iForth v 4.1
Glossary

! "store" **CORE**

(x a-addr --)

Store *x* at *a-addr*.

!! "do-it" **IFORTH**

(--)

This command tells the server to execute the command that has just been send. The server will then execute the command and prepare for sending the results. You need this word when extending the server protocol.

!+ "store-plus" **IFORTH**

(x a-addr1 -- a-addr2)

Store *x* at *a-addr1* and leave the incremented pointer *a-addr2*.

!LATEST "store-latest" **IFORTH**

(dea --)

Make the routine identified by the dictionary entry address *dea* the last routine defined in the wordlist in which new definitions are stored. This wordlist is selected with <*wid*> **SET-CURRENT** or with **DEFINITIONS** .

See also: **@LATEST**

!TERMINAL "store-terminal" **IFORTH**

(i1 .. ini no ni fnr -- o1 .. ono)

Send the server command *fnr* over the boot link to the server. Send *ni* integer parameters. Then wait to receive no parameters associated with the command. See appendix F for the currently available commands. It is not necessary to use **!TERMINAL** unless you have an enhanced version of the server that supports more functions. For some examples see the file include/terminal.frt.

"CHAR "quote-character" **IFORTH**

(-- addr)

A user variable that holds the delimiter character used by **S"** , **."** etcetera.

```
Example:
'~' "CHAR !    S"    "Hello, World!"~ TYPE   &"  "CHAR !
Prints:
"Hello, World!"      ok
```

"number-sign" **CORE**

(ud1 -- ud2)

Divide *ud1* by the number in **BASE** giving the quotient *ud2* and the remainder *n*. (*n* is the least significant digit of *ud1*). Convert *n* to external form and add the resulting character to the beginning of the pictured numeric output string. Typically used between **<#** and **#>** .

#> "number-sign-greater" **CORE**

(xd -- c-addr u)

Drop *xd*. Make the pictured numeric output string available as a character string. *c-addr* and *u* specify the resulting character string.

#CELLS "number-sign-cells" **IFORTH**

(chars -- cells)

cells is the minimum amount of cells needed to store *chars* characters.

#CLINES "number-sign-c-lines" **IFORTH**

(-- a-addr)

a-addr is the address of **#CLINES** . **#CLINES** contains the total number of lines compiled since last reset.

See also: **#LINES**

#CPU "number-sign-c-p-u" **IFORTH**

(-- a-addr)

a-addr is the address of **#CPU** . **#CPU** contains the model number of the processor iForth is currently running on. This variable is initialized when iForth boots. **#CPU** is used to determine which instructions from the assembler are not valid on this processor.

#LINES "number-sign-lines" **IFORTH**

(-- a-addr)

a-addr is the address of **#LINES** . **#LINES** contains the line number of the line from the file that is currently interpreted. The first line of a file is line number 1. If an error occurs in an included file, **#LINES** points to the offending line.

#LOCALS **IFORTH**

#LOCALS is an environment query. It returns the maximum number of local variables in a definition. In iForth no exact answer can be given as local variables are allocated on a special purpose stack. #LOCALS will thus depend on the nesting depth of the definition. The fact that iForth supports both single and double precision FLOCALS further complicates matters.

See also: **ENVIRONMENT?**

#PARAMS "number-of-parameters" **IFORTH**

(-- +n)

The number of command line parameters passed to iForth by the host operating system. The parameter delimiters follow the rules of the host system.

```
Example:
 C:\IFORTH> ith  11 22 33<cr>
 ( startup messages deleted ...)
 FORTH> . . .<cr> 33 22 11 ok ( parameters on the OS command line behave
                                like you typed them at the prompt)
 FORTH> #PARAMS .<cr> 3 ok
```

See also: **'PARAM**

#S "number-sign-s" **CORE**

(ud1 -- ud2)

Convert one digit of *ud1* according to the rule for # . Continue conversion until the quotient is zero. *ud2* is zero. Typically used between **<#** and **#>** .

#TASK "number-sign-task" **IFORTH**

(-- a-addr)

a-addr is the address of **#TASK** . **#TASK** contains an identification number that is unique for each subprocess of the Forth system. The first process that is running on the processor is called the root-process and has the exclusive value 0 stored in this variable.

#TIB "number-t-i-b" **CORE**

(-- a-addr)

a-addr is the address of **#TIB** . **#TIB** contains the number of characters in the text input buffer. A Standard Program may not directly alter the contents of **#TIB** .

$.dd "string-dot-d-d" **IFORTH**

(c-addr u --)

Shows the tokens contained in the string at *c-addr u*. This word is a factor of **.dd** .

```
FORTH> : foo  2 3 + 7 AND ;  ok
FORTH> HEAD' foo HEAD>FLAGS @ =TOKENIZE AND . 16777216  ok
FORTH> HEAD' foo HEAD> @ idis
$00533780  : foo
$00533786  push        $0052C590 d#
$0053378B  push        #72 b#
$0053378D  push        $00533740 d#
$00533792  push        $0075DAD0 d#
FORTH> $0052C590 72 CR $.dd
2 3 + 7 AND ;  ok
```

See also: **.dd =TOKENIZE**

$CR "string-c-r" **IFORTH**

(-- a-addr)

a-addr is the address of **$CR** . **$CR** contains the counted string that is displayed by **CR** . It is operating system dependent.

$PROCESS "string-process" **IFORTH**

(c-addr u xt --)

The counted string described by *c-addr* and *u* becomes the temporary input buffer during the execution of *xt*. The following code is equivalent:

```
CREATE foo 4 ,
S"     foo "  ' CREATE  $PROCESS 4 ,
```

See also: **EVALUATE**

$THROW "string-throw" **IFORTH**

(c-addr --)

c-addr is the address of a counted string. **$THROW** performs the same function as **THROW** with the numeric argument -2, which is also the code used with **ABORT"** . *c-addr* is saved in an internal variable. The string at *c-addr* is printed when the **CATCH** / **THROW** mechanism reaches the bottommost level.

%VAR "percent-var" **IFORTH**

(-- a-addr)

a-addr is the address of **%VAR** . **%VAR** contains a code number representing the action that is to be performed by a word that is created with **VALUE** or any other word that creates variables based on the to-concept. **%VAR** is initialized by **FROM** , **TO** and all other words that modify the

behavior of a variable. The standard to-concept operators are **FROM TO +TO 0TO 'OF SIZEOF** and **/OF** . The table below shows the contents of **%VAR** after one of these words has been executed:

```
Keyword         Value
-TO             -2
+TO             -1
FROM             0
TO               1
0TO              2
'OF              3
SIZEOF           4
/OF              5
```

When building new words based on the to-concept you can add your own constants. Note that **%VAR** must be manipulated with **IMMEDIATE** words. For example, **FROM** is defined as:

```
: FROM   0 %VAR ! ; IMMEDIATE
```

See also: **FROM TO 0TO 'OF /OF +TO**

%a "parameter-a" **IFORTH**

The first member of a set of fixed names used for parameters. A parameter holds one of the stack parameters of the word it is in. A parameter has no address. **%a** is the first parameter the caller pushed when calling the word.

See also: **PARAMS| %f %b %c %d %e**

%b "parameter-b" **IFORTH**

The second member of a set of fixed names used for parameters. A parameter holds one of the stack parameters of the word it is in. A parameter has no address. ***%b*** is the second parameter the caller pushed when calling the word.

See also: **PARAMS| %a %f %c %d %e**

%c "parameter-c" **IFORTH**

The third member of a set of fixed names used for parameters. A parameter holds one of the stack parameters of the word it is in. A parameter has no address. ***%c*** is the third parameter the caller pushed when calling the word.

See also: **PARAMS| %a %b %f %d %e**

%d "parameter-d" **IFORTH**

The fourth member of a set of fixed names used for parameters. A parameter holds one of the stack parameters of the word it is in. A parameter has no address. ***%d*** is the fourth parameter the caller pushed when calling the word.

See also: **PARAMS| %a %b %c %f %e**

%e "parameter-e" **IFORTH**

The fifth member of a set of fixed names used for parameters. A parameter holds one of the stack parameters of the word it is in. A parameter has no address. **%e** is the fifth parameter the caller pushed when calling the word.

See also: **PARAMS| %a %b %c %d %f**

%f "parameter-f" **IFORTH**

The sixth member of a set of fixed names used for parameters. A parameter holds one of the stack parameters of the word it is in. A parameter has no address. **%f** is the sixth parameter the caller pushed when calling the word.

See also: **PARAMS| %a %b %c %d %e**

&! "and-store" **IFORTH**

(n|u a-addr --)

And *n|u* to the single-cell number at *a-addr*.

' "tick" **CORE**

("name" -- xt)

Parse *name* delimited by a space, ignoring leading delimiters. Find *name* and return *xt*, the execution token for *name*. An ambiguous condition exists if *name* is not found or if *name* is a standard word with the **C** attribute. When interpreting, ' *name* **EXECUTE** is equivalent to *name*.

'?AT "tick-query-at" **IFORTH**

(-- a-addr)

a-addr is the address of **'?AT** . **'?AT** contains the execution token of the definition that is actually executed when **?AT** is executed. Change the contents of this variable when the way in which **?AT** works must be changed.

See also: **?AT**

'ACCEPT "tick-accept" **IFORTH**

(-- a-addr)

a-addr is the address of **'ACCEPT** . **'ACCEPT** contains the execution token that is actually executed when **ACCEPT** or **EXPECT** is executed. Change the contents of this variable when a line editor with a different functionality is needed. For an example see the file include/accept.frt .

See also: **ACCEPT EXPECT**

'AT-XY "tick-at-x-y" **IFORTH**

(-- a-addr)

a-addr is the address of **'AT-XY** . **'AT-XY** contains the execution token of the definition that is actually executed when **AT-XY** is executed. Change the contents of this variable when the way in which **AT-XY** works must be changed.

See also: **AT-XY**

'BOOT "tick-boot" **IFORTH**

(-- a-addr)

a-addr is the address of **'BOOT** . **'BOOT** contains the execution token of the definition that is executed by **COLD** after all initialization is done. Applications should store the execution token of their main routine in this variable so that saved binary images will start the application if the image is reloaded.

'DATE$ "tick-date-string" **IFORTH**

(-- c-addr u)

c-addr is the address of a buffer which contains the current date as ASCII text. The text is *u* characters long. The buffer used is also in use as the numeric conversion buffer so the string should be used immediately.

'EMIT "tick-emit" **IFORTH**

(-- a-addr)

a-addr is the address of **'EMIT** . **'EMIT** contains the execution token of the definition that is actually executed when **EMIT** is executed. By replacing this execution token you can redirect the terminal output of a program. In that case you probably also want to change the contents of the vectors **'EMIT?** and **'TYPE** .

See also: **EMIT**

'EMIT? "tick-emit-question" **IFORTH**

(-- a-addr)

a-addr is the address of **'EMIT?** . **'EMIT?** contains the execution token of the definition that is actually executed when **EMIT?** is executed. By replacing this execution token you can redirect the terminal output of a program. In that case you probably also want to change the contents of the vectors **'EMIT** and **'TYPE** .

See also: **EMIT?**

'ENV$ "tick-environment-string" **IFORTH**

(+n -- c-addr u)

Returns the address and count of the *n*-th environment string. The 'environment' is the **OS** environment space and has nothing to do with Forth's **ENVIRONMENT?**

```
Example ("4"  probably doesn't work on your system):
  C:/IFORTH> set FOO=Forth is great.<cr>
  C:/IFORTH> ith<cr>
  ( startup messages deleted ...)
  FORTH> 4 'ENV$ TYPE<cr> FOO=Forth is great. ok
```

'ERRM$ "tick-error-message" **IFORTH**

(-- a-addr)

a-addr is the address of **'ERRM$** . **'ERRM$** contains the address of a character string that would be printed by **ABORT"** . This is useful if the string is not actually printed because of a **CATCH** , especially if the abort is generated by iForth itself.

'EVAL "tick-eval" **IFORTH**

(-- a-addr)

a-addr is the address of **'EVAL** . **'EVAL** contains the execution token of the routine that interprets user input when **STATE** contains 0, or compiles this input when **STATE** is anything else. This means the **QUIT** routine can be changed from the inside out.

See also: **QUIT EVALUATE**

'FORGET' "tick-forget-tick" **IFORTH**

(dea --)

Forget the definition that corresponds to *dea* and all definitions that were defined later.

See also: **FORGET**

'KEY "tick-key" **IFORTH**

(-- a-addr)

a-addr is the address of **'KEY** . **'KEY** contains the execution token of the definition that is actually executed when **KEY** is executed. By replacing this execution token you can redirect the keyboard input of a program.

See also: **KEY**

'KEY? "tick-key-question" **IFORTH**

(-- a-addr)

a-addr is the address of **'KEY?** . **'KEY?** contains the execution token of the definition that is actually executed when **KEY?** is executed. By replacing this execution token the terminal input of a program can be redirected.

See also: **KEY?**

'NUMBER "tick-number" **IFORTH**

(-- a-addr)

a-addr is the address of **'NUMBER** . **'NUMBER** contains the execution token of a text interpreter that interprets all notations of the various types of numbers available on the system. The execution semantics of this execution token should be the same as the execution semantics of the definition **NUMBER?** . By replacing the contents of this variable you can add new notations for single, double and floating-point numbers. Note that it is not possible to add a new type of numbers to the system.

'OF "tick-of" **IFORTH**

Usage: **'OF** name

Execution:(-- addr)

Get the address of the storage for the data associated with *name*. An ambiguous condition exists if *name* was not defined by **VALUE** or other words that implement the to-concept.

See also: **VALUE**

'PAGE "tick-page" **IFORTH**

(-- a-addr)

a-addr is the address of **'PAGE** . **'PAGE** contains the execution token of the definition that is actually executed when **PAGE** or **CLS** is called. By changing the contents of **'PAGE** you can redirect the output or adapt the method used to clear your terminal screen.

'PANIC "tick-panic" **IFORTH**

(-- addr)

A variable that allows to revector **PANIC** . **PANIC** is the code that handles hardware exceptions.

```
Example:  ' CALM 'PANIC !
```

Note that **'PANIC** is _not_ reset to (*e*.g.) **ABORT** when a **SAVE-SYSTEM** is done. Also note that **'PANIC** is (purposely) not a **USER** variable.

See also: **PANIC CALM**

'PARAM "tick-parameter" **IFORTH**

(+n -- c-addr u)

Returns the *n*-th command line parameter as a string. The zero-th parameter is normally the full path name of the currently executing iForth. The parameters stay available throughout program execution.

```
Example:
  C:/IFORTH> ith  BL 2 3<cr>
  ( startup messages deleted ...)
  FORTH> . . .<cr> 3 2 32 ok ( parameters on the OS command line behave
                               like you typed them at the prompt)
  FORTH> 3 'PARAM TYPE<cr> BL ok
  FORTH> 3 'PARAM EVALUATE .<cr> 32
```

See also: **#PARAMS**

'PROMPT "tick-prompt" **IFORTH**

(-- a-addr)

a-addr is the address of **'PROMPT** . **'PROMPT** contains the execution token of the definition that is to be executed when the command line interpreter wants to have input.

'TAIL "tick-tail" **IFORTH**

(-- a-addr)

a-addr is the address of **'TAIL** . **'TAIL** contains the execution token of the definition that is to be executed when the command line interpreter has finished executing the current line.

'TIME$ "tick-time-string" **IFORTH**

(-- c-addr u)

c-addr is the address of a buffer of *u* characters long containing the current time as ASCII text. The text in this buffer will not be updated as time changes. The buffer used is also in use as the numeric conversion buffer so the contents of this buffer are likely to be overwritten with words that use the numeric conversion buffer, including **'TIME$** .

'TYPE "tick-type" **IFORTH**

(-- a-addr)

a-addr is the address of **'TYPE** . **'TYPE** contains the execution token of the definition that is actually gets control when **TYPE** is executed. By changing this execution token you can redirect the terminal output of a program.

See also: **TYPE**

'UP "tick-u-p" **IFORTH**

(n -- a-addr)

a-addr is the address that has an *offset* of *n* cells relative to the start of the user-variable space.

'buffers "tick-buffer" **IFORTH**

(-- a-addr)

a-addr is the address of the file buffer area associated with the current task. Each **FORTH-PROCESS** has its own file buffers, and multi-process file **I/O** is explicitly permitted. Note that multi-process keyboard and screen **I/O** work, but is probably not to your satisfaction unless careful use of semaphores is made.

See also: **'fbuffer**

'fbuffer "tick-f-buffer" **IFORTH**

(-- a-addr)

a-addr is the address of the file buffer associated with the current nesting level. At present, 8 file buffers of 128 bytes are possible. This is the minimum ANS requirement for 'conforming implementations'.

See also: **'buffers**

("paren" **CORE**, **FILE**

("ccc<)>" --)

Parse characters *ccc* delimited by a closing parenthesis ")". Ignore the resulting text. (is an immediate word. When parsing, the number of characters in *ccc* may be zero to the number of characters remaining in the input stream. When parsing from a text file, the number of characters in *ccc* may be 0 to the number of characters remaining in the file.

(!PALETTE) **IFORTH**

(a-address u --)

The aligned address must point to *u* consecutive palette entries, beginning with the color with index 0. Palette entry *i* consists of three bytes describing the intensity of the red, green and blue component of the color with index *i*. The *u* palette entries are transferred to the video hardware.

See also: **(@PALETTE)**

(("paren-paren" **IFORTH**

(--) (S: -- x)

Save the contents of the variable **%VAR** to the system stack. Initialize the variable **%VAR** with **FROM** . The value of **%VAR** is restored again by **))** . You need to save the value of **%VAR** in this way if you want to use arrays or other data structure that need a VALUE-type index to identify an object:

```
123 TO (( index )) array
```

to store the value 123 in one of the elements of an array. Note that by writing the control words immediately before the objects you don't need to use **((** at all, but the above example may be slightly more readable than the same example below:

```
123 index TO array
```

See also: **%VAR FROM))**

(* "paren-star" **IFORTH**

("ccc<)>" --)*

(* is functionally equivalent to (. "*)" closes a comment introduced by **(*** .

(.) "paren-dot-paren" **IFORTH**

(n -- c-addr u)

Convert the single number on the data stack to its character string representation. An ambiguous condition exists if the character string exceeds the maximum size of the pictured numeric output string buffer.

See also: **(D.R)**

(0DEC.R) "paren-zero-dec-dot-R-paren" **IFORTH**

(n -- c-addr u)

Convert the single number on the data stack to its decimal character string representation. Equivalent to: **BASE @ >R DECIMAL U>D** 0 **(D.R) R> BASE** ! An ambiguous condition exists if the character string exceeds the maximum size of the pictured numeric output string buffer.

See also: **(D.R) 0DEC.R**

(1)PARAMS **C** **IFORTH**

(--)

An immediate word used to specify the number of parameters to reserve.

```
: SUMMA (2)PARAMS ( a b -- u )  %a %b + ;
```

Here SUMMA is told to reserve space for two integer parameters. Parameters have fixed names from the set { **%a** **%*b*** **%*c*** **%*d*** **%e** **%f** }. The construct can be nested. Note that the stack diagram is reversed from that of **LOCALS|** . Parameters are much faster than locals. However, they have no defined address. Do not mix parameters and locals in the same word.

See also: **(2)PARAMS** ... **(6)PARAMS PARAMS| LOCALS|**

(2)PARAMS **C** **IFORTH**

(--)

See: **(1)PARAMS** ... **(6)PARAMS PARAMS| LOCALS|**

(3)PARAMS **C** **IFORTH**

(--)

See: **(1)PARAMS** ... **(6)PARAMS PARAMS| LOCALS|**

(4)PARAMS **C** **IFORTH**

(--)

See: **(1)PARAMS** ... **(6)PARAMS PARAMS| LOCALS|**

(5)PARAMS **C** **IFORTH**

(--)

See: **(1)PARAMS** ... **(6)PARAMS PARAMS| LOCALS|**

(6)PARAMS **C** **IFORTH**

(--)

See: **(1)PARAMS** ... **(6)PARAMS PARAMS| LOCALS|**

(@PALETTE) **IFORTH**

(a-address u --)

The aligned address must point to a buffer for *u* consecutive palette entries, which will start with the color with index 0. Palette entry *i* consists of three bytes describing the intensity of the red, green and blue component of the color with index *i*. The *u* palette entries are transferred from the video hardware. See also **(!PALETTE)**

(B.) "paren-b-dot-paren" **IFORTH**

(x -- addr len)

Convert *x* to a 2 digit hexadecimal number in the number conversion area. *addr* is the start of the conversion area and *len* is the length of the string.

(C/L) "paren-c-slash-l-paren" **IFORTH**

(-- a-addr)

a-addr is the address of **(C/L)** . **(C/L)** contains the number of characters per line of the screen. It is initialized by a server call executed by **COLD** , so it is initialized at system startup. **(C/L)** is used by various words that need to know the screen width to optimally present their output. By using this variable, it is possible to make programs independent of any assumed screen width. The file include/terminal.frt gives code for a word **C/L** that makes sure **(C/L)** is updated periodically. This is helpful in environments where the screen width is routinely changed.

See also: **!TERMINAL**

(D.) "paren-d-dot-paren" **IFORTH**

(d -- c-addr u)

Convert the double number on the data stack to its character string representation. An ambiguous condition exists if the character string exceeds the maximum size of the pictured numeric output string buffer.

See also: **D.**

(D.R) "paren-d-dot-r-paren" **IFORTH**

(d n -- c-addr u)

Display *d* right aligned in a field |*n*| characters wide. If the number of characters required to display *d* is greater than |*n*|, all digits are displayed with no leading spaces in a field as wide as necessary. (In **D.R** , R stands for RIGHT). Returns the address of the transient area that holds the output string. When *n* is negative the fill character will be '0', not **BL** .

```
FORTH> -25 8 .r      -25 ok
FORTH> 25 -8 .r 00000025 ok
```

See also: **D.R (UD.R)**

(DLOCAL) **C** **IFORTH**

("name" --)

Like **(LOCAL)** but creates a temporary variable that can hold a double number.

See also: **(LOCAL)**

(E.) "paren-e-dot-paren" **FLOAT**

(-- c-addr u) (F: r --)

Convert the top number on the floating-point stack to its character string representation using scientific notation;

```
     <significand><exponent>
where:
     <significand> := [-]<digit>.<digits0>
     <exponent> := E[-]<digits>
```

The exact number of digits used for precision to the right of the decimal point in the significand is determined by **PRECISION** . An ambiguous condition exists if the system base is not **DECIMAL** or if the character string exceeds the maximum size of the pictured numeric output string buffer.

See also: **>FLOAT**

(E.R) "paren-e-dot-r-paren" **IFORTH**

(n -- c-addr u) (F: r --)

Creates a temporary string from the floating point number *r*, using the conversion rules as for **E.** , right aligned in a field *n* characters wide. If the number of characters required to display *r* is greater than *n*, all digits are displayed with no leading spaces in a field as wide as necessary.

See also: **E.R**

(F.) "paren-f-dot-paren" **FLOAT**

(-- c-addr u) (F: r --)

Convert the top number on the floating-point stack to its character string representation using fixed point notation:

```
[-] <digit>.<digits0>
```

The number of digits after the decimal point is determined by **PRECISION** . An ambiguous condition exists if the system base is not **DECIMAL** or if the character string exceeds the maximum size of the pictured numeric output string buffer. When the fieldwidth is too small, **F.R** and its variants try to squeeze the number in by switching to exponential notation. When that doesn't help, they print a field of asterisks. The switch to exponential notation happens when |*r*| **>** 10^PRECISION . When |*r*| **<** 10^-PRECISION , "0.00.." is printed.

See also: **(E.) F. E. (F.R) F.R (E.R) E.R (FE.R) FE.R**

(F.R) "paren-f-dot-r-paren" **IFORTH**

(n -- c-addr u) (F: r --)

Creates a temporary string from the floating point number *r*, using the conversion rules as for **F.** , right aligned in a field *n* characters wide. If the number of characters required to display *r* is greater than *n*, all digits are displayed with no leading spaces in a field as wide as necessary.

See also: **F.R**

(FE.) "paren-f-e-dot-paren" **IFORTH**

(-- c-addr u) (F: r --)

Convert the top number on the floating-point stack to a character string using the standard engineering notation for floating point numbers. An ambiguous condition exists if the system basis is not **DECIMAL** or if the character representation exceeds the size of the pictured numeric output string buffer.

(FE.R) "paren-f-e-dot-r-paren" **IFORTH**

(n -- c-addr u) (F: r --)

Creates a temporary string from the floating point number *r*, using the conversion rules as for **FE.** , right aligned in a field *n* characters wide. If the number of characters required to display *r* is greater than *n*, all digits are displayed with no leading spaces in a field as wide as necessary.

See also: **FE.R**

(FLOCAL) **C** **IFORTH**

("name" --)

Like **(LOCAL)** but creates a variable that can hold a floating-point number.

See also: **(LOCAL)**

(GETWD) "get-working-directory" **IFORTH**

(c-addr u1 -- c-addr u2 ior)

Get the current working directory of the host operating system. Put the path-name of the directory in the buffer starting at *c-addr* and length *u1*. *u2* is length of the directory name as returned by the host operating system or the size of the buffer, whichever is less. *ior* is 0 when the operation succeeded. Note that the directory name is in a format determined by the host operating system. Almost any operation on the returned string will result in a non portable program.

(H.) "paren-h-dot-paren" **IFORTH**

(u -- c-addr u)

Convert the unsigned number on the data stack to its character string representation using the hexadecimal base. An ambiguous condition exists if the character string exceeds the maximum size of the pictured numeric output string buffer.

(ISNAN) "paren-is-nan-paren" **IFORTH**

(-- flag) (F: r --)

Test the floating-point number *r* for being a bit pattern not representing a valid number. If *r* is not a valid number, true is returned, otherwise false is returned. Note that **-INF** and **+INF** are considered to be valid numbers and thus cause a false *flag* to be returned.

See also: **(NOTFIN)**

(LOCAL) "paren-local-paren" **C** **LOCAL**

(c-addr u --)

When executed during compilation, **(LOCAL)** passes a message to the Forth system that has one of two meanings. If *u* is non-zero, the message identifies a new local whose word name is given by the string of characters identified by *c-addr u*. If *u* is zero, the message is 'last local' and *c-addr* has no significance. The result of executing **(LOCAL)** during compilation of a definition is to create a set of named local identifiers, each of which is a word name, that have

execution semantics within the scope of that definition's source only. An ambiguous condition exists when **(LOCAL)** is executed while in interpret state.

local (-- x)

Push the local's value, *x*, onto the stack. An ambiguous condition exists when **(LOCAL)** is executed while in interpret state. Note: This word is not intended for direct use in a definition to declare that definition's locals. It is instead used by system or user compiling words. These compiling words in turn define their own syntax, and may be used directly in definitions to declare locals.

See also: **LOCAL FLOCAL**

(NOTFIN) "paren-not-fin-paren" **IFORTH**

(-- flag) (F: r --)

Test the floating-point number *r* for being a bit pattern not representing a finite number. If *r* is not a finite number true is returned, otherwise false is returned. Note that -NAN, +NAN, **-INF** and **+INF** are not considered to be finite numbers and thus cause a true *flag* to be returned.

See also: **(ISNAN)**

(SETWD) "set-working-directory" **IFORTH**

(c-addr u -- ior)

Set the current working directory of the host operating system. *c-addr* is the address of the new directory name in a format that matches the directory name format of the host operating system. *u* is the length of the directory name. *ior* is 0 when the operation succeeded. For MS-DOS systems a directory consisting of just the drive letter and a ':' (colon) results in a drive change on the host operating system.

