Министерство образования Республики Беларусь

Учреждение образования «Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа № 3

За 6 семестр

По дисциплине: «Аппаратно-программное обеспечение ЭВМ и сетей»

Тема: «Ассемблерные вставки. Строки. Условия. Обработка символьных данных. Макроопределения»

Выполнил:

Студент 3-го курса

Группы АС-56

Ровенский Д.В.

Провери:

Булей Е.В.

Брест 2022

Цель работы: Изучить приемы разработки макроопределений использования их в программах.

Содержание работы

Изучить состав и средства задания макроопределений.

Написать макроопределение, реализующее функцию заданного преподавателем варианта работы.

Написать программу проверки работоспособности разработанного макроопределения.

Вариант 4

Дан текст – непустая последовательность не длиннее ста символов. Признаком конца ввода является точка, в сам текст точка не входит.

Преобразованный текст напечатать.

Ввод текста, проверка условия, обработка текста и печать результата должны выполняться последовательно, отдельными частями программы.

4) Переместить заданный символ, если он содержится в строке, в начало строки.

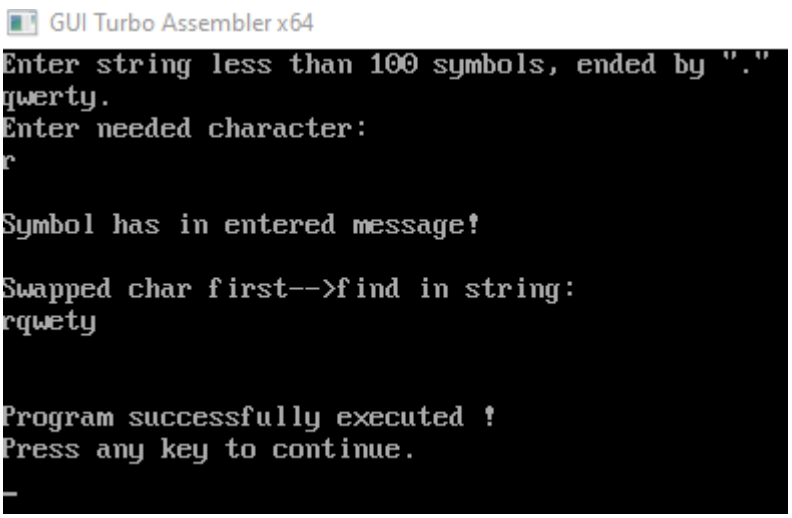
Код:

```
.MODEL SMALL
.STACK 256
.DATA
;Сегмент данных в которых могут храниться инициализированние переменные так и не
инициализированнные
info_string_for_input db 'Enter string less than 100 symbols, ended by "."',10,13,'$'
info_string_for_input_char db 10,13,'Enter needed character:',10,13,'$'
info_string_for_input_char_found db 10,13,10,13,'Symbol has in entered message!',10,13,'$'
info_string_for_input_char_notfound db 10,13,10,13,'Symbol not found!',10,13,'$'
info_string_for_output_char_swapped db 10,13,'Swapped char first-->find in
string:',10,13,'$'

work_input_string db  100 dup ('$')
;Строка байтовая, в 100 символов, оканчивается спец символом конца строки $

.CODE
;Сегмент кода, здесь распологаются все кодовые операции над переменными из сегменнта
данных

Swap_letter_in_first_pos macro source, char
;Создаём макроопределение в которое входными данными будут служить строка и искомый символ
в этой строке

    lea di, source
;Загрузка в выходной источник нашего сообщения, над которым оперируем
    mov bl, [di]
;В bl регистр загружаем адрес первого элемента(Понадобится для итерирования в обратную
сторону)
    mov al, char
;В регистр al грузим наш байтовый символ
    cld
;Очистка лагов
    repne scasb
;Строковая команда, сравнивает символ из al с сиволом в di если не совпадает, плюсует di,
```

```
при совпадении устанавливает флаги равенства, обход можно делать в две стороны в
зависимости от установленных флагов
    je found
;Если флаг равенства будет установлен
    mov ah, 09h
;Закидываем в регистр ah номер функции вывода строк
    lea dx, info_string_for_input_char_notfound
;Загружаем в dx адрес строки на вывод
    int 21h
;Передаём кправление ядру
    jmp exit
;Отлетаем в конец программы, если сюда дашло

found:
    mov ah, 09h
;Функция строк
    lea dx, info_string_for_input_char_found
;Сообщение
    int 21h
;Ядро

    dec di
;Уменьшаем наш di дрес в строке, чтобы символ был который нам нужен

looper_for_swap_letters_in_message:
    cmp byte ptr [di], bl
;Лоопер на проверку равенства нашего адреса из bl и адреса di
    je exit
;Если совпадает, в конец отлетаем
    mov ah, [di-1]
;Операция смещения символов
    mov [di], ah
;Смещение также
    dec di
;Уменьшаем наш адрес в di
    jmp looper_for_swap_letters_in_message
;Продолжаем итерирование

exit:
    mov [di], char
endm

enter_string_function proc near
;Функция ввода сообщения
    lea     dx, info_string_for_input
    mov     ah, 09h
    int     21h

    mov     cx, 100
    lea     bx, work_input_string

looper_input_char:
;Каждый символ при введение проверяется на точку, если точка то выход из функции
    mov     ah, 01h
    int     21h
    cmp     al,  '.'
    je input_char_exit
    mov     byte ptr [bx], al
    inc     bx
    loop    looper_input_char

    input_char_exit:
```

```
    ret
enter_string_function endp

output_string_function proc near
;Функция вывода строки нашей переделанной

    lea dx, info_string_for_output_char_swapped
    mov ah, 09h
    int 21h

    lea dx, work_input_string
    mov ah, 09h
    int 21h

    ret
output_string_function endp

start:
;Главный сегмент проги, которая исполняется
    mov bx, @data
;Закидываем в bx дрес даты
    mov ds, bx
;ds источкик закидываем наш сегмент даты
    mov es, bx
    call enter_string_function
;Вызываем нашу функции ввода сообщения

    lea     dx, info_string_for_input_char
;Сообщение на вывод
    mov     ah, 09h
    int     21h

    mov     ah, 01h
;Функция 01h отвечает за ввод с клавиатуры
    int     21h

    Swap_letter_in_first_pos work_input_string, al
;Выполняем наше макроопределение

    call output_string_function
;Выводим результат
    mov ah, 4ch
;Функция ДОС-завершения
    int 21h
END start
```

Вывод программы:



Вывод: ознакомились с основными методами реализации ассемблерных вставок, инструкциями условии и взаимодействие со строками, научились создавать макроопределения.