```
;------------------------------------------------------------
;String length procedure. Receives a string pointer
;(seg:offset) via the stack. If not a string, CF is set;
;otherwise, string length is returned in AX with CF = 0.
;Preserves all registers.
;------------------------------------------------------------
str_len PROC
        push    BP
        mov     BP,SP
        push    CX
        push    DI
        push    ES
        les     DI,STRING1 ; copy string pointer to ES:DI
        mov     CX,STR_MAX ; needed to terminate loop if BX
                           ;  is not pointing to a string
        cld                ; forward search
        mov     AL,0       ; NULL character
        repne   scasb
        jcxz    sl_no_string  ; if CX = 0, not a string
        dec     DI         ; back up to point to NULL
        mov     AX,DI
        sub     AX,[BP+4]  ; string length in AX
        clc                ; no error
        jmp     SHORT sl_done
sl_no_string:
        stc                ; carry set => no string
sl_done:
        pop     ES
        pop     DI
        pop     CX
        pop     BP
        ret     4          ; clear stack and return
str_len ENDP
```

```
;------------------------------------------------------------
;String copy procedure. Receives two string pointers
;(seg:offset) via the stack - string1 and string2.
;If string2 is not a string, CF is set;
;otherwise, string2 is copied to string1 and the
;offeset of string1 is returned in AX with CF = 0.
;Preserves all registers.
;------------------------------------------------------------
str_cpy PROC
        push    BP
        mov     BP,SP
        push    CX
        push    DI
        push    SI
        push    DS
        push    ES
        ; find string length first
        lds     SI,STRING2  ; source string pointer
        push    DS
        push    SI
        call    str_len
        jc      sc_no_string
        mov     CX,AX       ; source string length in CX
        inc     CX          ; add 1 to include NULL
        les     DI,STRING1 ; dest. string pointer
        cld                 ; forward copy
        rep     movsb
        mov     AX,[BP+4]  ; return dest. string pointer
        clc                 ; no error
        jmp     SHORT sc_done
sc_no_string:
        stc                 ; carry set => no string
sc_done:
        pop     ES
        pop     DS
        pop     SI
        pop     DI
        pop     CX
        pop     BP
        ret     8           ; clear stack and return
str_cpy ENDP
```

```
;-----------------------------------------------------------
;String concatenate procedure. Receives two string pointers
;(seg:offset) via the stack - string1 and string2.
;If string1 and/or string2 are not strings, CF is set;
;otherwise, string2 is concatenated to the end of string1
;and the offset of string1 is returned in AX with CF = 0.
;Preserves all registers.
;-----------------------------------------------------------
str_cat PROC
        push    BP
        mov     BP,SP
        push    CX
        push    DI
        push    SI
        push    DS
        push    ES
        ; find string length first
        les     DI,STRING1 ; dest. string pointer
        mov     CX,STR_MAX ; max string length
        cld                ; forward search
        mov     AL,0       ; NULL character
        repne   scasb
        jcxz    st_no_string
        dec     DI          ; back up to point to NULL
        lds     SI,STRING2 ; source string pointer
        push    DS
        push    SI
        call    str_len
        jc      st_no_string

        mov     CX,AX       ; source string length in CX
        inc     CX          ; add 1 to include NULL
        cld                 ; forward copy
        rep     movsb
        mov     AX,[BP+4]   ; return dest. string pointer
        clc                 ; no error
        jmp     SHORT st_done
st_no_string:
        stc                 ; carry set => no string
st_done:
        pop     ES
        pop     DS
        pop     SI
        pop     DI
        pop     CX
        pop     BP
        ret     8           ; clear stack and return
str_cat ENDP
```

```
;------------------------------------------------------------
;String compare procedure. Receives two string pointers
;(seg:offset) via the stack - string1 and string2.
;If string2 is not a string, CF is set;
;otherwise, string1 and string2 are compared and returns a
;a value in AX with CF = 0 as shown below:
;    AX = negative value  if string1 < string2
;    AX = zero            if string1 = string2
;    AX = positive value  if string1 > string2
;Preserves all registers.
