

**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: Использование арифметических операций над целыми числами и  
процедур в Ассемблере**

Студентка гр. 1381

Деркачева Д.Я.

Преподаватель

Ефремов М. А.

Санкт-Петербург

2022

## **Цель работы**

Изучить протекание арифметических процедур над целочисленными значениями на языке ассемблера и разработать код согласно выданному заданию.

## **Текст задания**

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.

## **Ход выполнения работы**

Задание: 16-битное число, с учетом знака, десятичная система счисления, способ вызова процедур – far, связь по данным между основной программой и подпрограммами через PОН.

В главной процедуре MAIN происходит запись в регистры исходного числа, которое берется из сегмента данных. Далее проверяется знак числа, если число положительное, то в SIGN кладется знак '+', если отрицательное, то '-', будет использоваться для посимвольного вывода представления числа. Затем с помощью процедуры Int\_to\_dec\_str исходное число преобразовывается в строку (в десятичной системе счисления), после чего записывается в DEC\_STR. Печатается строка с помощью процедуры WriteMsg. После того, как получено строковое представление числа, обнуляем регистр AX. В него будет записано

число после обратного преобразования. Затем процедурой `Dec_str_to_int` конвертируем строку в число. Результат, записанный в регистр `AX` должен совпадать со значением, находящемся в `NUMBER`.

Процедура `Int_to_dec_str`: число делится на 10, до тех пор, пока будет не 0, а к остатку от деления добавляется код символа '0', полученное значение кладется на стек, затем элементы из стека записываются в строку (таким образом мы получим нужный порядок цифр без дополнительных обработок).

Процедура `Dec_str_to_int`: циклом подсчитываем длину числа. Затем значение, лежащее в регистре `AX`, умножается на 10, и к нему добавляется разность кодов символов из записи числа и '0'. Данные действия осуществляются в цикле, пока не пройдем всю строку. В конце мы проверяем, было ли исходное число отрицательным, в случае, если да, применяем команду `neg` для регистра `AX`.


### **Текст исходного файла программы**

Текст исходной программы `lab7.asm` см. в приложении А.

### **Тестирование**

Рис.1. Результат тестирования

```
NUMBER DW 1111h
```



```
C:\>lab7.exe  
Decimal: +4369
```

## Выводы по работе

В ходе выполнения лабораторной работы была разработана программа , которая дает возможность разобраться в тонкостях работы с числами на языке ассемблера.

## Приложение А

### Текст исходного файла lab7.asm

```
AStack  SEGMENT STACK
        DB 1024 DUP(?)
AStack  ENDS

DATA     SEGMENT
        DECIM DB 'Decimal: ', '$'
        N DW 0
        DEC_STR DB ' ', '$'
        SIGN DB ' ', '$'
        NUMBER DW 1111h
DATA     ENDS

CODE     SEGMENT
        ASSUME CS:CODE, DS:DATA, SS:AStack

WriteMsg PROC NEAR
        mov AH, 9
        int 21h
        ret
WriteMsg ENDP

Int_to_dec_str proc FAR
        push ax
        push cx
        push dx
```

```

    push bx
    xor cx,cx
    mov bx,10
    mov di, offset DEC_STR

lp1:          ;oct
    xor dx,dx
    div bx
    add dl,'0'
    push dx
    inc cx
    test ax,ax
    jnz lp1

lp2:
    pop dx
    mov [di],dl
    inc di
    loop lp2

    mov bx, '$'
    mov [di], bx

    pop bx
    pop dx
    pop cx
    pop ax
    ret
Int_to_dec_str ENDP

Dec_str_to_int proc FAR
    push di
    push cx
    push bx
    push dx

    mov di, offset DEC_STR
    mov dx, '$'

    xor bx,bx
len:
    cmp [di+bx], dx
    je en

```

```

    inc bx
    jmp len
en:
    mov cx, bx

    mov bx, 10
    mov dx, 0

lp_1:
    mul bx
    mov dl, [di]
    sub dl, '0'
    add al, dl
    inc di
    loop lp_1

    mov di, offset N
    mov dx, [di]
    cmp dx, 0
    je pos_num

    neg ax

pos_num:

    pop dx
    pop bx
    pop cx
    pop di
    ret
Dec_str_to_int endp

MAIN PROC FAR
    push DS
    xor ax, ax
    push ax
    mov ax, DATA
    mov ds, ax

    mov dx, offset DECIM
    call WriteMsg

    mov ax, NUMBER

```

```

    mov di, offset SIGN
    mov bx, "+"
    cmp ax, 0
    jge set_sign
    mov bx, "-"
    neg ax
    push bx
    mov bx, 1
    mov N, bx
    pop bx

set_sign:
    mov [di], bx
    inc di
    mov bx, '$'
    mov [di], bx

    push ax
    mov dx, offset SIGN
    call WriteMsg
    pop ax

    call Int_to_dec_str
    push ax
    mov dx, offset DEC_STR
    call WriteMsg
    pop ax

    xor ax, ax
    call Dec_str_to_int
    ret
MAIN ENDP
CODE ENDS
    END MAIN

```