

Факультет программной инженерии и компьютерной техники

Низкоуровневое программирование

Лабораторная работа №1

Преподаватель:

Логинов Иван Павлович

Выполнил:

Рождественский Никита Сергеевич

Р33113

Задание

Необходимо реализовать на ассемблере словарь в виде связного списка. Каждое вхождение содержит адрес следующей пары в словаре, ключ и значение. Ключи и значения — адреса нуль-терминированых строк.

Словарь задаётся статически, каждый новый элемент добавляется в его начало. С помощью макросов мы автоматизируем этот процесс так, что указав с помощью новой конструкции языка новый элемент он автоматически добавится в начало списка, и указатель на начало списка обновится. Таким образом нам не нужно будет вручную поддерживать правильность связей в списке.

Создайте макрос colon с двумя аргументами: ключом и меткой, которая будет сопоставлена значению. Эта метка не может быть сгенерирована из самого значения, так как в строчке могут быть символы, которые не могут встречаться в метках, например, арифметические знаки, знаки пунктуации и т.д. После использования такого макроса можно напрямую указать значение, сопоставляемое ключу.

Выполнение

main.asm

%include "words.inc"

%define DICT\_START last\_address

%define KEY\_MAX\_SIZE 255

%define STDOUT 1

%define STDERR 2

global \_start

extern read\_word

extern find\_word

extern string\_length

extern print\_string

extern print\_newline

extern exit

section .data

reading\_error\_msg: db "Problem with reading", 0

not\_found\_msg: db "Non-Existent key", 0

section .text

\_start:

mov rsi, KEY\_MAX\_SIZE

sub rsp, KEY\_MAX\_SIZE

mov rdi, rsp

call read\_word ; Read key into rax

test rax, rax ; Check if word is null or error?

je .reading\_error\_exit

mov rdi, rax

mov rsi, DICT\_START

call find\_word ; Check dictionary for key

test rax, rax

je .not\_found\_exit ; Check if no such key

add rax, 8 ; Set accumulator as a key pointer

mov r10, rax ; Save rax in r10

mov rdi, rax ; Provide string address for string\_length

call string\_length ; calc key length

inc r10 ; Skipping key string

add r10, rax ; To access value string

mov rdi, r10 ; Printing value string

mov r8, STDOUT

call print\_string

jmp .exit

.reading\_error\_exit:

mov rdi, reading\_error\_msg

call string\_length

mov rsi, rax

jmp .error\_exit

.not\_found\_exit:

mov rdi, not\_found\_msg

call string\_length

mov rsi, rax

jmp .error\_exit

.error\_exit:

mov r8, STDERR

call print\_string

jmp .exit

.exit:

call print\_newline

mov rax, 60

mov rdi, rax

syscall

ret

dict.asm

find\_word:

mov r8, rdi

.loop:

mov r9, rsi

test rsi, rsi ; Check if dictionary is empty (rsi = 0)

je .exit\_not\_found

mov rdi, r8

add rsi, 8

call string\_equals ; Compare strings

mov rsi, r9

test rax, rax ; If equals then exit

jnz .exit\_found

mov rsi, [rsi] ; If not then next pair

jmp .loop

.exit\_found:

mov rax, rsi

ret

.exit\_not\_found:

mov rax, 0

ret

colon.inc

%define last\_address 0

%macro colon 2

%2:

dq last\_address

db %1, 0

%define last\_address %2

%endmacro

Makefile

AFLAGS=-felf64

ASM=nasm

all: main.asm lib.asm dict.asm

$(ASM) $(AFLAGS) main.asm

$(ASM) $(AFLAGS) lib.asm

$(ASM) $(AFLAGS) dict.asm

ld -o main main.o dict.o lib.o

clean:

find . -type f -name '\*.o' -delete

find . -type f -name '\*.out' -delete

Вывод: Был реализован словарь на языке ассемблер.