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. 9382 _____ Павлов Р.В.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Разработка процедур перевода из строки в число и из числа в строку.

Задание (вариант 2).

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.

Ход работы.

1) В отдельном кодовом сегменте написана процедура считывания строки (двоичного представления числа) с клавиатуры и перевода её в число, которая затем через стек возвращает два слова — значения регистров DX и AX.

2) В основном сегменте кода написана процедура, посредством побитового сдвига определяющая каждый последующий бит числа и заносящая соответствующий ему символ в место в памяти, где хранится результирующая строка.

3) Написана главная процедура, которая вызывает сначала первую, а потом вторую процедуры. Затем выводится результирующая строка. Цель — демонстрация работы программы.

Выводы.

В результате выполнения лабораторной работы написана программа, преобразующая строку в число и наоборот.

ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД

- имя файла : lab7.asm

```
stack segment stack
      dw 64 dup(0)
stack ends

data segment
      origin db 33, ?, 33 dup(0)
      result db 33 dup(0)
data ends

additional segment
      assume ds:data, cs:additional
      strToInt proc far

            push ax
            mov ax, data
            mov ds, ax
            pop ax

            xor cx, cx
            mov ah,0ah
            mov dx,offset origin          ; Считывание строки и запись её в
буфер, перевод на новую строку
            int 21h
            mov dl,0ah
            mov ah,02
            int 21h

            mov si,offset origin+2

            xor ax,ax                     ; Готовим регистры для записи: ax =
0, dx = 0, bx = 2 - основание СС
            xor dx, dx
            mov bx,2
```

```
mov cl, origin[1]

transformdx:
cmp cx, 17                    ; Расчёт двух старших байтов
jl sec_word

push cx

mov cl, [si]
cmp cl,'0'                    ; Проверка на соответствие цифре
jb err
cmp cl,'1'
ja err

sub cl,'0'                    ; Перевод из кода символа в цифру,
домножение на 10, прибавление в конец
mul bx
add ax,cx
inc si

pop cx

loop transformdx

sec_word:                     ; Расчёт двух младших байтов
push ax
xor ax, ax

transformax:
mov cl,[si]                   ; Проверка на последний символ
cmp cl,0dh
jz fin

cmp cl,'0'                    ; Проверка на соответствие цифре
jb err
cmp cl,'1'
ja err
```

```
            sub cl,'0'                    ; Перевод из кода символа в цифру,
домножение на 2, прибавление в конец
            mul bx
            add ax,cx
            inc si
            jmp transformax

            err:
            mov dx, offset error    ; Ошибка (если не цифра), выход
            mov ah,09
            int 21h
            int 20h

            fin:
            pop dx
            pop cx
            pop bx

            push ax                              ; Помещение числа в стек
            push dx

            push bx
            push cx
            ret

      error db "incorrect number$"
      strToInt endp
additional ends

code segment
      assume ds:data, cs:code, ss:stack
      intToStr proc near
            push ax
            push bx
            push cx
            push dx
            push di
```

```
            lea di, result                ; Переход в конец строки, запись
символа конца строки
            add di, 33
            mov cl, '$'
            mov [di], cl
            dec di

            mov cx, 16

            shiftax:
                  shr ax, 1
                  jc setax
                  mov ch, 48
                  jmp recax

                  setax:
                        mov ch, 49        ; Сдвиг вправо, заполнение первых 16
знаков

                  recax:
                  mov [di], ch
                  dec di
                  and cx, 00FFh
                  loop shiftax

            mov cx, 16

            shiftdx:
                  shr dx, 1
                  jc setdx
                  mov ch, 48
                  jmp recdx

                  setdx:
                        mov ch, 49        ; Сдвиг вправо, заполнение последних
16 знаков

                  recdx:
                  mov [di], ch
```

```
                dec di
                and cx, 00FFh
                loop shiftdx

        pop di
        pop dx
        pop cx
        pop bx
        pop ax
        ret
    intToStr endp

    main proc far
        xor ax, ax
        push ds
        push ax

        mov ax, offset data
        mov ds, ax

        call es:strToInt        ; Ввод числа
        assume cs:code

        pop dx                          ; Получение числа из стека
        pop ax

        call intToStr                ; Запись числа в виде строки

        mov dx, offset result
        mov ah, 9                    ; Вывод результата на экран
        int 21h

        ret
    main endp
code ends
end main
```

