Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №1
за IV семестр
по дисциплине: "Компьютерные системы и сети."
Тема: "Введение в ассемблер."

Выполнил:
студент 2 курса
Корнасевич И. Д.

Проверил:
Савицкий Ю. В.

Брест 2021

Написаит программу на ассемблере, вывудящую фамилию и имя.

```
section .bss
buffer resb 30

section .data new_line db 0xa surname        db "Ivan Karnasevich", 0        ;just my
↪  name surname_len

section .text
global _start

_start:
        push qword 10
        push buffer
        call read                        ; reads value from keyboard

        push rax
        call atoi                        ; parse the value to long

        cmp rax, [surname_len]
        jl .rax                                ; if rax < *surname_len
        jmp .len                        ;          jmp .rax
                                        ; else
.rax:                                        ;        jmp .len
        push rax
        jmp .print                        ; pushs rax as argument of println()

.len:
        push qword [surname_len]        ; pushs *surname_len as argument of printline()
        jmp .print

.print:
        push surname
        call println

        mov rax, 0x3c
        mov rdi, 0x0
        syscall

;-----------------------------------------------
; int atoi(string* str_to_parse)
atoi:
        push rbp
        mov rbp, rsp
        mov rdi, [rbp + 16]
        xor rax, rax            ; Set initial total to 0

.convert:
        movzx rsi, byte [rdi]   ; Get the current character
        test rsi, rsi                ; Check for \0
        je .done

        cmp rsi, 48             ; Anything less than 0 is invalid
        jl .error

        cmp rsi, 57             ; Anything greater than 9 is invalid
```

```
        jg .error

        sub rsi, 48             ; Convert from ASCII to decimal
        imul rax, 10            ; Multiply total by 10
        add rax, rsi            ; Add current digit to total

        inc rdi                 ; Get the address of the next character
        jmp .convert

.error:
        mov rax, -1             ; Return -1 on error

.done:
        pop rbp
            ret 8                   ; Return total or error code
;-------------------------------------------------
; void print(char* str, int len)
print:
        push rbp
        mov rbp, rsp

        mov rax, 0x1                        ; set sysfunction write()
        mov rdi, 0x1                         ; set output stdout
        mov rsi, [rbp + 16]                ; rcx <- str
        mov rdx, [rbp + 24]                ; rdx <-
        ↪  len
        syscall

        pop rbp
        ret 16

;-------------------------------------------------
; void println(char* str, int len)
println:
        push rbp
        mov rbp, rsp

        push qword [rbp + 24]
        push qword [rbp + 16]
        call print                         ; call print function to print data

        mov rax, 0x1                         ; set sysfunction write()
        mov rdi, 0x1                         ; set output stdout
        mov rsi, new_line                 ; "\n" -> rcx
        mov rdx, 0x1                         ; 1 -> rdx as size of "\n"
        syscall

        pop rbp
        ret 16
;-------------------------------------------------
; char* read(char* buffer, long size)
read:
        push rbp
        mov rbp, rsp

        xor rax, rax                        ; set sysfunction read()
```

```
        xor rdi, rdi                        ; set stdin as input
        mov rsi, [rbp + 16]                ; set a  buffet to write in
        mov rdx, [rbp + 24]                ; set size of the buffer
        syscall

        mov rcx, rdx                        ; rcx <- total size of
        ↪  buffer
.loop1:                                        ; finds last '\n'
        cmp [rsi + rcx], byte 0xa        ;
        je .done                           ; if rsi[rcx] == '\n' break
        loop .loop1                           ; while true
.done:
        mov [rsi + rcx], byte 0                ; replaces the "\n" by "\0"
        mov rax, rsi                        ; moves result to rax

        pop rbp
        ret 16

;-------------------------------------------------
; long strlen(char* buffer)
strlen:
        push rbp
        mov rbp, rsp
        xor rax, rax                          ; rax <- 0
        movzx esi, byte [rbp + 15]         ; esi <- first byte before buffer
.loop:                                         ; finds index of '\0' in buffer
        inc esi                                ; goto next symbol
        inc rax
        test esi, esi
        je .loop                           ; while esi != 0

        pop rbp
        ret 8
```

Пример выполнения:

```
ivan@pc:~/Labs/assembly/lab1$ ./task
10
Ivan Karna
```

Пример отладки:

```
Breakpoint 1, _start () at task.asm:13
13
(gdb)
(gdb) s
_start () at task.asm:14
14              push rax
(gdb) s
_start () at task.asm:15
15              call atoi                       ; parse the value to long
(gdb) s
read () at task.asm:108
108              xor rax, rax                        ; set sysfunction read()
(gdb) r i
The program being debugged has been started already.
```

```
Start it from the beginning? (y or n) n
Program not restarted.
(gdb) i r
rax            0x0                 0
rbx            0x0                 0
rcx            0x0                 0
rdx            0x0                 0
rsi            0x0                 0
rdi            0x0                 0
rbp            0x0                 0x0
rsp            0x7fffffffdf58      0x7fffffffdf58
r8             0x0                 0
r9             0x0                 0
r10            0x0                 0
r11            0x0                 0
r12            0x0                 0
r13            0x0                 0
r14            0x0                 0
r15            0x0                 0
rip            0x4010c5            0x4010c5 <read>
eflags         0x202               [ IF ]
cs             0x33                51
ss             0x2b                43
ds             0x0                 0
es             0x0                 0
fs             0x0                 0
gs             0x0                 0
(gdb) s
109                xor rdi, rdi                          ; set stdin as input
(gdb) s
111                mov rdx, [rbp + 24]                ; set size of the buffer
(gdb) s
112                syscall
(gdb) s
113
```

Вывод: Разработана простая программа на языке ассемблера и произведена её отладка.