**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА» (СПбГУТ)**

КАФЕДРА ПРОГРАММНОЙ ИНЖЕНЕРИИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Дисциплина: «Машинно-зависимые языки программированияе»

Лабораторная работа №1.

**Тема: «Вычисление целочисленных арифметических выражений»**

Выполнил:

Студент группы ИКПИ-21

Козлова А. И.

Подпись \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Принял:

Старший преподаватель кафедры ПИиВТ

Анохин Ю. В.

Подпись \_\_\_\_\_\_\_\_\_\_\_\_\_\_

**А. *Постановка задачи***

Имеется формула, по которой необходимо вычислить результат на С и на Assembler для разных типов данных и введённых пользователем значений. Значения принимает модуль на С



**Б. *Разработка алгоритма***

С помощью глобальных переменных в C и external-объявленных переменных будет осуществляться обмен данными между модулями на Assembler и C. С помощью операций mov, div, sub, add, модуль на Assembler будет высчитывать необходимые значения.

**В. *Описание программы***

4 варианта программы для разных типов данных. Программа принимает на вход числа – a, b, c, затем выдаёт результат, высчитанный на С и на Assembler

**Д. Контрольный расчёт**

Вариант программы для 1-битового знакового типа данных. a = 4, b = 5, c = 6

Числитель: 4 + 5\*6 – 1 = 33

Знаменатель: 4/3 + 1 = 2

Результат: 33/2 = 16;

**Д. *Результаты работы программы***

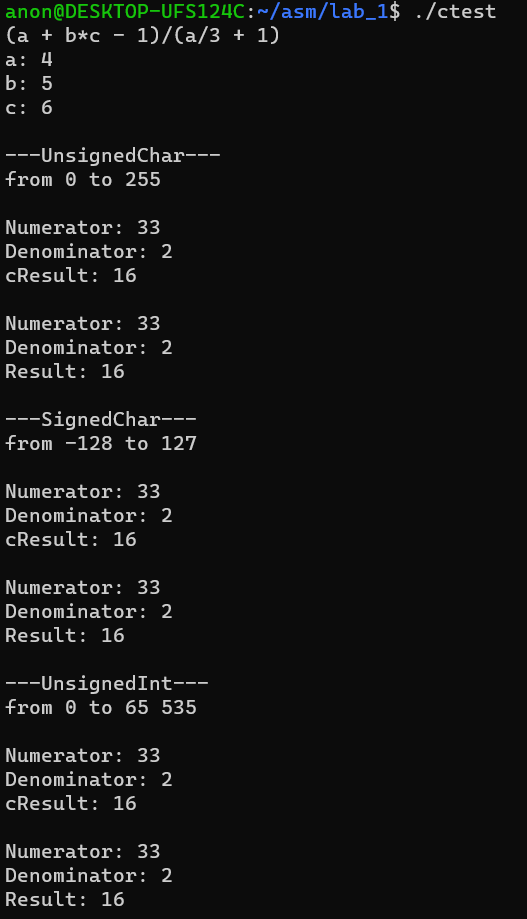


рис 1. Результаты работы программы

**E. Код программы.**

**C**

#include <stdlib.h>  
#include <stdio.h>  
#include <inttypes.h>  
  
extern void unch();   
extern void snch();  
extern void unint();  
extern int16\_t va, vb, vc;  
extern int32\_t res, divd, divr;  
  
int main() {  
 printf("(a + b\*c - 1)/(a/3 + 1)\n");  
 printf("a: ");  
 scanf("%hd", &va);  
 printf("b: ");  
 scanf("%hd", &vb);  
 printf("c: ");  
 scanf("%hd", &vc);  
 printf("\n");  
  