(UD.) "paren-u-d-dot-paren" **IFORTH**

(ud -- c-addr u)

Convert the unsigned double number on the data stack to its character string representation. An ambiguous condition exists if the character string exceeds the maximum size of the pictured numeric output string buffer.

See also: **UD.**

(UD.R) "paren-u-d-dot-r-paren" **IFORTH**

(ud n -- c-addr u)

Convert the unsigned double number on the data stack to its character string representation. The number *ud* is right aligned in a field |*n*| characters wide. If the number of characters required to format *ud* is greater than |*n*|, all digits are displayed with no leading spaces in a field as wide as necessary. Returns the address of the transient area that holds the output string. An ambiguous condition exists if the character string exceeds the maximum size of the pictured numeric output string buffer. When *n* is negative the fill character will be '0', not **BL** .

See also: **UD.R (UD.)**

(ZLOCAL) **C** **IFORTH**

("name" --)

Like **(FLOCAL)** but creates a variable that can hold a floating-point complex number.

See also: **(FLOCAL)**

(lex) **IFORTH**

(c-addr1 u1 del -- c-addr3 u3 c-addr1 u4 del true) or (c-addr1 u1 del -- false)

Break a string in three pieces: before the delimiter *del*, the delimiter itself, and after the delimiter. *c-addr3* points to the remaining string, *c-addr1* points to the string in front of the delimiter. If false is returned, the input string does not contain the delimiter character.

)) **IFORTH**

(--) (S: x --)

Restore the value of the variable **%VAR** as saved on the system stack by **((** .

See also: **((**

***** "star" **CORE**

(n1|u1 n2|u2 -- n3|u3)

Multiply *n1|u1* by *n2|u2* giving the product *n3|u3*.

***/** "star-slash" **CORE**

(n1 n2 n3 -- n4)

Multiply *n1* by *n2* producing the double-cell intermediate result *d*. Divide *d* by *n3* giving the single-cell quotient *n4*. An ambiguous condition exists if *n3* is zero or if the quotient *n4* lies outside the range of a signed number. If *d* and *n3* differ in sign the result returned will be the same as the phrase **>R M* R> SM/REM SWAP DROP** . Note that other implementations of the ANS standard may return the phrase **>R M* R> FM/MOD SWAP DROP** .

***/MOD** "star-slash-mod" **CORE**

(n1 n2 n3 -- n4 n5)

Multiply *n1* by *n2* producing the intermediate double-cell result *d*. Divide *d* by *n3* producing the single-cell remainder *n4* and the single-cell quotient *n5* . An ambiguous condition exists if *n3* is zero, or if the quotient *n5* lies outside the range of a single-cell signed integer. If *d* and *n3* differ in sign the result returned will be the same as the phrase **>R M* R> SM/REM** . Note that other implementations of the ANS standard may return the phrase **>R M* R> FM/MOD** .

+ "plus" **CORE**

(n1|u1 n2|u2 -- n3|u3)

Add *n2|u2* to *n1|u1*, giving the sum *n3|u3*.

+! "plus-store" **CORE**

(n|u a-addr --)

Add *n|u* to the single-cell number at *a-addr*.

+!+ "plus-store-plus" **IFORTH**

(n|u a-addr1 -- a-addr2)

Add *n|u* to the single-cell number at *a-addr1* and leave the incremented address *a-addr2*.

+INF "plus-inf" **IFORTH**

(--) (F: -- +inf)

Place a bit pattern representing positive infinity on the floating-point stack.

+LOOP "plus-loop" **C** **CORE**

Compilation: (dodest --)

Resolve the destination of all unresolved occurrences of **LEAVE** between the location given by *dodest* and the next location for a transfer of control, to execute the words following **+LOOP** . Append the execution semantics given below to the current definition.

Execution: (n --) (R: sys -- \| sys2)

Add *n* to the loop index. If the loop index was not incremented across the boundary between the loop limit minus one and the loop limit then continue execution at the location given by the top element of the control flow stack. However, if the loop was incremented across the boundary between the loop limit minus one and the loop limit then discard the current loop control parameters and continue execution immediately following **+LOOP** . The loop control parameters must have been available.

See also: **DO I LEAVE +LOOPu +LOOPd**

+LOOPd "plus-loop-down" **C** **CORE**

Compilation: (dodest --)

Execution: (n --) (R: sys -- \| sys2)

Like **+LOOP** , but specialized for down counting loops: does not "wrap around." Use with **dDO** .

See also: **dDO uDO +LOOPu +LOOP LOOPe**

+LOOPu "plus-loop-up" **C** **CORE**

Compilation: (dodest --)

Execution: (n --) (R: sys -- \| sys2)

Like **+LOOP** , but specialized for up counting loops: does not "wrap around." Use with **uDO** .

See also: **dDO uDO +LOOPd +LOOP LOOPe**

+NAN "plus-nan" **IFORTH**

(--) (F: -- +NaN)

Place a bit pattern representing positive Not a Number on the floating-point stack.

+SIGN "plus-sign" **IFORTH**

(x --)

If *x* is negative add a minus sign to the beginning of the pictured numeric output string. Otherwise add a plus sign. Typically used between **<#** and **#>** .

+TO "plus-to" **IFORTH**

(n --)

Usage: **n +TO name**

Add *n* to *name*. An ambiguous situation exists if *name* was not defined by **VALUE** .

, "comma" **CORE**

(x --)

Reserve one cell of data space and store *x* in the cell. An ambiguous condition exists if the address of the next available data space location is not aligned.

," "comma-quote" **IFORTH**

Compilation: ("text" --)

Parse text delimited with " (double-quote). Add the resulting string including the length byte to the code space.

Execution: (--)

The bytes added by the above sequence can **NOT** be executed.

- "minus" **CORE**

(n1|u1 n2|u2 -- n3|u3)

Subtract *n2|u2* from *n1|u1*, giving the difference *n3|u3*.

-! "minus-store" **IFORTH**

(n|u a-addr --)

Subtract *n|u* from the single-cell number at *a-addr*.

-- "minus-minus" **IFORTH**

-- is an alias for \ .

See also: ****

-FROT "minus-f-rot" **IFORTH**

(--) (F: r1 r2 r3 -- r3 r1 r2)

Rotate the top three floating-point stack entries. This word uses a different rotation direction compared with **FROT** . It is equivalent to **FROT FROT** .

See also: **FROT**

-INF "minus-inf" **IFORTH**

(--) (F: -- -inf)

Place a bit pattern representing negative infinity on the floating-point stack.

-NAN "minus-nan" **IFORTH**

(--) (F: -- -NaN)

Place a bit pattern representing negative Not a Number on the floating-point stack.

-OPT "minus-opt" **IFORTH**

(--)

Reset the internal optimizer. Flush the on-chip stacks to memory. This word is only needed with very low-level operations. Normally, iForth removes explicit stack operations at the boundary of macro-words like **@** ! **COUNT** and many more. When using flow control words of the type **IF WHILE LOOP** , this optimization must be switched off:

```
@ BEGIN 2+ WHILE ...
```

As **BEGIN** assembles no code, the optimizer might try combining **@** and **2+** , with possibly disastrous results. Therefore, **BEGIN** has **-OPT** built-in. You will need **-OPT** when building new flow control words completely of your own design. Note that a standard program can only use **POSTPONE IF** and other predefined words, which take care of the problem automatically.

-R "minus-r" **C** **IFORTH**

(--) (R: x --)

Remove *x* from the return stack.

-ROT "minus-rot" **IFORTH**

(r1 r2 r3 -- r3 r1 r2)

Rotate the top three stack entries. This word uses a different rotation direction compared with **ROT** . It is equivalent to **ROT ROT** .

See also: **ROT**

-S "minus-s" **IFORTH**

(--) (S: x --)

Remove *x* from the system stack.

-TO "minus-to" **IFORTH**

(n --)

Usage: n **-TO** name

Subtract *n* from *name*. An ambiguous situation exists if *name* was not defined by **VALUE** . This word works for any type of **VALUE** .

-TRAILING "dash-trailing" **STRING**

(c-addr u1 -- c-addr u2)

If *u1* is greater than zero, *u2* is equal to *u1* less the number of spaces at the end of the character string specified by *c-addr* and *u1*. If *u1* is zero or the entire string consists of spaces, *u2* is zero.

. "dot" **CORE**

(n --)

Display *n* in free field format.

." "dot-quote" **C** **CORE**

Compilation: ("ccc<">" --)

Parse characters *ccc* delimited by " (double-quote). Append the execution semantics specified below to the current definition.

Execution: (--)

Display *ccc*.

See also: **.(.~**

.$ "dot-string" **IFORTH**

(c-addr --)

Print the string at *c-addr*. *c-addr* is the start address of a counted string.

.("dot-paren" **CORE EXT**

("ccc<)>" --)

Parse and display characters *ccc* delimited by) (closing parenthesis). **.(** is an immediate word.

See also: **." .~**

.ABOUT "dot-about" **IFORTH**

("ccc" --)

Parse *ccc* delimited by a space ignoring leading delimiters. Find *ccc* in the current search order. If *ccc* can not be found or *ccc* is not a module name as defined by **REVISION** , display an error message. Otherwise display information about the module *ccc*. If the module does not have any extra information, a default message is displayed. Extra information can be added to a module name using **:ABOUT** .

See also: **:ABOUT .HELP REVISION**

.DATA "dot-data" **IFORTH**

(--)

Copy and display the values currently on the data stack.

See also: **.S**

.DATE "dot-date" **IFORTH**

(--)

Prints the current date in the form "Month dd, year", with Month the full month name, dd the 2 digit day of the month and year the 4 digit year.

.FLAGS "dot-flags" **IFORTH**

(flags --)

Display a summary of what flags means. Example:

```
FORTH> HEAD' IF HEAD>FLAGS @ CR .FLAGS
IMMEDIATE, COMPILE-ONLY, ANSI ok
```

See also: **HEAD>FLAGS**

.FLOAT "dot-float" **IFORTH**

(--) (F: --)

Copy and display the values currently on the floating point stack.

See also: **.S**

.HELP "dot-help" **IFORTH**

(--)

Display information about the last module loaded. Modules are defined using **REVISION** . If no information is present for this module or no modules are defined, a default message is displayed.

See also: **:ABOUT .ABOUT REVISION**

.ID "dot-i-d" **IFORTH**

(nfa --)

Print the name of a dictionary entry with name field address *nfa*. If *nfa* is 0, print the text '{NoName}' instead. If the dictionary entry is invisible, nothing is printed.

See also: **=VISIBLE ID$**

.MODULES "dot-modules" **IFORTH**

(--)

Display the names of the words created with **REVISION** . The descriptive string compiled by **REVISION** is also displayed. The modules are listed in historical order.

See also: **REVISION**

.R "dot-r" **CORE EXT**

(n1 n2 --)

Display *n1* right aligned in a field |*n2*| characters wide. If the number of characters required to display *n1* is greater than |*n2*|, all digits are displayed with no leading spaces in a field as wide as necessary. When *n* is negative the fill character will be '0', not **BL** .

.S "dot-s" **TOOLKIT EXT**

(--)

Copy and display the values currently on the data stack, system stack and floating-point stack. The contents of each stack is printed on a separate line with the top of the stack printed at the right. The format of this display may not be the same on other implementations of the ANS standard. When one of the stacks has underflowed the text "Underflow" is shown along with the number of elements that are 'missing' from the stack.

.SERIAL# "dot-serial-number" **IFORTH**

(--)

Print the serial number of this copy of iForth. Every copy of iForth is sold with a unique serial number. You need the serial number when you contact us.

.SIGNON "dot-sign-on" **IFORTH**

(--)

Display the sign-on banner. This banner contains the version number, generation date, copyright statement and information about the configuration.

.SYSTEM "dot-system" **IFORTH**

(--) (S: --)

Copy and display the values currently on the system stack.

See also: **.s**

.TIME "dot-time" **IFORTH**

(--)

Prints the current time in the form "hh:mm:ss", with hh the 2 digit hour, mm the 2 digit minutes and ss the 2 digit seconds. This time is in the 24-hour format.

.TIME$ "dot-time-string" **IFORTH**

(--)

Print the current time and date.

.WORDLISTS "dot-wordlists" **IFORTH**

(--)

List the names of all word lists in the system. A name in this list does not need to be in the current search order.

.dd "dot-d-d" **IFORTH**

(xt --)

Shows the tokens of a TOKENIZEd word.

```
FORTH> : foo  2 3 + 7 AND ;  ok
FORTH> HEAD' foo HEAD>FLAGS @ =TOKENIZE AND . 16777216  ok
FORTH> HEAD' foo HEAD> @ CR .dd
2 3 + 7 AND ;  ok
```

See also: **$.dd =TOKENIZE**

.fstack "dot-f-stack" **C** **IFORTH**

(--)

Shows the values currently on the optimizer fp stack structure. Because of the intended usage this word is **IMMEDIATE** . The format will stay undocumented.

See also: **.pstack .qstack .sstack**

.lstack "dot-l-stack" **C** **IFORTH**

(--)

Shows the values currently on the optimizer locals stack structure. Because of the intended usage this word is **IMMEDIATE** . The format will stay undocumented.

See also: **.pstack .fstack .qstack .sstack**

.pstack "dot-p-stack" **C** **IFORTH**

(--)

Shows the values currently on the optimizer data stack structure. Because of the intended usage this word is **IMMEDIATE** . The format will stay undocumented.

See also: **.fstack .qstack .sstack**

.qstack "dot-q-stack" **C** **IFORTH**

(--)

Shows the values currently on the optimizer return stack structure. Because of the intended usage this word is **IMMEDIATE** . The format will stay undocumented.

See also: **.fstack .pstack .sstack**

.sstack "dot-s-stack" **C** **IFORTH**

(--)

Shows the values currently on the optimizer system stack structure. Because of the intended usage this word is **IMMEDIATE** . The format will stay undocumented.

See also: **.fstack .pstack .qstack**

.~ "dot-tilde" **IFORTH**

(--)

An alternative for **.(** and **."** , expecting the string to be delimited with the '~' character. This gets around the annoying problem that Forth strings can't embed the quite common ')' (or when compiling: '"') character. Strings having both '~' and ')' (or '"') characters are still a problem. Break them up and use **EMIT** .

See also: **.(."**

/ "slash" **CORE**

(n1 n2 -- n3)

Divide *n1* by *n2*, giving the single-cell quotient *n3*. An ambiguous condition exists if *n2* is zero. If *n1* and *n2* differ in sign, the result returned will be the same as the phrase **>R S>D R> SM/REM SWAP DROP** . Note that other implementations of the ANS standard may return the phrase **>R S>D FM/MOD SWAP DROP** .

/* "slash-star" **IFORTH**

(--)

/* is equivalent to (. '*/' is used to close the comment text started with **/*** .

/BRANCH "slash-branch" **IFORTH**

(-- a-addr)

a-addr is the address of **/BRANCH** . **/BRANCH** contains the number of bytes that the assembler reserves for forward branches.

/COUNTED-STRING "slash-counted-string" **IFORTH**

/COUNTED-STRING is an environment query.

See also: **ENVIRONMENT?**

/DATA-SPACE "slash-data-space" **IFORTH**

/DATA-SPACE is an environment query.

See also: **ENVIRONMENT?**

/HOLD "slash-hold" **IFORTH**

/HOLD is an environment query.

See also: **ENVIRONMENT?**

/MOD "slash-mod" **CORE**

(n1 n2 -- n3 n4)

Divide *n1* by *n2*, giving the single-cell remainder *n3* and the single-cell quotient *n4*. An ambiguous condition exists if *n2* is zero. If *n1* and *n2* differ in sign the result returned will be

the same as the phrase **>R S>D R> SM/REM** . Note that other implementations of the ANS standard may return the phrase **>R S>D R> FM/MOD** .

/OF "slash-of" **IFORTH**

Usage: **/OF** name

(-- u)

u is the number of data elements contained in the variable *name*. If *name* is not an array or list the number of elements is 1. An ambiguous condition exists if *name* is not defined by **VALUE** or any other word that implements the to-concept .

See also: **VALUE** to-concept(?)

/PAD "slash-pad" **IFORTH**

/PAD is an environment query.

See also: **ENVIRONMENT?**

/PARSE "slash-parse" **IFORTH**

(-- c-addr u)

First, removes all white space in front of whatever is in the input stream, then inspects the next (non-blank) character. When the next character is a single or double quote, returns the string thus delimited, without the outer quote characters. Otherwise, returns the next white space delimited string.

See also: **PARSE <WORD>**

/STRING "slash-string" **STRING**

(c-addr1 u1 n -- c-addr2 u2)

Adjust the character string at *c-addr1* by *n* characters. The resulting character string specified by *c-addr2* and *u2* begins at *a-addr1* plus *n* characters and is *u1* minus *n* characters long.

/SYSTEM "slash-system" **IFORTH**

(-- a-addr)

a-addr is the address of **/SYSTEM** . **/SYSTEM** delimits available memory available with **ALLOT** . You can change the value of **/SYSTEM** and execute **INIT-MEM** if the default amount of memory reported by **UNUSED** is not enough for your programs.

See also: **INIT-MEM UNUSED MEMSIZE**

0! "zero-store" **IFORTH**

(addr --)

Clear the contents of *addr*. Equivalent to 0 *addr* ! .

0< "zero-less" **CORE**

(n -- flag)

flag is true if *n* is less than zero.

0<, **IFORTH-ASSEMBLER**

(--)

Signals a following conditional jump mechanism like IF, or UNTIL, that the processor flags should be inspected for "0<" result.

0<= "zero-less-equal" **CORE EXT**

(n|u -- flag)

flag is true if *n*|*u* is equal or less to 0.

0<> "zero-not-equals" **CORE EXT**

(n|u -- flag)

flag is true if *n*|*u* is not equal to 0.

0<>, **IFORTH-ASSEMBLER**

(--)

Signals a following conditional jump mechanism like IF, or UNTIL, that the processor flags should be inspected for "0<>" result.

0= "zero-equals" **CORE**

(n|u -- flag)

flag is true if *n*|*u* is equal to zero.

0=, **IFORTH-ASSEMBLER**

Compilation: (--)

Signals a following conditional jump mechanism like IF, or UNTIL, that the processor flags should be inspected for "0=" result.

0> "zero-greater" **CORE EXT**

(n -- flag)

flag is true if *n* is greater than zero.

0>= "zero-greater-equal" **CORE EXT**

(n|u -- flag)

flag is true if *n*|*u* is greater than or equal to 0.

0>=, **IFORTH-ASSEMBLER**

(--)

Signals a following conditional jump mechanism like IF, or UNTIL, that the processor flags should be inspected for "0>=" result.

0DEC.R "zero-dec-dot-R" **IFORTH**

(n --)

Convert the single number on the data stack to its decimal character string representation and display it. Equivalent to: **BASE @ >R DECIMAL U>D** 0 **D.R R> BASE** ! An ambiguous condition exists if the character string exceeds the maximum size of the pictured numeric output string buffer.

See also: **D.R (0DEC.R)**

0TO "zero-to" **IFORTH**

Usage: **0TO** name

(--)

Store 0 in *name*. An ambiguous condition exists if *name* was not defined by **VALUE** or any other variable that is based on the to-concept. Floating point variables store 0E in *name*. Any user defined type of variable that is not an integer or a floating-point variable should store a reasonable initialization value in *name*.

See also: **CLEAR VALUE**

1+ "one-plus" **CORE**

(n1|u1 -- n2|u2)

Add 1 to *n1|u1* giving the sum *n2|u2*.

1- "one-minus" **CORE**

(n1|u1 -- n2|u2)

Subtract 1 from *n1|u1* giving the difference *n2|u2*.

1/F "one-slash-F" **IFORTH**

(F: r -- 1/r)

Equivalent to *r* 1e **FSWAP F/** .

2! "two-store" **CORE**

(x1 x2 a-addr --)

Store the cell pair *x1 x2* at *a-addr*, with *x2* at *a-addr* and *x1* at the next consecutive cell. It is equivalent to the sequence **SWAP OVER** ! **CELL+** ! .

2* "two-star" **CORE**

(n1 -- n2)

Multiply *n1* by 2 giving *n2*. Since the processor is a two's-complement machine this word can also be used to perform a logical left shift over 1 bit. Note that other implementations of the standard might use a one's-complement machine for which this feature does not work.

2+ "two-plus" **IFORTH**

(n1|u2 -- n2|u2)

Add 2 to *n1|u1* giving the sum *n2|n2*.

2+! "two-plus-store" **IFORTH**

(ud a-addr --)

Add *ud* to the double-cell number at *a-addr*.

See also: **D+!**

2- "two-minus" **IFORTH**

(n1|u1 -- n2|u2)

Subtract 2 from *n1|u1* giving the difference *n2|u2*.

2/ "two-slash" **CORE**

(n1 -- n2)

n2 is the result of dividing *n1* by two.

2>R "two-to-r" **C** **CORE EXT**

(x1 x2 --) (R: -- x1 x2)

Transfer cell pair *x1 x2* to the return stack. Semantically equivalent to **SWAP >R >R** .

2@ "two-fetch" **CORE**

(a-addr -- x1 x2)

Fetch the cell pair *x1 x2* stored at *a-addr*. *x2* is stored at *a-addr* and *x1* at the next consecutive cell. It is equivalent to the sequence **DUP CELL+ @ SWAP @** .

See also: **2!**

2@+ "two-fetch-plus" **IFORTH**

(a-addr -- a-addr+ x1 x2)

Fetch the cell pair *x1 x2* stored at *a-addr* and increment address by two cells . *x2* is stored at *a-addr* and *x1* at the next consecutive cell. It is equivalent to the sequence **DUP** 2 **CELLS** + **SWAP 2@** .

See also: **2@ 2!**

2CONSTANT "two-constant" **D** **DOUBLE**

(x1 x2 "name" --)

Parse *name* delimited by a space, ignoring leading delimiters. Create a dictionary entry for *name* with the execution semantics defined below. *name* is referred to as a "two-constant."

Execution: ("name" -- x1 x2)

Place cell pair *x1 x2* on the stack.

2DROP "two-drop" **CORE**

(x1 x2 --)

Drop cell pair *x1 x2* from the stack.

2DUP "two-dupe" **CORE**

(x1 x2 -- x1 x2 x1 x2)

Duplicate cell pair *x1 x2*.

2LITERAL "two-literal" **C** **DOUBLE**

Compilation: (x1 x2 --)

Compile cell pair *x1 x2* as a literal.

Execution: (-- x1 x2)

Place cell pair *x1 x2* on the stack.

2NIPS "two-nips" **IFORTH**

(n1 n2 n3 -- n3)

Remove the two values underneath TOS , equivalent to **NIP NIP** . Be careful to differentiate this from **2SWAP 2DROP** which I'*d* have named 2NIP .

See also: **NIP**

2OVER "two-over" **CORE**

(x1 x2 x3 x4 -- x1 x2 x3 x4 x1 x2)

Copy cell pair *x1 x2* to the top of the stack.

2R> "two-r-from" **C** **CORE EXT**

(-- x1 x2) (R: x1 x2 --)

Transfer cell pair *x1 x2* from the return stack. Semantically equivalent to **R> R> SWAP** .

2R@ "two-r-fetch" **C** **CORE EXT**

(-- x1 x2) (R: x1 x2 -- x1 x2)

Copy cell pair *x1 x2* from the return stack. Semantically equivalent to **R> R> 2DUP >R >R SWAP** .

2ROT "two-rote" **DOUBLE EXT**

(x1 x2 x3 x4 x5 x6 -- x3 x4 x5 x6 x1 x2)

Rotate the top three cell pairs on the stack bringing cell pair *x1 x2* to the top of the stack.

2SWAP "two-swap" **CORE**

(x1 x2 x3 x4 -- x3 x4 x1 x2)

Exchange the top two cell pairs.

2VARIABLE "two-variable" **D** **DOUBLE**

("name" --)

Parse *name* delimited by a space, ignoring leading delimiters. Create a dictionary entry for *name* with the execution semantics defined below. Reserve two consecutive cells of data space. *name* is referred to as a "two-variable."

Execution: ("name" -- a-addr)

a-addr is the address of the first (lowest address) cell of two consecutive cells in data space reserved by **2VARIABLE** when it defined *name*. The application is responsible for initializing the contents.

See also: **VARIABLE**

3DROP "three-drop" **IFORTH**

(x1 x2 x3 --)

Drop three elements from the stack.

3DUP "three-dupe" **IFORTH**

(n1 n2 n3 -- n1 n2 n3 n1 n2 n3)

Duplicate the top 3 elements of the stack.

4DROP "four-drop" **IFORTH**

(x1 x2 x3 x4 --)

Drop four elements from the stack.

: "colon" **D** **CORE**, **TOOLKIT EXT**

("name" -- colon-sys)

Usage: : name <words> ;

Parse *name* delimited by a space, ignoring leading delimiters. Create a dictionary entry for *name*. **Enter** compilation state. The execution semantics of *name* will be determined by the words compiled into the body of the definition following execution of : (colon) until the execution of ; (semi-colon). The newly created word definition for *name* cannot be found in the dictionary until the definition is completed. If the contents of the variable **POSTFIX** is true, *name* is not parsed from the input buffer but it is taken from the *c-addr*/*u* combination on the stack. Note that this is not an ANS required feature and is thus not portable. *name* is called a "colon definition". *colon-sys* is balanced by the corresponding ; or **;CODE** .

*Execution: (i*x -- j*x) (R: -- sys)*

Save implementation-dependent information (*sys*) about the definition that called *name* and perform the body of the definition.

See also: [ ] **;CODE**

:ABOUT "colon-about" **IFORTH**

(--)

Create a headerless colon definition that executes when **.ABOUT** <module name> or **.HELP** requests it. The definition typically contains an explanation of the corresponding module. Most likely a simple message is printed, but anything is possible here. For this word to function as intended, the module must use **REVISION** .

See also: **.ABOUT .HELP REVISION**

:FAST "colon-fast" **IFORTH**

(--)

Undocumented experimental colon definition type.

See also: **FAST; I/O FI/FO [XCOMPILE]**

:NONAME "colon-no-name" **CORE EXT**

(-- xt colon-sys)

Create an execution token and enter compilation state. Information is added to the end of the dictionary so that code compiled at the next dictionary location will be associated with *xt*. This code can be executed later by using *xt* **EXECUTE** . *colon-sys* is balanced by the corresponding ; or **;CODE** . An ambiguous condition exists if **:NONAME** is executed while in compilation state.

*Execution: (i*x -- j*x) (R: -- sys)*

Save implementation-dependent information about the definition that executed *xt*. Typically, the execution semantics of *xt* are expanded by compiling additional words into the definition.

; "semicolon" **C** **CORE**

Compilation: (colon-sys --)

Compile **EXIT** (or an implementation-dependent word that performs an equivalent function) in the current definition. **End** the current word definition and allow it to be found in the

dictionary. **Enter** interpretation state. *colon-sys* is balanced by the corresponding : or **:NONAME** .

Execution: (--) (R: sys --)

Return control to the caller of the definition containing ; . *sys* is balanced by the corresponding : or **:NONAME** .

;CODE "semicolon-code" **C** **TOOLKIT EXT**

Compilation: (colon-sys1 -- colon-sys2)

Terminate the defining word containing **;CODE** and allow it to be found in the dictionary. **Enter** interpret state. Add the **ASSEMBLER** word list to the search order. *colon-sys1* is balanced by the corresponding : or **:NONAME** . *colon-sys2* is balanced by the corresponding **END-CODE** .

Execution: (--) (R: sys --)

When the defining word containing **;CODE** is later executed it creates a new word definition name. When name is later executed, the machine code sequence following **;CODE** is executed. *sys* is balanced by the corresponding : or **:NONAME** .

