**“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»**

**вариант №1**

Выполнил: Студент группы КПИ-95

Алюшин В.В.

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 (ASM):**

option casemap:none

includelib legacy\_stdio\_definitions.lib

extern MessageBoxA: PROC

extern ExitProcess: PROC

extern printf: proc

extern scanf:proc

extern \_getch: proc

.DATA

mode\_1\_str db "1 signed",10,0

mode\_2\_str db "2 unsigned",10,0

mode\_3\_str db "3 exit",10,0

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

mode\_scanf\_str db "%i"

mode dword 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 word 0

b\_sig word 0

a\_unsig word 0

b\_unsig word 0

x\_return word 0

.CODE

;mainCRTStartup

main PROC

sub rsp, 28h

@start:

sub rsp, 8

;1 Знаковый\n

lea rcx, mode\_1\_str

call printf

;2 беззнаковый\n

lea rcx, mode\_2\_str

call printf

;3 выход\n

lea rcx, mode\_3\_str

call printf

;Enter mode

lea rcx, enter\_mode\_str

call printf

add rsp, 8

;Enter

sub rsp, 8\*2

lea rcx, mode\_scanf\_str

lea rdx, mode

call scanf

add rsp, 8\*2

cmp mode, 1

je @m\_1

cmp mode, 2

je @m\_2

cmp mode, 3

je @next

jmp @start

@m\_1:

sub rsp, 8\*2

;a:

lea rcx, enter\_a\_str

call printf

;enter a

lea rcx, enter\_sword

lea rdx, a\_sig

call scanf

;b:

lea rcx, enter\_b\_str

call printf

;enter b

lea rcx, enter\_sword

lea rdx, b\_sig

call scanf

call signedF

;print x

lea rcx, print\_sword

mov dx, x\_return

call printf

add rsp, 8\*2

jmp @start

@m\_2:

sub rsp, 8\*2

;a:

lea rcx, enter\_a\_str

call printf

;enter a

lea rcx, enter\_uword

lea rdx, a\_unsig

call scanf

;b:

lea rcx, enter\_b\_str

call printf

;enter b

lea rcx, enter\_uword

lea rdx, b\_unsig

call scanf

call unsignedF

;print x

lea rcx, print\_sword

mov dx, x\_return

call printf

add rsp, 8\*2

jmp @start

@next:

;call \_getch

mov ecx, 0 ; ExitCode ; Функция ExitProcess ожидает 32-битный параметр

;add rsp, 28h

call ExitProcess

ret

;add rsp, 28h

main ENDP

; signed

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

;a == b

signed\_s1 PROC

mov ax, 21d

mov x\_return, ax

ret

signed\_s1 ENDP

;a > b

signed\_s2 PROC

mov ax, a\_sig

add ax, 2

cwd

idiv b\_sig

mov x\_return, ax

ret

signed\_s2 ENDP

;a < b

signed\_s3 PROC

mov ax, a\_sig

cwd

imul b\_sig

mov bx, 4d

idiv bx

mov x\_return, ax

ret

signed\_s3 ENDP

signedF PROC

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

signedF ENDP

; unsigned

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

;a == b

unsigned\_s1 PROC

mov ax, 21d

mov x\_return, ax

ret

unsigned\_s1 ENDP

;a > b

unsigned\_s2 PROC

mov ax, a\_unsig

add ax, 2

xor dx, dx

idiv b\_unsig

mov x\_return, ax

ret

unsigned\_s2 ENDP

;a < b

unsigned\_s3 PROC

mov ax, a\_unsig

xor dx, dx

imul b\_unsig

mov bx, 4d

idiv bx

mov x\_return, ax

ret

unsigned\_s3 ENDP

unsignedF PROC

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

unsignedF ENDP

END