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.

Студент гр.1381 _____ Сагидуллин Э.Р.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2022

Цель работы.

Разработать программу, которая переводит число в строку из одной системы счисления в другую.

Задание.

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. 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 19.

16 битное число без учета знака, в восьмиричной системе системе счисления, вызов – far, связь между процедурами – РОН.

Выполнение работы.

Были реализованы четыре процедуры: WriteMsg для вывода строки на экран. convert_to_eight_str для преобразования числа в восьмиричную систему. Основная идея состоит в том, что в bx заносится делитель 8 и в цикле происходит деление числа на bx, остатки деления сохраняются в стеке. После цикла остатки достаются из стека, к ним прибавляется код ‘0’ чтобы получить код нужной цифры и заносятся в строку eight_str. Благодаря этому остатки записываются в обратном порядке. В конце добавляется символ окончания строки $. eight_str_convert для преобразования числа из восьмиричной системы в десятичную. Подсчитывается длина строки eight_str и вызывается цикл (количество итераций определяется длиной строки) где из каждой цифры строки вычитается код символа ‘0’, она умножается на 8 и добавляется к результату. MAIN — основная процедура, где происходит вызов остальных процедур. Здесь проверяется знак числа, если оно отрицательно, число меняет знак с помощью команды neg.

Выводы.

В ходе лабораторной работы разработана программа, которая переводит число в заданную систему счисления и наоборот.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab7.asm

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

DATA    SEGMENT
    n DW 0
    eight_str DB 10, 13, ' ', '$'
    NUMBER DW -8
DATA    ENDS

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


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



convert_to_eight_str proc FAR
    push ax
    push cx
    push dx
    push bx
    xor cx,cx
    mov bx,8
    mov di, offset eight_str

loop1:
    xor dx,dx
    div bx
    add dl,'0'
    push dx
    inc cx
    test ax,ax
    jnz loop1

loop2:
    pop dx
    mov [di],dl
    inc di
    loop loop2
```

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

    pop bx
    pop dx
    pop cx
    pop ax
    ret
convert_to_eight_str ENDP


eight_str_convert proc FAR
    push di
    push cx
    push bx
    push dx

    mov di, offset eight_str
    mov dx, '$'

    xor bx,bx
    metka1:
        cmp [di+bx], dx
        je metka2
        inc bx
        jmp metka1

    metka2:
        mov cx, bx

    mov bx, 8
    mov dx, 0

    loop3:
        mul bx
        mov dl, [di]
        sub dl, '0'
        add al, dl
        inc di
    loop loop3

   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
eight_str_convert endp


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

    mov ax, NUMBER
    cmp ax, 0
    jge skip
    neg ax

    skip:
        call convert_to_eight_str
        push ax
        mov dx, offset eight_str
        call WriteMsg
        pop ax
        xor ax, ax
        call eight_str_convert
        ret
MAIN ENDP
CODE ENDS
    END MAIN
```