See also: **DOES>**

< "less-than" **CORE**

(n1 n2 -- flag)

flag is true if *n1* is less than *n2*.

<# "less-number-sign" **CORE**

(--)

Initialize pictured numeric process.

<, **IFORTH-ASSEMBLER**

(--)

Signals a following conditional jump mechanism like IF, or UNTIL, that the processor flags should be inspected for "<" result.

<= "less-equal" **IFORTH**

(n1 n2 -- flag)

flag is true if *n1* is less then, or equal to *n2*.

<=, **IFORTH-ASSEMBLER**

(--)

Signals a following conditional jump mechanism like IF, or UNTIL, that the processor flags should be inspected for "<=" result.

<> "not-equals" **CORE EXT**

(x1 x2 -- flag)

flag is true if *x1* is not bit-for-bit the same as *x2*.

<>, **IFORTH-ASSEMBLER**

(--)

Signals a following conditional jump mechanism like IF, or UNTIL, that the processor flags should be inspected for "<>" result.

<WORD> "fast-word" **IFORTH**

(c -- c-addr u)

Perform the scanning functions of **WORD** . However the result is given as a character string. In this way the command executes much faster than **WORD** .

= "equals" **CORE**

(x1 x2 -- flag)

flag is true if *x1* is bit-for-bit the same as *x2*.

=, **IFORTH-ASSEMBLER**

(--)

Signals a following conditional jump mechanism like IF, or UNTIL, that the processor flags should be inspected for "=" result.

=: **D** **IFORTH**

(n --)

=: is an alias for **CONSTANT** .

See also: **CONSTANT**

==: **D** **IFORTH**

(x1 x2 --)

==: is an alias for **2CONSTANT** .

See also: **2CONSTANT**

=ANSI **IFORTH**

(-- mask)

mask is a bit-mask with one bit set to be used with the flags of a dictionary entry. If this bit is set in the flags, the word belongs to the ANS Forth Standard. New words created during a session have this bit set in order not to produce spurious messages.

=CELL **IFORTH**

(-- cell)

cell is the length of one cell in bytes. It is equivalent to 1 **CELLS** or 0 **CELL+** .

=COMP **IFORTH**

(-- mask)

mask is a bit-mask with one bit set to be used with the flags of a dictionary entry. If this bit is set in the flags, the word is compile-only. New words created during a session have this bit turned on by executing **COMPILE-ONLY** directly after defining the word.

=DENOTATION **IFORTH**

(-- mask)

A mask that signals to the optimizer that thus adorned word will scan the input stream and takes full responsibility for that. Normally the optimizer will turn off tokenization when a word changes **>IN** . Example words that have **=DENOTATION** set: **[CHAR] S"** etc.

See also: **DENOTATION** **=TOKENIZE**

=FIFOMASK **IFORTH**

(-- mask)

An undocumented mask.

=IMMEDIATE **IFORTH**

(-- mask)

mask is a bit-mask with one bit set to be used with the flags of a dictionary entry. If this bit is set in the flags, the word is immediate. New words created during a session have this bit turned on by executing **IMMEDIATE** directly after defining the word.

=IOMASK **IFORTH**

(-- mask)

An undocumented mask.

=PRIVATE **IFORTH**

(-- mask)

mask is a bit-mask with one bit set to be used with the flags of a dictionary entry. If this bit is set, the word is private. New words created during a session have this bit turned on by executing **PRIVATE** directly after defing the word. Words marked as private are made invisible with the command **DEPRIVE** .

See also: **PRIVATE DEPRIVE**

=TOKENIZE **IFORTH**

(-- mask)

mask is a bit-mask with one bit set to be used with the flags of a dictionary entry. If this bit is set in the flags, the word has been TOKENIZEd, *i.e.* in addition to compiled code, the dictionary holds a string of equivalent tokens. The token string can be inspected with **.dd** . The optimizer may choose to inline the tokens instead of compiling a call to the machine code. It is not possible to prevent a word from being tokenized, but it is possible to turn off the **=TOKENIZE** bit. The optimizer does not inline **IMMEDIATE** words. (Making a word **IMMEDIATE** turns of the **=TOKENIZE** bit.) There is no high-level word, say TOKENIZE , to turn on the **=TOKENIZE** flags.

See also: **IMMEDIATE .dd**

=VISIBLE **IFORTH**

(-- mask)

mask is a bit-mask with one bit set to be used with the flags of a dictionary entry. If this bit is reset, the word is invisible, *i.e.* it cannot be found by words as ' **HEAD' FIND WORDS** .

> "greater-than" **CORE**

(n1 n2 -- flag)

flag is true if *n1* is greater than *n2*.

>, **IFORTH-ASSEMBLER**

(--)

Signals a following conditional jump mechanism like IF, or UNTIL, that the processor flags should be inspected for ">" result.

>< **IFORTH**

(16bit -- 16bit')

Swap the low and high bytes of the 16 bit word on the data stack.

>= "greater-equal" **IFORTH**

(n1 n2 -- flag)

flag is true if *n1* is greater then, or equal to *n2*.

>=, **IFORTH-ASSEMBLER**

(--)

Signals a following conditional jump mechanism like IF, or UNTIL, that the processor flags should be inspected for ">=" result.

>BODY "to-body" **CORE**

(xt -- a-addr)

a-addr is the data field address corresponding to *xt* for a word defined via **CREATE** .

>FLOAT "to-float" **FLOAT**

(c-addr u -- true) (F: -- r) or (c-addr u -- false)

An attempt is made to convert the string specified by *c-addr* and *u* to internal floating-point representation. If the string represents a valid floating-point number in the syntax below, its value *r* and true are returned. If the string does not represent a valid floating-point number only false is returned. A string of blanks is treated as a special case representing zero. Syntax:

```
Convertible string := <significand>[<exponent>]
<significand> := [<sign>]{<digits>[.<digits0>] | [<digits0>].<digits>}
<sign> := { + | - }
<exponent> := <marker><digits0>
<digits> := <digit><digits0>*
<digits0> := <digit>*
<digit> := { 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 }
<marker> := { <e-form> | < sign-form> }
<e-form> := <e-char>[<sign-form>]
<sign-form> := { + | - }
<e-char>:= { D | d | E | e }
```

>GRAPHIC "to-graphic" **IFORTH**

(c1 -- c2)

If the character c1 is in the range { 32 .. 126 } return c1, otherwise return '.' (full stop).

>GSCREEN "to-graphic-screen" **IFORTH**

(c-addr1 c-addr2 u --)

If *u* is greater than zero, copy *u* consecutive characters from *c-addr1* to *c-addr2*. Equivalent to **CMOVE** , **CMOVE>** or **MOVE** if *c-addr2* is not in graphic memory. Only use **>GSCREEN** to do block moves to the graphics screen. Block moves across screen memory are not supported. They will hang the machine.

See also: **GSCREEN> TSCREEN> >TSCREEN**

>HEAD "to-head" **IFORTH**

(xt -- dea | 0)

dea is the address of the dictionary entry whose execution token is *xt* . If the conversion was not possible a 0 is returned.

>IN "to-in" **CORE**

(-- a-addr)

a-addr is the address of **>IN** . **>IN** contains the *offset* in characters from the start of the current input stream to the next character to be parsed.

>LCMOVE "to-low-cmove" **IFORTH**

(addr l-addr u --)

Move *u* bytes from *addr* to *offset* l-*addr* in the first physical Megabyte of memory. MS-DOS only.

>LWC "to-lower-case" **IFORTH**

(C1 -- c2)

Convert a character to its lowercase equivalent.

See also: **>UPC**

>MS "to-m-s" **IFORTH**

(ticks -- ms)

Convert a time in timer ticks to a time in milliseconds. The process priority is correctly taken into account. Typically used with: TIMEOUT ?TIMEOUT

See also: **>TICKS MS ?MS**

>NUMBER "to-number" **CORE**

(ud1 c-addr1 u1 -- ud2 c-addr2 u2)

ud2 is the result of converting the characters within the character string specified by *c-addr1 u1* into digits, using the number in **BASE** , and adding each into *ud1* after multiplying *ud1* by the number in **BASE** . Conversion continues until a character that is not convertible is encountered or the string is entirely converted. *c-addr2* is the location of the first unconverted character or the first character past the end of the string if the string was entirely converted. *u2* is the number of unconverted characters in the string. An ambiguous condition exists if *ud2* overflows. iForth allows any amount of the following characters in a number string: { : , - . / }. They tell the system a double precision number is meant. Note that other full-ANS systems only recognize numbers that end with a '.' (full stop) as a double number. Furthermore iForth allows BASE-independent number input, controlled by the first character of the input string:

```
Character   Meaning        Example   equivalent to
---------   -------        -------   -------------
$           hexadecimal    $123      [ HEX ] 123
#           decimal        #123      [ DECIMAL ] 123
%           binary         %101      [ BINARY ] 101
&           literal        &a        [CHAR] a
'           literal        'a'       [CHAR] a
"           literal        "a"       [CHAR] a
^           control char   ^G        [CHAR] G [ DECIMAL ] 31 AND
```

When the variable **ANSI** contains **TRUE** a warning is issued about the usage of all numbers that might not be recognized by other ANS Forth systems. Note that **>NUMBER** does not handle negative numbers as '-' is considered an unconvertible character.

>R "to-r" **C** **CORE**

(x --) (R: -- x)

Move *x* to the return stack.

>S "to-s" **IFORTH**

(x --) (S: -- x)

Move *x* to the system stack.

>TICKS "to-ticks" **IFORTH**

(ms -- ticks)

Convert a time in milliseconds to a time in timer *ticks*. The process priority is correctly taken into account.

See also: **>MS MS ?MS**

>TSCREEN "to-text-screen" **STRING**

(c-addr1 c-addr2 u --)

If *u* is greater than zero, copy *u* consecutive characters from *c-addr1* to *c-addr2*. Equivalent to **CMOVE** , **CMOVE>** or **MOVE** if *c-addr2* is not in screen memory. Only use **>TSCREEN** to do block moves to the screen. Block moves across screen memory are not supported. They will hang the machine.

See also: **TSCREEN>**

>UPC "to-up-c" **IFORTH**

(c1 -- c2)

Convert a character to its uppercase equivalent.

>VOCNAME "to-vocabulary-name" **IFORTH**

(addr1 -- addr2)

Convert the data field address of a vocabulary word to its name field address.

? "question" **TOOLKIT EXT**

(a-addr --)

Display the value stored at *a-addr*.

?ALLOCATE **IFORTH**

(errno --)

errno is the error number returned by words from the MEMORY wordset. The error number, when representing a real error, is converted to an error message issued by **ABORT"** .

?AT "query-at" **IFORTH**

(-- column row)

Report the position where the next character output will appear. Format is *column*, *row*. The upper left corner of the output device is row zero, column zero. This word is a no-op when the operation cannot be performed on the current output device with the specified parameters.

See also: **'?AT AT-XY**

?DEF **IFORTH**

("name" -- flag)

Parse *name* delimited by a space ignoring leading delimiters. If *name* can be found using the current search order then *flag* is true, otherwise *flag* is false.

?DO "question-do" **C** **CORE EXT**

Compilation: (-- dodest)

The next location for a transfer of control (*dodest*) goes onto the control-flow stack. Append the execution semantics given below to the current definition.

Execution: (n n --) (R: --) or (n1|u1 n2|u2 --) (R: -- sys)

If *n1|u1* is equal to *n2|u2*, continue execution at the location given by the consumer of *dodest*. Otherwise set up loop control parameters with index *n2|u2* and limit *n1|u1* and continue executing immediately following **?DO** . Anything already on the return stack becomes unavailable until the loop control parameters are discarded. An ambiguous condition exists if *n1|u1* and *n2|u2* are not both of the same type.

See also: **I LEAVE LOOP UNLOOP**

?DUP "question-dupe" **CORE**

(x -- x x) or (0 -- 0)

Duplicate *x* if it is non-zero.

?ERROR" "question-error-quote" **C** **IFORTH**

Compilation: ("ccc<">" --)

Parse characters *ccc* delimited by a double quote mark.

*Execution: (i*x flag nr --) (R: j*x --)*

If all bits of *flag* are zero, execute the sequence of words after *ccc*<">. Otherwise, display *ccc* and perform an error abort sequence which includes the function of **ABORT** . Associate the error number nr with the error.

?FOR **C** **IFORTH**

Compilation: (-- fordest)

The next location for a transfer of control (*fordest*) goes onto the control flow stack. Append the execution semantics given below to the current definition.

Execution: (n|u --) (R: -- sys)

Set up the loop control parameter with limit *n|u*. Anything already on the return stack becomes unavailable until the loop control parameters are discarded. Each pass through the loop, the loop control parameter is tested for zero. If it becomes zero or is zero initially the loop is exited, if it does not, the control parameter is decremented by one. If the limit *n|u* is 0 to start with, no pass will be made. It is allowed to use **LEAVE** to exit from a **?FOR** ... **NEXT** loop.

See also: **FOR AFT NEXT**

?MS **IFORTH**

(-- time)

time is a time in milliseconds derived from the value of the system clock. The only property of *time* is the fact that it represents a time in milliseconds, there is no 'starting point'. Note that when the system clock is halted, which is only possible on some processor models, the time returned by **?MS** will not advance anymore. Note also that processes are free to load the system timer with a new value.

?REPEATED **C** **IFORTH**

*Compilation: (n*orgs dest --)*

Append the execution semantics given below to the current definition, resolving the backward reference *dest*. Resolve the *n* forward referenced *orgs* using the location following the

appended execution semantics. **?REPEATED** aborts with an error message if **SECURE** is **OFF** . This word allows the standard ANS phrase **BEGIN** ... **WHILE** ... **WHILE** ... **WHILE** .. **REPEAT REPEAT REPEAT** to be written as **BEGIN** ... **WHILE** ... **WHILE** ... **WHILE** ... **?REPEATED** .

Execution: (--)

Continue execution at the location given by *dest*.

See also: **BEGIN WHILE UNTIL**

?STACK **IFORTH**

(--)

Check all 5 stacks (data, float, local, system and return) for a possible underflow. If one of these stacks did underflow an error is issued. Note that overflow is not checked for any of the stacks, and only one message is given even though more than one stack may have underflowed.

?UNDEF **IFORTH**

("name" -- flag)

Parse *name* delimited by a space ignoring leading delimiters. If *name* can not be found using the current search order, *flag* is true, otherwise *flag* is false.

See also: **?DEF**

@ "fetch" **CORE**

(a-addr -- x)

x is the value stored at *a-addr*.

@+ "fetch-plus" **IFORTH**

(a-addr1 -- a-addr2 x)

Fetch *x* from *a-addr1*. Add 1 **CELLS** to *a-addr1* giving *a-addr2*.

@- "fetch-minus" **IFORTH**

(a-addr1 -- a-addr2 x)

Fetch *x* from *a-addr1*. Subtract 1 **CELLS** from *a-addr1* giving *a-addr2*.

@-frot, **IFORTH-ASSEMBLER**

Assembling: (--)

running (internal F: a b c --- c a b)

Assembler macro that generates code that will inversely rotate the three numbers on the internal floating-point stack during run-time.

@EXECUTE "fetch-execute" **IFORTH**

(a-addr --)

Fetch the execution token stored at *a-addr*. Execute the definition specified by the execution token if the execution token does not have all bits set to zero.

See also: **PERFORM**

@LATEST **IFORTH**

(-- dea)

Get the dictionary entry address of the latest header that was created in the current wordlist (the one accessed with **GET-CURRENT SET-CURRENT**).

@f2drop, **IFORTH-ASSEMBLER**

Assembling: (--)

running (internal F: a b --)

Assembler macro that generates code that will drop two numbers from the internal floating-point stack during run-time.

@f2dup, **IFORTH-ASSEMBLER**

Assembling: (--)

running (internal F: a b --- a b a b)

Assembler macro that generates code that will duplicate the top two numbers on the internal floating-point stack during run-time.

@fdrop, **IFORTH-ASSEMBLER**

Assembling: (--)

running (internal F: a ---)

Assembler macro that generates code that will drop the top number from the internal floating-point stack during run-time.

@fdup, **IFORTH-ASSEMBLER**

Assembling: (--)

running (internal F: a --- a a)

Assembler macro that generates code that will duplicate the top of the internal floating-point stack during run-time.

@fnip, **IFORTH-ASSEMBLER**

Assembling: (--)

running (internal F: a b --- b)

Assembler macro that generates code that will drop the second number from the internal floating-point stack during run-time.

@fnstsw, **IFORTH-ASSEMBLER**

Assembling: (--)

running (internal F: ---)

Assembler macro that assembles code that loads the FPU status word in EAX .

@fover, **IFORTH-ASSEMBLER**

Assembling: (--)

running (internal F: a b --- a b a)

Assembler macro that generates code that will copy the second number on the internal floating-point stack to the top. (FLD ST(1))

@frot, **IFORTH-ASSEMBLER**

Assembling: (--)

running (internal F: a b c --- b c a)

Assembler macro that generates code that will rotate the three numbers on the internal floating-point stack during run-time.

@fswap, **IFORTH-ASSEMBLER**

Assembling: (--)

running (internal F: a b --- b a)

Assembler macro that generates code that will swap the two numbers on the internal floating-point stack during run-time.

ABORT **CORE**

*(i*x --) (R: j*x --)*

Empty the data stack and perform the function of **QUIT** without displaying a message.

ABORT" "abort-quote" **C** **CORE**

Compilation: ("ccc<">" --)

Parse characters *ccc* delimited by a double quote mark.

*Execution: (i*x flag --) (R: j*x --)*

If all bits of *flag* are zero, execute the sequence of words after *ccc*<">. Otherwise, display *ccc* and perform an error abort sequence which includes the function of **ABORT** .

ABS "abs" **CORE**

(n -- u)

u is the absolute value of *n*. Note that on every 2's complement machine there is exactly one negative number (the most negative integer) that does not have a representable absolute value.

ACALL, "aligned-call-comma" **IFORTH**

(xt --)

Compile a processor assembler subroutine call to the routine identified by the execution token *xt*. The compiler handles this in such a way that it is valid for the routine being called to execute **R> @** , meaning that the next address in the dictionary is guaranteed to be aligned. Note that this is needed for all **DOES>** constructs. This word is only meaningful for Forths that do not have a separate data space. See /include/miscutil.frt, the word **EXEC:** for example usage.

See also: **COMPILE, CALL,**

ACCEPT **CORE**

(c-addr +n1 -- +n2)

Receive a string of at most *+n1* characters. An ambiguous condition exists if *+n1* is zero or greater than 32,767. Display graphic characters as they are received. Note that editing functions that the system performs in order to construct the string are implementation-defined. This system appends each typed character to the end of the string except for the backspace key which removes one key from the end of the string if available. Other implementations of the ANS standard may use different editing keys. Input terminates when "return" is received. When "return" is received, nothing is appended to the string. *+n2* is the length of the string stored at *c-addr*.

See also: **EXPECT**

ADDRESS-UNIT-BITS **IFORTH**

ADDRESS-UNIT-BITS is an environment query.

See also: **ENVIRONMENT?**

ADJUST-STACK **IFORTH**

Used in inline macro's to optimize parameter access. The word **ADJUST-STACK** is used together with **IN/OUT** and **FIN/FOUT** to flush registers and FPU contents to the memory data stacks after a macro (or set of macro's) has finished. **IN/OUT** , **FIN/FOUT** and **ADJUST-STACK** work together in order to remove superfluous stack traffic at macro boundaries. For an example see **IN/OUT** .

See also: **IN/OUT FIN/FOUT ASM{ }ASM**

AFT **C** **IFORTH**

(-- orig)

AFT is a special purpose version of **AHEAD** that is typically used in **FOR** ... **NEXT** loops. It makes sure the loop is exited immediately if the loop control parameter *n* is 0 to start with. Example:

```
: TEST1  0 FOR      I@ .       NEXT ;  \ prints "0"
: TEST0  0 FOR  AFT I@ . THEN  NEXT ;  \ prints nothing
```

Also note that a **FOR** ... **NEXT** loop using **AFT** executes *n* times, not *n*+1 times.

AGAIN **C** **CORE EXT**

Compilation: (dest --)

Append the execution semantics given below to the current definition, resolving the backward reference *dest*.

Execution: (--)

Continue execution at the location specified by *dest*. If no other control flow words are used, any program code after **AGAIN** will not be executed.

See also: **BEGIN**

AGAIN, **IFORTH-ASSEMBLER**

Assembling: (addr --)

Assemble a jump to the memory location *addr*.

Execution: (--)

Jump to the memory location *addr*.

AHEAD **C** **TOOLKIT EXT**

Compilation: (-- orig)

Put the location of a new unresolved forward reference *orig* onto the control flow stack. Append the execution semantics given below to the current definition. The semantics are incomplete until *orig* is resolved (*e*.g., by **THEN**).

Execution: (--)

Continue execution at the location specified by the resolution of *orig*.

AHEAD, "ahead-comma" **IFORTH-ASSEMBLER**

Assembling: (-- addr)

Place the value of the current code space pointer on the data stack. Assemble an unconditional forward branch that is to be resolved by, for instance, THEN, .

Execution: (--)

Jump to the memory location just after the word that resolved this forward branch.

ALIGN **CORE**

(--)

If the address of the next available data space location is not aligned, reserve enough space to align it. Note that **ALIGN** is both a standard word and an environment query.

See also: **ENVIRONMENT?**

ALIGN1024 **IFORTH**

(--)

If the address of the next available data space location is not aligned to a multiple of 1024 bytes, reserve enough space to align it.

ALIGN16 **IFORTH**

(--)

If the address of the next available data space location is not aligned to a multiple of 16 bytes, reserve enough space to align it.

ALIGN32 **IFORTH**

(--)

If the address of the next available data space location is not aligned to a multiple of 32 bytes, reserve enough space to align it.

ALIGN64 **IFORTH**

(--)

If the address of the next available data space location is not aligned to a multiple of 64 bytes, reserve enough space to align it.

ALIGN8 **IFORTH**

(--)

If the address of the next available data space location is not aligned to a multiple of 8 bytes, reserve enough space to align it.

ALIGNED **CORE**

(addr -- a-addr)

a-addr is the first aligned address greater than or equal to *addr*.

ALIGNS? **IFORTH**

(-- addr)

addr is a **USER** variable that is used to signal header generators like *e*.g. **CODE** and : to perform non-default alignment. The amount of alignment is controlled by the byte array _align_[7]. Typically, **ALIGNS?** is set to **ON** in the **CREATE** part of defining words. All header generators automatically reset **ALIGNS?** after use. The programmer only needs to "ALIGNS? ON" in the **CREATE** part of a new defining word. Note that this word will go away when iForth starts to rigorously separate code and data areas (post-Hammer era).

See also: **_calign_ _align_ ALIGN64 ALIGN32**

ALITERAL **IFORTH**

Compilation: (a-addr --)

Compile *a-addr* as a literal. This word is equivalent to **LITERAL** however a special optimization is used to generate a code sequence that is much shorter and results in relocatable code that can be saved to disk. Note that this word is not compile-only.

Execution: (-- a-addr)

Place *a-addr* on the stack.

See also: **ILITERAL LITERAL**

ALLOCATE **MEMORY**

(u -- a-addr ior)

Allocate *u* address units of contiguous data space. The address of the next available data space location is unaffected by this operation. The initial content of the allocated space is undefined. If the allocation succeeds, *a-addr* is the aligned starting address of the allocated space and *ior* is 0. If the operation fails, *a-addr* does not represent a valid address and *ior* is the implementation-defined **I/O** result code. Since all processes allocate memory from the same pool this word is protected with a semaphore to prevent collisions. However the presence of other processes makes the use of **AVAILABLE** somewhat insecure.

See also: **AVAILABLE FREE RESIZE INIT-MEM**

ALLOT **CORE**

(n --)

Reserve *n* address units of data space. This space is located between **HERE** and **NP @** and is a relatively expensive resource. For large areas you are advised to use **ALLOCATE** if possible.

See also: **ALLOCATE**

ALSO **SEARCH EXT**

(--)

Transform the search order consisting of *wid1*, ... *widn*-1, *widn* (where *widn* is searched first) into *wid1*, ... *widn*-1, *widn*, *widn*. If there are too many word lists in the search order the system will print the message 'search path full' and abort. Other ANS systems may react in another way on this condition.

ALT-used! **IFORTH**

(bool --)

Instructs the optimizer to use an alternate **NEXT** code sequence. Set up in the ~/include/iforth.prf preferences file.

See also: **ALT-used@ FPU-ovf! LEA-used! P6-used! EDX-used@ EDI-used@**

ALT-used@ **IFORTH**

(-- bool)

Get the current state of **NEXT** sequence handling.

See also: **ALT-used!**

AND **CORE**

(x1 x2 -- x3)

x3 is the bit-by-bit logical 'and' of *x1* with *x2*.

ANSI **IFORTH**

(-- a-addr)

a-addr is the address of **ANSI** . **ANSI** contains a flag that, when true, causes iForth to produce warnings when keywords or language constructions are used that can possibly fail to run on other ANS Forth Systems. The warning message reads: "xxx is not portable". Note that the minimal allowed standard Forth only needs to implement the words from the CORE wordset. There are no warnings for words that are not available on such minimal systems. The warnings are only issued for words that are compiled.

ANSI? **IFORTH**

(dea --)

dea is the address of a dictionary entry. If this is not a definition described by the ANS Forth standard and portability checking is on, a warning message is printed.

See also: **ANSI HEAD'**

ASHR **IFORTH**

(x1 x2 -- x3)

Arithmetically shift *x1* over *x2* bits to the right, giving the result *x3*. The sign bit is unchanged.

ASM{ **IFORTH**

(--)

Between **ASM{** and **}ASM** we are in the **ASSEMBLER** vocabulary, in interpretative mode. Because the action of "[" is specified in **ASM{** , the compiler flushes all stacks to memory. This, of course, includes the floating-point stack. By default the compiler assumes empty stacks when the word **}ASM** executes. You can use **IN/OUT** and **ADJUST-STACK** to modify this. The entry of Saturday, December 30, 2000, 12:52 AM in the ./bugs.txt file gives more information.

See also: **}ASM IN/OUT ADJUST-STACK FIN/FOUT**

ASSEMBLER **TOOLKIT EXT**

(--)

Replace the first word list in the search order with the **ASSEMBLER** word list.

AT-XY "at-x-y" **FACILITY EXT**

(u1 u2 --)

Perform steps so that the next character output will appear in column *u1*, row *u2* of the current output device. The upper left corner of the output device is row zero, column zero. It is a no-op when the operation cannot be performed on the current output device with the specified parameters. Note that for other implementations the result in that case is an ambiguous condition.

AVAILABLE **MEMORY EXT**

(-- u)

Return the number of address units contained in the largest contiguous region of data space that may be allocated by **ALLOCATE** or **RESIZE** . This word is safe for use in a multi-process environment. Note that in that case it is not guaranteed that the amount returned by **AVAILABLE** is available to use with **ALLOCATE** or **RESIZE** .

See also: **ALLOCATE FREE RESIZE**

AWARNING "a-warnings" **IFORTH**

(-- a-addr)

a-addr is the address of **AWARNING** . **AWARNING** contains a flag that, when true, causes iForth to produce warnings about assembler instructions that are not executable on the processor iForth is currently compiling for.

See also: **#CPU**

B! "b-store" **IFORTH**

(16bit addr --)

Put a 16-bit number at any *addr*. Note the difference with L! . Used to manipulate 16-bit VGA display memory.

B. "b-dot" **IFORTH**

(x --)

Print *x* as a 2 digit hexadecimal number.

B@ "b-fetch" **IFORTH**

(addr -- 16bit)

Get a 16-bit number from *addr*. Note the difference with L@ . Used to read from 16-bit VGA display memory.

