Федеральное государственное бюджетное образовательное учреждение высшего образования «Санкт-Петербургский государственный университет телекоммуникаций им. проф. М.А. Бонч-Бруевича»

Кафедра «Программная инженерия и вычислительная техника»

«Машино-зависимые языки программирования»

Отчет

по лабораторной работе №1

«Вычисление целочисленных арифметических выражений»

|  |  |
| --- | --- |
|  | Выполнил |
|  | студент группы ИКПИ-14 |
|  | А.В.Крылов |
|  |  |
|  | Проверил |
|  | Ст. преподаватель |
|  | А.О.Овчинников |
|  |  |

2022 г.

1. Задание

Вычислить заданное целочисленное выражение для исходных данных в знаковых и беззнаковых форматах длиной 8 и 16 бит: signed char, unsigned char и signed int , unsigned int , используя арифметические операции ADD, ADC, INC, SUB, SBB , DEC, NEG, MUL, IMUL, DIV, IDIV , CBW, CWD. Исходные значения переменных вводятся пользователем с клавиатуры. Они должны быть максимально приближены к максимально-возможным для тех типов данных, с которыми решается задача. При вводе данных рекомендуется вывести диапазон возможных значений. Размер и тип числителя, знаменателя и результата зависит от заданного выражения.

Вариант №11

(2 \* d — 96 / a) / (34 / b — a + 1)

2. Текст программы

2.1. Модуль main.c

#include <stdio.h>

#include "funcs.h"

int main() {

int n = 0;

puts(

"Zadacha №11, Krylov Artem Vyacheslavovich, IKPI-14\n"

"(2 \* d - 96 / a) / (34 / b - a + 1)");

void (\*arr[])() = {f\_ch, f\_uch, f\_shrt, f\_ushrt};

do {

puts(

"Viberite tip dannih:\n"

" 1) Signed char\n"

" 2) Unsigned char\n"

" 3) Short\n"

" 4) Unsigned short\n"

"Dlya zaversheniya raboti vvetide 0");

scanf("%d", &n);

if (n) arr[n - 1]();

puts("");

} while (n);

return 0;

}

2.2. Модуль funcs.h

#pragma once

void f\_ch();

void f\_uch();

void f\_shrt();

void f\_ushrt();

2.3. Модуль funcs.c

#include "funcs.h"

#include <stdio.h>

// (2 \* d - 96 / a) / (34 / b - a + 1)

// numerator, denominator, res

signed char num\_ch, den\_ch, res\_ch, a\_ch, b\_ch, d\_ch;

unsigned char num\_uch, den\_uch, res\_uch, a\_uch, b\_uch, d\_uch;

short num\_shrt, den\_shrt, res\_shrt, a\_shrt, b\_shrt, d\_shrt;

unsigned short num\_ushrt, den\_ushrt, res\_ushrt, a\_ushrt, b\_ushrt, d\_ushrt;

extern void f\_ch\_asm();

extern void f\_uch\_asm();

extern void f\_shrt\_asm();

extern void f\_ushrt\_asm();

void f\_ch() {

puts("Vvedite zhacheniya a, b i d: [-128; 127]");

scanf("%hhd%hhd%hhd", &a\_ch, &b\_ch, &d\_ch);

puts("S pomoshyu ASM:");

f\_ch\_asm();

printf("Num = %hhd, den = %hhd, res = %hhd\n", num\_ch, den\_ch, res\_ch);

puts("S pomoshyu C:");

num\_ch = 2 \* d\_ch - 96 / a\_ch;

den\_ch = 34 / b\_ch - a\_ch + 1;

res\_ch = num\_ch / den\_ch;

printf("Num = %hhd, den = %hhd, res = %hhd\n", num\_ch, den\_ch, res\_ch);

}

void f\_uch() {

puts("Vvedite zhacheniya a, b i d: [0; 255]");

scanf("%hhu%hhu%hhu", &a\_uch, &b\_uch, &d\_uch);

puts("S pomoshyu ASM:");

f\_uch\_asm();

printf("Num = %hhu, den = %hhu, res = %hhd\n", num\_uch, den\_uch, res\_uch);

puts("S pomoshyu C:");

num\_uch = 2 \* d\_uch - 96 / a\_uch;

den\_uch = 34 / b\_uch - a\_uch + 1;

res\_uch = num\_uch / den\_uch;

printf("Num = %hhu, den = %hhu, res = %hhu\n", num\_uch, den\_uch, res\_uch);

}

void f\_shrt() {

puts("Vvedite zhacheniya a, b i d: [-32,768; 32,767]");

scanf("%hi%hi%hi", &a\_shrt, &b\_shrt, &d\_shrt);

puts("S pomoshyu ASM:");

f\_shrt\_asm();

printf("Num = %hi, den = %hi, res = %hi\n", num\_shrt, den\_shrt, res\_shrt);

puts("S pomoshyu C:");

num\_shrt = 2 \* d\_shrt - 96 / a\_shrt;

den\_shrt = 34 / b\_shrt - a\_shrt + 1;

res\_shrt = num\_shrt / den\_shrt;

printf("Num = %hi, den = %hi, res = %hi\n", num\_shrt, den\_shrt, res\_shrt);