ПРИЛОЖЕНИЕ Б. ТЕКСТ ЛИСТИНГОВ

- имя файла: lab5.lst

```
#Microsoft (R) Macro Assembler Version 5.10                 12/7/20 16:14:32
                                                            Page     1-1


 0000                   stack segment stack
 0000  0040[                            dw 64 dup(0)
         0000
                    ]

 0080                   stack ends

 0000                   data segment
 0000  21 00                            origin db 33, ?, 33 dup(0)
       0021[
         00
                    ]

 0023  0021[                            result db 33 dup(0)
         00
                    ]

 0044                   data ends

 0000                   additional segment
                              assume ds:data, cs:additional
 0000                         strToInt proc far

 0000  50                               push ax
 0001  B8 ---- R                        mov ax, data
 0004  8E D8                                  mov ds, ax
 0006  58                               pop ax

 0007  33 C9                                  xor cx, cx
 0009  B4 0A                                  mov ah,0ah
 000B  BA 0000 R                        mov dx,offset origin
                              ; Считывание строки и
                        запись её в буфер, перевод
                         на новую строку
 000E  CD 21                                  int 21h
 0010  B2 0A                                  mov dl,0ah
 0012  B4 02                                  mov ah,02
 0014  CD 21                                  int 21h

 0016  BE 0002 R                        mov si,offset origin+2

 0019  33 C0                                  xor ax,ax
                              ; Готовим регистр
                        ы   для   записи:   ax   =   0   ,   bx   =   2   -   осЙ
                        ½ование СС
 001B  33 D2                                  xor dx, dx
 001D  BB 0002                                mov bx,2

 0020  8A 0E 0001 R                           mov cl, origin[1]

 0024                                   transformdx:
 0024  83 F9 11                               cmp cx, 17
```

```
0027  7C 18                             jl sec_word

#Microsoft (R) Macro Assembler Version 5.10                 12/7/20 16:14:32
                                                            Page     1-2


0029  51                          push cx

002A  8A 0C                             mov cl, [si]
002C  80 F9 30                          cmp cl,'0'
                                  ;         Проверка         на         сооэ
                        ⑫ветствие цифре
002F  72 2E                             jb err
0031  80 F9 31                          cmp cl,'1'
0034  77 29                             ja err

0036  80 E9 30                          sub cl,'0'
                                  ; Перевод из кода
                        символа           в          цифру,          домноженЙ
                        ¸е на 10, прибавление в коне
                        ц
0039  F7 E3                             mul bx
003B  03 C1                             add ax,cx
003D  46                          inc si

003E  59                          pop cx
003F  E2 E3                             loop transformdx

0041                              sec_word:
0041  50                          push ax
0042  33 C0                             xor ax, ax

0044                              transformax:
0044  8A 0C                             mov cl,[si]
                                  ;         Проверка         на         посЙ
                        »едний символ
0046  80 F9 0D                          cmp cl,0dh
0049  74 1D                             jz fin

004B  80 F9 30                          cmp cl,'0'
                                  ;         Проверка         на         сооэ
                        ⑫ветствие цифре
004E  72 0F                             jb err
0050  80 F9 31                          cmp cl,'1'
0053  77 0A                             ja err

0055  80 E9 30                          sub cl,'0'
                                  ; Перевод из кода
                        символа           в          цифру,          домноженЙ
                        ¸е      на      2,      прибавление      в      конеэ
                        ⑥
0058  F7 E3                             mul bx
005A  03 C1                             add ax,cx
005C  46                          inc si
005D  EB E5                             jmp transformax

005F                              err:
005F  BA 0070 R                   mov dx, offset error    ;               ОшЙ
                        ¸бка (если не цифра), выход
0062  B4 09                             mov ah,09
```

```
0064  CD 21                              int 21h
0066  CD 20                              int 20h

0068                                fin:
0068  5A                            pop dx
0069  59                            pop cx
006A  5B                            pop bx

006B  50                            push ax
006C  52                            push dx

006D  53                            push bx
006E  51                            push cx
006F  CB                            ret

0070  69 6E 63 6F 72 72            error db "incorrect number$"
      65 63 74 20 6E 75
      6D 62 65 72 24
0081                          strToInt endp
0081                    additional ends

0000                    code segment
                              assume ds:data, cs:code, ss:stack
0000                          intToStr proc near
0000  50                            push ax
0001  53                            push bx
0002  51                            push cx
0003  52                            push dx
0004  57                            push di

0005  8D 3E 0023 R                       lea di, result
0009  83 C7 21                           add di, 33
000C  B1 24                              mov cl, '$'
000E  88 0D                              mov [di], cl
0010  4F                            dec di

0011  B9 0010                            mov cx, 16

0014                                shiftax:
0014  D1 E8                                    shr ax, 1
0016  72 05                                    jc setax
0018  B5 30                                    mov ch, 48
001A  EB 03 90                                 jmp recax

001D                                      setax:
001D  B5 31                                          mov ch, 49

001F                                      recax:
001F  88 2D                                    mov [di], ch
0021  4F                                  dec di
0022  81 E1 00FF                          and cx, 00FFh
0026  E2 EC                                    loop shiftax

0028  B9 0010                             mov cx, 16
```

```
 002B                                shiftdx:
 002B  D1 EA                                 shr dx, 1
 002D  72 05                                 jc setdx
 002F  B5 30                                 mov ch, 48
 0031  EB 03 90                              jmp recdx

 0034                                setdx:
 0034  B5 31                                         mov ch, 49

 0036                                recdx:
 0036  88 2D                                 mov [di], ch
 0038  4F                                    dec di
 0039  81 E1 00FF                           and cx, 00FFh
 003D  E2 EC                                 loop shiftdx

 003F  5F                            pop di
 0040  5A                            pop dx
 0041  59                            pop cx
 0042  5B                            pop bx
 0043  58                            pop ax
 0044  C3                            ret
 0045                          intToStr endp

 0045                          main proc far
 0045  33 C0                                 xor ax, ax
 0047  1E                            push ds
 0048  50                            push ax

 0049  B8 0044 R                     mov ax, offset data
 004C  8E D8                                 mov ds, ax

 004E  9A 0000 ---- R                        call es:strToInt
                                     assume cs:code

 0053  5A                            pop dx
 0054  58                            pop ax

 0055  E8 0000 R                     call intToStr

 0058  BA 0023 R                     mov dx, offset result
 005B  B4 09                                 mov ah, 9
 005D  CD 21                                 int 21h

 005F  CB                            ret
 0060                          main endp
 0060                  code ends
                       end main
```

