Code(common.inc):

```
;Common macros and functions

;Push bx,cx,dx,si,di,bp; mov bp, sp
pushreg macro
  push bx
  push cx
  push dx
  push si
  push di
  push bp
  mov bp, sp
  endm pushreg

;mov sp, bp; pop bp,di,si,dx,cx,bx.
popreg macro
  mov sp, bp
  pop bp
  pop di
  pop si
  pop dx
  pop cx
  pop bx
  endm popreg

;Put a dollar-terminated string from ds:PrAddr.
put09H macro PrAddr
  push ax
  push dx
  mov ah, 09h
  lea dx, PrAddr
  int 21h
  pop dx
  pop ax
  endm putline

;Remember to put a newline after this.
;Use function 0AH using ds:buffer; put a '$' automatically at the end
read0AH macro buffer
  push ax
  push dx
  push bx
  mov ah, 0ah ;read buffered line
  lea dx, buffer
  int 21h

  lea bx, buffer
  inc bx
  xor ah, ah
  mov al, byte ptr [bx]
  inc bx
  add bx, ax
```

```
  mov byte ptr [bx], '$'

  pop bx
  pop dx
  pop ax
  endm read0AH

;Same as above but get a ASCIIZ or C-style string.
readCstr macro buffer
  push ax
  push dx
  push bx
  mov ah, 0ah ;read buffered line
  lea dx, buffer
  int 21h

  lea bx, buffer
  inc bx
  xor ah, ah
  mov al, byte ptr [bx]
  inc bx
  add bx, ax
  mov byte ptr [bx], 0

  pop bx
  pop dx
  pop ax
  endm readCstr

;Read dollar-terminated ASCII string from [si] and store as 16-bit unsigned integer in ax. Carry flag is cleared if no error, set if error. AX is undefined on error.
atou:
  pushreg
  xor ax, ax
  push ax ;now word ptr ss:[bp-2] is my integer.

atouLoop:
  mov al, [si]
  cmp al, '$'
  je atouLoopEnd
  cmp al, '0'
  jb atouError
  cmp al, '9'
  ja atouError
  sub al, '0' ;get digit
  xor ah, ah ;convert al to word ax; unsigned
  push ax ;store the digit
  mov ax, word ptr ss:[bp-2] ;load sum
  umul16 10 ;Multiply by 10
  mov word ptr ss:[bp-2], ax ;store sum*10
  pop ax ;get digit back
  add word ptr ss:[bp-2], ax ;store sum*10+d
```

```
  inc si
  jmp atouLoop
atouLoopEnd:
  mov ax, word ptr ss:[bp-2]
  popreg ;doesn't modify ax
  clc
  ret
atouError:
  popreg
  stc
  ret
  ;endp atou

utoa:
  pushreg
  cmp ax, 0
  jz utoaZero
  push ax; store our number temporarily at word ptr ss:[bp-2]

  xor cx, cx ;cx = count of the number of digits = length
utoaCount:
  udiv16 10 ;divide the number by 10
  inc di
  inc cx
  cmp ax, 0
  jnz utoaCount ;loop till quotient is zero, counting the digits

  mov byte ptr [di], '$' ;Terminator
  mov ax, word ptr ss:[bp-2] ;reload input integer
utoaConv:
  dec di
  udiv16 10 ;divide ax by 10; dl contains remainder
  add dl, '0'
  mov byte ptr [di], dl
  cmp ax, 0
  jnz utoaConv

  mov ax, cx ;Initialize count return value
  popreg
  ret

utoaZero:
  mov byte ptr [di], '0'
  mov byte ptr [di+1], '$'
  mov ax, 1
  popreg
  ret
```

```
;Parse a dollar-terminated string from byte ptr [si] as a binary string and return a 16-bit unsigned integer in ax.
atobin:
  pushreg
  xor ax, ax
  xor dx, dx

atobinLoop:
  mov dl, byte ptr [si]
  cmp dl, '$'
  je atobinLoopEnd
  sub dl, '0'
  cmp dl, 1
  ja atobinError
  shl ax, 1
  or ax, dx
  inc si
  jmp atobinLoop
atobinLoopEnd:

  popreg
  clc
  ret

atobinError:
  popreg
  stc
  ret

;Convert a 16-bit unsigned integer from ax to a binary string in byte ptr [di] of 16 characters; plus a terminator at 17th place. ax is unchanged.
bintoa:
  pushreg
  mov dx, 8000h

bintoaLoop:
  mov byte ptr [di], '0'
  test ax, dx
  jz bintoaSkip
  inc byte ptr [di]
bintoaSkip:
  inc di
  shr dx, 1
  cmp dx, 0
  jnz bintoaLoop

  mov byte ptr [di], '$'
  popreg
  ret
```