;------------------------------------------------------------
str_cmp PROC
        push    BP
        mov     BP,SP
        push    CX
        push    DI
        push    SI
        push    DS
        push    ES
        ; find string length first
        les     DI,STRING2  ; string2 pointer
        push    ES
        push    DI
        call    str_len
        jc      sm_no_string
        mov     CX,AX       ; string1 length in CX
        inc     CX          ; add 1 to include NULL
        lds     SI,STRING1 ; string1 pointer
        cld                 ; forward comparison
        repe    cmpsb
        je      same
        ja      above
below:
        mov     AX,-1       ; AX = -1 => string1 < string2
        clc
        jmp     SHORT sm_done
same:
        xor     AX,AX       ; AX = 0 => string match
        clc
        jmp     SHORT sm_done
above:
        mov     AX,1        ; AX = 1 => string1 > string2
        clc
        jmp     SHORT sm_done
sm_no_string:
        stc                 ; carry set => no string
sm_done:
```

```
sm_done:
        pop     ES
        pop     DS
        pop     SI
        pop     DI
        pop     CX
        pop     BP
        ret     8           ; clear and return
str_cmp ENDP
```

```
;------------------------------------------------------------
;String locate a character procedure. Receives a character
;and a string pointer (seg:offset) via the stack.
;char should be passed as a 16-bit word.
;If string1 is not a string, CF is set;
;otherwise, locates the first occurrence of char in string1
;and returns a pointer to the located char in AX (if the
;search is successful; otherwise AX = NULL) with CF = 0.
;Preserves all registers.
;------------------------------------------------------------
str_chr PROC
        push    BP
        mov     BP,SP
        push    CX
        push    DI
        push    ES
        ; find string length first
        les     DI,STRING1  ; source string pointer
        push    ES
        push    DI
        call    str_len
        jc      sh_no_string
        mov     CX,AX       ; source string length in CX
        inc     CX
        mov     AX,[BP+8]   ; read char. into AL
        cld                 ; forward search
        repne   scasb
        dec     DI          ; back up to match char.
        xor     AX,AX       ; assume no char match (AX=NULL)
        jcxz    sh_skip
        mov     AX,DI       ; return pointer to char.
sh_skip:
        clc                 ; no error
        jmp     SHORT sh_done
sh_no_string:
        stc                 ; carry set => no string
sh_done:
        pop     ES
        pop     DI
        pop     CX
        pop     BP
        ret     6           ; clear stack and return
str_chr ENDP
```

```
;------------------------------------------------------------
;String convert procedure. Receives two string pointers
;(seg:offset) via the stack - string1 and string2.
;If string2 is not a string, CF is set;
;otherwise, string2 is copied to string1 and lowercase
;letters are converted to corresponding uppercase letters.
;string2 is not modified in any way.
;It returns a pointer to string1 in AX with CF = 0.
;Preserves all registers.
;------------------------------------------------------------
str_cnv PROC
        push    BP
        mov     BP,SP
        push    CX
        push    DI
        push    SI
        push    DS
        push    ES
        ; find string length first
        lds     SI,STRING2  ; source string pointer
        push    DS
        push    SI
        call    str_len
        jc      sn_no_string
        mov     CX,AX       ; source string length in CX
        inc     CX          ; add 1 to include NULL
        les     DI,STRING1  ; dest. string pointer
        cld                 ; forward search
loop1:
        lodsb
        cmp     AL,'a'      ; lowercase letter?
        jb      sn_skip
        cmp     AL,'z'
        ja      sn_skip     ; if no, skip conversion
        sub     AL,20H      ; if yes, convert to uppercase
sn_skip:
        stosb
        loop    loop1
        rep     movsb
        mov     AX,[BP+4]   ; return dest. string pointer
        clc                 ; no error
        jmp     SHORT sn_done
sn_no_string:
        stc                 ; carry set => no string
sn_done:
```

```
         pop     ES
         pop     DS
         pop     SI
         pop     DI
         pop     CX
         pop     BP
         ret     8           ; clear stack and return
str_cnv ENDP
```

```
;------------------------------------------------------------
;String move procedure. Receives a signed integer
;and a string pointer (seg:offset) via the stack.
