Университет ИТМО

Кафедра ВТ

**Языки системного программирования**

Лабораторная работа №1

Группа P3210

Нгу Фыонг Ань

ПРОВЕРИЛ:

2018 год

1. Содержимое файла lib.inc

section .data

section .text

*Accepts a pointer to a string and returns its length*

**string\_length:**

xor rax, rax

.loop:

cmp byte [rdi+rax], 0

je .end

inc rax

jmp .loop

.end:

ret

; reset rax

; Check if the current symbol is null-terminator

; Jump if we found null-terminator

; Otherwise go to next symbol and increase counter

*Accepts a pointer to a null-terminated string and prints it to stdout*

**print\_string:**

mov rsi, rdi

call string\_length

mov rdx, rax

mov rax, 1

mov rdi, 1

syscall

ret

; address of string to output

; write to rdx the length of the string

; system call for write

; file handle 1 is stdout

; invoke operating system to do the write

*Accepts a character code directly as its first argument and prints it to stdout*

**print\_char:**

push rdi

mov rax, 1

mov rdi, 1

mov rsi, rsp

mov rdx, 1

; push character to the stack (rsp)

; write a character

syscall

pop rdi

ret

; invoke operating system to do the write

; get the top element of the stack and copy into rdi

*Prints a character with code 0xA.(10)*

**print\_newline:**

mov rdi, 10

call print\_char

*Outputs an unsigned 8-byte integer in decimal format*

**print\_uint:**

mov rax, rdi

mov r9, 10

mov r8, rsp

dec rsp

mov byte [rsp], 0

.loop:

xor rdx, rdx

div r9

add rdx, 48

dec rsp

mov [rsp], dl

test rax, rax

jnz .loop

mov rdi, rsp

call print\_string

mov rsp, r8

ret

; Write the divider (10)

; save the top of the stack

; Write the end of line character (0)

; divide the number by 10

; add rdx 48 (code zero)

; dl - lower 8-bit rdx

; set ZF to 1 if rax == 0

; jump to .loop if ZF == 0

; restore the value of the top of the stack

*Output a signed 8-byte integer in decimal format*

**print\_int:**

test rdi, rdi

jns print\_uint

mov rax, rdi

neg rax

; set SF(sign flag) to 1 if rdi < 0

; if positive, run the function for unsigned

;change the sign of the number

push rax

mov rdi, '-'

call print\_char

pop rax

mov rdi, rax

jmp print\_uint

;save rax value

*Accepts two pointers to strings and compares them. Returns 1 if they are equal, otherwise 0.*

**string\_equals:**

xor rcx, rcx

.loop:

mov al, byte[rdi+rcx]

mov dl, byte[rsi+rcx]

cmp al, dl

jne .not\_equal

inc rcx

cmp al, 0

je .equal

jmp .loop

.not\_equal:

mov rax, 0

ret

.equal:

mov rax, 1

ret

;Start picking compare strings

; jump if 2 strings are not equal (ZF=0)

; if equal then continue

; checks if this is the end of the line

*Read one character from stdin and return it. If the end of input stream occurs, return 0.*

**read\_char:**

push 0

mov rax, 0

mov rdi, 0

mov rsi, rsp

; Where we enter the string

mov rdx, 1

syscall

pop rax

ret

; read 1 character

*Accepts a buffer address and size as arguments. Reads next word from stdin (skipping whitespaces7 into buffer). Stops and returns 0 if word is too big for the buffer specified; otherwise returns a buffer address.*

*This function should null-terminate the accepted string.*

**read\_word:**

mov r8, -1

mov rdx, 256

mov rdi, 0

mov rsi, buffer

mov rax, 0

syscall

.check\_loop:

inc r8

cmp byte[buffer + r8], 0x09

jz .check\_loop

cmp byte[buffer + r8], 0x0A

jz .check\_loop

cmp byte[buffer + r8], 0x20

jz .check\_loop

mov r9, r8

cmp byte[buffer + r8], 0

jz .finish

inc r8

.read\_loop:

cmp byte[buffer + r8], 0x20

jz .finish

cmp byte[buffer + r8], 0

jz .finish

; r8- counter

; maximum 255 characters

; handle stdin

; read

; Check the character for a space etc

; Tab Check

; line break check

; check for space (gap)

; If the end of the line, finish

inc r8

jmp .read\_loop

.finish:

mov byte[buffer + r8], 0

lea rax, [buffer + r9]

sub r8, r9

mov rdx, r8

ret

; Write the end of the line

*Accepts a null-terminated string and tries to parse an unsigned number from its start. Returns the number parsed in rax, its characters count in rdx*

**parse\_uint:**

xor rcx,rcx

xor rax, rax

mov r9, 10

xor r8, r8

.check\_loop:

cmp byte[rdi+rcx], '0'

jb .end

cmp byte[rdi+rcx], '9'

ja .end

mul r9

mov r8b, byte[rdi+rcx]

sub r8b, '0'

add rax, r8

inc rcx

jnz .check\_loop

.end:

mov rdx, rcx

ret

;if this is not a number then finish

; jump if below (unsigned comparison)

; perform unsigned multiplication by 10

; put the character in the low byte r8

; get the number

*Accepts a null-terminated string and tries to parse a signed number from its start. Returns the number parsed in rax; its characters count in rdx (including sign if any). No spaces between sign and digits are allowed.*

**parse\_int:**

cmp byte[rdi], '-' ; Check if the first character is a minus

jne parse\_uint ; check for negativity and call parse\_uint if it is positive

inc rdi

push rdi

call parse\_uint

pop rdi

neg rax

inc rdx ; negative + '-'

ret

*Accepts a pointer to a string, a pointer to a buffer, and buffer’s length. Copies string to the destination. The destination address is returned if the string fits the buffer; otherwise zero is returned.*

**string\_copy:**

push rcx

.loop:

xor rcx, rcx ;reset rcx

mov cl, byte [rdi] ; cl-the lower 8 bits of rcx

mov byte[rsi], cl

inc rdi

inc rsi

test rcx, rcx ;check if rcx is 0

jnz .loop ;if rcx is not equal 0 then continue looping

pop rcx

ret

2. Result:



3. Вывод:

In this lab, I learned the basic of assembly and wrote several simple funtions in this language.