B@+ "b-store-plus" **IFORTH**

(addr -- addr' 16bit)

Get a 16-bit number from *addr*. Also leave the incremented address *addr'*. Used to read from 16-bit VGA display memory.

BASE **CORE**

(-- a-addr)

a-addr is the address of **BASE** . **BASE** contains the current number conversion radix

{{ 2 .. 72 }}.

BEGIN **C** **CORE**

Compilation: (-- dest)

Put the next location for a transfer of control, *dest*, onto the control flow stack.

Execution: (--)

Continue execution.

See also: **REPEAT UNTIL WHILE**

BEGIN, **IFORTH-ASSEMBLER**

Assembling: (-- addr)

Leave the contents of the current codespace pointer on the data stack. This address can later be used to resolve a backward branch. This backward branch is usually generated by AGAIN, REPEAT, or UNTIL, .

Execution: (--)

Do nothing.

See also: **AGAIN, REPEAT, UNTIL,**

BIN **IFORTH**

(fam2 -- fam1)

fam1 is a file access method such as **R/W R/O** or **W/O** . *fam2* is a file access method with the same properties as *fam1* but for which it is guaranteed that read or write actions to the opened file do not have any character translation. Files opened with **BIN** applied on their file access method are usually called 'binary files'. Binary mode can only be used in combination with **WRITE-FILE** and **READ-FILE** , use of **READ-LINE** and **WRITE-LINE** gives unpredictable results.

See also: **CREATE-FILE OPEN-FILE READ-FILE WRITE-FILE**

BINARY **IFORTH**

(--)

Sets contents of **BASE** to two.

BL "b-l" **CORE**

(-- char)

char is the character value for a space.

BLANK **STRING**

(c-addr u --)

If *u* is greater than zero, store the character value for space in *u* consecutive character positions beginning at *c-addr*.

BLK "b-l-k" **BLOCK**

(-- a-addr)

a-addr is the address of **BLK** . **BLK** contains the number of the mass storage block being interpreted as the input stream {{ 0 .. one less than the number of blocks available }}. If **BLK** contains zero, the input stream is being taken from the text input buffer specified by **TIB** .

BLOCK **BLOCK**, **FILE**

(u -- a-addr)

a-addr is the address of the first character of the block buffer assigned to mass storage block *u*. An ambiguous condition exists if *u* is not an available block number. If block *u* is already in a block buffer: *a-addr* is the address of that block buffer. If block *u* is not already in memory and there is an unassigned block buffer: transfer block *u* from mass storage to an unassigned block buffer. *a-addr* is the address of that block buffer. If block *u* is not already in memory, and there are no unassigned block buffers: unassign a block buffer. If the block in that buffer has been **UPDATE** -d, transfer the block to mass storage and transfer block *u* from mass storage into that buffer. *a-addr* is the address of that block buffer. At the conclusion of the operation, the block buffer pointed to by *a-addr* is the current block buffer and is assigned to *u*. An **UPDATE** -d block that came from a file must be transferred back to the same file when the block buffer is needed for another block. If the value stored in **BLOCK-FID** is not equal to zero, the block number *u* references block *u* of the file identified by the *fileid* stored in **BLOCK-FID** . **BLOCK** is also an environment query.

See also: **ENVIRONMENT?**

BLOCK-EXT **IFORTH**

BLOCK-EXT is an environment query.

See also: **ENVIRONMENT?**

BLOCK-FID "block-f-i-d" **FILE**

(-- a-addr)

Return the address of a *fileid*. The identified file contains blocks; block requests made by words in the **BLOCK** word set use the blocks in this file. If **BLOCK-FID** contains zero, block requests made by words in the **BLOCK** word set use an implementation-defined default block space.

BODY> "from-body" **IFORTH**

(dfa -- xt)

xt is the execution token corresponding to a data field address dfa for a word defined via **CREATE** .

BOOTLINK **IFORTH**

(-- addr)

This variable has no real function in iForth. (See the DFW tForth product).

BOUNDS **IFORTH**

(n1 n2 -- n3 n4)

A conventional equivalent to **OVER** + **SWAP** . Use with **DO** .

BREAD-LINE **IFORTH**

(c-addr u1 fileid -- u2 flag ior)

A synonym for **READ-LINE** . **BREAD-LINE** may perform more buffering than the standard **READ-LINE** and might therefore be faster.

See also: **READ-LINE**

BREAK? **IFORTH**

(-- flag)

Wait for a key to be pressed. If the key is the **ESC** key, return true, otherwise return false.

See also: **WAIT?**

BSWAP **IFORTH**

(32bits1 -- 32bits2)

Swap all bytes of 32bits1, *i.e.* a+*b*+*c*+*d* -> *d*+*c*+*b*+a.

See also: **><**

BUFFER **BLOCK**, **FILE**

(u -- a-addr)

a-addr is the address of the first character of the block buffer assigned to block *u*. The contents of the block are unspecified. An ambiguous condition exists if *u* is not an available block number. If block *u* is already in a block buffer: *a-addr* is the address of that block buffer. If block *u* is not already in memory and there is an unassigned buffer: *a-addr* is the address of that block buffer. If block *u* is not already in memory, and there are no unassigned block buffers: unassign a block buffer. If the block in that buffer has been **UPDATE** -d, transfer the block to mass storage. *a-addr* is the address of that block buffer. At the conclusion of the operation, the block buffer pointed to by *a-addr* is the current block buffer and is assigned to *u*. An **UPDATE** -d block that came from a file must be transferred back to the same file when the block buffer is needed for another block. If the value stored in **BLOCK-FID** is not equal to zero, the block number *u* references block *u* of the file identified by the *fileid* stored in **BLOCK-FID** .

BYE **TOOLKIT EXT**

(--)

Return control to the host operating system.

BYE! **IFORTH**

(--)

Send the 'disconnect server' command over the boot link to the server (if there is one). The server disconnects upon receiving this message and exits itself. All knowledge about open files is lost. If there is no server this command is the same as **BYE** .

See also: **BYE**

C! "c-store" **CORE**

(char c-addr --)

Store *char* at *c-addr*.

C" "c-quote" **CORE EXT**

Compilation: ("ccc<">" --)

Parse characters *ccc* delimited by " (double-quote) and append the execution semantics given below to the current definition.

Execution: (-- c-addr)

Return *c-addr*, a counted string consisting of the characters *ccc*. A standard program may not alter the returned string. Note that in iForth this word can be used when interpreting from the terminal.

C&! "c-and-store" **IFORTH**

(char c-addr --)

And *char* with the value at *c-addr* and store again at *c-addr*.

See also: **C^! C|! C+! C!**

C+! "c-plus-store" **IFORTH**

(char c-addr --)

Add *c* to the character stored at *c-addr*.

C, "c-comma" **CORE**

(char --)

Reserve space for one character in the data space and store *char* in the space. An ambiguous condition exists if the address of the next available data space location is not character-aligned.

C-! "c-and-store" **IFORTH**

(char c-addr --)

Subtract *char* from the value at *c-addr* and store result at *c-addr*.

See also: **C+!**

C0! "c-zero-store" **IFORTH**

(c-addr --)

Store 0 in the character cell at address *c-addr*.

C@ "c-fetch" **CORE**

(c-addr -- char)

Fetch the character stored at *c-addr*.

C@+ "c-fetch-plus" **IFORTH**

(c-addr1 -- c-addr2 c)

Fetch *c* from the character address *c-addr1*. Add 1 **CHARS** to *c-addr1* giving *c-addr2*.

C@- "c-fetch-minus" **IFORTH**

(c-addr1 -- c-addr2 c)

Fetch *c* from the character address *c-addr1*. Subtract 1 **CHARS** from *c-addr1* giving *c-addr2*.

CALL, "call-comma" **IFORTH**

(xt --)

Compile a processor assembler subroutine call to the routine identified by the execution token *xt*.

See also: **COMPILE,**

CALLBACK **IFORTH**

(addr #args "name" --)

This is a CREATEing word that generates a word called *name*. (The stack diagram applies to **CREATE** time.) An interface is build from a normal Forth word to an external library or

program. The result is that external code can call Forth words directly and asynchronously. The address of the Forth word to be called in this way is given by *addr*. The number of cell-sized parameters the external code pushes on the normal Forth stack is given by *#args*. Assuming the external code is written in "**C**", the Forth word at address expects the **C** arguments in the order written in the **C** documentation (*i.e.* right-most on TOS). See **PASCAL-CALLBACK** for an alternative convention. The ebp, ebx, ecx, edx, esi and edi registers are saved across a CALLBACK. The CREATEd word *name* can be executed from Forth: (*#args**{*n*} -- *u*) The arguments are to be consumed and a single result is to be left on top of the stack. The temporary return, local, and extra stack are 256 words deep. User area and (because the fp stack pointer is in the user area) fp stack are inherited from the current task. A re-entrant callback is therefore not possible. A severe limitation is that the stackpointers are not reflected in **SP0 RP0** etc. (because these are in the user area again), so **.S** won't work.

See also: **FCALLBACK PASCAL-CALLBACK FOREIGN FFOREIGN DFOREIGN VFOREIGN**

CALM **IFORTH**

(?? -- ??)

CALM is a machinecode routine that does not use any of the iForth stacks or datastructures: these do not exist or are instable when **CALM** is called straight from the DOS-extender's exception handler. **CALM** waits for a keypress. If this is 'E' (uppercase E) iForth exits to the **OS** with errorlevel $FF. Any other key and **ABORT** will be executed instead. **CALM** is meant for execution by **PANIC** in case of a hardware exception.

See also: **'PANIC PANIC**

CASE **C** **CORE EXT**

Compilation: (-- case-sys)

Mark the start of the **CASE** ... **OF** ... **ENDOF** ... **ENDCASE** structure.

Execution: (--)

Continue execution.

See also: **ENDCASE ENDOF OF**

CASESENSITIVE **IFORTH**

(-- a-addr)

a-addr is the address of **CASESENSITIVE** . **CASESENSITIVE** contains a flag that, when false, makes **FIND** case insensitive. *i.e.* words with the same spelling but not the same case are then considered equal.

CATCH **ERROR**

*(i*x xt -- j*x 0)* No THROW received*, or (i*x xt -- i*x n)* THROW received.

Pushes an error interception frame on the return stack. That frame remembers the current stack depth, and then executes the execution token *xt* (as with **EXECUTE**) in such a way that control may be transferred back to a point just after **CATCH** if **THROW** is executed during the execution of *xt*. If the execution of *xt* completes normally (*i.e.*, this **CATCH** does not receive a **THROW**) **CATCH** pops the error frame and returns 0 above whatever stack items would have been returned by *xt* **EXECUTE** . If this **CATCH** receives a throw, it returns a non-zero *n* above an indeterminate number *i* of stack items. The depth of the stack is now the same as it was just before **CATCH** began execution. The values of the *i*x* stack items could have been modified during the execution of *xt*. In general, there is nothing useful that can be done with those stack items. Since the stack depth is known, the application may **DROP** those items.

CELL+ "cell-plus" **CORE**

(a-addr1 -- a-addr2)

Add the size of a cell, specified in address units, to *a-addr1*, giving *a-addr2*.

CELL- "cell-minus" **IFORTH**

(a-addr1 -- a-addr2)

Subtract the size of a cell, specified in address units, from *a-addr1*, giving *a-addr2*.

CELL/MOD "cell-slash-mod" **IFORTH**

(x1 -- x2 x3)

Divide *x1* by the size of one cell in bytes, giving the single-cell remainder *x2* and the single-cell quotient *x3*.

CELLS **CORE**

(n1 -- n2)

n2 is the size, in address units, of *n1* cells.

CELL[] **IFORTH**

(addr1 index -- addr2)

Equivalent to **CELLS** + .

See also: **[]CELL**

CHAR "care" **CORE**

("ccc< >" -- char)

Parse characters *ccc* delimited by a space, ignoring leading delimiters. Put the integer value of its first character onto the stack.

CHAR+ "char-plus" **CORE**

(c-addr1 -- c-addr2)

Add the size of one character, specified in address units, to *c-addr1*, giving *c-addr2*.

CHAR- "char-minus" **IFORTH**

(c-addr1 -- c-addr2)

Subtract the size of one character, specified in address units, from *c-addr1*, giving *c-addr2*.

CHARS "chars" **CORE**

(n1 -- n2)

n2 is the size, in address units, of *n1* characters.

CIRCULAR **IFORTH**

(n mod -- n')

Equivalent **>S S>D S> FM/MOD DROP** .

CLEAR **IFORTH**

CLEAR is an alias for 0TO.

CLINK **IFORTH**

('vochead --)

Links a vocabulary into the start-up initialization of iForth. Only needed for user vocabularies that are to stay in an executable made with **SAVE-SYSTEM** .

CLOSE-FILE **FILE**

(fileid -- ior)

Close the file identified by *fileid*. *ior* is the implementation-defined **I/O** result code.

CLS "c-l-s" **IFORTH**

(--)

Clear the terminal screen.

See also: **PAGE**

CMOVE "c-move" **STRING**

(c-addr1 c-addr2 u --)

If *u* is greater than zero, copy *u* consecutive characters from *c-addr1* to *c-addr2*. If *c-addr2* lies within the source region, memory propagation occurs. (*c-addr2* lies within the source region if *c-addr2* is not less than *c-addr1* and *c-addr2* is less than the quantity *c-addr1 u* **CHARS** +). Contrast with: **CMOVE>**

See also: **MOVE**

CMOVE> "c-move-up" **STRING**

(c-addr1 c-addr2 u --)

If *u* is greater than zero, copy *u* consecutive characters from *c-addr1* to *c-addr2*. If *c-addr1* lies within the destination region, memory propagation occurs. (*c-addr1* lies within the destination region if *c-addr1* is greater than or equal to *c-addr2* and if *c-addr2* is less than the quantity *c-addr1 u* **CHARS** +). Contrast with: **CMOVE**

See also: **MOVE**

CODE **D** **TOOLKIT EXT**

("name" -- sys)

Parse *name* delimited by a space, ignoring leading delimiters. Create a dictionary entry for *name* with the execution semantics defined below. Add the **ASSEMBLER** word list to the search order to process the words between *name* and **END-CODE** . *sys* is balanced by the corresponding **END-CODE** . *name* is called a "code definition."

Execution: ("name" --)

Do nothing. Typically, the execution semantics of *name* are extended by compiling additional words into the code definition.

COLD **IFORTH**

*(i*x --)*

Effectively restart iForth. All stacks are cleared, all definitions are removed from the wordlist, all internal variables are reset to their original value and the memory manager is reset. Resetting the memory manager takes away all allocated memory from processes that are still running, possibly causing a crash later. This might be avoided by first FORGETting all user added words, so that the special `forget code' on a per word basis is executed and the system is properly cleaned up. If the **FORGET>** parts of the words initiating I/O, process creation, and

memory allocation are written with care, resetting the system becomes a harmless operation. However, this will not be an easy task in a multi-processor environment. Finally the sign-on banner is displayed and the execution token in **'BOOT** is executed. If **'BOOT** contains 0 (the default) then **QUIT** is called.

See also: **FORGET>**

COMPARE **STRING**

(c-addr1 u1 c-addr2 u2 -- n)

Compare the string specified by *c-addr1* and *u1* to the string specified by *c-addr2* and *u2*. The strings are compared, beginning at the given addresses, character by character up to the length of the shorter string, or until a difference is found. If both strings are the same up to the length of the shorter string, then the longer string is greater than the shorter string. *n* is -1 if the string specified by *c-addr1* and *u1* is less than the string specified by *c-addr2* and *u2*. *n* is zero if the strings are equal. *n* is 1 if the string specified by *c-addr1* and *u1* is greater than the string specified by *c-addr2* and *u2*.

COMPILE **IFORTH**

("name" --)

Parse *name* delimited by a space. If *name* is not a word that can be found in the current search list, issue an error. Compile a reference to *name*. This word is present for portability reasons only. It is recommended to use **POSTPONE** and **COMPILE,** .

See also: **[COMPILE] COMPILE, POSTPONE ['] EVALUATE EVAL"**

COMPILE, "compile-comma" **C** **CORE EXT**

(xt --)

Append the execution semantics of the definition represented by *xt* to the execution semantics of the current word definition. An ambiguous condition exists if **COMPILE,** is executed while interpreting.

See also: **[COMPILE] COMPILE POSTPONE ['] EVALUATE EVAL"**

COMPILE-ONLY **IFORTH**

(--)

Marks the last word that is created as a compile-only word.

CONST-DATA **IFORTH**

(addr size --)

Declare the area defined by *addr* size as containing constant data. The optimizer will from that point on treat code that does "area **@**" or "area **C@**" in a special way. See ./include/fsl-util.frt for example usage.

CONSTANT **D** **CORE**

(x "name" --)

Parse *name* delimited by a space, ignoring leading delimiters. Create a dictionary entry for *name* with the execution semantics defined below. *name* is referred to as a "constant."

Execution: ("name" -- x)

Place *x* on the stack.

CONTEXT **IFORTH**

(-- a-addr)

a-addr is the address of **CONTEXT** . **CONTEXT** contains the address of the field in the active wordlist where the *dea* of the last word in that list is kept. The active wordlist is the wordlist that is first in the search order. Example:

```
ALSO FORTH DEFINITIONS
: SayHello ." Hello!";
CONTEXT @ @ .ID
SayHello ok
```

The first **@** fetches the address where the *dea* is stored, the second **@** fetches the *dea* itself. **.ID** prints the name of that dictionary entry.

See also: **CURRENT HEAD'**

CONVERT **CORE EXT**

(ud1 c-addr1 -- ud2 c-addr2)

ud2 is the result of converting the characters within the text beginning at the first character after *c-addr1* into digits, using the number in **BASE** , and adding each digit to *ud1* after multiplying *ud1* by the number in **BASE** . Conversion continues until a character that is not convertible is encountered. *c-addr2* is the location of the first unconverted character. An ambiguous condition exists if *ud2* overflows. Note: This word is obsolescent and is included as a concession to existing implementations. Its function is superseded by **>NUMBER** . **CONVERT** does not support all of the extensions that are supported by **>NUMBER** .

See also: **>NUMBER**

CORE **IFORTH**

CORE is an environment query.

See also: **ENVIRONMENT?**

CORE-EXT **IFORTH**

CORE-EXT is an environment query.

See also: **ENVIRONMENT?**

COUNT **CORE**

(c-addr1 -- c-addr2 u)

Return the character string specification for the counted string stored at *c-addr1*. *c-addr2* is the address of the first character after *c-addr1*. *u* is the contents of the character at *c-addr1*, which is the length in characters of the string at *c-addr2*.

CP **IFORTH**

(-- a-addr)

a-addr is the address of **CP** . **CP** contains the pointer to just past the last byte of the code space. Any bytes allocated in the code space are placed at this address. This variable is adjusted by **ALLOT** and other words that use **ALLOT** .

See also: **HERE**

CR "c-r" **CORE**

(--)

Cause subsequent output to appear at the beginning of the next line.

CREATE **D** **CORE**

("name" --)

Parse *name* delimited by a space, ignoring leading delimiters. Create a dictionary entry for *name* with the execution semantics defined below. If the address of the next available data space location is not aligned, reserve enough data space to align it. This address defines *name*'s data field. **CREATE** does not allocate space in *name*'s data field.

Execution: ("name" -- a-addr)

a-addr is the address of *name*'s data field. An additional set of operations may be defined using **DOES>** and **FORGET>** .

CREATE-FILE **FILE**

(c-addr u fam -- fid ior)

Create the file named in the character string specified by *c-addr* and *u*, and open it with file access method *fam*. The file access methods are **R/W R/O** and **W/O** . Each of these may be followed by **BIN** , meaning a binary file should be created. If a file by the same name already exists, recreate it as an empty file. If the file was successfully created and opened, *ior* is zero, *fid* is the *fileid*, and the file has been positioned to the start of the file. Otherwise, *ior* is the implementation-defined **I/O** result code and *fid* is an unspecified value.

See also: **ERROR>TEXT** **OPEN-FILE**

CS-PICK **C** **TOOLKIT EXT**

Compilation: (sys(u) ... sys(1) sys(0) u -- sys(u) ... sys(1) sys(0) sys(u))

Remove *u*. Copy the *u*th element of the control flow stack to the top of the control flow stack. An ambiguous condition exists if there are less than *u*+2 items on the control flow stack before **CS-PICK** is executed, or if **CS-PICK** is executed while interpreting.

See also: **CS-ROLL**

CS-ROLL **C** **TOOLKIT EXT**

Compilation: (sys(u) sys(u-1) ... sys(0) u -- sys(u-1) ... sys(0) sys(u))

Remove *u*. Rotate *u*+1 elements on top of the control flow stack. An ambiguous condition exists if there are less than *u*+2 entries on the stack before **CS-ROLL** is executed or if **CS-ROLL** is executed while interpreting.

See also: **CS-PICK**

CTRL "control" **IFORTH**

("ccc< >" -- c)

Parse characters *ccc* delimited by a space, ignoring leading delimiters. Put the lowest 5 bits of the integer value of its first character onto the stack. *e*.g. **CTRL** E places the value 5 onto the stack which is also ^E. Because of iForth's augmented **>NUMBER** you may also enter ^E directly.

See also: **>NUMBER**

CURDIM "cur-dim" **IFORTH**

(-- a-addr)

Returns the address of the variable in which top and bottom line numbers of the present hardware cursor are stored. The first byte contains the top line, the second byte the bottom line. The default is top=6, bottom=7, giving a thin line in the middle of the cursor cell. An

updated **CURDIM** will come into effect when function #29 of **!TERMINAL** is subsequently executed (set the cursor).

CURRENT **IFORTH**

(-- a-addr)

a-addr is the address of **CURRENT** . **CURRENT** contains a pointer to the control structure associated with the wordlist in which new words are added. The structure of this control structure is implementation defined. In iForth **CURRENT @ @** returns the *dea* of the word defined last (unless **SET-CURRENT** has been used in between).

See also: **CONTEXT HEAD'**

C^! "c-xor-store" **IFORTH**

(char c-addr --)

Xor *char* with the value at *c-addr* and store result at *c-addr*.

See also: **C&! C|! C+! C! C-!**

C|! "c-or-store" **IFORTH**

(char c-addr --)

Or *char* with the value at *c-addr* and store result at *c-addr*.

See also: **C&! C^! C+! C! C-!**

D! "d-store" **DOUBLE EXT**

(d a-addr --)

Store *d* at *a-addr*.

D+ "d-plus" **DOUBLE**

(d1|ud1 d2|ud2 -- d3|ud3)

Add *d2*|*ud2* to *d1*|*ud1*, giving the sum *d3*|*ud3*.

D+! "d-plus-store" **IFORTH**

(d|ud a-addr --)

Add *d*|*ud* to the double-cell number at *a-addr*.

D- "d-minus" **DOUBLE**

(d1|ud1 d2|ud2 -- d3|ud3)

Subtract *d2*|*ud2* from *d1*|*ud1*, giving the difference *d3*|*ud3*.

D-! "d-minus-store" **IFORTH**

(d|ud a-addr --)

Subtract *d*|*ud* to the double-cell number at *a-addr*.

D. "d-dot" **DOUBLE**

(d --)

Display *d* in free field format.

D.R "d-dot-r" **DOUBLE**

(d n --)

Display *d* right aligned in a field |*n*| characters wide. If the number of characters required to display *d* is greater than |*n*|, all digits are displayed with no leading spaces in a field as wide as necessary. (In **D.R** , R stands for RIGHT). When *n* is negative the fill character will be '0', not **BL** .

See also: **UD.R (D.R) (UD.R) .R**

D0! "d-zero-store" **IFORTH**

(a-addr --)

Store 0 in the double-cell at *a-addr*.

D0< "d-zero-less" **DOUBLE**

(d -- flag)

flag is true if *d* is less than zero.

D0= "d-zero-equals" **DOUBLE**

(d|ud -- flag)

flag is true if *d*|*ud* is equal to zero.

D2* "d-two-star" **DOUBLE**

(d1 -- d2)

Multiply *d1* by 2 giving *d2*. Since the processor is a two's-complement computer this word can be used to perform a left shift over 1 bit. Note that other implementations of the ANS standard might not have this feature.

D2/ "d-two-slash" **DOUBLE**

(d1 -- d2)

d2 is the result of dividing *d1* by two.

D< "d-less-than" **DOUBLE**

(d1 d2 -- flag)

flag is true if *d1* is less than *d2*.

D<> "d-not-equal" **IFORTH**

(d1 d2 -- f)

Tests to see if *d1* and *d2* are different. Equivalent to **D- OR** .

D= "d-equals" **DOUBLE**

(xd1 xd2 -- flag)

flag is true if *xd1* is bit-for-bit the same as *xd2*.

D> "d-greater" **IFORTH**

(d1 d2 -- flag)

flag is true when *d1* is greater than *d2*.

D>F "d-to-f" **FLOAT**

(d --) (F: -- r)

r is the floating-point equivalent of *d*. An ambiguous condition exists if *d* cannot be precisely represented as a floating-point value.

D>S "d-to-s" **DOUBLE**

(d -- n)

n is the equivalent of *d*. An ambiguous condition exists if *d* lies outside the range of a signed single-cell number.

D@ "d-fetch" **DOUBLE EXT**

(a-addr -- d)

d is the value stored at *a-addr*.

DABS "d-abs" **DOUBLE**

(d -- ud)

+ud is the absolute value of *d*. Note that on 2's complement systems such as the Intel '386 the most negative number does not have an absolute value.

DATASEG "data-seg" **IFORTH**

(-- selector)

Returns the *selector* for the iForth data segment. This is occasionally useful in a **CODE** definition.

DATE "date" **IFORTH**

(-- day month year)

Place the numbers of the current day, month and year on the data stack. *day* is in the range { 1 .. 31 }, *month* is in the range { 1 .. 12 }, *year* is the year including the century.

DEC. "dec-dot" **IFORTH**

(x --)

Print *x* as a decimal number. The current number base is not changed.

DECIMAL **CORE**

(--)

Set the numeric conversion radix to ten (decimal).

DEFINITIONS **SEARCH**

(--)

Make the compilation word list the same as the first word list in the search order. Specifies that the names of subsequent definitions will be placed in the compilation word list. Subsequent changes in the search order will not affect the compilation word list.

DELETE-FILE **FILE**

(c-addr u -- ior)

Delete the file named in the character string specified by *c-addr u*. *ior* is the implementation-defined **I/O** result code.

See also: **ERROR>TEXT**

DENOTATION **IFORTH**

(--)

Used like *e*.g. **IMMEDIATE** . Apply a mask to a word's flags that signals to the optimizer that the word scans the input stream and takes full responsibility for that. Normally the optimizer turns off tokenization when a word changes **>IN** . Example **DENOTATION** words: **[CHAR] S"**

See also: **=DENOTATION =TOKENIZE IMMEDIATE**

DEPRIVE **IFORTH**

(--)

Traverses the wordlist to which definitions are being added and modifies the headers of all words **PRIVATE** was applied to. This process stops when the first definition is found that got compiled after the execution of **PRIVATES** . The result of the header modification is that words are effectively made invisible to the iForth interpreter/compiler. The process cannot be undone.

See also: **PRIVATES PRIVATE** 'PRIVATE **HIDE**

DEPTH **CORE**

(-- +n)

+*n* is the number of one cell values contained in the data stack before +*n* was placed on the stack.

DF! "d-f-store" **FLOAT EXT**

(a-addr --) (F: r --)

Store the floating-point number *r* as a 64-bit IEEE double precision number at *a-addr*. Note that in other implementations of the ANS standard *r* may have more digits of precision or may be too large for representation as a 64-bit IEEE number, so rounding or overflow might occur.

