Министерство образования Республики Беларусь Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей Кафедра Информатики

Дисциплина «Конструирование программ»

## ОТЧЕТ

к лабораторной работе №3 на тему:

## «Создание простой программы на языке ассемблера. Обработка символьных данных»

## БГУИР 6-05-0612-02 85

Выполнил студент группы 353503 МАРТИНОВИЧ Андрей Александрович

(дата, подпись студента)

Проверил ассистент каф. Информатики

РОМАНЮК Максим Валерьевич

(дата, подпись преподавателя)

Минск 2024

# ЗАДАНИЕ

## Задание 1. Вариант 1.

Написать программу «Hello, world!».

## Задание 2. Вариант 17.

Заменить в строке слова являющиеся числами на слово «<number>».

# ВЫПОЛНЕНИЕ РАБОТЫ

## Задание 1

Для выполнения программы в сегмент data сначала помещаем строку, которую собираемся вывести в консоль.. После этого мы загружаем адрес строки в dx, в ah загружаем функци 09h для вывода строки. После этого вызываем прерывания для функции 09h.

Исходный код задания представлен в листинге 1.

Листинг 1 – Исходный код задания 1

.model tiny

.data

str db 'Hello word!', 0Dh, '$'

.code

main:

lea dx, str

mov ah, 09h

int 21h

mov ah, 4Ch

int 21h

end main

Результат работы представлен на рисунке 1.

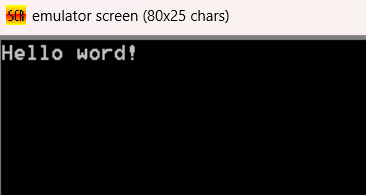


Рисунок 1 – Результат выполнения программы

Как видно из рисунка 1 программа работает корректно.

## Задание 2

Для выполнения задачи ищем слова, которые являются числами, а далее заменяем их на <number>.

Исходный код представлен в листинге 2.

Листинг 2 – Исходный код задания 2

.model tiny

.stack 100h

.data

str db '<number>'

enter\_string db 'Enter a string:', 0Dh, 0Ah, '$'

buffer db 200, ?, 200 dup('$')

is\_number db 1

start\_word dw 0

count\_word db 0

.code

main:

; enter string

mov ah, 09h

lea dx, enter\_string

int 21h

mov ah, 0Ah

lea dx, buffer

int 21h

lea si, buffer + 2

mov start\_word, si

lea ax, buffer

iterate:

lodsb

cmp al, 0Dh

je end\_iterate ;exit

cmp al, ' ' ;space

jne start\_of\_word

jmp iterate

start\_of\_word: ;set si on begin of string

mov is\_number, 1

dec si

mov start\_word, si

lodsb

cmp al, '-'

je if\_minus

dec si

jmp iterate\_in\_word

if\_minus:

cmp [si], ' '

je not\_digit\_word

cmp [si], '$'

je not\_digit\_word

cmp [si], 0Dh

je not\_digit\_word

jmp iterate\_in\_word

iterate\_in\_word:

lodsb

cmp al, ' '

je end\_of\_word

cmp al, 0Dh

je end\_of\_string

cmp al, '0' ;number check

jb not\_digit\_word

cmp al, '9'

ja not\_digit\_word

jmp iterate\_in\_word

not\_digit\_word: ;set is\_number false

mov is\_number, 0

jmp iterate\_in\_word

end\_of\_word: ;to the next word

cmp is\_number, 0

je iterate

mov si, start\_word

jmp replace\_digits

end\_of\_string: ;end check string

cmp is\_number, 0

je end\_iterate

mov si, start\_word

jmp replace\_digits\_end

replace\_digits\_end: ;replace last number

lodsb

cmp al, 0Dh

je end\_iterate

mov byte ptr [si-1], 1

jmp replace\_digits\_end

replace\_digits: ;replace numbers

lodsb

cmp al, ' '

je iterate

mov byte ptr [si-1], 1

jmp replace\_digits

end\_iterate:

lea si, buffer + 2

iterate\_del: ;delete same symbols

lodsb

cmp al, 0Dh

je end\_del

cmp [si], 1

jne iterate\_del

cmp al, 1

jne iterate\_del

dec si

mov bx, 0

jmp del

del:

cmp [si+bx], 0Dh

je iterate\_del

mov al, [si+bx+1]

mov [si+bx], al

inc bx

jmp del

end\_del:

lea si, buffer + 2

get\_count: ;get count of numbers in string

lodsb

cmp al, '$'

je replace

cmp al, 1

jne skip\_char

inc count\_word

skip\_char:

jmp get\_count

replace:

cmp count\_word, 0

je print\_string ;algorithm insert <number>

dec count\_word

lea si, buffer+2

jmp find\_end

find\_end:

lodsb

cmp al, '$'

jne find\_end

dec si

mov bx, 0

lea di, si+7

jmp iterate\_insert

insert: ; insert <number>

lea di, si-bx

lea si, str

mov cx, 8

rep movsb

cmp count\_word, 0

je print\_string

jmp replace

iterate\_insert: ;increasing the insertion space

lea ax, buffer

add ax, 2

cmp ax, si+bx

ja print\_string

cmp [si+bx], 1

je insert

mov ax, [si+bx]

mov [di+bx], ax

dec bx

jmp iterate\_insert

print\_string: ;print string

mov ah, 02h

mov dl, 0Dh

int 21h

mov dl, 0Ah

int 21h

mov ah, 09h

lea dx, buffer+2

int 21h

mov ah, 4Ch

int 21h

end main

После выполнения данной программы можно убедиться в правильности полученного результата с помощью рисунке 2.

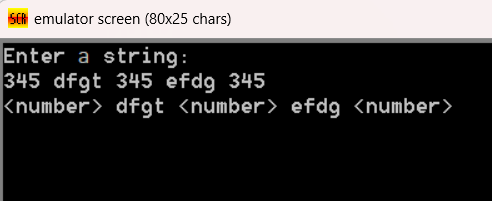


Рисунок 2 – Результат работы программы

Как видно из рисунка 2 программа работает корректно и выполняет поставленную перед ней задачу.

**ВЫВОД**

В ходе выполнения лабораторной работы была написана программа на ассемблере для процессора Intel 8086, который является 16-битным микропроцессором, ставшим основой архитектуры x86. Этот процессор работает с регистрами шириной 16 бит, что позволяет эффективно обрабатывать данные и управлять памятью.