}

void f\_ushrt() {

puts("Vvedite zhacheniya a, b i d: [0; 65,535]");

scanf("%hu%hu%hu", &a\_ushrt, &b\_ushrt, &d\_ushrt);

puts("S pomoshyu ASM:");

f\_ushrt\_asm();

printf("Num = %hu, den = %hu, res = %hu\n", num\_ushrt, den\_ushrt, res\_ushrt);

puts("S pomoshyu C:");

num\_ushrt = 2 \* d\_ushrt - 96 / a\_ushrt;

den\_ushrt = 34 / b\_ushrt - a\_ushrt + 1;

res\_ushrt = num\_ushrt / den\_ushrt;

printf("Num = %hu, den = %hu, res = %hu\n", num\_ushrt, den\_ushrt, res\_ushrt);

}

2.4. Модуль afuncs.asm

section .data

extern num\_ch, den\_ch, res\_ch, a\_ch, b\_ch, d\_ch

extern num\_uch, den\_uch, res\_uch, a\_uch, b\_uch, d\_uch

extern num\_shrt, den\_shrt, res\_shrt, a\_shrt, b\_shrt, d\_shrt

extern num\_ushrt, den\_ushrt, res\_ushrt, a\_ushrt, b\_ushrt, d\_ushrt

global f\_ch\_asm

global f\_uch\_asm

global f\_shrt\_asm

global f\_ushrt\_asm

section .text

f\_ch\_asm:

mov al, 2

imul byte [d\_ch] ; 2 \* d

mov bl, al

mov al, 96

cbw

idiv byte [a\_ch] ; 96 / a

sub bl, al ; 2 \* d - 96 / a

mov [num\_ch], bl

mov al, 34

cbw

idiv byte [b\_ch] ; 34 / b

sub al, [a\_ch] ; - a

inc al

mov [den\_ch], al

mov al, [num\_ch]

cbw

idiv byte [den\_ch]

mov [res\_ch], al

ret

f\_uch\_asm:

mov al, 2

mul byte [d\_uch] ; 2 \* d

mov bl, al

mov al, 96

cbw

div byte [a\_uch] ; 96 / a

sub bl, al ; 2 \* d - 96 / a

mov [num\_uch], bl

mov al, 34

cbw

div byte [b\_uch] ; 34 / b

sub al, [a\_uch] ; - a

inc al

mov [den\_uch], al

mov al, [num\_uch]

xor ah, ah

div byte [den\_uch]

mov [res\_uch], al

ret

f\_shrt\_asm:

mov ax, 2

imul word [d\_shrt] ; 2 \* d

mov bx, ax

mov ax, 96

cwd

idiv word [a\_shrt] ; 96 / a

sub bx, ax ; 2 \* d - 96 / a

mov [num\_shrt], bx

mov ax, 34

cwd

idiv word [b\_shrt] ; 34 / b

sub ax, [a\_shrt] ; - a

inc ax

mov [den\_shrt], ax

mov ax, [num\_shrt]

cwd

idiv word [den\_shrt]

mov [res\_shrt], ax

ret

f\_ushrt\_asm:

mov ax, 2

mul word [d\_ushrt] ; 2 \* d

mov bx, ax

mov ax, 96

cwd

div word [a\_ushrt] ; 96 / a

sub bx, ax ; 2 \* d - 96 / a

mov [num\_ushrt], bx

mov ax, 34

cwd

div word [b\_ushrt] ; 34 / b

sub ax, [a\_ushrt] ; - a

inc ax

mov [den\_ushrt], ax

mov ax, [num\_ushrt]

xor dx, dx

div word [den\_ushrt]

mov [res\_ushrt], ax

ret

2.5. Модуль Makefile

# common

# P.S. turning on optimizations messes up the output

C = gcc -Wall -Wextra -Wpedantic -fsanitize=address,undefined

# compile-only args

CO = -c

S = src/

O = obj/

OBJS = $(O)main.o $(O)funcs.o $(O)afuncs.o

all: $(OBJS)

$(C) $(OBJS) -fno-pie -no-pie -o L1

$(O)main.o: $(S)main.c

$(C) $(CO) $(S)main.c -o $(O)main.o

$(O)funcs.o: $(S)funcs.c

$(C) $(CO) $(S)funcs.c -o $(O)funcs.o

$(O)afuncs.o: $(S)afuncs.asm

yasm -f elf64 $(S)afuncs.asm -o $(O)afuncs.o

clean:

rm $(O)\*.o && rm L1

3. Сборка проекта

make

4. Выполнение программы

4.1. Запуск программы

./L1

4.2. Входные данные

3

-275 33 5928

4.3. Ожидаемый результат выполнения

S pomoshyu ASM:

Num = 11856, den = 277, res = 42

S pomoshyu C:

Num = 11856, den = 277, res = 42

4.4. Результат выполнения

S pomoshyu ASM:

Num = 11856, den = 277, res = 42

S pomoshyu C:

Num = 11856, den = 277, res = 42

5. Вывод

Результат выполнения программы соответствует ожидаемому результату. Работа выполнена в полном объеме.