Министерство образования Республики Беларусь

Учреждение образования Белорусский государственный университет информатики и радиоэлектроники

Кафедра ЭВМ

Отчёт по лабораторной работе №2

“Обработка символьных данных”

Проверил: Выполнил:

ассистент каф. ЭВМ студент гр.030501

Туровец Николай Олегович Вайтехович П.В.

Минск 2022

**Цель работы:** Ознакомиться с директивами определения данных, изучить команды пересылки данных и передачи управления, изучить строчные операции и прерывания консольного ввода-вывода высокого уровня.

**Алгоритм**

* определение данных
* пересылка данных
* передача управления
* прерывания ввода/вывода
* макросы
* рекурсия

Директивы определения данных указывают ассемблеру, что в соответствующем месте программы располагается переменная, определяют тип переменной (байт, слово и т.д.), задают ее начальное значение и ставят в соответствие переменной метку, которая будет использоваться для обращения к этим данным. Базовой командой пересылки данных является команда MOV. Команды передачи управления служат для организации ветвления вычислительного процесса. Прерывания ввода-вывода – специальные команды передачи управления, вызывающие функции BIOS или DOS, предоставляющие сервис по работе с аппаратурой ПЭВМ. Прерывания ввода-вывода – специальные команды передачи управления, вызывающие функции BIOS или DOS, предоставляющие сервис по работе с аппаратурой ПЭВМ.

**Листинг программы**

name "Reverse"

.model tiny

.code

org 100h

output macro output\_string, offset\_space

mov dx, offset output\_string

add dx, offset\_space

mov ah, 09h

int 21h

endm

start:

output message, 0 ; Welcoming message

mov bx, offset string

mov [bx], SIZE ; setting buffer size (so we can input all 200 symboles if needed)

mov dx, bx

mov ah, 0Ah

int 21h

add bl, [string + DOLLAR\_OFFSET] ; getting string length + '$' offset

add bx, SYMBOL\_OFFSET ; and adding 0Ah and 0Dh to it (means incrementing by two)

mov ax, dx ; checking if the string is not empty

add ax, SYMBOL\_OFFSET

cmp ax, bx

je exit\_failure

mov [bx], '$'

xor dx,dx ; set dx to null

xor ax,ax ; set ax to null

lea bx, string ; repeating the operation

add bx, SYMBOL\_OFFSET ; after setting '$'

mov si, bx ; saving the begining position

mov di, bx ; of a string

dec si

next\_symbol:

inc si ; going through the string

mov al, [si] ; reading a symbol

cmp al, ' '

je end\_found

cmp al, '$'

je end\_found

jmp next\_symbol

end\_found:

mov dx, si ; found space or '$'

dec si ; going one symbol back

mov bx, di ; in order to know the word's last symbol

reverse\_word:

cmp bx, si ; inverse the word

jae done

mov al, [bx]

mov ah, [si]

mov [si], al

mov [bx], ah

inc bx

dec si

jmp reverse\_word

done:

mov si, dx ; restoring the position of the symbol after the reversed word

inc dx ; getting the position of

mov bx, dx ; possible first symbol of the next word

mov di, bx ; saving it

mov ah, [si]

cmp ah, '$'

jne next\_symbol

output result\_message, 0

output new\_string, 0

output string, SYMBOL\_OFFSET

mov ah, 01h ; waiting for any symbol

int 21h ; to exit the programm

ret

exit\_failure:

output error\_message, 0

mov ah, 01h ; waiting for any symbol

int 21h ; to exit the programm

ret

string db 200 dup('$')

new\_string db 0Ah, 0Dh, '$'

message db "Please, enter the string:", 0Dh, 0Ah, '$'

result\_message db 0Dh, 0Ah, "Result : ", '$'

error\_message db 0Dh, 0Ah, "Error : the string is empty!", 0Dh, 0Ah, '$'

DOLLAR\_OFFSET equ 1

SYMBOL\_OFFSET equ 2

SIZE equ 200

end startR\_OFFSET equ 1

SYMBOL\_OFFSET equ 2

SIZE equ 200

end startSET equ 2

SIZE equ 200

end startT equ 2

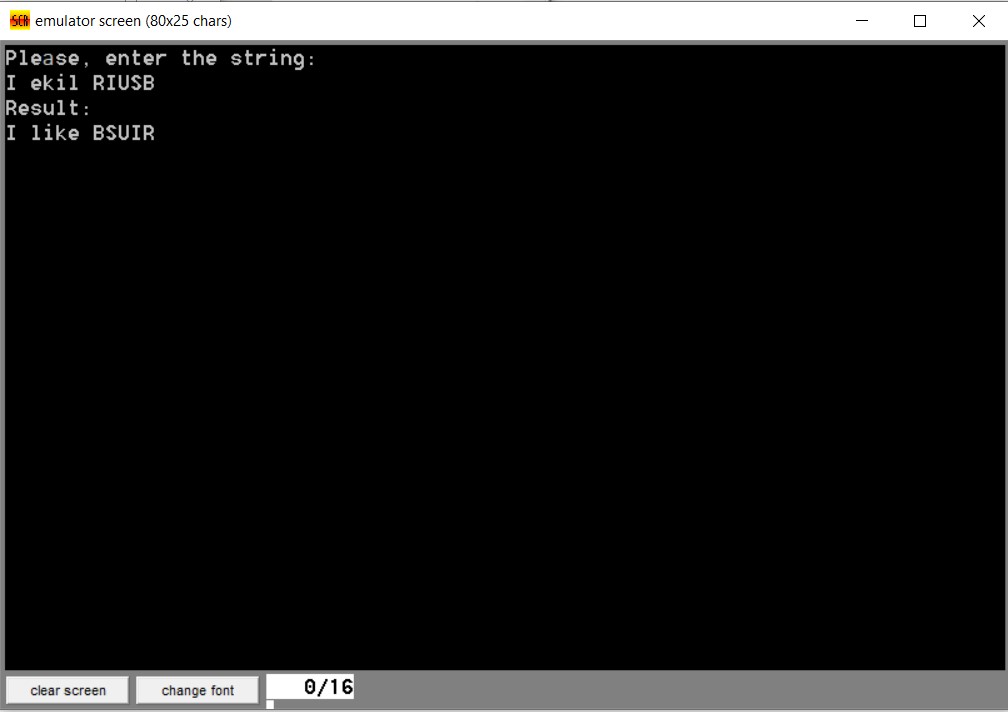
SIZE equ 200

end startequ 1

SYMBOL\_OFFSET equ 2

SIZE equ 200

end startt

**Тест**

*Рисунок 1 –* результат выполнения программы

**Вывод:** В данной лабораторной работе разработана программа, которая осуществляет реверс символов слов строки и выводит их на экран.