**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:**

***«***Преобразование данных при выводе в программе на ЯА***»***

*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 в Турбо-ассемблере для реализации циклического процесса в соответствии с вариантом задания, выполненным на занятии 2

*Задание 2:* 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***

org 100h

mov cx, 9 ; количество проходов по циклу

cycle: ; метка цикла

cmp [x], 0 ; сравниваем x с 0 получаем некоторый флаг

je y2 ; если равно 0 – условной переход к метке y2

jg y3 ; если больше 0 – условной переход к метке y3

; 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)

push ax ; закидываем в стек ax

jmp end\_ass ; прыг

end\_ass: ; метка окончания блока

pop ax ; забираем из стека ax

push cx ; на всякий случай закидываем в стек значение cx

call OutInt ; вызываем функцию вывода результата

call print\_nl ; вызываем функцию вывода пробела

inc [x] ; x++

pop cx ; забираем из стека cx

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 ax, bx ; сохраняем результат

push ax ; закидываем в стек ax

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] ; c + d

push bx ; сохраним результат прошлых вычислений в стек

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 ; (a \* b \* x)

pop bx ; из стека 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

push ax ; закидываем в стек ax

jmp end\_ass ; прыгаем на метку конца блока

end\_end: ; метка конца

mov ah, 1 ; ждемс нажатия любой кнопки

int 21h

ret

OutInt proc

test ax, ax ; проверяем число на знак

jns oi1 ; jump if not sign

mov cx, ax

mov ah, 02h ; если число отрицательное

mov dl, '-' ; то выведем минус и

int 21h ; оставим его модуль

mov ax, cx ; количество цифр будет держать в cx

neg ax ; меняем знак

oi1:

xor cx, cx ; чистим мусор

mov bx, 10 ; основание 10 СС

oi2:

xor dx, dx ; чистим мусор

div bx ; делим на основанием СС

; в остатке получается последняя цифра

push dx ; сразу выводить не будем поэтому закинем это в стек

inc cx ; увеличиваем количество цифр

test ax, ax ; с частным делаем то же самое

jnz oi2 ; пока не останется ноль делаем все тоже самое

mov ah, 02h ; приступаем к выводу

oi3:

pop dx ; извлекаем из стека очередную цифру и выводим

add dl, '0'

int 21h

loop oi3 ; повторим столько раз, сколько цифр насчитали

ret

OutInt endp

print\_nl proc ; идем на след строку

pusha ; закидываем все регистры в стек

mov ah, 2

mov dl, 0Dh

int 21h

mov dl, 0Ah

int 21h

popa ; забираем все регистры из стека

ret

endp

a dw 14 ; 14

b dw 2 ; 2

c dw 1 ; 1

d dw -6 ; -6

x dw -6 ; -6

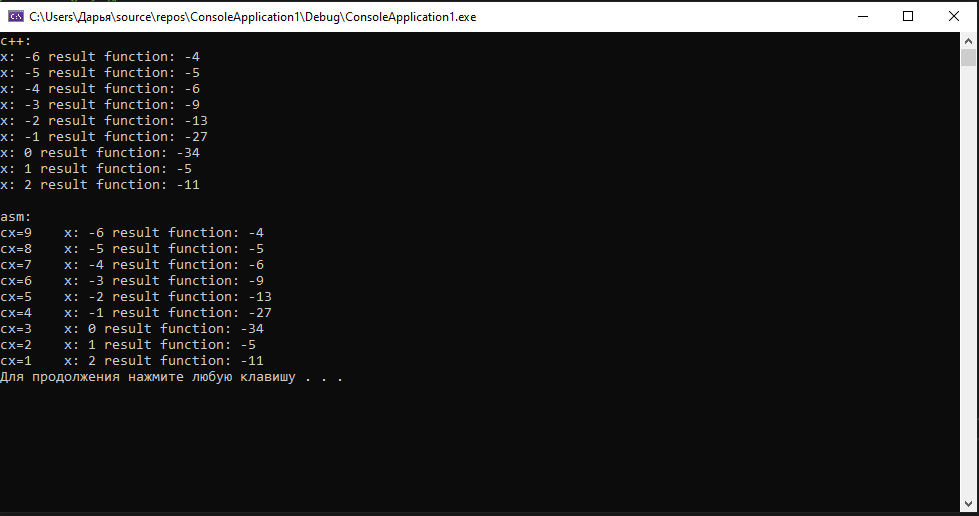
***Тесты***

**Тест 1**

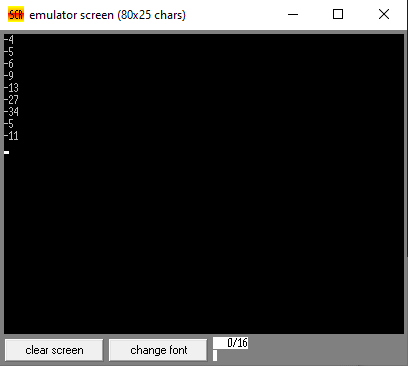
***Цель:*** проверить работу программы при изначальных исходных данных

