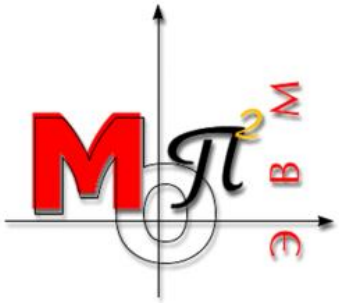
МИНОБРНАУКИ РОССИИ
Федеральное государственное автономное образовательное учреждения высшего образования
«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт компьютерных технологий и информационной безопасности
Кафедра математического обеспечения и применения ЭВМ



ЛАБОРАТОРНАЯ РАБОТА № 6

по дисциплине
«Машинно-ориентированное программирование»

на тему:
«Подпрограммы в языке ассемблер»
Вариант № 7

Выполнили:
Студенты группы
КТбо2-8

Жалнин Д. И.
Нестеренко П. А.

Проверил:
ассистент кафедры
МОП ЭВМ

_____ Гуляев Н. А.
подпись

Оценка
_____

«____» _____ 2020 г.

Таганрог 2020

1 ЦЕЛЬ РАБОТЫ

1.1 Дидактическая цель работы

Ознакомление с методами составления подпрограмм для программ на языке «Ассемблер», использующих ввод/вывод информации в консоли пользователя, обработку символьных строк.

1.2 Практическая цель работы

В рамках лабораторной работы необходимо разработать программу на языке ассемблера, алгоритм которой выполняет задачу согласно описанному индивидуальному заданию, скомпилировать и запустить код программы с помощью программного пакета «TASM».

2 ВАРИАНТ ЗАДАНИЯ

2.1 Общие требования

Для всех вариантов требуется выполнить разработку программного модуля при помощи СРПО «Turbo Assembler», реализующего некоторую обработку массива символов с помощью подпрограммы, введенных из консоли, а также вывод результата работы.

2.2 Индивидуальное задание, вариант № 7

Разработать подпрограмму, которая вставляет подстроку в строку, начиная с заданной позиции. Разработать программу, которая вводит с клавиатуры исходную строку, вводит подстроку и позицию вставки, вставляет подстроку в строку.

3 ХОД РАБОТЫ

3.1 Описание высокоуровневой реализации

В ход работы была составлена программа на языке программирования «Ассемблер», используя модель памяти «**small**», в котором допускается наличие одного сегмента кода и единственного сегмента данных. Размер стека программы – 256 байт.

Перед началом основного сегмента кода происходит инициализация структур данных, необходимых для работы программы:

```
locals __
model small
stack 100h

dataseg
; Сообщения на вывод
;MESS1 db 0dh,0ah,"Enter the string:",'$'
MESS1 db 0dh,0ah,"Enter string:","$"
MESS2 db 0dh,0ah,"Enter substring:","$"
MESS3 db 0dh,0ah,"Enter position:","$"
MESS4 db 0dh,0ah,"Result:","$"

; Основная строка
S_BUFLEN db 80 ; Макс. длина основной строки
S_FACTLEN db ? ; Длина фактически введенной основной строки
S_INPBUF db 80 dup(?) ; Введенная основная строка

; Подстрока
S_BUFLEN_SUB db 20 ; Максимальная длина подстроки
S_FACTLEN_SUB db ?; Фактическая длина подстроки
S_INPBUF_SUB db 20 dup(?) ; Введенная подстрока
TEMP dw ? ; Переменная общего назначения
; Индекс вставки
N_BUFLEN db 3 ; Макс. длина числа при вводе
N_FACTLEN db ? ; Фактическая длина
N_INPBUF db 3 dup(?) ; Строка представления числа
POSINS dw ? ; Позиция, начиная с которой вставляем
```

