Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики.

Кафедра вычислительной техники

Языки системного программирования

Лабораторная работа №2

Выполнил

Студент группы P3210

Глушков Дмитрий Сергеевич

Санкт-Петербург
2018 г.

Задание:
Написать словарь представленный в виде связного списка.
Файл main.asm:
```
section .text
%include "colon.inc"
%include "words.inc"
%include "lib.asm"
%include "dict.asm"
global _start

_start:
    push rbp
    mov rbp, rsp
    sub rsp, 256
    mov rdi, rsp
    call read_word
    mov rdi, rax
    mov rsi, x
    call find_word
    test rax, rax
    jz .nfound
    add rax, 8
    push rax
    mov rax, [rsp]
    mov rdi, rax
    call string_length
    pop rdi
    add rdi, rax
    inc rdi
    call print_string
    mov rsp, rbp
    pop rbp
    mov rdi, 0
    call exit

.nfound:
    mov rdi, not_found
    call print_error
    mov rsp, rbp
    pop rbp
    mov rdi, 0
    call exit
```

Файл colon.inc:
```
%define x 0
%macro colon 2
%%x: dq x
db %1, 0
xt_ %+ %2:
%define x %%x
%endmacro
```

Файл dict.asm:
```
global find_word
```

```
section .rodata
not_found: db "Not found",0
section .text

find_word:
    xor rax, rax
.loop:
    test rsi, rsi
    jz .end
    push rdi
    push rsi
    add rsi, 8
    call string_equals
    pop rsi
    pop rdi
    test rax, rax
    jnz .found
    mov rsi, [rsi]
    jmp .loop
.found:
    mov rax, rsi
.end:
    ret
```

Файл words.inc:

```
colon "yeah", yeah
db "Asm is a great thing",0
colon "please", please
db "rate this at 100", 0
colon "asd", asd
db "no asd allowed", 0
colon "itsmo", itsmo
db " than a university?", 0
```

Файл lib.asm:

```
section .text
global print_error
global string_length
global print_char
global print_newline
global print_string
global print_uint
global print_int
global string_equals
global parse_uint
global parse_int
global read_word
global string_copy
global exit

print_error:
    push rdi
    call string_length
    pop rsi
    mov rdx, rax
```

```
    mov rax, 1
    mov rdi, 2 ;stderr
    syscall
    ret


string_length:
    xor rax, rax
.loop:
    cmp byte [rdi+rax], 0 ;rax- счетчик, в нем же длина строки
    je .end
    inc rax
    jmp .loop
.end:
    ret


print_string: ;просто вызываем write syscall number, в rdx длина
    mov rsi, rdi
    call string_length
    mov rdx, rax
    mov rax, 1
    mov rdi, 1
    syscall
    ret


print_char:  ;помещаем char в стек и вызываем print_string с вершиной
стека
     push rdi
    mov rdi, rsp
    call print_string
    pop rdi
    ret


print_newline: ;print_char со знаком конца строки
    mov rdi, 10
     jmp print_char


print_uint:
     xor rax, rax
     mov rax, rdi
     mov r8, 0xA
     mov r9, 0x30 ;сдвиг от числа к коду числа
     mov r10, 0 ;счётчик
.loop:
     xor rdx, rdx
     div r8 ;
     add rdx, r9
     push rdx
     inc r10
     test rax, rax
     jnz .loop
```

```
.p_loop:
      test r10, r10
      jz .end ;если 0, то конец
      pop rdi
      call print_char
      dec r10
      jmp .p_loop
.end:
      ret


print_int:
    test rdi, rdi
    jns print_uint
    push rdi
    mov rdi, '-'
    call print_char
    pop rdi
    neg rdi
    jmp print_uint


string_equals: ; rdi - str1, rsi - str2, результат в rax
    xor rcx, rcx
.loop:
      mov al, byte[rdi+rcx]
      cmp al, byte[rsi+rcx]
      jnz .neq
      inc rcx
      cmp al, 0
      jz .eq
      jmp .loop
.neq:
      xor rax, rax
      ret
.eq:
      mov rax, 1
      ret


read_char:
      xor rax, rax
      xor rdi, rdi ; 0 - stdin дескриптор
      push rax
      mov rsi, rsp ; в rsi адрес выршины стека
      mov rdx, 1
      syscall
      pop rax
      ret


section .data
word_buffer times 256 db 0

section .text
```

```
read_word: ;возвращать в rax строку, в rdx длину, буфер - 256 байт
      xor r10, r10 ;количество символов, счётчик
.before:
      call read_char
      cmp rax, 0x9
      jz .before
      cmp rax, 0xA
      jz .before
      cmp rax, 0x20
      jz .before
.loop:
      cmp rax, 0
      jz .end
      cmp rax, 0x9 ; tab
      jz .end
      cmp rax, 0xA
      jz .end
      cmp rax, 0x20 ; пробел
      jz .end
      mov byte[word_buffer+r10], al
      call read_char
      inc r10
      jmp .loop
.end:
      cmp r10, 256
      jnb .end_uns
      mov byte[word_buffer+r10], 0
      mov rax, word_buffer
      mov rdx, r10
      ret
.end_uns:
      mov rax, 0
      mov rdx, 0
      ret


; rdi points to a string
; returns rax: number, rdx : length
parse_uint:
    xor rax, rax
      xor r9, r9
      xor rcx, rcx ;счётчик
      mov r10, 10
.loop:
      mov r9b, byte[rdi + rcx]
      cmp r9, 0
      jz .end
      cmp r9, '0'
      jb .end
      cmp r9, '9'
      ja .end
      mul r10 ;сдвиг цифр для записи новой
      sub r9, '0' ; код знака -> число
      add rax, r9
```

```
      inc rcx
      jmp .loop
.end:
      mov rdx, rcx
      ret


; rdi points to a string
; returns rax: number, rdx : length
parse_int:
      cmp byte[rdi], '-'
      je .sign
      jmp parse_uint
.sign:
      inc rdi ; для чтения после минуса
      call parse_uint
      neg rax
      inc rdx ; +1 для знака
      ret


string_copy:
      push rdx
      push rdi
      call string_length
      mov rdx, rax
      mov rdi, rsi
      call string_length
      pop rdi
      cmp rax, rdx
      jb .end
      xor rax, rax
.loop:
      mov dl, byte[rdi+rax]
      mov byte[rsi+rax], dl
      inc rax
      cmp dl, 0
      je .end
      jmp .loop
.end:
      pop rdx
      ret


exit:
    xor rdi, rdi
    mov rax, 60
    syscall
```

Вывод:

В результате проделанной работы на языке Ассемблер был написан словарь, представленный в виде связного списка.