;The integer indicates the number of positions to move
;the string:
;      -ve number => left move
;      +ve number => right move
;If string1 is not a string, CF is set;
;otherwise, string is moved left or right and returns
;a pointer to the modified string in AX with CF = 0.
;Preserves all registers.
;------------------------------------------------------------
str_mov PROC
        push    BP
        mov     BP,SP
        push    CX
        push    DI
        push    SI
        push    DS
        push    ES
        ; find string length first
        lds     SI,STRING1  ; string pointer
        push    DS
        push    SI
        call    str_len
        jnc     sv_skip1
        jmp     sv_no_string
sv_skip1:
        mov     CX,AX       ; string length in CX
        inc     CX          ; add 1 to include NULL
        les     DI,STRING1
        mov     AX,[BP+8]   ; copy # of positions to move
        cmp     AX,0        ; -ve number => left move
        jl      move_left   ; +ve number => right move
        je      finish      ; zero => no move
move_right:
        ; prepare SI and DI for backward copy
        add     SI,CX       ; SI points to the
        dec     SI          ;  NULL character
        mov     DI,SI       ; DI = SI + # of positions to move
        add     DI,AX
        std                 ; backward copy
        rep     movsb
        ; now erase the remainder of the old string
        ;  by writing blanks
        mov     CX,[BP+8]   ; # of positions moved
        ; DI points to the first char of left-over string
        mov     AL,' '      ; blank char to fill
        ; direction flag is set previously
        rep     stosb
        jmp     SHORT finish
move_left:
        add     DI,AX
        cld                   ; forward copy
        rep     movsb
finish:
        mov     AX,[BP+8]     ; add # of positions to move
        add     AX,[BP+4]     ; to string pointer (ret value)
        clc                   ; no error
        jmp     SHORT sv_done
sv_no_string:
        stc                   ; carry set => no string
sv_done:
        pop     ES
        pop     DS
        pop     SI
        pop     DI
        pop     CX
        pop     BP
        ret     6             ; clear stack and return
str_mov ENDP
```

```
                    . . .
.DATA
proc_ptr_table  DW  str_len_fun,str_cpy_fun,str_cat_fun
                DW  str_cmp_fun,str_chr_fun,str_cnv_fun
                DW  str_mov_fun
MAX_FUNCTIONS   EQU ($ - proc_ptr_table)/2

choice_prompt   DB  'You can test several functions.',CR,LF
                DB  '    To test       enter',CR,LF
                DB  'String length       1',CR,LF
                DB  'String copy         2',CR,LF
                DB  'String concatenate  3',CR,LF
                DB  'String compare      4',CR,LF
                DB  'Locate character    5',CR,LF
                DB  'Convert string      6',CR,LF
                DB  'Move string         7',CR,LF
                DB  'Invalid response terminates program.',CR,LF
                DB  'Please enter your choice: ',0

invalid_choice  DB  'Invalid chioce - program terminates.',0

string1         DB  STR_MAX DUP (?)
string2         DB  STR_MAX DUP (?)
                    . . .
```

```
main    PROC
        .STARTUP
        mov     AX,DS
        mov     ES,AX
query_choice:
        xor     BX,BX
        PutStr  choice_prompt    ; display menu
        GetCh   BL               ; read response
        nwln
        sub     BL,'1'
        cmp     BL,0
        jb      invalid_response
        cmp     BL,MAX_FUNCTIONS
        jb      response_ok
invalid_response:
        PutStr  invalid_choice
        jmp     SHORT done
response_ok:
        shl     BL,1                ; multiply BL by 2
        call    proc_ptr_table[BX]  ; indirect call
        jmp     query_choice
done:
        .EXIT
main    ENDP
                  . . .
        END     main
```