DF!+ "d-f-store-plus" **IFORTH**

(a-addr1 -- a-addr2) (F: r --)

Store the floating-point number *r* as a 64-bit IEEE double precision number at *a-addr1*, and leave the incremented pointer as *a-addr2*. Note that in other implementations of the ANS standard *r* may have more digits of precision or may be too large for representation as a 64-bit IEEE number, so rounding or overflow might occur.

See also: **DF! SF!+ XF!+**

DF+! "d-f-plus-store" **IFORTH**

(F: r --) (a-addr --)

Add the IEEE double *r* to the double at *a-addr*.

DF+!+ "d-f-plus-store-plus" **IFORTH**

(F: r --) (a-addr1 -- a-addr2)

Add the IEEE double *r* to the double at *a-addr1* and leave the incremented pointer as *a-addr2*.

See also: **DF+!**

DF, "d-f-comma" **IFORTH**

(--) (F: r --)

Reserve the size of a double precision IEEE floating-point number in the data-space and store *r* in that space. An ambiguous condition exists if the address of the next available data space location is not aligned.

See also: **F, SF,**

DF-! "d-f-minus-store" **IFORTH**

(F: r --) (a-addr --)

Subtracts the IEEE double *r* from the double at *a-addr*.

DF@ "d-f-fetch" **FLOAT EXT**

(a-addr --) (F: -- r)

Fetch the 64-bit IEEE double precision number stored at *a-addr* to the floating-point stack as *r* in the internal representation. If the IEEE double precision significand has more precision than the internal representation it will be rounded to the internal representation using the "round to nearest" rule. If the exponent of the IEEE double precision representation is too large the bit representation for infinite with the correct sign will be pushed on the floating-point stack.

DF@+ "d-f-fetch-plus" **IFORTH**

(a-addr1 -- a-addr2) (F: -- r)

Fetch the 64-bit IEEE double precision number stored at *a-addr1* to the floating-point stack as *r* in the internal representation. Also leave the incremented pointer as *a-addr2*. If the IEEE double precision significand has more precision than the internal representation it will be rounded to the internal representation using the "round to nearest" rule. If the exponent of the IEEE double precision representation is too large the bit representation for infinite with the correct sign will be pushed on the floating-point stack.

See also: **DF@**

DFALIGN **FLOAT**

(--)

If the address of the next available data space location is not aligned, reserve enough space to align it to a location which is suitable to hold a IEEE double precision floating point number.

DFALIGNED **ALIGN**

(addr -- a-addr)

a-addr is the first aligned address greater than or equal to *addr* and which is suitable to reference a IEEE double precision floating point number.

DFLOAT+ "d-float-plus" **FLOAT**

(a-addr1 -- a-addr2)

Add the size of an IEEE double precision floating point number, specified in address units, to *a-addr1*, giving *a-addr2*.

DFLOAT- "d-float-minus" **IFORTH**

(a-addr1 -- a-addr2)

Subtract the size of an IEEE double precision floating point number, specified in address units, to *a-addr1*, giving *a-addr2*.

DFLOATS "d-floats" **FLOAT**

(n1 -- n2)

n2 is the size, in address units, of *n1* IEEE double precision floating point numbers.

DFLOAT[] **IFORTH**

(addr1 index -- addr2)

Equivalent to **DFLOATS** + .

See also: **[]DFLOAT**

DFOREIGN **IFORTH**

(n{u} n 'service -- ud)*

Calls a dynamically linked "**C**" library routine at address *'service*, with *n* unsigned arguments *u*. The result is a 64bit integer *ud*.

See also: **FOREIGN CALLBACK FPUSHS FPUSHD**

DFVARIABLE "d-f-variable" **D** **FLOAT**

("name" --)

Parse *name* delimited by a space, ignoring leading delimiters. Create a dictionary entry for *name* with the execution semantics defined below. Reserve 1 **DFLOATS** address units of data space at an aligned address. *name* is referred to as an "d-f-variable."

Execution: ("name" -- a-addr)

a-addr is the address of the data space reserved by **DFVARIABLE** when it created *name*. The application is responsible for initializing the contents of the reserved space.

DIGIT **IFORTH**

(u -- c)

c is the alphanumeric character that represents the value *u*. The current number base is not checked.

DIGIT? **IFORTH**

(c base -- u flag)

u is the value represented by the character alphanumeric character *c*. *flag* is true when *c* is a valid character under the number base *base*, *i.e. u* is in { 0 .. base-1 }, otherwise *flag* is false.

DLOCAL "d-local" **C** **IFORTH**

Compilation: ("name" --)

Parse *name* delimited by a space, ignoring leading delimiters. Create a temporary dictionary entry for *name* with the execution semantics defines below. **DLOCAL** can only be used inside a definition. *name* remains in the dictionary until the current definition is finished with ; **DOES>** or **;CODE** .

Execution: ("name" --) (D: -- d)

Place the double number *d* on the floating-point stack. The value of *d* is unspecified until the phrase *d* **TO** *name* is executed, causing *d* to be associated with *name*.

DLOCALS| **C** **IFORTH**

*Compilation: ("name"*n "|" --)*

Parse names delimited by a blank, ignoring leading delimiters. The list of names is terminated by a '|' (bar). Create a temporary dictionary entry for each *name* with the execution semantics defined below. **DLOCALS|** can only be used inside a definition. The names remain in the dictionary until the current definition is finished with ; **;CODE** or **DOES>** . iForth accepts an unlimited number of names. The ANS standard requires a minimum of eight.

Execution: ("name" -- d)

Place *d* on the stack. The value of *d* is unspecified until the phrase *d* **TO** *name* is executed, causing *d* to be associated with *name*.

See also: **LOCALS|**

DLSHIFT "d-l-shift" **IFORTH**

(d1 x -- d2)

Logically shift *d1* over *x* bits to the left, giving the result *d2*. A 0 bit is inserted at the right end of the value.

DMAX "d-max" **DOUBLE**

(d1 d2 -- d3)

d3 is the greater of *d1* and *d2*.

DMIN "d-min" **DOUBLE**

(d1 d2 -- d3)

d3 is the lesser of *d1* and *d2*.

DNEGATE "d-negate" **DOUBLE**

(d1 -- d2)

d2 is the negation of *d1*.

DO **C** **CORE**

Compilation: (-- dodest)

The next location for a transfer of control (*dodest*) goes onto the control flow stack. Append the execution semantics given below to the current definition.

Execution: (n1|u1 n2|u2 --) (R: -- sys)

Set up loop control parameters with index *n2|u2* and limit *n1|u1*. An ambiguous condition exists if *n1|u1* and *n2|u2* are not both the same type. Anything already on the return stack becomes unavailable until the loop control parameters are discarded.

See also: **+LOOP LOOP UNLOOP LEAVE**

DOC **IFORTH**

(--)

Read characters from the input stream until the text "ENDDOC" is encountered. You can place documentation text between the **DOC** command and the "ENDDOC" text. Note that all constructions between **DOC** and "ENDDOC" will be ignored, including the constructions that would normally make "ENDDOC" part of a regular comment such as (or \ . **DOC** cannot be nested.

DOES> "does" **C** **CORE**

Compilation: (colon-sys1 -- colon-sys2)

Append the execution semantics below to the current definition.

Execution: (--) (R: sys1 --)

Modify the execution semantics of the most recently defined word as shown below. Return control to the caller of the definition containing **DOES>** . The most recently defined word must have been defined with **CREATE** or a user-defined word that calls **CREATE** .

Execution: ("name" -- a-addr) (R: -- sys2)

name is the word modified by **DOES>** . Save implementation-dependent information about the definition that called *name*, and place *name*'s data field address on the stack. Execute the code following the **DOES>** that modified *name*.

See also: **CREATE**

DOS-ERROR@ **IFORTH**

(-- u)

Gets the number of the latest **OS** error that occurred. You can use *u* with **ERROR>TEXT** to convert it to a printable string. It may not always be clear when **DOS-ERROR@** is updated, because some iForth operations by-pass DOS or do multiple unrelated **OS** calls. Note: "DOS" doesn't mean "MS-DOS-only".

See also: **ERROR>TEXT**

DOUBLE **IFORTH**

DOUBLE is an environment query.

See also: **ENVIRONMENT?**

DOUBLE-EXT **IFORTH**

DOUBLE-EXT is an environment query.

See also: **ENVIRONMENT?**

DOUBLE-PRECISION **IFORTH**

DOUBLE-PRECISION is an environment query.

See also: **ENVIRONMENT?**

DOUBLE[] **IFORTH**

(addr1 index -- addr2)

Equivalent to **2* CELLS** + .

See also: **[]DOUBLE**

DPL **IFORTH**

(-- a-addr)

a-addr is the address of **DPL** . **DPL** contains the "decimal point" location if a double number is input, measured from the right. If a single number is input, **DPL** contains -1. **DPL** always contains 0 for double numbers if **ANSI** is **ON** . iForth allows several other characters to signal double numbers apart from '.' (full stop). In the case that more than one decimal point is present **DPL** points to the last one.

See also: **>NUMBER CONVERT NUMBER?**

DROP **CORE**

(x --)

Remove *x* from the stack.

DRSHIFT "d-r-shift" **IFORTH**

(d1 x -- d2)

Logically shift *d1* over *x* bits to the right, giving the result *d2*. A 0 bit is inserted at the left end of the value. Since this bit is also the sign bit for signed numbers, the sign bit is cleared by this command.

DU< "d-u-less" **DOUBLE EXT**

(ud1 ud2 -- flag)

flag is true if *ud1* is less than *ud2*.

DUMP **TOOLKIT EXT**

(addr u --)

Display the contents of *u* consecutive addresses starting at *addr*. The format of the display might be different between different ANS forth implementations. iForth uses the following scheme to print this listing:

```
$0001CE7A  04 44 55 4D  50 20 20 4C  .DUMP  L
$0001CE82  29 20 21 20  48 45 52 45  ) ! HERE
$0001CE8A  20 38 30 20  44 55 4D 50   80 DUMP
```

At the left the absolute memory address is shown. After the addres, memory is dumped in HEX, in groups of 4 bytes. At the extreme right, the contents are shown again, but now converted and printed with **SEMIT** . **DUMP** is sensitive to the contents of **(C/L)** and will use as many columns as will fit on the display.

DUP "dupe" **CORE**

(x -- x x)

Duplicate *x*.

DVALUE "d-value" **D** **IFORTH**

("name" --)

Parse *name* delimited by a space, ignoring leading delimiters. Create a dictionary entry for *name* with the execution semantics defined below. *name* is referred to as a "d-value".

Execution: ("name" -- d)

Place *d* on the stack. The value of *d* is unspecified until the phrase *d* **TO** *name* is executed, causing *d* to be associated with *name*.

E. "e-dot" **FLOAT**

(--) (F: r --)

Convert the top number on the floating-point stack to a character string using the rules of **(E.)** and display the resulting string with a trailing space. For example, the sequence

```
4 SET-PRECISION 123.00E0 E.
```

will display

```
1.2300E2
```

An ambiguous condition exists if the system base is not **DECIMAL** or if the character string exceeds the maximum size of the pictured numeric output string buffer.

E.R "e-dot-r" **IFORTH**

(n --) (F: r --)

Display the floating point number *r*, using the conversion rules as for **E.** , right aligned in a field *n* characters wide. If the number of characters required to display *r* is greater than *n*, all digits are displayed with no leading spaces in a field as wide as necessary.

See also: **(E.R)**

EAT-SPACES **IFORTH**

(--)

Removes all white space from the current input stream.

EDI-used! **IFORTH**

(bool --)

Instructs the optimizer how to handle the EDI register. In iForth 2.0 this register is used internally, so it must be saved and restored by user code in addition to signalling the optimizer. Set up in the ~/include/iforth.prf preferences file.

See also: **FPU-ovf! LEA-used! P6-used! EDI-used@**

EDI-used@ **IFORTH**

(-- bool)

Get the current state of EDI register handling.

See also: **EDI-used!**

EDITOR **TOOLKIT EXT**

(--)

Replace the first word list in the search order with the **EDITOR** word list.

EDX-used! **IFORTH**

(bool --)

Instructs the optimizer how to handle the EDX register. In iForth 2.0 this register is used internally, so it must be saved and restored by user code in addition to signalling the optimizer. Set up in the ~/include/iforth.prf preferences file.

See also: **FPU-ovf! LEA-used! P6-used! EDX-used@ EDI-used@**

EDX-used@ **IFORTH**

(-- bool)

Get the current state of EDX register handling.

See also: **EDX-used!**

EKEY "e-key" **FACILITY EXT**

(-- u)

Receive one keyboard event *u*. If the keyboard event corresponds to a character in the 8-bit ASCII character set, the numerical value of *u* is as specified by that character set. Otherwise, the value of *u* is numerically greater than the value of any character in the character set. When using iForth with the server on PC computers, **EKEY** returns the ASCII code for any normal key and otherwise returns the so called scan code of the key multiplied by 256.

See also: **KEY? KEY**

EKEY>CHAR "e-key-to-char" **FACILITY EXT**

(u1 -- u2 flag)

If the keyboard event *u1* was an extended standard key event, *flag* is true and *u2* is the value of the character received. Otherwise *flag* is false and *u2* is the value of the extended keyboard event.

EKEY? "e-key-question" **FACILITY EXT**

(-- flag)

If a keyboard event is available, returns true. Otherwise returns false. After **EKEY?** returns with a value of true, subsequent executions of **EKEY?** prior to the execution of **KEY** , **KEY?** or **EKEY** also return true, referring to the same event. The next execution of **EKEY** will return the same event without indefinite delay.

ELSE **C** **CORE**

Compilation: (orig1 -- orig2)

Put the location of a new unresolved forward reference *orig2* onto the control flow stack. Append the execution semantics given below to the current definition. The semantics will be incomplete until *orig2* is resolved (*e*.g. by **THEN**). Resolve the forward reference *orig1* using the location following the appended execution semantics.

Execution: (--)

Continue execution at the location given by the resolution of *orig2*.

See also: **IF THEN**

ELSE, "else-comma" **IFORTH-ASSEMBLER**

Assembling: (addr1 -- addr2)

Resolve the forward branch as generated by IF, , this forward branch is located at memory address *addr1*. Create an unconditional forward branch that is to be resolved by THEN, .

Execution: (--)

Jump to the corresponding THEN, statement.

See also: **IF, THEN,**

EMIT **CORE**

(x --)

If *x* is a displayable character in the implementation-defined character set, display *x*. The effect of **EMIT** for all other values of *x* is implementation-defined. Because of the potential non-transportable action by terminal devices of control characters, the use of control characters is an environmental dependency.

See also: **TYPE**

EMIT? "emit-question" **FACILITY EXT**

(-- flag)

flag is true if the output device is ready to display a character.

EMPTY-BUFFERS **BLOCK EXT**

(--)

Unassign all block buffers. Do not transfer the contents of any **UPDATE** -d block buffer to mass storage.

See also: **BLOCK**

EMULATED **IFORTH**

EMULATED is an environment query.

See also: **ENVIRONMENT?**

END-CODE **TOOLKIT EXT**

(sys --)

Complete the current word definition. Remove the **ASSEMBLER** word list from the search order. *sys* is balanced by the corresponding **CODE** or **;CODE** .

END-LOCALS **C** **IFORTH**

(--)

Mark the end of the section in which all local variables for the current definition are defined.

END-STREAM? "end-stream-question" **IFORTH**

(-- flag)

Test if the input buffer is exhausted and needs REFILLing. If so, *flag* is true, otherwise *flag* is false.

ENDCASE "end-case" **C** **CORE EXT**

Compilation: (case-sys1 ... case-sys2 of-sys --)

Mark the end of the **CASE** ... **OF** ... **ENDOF** ... **ENDCASE** structure. Resolve all of the orign which were processed after the *dest* left by **CASE** . Append the execution semantics given below to the current definition.

Execution: (x --)

Discard the case selector *x* and continue execution.

See also: **CASE ENDOF OF**

ENDDOC "end-doc" **IFORTH**

(--)

The delimiter word that is searched for by **DOC** . It signals the end of a text block used for documentation purposes. **ENDDOC** itself is just a NO-OP . Note that **DOC** ... **ENDDOC** and doc enddoc will both work when **CASESENSITIVE** is off.

See also: **DOC**

ENDIF "end-if" **C** **IFORTH**

ENDIF is an alias for THEN .

See also: **THEN**

ENDIF, **IFORTH-ASSEMBLER**

Assembling: (addr --)

Execution: (--)

ENDIF, is an alias for THEN,

See also: **THEN,**

ENDOF "end-of" **C** **CORE EXT**

Compilation: (case-sys1 of-sys -- case-sys2)

mark the end of the ... **OF** ... **ENDOF** ... part of the **CASE** structure. The next location for a transfer of control resolves the reference given by the top element of the control flow stack. The location of the new unresolved forward reference is placed beneath the *dest* left by **CASE** . Append the execution semantics given below to the current definition.

Execution: (--)

Continue execution at the location specified by the consumer of *orig2*.

See also: **CASE ENDCASE OF**

ENVIR **IFORTH**

(--)

ENVIR is a wordlist as defined by **WORDLIST** . This wordlist contains all strings recognized by **ENVIRONMENT?** as Forth words. In fact **ENVIRONMENT?** searches this wordlist to find a string. When found, **ENVIRONMENT?** executes the word to get the associated value. Words added to this wordlist are automatically recognized by **ENVIRONMENT?** .

ENVIRONMENT? "environment-query" **CORE**

(c-addr u -- value true) when known or (c-addr u -- false) when unknown

c-addr is the address of a character string and *u* is the string's character count. The character string should be an element of the table shown below. The table contains keyword/value and keyword/flag pairs. The value or flag associated with the string is placed on the stack. You can use the program in the file include/whatenv.frt to list the current values of the environment queries. The following strings are recognized by **ENVIRONMENT?** :

```
String      Value type  Interpretation
------      ----------  --------------
ALIGN       n           alignment granularity, in address
                        units, of an aligned address.
/COUNTED-STRING
            n           maximum number of characters in a
                        counted string
/HOLD       n           maximum size of a pictured numeric
```

```
                        output string in characters
/PAD          n         size of the scratch area pointed to by
                        PAD, in characters
ADDRESS-UNIT-BITS
              n         Size of one address unit in bits
CORE          flag      core word set present
CORE-EXT      flag      core extension word set present
MAX-CHAR      u         the maximum value of any character in
                        the implementation-defined character set
MAX-D         u         largest usable signed double number
MAX-N         n         largest usable signed integer
MAX-U         u         largest usable unsigned integer
MAX-UD        ud        largest usable unsigned double number
OPERATING-SYSTEM
              u         identifies the current OS: MS-DOS, Linux,
                        Windows 95, Windows NT/2K or MMURTL.
RETURN-STACK-CELLS
              n         maximum size of the return stack in
                        cells
STACK-CELLS   n         maximum size of the data stack in cells
BLOCK         flag      block word set present
BLOCK-EXT     flag      block extension word set present
DOUBLE        flag      double number word set present
DOUBLE-EXT    flag      double number extension word set present
ERROR-HANDLING
              flag      error handling word set present
ERROR-HANDLING-EXT
              flag      error handling extension word set present
FILE          flag      file word set present
FILE-EXT      flag      file extension word set present
FLOATING      flag      floating-point word set present
FLOATING-EXT
              flag      floating-point extension word set present
FLOATING-STACK
              n         n is the maximum depth of the separate
                        floating-point stack.
MAX-FLOAT     r         largest usable floating-point number
#LOCALS       n         maximum number of local variables in a
                        definition
LOCALS        flag      locals word set present
LOCALS-EXT    flag      locals extension word set present
MEMORY-ALLOC
              flag      memory-allocation word set present
MEMORY-ALLOC-EXT
              flag      memory-allocation extension word set
                        present
SEARCH-ORDER
              flag      search order word set present
SEARCH-ORDER-EXT
              flag      search order extension word set present
WORDLISTS     n         maximum number of word lists usable in
                        the search order
TOOLS         flag      programming tools word set present
TOOLS-EXT     flag      programming tools extension word set
                        present
STRING        flag      string word set present
STRING-EXT    flag      string extension word set present
/DATA-SPACE   n         the maximum number of cells that an
                        application program may attempt to ALLOT
DOUBLE-PRECISION
              flag      returns the precision of the
                        floating-point package
EMULATED      flag      true if floating-point is emulated,
                        false if hardware is used
FACILITY      flag      facility word set present
FACILITY-EXT
              flag      facility extension word set present
SERVER        char      returns a character that identifies the
                                server being used
SYSTEM-STACK-CELLS
              n         maximum size of the system stack in cells
```

```
IFORTH      flag        true if this is a iForth system
VER         n           the version number of iForth
```

ERASE **CORE EXT**

(addr u --)

If *u* is greater than zero, clear all bits in each of *u* consecutive address units of memory beginning at *addr*.

ERROR-HANDLING **IFORTH**

ERROR-HANDLING is an environment query.

See also: **ENVIRONMENT?**

ERROR-HANDLING-EXT **IFORTH**

ERROR-HANDLING-EXT is an environment query.

See also: **ENVIRONMENT?**

ERROR>TEXT "error-to-text" **IFORTH**

(c-addr1 u1 errnum -- c-addr2 u2 ior)

ERROR>TEXT asks the server to return a string describing the **I/O** result errnum. The string is placed in the buffer at *c-addr1* and contains at most *u1* characters.

EVAL" "eval-quote" **C** **IFORTH**

Compilation: ("ccc<">" --)

Parse characters *ccc* delimited by " (double quote). Append the execution semantics specified below to the current definition.

*Execution: (i*x -- j*x)*

Save the current input stream specification. Make *ccc* the current input string and interpret its contents. When the input stream is exhausted, restore to the prior input specification.

See also: **EVALUATE**

EVALUATE **CORE**

*(i*x c-addr u -- j*x)*

Save the current input stream specification. Make the string described by *c-addr* and *u* the current input stream and interpret its contents. When the input stream is exhausted, restore the prior input stream specification. When **QUIT** gets executed while in **EVALUATE** , **TIB** will be set to *c-addr*. As this could be right in the middle of a colon definition, all sorts of puzzling errors may result.

EXECUTE **CORE**

*(i*x xt -- j*x)*

Execute the definition specified by *xt*. In iForth the definition is not executed if *xt* is 0.

See also: ' **[']**

EXIT **C** **CORE**

(--) (R: sys --)

Return control to the caller of the definition containing **EXIT** .

EXPECT **CORE EXT**

(c-addr +n --)

Receive a string of at most *+n* characters. Display graphic characters as they are received. The editing functions that the system performs in order to construct the string of characters are implementation-defined. This system adds any character to the end of the string, backspace removes the last character of the string. Input terminates when 'return' is received or when the string is *+n* characters long. When 'return' is received nothing is appended to the string and the display is maintained. Store the string at *c-addr* and its length in **SPAN** . Note: This word is obsolescent and is included as a concession to existing implementations. Its function is superseded by **ACCEPT** .

See also: **ACCEPT**

EXTENDED-PRECISION **IFORTH**

EXTENDED-PRECISION is an environment query.

See also: **ENVIRONMENT?**

EXTRACT **IFORTH**

(ud1 base -- ud2 c)

c is the last character of the representation of *ud1* under the numeric base *base*. *ud2* is *ud1* divided by *base*. Note that when **EXTRACT** is repeatedly executed, each character of the representation of *ud1* will be calculated.

F! "f-store" **FLOAT**

(a-addr --) (F: r --)

Store *r* at *a-addr*.

F!+ "f-store-plus" **IFORTH**

(a-addr1 -- a-addr2) (F: r --)

Store the floating-point number *r* at *a-addr1*, and leave the incremented pointer as *a-addr2*.

See also: **DF! SF!+ XF!+**

F* "f-star" **FLOAT**

(--) (F: r1 r2 -- r3)

Multiply *r1* by *r2* giving *r3*.

F** "f-star-star" **FLOAT EXT**

(--) (F: r1 r2 -- r3)

Raise *r1* to the power *r2*, giving the product *r3*.

F+ "f-plus" **FLOAT**

(--) (F: r1 r2 -- r3)

Add *r1* to *r2* giving the sum *r3*.

F+! "f-plus-store" **IFORTH**

(a-addr --) (F: r --)

Add *r* to the floating-point number at *a-addr*.

F+!+ "f-plus-store-plus" **IFORTH**

(F: r --) (a-addr1 -- a-addr2)

Add the float *r* to the float at *a-addr1* and leave the incremented pointer as *a-addr2*.

See also: **DF+!+ SF+!+ XF+!+**

F, "f-comma" **IFORTH**

(--) (F: r --)

Reserve the size of a floating-point number in the data-space and store *r* in that space. An ambiguous condition exists if the address of the next available data space location is not aligned.

F- "f-minus" **FLOAT**

(--) (F: r1 r2 -- r3)

Subtract *r2* from *r1*, giving *r3*.

F-! "f-minus-store" **IFORTH**

(F: r --) (a-addr --)

Subtracts the float *r* from the float at *a-addr*.

F. "f-dot" **FLOAT**

(--) (F: r --)

Convert the top number on the floating-point stack to a character string using the rules of **(F.)** . and display the resulting string with a trailing space. For example, the sequence

```
4 SET-PRECISION 1.23E5 F.
```

will display

```
123000.0000
```

An ambiguous condition exists if the system basis is not **DECIMAL** or if the character representation exceeds the size of the pictured numeric output string buffer. When the fieldwidth is too small, **F.** and its variants try to squeeze the number in by switching to exponential notation. When that doesn't help, they print a field of asterisks. The switch to exponential notation happens when |*r*| **>** 10^PRECISION . When |*r*| **<** 10^-PRECISION , "0.00.." is printed.

See also: **(E.) F. E. (F.R) F.R (E.R) E.R (FE.R) FE.R**

F.R "f-dot-r" **IFORTH**

(n --) (F: r --)

Display the floating point number *r* using the conversion rules as for F., right aligned in a field *n* characters wide. If the number of characters required to display *r* is greater than *n*, all digits are displayed with no leading spaces in a field as wide as necessary. When the fieldwidth is too small, **F.R** and its variants try to squeeze the number in by switching to exponential notation. When that doesn't help, they print a field of asterisks. The switch to exponential notation happens when |*r*| **>** 10^PRECISION . When |*r*| **<** 10^-PRECISION , "0.00.." is printed.

See also: **(F.) (E.) F. E. (F.R) F.R (E.R) E.R (FE.R) FE.R**

F/ "f-slash" **FLOAT**

(--) (F: r1 r2 -- r3)

Divide *r1* by *r2*, giving the quotient *r3*. An ambiguous condition exists if *r2* is zero, or the quotient lies outside of the range of a floating-point number.

F0! "f-zero-store" **IFORTH**

(-- a-addr)

Store the floating-point value 0E into the memory location starting at *a-addr*.

F0< "f-zero-less-than" **FLOAT**

(-- flag) (F: r --)

flag is true if *r* is less than zero.

F0<= "f-zero-smaller-or-equal" **IFORTH**

(-- f) (F: r --)

Tests *r* for being negative or zero. Equivalent to 0e **F> INVERT** .

F0<> "f-zero-not-equal" **IFORTH**

(-- flag) (F: r --)

flag is true if *r* is not equal to 0.

F0= "f-zero-equals" **FLOAT**

(-- flag) (F: r --)

flag is true if *r* is equal to zero.

F0> "f-zero-greater" **IFORTH**

(-- flag) (F: r --)

flag is true if *r* is greater than zero.

F0>= "f-zero-greater-or-equal" **IFORTH**

(-- f) (F: r --)

Tests *r* for being positive or zero. Equivalent to 0e **F< INVERT** .

F1+ "f-one-plus" **IFORTH**

(--) (F: r1 -- r2)

Add 1E0 to the floating-point number *r1*, giving *r2*.

