Университет ИТМО, кафедра ВТ

Лабораторная работа №1 по “Языкам Системного Программирования”

Работу выполнил

студент группы P3200

Рогов Я. С.

Преподаватели:

Жирков И. О.

Балакшин П. В.

Санкт-Петербург, 2016

Задание: реализовать на языке ассемблера (синтаксим Intel) библиотеку базовых операций ввода/вывода символов, строк, и целых чисел

Код библиотеки:

```
section .text

; rdi - string pointer
; returns rax: string's length
string_length:
    xor rcx, rcx

    .loop:
        mov al, byte[rdi + rcx]
        inc rcx
        test al, al
        jnz .loop

    lea rax, [rcx-1]
    ret

; rdi - string pointer
print_string:
    push rdi
    call string_length
    pop rsi
    mov rdx, rax
    mov rax, 1
    mov rdi, 1
    syscall
    ret

; rdi - ASCII character's code
print_char:
    push di
    mov rax, 1
    mov rdi, 1
    mov rsi, rsp
    mov rdx, 1
    syscall
    pop di
    ret

print_newline:
    mov rdi, 0xa
    call print_char
    ret

; rdi - number
; returns rax: string's length
print_uint:
    mov rax, rdi
    mov r8, rsp       ; saving original address
    dec rsp
    mov byte[rsp], 0 ; NUL
    mov rsi, 10

    .loop:
       dec rsp
       xor rdx, rdx
       div rsi
       add dl, 0x30 ; from int to char
       mov byte[rsp], dl
       test rax, rax
       jnz .loop

    mov rdi, rsp
    push r8           ; pushing original address
    push rsp          ; pushing string pointer
    call print_string
    pop rax           ; poping string pointer
    pop rsp           ; poping original address
    sub rax, rsp      ; 8 * (strlen + 1)
    shr rax, 3        ; strlen + 1
    dec rax           ; strlen
    ret

; rdi - number in machine interpretation - two's
complement
; returns rax : string's length
print_int:
    test rdi, rdi
    jns .positive
    push rdi
    mov rdi, 0x2d ; 0x2d = '-'
    call print_char
    pop rdi
    not rdi
    inc rdi

.positive:
    call print_uint
    ret

; rdi and rsi - two string pointers
; returns 1, if strings are equal, 0 otherwise
string_equals:
    xor rcx, rcx

    .loop:
        mov al, [rdi+rcx]
        mov dl, [rsi+rcx]
        xor dl, al
        jnz .not_equal
        inc rcx
        test al,al
        jnz .loop

    mov rax, 1
    ret

.not_equal:
    xor rax, rax
    ret

; returns ax : ASCII character
read_char:
    xor rax, rax
    mov rdi, 0
    lea rsi, [rsp-1]
    mov rdx, 1
    syscall
    mov ax, [rsp-1]
    ret

section .data
word_buffer times 256 db 0

section .text

; returns rax: word's string pointer, rdx :
word's length
read_word:
    xor rax, rax
    mov rdi, 0
    mov rsi, word_buffer
    mov rdx, 256
    syscall

    mov rdx, word_buffer

    .loop:
        mov al, byte[rdx]
        test al, 0xdf ; space and NUL
        jz .found
        cmp al, 0x9 ; \t
        jz .found
        cmp al, 0xa ; \n
        jz .found
        inc rdx
        jmp .loop

.found:
    mov byte[rdx], 0
    sub rdx, rsi
    mov rax, word_buffer
    ret
```

```
; rdi points to a string
; returns rax: number, rdx : length
parse_uint:
    xor rax, rax
    xor rcx, rcx
    mov rsi, 10

    .loop:
        xor r8, r8
        mov r8b, byte[rdi+rcx]

        sub r8b, 0x30
        js .end ; < 0x30
        cmp r8b, 9
        ja .end ; > 0x39
        mul rsi
        add rax, r8
        inc rcx
        jmp .loop

    .end:
    mov rdx, rcx
    ret
```

```
; rdi points to a string
; returns rax: number, rdx : length
parse_int:
    mov al, byte[rdi]
    cmp al, 0x2d
    jnz .positive
    inc rdi
    call parse_uint
    not rax
    inc rax
    inc rdx
    ret

.positive:
    call parse_uint
    ret

string_copy:
    xor rcx, rcx

    .loop:
        mov al, byte[rdi+rcx]
        mov byte[rsi+rcx], al
        inc rcx
        test al, al
        jnz .loop
    ret
```

Вывод: в ходе выполнения данной лабораторной работы я познакомился с основами архитектуры современных компьютеров, освоил язык ассемблера синтаксиса Intel, ознакомился с наиболее используемыми командами и системными вызовами, а также узнал и применил стандарты и правила написания процедур и функций.