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 №2

«Организация условных переходов»

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

2022 г.

1. Задание

Вычислить заданное условное целочисленное выражение для данных в форматах INTEGER (int), WORD (unsigned int), используя команды сравнения, условного и безусловного переходов. Результат X тоже целочисленный и его диапазон (формат) зависит от специфики решаемого условного выражения. Исходные данные должны вводиться корректно(с проверкой на область допустимых значений). Результат также должен быть проверен на область допустимых значений. Обмен данными между Си и ASM - модулем должен осуществляться через глобальные переменные, определённые в модуле Си.

Вариант №11

(b + a) / (a - 1), если a > b

23, если a = b

b \* b / 8, если a < b

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"

"(b + a) / (a - 1), if a > b\n"

"23, if a = b\n"

"b \* b / 8, if a < b\n");

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

do {

puts(

"Viberite tip dannih:\n"

" 1) Signed short\n"

" 2) 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\_shrt();

void f\_ushrt();

2.3. Модуль funcs.c

#include "funcs.h"

#include <stdio.h>

short a\_shrt, b\_shrt;

unsigned short a\_ushrt, b\_ushrt;

extern short f\_shrt\_asm();

extern unsigned short f\_ushrt\_asm();

void f\_shrt() {

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

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

if (a\_shrt < b\_shrt) {

if (b\_shrt < -181 || b\_shrt > 181) {

puts("Ne podhodyasheye zhacneniye b");

return;

}

} else if (a\_shrt > b\_shrt)

if (a\_shrt + b\_shrt > 32767 || a\_shrt + b\_shrt < -32768) {

puts("Ne podhodyashiye zhacneniya a i b");

return;

}

printf("ASM: %hi\n", f\_shrt\_asm());

short res;

if (a\_shrt > b\_shrt)

res = (b\_shrt + a\_shrt) / (a\_shrt - 1);

else if (a\_shrt == b\_shrt)

res = 23;

else if (a\_shrt < b\_shrt)

res = b\_shrt \* b\_shrt / 8;

printf("C: %hi\n", res);

}

void f\_ushrt() {

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

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

if (a\_ushrt < b\_ushrt) {

if (b\_ushrt > 254) {

puts("Ne podhodyasheye zhacneniye b");

return;

}

} else if (a\_ushrt > b\_ushrt)

if (a\_ushrt + b\_ushrt > 65535) {

puts("Ne podhodyashiye zhacneniya a i b");

return;

}

printf("ASM: %hu\n", f\_ushrt\_asm());

unsigned short res;

if (a\_ushrt > b\_ushrt)

res = (b\_ushrt + a\_ushrt) / (a\_ushrt - 1);

else if (a\_ushrt == b\_ushrt)

res = 23;

else if (a\_ushrt < b\_ushrt)

res = b\_ushrt \* b\_ushrt / 8;

printf("C: %hu\n", res);

}

2.4. Модуль afuncs.asm

section .data

extern a\_shrt, b\_shrt

extern a\_ushrt, b\_ushrt

global f\_shrt\_asm, f\_ushrt\_asm

; sh/ush + gr/eq/le/fin

section .text

f\_shrt\_asm:

mov bx, [a\_shrt]

cmp bx, [b\_shrt]

jg shg

je she

; a < b

mov ax, [b\_shrt]

mov bx, [b\_shrt]

imul bx ; b \* b

mov bx, 8

cwd

idiv bx ; b \* b / 8

jmp shf

shg: ; a > b

mov ax, [a\_shrt]

add ax, [b\_shrt] ; b + a

mov bx, [a\_shrt]

dec bx ; a - 1

cwd

idiv bx ; (b + a) / (a - 1)

jmp shf

she: ; a = b

mov ax, 23

shf:

ret

f\_ushrt\_asm:

xor eax, eax

xor ebx, ebx

mov ax, [a\_ushrt]

mov bx, [b\_ushrt]

cmp eax, ebx

jg ushg

je ushe

; a < b

mov ax, [b\_ushrt]

mul bx ; b \* b

mov bx, 8

xor dx, dx

div bx ; b \* b / 8

jmp ushf

ushg: ; a > b

add eax, ebx ; b + a

mov bx, [a\_ushrt]

dec bx ; a - 1

xor dx, dx

div bx ; (b + a) / (a - 1)

jmp ushf

ushe: ; a = b

mov ax, 23

ushf:

ret

2.5. Модуль Makefile

# common

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 L2

$(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 L2

3. Сборка проекта

make

4. Выполнение программы

4.1. Запуск программы

./L2

4.2. Входные данные

1

-55 19

4.3. Ожидаемый результат выполнения

ASM: 45

C: 45

4.4. Результат выполнения

ASM: 45

C: 45

5. Вывод

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