F1- ""f-one-minus"“ **IFORTH**

(--) (F: r1 -- r2)

Subtract 1E0 form the floating-point number *r1*, giving *r2*.

F2* "f-two-star" **IFORTH**

*(F: r -- r*2)*

Multiply the number *r* by 2.

F2/ "f-two-slash" **IFORTH**

(F: r -- r/2)

Divides the number *r* by 2.

F2DROP "f-two-drop" **IFORTH**

(--) (F: r1 r2 --)

Remove *r1* and *r2* from the floating-point stack.

F2DUP "f-two-dupe" **IFORTH**

(--) (F: r1 r2 -- r1 r2 r1 r2)

Duplicate the floating-point number pair *r1 r2*.

F2OVER "f-two-over" **IFORTH**

(--) (F: r1 r2 r3 r4 -- r1 r2 r3 r4 r1 r2)

Duplicate the floating-point number pair *r1 r2*.

F2ROT "f-two-rot" **IFORTH**

(--) (F: r1 r2 r3 r4 r5 r6 -- r3 r4 r5 r6 r1 r2)

Move the floating-point number pair *r1 r2* to FTOS.

F2SWAP "f-two-swap" **IFORTH**

(--) (F: r1 r2 r3 r4 -- r3 r4 r1 r2)

Swap the floating-point number pairs *r1 r2* and *r3 r4*.

F2^X **IFORTH**

(F: r1 -- r2)

Compute *r2* = 2^*r1*.

See also: **F****

F< "f-less-than" **FLOAT**

(-- flag) (F: r1 r2 --)

flag is true if *r1* is less than *r2*.

F<= "f-less-than-or-equal" **IFORTH**

(-- f) (F: r1 r2 --)

Tests *r1* for being less than or equal to *r2*. Equivalent to **F> INVERT** .

F<> "f-equal" **IFORTH**

(-- flag) (F: r1 r2 --)

If the bit-patterns of *r1* and *r2* are not equal, *flag* is true.

See also: **F= F~**

F= "f-equal" **IFORTH**

(-- flag) (F: r1 r2 --)

If the bit-patterns of *r1* and *r2* are equal, *flag* is true.

See also: **F<> F~**

F> "f-greater" **IFORTH**

(-- flag) (F: r1 r2 --)

flag is true if *r1* is greater than *r2*.

F>= "f-greater-than-or-equal" **IFORTH**

(-- f) (F: r1 r2 --)

Tests *r1* for being greater than or equal to *r2*. Equivalent to **F< INVERT** .

F>D "f-to-d" **FLOAT**

(-- d) (F: r --)

d is the double-cell signed integer equivalent to the integer portion of *r*. The fractional portion of *r* is discarded. An ambiguous condition exists if the integer portion of *r* cannot be precisely represented as a double-cell signed integer.

F>S "f-to-s" **IFORTH**

(-- x) (F: -- r)

x is the single-cell signed integer equivalent to the integer portion of *r*. The fractional part of *r* is discarded. An ambiguous condition exists if the integer portion of *r* cannot be precisely represented as a single-cell signed integer.

F@ "f-fetch" **FLOAT**

(a-addr --) (F: -- r)

r is the value stored at *a-addr*.

F@+ "f-fetch-plus" **IFORTH**

(a-addr1 -- a-addr2) (F: -- r)

Fetch the floating point number *r* from *a-addr1*. Add 1 **FLOATS** to *a-addr1* giving *a-addr2*.

FABS "f-abs" **FLOAT EXT**

(--) (F: r1 -- r2)

r2 is the absolute value of *r1*.

FACILITY **IFORTH**

FACILITY is an environment query.

See also: **ENVIRONMENT?**

FACILITY-EXT **IFORTH**

FACILITY-EXT is an environment query.

See also: **ENVIRONMENT?**

FACOS "f-a-cos" **FLOAT EXT**

(--) (F: r1 -- r2)

r2 is the principle radian angle whose cosine is *r1*. An ambiguous condition exists if |*r1*| is greater than 1.

FACOSH "f-a-cos-h" **FLOAT EXT**

(--) (F: r1 -- r2)

r2 is the inverse hyperbolic cosine of *r1*.

FALIGN **FLOAT**

(--)

If the address of the next available data space location is not aligned for a floating point number, reserve enough space to align it.

FALIGNED **FLOAT**

(addr -- a-addr)

a-addr is the first floating point aligned address greater than or equal to *addr*.

FALOG "f-a-log" **FLOAT EXT**

(--) (F: r1 -- r2)

Raise 10 to the power *r1*, giving *r2*.

FALSE **CORE EXT**

(-- false)

Return a false flag.

FASIN "f-a-sine" **FLOAT EXT**

(--) (F: r1 -- r2)

r2 is the principal radian angle whose sine is *r1*. An ambiguous condition exists if |*r1*| is greater than 1.

FASINH "f-a-sine-h" **FLOAT EXT**

(--) (F: r1 -- r2)

r2 is the inverse hyperbolic sine of *r1*.

FAST; **IFORTH**

(--)

Undocumented experimental colon definition type.

See also: **:FAST I/O FI/FO [XCOMPILE]**

FATAN "f-a-tan" **FLOAT EXT**

(--) (F: r1 -- r2)

r2 is the principal radian angle whose tangent is *r1*.

FATAN2 "f-a-tan-two" **FLOAT EXT**

(--) (F: r1 r2 -- r3)

r3 is the radian angle whose tangent is *r1*/*r2*. An ambiguous condition exists if *r1* and *r2* are zero.

FATANH "f-a-tan-h" **FLOAT EXT**

(--) (F: r1 -- r2)

r2 is the inverse hyperbolic tangent of *r1*.

FCALLBACK **IFORTH**

(addr #args "name" --)

This is a CREATEing word that generates a word called *name*. (The stack diagram applies to **CREATE** time.) The only difference with **CALLBACK** is that the Forth word should not only leave an integer result on the normal parameter stack, but also push a float result on the Forth float stack. Assuming the external code is written in "**C**", the Forth word at address expects the **C** arguments in the order written in the **C** documentation (*i.e.* right-most on TOS). Here FP input parameters are not passed on the FPU stack (like the final result) but as 64-bit items (counting for two items in *#args*) on the normal parameter stack. To streamline the traffic of these 64-bit items to and from the Forth stacks use **FPUSHS** etc.

See also: **CALLBACK FOREIGN FPUSHS FPUSHD FPOPS FPOPD**

FCONSTANT "f-constant" **D** **FLOAT**

("name" --) (F: r --)

Parse *name* delimited by a space, ignoring leading delimiters. Create a dictionary entry for *name* with the execution semantics defined below. *name* is referred to as an "f-constant."

Execution: ("name" --) (F: -- r)

Place *r* on the floating-point stack.

FCOS "f-cos" **FLOAT EXT**

(--) (F: r1 -- r2)

r2 is the cosine of the radian angle *r1*.

FCOSH "f-cos-h" **FLOAT EXT**

(--) (F: r1 -- r2)

r2 is the hyperbolic cosine of *r1*.

FDEC "f-dec" **IFORTH**

(-- a-addr)

a-addr is the address of **FDEC** . **FDEC** contains the character that is to be used as the decimal dot when formatting floating-point numbers with **(E.)** , **(F.)** and all words that use these words such as **E.** and **F.** .

FDEG "f-deg" **IFORTH**

(--) (F: r1 -- r2)

r2 is the equivalent angle in degrees of the angle *r1* in radians.

FDEPTH "f-depth" **FLOAT**

(-- +n)

+n is the number of values contained on the default separate floating-point stack. Some implementations of the ANS standard keep the floating-point numbers on the data stack, in that case *+n* is the current number of possible floating-point values contained on the data stack.

FDROP "f-drop" **FLOAT**

(--) (F: r --)

Remove *r* from the floating-point stack.

FDUP "f-dupe" **FLOAT**

(--) (F: r -- r r)

Duplicate *r*.

FE. "f-e-dot" **FLOAT**

(--) (F: r --)

Convert the top number on the floating-point stack to a character string using the standard engineering notation for floating point numbers and display the resulting string with a trailing space. For example, the sequence

```
4 SET-PRECISION 1.23E5 FE.
```

will display

```
123.0000E3
```

An ambiguous condition exists if the system basis is not **DECIMAL** (not a problem for iForth) or if the character representation exceeds the size of the pictured numeric output string buffer.

FE.R "f-e-dot-r" **IFORTH**

(n --) (F: r --)

Display the floating point number *r* using the conversion rules as for **FE.** right aligned in a field *n* characters wide. If the number of characters required to display *r* is greater than *n*, all digits are displayed with no leading spaces in a field as wide as necessary.

FECHAR "f-e-char" **IFORTH**

(-- a-addr)

a-addr is the address of **FECHAR** . **FECHAR** contains the character that is to be used as the exponent character when formatting floating-point numbers with **(E.)** and all words that use this word such as **E.** .

FELEN "f-e-len" **IFORTH**

(-- a-addr)

a-addr is the address of **FELEN** . **FELEN** contains the number of decimals of the exponent value that is to be used when formatting floating-point numbers with **(E.)** and all words that use these words such as **E.** .

FENCE **IFORTH**

(-- a-addr)

a-addr is the address of **FENCE** . **FENCE** contains a pointer into the code space. Attempts to remove any code before this address will give an error. Where the execution tokens of iForth are pointers into code space, this can be used as follows. By storing the execution token of a word into **FENCE** , all words defined prior to that word are protected. Initially all pre-compiled words are protected.

FESIGN "f-e-sign" **IFORTH**

(-- a-addr)

a-addr is the address of **FESIGN** . **FESIGN** contains the number of the method that is to be used when generating the sign of the exponent of floating-point numbers with **(E.)** and all

words that use these words such as **E.** . When **FESIGN** is 0, the sign character is either '-' or absent, otherwise the sign character is '-' or '+'.

FEXCHANGE "f-exchange" **IFORTH**

(u --) (F: ?? -- ??)

Exchanges top and *u*-th item (zero-based) of the FP stack. This operation takes zero cycles on Intel hardware. Example:

```
FORTH> 1e 2e 3e .S 2 FEXCHANGE .S
  Data: ---
System: ---
 Float: 1.000000 2.000000 3.000000 ---
  Data: ---
System: ---
 Float: 3.000000 2.000000 1.000000 --- ok
```

See also: **FROT -FROT FPICK**

FEXP "f-e-x-p" **FLOAT EXT**

(--) (F: r1 -- r2)

Raise *e* to the power *r1*, giving *r2*.

FEXPM1 "f-e-x-p-m-one" **FLOAT EXT**

(--) (F: r1 -- r2)

Raise *e* to the power *r1* and subtract 1, giving *r2*. This word has full machine precision even when *r1* is close to zero (so this result could not be obtained using **FEXP**).

FFOREIGN **IFORTH**

(n{u} n 'service --) (F: -- result)*

Calls a dynamically linked "**C**" library routine at address *'service*, with *n* unsigned arguments *u*. (Floating point parameters are passed as 32 or 64-bit items, to be converted with **FPUSHS** FPUSHD).

See also: **FOREIGN CALLBACK FPUSHS FPUSHD**

FFSP "f-f-s-p" **IFORTH-ASSEMBLER**

(-- offset)

offset is the *offset* in the workspace at which the floating-point stack pointer is stored.

FI/FO **IFORTH**

(--)

Undocumented experimental word.

See also: **:FAST FAST; I/O FI/FO [XCOMPILE]**

FID>NAME "file-i-d-to-name" **IFORTH**

(c-addr1 u1 fid -- c-addr2 u2 ior)

FID>NAME asks the server to return a string describing the file identifier *fid*. The string is placed in the buffer at *c-addr1* and contains at most *u1* characters.

FILE **IFORTH**

FILE is an environment query.

See also: **ENVIRONMENT?**

FILE-EXT **IFORTH**

FILE-EXT is an environment query.

See also: **ENVIRONMENT?**

FILE-POSITION **FILE**

(fileid -- ud ior)

ud is the current file position for the file identified by *fileid*. *ior* is the implementation-defined **I/O** result code. If the file is (also) opened for writing and the file position is moved after the end of the file, the next write command will extend the file to at least the new position.

See also: **ERROR>TEXT**

FILE-SIZE **FILE**

(fileid -- ud ior)

ud is the size, in characters, of the file identified by *fileid*. *ior* is the implementation-defined **I/O** result code. This operation does not affect the value returned by **FILE-POSITION** .

See also: **ERROR>TEXT**

FILE-STATUS **FILE EXT**

(c-addr u -- x ior)

Return the status of the file identified by the character string *c-addr u*. If the file exists, *ior* is zero; otherwise *ior* is the implementation-defined **I/O** result code. *x* contains implementation-defined information about the file. The result *x* returned by **FILE-STATUS** is undefined in iForth. This means that this word can only be used to verify that a certain file exists, nothing more.

FILL **CORE**

(c-addr u char --)

If *u* is greater than zero, store *char* in each of *u* consecutive characters of memory beginning at *c-addr*.

FIN/FOUT **IFORTH**

(#fin #fout --)

Used in inline macro's to optimize parameter access. *#fin* is the number of float items expected on the FPU stack on entry. Presently *#fin* can only be 0 .. 2. *#fout* is the number of parameters on the FPU stack at exit of the macro. Presently *#out* can only be 0 .. 2. The word **ADJUST-STACK** must be used together with **FIN/FOUT** to flush FPU items to the float memory stack after a macro (or set of macro's) has finished. **FIN/FOUT** and **ADJUST-STACK** work together in order to remove superfluous data stack traffic at macro boundaries. Example:

```
ALSO ASSEMBLER
: myfadd   2 1 FIN/FOUT
           POSTPONE ASM{ ST(1) -> ST faddp, }ASM
           ADJUST-STACK ; IMMEDIATE COMPILE-ONLY
PREVIOUS
: 2add     33e 22e myfadd F. ;
```

The generated code for 2add is:

```
: 2add
fld         $004CE0C8 qword-ptr     ( 33e )
fld         $004CE0C0 qword-ptr     ( 22e )
faddp       ST(1), ST               ( your macro )
mov         eax, [edi #16 +] dword  ( adjust-stack )
lea         eax, [eax #-16 +] dword
mov         [edi #16 +] dword, eax
fxch        ST(2)
fstp        [eax 8 +] qword
fstp        [eax 0 +] qword
call        F. ( $0045FE90 ) offset NEAR
;
```

See also: **ADJUST-STACK FIN/FOUT IN/OUT**

FIND **CORE**

(c-addr -- c-addr 0) or (c-addr -- xt 1) or (c-addr -- xt -1)

Find the Forth word named in the counted string at *c-addr*. If the word is not found, return *c-addr* and 0. If the word is found, return *xt*. If the word is immediate, also return 1, otherwise also return -1.

FINITIALIZE "floating-point-initialize" **IFORTH**

(--) (F: --)

Initialize the floating point hardware. In some environments this word is needed after having used **SYSTEM** , or when booting.

FLITERAL "f-literal" **C** **FLOAT**

Compilation: (F: r --)

Compile floating-point number *r* as a literal.

Execution: (F: -- r)

Place *r* on the floating-point stack.

FLN "f-l-n" **FLOAT EXT**

(--) (F: r1 -- r2)

r2 is the natural logarithm of *r1*. An ambiguous condition exists if *r1* is less than or equal to zero.

FLNP1 "f-l-n-p-one" FORTH

(--) (F: r1 -- r2)

r2 is the natural logarithm of the quantity *r1* plus one. An ambiguous condition exists if *r1* is less than or equal to minus one.

FLOAT+ "float-plus" **FLOAT**

(a-addr1 -- a-addr2)

Add the size of a floating-point number, specified in address units, to *a-addr1*, giving *a-addr2*.

FLOAT- "float-minus" **IFORTH**

(a-addr1 -- a-addr2)

Subtract the size of a floating-point number, specified in address units, to *a-addr1*, giving *a-addr2*.

FLOATING **IFORTH**

FLOATING is an environment query.

See also: **ENVIRONMENT?**

FLOATING-EXT **IFORTH**

FLOATING-EXT is an environment query.

See also: **ENVIRONMENT?**

FLOATING-STACK **IFORTH**

FLOATING-STACK is an environment query.

See also: **ENVIRONMENT?**

FLOATS **FLOAT**

(n1 -- n2)

n2 is the size, in address units, of *n1* floating-point numbers.

FLOAT[] **IFORTH**

(addr1 index -- addr2)

Equivalent to **FLOATS** + .

See also: **[]FLOAT**

FLOCAL "f-local" **C** **IFORTH**

Compilation: ("name" --)

Parse *name* delimited by a space, ignoring leading delimiters. Create a temporary dictionary entry for *name* with the execution semantics defined below. **FLOCAL** can only be used inside a definition. *name* remains in the dictionary until the current definition is finished with ; **DOES>** or **;CODE** .

Execution: (F: x --)

Initialize *name* with the floating-point value *x*.

Execution: (--) (F: -- x)

Place *x* on the floating-point stack.

FLOCALS| **C** **IFORTH**

*Compilation: ("name"*n "|" --)*

Parse names delimited by a blank, ignoring leading delimiters. The list of names is terminated by a '|' (bar). Create a temporary dictionary entry for each *name* with the execution semantics defined below. **FLOCALS|** can only be used inside a definition. The names remain in the dictionary until the current definition is finished with ; **;CODE** or **DOES>** . iForth accepts an unlimited number of names. The ANS standard requires a minimum of eight.

Execution: (F: -- r)

Place *r* on the stack. The value of *r* is unspecified until the phrase *r* **TO** *name* is executed, causing *r* to be associated with *name*.

See also: **LOCALS|**

FLOG "f-log" **FLOAT EXT**

(--) (F: r1 -- r2)

r2 is the base 10 logarithm of *r1*. An ambiguous condition exists if *r1* is less than or equal to zero.

FLOG2(X) "f-log-two-x" **IFORTH**

(--) (F: r1 -- r2)

r2 is the base 2 logarithm of *r1*. An ambiguous condition exists if *r1* is less than or equal to zero.

FLOOR **FLOAT**

(--) (F: r1 -- r2)

Round *r1* using the "round toward negative infinity" rule, giving *r2*.

FLOORED **IFORTH**

Environment query.

FLSP "f-l-s-p" **IFORTH-ASSEMBLER**

(-- offset)

offset is the *offset* in the workspace at which the locals stack pointer is stored.

FLUSH **BLOCK**

(--)

Perform the function of **`SAVE-BUFFERS`** , then unassign all block buffers.

FLUSH-FILE **FILE EXT**

(fileid -- ior)

Attempt to force any buffered information written to the file referred to by *fileid* to mass storage, and the size information for the file to be recorded in the storage directory if changed. If the operation is successful, *ior* is zero. Otherwise, it is an implementation-defined `I/O` result code.

FM/MOD "f-m-slash-mod" **CORE**

(d1 n1 -- n2 n3)

Divide *d1* by *n1*, giving the floored quotient *n3* and the remainder *n2*. Input and output stack arguments are signed. An ambiguous condition exists if *n1* is zero or if the quotient lies outside the range of a single-cell signed integer.

FMAX "f-max" **FLOAT**

(--) (F: r1 r2 -- r3)

r3 is the greater of *r1* and *r2*.

FMIN "f-min" **FLOAT**

(--) (F: r1 r2 -- r3)

r3 is the lesser of *r1* and *r2*.

FMSIGN "f-m-sign" **IFORTH**

(-- a-addr)

a-addr is the address of **FMSIGN** . **FMSIGN** contains the number of the method that is to be used when generating the sign of the mantissa of floating point numbers with **(E.)** , **(F.)** and all words that use these words such as **E.** and **F.** . When **FMSIGN** is 0, the sign character is either '-' or absent, when **FMSIGN** is 1, the sign character is either '-' or '+', when **FMSIGN** is **2,** the sign character is either '-' or space, otherwise the sign character is undefined.

FNEGATE "f-negate" **FLOAT**

(--) (F: r1 -- r2)

r2 is the negation of *r1*.

FNIP "f-nip" **IFORTH**

(--) (F: r1 r2 -- r2)

Drop the first item below the top of the floating-point stack.

FOR **IFORTH**

Compilation: (-- fordest)

The next location for a transfer of control (*fordest*) goes onto the control flow stack. Append the execution semantics given below to the current definition.

Execution: (n|u --) (R: -- sys)

Set up the loop control parameter with limit *n|u*. Anything already on the return stack becomes unavailable until the loop control parameters are discarded. Each pass through the loop, the loop control parameter is tested for zero. If it becomes zero the loop is exited, if it does not, the control parameter is decremented by one. If the limit *n|u* is 0 to start with, one pass will be made. As this behavior is not very useful, the **FOR AFT** ... **THEN NEXT** construct has gained popularity. It is allowed to use **LEAVE** to exit from a **FOR** ... **NEXT** loop.

See also: **I@ NEXT UNNEXT**

FOREIGN **IFORTH**

(n{u} n 'service -- result)*

Calls a dynamically linked "**C**" library routine at address *'service*, with *n* unsigned arguments *u*. Prevents the **OS** from disrupting correct Forth operation by saving and restoring all Forth registers. The **C** (machine) and iForth (data) stacks are swapped for the duration of the call. This is essential as some **C** routines use enormous amounts of stack space. **C** sees the arguments in the same order as Forth (top Forth == last *u* in **C**'s argument list). The pre-compiled Forth kernel assumes **C** returns function results in the EAX register. This is true for the Borland and gcc compilers.

See also: **SYSCALL**

FORGET **TOOLKIT EXT**

("name" --)

Parse *name* delimited by a space, ignoring leading delimiters. Beginning with the last word defined, start deleting definitions from the dictionary until *name* itself is found and deleted. An ambiguous condition exists if *name* cannot be found. If a definition to be deleted has a non-empty forget field, the execution token found in this field is executed with the word's data field address as its parameter, before deletion takes place. This special behavior allows memory deallocation and process descheduling to take place automatically. If word lists are present, **FORGET** searches the compilation word list. When the compilation wordlist is removed, the **FORTH** wordlist is installed as the compilation wordlist. Note that other

implementations of the standard might fail if the compilation wordlist is removed. Note also that other implementations of the ANS standard might not have forget-fields or do not take the trouble of removing every word in the list separately.

See also: **'FORGET' FORGET>**

FORGET> "forget-part" **C** **IFORTH**

Compilation: (colon-sys1 -- colon-sys2)

Append the execution semantics below to the current definition.

Execution: (--) (R: sys1 --)

Modify the forgetting semantics of the most recently defined word as shown below. Return control to the caller of the definition containing **FORGET>** . The most recently defined word must have been defined with **CREATE** . When forgetting names, **FORGET** will encounter the non-empty forget field of each *name*. *name* is the word modified by **FORGET>** . The token found is executed as follows:

(exec-token -- a-addr) (R: -- sys2)

Save implementation-dependent information about the definition that called *name*, and place *name*'s data field address on the stack. Execute the code following the **FORGET>** that modified *name*. The minimal action that should be performed by user-written forget parts is to drop the data field address. **FORGET>** is optional.

See also: **CREATE DOES> FORGET IS-FORGET**

FORTH **SEARCH EXT**

(--)

Transform the search order consisting of *wid1*, ... *widn*-1, *widn* (where *widn* is searched first) into *wid1*, ... *widn*-1, FORTH-WORD-LIST .

FORTH-IO "forth-i-o" **IFORTH**

(--)

Sets all **I/O** vectors to their default routines. The default routines are the fastest possible on the given platform. For the MS-DOS IFORTH the disadvantage of **FORTH-IO** is that it does not obey OS-level redirection. The Linux, WIN95 and WINNT implementations of **FORTH-IO** are exactly equal to **OS-IO** and support redirection. The switched vectors are: **'KEY? 'KEY 'EMIT 'EMIT? 'TYPE 'AT-XY '?AT** and **'PAGE** .

See also: **IO-SYSTEM OS-IO**

FORTH-PROCESS **IFORTH**

Compile: (<name> --) (exec: xt --)

Allocate resources for a Forth process (task). To run any Forth word as a parallel task, do something like:

```
FORTH-PROCESS test

   : my-routine  #100 0 DO pause #10 ms LOOP
                   CR ."    Hello, world!"STOP ;

   ' my-routine RUN test
```

Do not forget the **STOP** before the final ";" ... A process should not have its own interpreter, or compile code (although it might work). As the multi-tasker is not pre-emptive, **PAUSE** should be used as much as possible.

FORTH-WORDLIST **SEARCH**

(-- wid)

Return *wid*, the identifier of the word list that includes all standard words provided by the implementation. This word list is initially the compilation word list and is part of the initial search order.

FOVER "f-over" **FLOAT**

(--) (F: r1 r2 -- r1 r2 r1)

Place a copy of *r1* on top of the floating-point stack.

FPICK "f-pick" **IFORTH**

(u --) (F: r(u) ... r(1) r(0) -- r(u) ... r(1) r(0) r(u))

Remove *u*. Copy the *u*th item of the floating-point stack to the top of the floating-point stack. An ambiguous condition exists is there are less than *u*+2 items on the floating-point stack before **FPICK** is executed.

FPOPD "f-pop-d" **IFORTH**

(ud --) (F: -- r)

Move a 64-bit item from the parameter to the float stack. This is necessary because a Forth double is sometimes not the same as a native hardware 64-bit number. For iForth a swap would be necessary, *i.e*:

```
FORTH> 1e PAD DF!  PAD 2@ .S 2DROP
  Data: 1072693248 0 ---
System: ---
 Float: --- ok
FORTH> 1e FPUSHD .S 2DROP
  Data: 0 1072693248 ---
System: ---
 Float: --- ok
```

See also: **FPOPS FPUSHS FPUSHD**

FPOPS "f-pop-s" **IFORTH**

(u --) (F: -- r)

Move a 32-bit item from the parameter to the float stack, i.*e*:

```
FORTH>  1e PAD SF!  PAD @ .S DROP
  Data: 1065353216 ---
System: ---
 Float: --- ok
FORTH>  1e FPUSHS .S DROP
  Data: 1065353216 ---
System: ---
 Float: --- ok
```

See also: **FPOPD FPUSHD FPUSHS**

FPU-ovf! **IFORTH**

(bool --)

Instructs the optimizer to take care that the FPU stack does not overflow. Because the optimizer is very careful, this may result in non-optimal code, so it is possible to turn this option off. It is advised to do this only for short segments of critical and carefully checked code. Set up in the ~/include/iforth.prf preferences file.

See also: **FPU-ovf@ LEA-used! EDI-used! P6-used!**

FPU-ovf@ **IFORTH**

(-- bool)

Get the current state of FPU stack overflow checking .

See also: **FPU-ovf!**

FPUSHD "f-push-d" **IFORTH**

(-- ud) (F: r --)

Move a 64-bit item from the float to the parameter stack. This is necessary because a Forth double is sometimes not the same as a native hardware 64-bit number. For iForth a swap would be necessary,

See also: **FPOPS FPOPD FPUSHS**

FPUSHS "f-push-s" **IFORTH**

(u --) (F: -- r)

Move a 32-bit item from the float to the parameter stack.

See also: **FPOPS FPOPD FPUSHD**

FRAD "f-rad" **IFORTH**

(--) (F: r1 -- r2)

r2 is the equivalent angle in radians of the angle *r1* degrees.

FREE **MEMORY**

(a-addr -- ior)

Return the contiguous region of data space indicated by *a-addr* to the system for later allocation. *a-addr* must indicate a region of data space that was previously obtained by **ALLOCATE** or **RESIZE** . The address of the next available data space location is unaffected by this operation. If the operation succeeds, *ior* is 0. If the operation fails, *ior* is the implementation-defined **I/O** result code. Since multiple processes might access the memory manager at the same time this word is protected by a semaphore.

