```
false 0 ; true -1 ;
off ( a -- ) true swap ! ; on ( a -- ) false swap ! ;
pad ( -- a ) temporary space Vhere @ 256 + ;
The number conversion routines use A and B address registers.
hex 16 Base ! ; decimal 10 Base ! ; octal 8 Base ! ;
8 Infd 8 Outfd
type ( a n -- ) Outfd @ -rot ( fd a n ) syswrite drop ;
negate ( n -- -1*n ) not 1 + ;
abs ( n -- |n| ) test -if negate then ;
asciise ( u1 -- u2 ) converts a digit u1 to an ascii value
    10 - -if 1 + 57 ascii code of 9 + ; then
    97 ascii code of a + ;
hold ( c -- ) put c in the ascii string c!- ;
sign ( n -- ) test -if 45 ascii code of - hold ; then drop ;
<# start number conversion 48 ascii code of 0 pad dup A! dup B! c! ;
#> ( -- a n ) A 1 + B A - ;
# ( u1 -- u2 ) write a single digit in ascii at the address in A
    Base @ /mod asciise c!- ;
#s ( u1 -- ) convert u1 to an ascii string at the adress in A
    begin # until ;
. ( n -- ) A push B push dup abs <# #s sign #> type pop B! pop A! ;
OREAD 0 ; OWRITE 1 ; ORDWR 2 ; OTRUNC 16 ;
close ( fd -- ) sysclose drop ;
exit Infd @ close Outfd @ close bye ;
264 Str 24 Tag
read ( a n -- ) Infd @ -rot ( fd a n ) sysread test if drop ; then eof exit ;
write ( a n -- ) Outfd @ -rot ( fd a n ) syswrite drop ;
swrite ( counted_string -- ) Outfd @ swap count syswrite drop ;
cwrite ( c -- ) writes c to Outfd Str c! Str 1 write ;
space writes a space to outfd   count write ;
newline writes a newline to outfd   count write ;
tab writes a tab to outfd   count write ;
accumulate ( n1 n2 -- n1*base+n2 ) swap Base @ * + ;
deascii ( c -- c-48 ) convert an ascii digit to a number
    48 ascii code of 0 - ;
digit ( n -- n ) Str 1 read Str c@ ( n c )
    dup ( n c c ) 48 ascii code of 0 57 ascii code of 9 belongs
    ( n c ) if deascii accumulate digit ; then drop ;
numerate 0 digit ;
>name read the length and the following string to Str
    numerate dup Str c! Str 1 + swap ( Str+1 len ) read ;
name> writes the counted string at Str to Outfd Str swrite ;
name read the name and write it to Outfd >name name> ;
base numerate Str c! Str 1 write ;
num read the base and number and write them base name ;
2drop ( a b -- ) drop drop ;
skip do nothing on a space ;
done ( c 'a -- ) execute 'a if Tag starts with c
    swap ( 'a c ) Tag c@ - ( 'a c-*Tag )
    if 2drop ; then pop 2drop execute ;
```

```
do ( c -- ) skip the following word in the caller if Start does not start with c
    Tag c@ - if pop 5 + push then drop ;
tag ( type n -- ) write type to Outfd and read n characters to Tag+1
    Tag 1 + swap read cwrite ;
textcomment 1 11 tag name ; execute 2 7 tag name ;
number 3 6 tag num ; compile 4 7 tag name ;
literalcompile 5 14 tag num ; macrocompile 6 12 tag name ;
define 7 6 tag name ; variable 8 8 tag name ;
address 11 7 tag name ; string 12 6 tag name ;
opens 1 arg OREAD sysopen Infd ! 2 arg OWRITE OTRUNC or sysopen Outfd ! ;
interpret Tag 1 read
    84 ascii code of T textcomment done
    69 ascii code of E execute done
    78 ascii code of N number done
    67 ascii code of C compile done
    76 ascii code of L literalcompile done
    77 ascii code of M macrocompile done
    68 ascii code of D define done
    86 ascii code of V variable done
    65 ascii code of A address done
    83 ascii code of S string done
    10 ascii code of LF newline done
    9 ascii code of HT tab done
    0 exit done
    32 ascii code of space skip done
    write an error message about unknown tag skip ;
loop interpret loop ;
main opens loop ;
```