**Московский авиационный институт**

**(национальный исследовательский университет)**

Институт №3

Системы управления, информатики и электроэнергетики

Кафедра 304

Вычислительные машины, системы и сети

**Отчет по лабораторной работе**

**по учебной дисциплине**

**«Языки Ассемблера»**

**на тему:**

***«***Команды передачи управления***»***

*Группа:* ***М30-109Б-19***

*Выполнил:*

***Кузнецов И.И.***

[*iluxandro@gmail.com*](mailto:iluxandro@gmail.com)

*Принял:*

***Ратников Максим Олегович***

***Москва 2020***

**Содержание.**

**Задание\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_3**

**Код программы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_4-5**

**Тесты\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_6-8**

**Вывод\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_9**

***Задание***

**Вариант №7**

Разработать программу реализации циклического процесса в соответствии с вариантом задания.

Вычисляется выражение типа:

y1, x<0

Y = y2 , x=0

y3 , x>0





***Код программы***

using namespace std;

#include <iostream>

short int result;

short int y;

short int a = 14;

short int b = 2;

short int c = -25;

short int d = -6;

short int x = -6;

short int i = 0;

void print\_ass() {

cout << "cx=" << i << "\tx: " << x << " result function: " << result << endl;

}

int main()

{

// { (a\*b-c) div x , x < 0

// Y = { (d-a\*b) , x = 0 x [-6;2] a = 14, b = 2, c = -25, d = -6

// { (a\*b\*x) div (c+d) , x > 0

cout << "c++: " << endl;

for (x; x <= 2; x++) {

if (x < 0) {

y = (a \* b - c) / x;

}

if (x == 0) {

y = d - (a \* b);

}

if (x > 0) {

y = (a \* b \* x) / (c + d);

}

cout << "x: " << x << " result function: " << y << endl;

}

cout << endl << "asm: " << endl;

x = -6;

\_\_asm {

mov ecx, 9 //количество проходов по циклу

cycle: //цикл

cmp x, 0 //сравниваем x с 0 получаем некоторый флаг

//jl y1 //если меньше 0 - условной переход к метке y1

je y2 //если равно 0 - условной переход к метке y2

jg y3 //если больше 0 - условной переход к метке y3

//y1: //y = (a \* b - c) / x;

mov ax, a //в регистр ax помещаем значение переменной a

imul b //умножаем b на ax сохраняя значение в ax

mov bx, c //в регистр bx помещаем значение переменной c

sub ax, bx //вычитаем из ax bx сохраняем в ax

mov bx, x //в регистр bx помещаем значение x

xor dx, dx //очищаем мусор лежащий в dx

cwd //расширение регистра ax на два регистра ax и dx

idiv bx //знаковое деление - делим ax на bx(x)

//в регистр dx попадет остаток от деления, а в ax делитель

mov result, ax //сохраняем результат

jmp end\_ass //прыг

end\_ass : //метка окончания блока

push ecx //складываем в стек ecx

mov i, cx //в i записываем значени регисра cx

call print\_ass //вызываем функцию вывода, выводим

inc x //x++

pop ecx //забираем из стека ecx

loop cycle //зацикливаемся

jmp end\_end //по окончанию цикла переходим на метку завершения

y2 : //y = d - a \* b;

mov ax, a //в регистр ax помещаем значение переменной a

imul b //умножаем b на ax сохраняя значение в ax

mov bx, d //в регистр bx помещаем значение переменной d

sub bx, ax //вычитаем из bx ax сохраняя полученный результат в ax

mov result, bx //сохраняем результат

jmp end\_ass //прыгаем на метку конца блока

y3 : //y = (a \* b \* x) / (c + d);

mov bx, c //в регистр bx помещаем значение переменной c

add bx, d //складываем регистр bx и переменную d сохраняя результ в регистре bx (c+d)

mov result, bx //сохраним результат прошлых вычислений в буфер

mov ax, x //в регистр ax помещаем значение переменной x

mov bx, b //в регистр bx помещаем значение переменной b

imul bx //умножаем ax на bx сохраняя результат в ax (b\*x)

mov bx, a //в регистр bx помещаем значение переменной a

imul bx //умножаем прошлый результат на bx сохраняя результат в ax (a\*b\*x)

mov bx, result //из буфера вынимает результат прошлых вычислений

xor dx, dx //очищаем мусор лежащий в dx

cwd //расширение регистра ax на два регистра ax и dx

idiv bx //знаковое деление - делим ax на bx

//в регистр dx попадет остаток от деления, а в ax делитель

mov result, ax //сохраняем результат

jmp end\_ass //прыгаем на метку конца блока

end\_end : //метка конца

}

system("pause");