See also: **ALLOCATE AVAILABLE RESIZE**

FROM **IFORTH**

Usage: **FROM** name

(-- x)

Fetch the value stored in *name*. *name* must be a variable created by **VALUE** or **LOCAL** . **FROM** itself only sets a flag, *name* determines what to do based on that flag. Since fetching the contents is the default action, **FROM** can be removed from the code. It is only needed to reset the flag when the flag was accidently set. Other types of variables can also use **FROM** to place their contents on the appropriate stack.

FROT "f-rote" **FLOAT**

(--) (F: r1 r2 r3 -- r2 r3 r1)

Rotate the top three floating-point stack entries.

FROUND "f-round" **FLOAT**

(--) (F: r1 -- r2)

Round *r1* using the "round to nearest" rule, giving *r2*.

FRP "f-r-p" **IFORTH-ASSEMBLER**

(-- offset)

offset is the *offset* in the workspace at which the return stack pointer is stored.

FS. "f-s-dot" **FLOAT**

(--) (F: r --)

Convert the top number on the floating-point stack to a character string using the rules of **(F.)** . and display the resulting string with a trailing space. For example, the sequence

```
4 SET-PRECISION 1.23E5 FS.
```

will display

```
123000.0000
```

An ambiguous condition exists if the system basis is not **DECIMAL** (not a problem for iForth) or if the character representation exceeds the size of the pictured numeric output string buffer.

FSCALE **IFORTH**

(n --) (F: r1 -- r2)

The floating-point number *r2* is equal to *r1**2^*n* .

FSIN "f-sine" **FLOAT EXT**

(--) (F: r1 -- r2)

r2 is the sine of the radian angle *r1*.

FSINCOS "f-sine-cos" **FLOAT EXT**

(--) (F: r1 -- r2 r3)

r2 is the sine of the radian angle *r1*. *r3* is the cosine of the radian angle *r1*.

FSINH "f-sin-h" **FLOAT EXT**

(--) (F: r1 -- r2)

r2 is the hyperbolic sine of *r1*.

FSP "f-s-p" **IFORTH-ASSEMBLER**

(-- offset)

offset is the *offset* in the workspace at which the data stack pointer is stored.

FSP! "f-s-p-store" **IFORTH**

(a-addr --)

Set the floating-point stack pointer to *a-addr*.

FSP0 "f-s-p-zero" **IFORTH**

(-- a-addr)

a-addr is the address of **FSP0** . **FSP0** contains the pointer to the base of the floating-point stack. The phrase **FSP0 @ FSP!** removes all elements from the floating-point stack.

FSP@ "f-s-p-fetch" **IFORTH**

(-- a-addr)

Get the floating-point stack pointer. Note that iForth uses stacks that grow towards lower addresses. The top element of the floating-point stack is at **FSP@** , however you should never access stacks in this way.

FSPLIT "f-split" **IFORTH**

(-- e) (F: r -- m)

The floating-point number *r* is split into a mantissa m in the range { 1 .. 2 } and an integer *e* with which to double m *e* times to get the original number *r*. *r* must not be an IEEE unnormalized number, which is approximately one over the maximum number representable.

See also: **MAX-FLOAT**

FSQR "f-square" **IFORTH**

(--) (F: r1 -- r2)

r2 is the square of *r1*. It is the same as *r1 r1* **F*** .

FSQRT "f-square-root" **FLOAT EXT**

(--) (F: r1 -- r2)

r2 is the square root of *r1*. An ambiguous condition exists if *r1* is less than zero.

FSSP "f-s-s-p" **IFORTH-ASSEMBLER**

(-- offset)

offset is the *offset* in the workspace at which the system stack pointer is stored.

FSWAP "f-swap" **FLOAT**

(--) (F: r1 r2 -- r2 r1)

Exchange the top two floating-point stack items.

FTAN "f-tan" **FLOAT EXT**

(--) (F: r1 -- r2)

r2 is the tangent of the radian angle *r1*. An ambiguous condition exists if cos(*r1*) is zero.

FTANH "f-tan-h" **FLOAT EXT**

(--) (F: r1 -- r2)

r2 is the hyperbolic tangent of *r1*.

FTL "f-t-l" **IFORTH-ASSEMBLER**

(-- offset)

offset is the *offset* in the workspace at which the task link pointer is stored.

FTRUNC "f-trunc" **IFORTH**

(--) (F: r1 -- r2)

Round *r1* using truncation, giving *r2*.

```
FORTH> 3.12e ftrunc f. 3.000000  ok
FORTH> -3.12e ftrunc f. -3.000000  ok
```

See also: **FLOOR**

FTUCK "f-tuck" **IFORTH**

(--) (F: r1 r2 -- r2 r1 r2)

Copy *r2* and tuck it under *r1* .

See also: **TUCK**

FUP "f-u-p" **IFORTH-ASSEMBLER**

(-- offset)

offset is the *offset* in the workspace where the user area starts.

FVALUE "f-value" **D** **IFORTH**

("name" --)

Parse *name* delimited by a space, ignoring leading delimiters. Create a dictionary entry for *name* with the execution semantics defined below. *name* is referred to as a "f-value".

Execution: (--) (F: -- r)

Place *r* on the floating-point stack. The value of *r* is unspecified until the phrase *r* **TO** *name* is executed, causing *r* to be associated with *name*.

FVARIABLE "f-variable" **D** **FLOAT**

("name" --)

Parse *name* delimited by a space, ignoring leading delimiters. Create a dictionary entry for *name* with the execution semantics defined below. Reserve 1 **FLOATS** address units of data space at an aligned address. *name* is referred to as an "f-variable."

Execution: ("name" -- a-addr)

a-addr is the address of the data space reserved by **FVARIABLE** when it created *name*. The application is responsible for initializing the contents of the reserved space.

F~ "f-proximate" **FLOAT EXT**

(-- flag) (F: r1 r2 r3 --)

If *r3* is positive, *flag* is true if the absolute value of (*r1* minus *r2*) is less than *r3*. If *r3* is zero, *flag* is true if the implementation-dependent encoding of *r1* and *r2* are exactly identical (positive and negative zero are unequal if they have distinct encodings) If *r3* is negative, *flag* is true if the absolute value of (*r1* minus *r2*) is less than the absolute value of *r3* times the sum of the absolute values of *r1* and *r2*.

GET-CURRENT **SEARCH**

(-- wid)

Return *wid*, the identifier of the compilation word list.

GET-ORDER **SEARCH**

(-- wid1 ... widn n)

Returns the number of word lists *n* in the search order and the word list identifiers *wid1 ... widn* identifying these word lists. *widn* identifies the word list that is searched first, and *wid1* the word list that is searched last. The search order is unaffected.

GETPRIORITY "get-priority" **IFORTH**

(-- 0|1)

Get the priority of the current process. If the current process is a high-priority process, 0 is returned, otherwise 1 is returned.

GROW **IFORTH**

(+n --)

Enlarge the dictionary space available to the system. **GROW** only works under certain conditions: **ALLOCATE** must not have been used yet, there may be no user definitions, and multitasking should be off. This limits **GROW**'s use to invocation just after booting the iForth executable. If you need more dictionary space (available to **ALLOT**) it might be better to ask your distributor for extra executables with the specified amount of free space compiled into it. Note that invoking **GROW** means the header array of iForth is moved up in memory, making all header addresses compiled into words illegal. The base system as distributed can be **GROW**n without problem, but this may not be the case after a **SAVE-SYSTEM** , depending on the code added.

GSCREEN> "graphics-screen-from" **IFORTH**

(c-addr1 c-addr2 u --)

If *u* is greater than zero, copy *u* consecutive characters from *c-addr1* to *c-addr2*. Equivalent to **CMOVE** , **CMOVE>** or **MOVE** if *c-addr1* is not in graphics memory. Only use **GSCREEN>** to do block moves from the screen to normal memory. Block moves across screen memory are not supported. They will hang the machine.

See also: **>GSCREEN**

H. "h-dot" **IFORTH**

(x --)

Display *x* as an unsigned hexadecimal number. The current number base is not changed. The format is '$dddddddd' with *d* an hexadecimal digit. For 16 bit models the format is '$dddd'.

HAND **IFORTH**

(--)

Initialize the contents of **I/O** to contain the execution tokens of the routines that are used for the standard interactive environment.

HEAD' **IFORTH**

("name" -- dea)

Parse *name* delimited by a space, ignoring leading delimiters. If *name* can be found using the current search order, leave its dictionary entry address, else generate an error condition.

See also: **'**

HEAD> "head-from" **IFORTH**

(dea -- addr)

addr is the address where the execution token of the dictionary entry corresponding to *dea* can be found.

HEAD>FLAGS **IFORTH**

(dea -- ffa)

ffa is the flag field address of the dictionary entry identified by *dea*. It is considered as an array of flags denoting properties of the dictionary entry. It can be manipulated using bit-masks. For example: the phrase flags **=ANSI AND =ANSI** = leaves true if flags are the flags from an ANS standard word.

See also: **=ANSI =COMP =IMMEDIATE =MACRO =PRIVATE =VISIBLE**

HEAD>FORGET "head-to-forget" **IFORTH**

(dea -- ffa)

ffa is the forget field address of the dictionary entry identified by *dea*.

HEAD>HASH **IFORTH**

(dea -- hfa)

hfa is the hash field address of dictionary entry identified by *dea*.

HEAD>LINK **IFORTH**

(dea -- lfa)

lfa is the link field address of the dictionary entry identified by *dea*. It contains the *dea* of the next entry in the dictionary.

HEAD>NAME "head-to-name" **IFORTH**

(dea -- nfa)

nfa is the name field address of the dictionary entry identified by *dea*. It contains a character string with the name of the dictionary entry.

HEAD>comp "head-to-compile-field" **IFORTH**

(dea -- cf)

cf is the compile field address of the dictionary entry identified by *dea*. The compile field is used in compile state and makes it unnecessary that such a word is **IMMEDIATE** . Not every word has this field.

See also: **HEAD>exec**

HEAD>exec "head-to-execute-field" **IFORTH**

(dea -- ef)

ef is the execution field address of the dictionary entry identified by *dea*. The execution field is used in execution state and makes it unnecesary that such a word is **IMMEDIATE** . TOKENIZEd words have this field.

See also: **HEAD>exec**

HELP **IFORTH**

("name" --)

Parse *name* delimited by a space ignoring leading delimiters. Find an entry in the glossary help file forth.hlp and list that entry to the screen. **HELP** is not part of the kernel system but can be loaded by including the file help.frt. By default the file forth.hlp contains a copy of the glossary of Forth words from your manual. Thus **HELP** provides an online manual for Forth words. Note that any typesetting that is in the manual is removed so only ASCII text is left. The text in the help file is automatically generated from the original glossary documents, so there are no textual differences between the printed and the online version of the manual.

HERE **CORE**

(-- addr)

addr is the address of the next available data space location.

HEX **CORE EXT**

(--)

Set contents of **BASE** to sixteen.

HI-PRIO **IFORTH**

(-- 0)

Place the number denoting a high priority process on the stack. This number is the constant zero.

HIDE **IFORTH**

("name" --)

Parse *name* delimited by a space ignoring leading delimiters. If *name* can not be found using the current search order generate an error condition. Otherwise clear the visibility bit in *name*'s header. This action will make *name* unaccessible to iForth's compiler and interpreter.

See also: **PRIVATE DEPRIVE =VISIBLE**

HLD **IFORTH**

(-- a-addr)

a-addr is the address of **HLD** . **HLD** contains the pointer to the first character in the numeric conversion buffer. This variable is reset by **<#** and it is updated by **HOLD** .

HOLD **CORE**

(char --)

Add *char* to the beginning of the pictured numeric output string. Typically used between **<#** and **#>** .

I **C** **CORE**

(-- n|u) (R: sys -- sys)

n|u is a copy of the current (innermost) loop index. The loop control parameters must be available. Because an ANS system must support both **LOOP** and **+LOOP** , it is impossible for a one-pass compiler to generate optimal code for both **DO** ... **LOOP** and **DO** ... **+LOOP** type loops. Especially, access to the loop control parameter **I** is relatively slow. For time-critical applications it is recommended to use **FOR** ... **NEXT** type loops.

I' **C** **IFORTH**

(-- u) (R: sys -- sys)

u is a copy of the current (innermost) loop limit of any **DO** or **FOR** loop. The loop control parameters must be available.

See also: **DO LOOP FOR NEXT I I'' J J' J'' K**

I'' **C** **IFORTH**

(-- u) (R: sys -- sys)

u is a copy of the third loop parameter of any **DO** or **FOR** loop. The loop control parameters must be available.

See also: **DO LOOP FOR NEXT I I' J J' J'' K**

I/O **IFORTH**

(--)

Undocumented experimental word.

See also: **:FAST FAST; FI/FO [XCOMPILE]**

I@ **C** **IFORTH**

(-- u) (R: sys -- sys)

u is a copy of the current (innermost) loop counter of a **FOR** ... **NEXT** loop. The loop control parameters must be available. The loop counter counts down to zero.

See also: **FOR NEXT**

ID$ "i-d-string" **IFORTH**

(dea -- #spaces c-addr u)

Leave the name of the word whose name field address is *dea*, ready to be used by **TYPE** , followed by enough (*#spaces*) spaces to fill a field of *n**18 positions (*n* is arbitrary). If the name is 'smudged', *u* is 0 and *#spaces* is 18.

See also: **.ID =VISIBLE**

IDATA! "internal-data-send" **IFORTH**

(--)

Send the 'data follows' command over the boot link to the server. A data block should be sent next, formatted as a count followed by a string. This data block will be acted on by the server.

IDLITERAL "i-d-literal" **IFORTH**

(d --) or (d -- d)

When the contents of the variable **STATE** is not zero, perform the function of **SWAP 2LITERAL** . Otherwise leave *d* on the stack. **IDLITERAL** is an immediate command.

See also: **ILITERAL LITERAL 2LITERAL**

IEXE! "internal-exe-send" **IFORTH**

(--)

Send the 'a command follows' command over the link to the server. A function number is sent next. The available functions are listed in appendix **F**.

IF **C** **CORE**

Compilation: (-- orig)

Put the location of a new unresolved forward reference *orig* onto the control flow stack. Append the execution semantics given below to the current definition. The semantics are incomplete until *orig* is resolved (*e*.g., by **THEN**).

Execution: (x --)

If all bits of *x* are zero, continue execution at the location specified by the resolution of *orig*.

See also: **ELSE THEN**

IF, **IFORTH-ASSEMBLER**

Assembling: (-- addr)

Place the value of the current code space pointer on the data stack. Assemble a conditional forward branch that is to be resolved by either else, or then, .

Execution: (--)

If flag is true, take the branch. Otherwise do not take the branch.

IFF! "internal-f-f-send" **IFORTH**

(--)

All server commands are introduced on the link by a byte with the value $FF, all other bytes are interpreted as bytes to be send to the screen by the server. **IFF!** sends a command to the server which instructs the server to write the byte with the value $FF to the screen. The command consists of 2 bytes $FF, the first one introduces a command and the second is the command number.

IFORTH **IFORTH**

IFORTH is an environment query.

See also: **ENVIRONMENT?**

IFORTH! "internal-forth-send" **IFORTH**

(--)

Send the 'a forth command follows' command over the link to the server. A Forth command should be sent next, formatted as a count followed by a string. This Forth command will be executed by the server. This Forth command will be executed only on iForth versions that use a server that is equipped with a Forth compiler/interpreter.

ILITERAL "i-literal" **IFORTH**

(n --) or (n -- n)

When the contents of the variable **STATE** is not zero, perform the function of **LITERAL** . Otherwise leave *n* on the stack. **ILITERAL** is an immediate command.

IMMEDIATE **CORE**

(--)

Mark the most recently created dictionary entry as an immediate word.

IN/OUT **IFORTH**

(#in #out --)

Used in inline macro's to optimize parameter access. *#in* is the number of integer items expected in registers on entry. Presently *#in* can only be 0, 1 or 2. If it is 1, the top of stack is expected in the EBX register, if it is **2**, TOS and NOS are expected in EBX and ECX respectively (newer versions of iForth allow more registers). *#out* is the number of parameters held in registers on exit of the macro. Presently *#out* can only be 0, 1 or 2. If it is 1, EBX holds the current top of stack, if it is 2 EBX holds TOS and ECX holds NOS. The word **ADJUST-STACK** must be used together with **IN/OUT** to flush registers to the real data stack after a macro (or set of macro's) has finished. **IN/OUT** and **ADJUST-STACK** work closely together in order to remove superfluous data stack traffic at macro boundaries. Example:

```
: add       1 1 IN/OUT
            POSTPONE ASM{ eax pop, ebx add, }ASM
            ADJUST-STACK ; IMMEDIATE COMPILE-ONLY
: 3add      add add ;
```

The generated code for 3add is:

```
"ebx pop, eax pop, ebx add,  eax pop, ebx add,  ebx push,"
```

See also: **ADJUST-STACK FIN/FOUT**

INCLUDE **IFORTH**

*(i*x "file" -- j*x)*

Parse file delimited by a space, ignoring leading delimiters. Open the file for read-only. Execute all commands that are inside this file. Close the file.

INCLUDE-FILE **FILE**

*(i*x fileid -- j*x)*

Save the specification of the input stream, including the current value of SOURCE-FILE . Set the value returned by SOURCE-FILE to *fileid*. Store 0 in **BLK** . Repeat until end of file: Read a line from the file, fill the input stream from the contents of that line, and interpret the input stream. Interpretation begins at the file position where the next file read would occur. When the end of the file is reached, close the file and restore the specification of the input stream to its saved value. After an error occurs, all files that were being interpreted are closed. Note however that other implementations of the standard may leave these files open. An ambiguous condition exists if *fileid* is invalid, if there is an **I/O** error reading *fileid*, or if there is an **I/O** error closing *fileid*.

INCLUDED **FILE**

*(i*x c-addr u -- j*x)*

Save the specification of the input stream, including the current value of SOURCE-FILE . Open the file specified by *c-addr* and *u* and change the value of SOURCE-FILE to the file's *fileid*. Store zero in **BLK** . Repeat until end of file: Read a line from the file, fill the input stream from the contents of that line, and interpret the input stream. Interpretation begins at the file position where the next file read would occur. When the end of the file is reached, close the file and restore the specification of the input stream to its saved value. An ambiguous condition exists if the named file can not be opened, if an **I/O** error occurs reading the file, or if an **I/O** error occurs while closing the file. When an ambiguous condition exists, the status (open or closed) of any files that were being interpreted is implementation-defined. File input (as with **INCLUDE-FILE**), block input (as with **LOAD**), and string input (as with **EVALUATE**) may be nested in any order.

See also: **INCLUDE-FILE**

INIT-MEM **IFORTH**

(--)

Initialize the memory manager. All allocated blocks are freed. Indiscriminate use of **INIT-MEM** will crash the system if any processes are still running. The memory manager handles all memory between the end of the default terminal input buffer and **MEMSTART @** . Put a new value into **MEMSIZE** and execute **COLD** when the amount of memory available is not equal to the 1 Mbyte expected by the standard binaries. Changing **MEMSIZE** can be used to lock iForth out of memory used by alien processor programs.

See also: **ALLOCATE FREE MEMSIZE MEMSTART /SYSTEM**

INSET? **IFORTH-ASSEMBLER**

(x a-addr -- flag)

a-addr is a list of cells with the first cell containing the number of the cells in the list. *flag* is true when *x* is an element of this list. Otherwise *flag* is false.

INVERT **CORE**

(x1 -- x2)

Invert all bits of *x1*, giving *x2*.

IO-SYSTEM "i-o-system" **IFORTH**

(-- xt)

Returns the *xt* of either **OS-IO** or **FORTH-IO** , depending on which set of **I/O** vectors is current.

See also: **IO-SYSTEM** **OS-IO**

IS-FORGET **IFORTH**

(xt "name" --)

Parse *name* delimited by blanks. If *name* is not found, issue an error message, otherwise fill the forget field of *name* with the execution token *xt*. The data field address (dfa) of the word that is forgotten is put on the data stack for use by the definition that corresponds to the execution token *xt*. Consumption of this data field address is mandatory. Note that **IS-FORGET** can attach a forget field to words that have no data field, because they were not defined by **CREATE** or words that use **CREATE** . The only action allowed on such an invalid dfa is to drop it.

See also: **FORGET>**

ITERM! **IFORTH**

(--)

Send the 'terminal command follows' command over the link to the server. A function number should be send after this. A list of available commands is listed in appendix **F.**

IUSER! **IFORTH**

(--)

Send the 'user command follows' command over the link to the server. A user function number should be send next. A list of available commands is listed in appendix **F.**

J **C** **CORE EXT**

(-- n|u) (R: sys -- sys)

n|u is a copy of the index of the next outer loop. Syntax: May only be used with a nested **DO** ... **LOOP** , **DO** ... **+LOOP** , **?DO** ... **LOOP** , or **?DO** ... **+LOOP** in the form, for example:

```
: X ... DO ... DO ... J ... LOOP ... +LOOP ... ;
```

The loop control parameters of the next outer nested loop must be available.

See also: **I** **K**

J' **C** **IFORTH**

(-- n|u) (R: sys -- sys)

n|u is a copy of the loop limit of the next outer loop. Syntax: May only be used with a nested **DO** ... **LOOP** , **DO** ... **+LOOP** , **?DO** ... **LOOP** , or **?DO** ... **+LOOP** in the form, for example:

```
: X ... DO ... DO ... J' ... LOOP ... +LOOP ... ;
```

The loop control parameters of the next outer nested loop must be available.

See also: **I** **I'** **I''** **J** **J''** **K**

J" **C** **IFORTH**

(-- n|u) (R: sys -- sys)

n|u is a copy of the third parameter of the next outer loop. Syntax: May only be used with a nested **DO** ... **LOOP** , **DO** ... **+LOOP** , **?DO** ... **LOOP** , or **?DO** ... **+LOOP** in the form, for example:

```
: X ... DO ... DO ... J'' ... LOOP ... +LOOP ... ;
```

The loop control parameters of the next outer nested loop must be available.

See also: **I I' I'' J J' K**

K **C** **IFORTH**

(-- n|u) (R: sys -- sys)

n|u is a copy of the index of the second outer loop. Syntax: May only be used with a double nested **DO** ... **LOOP** , **DO** ... **+LOOP** , **?DO** ... **LOOP** , **?DO** ... **+LOOP** in the form, for example:

```
: X ... DO ... DO ... DO ... K ... LOOP ... +LOOP ... LOOP ;
```

The loop control parameters of the second outer loop must be available.

See also: **I J**

KEY **CORE**

(-- char)

Receive one character *char*, a member of the 8-bit ASCII character set. Keyboard events that do not correspond to such characters are discarded until a valid character is received, and those events are subsequently unavailable. All standard characters can be received. Characters received by **KEY** are not displayed. The ability to receive control characters is an environmental dependency, however this system returns the value of control characters correctly.

See also: **EKEY KEY?**

KEY? "key-question" **FACILITY EXT**

(-- flag)

If a character is available, return true. Otherwise, return false. If non-character keyboard events are available before the first valid character, they are discarded, and are subsequently unavailable. After **KEY?** returns with a value of true, subsequent executions of **KEY?** prior to the execution of **KEY** or **EKEY** also return true, without discarding keyboard events. The next execution of **KEY** will return the character without indefinite delay.

LCMOVE> "low-cmove-from" **IFORTH**

(l-addr addr u --)

Move *u* bytes from *offset* l-*addr* in the first physical Megabyte of memory to *addr*. MS-DOS only.

LEA-used! **IFORTH**

(bool --)

Instructs the optimizer to use the LEA instruction as much as possible. Do not use lightly. Set up in the ~/include/iforth.prf preferences file.

See also: **FPU-ovf! EDI-used! P6-used! LEA-used@**

LEA-used@ **IFORTH**

(-- bool)

Get the current state of LEA instruction handling.

See also: **LEA-used!**

LEAVE **C** **CORE**

(--) (R: sys --)

Discard the current loop control parameters. The loop parameters must have been available. Continue execution immediately following the syntactically enclosing **DO** ... **LOOP** or **DO** ... **+LOOP** . Syntax: May only be used with a **DO** ... **LOOP** , **DO** ... **+LOOP** , **?DO** ... **LOOP** , or **?DO** ... **+LOOP** in the form, for example:

```
DO ... IF ... LEAVE THEN ... LOOP
```

See also: **+LOOP** **LOOP**

LINK>HEAD "link-to-head" **IFORTH**

(lfa -- dea)

dea is the dictionary entry address of the word for which *lfa* is the link field address.

LINUX? **IFORTH**

(-- bool)

TRUE if this is iForth for Linux.

See also: **MS-DOS?** **TFORTH?** **WINNT?** **XLINUX?**

LIST **BLOCK EXT**

(u --)

Display block *u* in an implementation-defined format. Store *u* in **SCR** . iForth lists blocks with 16 lines of 64 characters. For an example see the output below. The screen and line numbers are always printed in decimal.

See also: **BLOCK**

```
SCR#10
 0 ( Eratosthenes sieve benchmark program )
 1
 2 DECIMAL
 3 8190 CONSTANT SIZE CREATE FLAGS SIZE ALLOT
 4
 5 : DO-PRIME ( --- ) FLAGS SIZE 1 FILL
 6 0 SIZE 0
 7 DO FLAGS I + C@
 8 IF I 2* 3 + DUP I +
 9 BEGIN DUP SIZE U<
10 WHILE 0 OVER FLAGS + C! OVER +
11 REPEAT 2DROP 1+
12 THEN
13 LOOP ( . ."Primes "      ) DROP ;
14 : BENCHMARK 25 0 DO DO-PRIME LOOP ( 7 EMIT ) ;
15 : T 0 DO BENCHMARK LOOP CR ."   1899 primes." 7 EMIT ;
```

LITERAL **C** **CORE**

Compilation: (x --)

Compile *x* as a literal. in iForth **LITERAL** is immediate and compile-only. To 'comma' a value interpretively, use **LITERAL,** .

Execution: (-- x)

Place *x* on the stack.

See also: **ALITERAL ILITERAL LITERAL,**

LITERAL, **IFORTH**

Compile x as a literal. This is only useful when in compile state.

Execution: (-- x)

Place *x* on the stack.

See also: **ALITERAL ILITERAL LITERAL**

LN2 "l-n-two" **IFORTH**

(--) (F: -- ln(2))

Place the constant value ln(2) (0.69314718056...) on the floating-point stack.

LO-PRIO **IFORTH**

(-- 1)

Place the number denoting a low priority process on the stack. This number is the constant one.

LOAD **BLOCK EXT**

*(i*x u -- j*x)*

Save the current input stream specification. Make block *u* the current input stream and interpret the contents. When the input stream is exhausted, restore the prior input stream specification. An ambiguous condition exists if *u* is not a valid block number. If it is zero nothing happens, but for other standard implementations this may be an ambiguous condition. If the contents of **OFFSET** plus *u* point to a block that is not in the block file an end-of-file message is printed and the system aborts.

LOCAL **D** **IFORTH**

Compilation: ("name" --)

Parse *name* delimited by a space, ignoring leading delimiters. Create a temporary dictionary entry for *name* with the execution semantics defined below. **LOCAL** can only be used inside a definition. *name* remains in the dictionary until the current definition is finished with ; **;CODE** or **DOES>** .

Execution: (x --)

Associate the stack value *x* with *name*.

Execution: ("name" -- x)

Place *x* on the stack.

LOCALS **IFORTH**

LOCALS is an environment query.

See also: **ENVIRONMENT?**

LOCALS-EXT **IFORTH**

LOCALS-EXT is an environment query.

