

Факультет программной инженерии и компьютерной техники

Низкоуровневое программирование

Лабораторная работа №1

Преподаватель:

Логинов Иван Павлович

Выполнил:

Рождественский Никита Сергеевич

Р33113

Задание: реализовать библиотеку ввода-вывода на ассемблере.

Выполнение:

section .text

write\_to\_stdout:

mov rax, 1

mov rdi, 1

syscall

ret

*; Принимает код возврата и завершает текущий процесс*

exit:

xor rax, rax

mov rax, 60

syscall

*; Принимает указатель на нуль-терминированную строку, возвращает её длину*

string\_length:

xor rax, rax

.loop:

cmp byte[rdi + rax], 0

je .end

inc rax

jmp .loop

.end:

ret

*; Принимает указатель на нуль-терминированную строку, выводит её в stdout*

print\_string:

xor rax, rax

mov rsi, rdi

call string\_length

mov rdx, rax

call write\_to\_stdout

xor rax, rax

ret

*; Принимает код символа и выводит его в stdout*

print\_char:

xor rax, rax

push rdi

mov rsi, rsp

mov rdx, 1

call write\_to\_stdout

add rsp, 8

ret

*; Переводит строку (выводит символ с кодом 0xA)*

print\_newline:

mov rdi, 10

call print\_char

ret

*; Выводит беззнаковое 8-байтовое число в десятичном формате*

*; Совет: выделите место в стеке и храните там результаты деления*

*; Не забудьте перевести цифры в их ASCII коды.*

print\_uint:

push r12

push r13

mov r12, rsp

mov rax, rdi

mov r13,10

dec rsp

mov byte[rsp], 0

.loop:

xor rdx, rdx

dec rsp

div r13

add rdx, 48

mov byte[rsp],dl

test rax,rax

jz .print

jmp .loop

.print:

mov rdi,rsp

call print\_string

mov rsp,r12

pop r13

pop r12

ret

*; Выводит знаковое 8-байтовое число в десятичном формате*

print\_int:

xor rax, rax

mov rax, rdi

test rax, rax

jns .positive

push rax

mov rdi, '-'

call print\_char

pop rax

neg rax

mov rdi, rax

.positive:

call print\_uint

ret

*; Принимает два указателя на нуль-терминированные строки, возвращает 1 если они равны, 0 иначе*

string\_equals:

call string\_length

mov rcx,rax

xchg rdi, rsi

call string\_length

cmp rax,rcx

jne .not\_equals

repe cmpsb

jne .not\_equals

mov rax,1

ret

.not\_equals:

mov rax,0

ret

*; Читает один символ из stdin и возвращает его. Возвращает 0 если достигнут конец потока*

read\_char:

dec rsp

mov rax,0

mov rdi,0

mov rsi,rsp

mov rdx,1

syscall

mov rax, [rsp]

inc rsp

ret

*; Принимает: адрес начала буфера, размер буфера*

*; Читает в буфер слово из stdin, пропуская пробельные символы в начале, .*

*; Пробельные символы это пробел 0x20, табуляция 0x9 и перевод строки 0xA.*

*; Останавливается и возвращает 0 если слово слишком большое для буфера*

*; При успехе возвращает адрес буфера в rax, длину слова в rdx.*

*; При неудаче возвращает 0 в rax*

*; Эта функция должна дописывать к слову нуль-терминатор*

read\_word:

push rbx

mov r8, rsi

mov r9, rdi

xor r10, r10

push rdi

xor rdi, rdi

mov rdx, 1

.next:

inc r10 ;

xor rax, rax ;

mov rsi, r9 ;

syscall

cmp r10, r8 ;

je .end ;

cmp byte [r9], 0xca ;

je .end ;

cmp byte [r9], 0x21 ;

jb .next ;

.again:

dec r8

cmp r8, 1

je .overflow

inc r9

xor rax, rax

mov rsi, r9

syscall

cmp byte [r9], 0x21

jb .end

cmp byte [r9], 0xca

je .end

jmp .again

.end:

mov byte [r9], 0

pop rax

mov rdx, r9

sub rdx, rax

pop rbx

ret

.overflow:

pop rax

mov rax, 0

mov rdx, 0

pop rbx

ret

*; Принимает указатель на строку, пытается*

*; прочитать из её начала беззнаковое число.*

*; Возвращает в rax: число, rdx : его длину в символах*

*; rdx = 0 если число прочитать не удалось*

parse\_uint:

xor rax, rax

push r13

mov r13, 10

xor rax, rax

xor rcx, rcx

xor rdx, rdx

xor rsi, rsi

.parse\_char:

mov sil, [rdi + rcx]

cmp sil, '0'

jl .return

cmp sil, '9'

jg .return

inc rcx

sub sil, '0'

mul r13

add rax, rsi

jmp .parse\_char

.return:

mov rdx, rcx

pop r13

ret

*; Принимает указатель на строку, пытается*

*; прочитать из её начала знаковое число.*

*; Если есть знак, пробелы между ним и числом не разрешены.*

*; Возвращает в rax: число, rdx : его длину в символах (включая знак, если он был)*

*; rdx = 0 если число прочитать не удалось*

parse\_int:

xor rax, rax

cmp byte [rdi], 0x2d

je .parse\_ng

call parse\_uint

ret

.parse\_ng:

inc rdi

call parse\_uint

cmp rdx, 0

je .return

neg rax

inc rdx

.return:

Ret

*; Принимает указатель на строку, указатель на буфер и длину буфера*

*; Копирует строку в буфер*

*; Возвращает длину строки если она умещается в буфер, иначе 0*

string\_copy:

xor rax, rax

xor r9, r9

xor rcx, rcx

call string\_length

push rax

push rsi

.loop:

cmp rcx, rdx

je .error

mov r8, [rdi+rcx]

mov [rsi+rcx], r8

cmp rax, 0

je .end

dec rax

inc rcx

jmp .loop

.end

pop rsi

pop rax

mov byte [rsi+rax], 0

ret

.error:

pop rsi

pop rax

mov rax, 0

ret

Вывод: Была реализована библиотека, которая в будущем, надеюсь, поможет мне в выполнение следующих лабораторных работ.