|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

**ФАКУЛЬТЕТ** ***ИУК «Информатика и управление»***

**КАФЕДРА** \_\_***ИУК4 «Программное обеспечение ЭВМ, информационные технологии»***

**ЛАБОРАТОРНАЯ РАБОТА №8**

**«Обработка строк»**

**ДИСЦИПЛИНА: «Машинно-зависимые языки программирования»**

|  |  |  |
| --- | --- | --- |
| Выполнил: студент гр. ИУК4-32Б | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Карельский М.К. )  (Подпись) |
| Проверил: | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Амеличева К.А. )  (Подпись) |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | | |
|  | - Балльная оценка:  - Оценка: | |

Калуга, 2021

**Цель:** практическое овладение навыками разработки программного кода на языке Ассемблер. Изучение организации обработки цепочек данных и команд строковых примитивов.

**Вариант 9**

Слова в строке разделены запятыми. Вывести каждое слово на экран в отдельной строке лесенкой, т.е. выдать каждое слово в столбик, с фиксированным сдвигом относительно начала предыдущего;

1. в строке ST2 найти символы цифр и заменить их пробелами, по количеству равными значению цифры. Подсчитайте количество замен.
2. Переслать в строку ST1 слова из ST2 начинающиеся и заканчивающиеся на согласную букву.

**Блок-схемы:**



**Рисунок 1.** Формирование «лесенки»



**Рисунок 2.** Замена цифр пробелами



**Рисунок 3.** Пересылка слов с первой и последней согласными буквами

**Листинг:**

.model small

.stack 100h

.386

.data

ST1 db 'the,number,pi,is,3,14,if,rounded', '$', 47 dup(0)

input db 80, 0

ST2 db 80 dup(0)

temp db 80 dup(0)

counter db 0

S\_ST1 db 'ST1: ', '$'

S\_inputST2 db 'Input ST2: ', '$'

S\_ST2 db 'ST2: ', '$'

S\_count db 'Count: ', '$'

mSetColorMode macro

push ax

push bx

mov ah, 10h

mov al, 3

sub bl, bl

int 10h

pop bx

pop ax

endm

mClearScreen macro

push ax

push bx

push cx

push dx

mov ah, 6

sub al, al

mov bh, 0F0h

sub cx, cx

mov dx, 184Fh

int 10h

pop dx

pop cx

pop bx

pop ax

endm

mPrintSymbol macro symbol

push ax

push bx

push cx

mov ah, 9

mov al, symbol

mov bl, 0F0h

mov cx, 1

int 10h

pop cx

pop bx

pop ax

endm

mPrintString macro string

push ax

push dx

mov ah, 09h

lea dx, string

int 21h

pop dx

pop ax

endm

mMoveCursor macro row, column

push ax

push bx

push dx

mov ah, 2

mov dh, row

mov dl, column

sub bh, bh

int 10h

pop dx

pop bx

pop ax

endm

mShiftCursor macro row, column

push ax

push bx

push cx

push dx

mov ah, 3

sub bh, bh

int 10h

add dh, row

add dl, column

mov ah, 2

int 10h

pop dx

pop cx

pop bx

pop ax

endm

mIncCursor macro

push ax

push bx

push cx

push dx

mov ah, 3

sub bh, bh

int 10h

inc dl

mov ah, 2

int 10h

pop dx

pop cx

pop bx

pop ax

endm

mNextLine macro

push ax

push bx

push cx

push dx

mov ah, 3

sub bh, bh

int 10h

inc dh

mov dl, 0

mov ah, 2

int 10h

pop dx

pop cx

pop bx

pop ax

endm

mOutput macro

LOCAL NEGATIVE

LOCAL SPLIT

LOCAL OUTPUT

push dx

push si

push di

mov si, 10

sub di, di

cmp ax, 0

jl NEGATIVE

jmp SPLIT

NEGATIVE:

neg ax

push ax

mPrintSymbol '-'

mIncCursor

pop ax

SPLIT:

inc di

sub dx, dx

div si

add dx, 30h

push dx

cmp ax, 0

jg SPLIT

OUTPUT:

pop ax

mPrintSymbol al

mIncCursor

dec di

cmp di, 0

jg OUTPUT

pop di

pop si

pop dx

endm

mInputString macro buffer

push ax

push dx

push si

mov ah, 0Ah

lea dx, buffer

int 21h

mov al, buffer[1]

inc al

inc al

xor ah, ah

mov si, ax

mov buffer[si], '$'

pop si

pop dx

pop ax

endm

mLadder macro string, size

LOCAL CYCLE

LOCAL REPLACE

LOCAL END\_MACRO

push ax

push bx

push cx

push di

lea bx, string

mov al, ','

mov cx, size

lea di, string

CYCLE:

repne scasb

jz REPLACE

jmp END\_MACRO

REPLACE:

dec di

mov byte ptr[di], '$'

