

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**  
**Кафедра МО ЭВМ**

**ОТЧЕТ**  
**по лабораторной работе №7**  
**по дисциплине «Организация ЭВМ и систем»**  
**Тема: Преобразование целых чисел. Использование процедур в**  
**Ассемблере.**

Студент гр. 1383

\_\_\_\_\_

Валиев Р.Р.

Преподаватель

\_\_\_\_\_

Ефремов М.А.

Санкт-Петербург

2022

### **Цель работы.**

Составить программу для преобразования чисел из одной системы счисления в другую.

### **Задание.**

Разработать на языке Ассемблер IntelX86 две процедуры: одна - прямого и другая - обратного преобразования целого числа, заданного в регистре AX или в паре регистров DX:AX, в строку, представляющую его символьное изображение в заданной системе счисления (с учетом или без учета знака в зависимости от варианта задания).

Строка должна храниться в памяти, а также выводиться на экран для индикации.

Отрицательные числа при представлении с учетом знака должны в памяти храниться в дополнительном коде, а на экране изображаться в прямом коде с явным указанием знака или в символьном виде со знаком.

Вариант 1.1.2.1.B

### **Выполнение работы.**

В процедуре `str_8_to_num` каждая цифра умножается на восемь в степени позиции этой цифры с помощью функции `digits_proc`. Затем результат складывается. В процедуре `num_to_str_8` число делится на восемь, до тех пор, пока не будет нулем. Остаток от деления записывается в строку `oct_str`. В процедуре `num_to_str_16` делим число на шестнадцать, получаем остаток. Переводим его в шестнадцатеричную систему и добавляем к строке `hex_str`. В главной процедуре `main` происходит запись в регистр `ax` исходного числа (`number` в сегменте данных). Далее проверяется знак числа, если число положительное, то в `sign` кладем знак плюс, если отрицательное, то минус. Далее это пригодится для вывода числа со знаком. Затем с помощью процедуры `num_to_str_8` преобразовываем исходное число в строку (в восьмеричной системе счисления) и записываем это значение в `oct_str`. С помощью процедуры

reverse переворачиваем эту строку и с помощью процедуры write\_message выводим. Затем с помощью процедуры str\_8\_to\_num получаем из строки число, помещаем его в dx. Далее переводим число в шестнадцатеричную систему с помощью функции num\_to\_str\_16 и выводим его. Тот результат, который мы получили на экране должен совпадать со значением, находящемся в number.

Таблица 1 – результаты тестирования.

№ п/п	Входные данные	Выходные данные	Комментарии
1.	1AAfh	15257	Программа работает корректно.
2.	0A1h	241	Программа работает корректно.
3.	372	1562	Программа работает корректно.
4.	-1Fh	-37	Программа работает корректно.

### **Выводы.**

Разработана программа, преобразующая число из регистра dx:ax в восьмеричное число, представленное в символьном строчном отображении и обратно.

## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла lb7.asm:

ASSUME CS:Code, DS:Data, SS:AStack

AStack SEGMENT STACK

DB 1024 DUP(?)

AStack ENDS

Data SEGMENT

hex\_info DB 'Before: ', '\$'

oct\_info DB 'After: ', '\$'

hex\_str DB ' ', '\$'

oct\_str DB ' ', '\$'

sign DB ' ', '\$'

number DW -1Fh

Data ENDS

Code SEGMENT

write\_message PROC NEAR

mov AH, 9

int 21h

ret

write\_message ENDP

digits\_proc PROC NEAR

push bx

```
push dx
push cx
cmp cx,0
je digits_end
```

```
mov bx,08h
digits_processing:
    xor dx,dx
    mul bx
    sub cx,1
    cmp cx,0
    jne digits_processing
```

```
digits_end:
pop cx
pop dx
pop bx
ret
digits_proc ENDP
```

```
reverse PROC NEAR
    pop cx

    pop di
    pop bx

    push cx

    xor ax,ax
```

```

reverse_processing:
    push bx
    mov bx,ax
    mov cx,[di+bx]
    pop bx
    mov dx,[di+bx]

    push bx
    mov bx,ax
    mov [di+bx],dx
    pop bx
    mov [di+bx],cx

    add ax,2
    sub bx,2
    cmp ax,bx
    jl reverse_processing

ret

```

reverse ENDP

str\_8\_to\_num PROC NEAR

```

    pop cx

```

```

    pop di

```

```

    pop bx

```

```

    push cx

```

```

    push ax

```

```

    xor dx,dx

```

```

    mov cx,0
str_processing:
    mov ax,[di+bx]
    sub ax,'0'
    call digits_proc
    add dx,ax
    inc cx
    sub bx,2
    cmp bx,0
    jnl str_processing

    pop ax
    pop cx

    push dx

    push cx
    ret
str_8_to_num ENDP

```

```

num_to_str_8 PROC NEAR
    pop cx

    pop di
    pop dx

    push cx
    push ax

```



```

    sub bx,bx
    mov ax,dx
    mov cx, 08h
oct_processing:
    sub dx,dx
    div cx
    add dx,'0'
    mov [di+bx],dx
    add bx,2
    cmp ax,0
    jne oct_processing
    mov cx','$'
    mov [di+bx],cx
    sub bx,2

    pop ax
    pop cx

    push bx

    push cx

    ret
num_to_str_8 ENDP

num_to_str_16 PROC NEAR
    pop cx

    pop di
    pop dx

```

```

push cx
push ax

sub bx,bx
mov ax,dx
mov cx, 10h
hex_begin:
    sub dx,dx
    div cx
    add dx,'0'
    cmp dx,'9'
    jle end_hex
    add dx,7

    end_hex:
    mov [di+bx],dx
    add bx,2
    cmp ax,0
    jne hex_begin
mov cx','$'
mov [di+bx],cx
sub bx,2

pop ax
pop cx

push bx

```

push cx

ret

num\_to\_str\_16 ENDP

main PROC FAR

push ds

sub ax,ax

push ax

mov ax, DATA

mov ds, ax

mov dx,offset oct\_info

call write\_message

mov ax,number

mov di,offset sign

mov bx,'+'

cmp ax,0

jnl set\_sign

mov bx,'-'

neg ax

set\_sign:

push bx

mov [di],bx

push ax

mov dx,offset sign

call write\_message

pop ax

mov di,offset oct\_str

push ax

push di

call num\_to\_str\_8

pop bx

push bx

mov di,offset oct\_str

push bx

push di

call reverse

mov dx,offset oct\_str

call write\_message

pop bx

mov di,offset oct\_str

push bx

push di

call str\_8\_to\_num

pop dx

```
pop bx
cmp bx,'-'
jne skip
neg dx
skip:

mov di,offset hex_str
push dx
push di
call num_to_str_16
pop bx

push bx
push di
call reverse

mov dx,offset hex_info
call write_message
mov dx,offset hex_str
call write_message

ret
```

```
main ENDP
```

```
Code ENDS
```

```
END main
```