#Microsoft (R) Macro Assembler Version 5.10 12/7/20 16:14:32
Symbols-1

Segments and Groups:

N a m e	Length	Align	Combine Class
ADDITIONAL	0081	PARA	NONE

```
CODE . . . . . .    0060  PARA  NONE
DATA . . . . . .    0044  PARA  NONE
STACK  . . . . . .  0080  PARA  STACK

Symbols:

                N a m e             Type   Value       Attr

ERR  . . . . . .    L NEAR      005F  ADDITIONAL
ERROR  . . . . . .  L BYTE      0070  ADDITIONAL

FIN  . . . . . .    L NEAR      0068  ADDITIONAL

INTTOSTR . . . . . .  N PROC     0000  CODE  Length = 0045

MAIN . . . . . .    F PROC      0045  CODE  Length = 001B

ORIGIN . . . . . .  L BYTE      0000  DATA

RECAX  . . . . . .  L NEAR      001F  CODE
RECDX  . . . . . .  L NEAR      0036  CODE
RESULT . . . . . .  L BYTE      0023  DATA  Length = 0021

SEC_WORD . . . . . .  L NEAR    0041  ADDITIONAL
SETAX  . . . . . .  L NEAR      001D  CODE
SETDX  . . . . . .  L NEAR      0034  CODE
SHIFTAX  . . . . . .  L NEAR    0014  CODE
SHIFTDX  . . . . . .  L NEAR    002B  CODE
STRTOINT . . . . . .  F PROC    0000  ADDITIONAL  Length     =
0081

TRANSFORMAX  . . . . . .  L NEAR  0044  ADDITIONAL
TRANSFORMDX  . . . . . .  L NEAR  0024  ADDITIONAL

@CPU . . . . . .    TEXT  0101h
@FILENAME  . . . . . .  TEXT  lab7
@VERSION . . . . . .  TEXT  510


    177 Source  Lines
    177 Total   Lines
     26 Symbols

  47900 + 457310 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors
```