Алгоритм работы программы:

```
;Разработать подпрограмму, которая вставляет подстроку в строку,
;начиная с заданной позиции. Разработать программу, которая вводит с
;клавиатуры исходную строку, вводит подстроку и позицию вставки,
;вставляет подстроку в строку.
codeseg
startupcode

; Ввод основной строки
MLOOP: lea DX, MESS1
    mov AH, 09h
    int 21h ; приглашение
    lea DX,S_BUFLEN
    mov AH, 0Ah
    int 21h ; Ввод строки
```

```
    mov BL, S_FACTLEN
    cmp BL,0 ; если строка пустая, то заканчиваем
    jne LLL0 ;Нет – продолжать
    jmp QUIT

LLL0: mov BH, 0
    ;Дополнить длину до слова
    add BX, 2 ; и получить адрес позиции
    add BX, DX ; сразу после конца строки
    mov byte ptr [BX],0 ;Записать признак конца строки

; Ввод подстроки
LLL1: lea DX, MESS2
    mov AH,09h
    int 21h
    lea DX,S_BUFLEN_SUB
    mov AH, 0Ah
    int 21h ; Ввод строки
    mov BL, S_FACTLEN_SUB

; Ввода числа
LLL2: lea DX, MESS3
    mov AH, 09h
    int 21h
    lea DX, N_BUFLEN
    mov AH, 0Ah
    int 21h ;Ввод строки числа
    lea BX, N_INPBUF ;Адрес строки представления числа
    mov CL, N_FACTLEN;Длина этой строки

    call TO_NUM; Вызов функции перевода в число

    ; Обрабатываем "нештатные" ситуации
    jc LLL1 ; Ошибка? Повторить ввод
    cmp AL, 0 ;Ноль?
    je LLL1
    cmp AL, S_FACTLEN;Превышает длину строки?
    jg LLL1

    mov BL, S_FACTLEN
    SUB BL, AL
    add BX, 1h
    mov POSINS, BX ;Запомнить позицию удаления
; Вывод строки
LLL3:
    call PRINT
    jmp MLOOP
QUIT:
    exitcode 0;


; Подпрограмма вывода результата
```

```
PRINT proc near
    _1: ; Выводим сообщение result:
        lea DX, MESS4
        mov AH, 09h
        int 21h
        lea BX, S_INPBUF
        mov Cl, S_FACTLEN
    _2: ; Выводим первую строку до индекса вставки
        mov DX, [BX]
        cmp CX,POSINS
        je _3
        mov AH, 02h
        int 21h
        inc BX
        mov DX,[BX] ; для того чтобы последний символ выводился
        loop _2
    _3: ; Подготовка к выводу подстроки
        cmp CX,0
        je _6
        dec CX
        push CX BX
        lea BX, S_INPBUF_SUB
        mov Cl, S_FACTLEN_SUB
    _4: ; Вывод подстроки
        mov DX, [BX]
        mov AH, 02h
        int 21h
        inc BX
        loop _4
    _5: ; Вытаскиваем данные из стека
        pop BX CX
        jmp _2
    _6: ; Возвращаемся в основу
        mov AH, 02h
        int 21h
        ret
endp PRINT

; Подпрограмма перевода в число
TO_NUM proc near
push DX ;Сохранить все изменяемые регистры,
; кроме AX, в котором результат
mov CH, 0 ;Расширяем длину до слова
mov AX, 0 ;Начальное значение результата
mov DL, 10 ;Основание системы счисления
__1: imul DL ;Умножить на основание
    jc __2 ;Переполнение байта?
    mov DH, [BX] ;Очередная цифра
    sub DH, '0' ;Получить значение цифры
    jl __2 ;Это была не цифра!
    cmp DH, 9
    jg __2 ;Это опять же была не цифра!
```

```
    add AL, DH ;+ значение цифры к результату
    jc __2 ;Переполнение байта?
    inc BX ;Сдвиг по строке
    loop __1 ;Цикл по строке
    jmp __3 ;Нормальное число
__2: stc ;Было переполнение – устанавливаем CF
__3: pop DX ;Восстановить все, что сохраняли
    ret
TO_NUM endp
end
```

3.4 Описание полученных результатов

Программный модуль был скомпилирован, запущен и отлажен в рамках среды «DOS BOX». При вызове «TASM» были заданы ключи «-L -ZI», которые позволили получить отладочные файлы. При вызове «TLINK» были использованы ключи «-V». Результат работы можно наблюдать на рисунке 1.

Рисунок 1 – результат работы программы

4 ВЫВОДЫ

4.1 Полученные знания, навыки, умения

В ходе выполнения лабораторной работы была разработан и отлажен программный модуль, который считывает с консоли две строки и число, на позицию которого к первой строке добавляется вторая.