***Тесты***

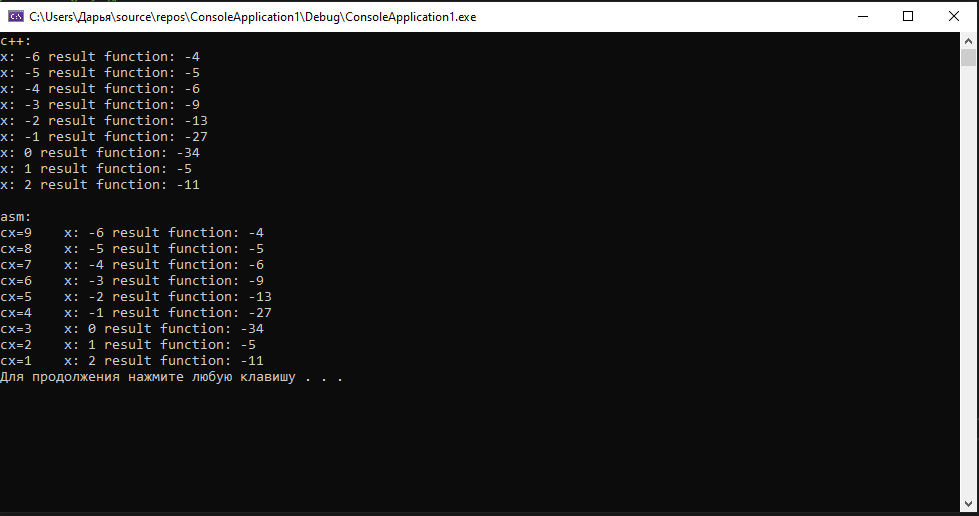
**Тест 1**

***Цель:*** проверить работу программы при изначальных исходных данных

a = 14, b = 2, c = 1, d = -6

***Ожидаемый результат****:* корректная работа

***Полученный результат:***



**Тест 2**

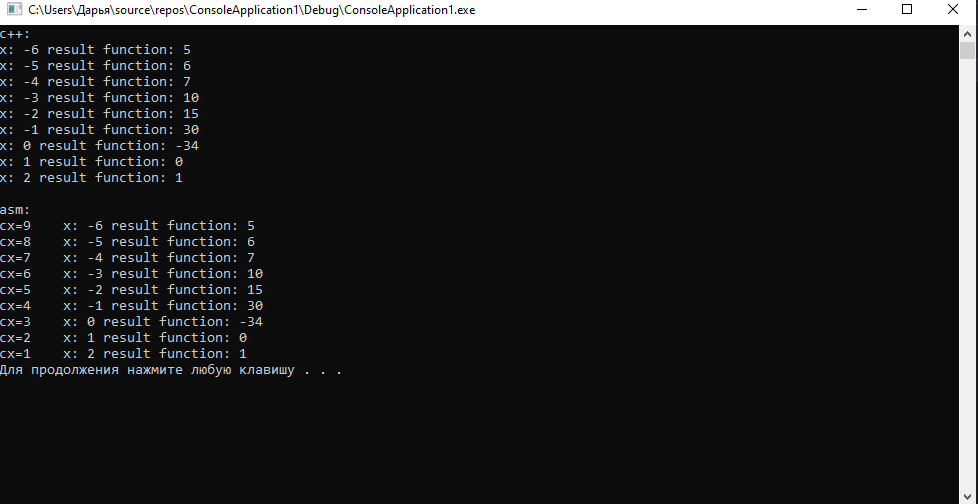
***Цель:*** проверить работу программы, когда в первом блоке (y1 область)

**(a \* b - c)** / x получается отрицательное выражение, при переменных

a = 14, b = 2, c = 58, d = -6

***Ожидаемый результат****:* корректная работа

***Полученный результат:***



**Тест 3**

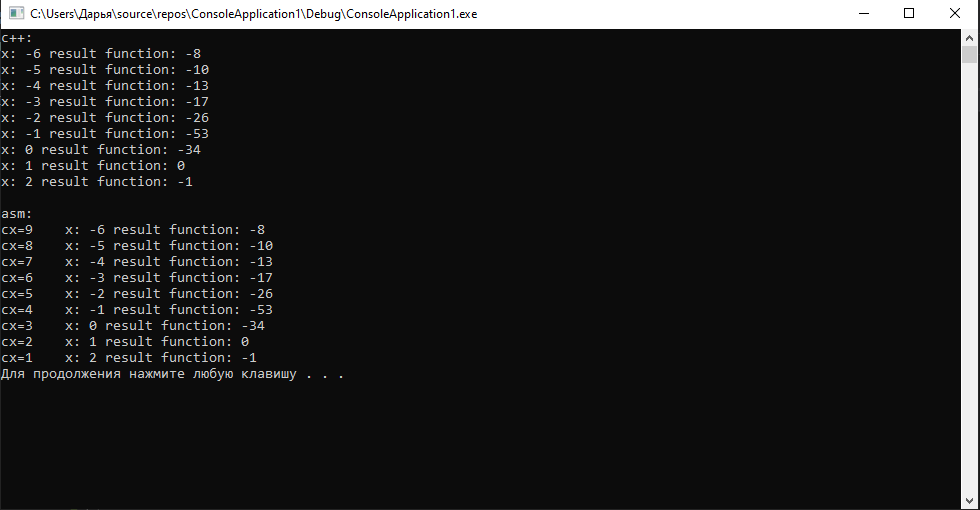
***Цель:*** проверить работу программы, когда во втором блоке (y3 область)

(a \* b \* x) / (**c + d)** получается отрицательное выражение, при переменных

a = 14, b = 2, c = -25, d = -6

***Ожидаемый результат****:* корректная работа

***Полученный результат:***



**Вывод:**

Мы изучили циклы, условные и безусловные переходы, метки.

Программа работает без ошибок. Набор тестов считают достаточным.