mPrintString [bx]

mShiftCursor 1, 0

mov byte ptr[di], ','

inc di

mov bx, di

jmp CYCLE

END\_MACRO:

mPrintString [bx]

pop di

pop cx

pop bx

pop ax

endm

mReplaceDigits macro string, temp, size, counter

LOCAL CYCLE

LOCAL SEARCH

LOCAL REPLACE

LOCAL SEARCH\_END

LOCAL SEARCH\_ZERO

LOCAL REPLACE\_ZERO

LOCAL SEARCH\_ZERO\_END

push ax

push bx

push cx

push dx

push si

push di

mov counter, 0

xor cx, cx

mov cl, size

lea di, temp

lea si, string

rep movsb

xor dx, dx

mov bx, 9

CYCLE:

mov al, bl

add al, 30h

mov cl, size

add cx, dx

lea di, string

SEARCH:

repne scasb

jz REPLACE

jmp SEARCH\_END

REPLACE:

push cx

push di

mov si, di

lea di, temp

push si

lea si, string

sub di, si

pop si

add di, si

add di, bx

dec di

rep movsb

lea di, temp

lea si, string

sub di, si

pop si

push si

dec si

add di, si

mov cx, bx

push ax

mov al, ' '

rep stosb

add dx, bx

dec dx

mov cl, size

add cx, dx

lea di, string

lea si, temp

rep movsb

pop ax

pop di

pop cx

add di, bx

dec di

inc counter

jmp SEARCH

SEARCH\_END:

dec bx

cmp bx, 0

jne CYCLE

mov al, '0'

mov cl, size

add cx, dx

lea di, string

SEARCH\_ZERO:

repne scasb

jz REPLACE\_ZERO

jmp SEARCH\_ZERO\_END

REPLACE\_ZERO:

push cx

push di

mov si, di

lea di, temp

push si

lea si, string

sub di, si

pop si

add di, si

dec di

rep movsb

mov byte ptr [di], 0

mov cl, size

add cx, dx

lea di, string

lea si, temp

rep movsb

pop di

pop cx

dec dx

dec di

inc counter

jmp SEARCH\_ZERO

SEARCH\_ZERO\_END:

pop di

pop si

pop dx

pop cx

pop bx

pop ax

endm

mCheckConsonant macro symbol, return

LOCAL TRUE

LOCAL FALSE

LOCAL END\_MAC

cmp byte ptr symbol, 'b'

je TRUE

cmp byte ptr symbol, 'B'

je TRUE

cmp byte ptr symbol, 'c'

je TRUE

cmp byte ptr symbol, 'C'

je TRUE

cmp byte ptr symbol, 'd'

je TRUE

cmp byte ptr symbol, 'D'

je TRUE

cmp byte ptr symbol, 'f'

je TRUE

cmp byte ptr symbol, 'F'

je TRUE

cmp byte ptr symbol, 'g'

je TRUE

cmp byte ptr symbol, 'G'

je TRUE

cmp byte ptr symbol, 'h'

je TRUE

cmp byte ptr symbol, 'H'

je TRUE

cmp byte ptr symbol, 'j'

je TRUE

cmp byte ptr symbol, 'J'

je TRUE

cmp byte ptr symbol, 'k'

je TRUE

cmp byte ptr symbol, 'K'

je TRUE

cmp byte ptr symbol, 'l'

je TRUE

cmp byte ptr symbol, 'L'

je TRUE

cmp byte ptr symbol, 'm'

je TRUE

cmp byte ptr symbol, 'M'

je TRUE

cmp byte ptr symbol, 'n'

je TRUE

cmp byte ptr symbol, 'N'

je TRUE

cmp byte ptr symbol, 'p'

je TRUE

cmp byte ptr symbol, 'P'

je TRUE

cmp byte ptr symbol, 'q'

je TRUE

cmp byte ptr symbol, 'Q'

je TRUE

cmp byte ptr symbol, 'r'

je TRUE

cmp byte ptr symbol, 'R'

je TRUE

cmp byte ptr symbol, 's'

je TRUE

cmp byte ptr symbol, 'S'

je TRUE

cmp byte ptr symbol, 't'

je TRUE

cmp byte ptr symbol, 'T'

je TRUE

cmp byte ptr symbol, 'v'

je TRUE

cmp byte ptr symbol, 'V'

je TRUE

cmp byte ptr symbol, 'w'

je TRUE

cmp byte ptr symbol, 'W'

je TRUE

cmp byte ptr symbol, 'x'

je TRUE

cmp byte ptr symbol, 'X'

je TRUE

cmp byte ptr symbol, 'y'

je TRUE

cmp byte ptr symbol, 'Y'

je TRUE

cmp byte ptr symbol, 'z'

je TRUE

cmp byte ptr symbol, 'Z'

je TRUE

jmp FALSE

TRUE:

mov return, 1

jmp END\_MAC

FALSE:

