

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

ОТЧЕТ
по лабораторной работе №7
по дисциплине «Организация ЭВМ и систем»
ТЕМА: ПРЕОБРАЗОВАНИЕ ЦЕЛЫХ ЧИСЕЛ. ИСПОЛЬЗОВАНИЕ ПРОЦЕДУР В
АССЕМБЛЕРЕ.

Студент гр. 1303

Беззубов Д.В.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2022

Цель работы.

Получить навыки программирования на языке Ассемблера. Изучить работу с целыми числами с использованием процедур на языке Ассемблера.

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:

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

Выполнение работы.

В главной процедуре MAIN происходит запись в регистр ах исходного числа (NUMBER в сегменте данных). Далее проверяется знак числа, если число положительное, то в SIGN кладется знак '+', если отрицательное, то '-'. Это необходимо для вывода символьного представления числа. Затем с помощью процедуры Int_to_dec_str преобразовываем исходное число в строку (в десятичной системе счисления) и записываем это значение в DEC_STR. С помощью процедуры WriteMsg выводим. После того, как получено строковое представление числа, обнулим регистр AX. В него запишем число после обратного преобразования. Затем с помощью процедуры Dec_str_to_int

получаем из строки число. Тот результат, который мы записали 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).


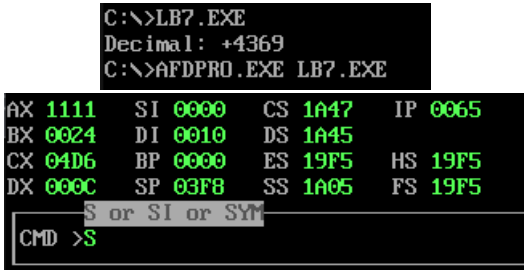
Процедура Dec_str_to_int: циклом подсчитываем длину числа. Затем значение, лежащее в регистре AX (в начале 0, т.к. мы обнулили его в MAIN), умножается на 10, и к нему добавляется разность кодов символов из записи числа и '0'. Данные действия осуществляются в цикле, пока не пройдем всю строку. В конце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


Исходный код программы см. в приложении А.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	0001h		
2.	1111h		

3.	0FFF1h		
----	--------	--	--

Выводы.

В ходе выполнения лабораторной работы были получены навыки программирования на языке Ассемблера. Была разработана программа переводящая число в 10-ричную систему в строковом виде и обратно.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название 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 0FFF1h
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                ;Обнуление CX
        mov bx,10                ;В BX делитель (10 для десятичной
системы)
        mov di, offset DEC_STR

lp1:                ;Цикл получения остатков от деления
        xor dx,dx                ;Обнуление старшей части двойного
слова
        div bx                    ;Деление AX=(DX:AX)/BX, остаток в DX
        add dl,'0'                ;Преобразование остатка в код символа
        push dx                    ;Сохранение в стеке
        inc cx                    ;Увеличение счетчика символов
        test ax,ax                ;Проверка AX
        jnz lp1                ;Переход к началу цикла, если частное не 0.

lp2:                ;Цикл извлечения символов из стека
        pop dx                    ;Восстановление символа из стека
        mov [di],dl                ;Сохранение символа в буфере
        inc di                    ;Инкремент адреса буфера
```

```

        loop lp2                ;Команда цикла

        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
```