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

# ОТЧЕТ

**по лабораторной работе №7**

**по дисциплине «Организация ЭВМ и систем»**

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

Студент гр. 9383 _____ Арутюнян С.Н.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2020

# Содержание

## 1. Цель работы

Разработать на языке Ассемблер процессора IntelX86 две процедуры:

• одна – выполняет прямое преобразование 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);

• другая - обратное преобразование строки, представляющей символьное изображение числа в заданной системе счисления в целое число, помещаемое в регистр AX ( или в пару регистров DX:AX)

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

## 2. Задание. Вариант 2.1.1

Число — 32 битное, с учетом знака в двоичной системе. Взаимодействие между главной функцией и процедурами происходит через РОНы и общедоступные переменные (в моем случае - STRING_REPR).

## 3. Тестирование

На вход подается число dx:ax = 7000:0001:

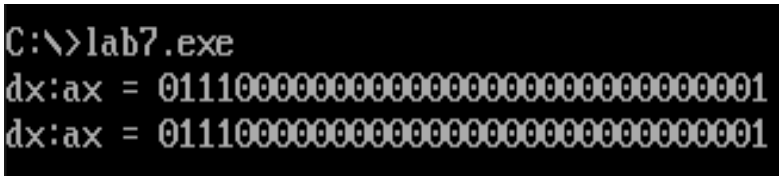


Рис. 1. Пример работы программы

На вход подается число dx:ax = f000:0002. Первая строка в прямом коде, вторая строка в дополнительном коде:


Рис. 2. Пример работы программы

## 3. Текст программы lab7.asm

```
AStack SEGMENT STACK
    dw 256 DUP(?)   ; 1 килобайт
AStack ENDS


DATA SEGMENT

STRING_REPR db "00000000000000000000000000000000", 0ah, '$'  ; 32
символов для 32 бит
DXAX_STRING db "dx:ax = $"
ZERO_SYMBOL EQU '0'
ONE_SYMBOL EQU '1'
BITS_NUMBER EQU 32

DATA ENDS


CODE SEGMENT
    ASSUME cs:CODE, ds:DATA, ss:AStack


Main PROC FAR

    mov ax, DATA
    mov ds, ax

    mov dx, 07000h
    mov ax, 1
    call int_to_string

    mov ah, 09h
    mov dx, offset DXAX_STRING
    int 21h
    mov dx, offset STRING_REPR
    int 21h
```

```
    mov di, offset STRING_REPR
    call string_to_int

    ; теперь в dx:ax число из STRING_REPR
    call int_to_string
    mov ah, 09h
    mov dx, offset DXAX_STRING
    int 21h
    mov dx, offset STRING_REPR
    int 21h



    mov ah, 4ch
    int 21h

Main ENDP

int_to_string proc near

    push ax
    push dx
    push bx
    push cx
    push di

    ; если dx:ax >= 0, то доп.код совпадает с прямым
    mov bx, dx
    mov cl, 15
    shr bx, cl
    cmp bx, 1
    jne init_vars

    ; ===========
    ; в ax - часть доп.кода => отнимаем единицу до обратного кода
    sub ax, 1
```

```
; если произошелся заем в старший значащий бит, то cf = 1
; если cf = 1, то нужно отнять 1 еще и из dx
jnc end_of_ready
sub dx, 1

end_of_ready:
    ; теперь инвертируем все биты кроме первого до прямого кода
    xor ax, 0ffffh
    xor dx, 07fffh
; ===========

init_vars:
    mov di, offset STRING_REPR
    mov ch, 32 ; просто счетчик

restart:
    mov cl, 16

write_loop:
    dec ch
    dec cl

    cmp ch, 15
    jle mov_ax

    mov bx, dx
    jmp continue

mov_ax:
    mov bx, ax

continue:
    shr bx, cl
    and bx, 1
    cmp bx, 1
    je one_write
```

```
    zero_write:
        mov byte ptr [di], ZERO_SYMBOL
        jmp end_loop

    one_write:
        mov byte ptr [di], ONE_SYMBOL

    end_loop:
        inc di
        cmp ch, 16
        je restart
        cmp ch, 0
        jne write_loop


    pop di
    pop cx
    pop bx
    pop dx
    pop ax

    ret

int_to_string endp


; указатель на строку в di, возврат числа в ax
string_to_int proc near

    push di
    push cx
    push bx

    xor ax, ax
    xor dx, dx
```

```
mov ch, BITS_NUMBER

restart_2:
    mov cl, 16

for_loop:
    dec ch
    dec cl

    mov bl, [di]
    cmp bl, ONE_SYMBOL
    jne loop_end

one_read:
    mov bx, 1
    shl bx, cl

    cmp ch, 15
    jle to_ax

    add dx, bx
    jmp loop_end

to_ax:
    add ax, bx

loop_end:
    inc di
    cmp ch, 16
    je restart_2
    cmp ch, 0
    jne for_loop


pop bx
pop cx
```

```
    pop di

    ret

string_to_int endp


CODE ENDS
END Main
```

**Выводы**

В процессе выполнения лабораторной работы была изучена работа с арифметическими операциями над целыми числами.