mov return, 0

END\_MAC:

endm

mConsonants macro source, destination, temp, size

LOCAL CYCLE

LOCAL CHECK

LOCAL MOVE

LOCAL LAST\_MOVE

LOCAL END\_CYCLE

LOCAL CHECK\_COMMA

LOCAL REMOVE\_COMMA

LOCAL END\_MACRO

push ax

push bx

push cx

push si

push di

lea bx, destination

mov al, ','

mov cl, size

lea di, source

mov si, di

CYCLE:

repne scasb

jz CHECK

jmp END\_CYCLE

CHECK:

dec di

dec di

mCheckConsonant [si], temp

mCheckConsonant [di], temp[1]

inc di

inc di

cmp word ptr temp, 0101h

je MOVE

mov si, di

jmp CYCLE

MOVE:

push cx

push di

mov cx, di

sub cx, si

mov di, bx

rep movsb

mov bx, di

pop di

pop cx

jmp CYCLE

END\_CYCLE:

dec di

dec di

mCheckConsonant [si], temp

mCheckConsonant [di], temp[1]

cmp word ptr temp, 0101h

je LAST\_MOVE

mov di, bx

mov byte ptr [di], '$'

jmp CHECK\_COMMA

LAST\_MOVE:

mov cx, di

sub cx, si

inc cx

mov di, bx

rep movsb

mov byte ptr [di], '$'

jmp END\_MACRO

CHECK\_COMMA:

std

mov cx, di

lea si, destination

sub di, si

inc cx

repne scasb

jz REMOVE\_COMMA

jmp END\_MACRO

REMOVE\_COMMA:

inc di

mov byte ptr [di], '$'

inc di

mov byte ptr [di], 0

END\_MACRO:

cld

pop di

pop si

pop cx

pop bx

pop ax

endm

.code

Start:

mov ax, @data

mov ds, ax

mov es, ax

xor ax, ax

mSetColorMode

mClearScreen

mMoveCursor 0, 0

mPrintString S\_ST1

mLadder ST1, 33

mNextLine

mPrintString S\_inputST2

mInputString input

inc input[1]

mNextLine

mPrintString S\_ST2

mLadder ST2, 80

mNextLine

lea si, ST1

lea di, ST2

mov cx, 80

rep cmpsb

je EQUAL

jmp DIFFERENT

EQUAL:

mReplaceDigits ST2, temp, input[1], counter

xor ax, ax

mov al, counter

mPrintString S\_ST2

mPrintString ST2

mNextLine

mPrintString S\_count

mOutput

jmp END\_PROG

DIFFERENT:

mConsonants ST2, ST1, temp, input[1]

mPrintString S\_ST1

mPrintString ST1

END\_PROG:

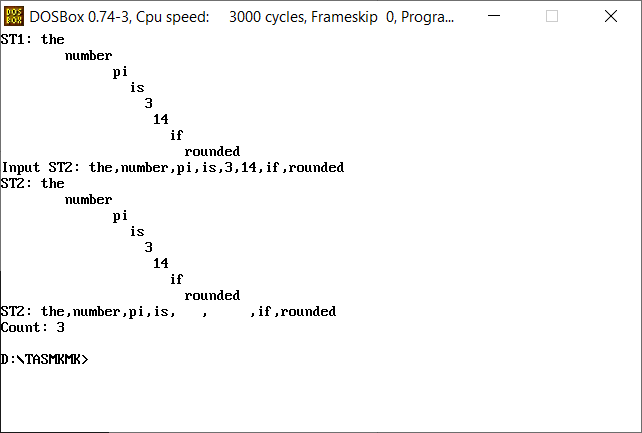
mov ax, 4C00h

int 21h

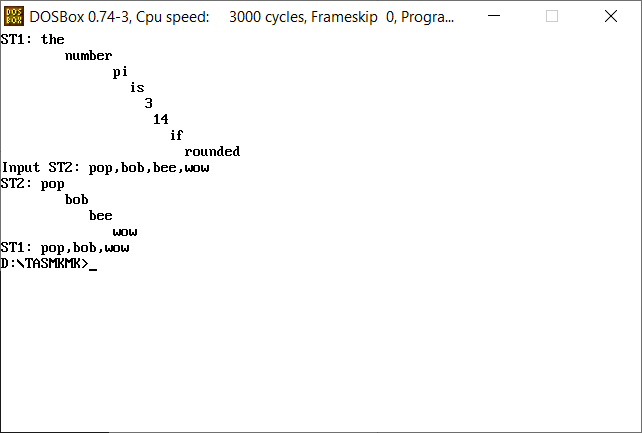
end start

END

**Результаты работы:**



**Рисунок 4.** Совпадающие строки



**Рисунок 5.** Различные строки

**Вывод:** в ходе выполнения лабораторной работы были получены навыки сравнения, копирования строк, поиска в них символа.