**Тема:** «Функции DOS»

**Цель работы:** Целью работы является функций DOS, их применение в программах на языке ассемблера.

**Ход работы:**

Вариант №4

Я ознакомился с теоретической частью и приступил к выполнению задания. Мне необходимо ввести с клавиатуры строку. Ввести с клавиатуры коротенькую строку - шаблон. Найти шаблон во введенной строке. Вывести на экран “ДА”, если шаблон есть и “НЕТ”, если нет.

Код:

data segment ; директива начала сегмента данных

msg1 DB "Введите строку: $"

msg2 DB 0Ah, 0Dh, "Введите шаблон: $"

msg3 DB 0Ah, 0Dh, "Да!$"

msg4 DB 0Ah, 0Dh, "Нет$"

str1ml DB 200

str1l DB '$'

str1 DB 200 dup('$')

str2ml DB 200

str2l DB '$'

str2 DB 200 dup('$')

data ends

code segment ; директива начала сегмента кодов

assume cs: code, ds: data ; установка соответсвий сегментов и их адресов

start:

mov ax, data

mov ds, ax

mov es,ax

xor ax,ax

lea dx, msg1 ;вывод msg1

call strout

lea dx, str1ml ;ввод строки str1

call strin

lea dx, msg2 ;вывод msg2

call strout

lea dx, str2ml ;ввод строки str2

call strin

xor cx, cx

mov cl, str1l ;устанавливается кол-во повторений

sub cl, str2l

inc cl

cld ;Сбросить флаг направления

lea di, str2 ;в di - строка, которую надо найти

lea si, str1 ;в si - строка, в которой ищем

xor ax, ax

beg: ;повторяется для str1 до конца строки минус длина str2

push cx

push di

push si

mov bx, si

mov cl, str2l

compare\_strings:

mov al, [si] ; Считываем байт из строки source

mov bl, [di] ; Считываем байт из строки destination

cmp al, bl ; Сравниваем байты

jne \_neq ; Если не равны, переход

; Если байты равны:

inc si ; Переходим к следующему байту строки source

inc di ; Переходим к следующему байту строки destination

loop compare\_strings ; Уменьшаем CX и повторяем, если CX > 0

jmp \_eq ; Все байты равны

\_eq:

lea dx, msg3 ;вывод msg1

call strout

jmp \_end

\_neq:

pop si

pop di

pop cx

inc si

loop beg

lea dx, msg4

call strout

\_end:

mov ah, 4ch

int 21h

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Процедуры\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

strin proc

mov ah, 0Ah

int 21h

ret

strin endp

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

strout proc

mov ah, 09h

int 21h

ret

strout endp

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

code ends

end start

Вывод: в ходе выполнения лабораторной работы я научился применять функции DOS в программах на языке ассемблера

**Контрольные вопросы**

**1. Что такое программное прерывание?**

Прерыванием (interrupt) называется способ общения центрального

процессора с периферийными устройствами. Программные прерывания позволяют использовать предоставляемые системой MS-DOS и

BIOS большой набор подпрограмм, выполняющих различные полезные действия.

**2. Какие возможности работы с клавиатурой имеются у программиста?**

Программист может выполнять ввод с клавиатуры в буфер строки символов, ввод символа с клавиатуры и отображать/не отображать его на экране, отображать его на стандартный дисплей, на принтер, отображать строку на стандартный вывод, а также выполнять опрос состояния буфера клавиатуры.

**3. Чем отличаются друг от друга различные функции DOS, выполняющие ввод с клавиатуры?**

Есть 4 функции DOS для ввода с клавиатуры: 01H, 07H, 08H и 0AH.

Первая - выполняет ввод с клавиатуры одного символа и отображает его на

экране.

Вторая и третья - выполняет ввод с клавиатуры одного символа и не отображает его на экране. Отличие 07H от 08H в том, что первая функция не отслеживает нажатие Ctrl/C (или Ctrl/Break), в отличие от второй.

Четвёртая - выполняет ввод с клавиатуры в буфер строки символов.

**4. Как работает команда xlat?**

Команда xlat заносит ASCII-код символа в регистр al

**5. Можно ли выдать на экран текст ‘$1.00=25,00 rub/’, используя функции DOS 09h?**

Нет, нельзя, потому что знак $ указывает на конец строки, следовательно если он стоит в начале, то ничего выводиться не будет

**6.Какие режимы адресации удобно использовать при работе с одномерными массивами?**

Удобно пользоваться регистровыми и индексно-базовым режимами адресации

**7.Что означают выражения в поле операндов в строках примера leabx, IN\_STR+2 и mov OUT\_STR+3, al?**

Первая команда означает загрузку адреса буфера ввода (начала введённой строки), а вторая – помещение результата в буфер вывода

**8.Как выделить младшую тетраду байта?**   
Для выделения младшей тетрады байта необходимо использовать команду and al,00001111b , что можно расшифровать как: сбросить 4-7 разряды числа

**9.Как выделить старшую тетраду байта?**

Для выделения старшей тетрады байта необходимо использовать команду and al,11110000b , что можно расшифровать как: сбросить 0-3 разряды числа

**10.В чем отличие команд lea BX, STR и mov BX, offset STR?**

1) lea BX, STR – при использовании этой команды указатель вычисляется в процессе выполнения программы. Здесь можно пользоваться косвенной адресацией.

2) mov BX, offset STR – при использовании этой команды указатель вычисляется при компилировании. Здесь нельзя пользоваться косвенной адресацией.

В итоге в BX помещается адрес (указатель) ячейки памяти, в которой расположен mass.