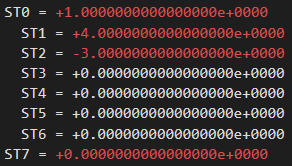


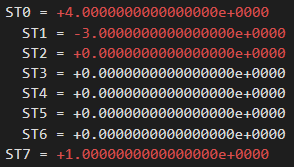
<- 1 – 4 = 3🡪 Верно

91 строка:



95 строка:

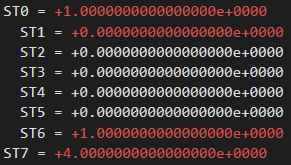
96 строка:

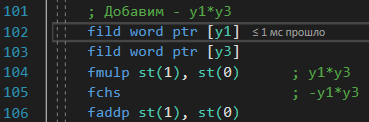
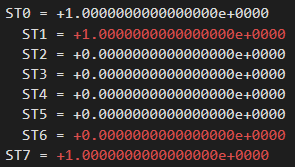


<- 4 \* 1 = 4🡪 Верно

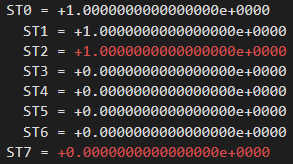
97 строка:

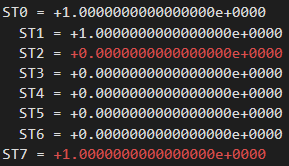
<- 4 -3 = 1 🡪 Верно

98 строка:



102 строка:

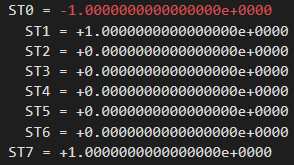
103 строка:



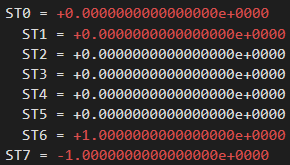
<- 1 \* 1 = 1🡪 Верно

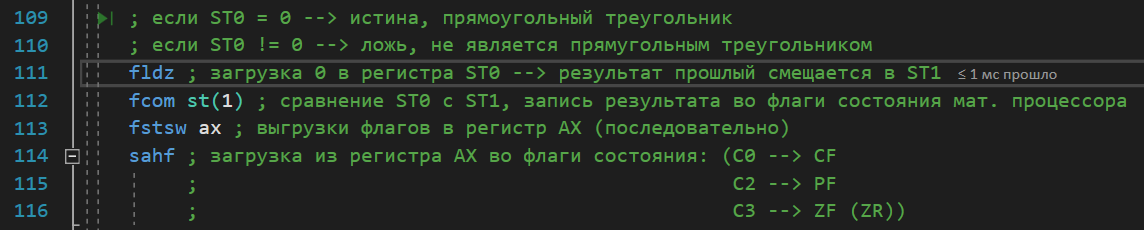
104 строка:

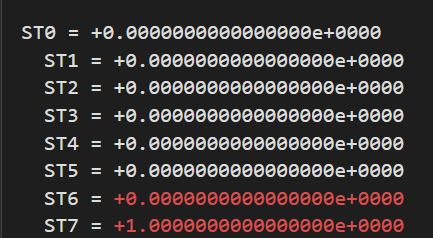
<- инверсия знака 1🡪 Верно

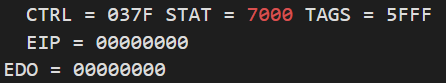
105 строка:

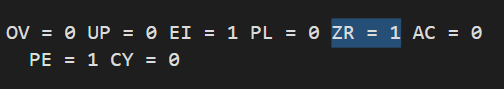
<- 1 – 1 = 0🡪 Верно

106 строка:



111 строка:

112 строка:



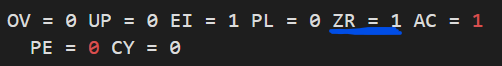
Флаги FPU: (флаги состояния мат. процессора)

С0 = С2 = С3 = 0, если ST(0) больше

С0 = 1, С2 = С3 = 0, если ST(0) меньше

С0 = С2 = 0, С3 = 1, если ST(0) равно

С0 = С2 = С3 = 1, если ошибка сравнения

113 строка:

114 строка:

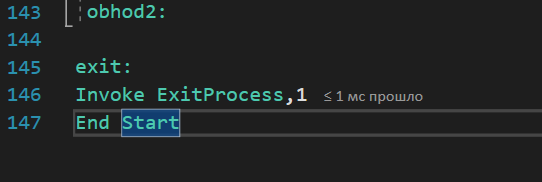
Флаги переписываются:

C0 🡪 CF (флаг переноса)

C2 🡪 PF (флаг паритета)

C3 🡪 ZF (флаг нуля)

118 строка:

143 строка:

Код программы:

; 15 вариант

; На плоскости задано три точки, каждая из которых задана парой

; координат (x, y). Определить задают ли эти точки

; прямоугольный треугольник. Координаты точек -

; действительные числа.

