**“ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА”**

**(СПбГУТ)**

**Факультет инфокоммуникационных Сетей и систем (иксс)**

**кафедра программной инженерии и вычислительной техники (пи и вт)**

Дисциплина: «Машинно-зависимые языки программирования»

Лабораторная работа №3.

**Тема: «ввод-вывод на ассемблере (linux)»**

**вариант №17**

Выполнил: Студент группы КПИ-95

Новиков С.С.

Подпись\_\_\_\_\_\_\_\_\_\_\_\_

Принял: Коробов С.А.

Подпись\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_\_»\_\_\_\_\_\_\_\_ 2020

2020г

Реализовать целочисленные вычисления, сделанные в лабораторной работе **№1** или **№2** (по указанию преподавателя), полностью используя язык Ассемблера для организации корректного ввода-вывода информациию. Реализация задачи должна быть выполнена для **Unix**.

Задача взята со 2ой лабораторной работы

**Код:**

**Файл ASM Linux (NASM):**

; Assemble: nasm -f elf64 -l main\_asm.lst main\_asm.asm

; Link: gcc -m64 -o main\_asm main\_asm.o

; Run: ./main\_asm

; Реализовать целочисленные вычисления, сделанные в лабораторной работе №2,

; полностью используя язык Ассемблера для организации корректного ввода-вывода информации:

; Я взял 2 лабу для основы

extern printf

extern scanf

extern exit

extern a\_sig

extern b\_sig

extern a\_unsig

extern b\_unsig

extern x\_return

global main

segment .data

mode\_1\_str: db "1 signed",0Dh,0Ah,0

mode\_2\_str: db "2 unsigned",0Dh,0Ah,0

mode\_3\_str: db "3 exit",0Dh,0Ah,0

enter\_mode\_str: db "Enter node: ",0Dh,0Ah,0

mode\_scanf\_str: db "%d",0

mode: dq 0

enter\_a\_str: db "a: ", 0

enter\_b\_str: db "b: ", 0

enter\_sword: db "%hd",0

enter\_uword: db "%hu",0

print\_sword: db "x: %hd",10,0

a\_sig: dw 0

b\_sig: dw 0

a\_unsig: dw 0

b\_unsig: dw 0

x\_return: dw 0

segment .text

main:

sub rsp, 28h ; выделяем память в стеке

@start:

sub rsp, 8 ; теневая область для printf

;1 Знаковый\n

mov rdi, mode\_1\_str

mov rax, 0

call printf

;2 беззнаковый\n

mov rdi, mode\_2\_str

mov rax, 0

call printf

;3 выход\n

mov rdi, mode\_3\_str

mov rax, 0

call printf

;Enter mode

mov rdi, enter\_mode\_str

mov rax, 0

call printf

add rsp, 8 ;очвобождение теневой области область для printf

;Enter

sub rsp, 8\*2

mov rdi, mode\_scanf\_str

mov rsi, mode

mov rax, 0

call scanf

add rsp, 8\*2

mov rax, 1

cmp rax, [mode]

je @m\_1

mov rax, 2

cmp rax, [mode]

je @m\_2

mov rax, 3

cmp rax, [mode]

je @next

jmp @start

@m\_1:

sub rsp, 8\*2 ;теневая область для printf

;a:

mov rdi, enter\_a\_str

call printf

;enter a

mov rdi, enter\_sword

mov rsi, a\_sig

call scanf

;b:

mov rdi, enter\_b\_str

call printf

;enter b

mov rdi, enter\_sword

mov rsi, b\_sig

call scanf

call signedF

;print x

mov rdi, print\_sword

mov si, [x\_return]

call printf

add rsp, 8\*2

jmp @start

@m\_2:

sub rsp, 8\*2 ;теневая область для printf

;a:

mov rdi, enter\_a\_str

call printf

;enter a

mov rdi, enter\_uword

mov rsi, a\_unsig

call scanf

;b:

mov rdi, enter\_b\_str

call printf

;enter b

mov rdi, enter\_uword

mov rsi, b\_unsig

call scanf

call unsignedF

;print x

mov rdi, print\_sword

mov si, [x\_return]

call printf

add rsp, 8\*2

jmp @start

@next:

mov esi, 0

call exit

ret

; signed

;;;;;;;;;;;;;;;;;;;;;;;;

;a == b

signed\_s1:

mov ax, -1d

mov [x\_return], ax

ret

;a > b

; a\*b-1 если a > b

signed\_s2:

mov bx, [b\_sig]

mov ax, [a\_sig]

imul bx

sub ax, 1d

mov [x\_return], ax

ret

;a < b

; a/b+2 , если a < b

signed\_s3:

mov ax, [a\_sig]

mov bx, [b\_sig]

;cwd ; преобразовать слово, находящееся в регистре АХ, в двойное слово — регистры < DX:AX >).

mov [x\_return], ax

idiv bx

mov dx, 2d

add ax, dx

mov [x\_return], ax

ret

signedF:

mov ax, [a\_sig]

mov bx, [b\_sig]

cmp ax, bx

je signed\_s1 ;ax==bx

jg signed\_s2 ;ax>bx

jl signed\_s3 ;ax<bx

; unsigned

;;;;;;;;;;;;;;;;;;;;;;;;

;a == b

unsigned\_s1:

mov ax, -1d

mov [x\_return], ax

ret

;a > b

; a\*b-1 если a > b

unsigned\_s2:

mov bx, [b\_unsig]

mov ax, [a\_unsig]

mul bx

sub ax, 1d

mov [x\_return], ax

ret

;a < b

; a/b+2 , если a < b

unsigned\_s3:

mov ax, [a\_unsig]

mov bx, [b\_unsig]

div bx

mov dx, 2d

add ax, dx

mov [x\_return], ax

ret

unsignedF:

mov ax, [a\_unsig]

mov bx, [b\_unsig]

cmp ax, bx

je unsigned\_s1 ; ax==bx

jg unsigned\_s2 ; ax>bx

jl unsigned\_s3 ; ax<bx