TEXT= (addr1 addr2 count -- flag)

Compare COUNT characters starting at ADDR1 and ADDR2. If they are all equal, return TRUE. This is NOT case sensitive.

`Related words:  MATCH? COMPARE`

TEXTROM (<name> -- , when created)

(index -- addr count , when used)

Creates a read-only string table. For example:

```
$ROM  MESSAGES  ," Hello" ,"  Abandon ship!"  ," Who am I?"
2 MESSAGES TYPE
```

`Related words: $ROM  ,"  $ARRAY`

THEN (--)

Terminate an IF .. THEN statement. See IF.

TIB (-- addr-buffer)

Address of the buffer where user input goes. Filled by QUERY. Read by INTERPRET.

```
TIB  #TIB @  .S TYPE
```

TOLOWER (char -- char')

If a character is UPPER case, convert it to lower.

TOUPPER (char -- char')

If a character is lower case, convert it to UPPER.

TRUE (-- -1 , Boolean constant)

TUCK (a b -- b a b)

Places a copy of the top item on the stack below the second item. Defined as SWAP OVER .

TYPE (addr count -- , type a string)

Types COUNT characters starting at ADDR to the output stream. TYPE is a deferred word.

Technical Note: In most Forths, TYPE calls EMIT. In HForth, EMIT calls TYPE for speed.

U* (u1 u2 -- du)

Same as UM*

U. (n --)

Print N as an unsigned number. Normally, any number with the most significant bit set is considered positive. U. is mostly used when printing big hex numbers or addresses that you think of as positive.

```
HEX AEC491B5  DUP . U.
```

U.R (u width --)

Print unsigned number in a field WIDTH characters wide. See .R .

U/ (du u -- u-rem u-quotient)

Divide a double precision number DU by a single precision number U. Generate an unsigned remainder and an unsigned quotient.

`Related words: M/ /`

U2/ (u -- u/2)

Fast unsigned divide by two. Use a logical right shift.

```
HEX BA987654 U2/ . ( prints 5D4C3B2A )
HEX BA987654 2/ . ( prints -22B3C4D6 )
```

U< (u1 u2 -- flag)

Compare two unsigned numbers. TRUE if U1 less than U2.

```
10 -20 U< . ( true !! cuz -20 is really big )
-20 U.
10 -20 < . ( false )
```

U> (u1 u2 -- flag)

Compare two unsigned numbers. TRUE if U1 greater than U2.

UNSMUDGE (--)

Clear a flag bit in the header of the most recently defined word to make it visible. FIND can now find it.

`Related words: SMUDGE`

UNTIL (flag --)

Terminate a BEGIN UNTIL statement. If the flag is FALSE, go back to the code immediately after the BEGIN. If the flag is true, continue. Think of is at "Keep looping until the contition is true".

```
: MANYHI  ( -- )
   BEGIN ." Stop me!" CR
      ?TERMINAL  ( returns TRUE if key hit )
   UNTIL
   ." You hit a " KEY EMIT CR
;
```

`Related words: BEGIN WHILE REPEAT IF DO LOOP`

USE->REL (usable-addr -- rel-addr)

Convert a usable address to a relocatable relative address.

See REL->USE.

VALUE (n <name> --)

Define a self fetching data structure. The initial value will be N. You can change the value using -> (same symbol as for local variables). You can use these like constants but VALUEs are often better because you can change them.

```
123 VALUE V1
V1 .  ( prints 123 )
987 -> V1  ( set V1 )
V1 . ( prints 987 )
```

VARIABLE (<name> --)

Define a simple data structure that returns its address. You can then use @ and ! to fetch or store values into this variable. The initial value of a variable if 0.

```
VARIABLE VAR1
VAR1 . ( print address of VAR1 )
654 VAR1 !  ( store new value for VAR1 )
VAR1 @ . ( fetch value from VAR1 , prints 654 )
```

VARIABLE is the same as V: which you will sometimes see in HMSL (saves typing!).

Warning: some older Forths take an initial value on the stack when a VARIABLE is created.

`Related words: CONSTANT CREATE VALUE @ ! +!`

W! (n addr --)

Store the lower 16 bits of N at address ADDR. ADDR must be even.

`Related words: W@ W,`

W, (n --)

Store the lower 16 bits of N at the current dictionary location and advance the dictionary pointer by 2.

Related words: , ALLOT W! CELL

W->S (w -- n)

Sign extend a 16 bit value to a 32 bit value.This is important if you want to store signed numbers in a 16 bit storage location because you will lose the "sign bits".

```
VARIABLE WVAR
-100 WVAR W!
WVAR W@ .  ( not -100 !!! )
WVAR W@ W->S . ( prints -100 )
```

Related words: W@ B->S S->D

W/ (n w -- n/w)

Fast divide of a 32 bit number by a 16 bit number. About 4 times faster than the full /. W and N/W must both be between -32768 and 32767.

