

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ «ЛЭТИ»
ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)»
Кафедра МО ЭВМ**

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

Студент гр. 1303

Преподаватель

Бутыло Е.А.

Ефремов М.А.

Санкт-Петербург

2022

Цель работы.

Разработать программу, реализующую преобразование 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.

Представление числа: с учетом знака. Система счисления для изображения числа — десятичная. Связь данных между основной программой и подпрограммами осуществляется через сегмент стека.

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

Выполнение работы.

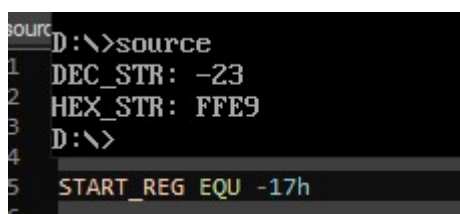
С помощью стека передаём значение регистра AX, внесённое в процедуру Main, в процедуру CAST_TO_DEC. Данная процедура реализует проверку числа на знак в случае, если число окажется отрицательным, то в заранее отведённую строку DEC_STR добавится символ минуса, иначе выполняется деление числа на 10, соответствующее сохранение остатков в стеке. После чего извлекаются остатки и записываются в строку DEC_STR, также добавляется символ конца строки и выход из процедуры. Далее строка DEC_STR печатается на экран.

Далее DEC_STR передаем в процедуру GO_TO_REG с помощью стека. В процедуре GO_TO_REG сперва идет проверка на знак минус, если он есть, то в DI помещаем соответствующий флаг. Далее в BX передаем систему счисления, после чего идет проверка на конец строки. После чего символ преобразуем в 16-ю систему и записываем в регистр AX.

Если в DI был флаг 1, то берется противоположное со знаком число в

АХ. После окончания процедуры переход в процедуру CAST_TO_HEX. В процедуре CAST_TO_HEX в CL заносится количество бит для считывания из строки, то есть вывод по 4 бита. Далее в AL заносится цифра в соответствии с 16-ой системой, после чего в AL получаем символ цифры и происходит проверка на цифру, если это цифра, то переходим к Digit, где записываем в строку и если CL еще больше или равно 0, то повторяем действия. После окончания процедуры выводим строку HEX_STR на экран.

Проверка работы программы.



```
D:\>source
1 DEC_STR: -23
2 HEX_STR: FFE9
3 D:\>
4
5 START_REG EQU -17h
6
```

Рис. 1 – корректное выполнение программы.

Выводы

Разработана программа преобразования число из 16-ой системы в 10-ую систему в строчном представлении, а также из строкового вида 10-ой системы в строковое представление 16-ой системы.

ПРИЛОЖЕНИЕ А

КОД ПРОГРАММ

Название файла: source.asm

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

START_REG EQU -17h

DATA    SEGMENT
        HEX_NAME DB 'HEX_STR: ', '$'
        DEC_NAME  DB 'DEC_STR: ', '$'
        DEC_STR  DB ' ', '$'
        HEX_STR  DB ' ', '$'
DATA     ENDS

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

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

CAST_TO_DEC  proc
        pop cx
        pop di
        pop ax

        push  bx
        push  cx
        push  dx

        mov   bx,      10
        xor   cx,      cx
        or    ax,      ax
        jns   div1
        neg   ax
        push  ax
        mov   dl,      '-'
        mov [di], dl
        inc di
        pop   ax
div1:
        xor   dx,      dx
        div   bx
        push  dx
        inc   cx
        or    ax,      ax
        jnz   div1
sto:
        pop   dx
        add   dl,      '0'
        mov [di], dl
        inc di
```

```

        loop    sto

    mov dl, '$'
    mov [di], dl
    inc di

    pop    dx
    pop    cx
    pop    bx

    push cx
    ret
CAST_TO_DEC    endp

GO_TO_REG proc
    pop cx
    pop dx

    push cx
    push si
    push di

    mov si, OFFSET DEC_STR
    cmp byte ptr [si], "-"
    jnz l1
    mov di, 1
    inc si
l1:
        xor ax, ax
        mov bx, 10
l2:
        mov cl, [si]
        cmp cl, '$'
        jz endin

        sub cl, '0'
        mul bx
        add ax, cx
        inc si
        jmp l2
    endin:
        cmp di, 1
        jnz l3
        neg ax
l3:
        pop di
        pop si
        pop cx

        push ax
        push cx

    ret
GO_TO_REG endp

CAST_TO_HEX proc
    pop cx
    pop di
    pop ax

```

```

push    cx
push    dx

mov     cl,      ((16-1)/4)*4
xchg    dx,      ax

```

Repeat:

```

mov     ax,      dx
shr     ax,      cl
and     al,      0Fh
add     al,      '0'
cmp     al,      '9'
jbe     Digit
add     al,      'A'-'9'+1)

```

Digit:

```

push    dx
mov     dl, al
mov     [di], dl
inc     di
pop     dx
sub     cl, 4
jnc     Repeat

```

```

mov     dl, '$'
mov     [di], dl
inc     di

```

```

pop     dx

```

```

ret

```

CAST_TO_HEX endp

Main PROC FAR

```

push    ds
sub     ax, ax
push    ax
mov     ax, DATA
mov     ds, ax

```

```

mov     ax, START_REG
push    ax
mov     di, OFFSET DEC_STR
push    di
call    CAST_TO_DEC

```

```

mov     dx, OFFSET DEC_NAME
call    WriteMsg
mov     dx, OFFSET DEC_STR
call    WriteMsg

```

```

push    dx
push    ax
mov     ah, 2
mov     dl, 10
int     21h
pop     ax

```

```
    pop dx

    mov dx, OFFSET DEC_STR
    push dx
    call GO_TO_REG

    mov di, OFFSET HEX_STR
    push di
    call CAST_TO_HEX

    mov dx, OFFSET HEX_NAME
    call WriteMsg
    mov dx, OFFSET HEX_STR
    call WriteMsg

    ret
Main ENDP
CODE ENDS
END Main
```