Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Лабораторная работа № 2

по дисциплине ПнаЯА

Вариант 14

Выполнил студент гр. 150502: Снитко Д.А.

Проверил: Туровец Н.О.

Минск 2022

Теоретические сведения:

1. Директивы определения данных.
2. Команды пересылки данных и способы адресации.
3. Команды передачи управления
4. Строковые операции
5. Прерывания ввода-вывода
6. Макросы

Задание:

Заменить в строке слова, содержащие заданный набор букв, на другое заданное слово.

Код:

.model small

.stack 100h

.data

msg1 db "Input string:", 0dh, 0ah, '$' ; текст для вывода

msg2 db 0dh, 0ah, "Enter the symbol set:", 0dh, 0ah, '$'

msg3 db 0dh, 0ah, "Result:",0dh,0ah,'$'

msg4 db 0dh, 0ah, "Enter replacing word:",0dh,0ah,'$'

emsg5 db 0dh, 0ah, "Error",0dh,0ah,'$'

emsg6 db 0dh, 0ah, "Error:There is space in set of symbols",0dh,0ah,'$'

string db 202 dup("$") ; исходная строка

symset db 202 dup("$") ; нужная подстрока

repword db 202 dup("$") ; новое слово

capacity EQU 200

flagSpstr db 0

flagEnd db 0

flagIn db 0

.code

start proc

mov ax, @data ; переместить адрес сегмента данных в DS и ES

mov ds, ax

mov es, ax

mov ah, capacity ; установить максимальный размер строки в первом байте

mov string[0], ah

mov symset[0], ah

mov repword[0], ah

lea dx, msg1 ; ввод вывод

call puts

lea dx, string

call gets

lea dx, msg2

call puts

lea dx, symset

call gets

lea dx, msg4

call puts

lea dx, repword

call gets

call checkSet ; вызывающая процедура, которая будет проверять подстроку на пустоту и пробелы

cld

lea si, string[2]

lea di, symset[2]

lea di, symset[2]

xor cx,cx

call ReplaceWord ; вызов основной процедуры

ret

endp start

checkSet proc ; вызывающая процедура, которая будет проверять подстроку на пустоту и пробелы

push di

push cx

lea cx,symset[2]

add cl,symset[1]

mov di,cx ; установить di в начале подстроки, двигаясь назад

dec di

xor cx,cx

mov cl,symset[1]

cmp cx,0 ; проверка на пустоту

jne lloop ; если не равно

pop cx

pop di

jmp error1

lloop:

cmp byte ptr [di],' ' ; проверка на sпробелы

jne decrease

pop cx

pop di

jmp error2 ; безусловный переход

decrease:

dec di ; уменьшает число на единицу

loop lloop

pop cx

pop di

ret ; возврат из ближней процедуры

endp checkSet

endProg proc ; завершение программы

jmp fin

error1:

lea dx, emsg5

call puts

jmp fin

error2:

lea dx,emsg6

call puts

fin:

mov ah, 4ch

int 21h

ret

endp endProg

showResult proc ; показать строковый буфер

push ax

push dx

lea dx, msg3

call puts

lea dx,string[2]

call puts

pop dx

pop ax

ret

endp showResult

gets proc ; ввод

mov ah, 0Ah

int 21h

ret

endp gets

puts proc ; вывод

mov ah, 9

int 21h

ret

endp puts

ReplaceWord proc ; замена слова

cycle\_:

push si

push di

push ax

xor ax,ax

lodsb ; загружать символ, пока не найдем пробел

cmp al,' ' ; если первый символ не пробел - найти конец слова

jne fndEnd

call toBegin ; пропуск пробелов

fndEnd:

dec si

mov bx,si ; запоминаем начало строки

call findEnd ; нахождение конца слова и установка dx на символ после него

xor ax,ax

mov ax,dx ; колво символов для удаления

sub ax,bx

call isDelim ; поиск подстроки

cmp flagIn,1 ; если нашли

je proccess

push di

mov di,dx

cmp byte ptr[di],24h ; проверка на конец строки

pop di

je exit

jmp cycle\_ ; если не было подстроки - начать заново

proccess:

mov flagIn,0

; call showResult

call delWord

; call showResult

call InsWord

; call showResult

call changeLen ; изменение длины для правильного вывода

add bl,repWord[1]

mov si, bx

mov di,dx

cmp byte ptr[di],24h ; проверка на конец строки

je exit

jmp cycle\_

pop ax

pop di

pop si

jmp exit

preExit:

mov flagEnd,1

exit:

call showResult

call endProg

ret

endp ReplaceWord

changeLen proc ; изменение длины для правильного вывода

push ax

mov cx,ax

cmp al,repWord[1]

ja decLen

jb incLen

sub cl,repWord[1]

jmp m

decLen:

sub al,repWord[1]

mov cx,ax

decloop:

dec byte ptr string[1]

loop decloop

jmp m

incLen:

mov cl,repWord[1]

sub cl,al

incloop:

inc byte ptr string[1]

loop incloop

m:

pop ax

ret

endp changeLen

isDelim proc ; поиск подстроки

push cx

push si

push di

push ax

mov si,bx ; установка si в начале слова

push bx

lea cx,symset[2] ;

mov di,cx ; установить di в начало подстроки

mov bx,di

xor cx,cx

mov cx,ax ; количество символов для сравнения

lodsb

loop\_:

cmp al, byte ptr [di] ; проверка на совпадение

je incrr

lodsb

mov di,bx

loop loop\_

jmp mark

incrr: ; если символы равны

push ax

lea ax,symset[2]

add al, symset[1]

dec ax

cmp ax,di ; проверить конец подстроки

pop ax

je setFlag ; если конец то ставим флаг

inc di

lodsb

loop loop\_

jmp mark

setFlag:

mov flagIn,1

mark:

pop bx

pop ax

pop di

pop si

pop cx

ret

endp isDelim

delWord proc ; удаление символов

push si

push di

push cx

xor cx,cx

mov cl,string[1]

mov di,bx

repe movsb

pop cx

pop di

pop si

ret

endp delWord

InsWord proc ; нужные символы

push si

push di

push cx

lea cx, string[2]

add cl,string[1]

sub cl, al

mov si,cx

dec si ; установить si на последний символ исходной строки

push bx

mov bx,si

add bl,repword[1]

mov di,bx ; установить di на последний символ новой строки

pop bx

call setNum ; колво символов для копирования

std

repe movsb

lea si, repword[2]

mov di,bx

xor cx,cx

mov cl,repword[1]

cld

repe movsb

pop cx

pop di

pop si

ret

endp InsWord

setNum proc ; колво для копирования

push ax

mov ax, si

sub ax,bx

cmp ax,0

js \_\_end

je \_\_end

mov cx,ax

add cl,repWord[1]

jmp \_\_\_end:

\_\_end:

mov cx,0

\_\_\_end:

pop ax

ret

endp setNum

toBegin proc ; пропуск пробелов

\_loop:

cmp byte ptr [si],24h ; проверка на конец строки

je exit

lodsb

cmp al,' '

je \_loop

ret

endp

findEnd proc ; поиск конца слова и установка dx на символ после него

\_loop\_:

lodsb

cmp byte ptr [si],24h ; проверка на конец строки

je end\_\_

cmp al,' '

jne \_loop\_

end\_:

dec si

end\_\_:

mov dx,si

ret

endp

Демонстрация работы:

