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 №3

«Ввод-вывод на Ассемблере»

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

2022 г.

1. Задание

Реализовать целочисленные вычисления, сделанные в лабораторной работе №1 или №2(по указанию преподавателя), полностью используя язык Ассемблера для организации корректного ввода-вывода информации:

• исходные данные должны вводиться с проверкой правильности вводимых символов;

• входные данные и результат должны быть проверены на область допустимых значений;

• при наличии ошибки должно быть выдано соответствующее сообщение.

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.asm

global main

extern puts, scanf

extern f\_shrt, f\_ushrt

section .data

intro db "Zadacha №11, Krylov Artem Vyacheslavovich, IKPI-14", 10,\

"(b + a) / (a - 1), if a > b", 10,\

"23, if a = b", 10,\

"b \* b / 8, if a < b", 10, 0

menu db "Viberite tip dannih:", 10,\

" 1) Signed short", 10,\

" 2) Unsigned short", 10,\

"Dlya zaversheniya raboti vvetide 0", 0

format1 db "%d", 0

format2 db "", 0

section .bss

n dd 0

section .text

mainarr:

dq f\_shrt

dq f\_ushrt

main:

push rbp

mov rbp, rsp

mov edi, intro

call puts

mainloop:

mov edi, menu

call puts

mov edi, format1

mov esi, n

call scanf

mov eax, [n]

cmp eax, 0

je fin

dec eax

call [mainarr + 8 \* eax]

mov edi, format2

call puts

jmp mainloop

fin:

xor eax, eax

mov rsp, rbp

pop rbp

ret

2.2. Модуль funcs.asm

global f\_shrt, f\_ushrt

global a\_shrt, b\_shrt

global a\_ushrt, b\_ushrt

extern puts, scanf, printf

extern f\_shrt\_asm, f\_ushrt\_asm

section .data

sh\_range db "Vvedite zhacheniya a, b: [-32,768; 32,767]", 0

ush\_range db "Vvedite zhacheniya a, b: [0; 65,535]", 0

sh\_form1 db "%hi%hi", 0

ush\_form1 db "%hu%hu", 0

sh\_form2 db "ASM: %hi", 10, 0

ush\_form2 db "ASM: %hu", 10, 0

err1 db "Ne podhodyasheye zhacneniye b", 0

err2 db "Ne podhodyashiye zhacneniya a i b", 0

section .bss

a\_shrt:

dw 0

b\_shrt:

dw 0

a\_ushrt:

dw 0

b\_ushrt:

dw 0

section .text

f\_shrt:

push rbp

mov rbp, rsp

mov edi, sh\_range

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

mov edi, sh\_form1

mov esi, a\_shrt

mov edx, b\_shrt

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

mov ax, [a\_shrt]

mov bx, [b\_shrt]

cmp ax, bx

jl sh\_branch1 ; if (a\_shrt < b\_shrt)

cmp ax, bx

jg sh\_branch2 ; else if (a\_shrt > b\_shrt)

sh\_res:

call f\_shrt\_asm

mov edi, sh\_form2

mov esi, eax

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

sh\_fin:

mov rsp, rbp

pop rbp

ret

sh\_branch1:

cmp bx, -181

jl error1

cmp bx, 181

jg error1

jmp sh\_res

sh\_branch2:

add ax, bx

jno sh\_res

jmp error2

f\_ushrt:

push rbp

mov rbp, rsp

mov edi, ush\_range

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

mov edi, ush\_form1

mov esi, a\_ushrt

mov edx, b\_ushrt

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

xor eax, eax

xor ebx, ebx

mov ax, [a\_ushrt]

mov bx, [b\_ushrt]

cmp eax, ebx

jl ush\_branch1 ; if (a\_ushrt < b\_ushrt)

cmp eax, ebx

jg ush\_branch2 ; else if (a\_ushrt > b\_ushrt)

ush\_res:

call f\_ushrt\_asm

mov edi, ush\_form2

mov esi, eax

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

mov rsp, rbp

pop rbp

ret

ush\_branch1:

cmp bx, 254

jg error1

jmp ush\_res

ush\_branch2:

add eax, ebx

cmp eax, 65535

jg error2

jmp ush\_res

error1:

mov edi, err1

call puts

jmp sh\_fin

error2:

mov edi, err2

call puts

jmp sh\_fin

2.3. 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

2.4. Модуль Makefile

# common

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

A = yasm -f elf64

# 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 L3

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

$(A) $(S)main.asm -o $(O)main.o

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

$(A) $(S)funcs.asm -o $(O)funcs.o

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

$(A) $(S)afuncs.asm -o $(O)afuncs.o

clean:

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

3. Сборка проекта

make

4. Выполнение программы

4.1. Запуск программы

./L3

4.2. Входные данные

1

-55 19

4.3. Ожидаемый результат выполнения

ASM: 45

4.4. Результат выполнения

ASM: 45

5. Вывод

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