 //(a + b\*c - 1)/(a/3 + 1)  
  
 unch();  
 uint8\_t a = va, b = vb, c = vc;  
 printf("---UnsignedChar---\n");  
 printf("from 0 to 255\n\n");  
 if ((uint16\_t)(a/3 + 1) != 0)  
 {  
 printf("Numerator: %hu\n", (uint16\_t)(a + b\*c - 1));  
 printf("Denominator: %hu\n", (uint16\_t)(a/3 + 1));  
 printf("cResult: %hu\n\n", (uint16\_t)(a + b\*c - 1) / (uint16\_t)(a/3 + 1));  
 }  
 else  
 printf("cResult: division by zero\n\n");  
 printf("Numerator: %hu\n", divd);  
 printf("Denominator: %hu\n", divr);  
 printf("Result: %hu\n\n", res);  
  
  
 snch();  
 printf("---SignedChar---\n");  
 printf("from -128 to 127\n\n");  
 if ((int16\_t)(va/3+1) != 0)  
 {  
 printf("Numerator: %hd\n", (int16\_t)(va + vb\*vc - 1));  
 printf("Denominator: %hd\n", (int16\_t)(va/3 + 1));  
 printf("cResult: %hd\n\n", (int16\_t)(va + vb\*vc - 1) / (int16\_t)(va/3 + 1));  
 }  
 else  
 printf("cResult: division by zero\n\n");  
 printf("Numerator: %hd\n", divd);  
 printf("Denominator: %hd\n", divr);  
 printf("Result: %hd\n\n", res);  
  
  
 unint();  
 uint16\_t wa = va, wb = vb, wc = vc;  
 printf("---UnsignedInt---\n");  
 printf("from 0 to 65 535\n\n");  
 if ((uint32\_t)(wa/3 + 1) != 0)  
 {  
 printf("Numerator: %u\n", (uint32\_t)(wa + wb\*wc - 1));  
 printf("Denominator: %u\n", (uint32\_t)(wa/3 + 1));  
 printf("cResult: %u\n\n", (uint32\_t)(wa + wb\*wc - 1) / (uint32\_t)(wa/3 + 1));  
 }  
 else  
 printf("cResult: division by zero\n\n");  
 printf("Numerator: %u\n", divr);  
 printf("Denominator: %u\n", divd);  
 printf("Result: %u\n\n", res);  
}

**Asm**

global unint  
global unch  
global snch  
global res  
global divd  
global divr  
global va  
global vb  
global vc  
  
  
section .bss  
  
 res resb 4  
 divd resb 4  
 divr resb 4  
 va resb 2  
 vb resb 2  
 vc resb 2  
  
section .text  
;(b+a)/(a-1) , если a > b  
;23 , a = b  
;b\*b/8 , если a < b  
  
snch:  
 xor eax, eax  
 xor ebx, ebx  
 xor ecx, ecx  
 xor edx, edx  
 mov al, [vc]  
 mov ah, [vb]  
 imul ah  
 cwd  
 mov bx, ax  
 mov ax, [va]  
 add bx, ax  
 sub bx, 1  
 mov [divd], bx  
 mov ax, [va]  
 mov cl, 3  
 idiv cl  
 cbw  
 mov bx, ax  
 mov ax, 1  
 add bx, ax  
 mov [divr], bx  
 jz divz  
 mov ax, [divd]  
 cwd  
 idiv bx  
 mov [res], ax  
 ret  
  
unch:  
 xor eax, eax  
 xor ebx, ebx  
 xor ecx, ecx  
 xor edx, edx  
 mov al, [vc]  
 mov ah, [vb]  
 mul ah  
 mov bx, ax  
 mov al, [va]  
 xor ah, ah  
 add bx, ax  
 sub bx, 1  
 mov [divd], bx  
 mov al, [va]  
 xor ah, ah  
 mov cl, 3  
 div cl  
 xor ah, ah  
 mov bx, ax  
 add bx, 1  
 mov [divr], bx  
 jz divz  
 mov ax, [divd]  
 cwd  
 idiv bx  
 mov [res], ax  
 ret  
  
unint:  
 xor eax, eax  
 xor ebx, ebx  
 xor ecx, ecx  
 xor edx, edx  
 mov ax, [vb]  
 xor edx, edx  
 mov bx, [vc]  
 mul ebx  
 mov bx, [va]  
 add eax, ebx  
 dec eax  
 mov [divr], eax  
 xor eax, eax  
 mov ax, [va]  
 mov bx, 3  
 ;xor edx, edx  
 div bx  
 inc eax  
 mov [divd], eax  
 test eax, eax  
 jz divz  
 mov ecx, eax  
 mov eax, [divr]  
 cdq  
 idiv ecx  
 mov [res], eax  
 ret  
  
divz:  
 mov dword [res], 0  
 ret