W@ (addr -- w)

Fetch a 16 bit number W from the address ADDR. If you want a signed number you should then use W->S.

WARNING" (flag message" --)

If the flag is true, print the quote delimited message. Does not ABORT. Execution continues with the following statement.

```
DUP 127 > WARNING" Velocity too high!"
```

Related words: abort" ."

WHAT'S (<name> -- cfa)

Find out what a deferred word is supposed to be. See the section on deferred words.

Related words: DEFER IS

WHILE (flag --)

While the flag is true, continue executing the code in the BEGIN .. WHILE .. REPEAT loop. If FALSE, skip to the code following REPEAT.

```
: TEST  ( -- print lines of stars )
   BEGIN
      10 CHOOSE ?DUP 0>  ( loop until it picks 0 )
   WHILE
      0 DO ascii * EMIT  ( print line of '*'s )
      LOOP CR
   REPEAT
;
```

Related words: BEGIN REPEAT UNTIL IF DO LOOP

WITHIN? (num low high -- flag)

Accepts three numbers, num, low and high, and returns a value of true if low <= num <= high, returns false otherwise. Useful for range checking.

For example:

```
7 8 10 WITHIN? .
```

returns false, while

```
4 3 5 WITHIN? .
```

returns true.

`Related words:  CLIPTO`

WORD (delimiter-char <text> -- $addr)

Read one word from the input stream delimited by the given char and places it at $ADDR. The string is usually placed at HERE. Advances >IN which point to the next character to be read. If the length of the string at $ADDR is zero then we have hit the end of the line.

Forth uses word to parse Forth code by calling WORD with BL (blank space) as the delimiter. Imagine you want to print a message that contains a " which means you can't use .". Try this.

```
: .<  ( message> -- , print message )
   ASCII > WORD  ( get text )
   $TYPE
;
.<  He said "Hi!".>
```

`Related words: PARSE SCAN SKIP`

WORDS (-- , print all words in the dictionary)

WORDS.LIKE (<string> --)

Print a list of all words in the dictionary that contain the given string. This is very handy when you can't exactly remember a name.

WORDS.LIKE @

WORDS.LIKE PUT.

XDROP (i1 i2 i3 … in n --)

Drop N items from the stack.

XDUP (i1 i2 i3 … in n -- i1 i2 … in i1 i2 … in)

Duplicates N items on the stack.

Example: 4 5 7 9 4 XDUP .S (prints 4 5 7 9 4 5 7 9)

`Related words: DUP XDROP`

XOR (n1 n2 -- n3)

Perform a bit-wise exclusive OR of N1 and N2 and leave reult N3. The effects of an exclusive OR are best expressed by a truth table.

N1	N2	N3
0	0	0
1	0	1
0	1	1
1	1	0

Notice that doing an XOR with ones reverses the values of the bits.

```
BINARY  1010,1110 0000,1111 XOR . ( prints 1010,0001 )
```

`Related words: OR AND`

Y/N (-- flag)

Prints "(y/n)" and waits for user to hit 'y' or 'n'. Then returns TRUE for 'y' and FALSE for 'n'.

```
: SAVEIT
   ." Want to save it? " y/n
   IF ." Now saving!" ( not really )
   THEN
;
```

`Related words: Y/N/Q DIALOG.Y/N/C`

Y/N/Q (-- flag , or quits)

Prints "(y/n/q)" and waits for user to hit 'y' or 'n' or 'q'. Then returns TRUE for 'y' and FALSE for 'n'. If the user hits 'q', then call QUIT.

`Related words: Y/N DIALOG.Y/N/C`

[(--)

Put Forth into interpret mode by setting STATE to 0.

`Related words: LITERAL ] : ;`

[COMPILE] (<name> --)

Compile a call to the word that follows even if it is IMMEDIATE.

```
VARIABLE MSG-NUMBER
: #MSG  ( -- , print Message Number )
   >NEWLINE  ( start on new line )
   ." Message#" MSG-NUMBER @ .
   ." : "
   1 MSG-NUMBER +!  ( advance message number )
;
: MSG"  ( message" -- , print numberred message )
   COMPILE #MSG
   [COMPILE] ."  ( no text yet, get it later )
; IMMEDIATE
: TEST
   MSG" Phone home!"
   MSG" Buy some gas."
;
TEST
TEST
```

[] (-- syntactic tool for ODE)

Used to indicate late binding. See ODE chapter.

```
OBJ1  PRINT: []
```

\ (<comment line> --)

All text on the line after the '\' will be treated as a comment, ie. ignored.

```
\ say whatever you want
23 45 + . \ add some numbers together
```

] (--)

Enter Compile mode by setting STATE to 1.

`Related words: LITERAL [ : ;`