Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Кафедра информатики

Отчет по лабораторной работе №4

Работа со строками

Студент: гр. 753502

Саттарова Александра Сергеевна

Руководитель: ассистент Шиманский В.В.

Минск 2018

СОДЕРЖАНИЕ

1. Введение
2. Постановка задачи
3. Программная реализация
4. Выводы

Литература

Приложение 1. Текст программы

1. Введение
   1. Данная программа реализует ввод строки и вывод результата обработки строки.
   2. Цель лабораторной работы — ознакомиться с командами работы со строками, работой с индексами SI и DI
   3. В ходе выполнения лабораторной работы будет использован ассемблер NASM версии 2.13.03-1, работающий на процессорах архитектуры Intel x86. NASM (*Netwide Assembler*) — это свободный (LGPL и лицензия BSD) ассемблер для архитектуры Intel x86, использующийся для написания 16-, 32- и 64-разрядных программ. В NASM используется Intel-синтаксис записи инструкций.
   4. Исполняться написанная работа будет в консоли системы Arch Linux с версией ядра 4.18.16.
2. Постановка задачи
   1. Текст задания  
        
      С клавиатуры вводится строка. Необходимо удалить из неё все гласные буквы.
   2. Условие задания

В качестве языка вводимой строки выбран английский. Гласными буквами в нем считаются прописные и заглавные буквы «A», «E», «I», «O», «U», «Y».

1. Программная реализация
   1. Строка вводится с клавиатуры. Далее, проходя по каждому символу строки, сравниваем его с гласными буквами. Если символ является гласной, вся стоящая после этого символа часть строки сдвигается влево на 1. Каждую согласную выводим в консоль.
   2. Результат можно видеть в консоли.
   3. Примеры:

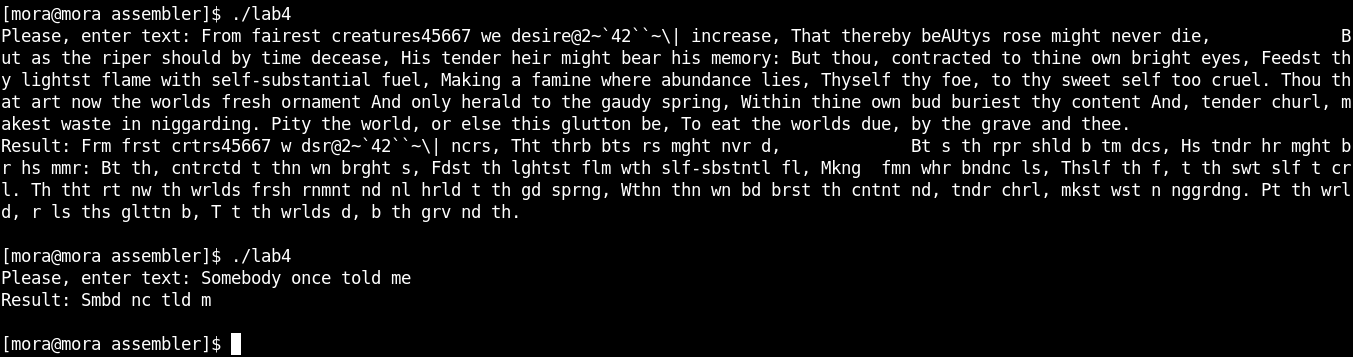


Рис. 1. Примеры ввода различных строк

4. Выводы

В ходе выполнения лабораторной работы была создана программа на языке ассемблера, осуществляющая обработку введенной строки. Были использованы индексы SI и DI, команды для работы со строками, например MOVSB (Записывает в ячейку по адресу ES:(E)DI байт из ячейки с адресом DS:(E)SI); были также реализованы процедуры для работы с индексами.

Литература

1. Learn Assembly Language - <http://asmtutor.com/>

2. Столяров А. В. Программирование на языке ассемблера NASM для ОС Unix: Уч. пособие. - 2-е изд. - М.: МАКС Пресс, 2011. - 188 с.

3. Stack Overflow — https://stackoverflow.com/

4. NASM - Википедия — https://ru.wikipedia.org/wiki/NASM

5. X86 Assembly/Interfacing with Linux - Wikibooks — https://en.wikibooks.org/wiki/X86\_Assembly/Interfacing\_with\_Linux

Приложение 1. Текст программы

%define BUFSIZE 4096

section .data

string\_input: db 'Please, enter text: ', 0

string\_res: db 'Result: ', 0

nl: db 10, 0

letters: db 'aeiouyAEIOUY', 0

section .bss

output resb 8

input resb BUFSIZE

buffer resb BUFSIZE ; buffer for read operations

section .text

global \_start

\_start:

mov rsi, string\_input

call print

call read

mov rsi, string\_res

call print

call delete\_letter

mov rsi, nl

call print

xor rbx, rbx

mov rax, 1

int 80h

strlen:

mov rax, rsi

.nextchar:

cmp byte [rax], 0

je strlen.finished

inc rax

jmp strlen.nextchar

.finished:

sub rax, rsi

ret

print:

call strlen

mov rdx, rax ; number of bytes

mov rax, 1 ; system call for write

mov rdi, 1 ; file handle 1 is stdout

;mov rsi, message ; address of string to output

syscall

ret

; rax read(rdi) - reads from terminal at EDI, rax shows how many was read

read:

mov rax, 0 ; system call for read

mov rdi, 0 ; file handle 0 is stdin

mov rsi, input ; address of string to input

mov rdx, BUFSIZE ; number of bytes

syscall

mov byte [rsi + rax - 1], 0 ; remove linefeed

ret

;delete letter from input string and store result in output string

delete\_letter:

push rcx

push rsi

mov rsi, input

call strlen

mov rdx, rax

.loo: ; for( ; rsi != '\0'; inc rsi)

cmp rdx, 0

je .fin

mov rcx, 12

mov rdi, 0

mov rdi, letters

.check: ; for(rcx = 12; rcx > 0; dec rcx)

mov bl, byte [rdi]

cmp byte [rsi], bl ; if([rsi] == letters[rcx]) {

je .delete ; call delete

inc rdi ; break;}

loop .check ; else continue

mov al, byte [rsi]

mov [output], al

inc rsi

call append

.break:

dec rdx

jmp .loo

.fin:

pop rsi

pop rcx

ret

;delete letter shifting string left by 1

.delete:

push rsi

push rbx

push rcx

call strlen

mov rcx, rax

mov rdi, buffer

.loopy:

mov bl, byte [rsi+1]

mov byte [rsi], bl ; shift left by 1

movsb

loop .loopy

.exit:

mov rsi, rdi

pop rcx

pop rbx

pop rsi

jmp .break

;void append - prints symbol

append:

push rsi

push rdx

push rax

push rdi

mov rsi, output

call print

pop rdi

pop rax

pop rdx

pop rsi

ret