```
\ prelude.9th

\ Y: Instruction Pointer
\ X: temporary W register
\ U: Param Stack
\ S: Return Stack

2 allot here
2 allot latest
80 allot inbuf
32 allot wordbuf
80 allot tmpbuf

code execute
  pulU x       ; arg -> W
  ldd 0,x      ; goto W+[W]
  jmp D,X
  ;

code swap
  ldx 2,u
  ldd 0,u
  stx 0,u
  std 2,u
  ;

code rot
  ldx 4,u
  ldd 2,u
  std 4,u
  ldd 0,u
  std 2,u
  stx 0,u
  ;

code dup
  ldd 0,u
  std ,--u
  ;

code drop
  leau 2,u
  ;

code c*     ; only multiplies two bytes.
  ldd ,u++  ; really we want b.
  lda 1,u   ; low byte into a.
  mul       ; D := A * B
  std 0,u
  ;
code +
  ldd ,u++
  addd 0,u
  std 0,u
  ;
code -
  ldd ,u++
  subd 0,u
  std 0,u
  ;
code 1+
  ldd ,u
  addd #1
  std ,u
  ;
code 1-
  ldd ,u
  subd #1
  std ,u
  ;
code bitand
  ldd ,u++
  anda 0,u
  andb 1,u
  std 0,u
  ;
code bitor
  ldd ,u++
  ora 0,u
  orb 1,u
  std 0,u
  ;
code bitxor
  ldd ,u++
  eora 0,u
  eorb 1,u
  std 0,u
  ;
code sex
  ldd ,u
  sex
  std ,u
  ;

code .
*** Currently prints only in HEX.
  ldd ,u++
  jsr PrintD,pcr
  ldb #32
  jsr putchar,pcr
  ;

code getline
  pshs x,y
retry_getline
  lda #2
  leax d_inbuf,pcr
  ldy #80

  os9 I$ReadLn
  bcc l1_getline

  *bcs retry_getline
  bcs l2_getline

  cmpb #$D3
  beq retry_getline
  bra l2_getline

l1_getline
  clra
  clrb
l2_getline
  sty ,--u  ; push length
  std ,--u  ; push error number, or 0.
  puls x,y
  ;

\ code getchar
\   jsr getchar,pcr
```

```
\   *tfr a,b
\   *sex
\   std ,--u
\   ;

code putchar
  ldd ,u++
  jsr putchar,pcr
  ;

code cr
  ldb #13  ; CR
  jsr putchar,pcr
  ;

code lit
  ldd ,Y++  ; get next cell from IP
  std ,--u  ; and stack it.
  ;

code branch
  ldd ,Y++  ; get next cell from IP
  leay d,y  ; add that offset to the IP.
  jmp Next,pcr
  ;

code 0branch
  ldd ,Y++  ; get next cell from IP
  ldx ,u++  ; pop a cell from the param stack
  bne b0001 ; only change IP if that was zero.
  leay d,y  ; add that offset to the IP.
b0001
  jmp Next,pcr
  ;

code not0branch
  ldd ,u++  ; pop a cell from the param stack
  beq b0002 ; only change IP if that was not
zero.
  ldd ,Y++  ; get next cell from IP
  leay d,y  ; add that offset to the IP.
b0002
  jmp Next,pcr
  ;

code =
  ldd ,u++
  cmpd ,u
  beq SetTrue
  bra SetFalse
  ;
code <>
  ldd ,u++
  cmpd ,u
  bne SetTrue
  bra SetFalse
  ;
code <
  ldd ,u++
  cmpd ,u
  bgt SetTrue
  bra SetFalse
  ;
code <=
  ldd ,u++
  cmpd ,u
  bge SetTrue
  bra SetFalse
  ;
code >
  ldd ,u++
  cmpd ,u
  blt SetTrue
  bra SetFalse
  ;
code >=
  ldd ,u++
  cmpd ,u
  ble SetTrue
  bra SetFalse
SetTrue
  clra  ; replace TOS with 1.
  clrb
  incb
  std ,u
  jmp Next,pcr
SetFalse
  clra  ; replace TOS with 0.
  clrb
  std ,u
  jmp Next,pcr
  ;

code r0
  ldd ,s
  std ,--u
  jmp Next,pcr
  ;
code r1
  ldd 2,s
  std ,--u
  jmp Next,pcr
  ;
code i
  ldd 2,s
  std ,--u
  jmp Next,pcr
  ;
code r1!
  ldd ,u++
  std 2,s
  jmp Next,pcr
  ;
code i!
  ldd 2,s
  std ,--u
  jmp Next,pcr
  ;

code >r
  ldd ,u++    ; pop from u, push to s.
  std ,--s
  jmp Next,pcr
  ;

code r>
  ldd ,s++    ; pop from s, push to u.
  std ,--u
  jmp Next,pcr
  ;
```

```
code rdrop
  leas 2,s    ; pop from s and discard.
  jmp Next,pcr
  ;

code !
  ldx ,u++    ; pop address
  ldd ,u++    ; pop value
  std ,x      ; poke
  jmp Next,pcr
  ;

code @
  ldx ,u      ; the address from the stack
  ldd ,x      ; what was at the address
  std ,u      ; value onto stack
  jmp Next,pcr
  ;

code c!
  ldx ,u++    ; pop address
  ldd ,u++    ; pop value
  stb ,x      ; poke just the low byte.
  jmp Next,pcr
  ;

code c@
  ldx ,u      ; the address from the stack
  ldb ,x      ; what byte was at the address
  clra
  std ,u      ; value onto stack
  jmp Next,pcr
  ;

code exit
  pulS y       ; pop previous IP.
  jmp Next,pcr ; and keep going.
  ;

code bye
OsExit
  ldb #13  ; CR
  jsr putchar,pcr
  clrb
  os9 F$Exit
  ;

code must
  ldx ,u++      tag
  ldd ,u++      value
  bne ret_must
  ldb #$3F  '?'
  jsr putchar,pcr
  tfr x,d
  jsr PrintD,pcr
  ldb #$3F  '?'
  jsr putchar,pcr
  ldb #$0D  CR
  jsr putchar,pcr
  ldb #255
  os9 F$Exit
ret_must
  jmp Next,pcr
  ;
```

```
: bl 32 ;

: words
  latest @
  begin
    dup
  while
    dup
    2 + c@  0  do
      dup 3 + i + c@ putchar
      loop
    bl putchar
    dup @  dup if  +  else  drop 0  then \ add
offset to base, unless offset is 0.
  repeat
  ;


: double
    dup + ;
: main1
    $D putchar $2 dup . double dup . double dup
. double .
    here . here @ .
    latest . latest @ .
    inbuf . tmpbuf .
    cr getline cr . cr
    ;

: main_getline  getline . . cr cr ;
\ : main_getchar  getchar . getchar . getchar .
getchar . cr cr ;

2 allot tmp

: main
    1 $888 must
    \ 0 $666 must
    cr
    64 putchar
    0 if 33 putchar then
    5 if 53 putchar then
    cr
    0 if 33 else 126 then putchar
    5 if 53 else 126 then putchar
    cr
    1 tmp !
    begin
      99 putchar
      tmp @
    while
      100 putchar
      0 tmp !
    repeat cr

    10 0 do i . loop  cr

    words cr

    bye
  ;

\ END
```