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

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



***Полученный результат:***



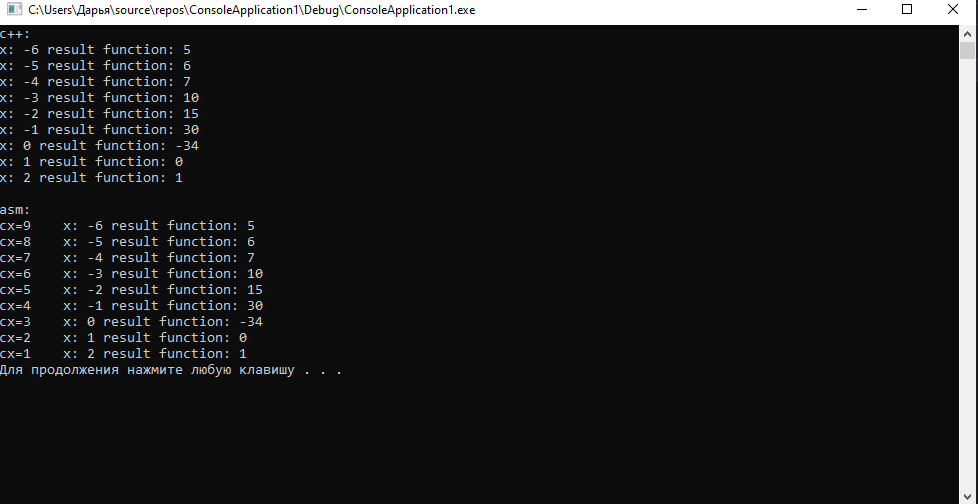
**Тест 2**

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

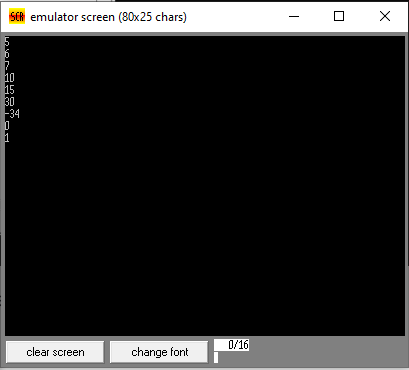
**(a \* b - c)** / x получается отрицательное выражение, при переменных

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

***Ожидаемый результат****:*



***Полученный результат:***



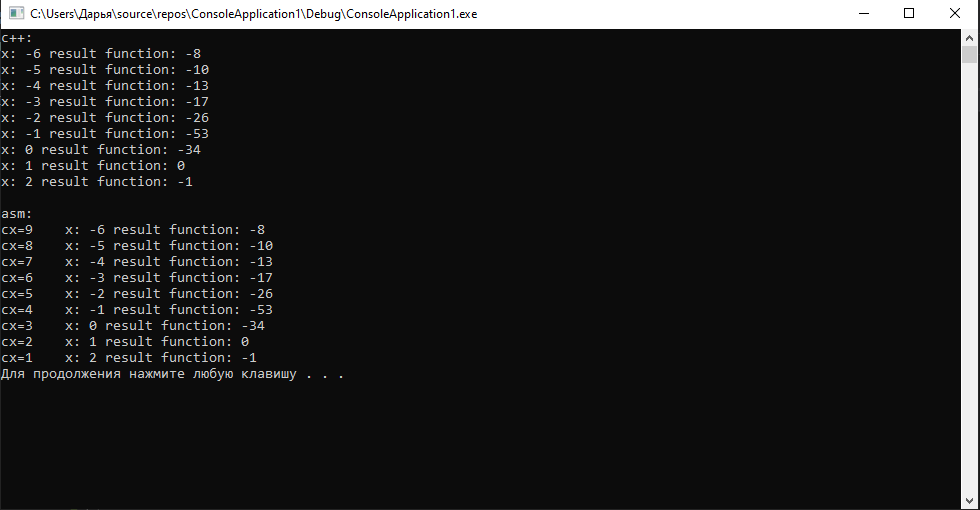
**Тест 3**

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

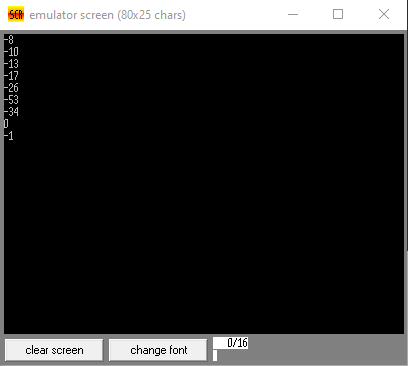
(a \* b \* x) / (**c + d)** получается отрицательное выражение, при переменных

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

***Полученный результат:***



***Ожидаемый результат****:*



**Вывод:**

Мы изучили циклы, условные и безусловные переходы, метки. И смогли преообразовать вывод из ассемблерной вставки в эмулятор.

Программа работает без ошибок. Набор тестов считают достаточным.