; для проверки, является ли фигура прямоугольным треугольником, можно использовать теорему Пифагора

;

; Формула, при истинности которой объект является прямоугольным треугольником:

; x1^2+y1^2-x1x2+x2x3-x1x3-y1y2+y2y3-y1y3 = 0

.686

.model flat, stdcall

.stack 100h

.data

; задача переменных

x1 dw 1 ; координаты 1 точки

y1 dw 1

x2 dw 1 ; координаты 2 точки

y2 dw 4

x3 dw 5 ; координаты 3 точки

y3 dw 1

.code

ExitProcess PROTO STDCALL :DWORD

Start:

; очистка регистров

xor eax, eax

xor ebx, ebx

xor ecx, ecx

xor edx, edx

finit ; инициализация сопроцессора

mov cx, 3 ; кол-во итераций цикла

smeshenie:

call vichislenia ; вызов функции

biba:

loop smeshenie

jmp obhod2

vichislenia PROC ; начало функции

; Вычислим x1^2

fild word ptr [x1] ; загрузка целого числа x1 (2 байта - world)

fld st(0) ; копирование ST(0) в вершину стека

fmulp st(1), st(0) ; x1^2, умножение на само себя

; верно

; Добавим y1^2

fild word ptr [y1] ; загрузка целого числа y1 (4 байта - dworld)

fld st(0) ; копирование ST(0) в вершину стека

fmulp st(1), st(0) ; y1^2, умножение на само себя

faddp st(1), st(0) ; x1^2 + y1^2

; верно

; Вычислим - x1\*x2

fild word ptr [x1]

fild word ptr [x2]

fmulp st(1), st(0) ; x1\*x2

fchs ; -x1\*x2

faddp st(1), st(0) ; + (-x1\*x2)

; верно, результат в ST0

; Добавим + x2\*x3

fild word ptr [x2]

fild word ptr [x3]

fmulp st(1), st(0) ; x2\*x3

faddp st(1), st(0)

; верно, результат в ST0

; Добавим - x1\*x3

fild word ptr [x1]

fild word ptr [x3]

fmulp st(1), st(0) ; x1\*x3

fchs ; -x1\*x3

faddp st(1), st(0)

; верно, результат в ST0

; Добавим - y1\*y2

fild word ptr [y1]

fild word ptr [y2]

fmulp st(1), st(0) ; y1\*y2

fchs ; -y1\*y2

faddp st(1), st(0)

; верно, результат в ST0

; Добавим + y2\*y3

fild word ptr [y2]

fild word ptr [y3]

fmulp st(1), st(0) ; y2\*y3

faddp st(1), st(0)

; верно, результат в ST0

; Добавим - y1\*y3

fild word ptr [y1]

fild word ptr [y3]

fmulp st(1), st(0) ; y1\*y3

fchs ; -y1\*y3

faddp st(1), st(0)

; верно, результат в ST0

; если ST0 = 0 --> истина, прямоугольный треугольник

; если ST0 != 0 --> ложь, не является прямугольным треугольником

fldz ; загрузка 0 в регистра ST0 --> результат прошлый смещается в ST1

fcom st(1) ; сравнение ST0 с ST1, запись результата во флаги состояния мат. процессора

fstsw ax ; выгрузки флагов в регистр AX (последовательно)

sahf ; загрузка из регистра AX во флаги состояния: (C0 --> CF

; C2 --> PF

; C3 --> ZF (ZR))

je obhod ; является прямоугольным

; не является прямоугольным --> повторная проверка со сдвигом координат

mov dx, x2

mov ax, x1

mov x2, ax

mov ax, x3

mov x3, dx

mov x1, ax

mov dx, y2

mov ax, y1

mov y2, ax

mov ax, y3

mov y3, dx

mov y1, ax

jmp biba

vichislenia ENDP ; конец функции

obhod:

; ZF (ZR) = результат

; если ZF (ZR) = 1 --> является прямоугольным

; если ZF (ZR) != 1 --> не является прямоугольным

obhod2:

exit:

Invoke ExitProcess,1

End Start

Вывод:

В данной работе я изучил принципы выполнения арифметических команд с помощью математического сопроцессора FPU микропроцессоров с архитектурой х86 в MASM.

* .686 – данная директива указывает ассемблеру использовать набор команд процессора Pentium Pro или Pentium II.
* .model flat - это несегментированная модель памяти, используемая в 32-разрядных операционных системах. В этой модели все данные и код программы размещаются в одном логическом адресном пространстве, что упрощает адресацию памяти. Сегментные регистры не используются для адресации данных, а вместо них используются 32-разрядные смещения
* .model stdcall - это соглашение о вызовах процедур, которое определяет порядок передачи параметров и очистки стека. При использовании stdcall, параметры передаются через стек в обратном порядке, а очистка стека производится вызываемой процедурой
* .stack100h – данная директива определяет размер стека программы. В данном случае, стек будет иметь размер 256 байт (100h — это шестнадцатеричное представление 256 в десятичной системе)
* ExitProcess PROTO STDCALL: DWORD:
  + ExitProcess:
    - Это имя функции из Windows API, которая завершает выполнение текущего процесса.
    - Она принимает один параметр — код завершения процесса (тип DWORD).
  + PROTO:
    - Ключевое слово MASM для объявления прототипа функции.
    - Прототип позволяет ассемблеру знать, как вызывать функцию, какие параметры она принимает и в каком порядке.
  + STDCALL:
    - Указывает соглашение о вызове функции.
    - В соглашении stdcall параметры передаются через стек в обратном порядке (сначала последний параметр), а очистка стека выполняется вызываемой функцией.
    - Это стандартное соглашение для большинства функций Windows API.
  + DWORD:
    - Описывает тип параметра функции. В данном случае функция принимает один параметр типа DWORD (32-битное беззнаковое целое число).
* API (Application Programming Interface) — это набор способов и правил, по которым различные программы общаются между собой и обмениваются данными.