See also: **ENVIRONMENT?**

LOCALS| **C** **IFORTH**

*Compilation: ("name"*n "|" --)*

Parse names delimited by a blank, ignoring leading delimiters. The list of names is terminated by a '|' (bar). Create a temporary dictionary entry for each *name* with the execution semantics defined below. **LOCALS|** can only be used inside a definition. The names remain in the dictionary until the current definition is finished with ; **;CODE** or **DOES>** . iForth accepts an unlimited number of names. The ANS standard requires a minimum of eight.

Execution: ("name" -- x)

Place *x* on the stack. The value of *x* is unspecified until the phrase *x* **TO** *name* is executed, causing *x* to be associated with *name*.

LOOP **C** **CORE**

Compilation: (dodest --)

Resolve the destination of all unresolved occurrences of **LEAVE** between the location given by *dodest* and the next location for a transfer of control, to execute the words following the LOOP. Append the execution semantics given below to the current definition.

Execution: (--) (R: sys1 -- | sys2)

Loop control parameters must be available. Add one to the loop index. If the loop index is then equal to the loop limit, discard the loop parameters and continue execution immediately following the loop. Otherwise continue execution at the beginning of the loop.

See also: **DO I LEAVE**

LOOPe "loop-e" **C** **IFORTH**

Compilation: (dodest --)

Execution: (n --) (R: sys -- | sys2)

Like **LOOP** , but does not "wrap around." Use with **DO** .

See also: **dDO uDO +LOOPu +LOOP LOOPe**

LSHIFT **CORE**

(x1 n -- x2)

Perform a logical shift of *n* bit-places on *x1*, giving *x2*. Shift the bits *n* places toward the most significant bit. Put zero into the places "uncovered" by the shift.

LSP! "l-s-p-store" **IFORTH**

(a-addr --)

Set the local stack pointer to *a-addr*.

LSP0 "l-s-p-zero" **IFORTH**

(-- a-addr)

a-addr is the address of **LSP0** . **LSP0** contains the pointer to the base of the locals stack. The phrase **LSP0 @ LSP!** removes all elements from the locals stack.

LSP@ "l-s-p-fetch" **IFORTH**

(-- a-addr)

Get the local stack pointer. Note that iForth uses stacks that grow towards lower addresses. The top element of the local stack is at **LSP@** , however you should never access stacks in this way.

M* "m-star" **CORE**

(n1 n2 -- d)

d is the signed product of *n1* times *n2*.

M*/ "m-star-slash" **DOUBLE**

(d1 n1 +n2 -- d2)

Multiply *d1* by *n1* producing the triple-cell intermediate result t. Divide t by *+n2* giving the double-cell quotient *d2*. An ambiguous condition exists if *+n2* is zero, or the quotient lies outside of the range of a double-precision signed integer.

M+ "m-plus" **DOUBLE**

(d1|ud1 n -- d2|ud2)

Add *n* to *d1|ud1*, giving the sum *d2|ud2*.

M- "m-minus" **IFORTH**

(d1|ud1 n -- d2|ud2)

Subtract *n* from *u1|ud1*, giving the result *d2|ud2*.

M/ "m-slash" **IFORTH**

(d1|ud1 n -- quot)

Equivalent to **UM/MOD NIP** .

MAP-FILE **IFORTH**

(file-id -- c-addr u1 ior)

Return the internal buffer address connected to file-id. The buffer contains all the data in the file. Only OPEN-FILEd file-ids can be mapped, and only in the case that the file is **R/O** or **R/O BIN** .

See also: **CREATE-FILE OPEN-FILE**

MARKER **D** **CORE EXT**

("name" --)

Parse *name* delimited by a space, ignoring leading delimiters. Create a dictionary entry for *name* with the execution semantics defined below.

Execution: ("name" --)

Restore all dictionary allocation and search order pointers to the state they had just prior to the definition of *name*. Remove *name* and all subsequent word definitions. Restoration of any structures still existing that may refer to deleted definitions or deallocated data space is not provided. The forget fields can be used for this. No other contextual information such as numeric base is affected.

MAX **CORE**

(n1 n2 -- n3)

n3 is the greater of *n1* and *n2*.

MAX-CHAR **IFORTH**

MAX-CHAR is an environment query.

See also: **ENVIRONMENT?**

MAX-D **IFORTH**

MAX-D is an environment query.

See also: **ENVIRONMENT?**

MAX-FLOAT **IFORTH**

MAX-FLOAT is an environment query.

See also: **ENVIRONMENT?**

MAX-N **IFORTH**

MAX-N is an environment query.

See also: **ENVIRONMENT?**

MAX-U **IFORTH**

MAX-U is an environment query.

See also: **ENVIRONMENT?**

MAX-UD "max-u-d" **IFORTH**

MAX-UD is an environment query.

See also: **ENVIRONMENT?**

MAXINT "max-int" **IFORTH**

(-- maxint)

MAXINT places the value *maxint* on the data stack. *maxint* is the largest representable integer for the current architecture. It has the value $7FFF.FFFF on Intel 32-bit architectures and the value $7FFF on Intel 16-bit architectures.

MEMORY-ALLOC **IFORTH**

MEMORY-ALLOC is an environment query.

See also: **ENVIRONMENT?**

MEMORY-ALLOC-EXT **IFORTH**

MEMORY-ALLOC-EXT is an environment query.

See also: **ENVIRONMENT?**

MEMSIZE "mem-size" **IFORTH**

(-- a-addr)

a-addr is the address of **MEMSIZE** . **MEMSIZE @** returns the maximum amount of available memory, *i.e*. the iForth memory manager will control all memory between **MEMSTART @** and **MEMSTART @ MEMSIZE @** + . You should change the value of **MEMSIZE** and execute **INIT-MEM** if the default amount of memory is not needed by iForth or if that memory should not be managed by iForth.

See also: **INIT-MEM MEMSTART AVAILABLE**

MEMSTART "mem-start" **IFORTH**

(-- a-addr)

a-addr is the address of **MEMSTART** . **MEMSTART** delimits available memory, *i.e*. the iForth memory manager will control all memory between **MEMSTART @** and **MEMSTART @ MEMSIZE @** + . You should change the value of **MEMSIZE** and execute **INIT-MEM** if the default amount of memory is not needed by iForth or if that memory should not be managed by iForth.

See also: **INIT-MEM**

MIN **CORE**

(n1 n2 -- n3)

n3 is the lesser of *n1* and *n2*.

MININT "min-int" **IFORTH**

(-- minint)

MININT Places the value *minint* on the data stack. *minint* is the minimal representable integer for the current architecture. It has the value -$8000.0000 on Intel 32-bit architectures and the value -$8000 on Intel 16-bit architectures.

MOD **CORE**

(n1 n2 -- n3)

Divide *n1* by *n2*, giving the single-cell remainder *n3*. An ambiguous condition exists if *n2* is zero. If *n1* and *n2* differ in sign, the result returned will be the same as the phrase **>R S>D R>** SM/MOD **DROP** . Note that other implementations of the standard may return the equivalent of **>R S>D R> FM/MOD DROP** .

MOVE **CORE**

(addr1 addr2 u --)

If *u* is greater than zero, copy the contents of *u* consecutive address units at *addr1* to the *u* consecutive address units at *addr2*. After **MOVE** completes, the *u* consecutive address units at *addr2* contain exactly what the *u* consecutive address units at *addr1* contained before the move.

MS "m-s" **FACILITY EXT**

(u --)

Wait at least *u* milliseconds. Note: The actual length and variability of the time period depends upon the implementation-defined resolution of the system clock. The system call used on the PC is not accurate enough to measure an interval of one millisecond accurate to one percent.

MS-DOS? **IFORTH**

(-- bool)

TRUE if this is iForth for MS-DOS.

See also: **LINUX? TFORTH? WINNT? XLINUX?**

MT* "m-t-star" **IFORTH**

(lo hi n -- tlo tmid thi)

Signed multiplication of a double *hi*:*lo* with *n*, returning a triple result.

See also: **UT* UT/**

NEAREST-HEAD **IFORTH**

(addr -- dea u)

Finds the Forth word whose code starts nearest to *addr*. If *addr* is outside the code space the returned *dea* is the dictionary entry address of the latest word defined. The *offset* of *addr* from the found word's *xt* is returned as *u*. The *dea* of the latest word defined is used when no suitable word is found. In that case *u* is a very large number.

See also: **>HEAD**

NEEDS **IFORTH**

("name" --)

Specifies that the current module depends of the module with the **REVISION** *name* given. Assumes the convention that the file name of a module can be found from the **REVISION** *name* by removing the leading '-' and extending it with '.frt'. Example:

```
NEEDS -fred
```

This example tries to load fred.frt if -fred is not found in the dictionary. Note: the extension is in lower case. The host system may use case sensitive file names.

NEGATE **CORE**

(n1 -- n2)

n2 is the negation of *n1*.

NESTING **IFORTH**

(-- a-addr)

a-addr is the address of **NESTING** . **NESTING** contains the number of nested non-interactive interpreters.

NETWORK-INFO@ **IFORTH**

(-- me total)

total is the total number of processors that are available in the current network. *me* is the number of the current processor. This number is always in the range { 0 .. total - 1 }.

NEXT **C** **IFORTH**

(--)

Compilation:(dodest --)

Append the execution semantics given below to the current definition.

Execution: (--) (R: sys1 -- | sys2)

The loop control parameters must be available. If the loop index is equal to 0, discard the loop parameter and continue execution immediately following the loop. Otherwise subtract one from the loop index and continue execution at the beginning of the loop.

See also: **FOR FOR?**

NIP **CORE EXT**

(x1 x2 -- x2)

Drop the first item below the top of stack.

NOOP **IFORTH**

(--)

Does nothing, but is sure to be called because it is never optimized away.

NOT **IFORTH**

(n1 -- n2)

Equivalent to **0=**

See also: **0= INVERT**

NP "n-p" **IFORTH**

(-- a-addr)

a-addr is the address of **NP** . **NP** contains a pointer to next byte to be allocated for the name table.

NUMBER? **IFORTH**

(c-addr u -- c-addr 0 | n 1 | d 2) or (c-addr u -- 9) (F: -- r)

c-addr and *u* is the start address of a counted string. An attempt is made to convert the string into a literal. First an attempt is made to convert the string to a single number, then an attempt is made to convert the string to a double number and finally it is attempted to convert the string to a floating point number. If all conversions fail, return *c-addr u* and the constant 0. Otherwise place the value of the conversion on the appropriate stack and push the constant 1, 2 or 9 for a single number, a double number or a floating point number. **'NUMBER** contains the execution token of **NUMBER?** when a standard iForth system is booted. The system can be made to recognize additional number types by defining the new word XNUMBERS? . This word first calls the old **NUMBER?** and reacts on a (*c-addr u* 0) with a special parsing routine. The iForth compiler will compile as many literals as **NUMBER?** tells it there are. So if XNUMBERS? returns (*c-addr u -- n1 n2 n3* 3) the effect will be *n1* **LITERAL** *n2* **LITERAL** *n3* **LITERAL** . The maximum number of LITERALs that can be compiled in this way is 8. This setup does not work for floating-point literals, so (*c-addr* -- 10) (F: -- *r1 r2*) will not result in *r1* **FLITERAL** *r2* **FLITERAL** , but generates an error condition instead.

See also: **>NUMBER CONVERT 'NUMBER**

OCTAL **IFORTH**

(--)

Set the contents of **BASE** to eight.

OF **C** **CORE EXT**

Compilation: (case-sys1 -- case-sys2 of-sys)

The location of the new unresolved forward reference goes onto the control flow stack. Append the execution semantics given below to the current definition.

Execution: (x1 x1 --) or (x1 x2 -- x1)

If the two values on the stack are not equal, discard the top value and continue execution at the location specified by the consumer of *orig* (*e*.g., following the next **ENDOF**). Otherwise, discard both values and continue execution in line.

See also: **CASE ENDCASE ENDOF**

OFF **IFORTH**

(a-addr --)

Store false in the cell at address *a-addr*.

OFFSET **IFORTH**

(-- a-addr)

a-addr is the address of **OFFSET** . **OFFSET** contains an *offset* that is added to every block number passed to a word of the **BLOCK** and **BLOCK** EXT wordsets. This variable makes it possible to make programs unaware of their exact location in the block storage.

ON **IFORTH**

(a-addr --)

Store true in the cell at address *a-addr*.

ONLY **SEARCH EXT**

(--)

Set the search order to the implementation-defined minimum search order. The minimum search order must include the ability to interpret the words **FORTH-WORDLIST** and **SET-ORDER** .

OPEN-FILE **FILE**

(c-addr u x1 -- x2 ior)

Open the file named in the character string specified by *c-addr u*, with file access method indicated by *x1*. The meaning of the values of *x1* is implementation-defined. If the file is successfully opened, *ior* is zero, *x2* is the *fileid*, and the file has been positioned to the start of the file. Otherwise *ior* is the implementation-defined **I/O** result code and *x2* is an unspecified value. In ANS Forth, trying to open a file that does not exist is an ambiguous condition. iForth will *not* perform the equivalent of **CREATE-FILE** in such cases.

See also: **CREATE-FILE MAP-FILE**

OPERATING-SYSTEM **IFORTH**

OPERATING-SYSTEM is an environment query.

See also: **ENVIRONMENT?**

OR **CORE**

(x1 x2 -- x3)

x3 is the bit-by-bit inclusive-or of *x1* with *x2*.

ORDER **SEARCH EXT**

(--)

Identify the word lists in the search order in their search order sequence, from first searched to last searched. Also identify the word list into which new definitions will be placed. The form of the identification is implementation-dependent.

OS-IO "o-s-i-o" **IFORTH**

(--)

Sets the **I/O** vectors to special routines. These routines are the most general possible on the given platform. For the MS-DOS IFORTH the disadvantage of **OS-IO** is that it is very slow. The Linux and WIN95/NT implementations of **OS-IO** are equal to **FORTH-IO** but are quite fast. The switched vectors are: **'KEY? 'KEY 'EMIT 'EMIT? 'TYPE 'AT-XY '?AT** and **'PAGE** .

See also: **IO-SYSTEM FORTH-IO**

OVER **CORE**

(x1 x2 -- x1 x2 x1)

Place a copy of *x1* on top of the stack.

P! "port-store" **IFORTH**

(n port# --)

Output the 16-bit number *n* to the 16-bit port with address *port#* .

P6-used! **IFORTH**

(bool --)

Instructs the optimizer to use instructions that work only for the P6 CPU and above. Do not use lightly. Set up in the ~/include/iforth.prf preferences file.

See also: **FPU-ovf! LEA-used! EDI-used! P6-used@**

P6-used@ **IFORTH**

(-- bool)

Get the current state of special instruction handling.

See also: **P6-used!**

P@ "port-fetch" **IFORTH**

(port# -- n)

Input the 16-bit number *n* from the 16-bit port with address *port#* .

PACK **IFORTH**

(c-addr1 u c-addr2 -- c-addr2)

Copy the string specified by *c-addr1 u* to a counted string at the character address *c-addr2*. Return *c-addr2*.

PAD **CORE EXT**

(-- c-addr)

c-addr is the address of a transient region that can be used to hold data for intermediate processing. The area is safe for use in a multi-process environment.

PAGE **FACILITY EXT**

(--)

Move to another page for output. Actual function depends on the output device. On a terminal, **PAGE** clears the screen and resets the cursor position to the upper left corner. On a printer, **PAGE** performs a form feed.

See also: **CLS**

PANIC **IFORTH**

(--)

iForth is written in such a way that pressing ^**C** , ^BREAK, or causing a GO32 exception (MS-DOS) will eventually lead to the execution of iForth's **ABORT** . However, directly calling **ABORT** causes an infinite loop with errors that demolish **CATCH** frames on the Forth return stack. (GO32 exceptions can be that serious.) Therefore we call **PANIC** , not **ABORT** . **PANIC** is defined as

```
: PANIC 'PANIC @ EXECUTE ;
```

The default content of **'PANIC** is the address of **ABORT** (so you get an occasional infinite loop). Don't worry, the word **CALM** is available: ' **CALM 'PANIC** ! . **CALM** waits for a keypress. If this is 'E' (uppercase E) iForth exits to the **OS** with errorlevel $FF. Any other key and **ABORT** will be executed instead. See **CALM** 's documentation on how to customize it. Note that **'PANIC** is _not_ reset to **ABORT** when you do **SAVE-SYSTEM** . Also note that **'PANIC** is (purposely) not a **USER** variable. With GO32 iForth's ^**C** and ^Break processing is erratic at times, in the sense that the DOS-extender doesn't seem to notice the break signal. You have to wait for the error message of the extender before pressing 'E' or some other key. A fix is to press the CTRL, ALT of SHIFT key once or twice (this generates an interrupt but doesn't send a real key message). Error messages of the extender can not be turned off, sorry.

See also: **'PANIC CALM**

PARAMS| **C** **IFORTH**

*Compilation: ("name"*n "|" --)*

Parse names delimited by a blank, ignoring leading delimiters. The list of names is terminated by a '|' (bar). Create a string substitution for each *name*, mapping it on the set **%a** .. %f. . **PARAMS|** can only be used inside a definition, and only once. The names remain aliased with **%a** .. **%f** until the current definition is finished with ; **;CODE** or **DOES>** . This mechanism is many times faster than **LOCALS|** . iForth accepts only six names, and the **'OF** operator is not supported. Do not use any form of **LOCALS|** (DLOCALS| **FLOCALS|** ..) together with **PARAMS|** . Note that, in contrast to **LOCALS|** , names are associated left to right: { : SUMMA 11 22 **PARAMS|** ape bear | ape . bear . ;

SUMMA \ prints 11 22 ok

Execution: ("name" -- x)

Place *x* on the stack. The value of *x* is unspecified until the phrase *x* **TO** *name* is executed, causing *x* to be associated with *name*.

See also: **LOCALS| (1)PARAMS %a**

PARSE **CORE EXT**

(char "ccc<char>" -- c-addr u)

char is a single character delimiter. Parse characters *ccc* delimited by *char*, leading delimiters. *c-addr* is the address within the input stream and *u* is the length of the parsed string. If the current input stream was empty, the resulting string has a zero length.

PASCAL-CALLBACK **IFORTH**

(addr #args "name" --)

This is a CREATEing word that generates a word called *name*. (The stack diagram applies to **CREATE** time.) In use it is exactly the same as **CALLBACK** . This word implements the calling convention of Visual Basic. See the Neural Net demo in ./examples/nn/nn51.frt for an example of use. Visual Basic assumes the callee cleans up the stack, but the order of the arguments is still the same as for "**C**" DLLs. The stack cleanup is an internal detail of the **PASCAL-CALLBACK** mechanism, from Forth a **CALLBACK** and **PASCAL-CALLBACK** have the same interface (when applied to external code that really expects this convention!)

See also: **FCALLBACK CALLBACK FOREIGN**

PAUSE **IFORTH**

(--)

Give other processes a chance to run.

PC! "port-cstore" **IFORTH**

(n port# --)

Output the 8-bit number *n* to the 8-bit port with address *port#* .

PC@ "port-cfetch" **IFORTH**

(port# -- n)

Input the 8-bit number *n* from the 8-bit port with address *port#* .

PEEK-CHAR **IFORTH**

(-- char | -1)

Gets next *char* from input stream without removing it. Returns -1 when the input stream is empty.

PERFORM **IFORTH**

(?? a-addr -- ??)

Fetch the execution token stored at *a-addr*. Execute the definition specified by the execution token unconditionally.

See also: **EXECUTE @EXECUTE**

PI **IFORTH**

(--) (F: -- π)

Place the constant value π or pi (3.14159265358979323846264...) on the floating-point stack.

PI*2 "pi-star-two" **IFORTH**

*(--) (F: -- π*2)*

Place the constant value 2π or pi*2 (6.2831...) on the floating-point stack.

PI/2 "pi-slash-two" **IFORTH**

(--) (F: -- π /2)

Place the constant value π /2 or pi/2 (1.5707...) on the floating-point stack.

PI/4 "pi-slash-four" **IFORTH**

(--) (F: -- π /4)

Place the constant value π /4 or pi/4 (0.7853...) on the floating-point stack.

PICK **CORE EXT**

(x(u) ... x(1) x(0) u -- x(u) ... x(1) x(0) x(u))

Remove *u*. Copy the *u*th item to the top of the stack. An ambiguous condition exists if there are less than *u*+2 items on the stack before **PICK** is executed.

PID "p-i-d" **IFORTH**

(-- a-addr)

a-addr is the workspace pointer of the current process. iForth never visibly changes the workspace pointer for any process. *a-addr* is also called the process-identifier or processid because the workspace pointer is used by the processor hardware to identify processes.

POSTFIX **IFORTH**

(-- a-addr)

a-addr is the address of **POSTFIX** . **POSTFIX** contains a flag that when false causes : to determine the name for the new definition from a address/length pair left on the stack.

POSTPONE **C** **CORE**

Compilation: ("name" --)

Parse *name* delimited by a space, ignoring leading delimiters. Append the compilation semantics of *name* to the current definition. An ambiguous condition exists if *name* is not found.

Execution: (--)

Perform the compilation semantics of *name*.

PRECISION **FLOAT**

(-- u)

PRECISION returns the number of decimal places (digits to the right of the radix point) displayed by **E.** , **FE.** or **FS.** .

See also: **SET-PRECISION**

PREVIOUS **SEARCH EXT**

(--)

Transform the search order consisting of *wid1*, ... *widn*-1, *widn* (where *widn* is searched first) into *wid1*, ... *widn*-1. In a standard system an ambiguous condition exists if the search order was empty before **PREVIOUS** was executed. In iForth one can never empty the search order completely.

PRIVATE **IFORTH**

(--)

Marks the last definition in the current wordlist as a candidate for **DEPRIVE** . The word will become inaccessible when **DEPRIVE** is executed.

See also: **DEPRIVE HIDE PRIVATES 'PRIVATES**

PRIVATES **IFORTH**

(--)

Turn the last definition in the current wordlist into a sentinel for **DEPRIVE** . All yet to be defined words in the current wordlist followed by **PRIVATE** are made invisible when **DEPRIVE** is executed. For large source files in multi-programmer projects, this is far more convenient than **HIDE** name. The idea behind **PRIVATES** is to prevent name space clutter but encourage factoring. All this without burdening the programmer with the requirement of writing a lot of (external) documentation. **PRIVATES** ... **DEPRIVE** can be nested.

See also: **DEPRIVE HIDE 'PRIVATES**

QUERY **CORE EXT**

(--)

Receive input into the text input buffer whose address is given by **TIB** , replacing its previous contents if any, and make the result the current input stream.

QUIT **CORE**

*(--) (R: i*x --)*

Empty the return stack, enter interpretation state, accept new input from the current input device, and begin text interpretation. Do not display a message. When the input stream has been exhausted, all processing has been completed and no ambiguous condition exists, the implementation-defined system prompt is displayed. The interpreter then waits for further input.

R! **IFORTH**

(n --)

Overwrites the value on top of the return stack with *n*. Equivalent to **R> DROP >R** .

See also: **S!**

R+! **IFORTH**

(n --)

Adds *n* to the value on top of the return stack. Equivalent to **R>** + **>R** .

See also: **S+!**

R/O "r-o" **FILE**

(-- x)

x is the implementation-dependent value for selecting the "read only" file access method.

See also: **CREATE-FILE OPEN-FILE**

R/W "r-w" **FILE**

(-- x)

x is the implementation-dependent value for selecting the "read/write" file access method.

See also: **CREATE-FILE** **OPEN-FILE**

R> "r-from" **C** **CORE**

(-- x) (R: x --)

Move *x* from the return stack to the data stack.

R@ "r-fetch" **C** **CORE**

(-- x) (R: x -- x)

Copy *x* from the return stack to the data stack.

READ-FILE **FILE**

(c-addr u1 fileid -- u2 ior)

Read *u1* consecutive characters to *c-addr* from the current position of the file identified by *fileid*. If *u1* characters are read without error, *ior* is zero and *u2* is equal to *u1*. If the end of the file is reached before *u1* characters are read, *ior* is zero and *u2* is the number of characters actually read. If the operation is initiated when the value returned by **FILE-POSITION** is equal to the value returned by **FILE-SIZE** for the file identified by *fileid*, *ior* is zero and *u2* is zero. If an error occurs, *ior* is the implementation-defined **I/O** result code and *u2* is the number of characters transferred to *c-addr* without error. If the operation is initiated when the value returned by **FILE-POSITION** is greater than the value returned by **FILE-SIZE** for the file identified by *fileid* the operation will read 0 bytes. If the requested operation attempts to read portions of the file not written then garbage will be read. Note that other ANS systems may react different on these conditions. At the conclusion of the operation, **FILE-POSITION** returns a value past the characters consumed by the operation.

READ-LINE **FILE**

(c-addr u1 fileid -- u2 flag ior)

Read the next line from the file specified by *fileid* into memory at the address *c-addr*. At most *u1* characters are read. The implementation-dependent line terminator, if any, may be read into memory at the end of the line, but its length is not included in the count *u2*. The line buffer provided by *c-addr* should be at least *u1*+2 characters long. If the operation succeeded, *flag* is true and *ior* is zero. If a line terminator was received before *u1* characters were read, then *u2* is the number of characters, not including the line terminator, actually read (0 **<=** *u2* **<=** *u1*). When *u1* = *u2* the line terminator has yet to be reached. If the operation is initiated when the value returned by **FILE-POSITION** is equal to the value returned by **FILE-SIZE** for the file identified by *fileid*, *flag* is false, *ior* is zero, and *u2* is zero. If *ior* is non-zero, an error occurred during the operation and *ior* is the implementation-defined **I/O** result code. If the operation is initiated when the value returned by **FILE-POSITION** is greater than the value returned by **FILE-SIZE** for the file identified by *fileid* the operation will read 0 bytes. If the requested operation attempts to read portions of the file not written then garbage will be read. Note that other ANS systems may react different on these conditions. At the conclusion of the operation, **FILE-POSITION** returns a value past the characters consumed by the operation. The iForth server will translate any OS-dependent end of line sequence to an internal format of a single $0A (line feed). See also **$CR** . Note that the Standard requires us to disclose this fact. You will need the end of line sequence when using **READ-FILE** and **WRITE-FILE** for text. We do not recommend this practice.

RECURSE **C** **CORE**

Compilation: (--)

Append the execution semantics of the current definition to the current definition.

See also: **RECURSIVE**

RECURSIVE **C** **IFORTH**

(--)

Makes the current definition available to the system. Normally this happens automatically when executing ; . When the current word is available to the system a reference to its name produces a recursive call to the definition. If **RECURSIVE** is not executed a reference to that name will result in calling a previous definition with the same name, if one exists.

See also: *;*

REDUCE.2PI **IFORTH**

(r1 -- r2)

r2 is calculated by taking the value of *r1* modulo 2ã. *r2* lies in the range { -ã .. ã }. This operation is not trivial.

REDUCE.PI **IFORTH**

(r1 -- r2)

r2 is calculated by taking the value of *r1* modulo ã. *r2* lies in the range { -ã/2 .. ã/2 }. This operation is not trivial.

REFILL **CORE EXT**, **BLOCK EXT**, **FILE EXT**

(-- flag)

Attempt to fill the current input stream, returning a true *flag* if successful. The action depends on the source of the current input stream. If the input-stream source is a string from **EVALUATE** , **REFILL** returns false and performs no other action. Otherwise, **REFILL** attempts to receive input into the text-input buffer whose address is given by **TIB** , making the result the current input stream and returning a true *flag* if successful. Receipt of a line containing no characters is considered successful. A false *flag* is returned only when there is no input available from the current input-stream source. If the input-stream source is a block, **REFILL** makes the next block the current input stream by adding one to the value of **BLK** and setting **>IN** to zero. True is returned if the new value of **BLK** is a valid block number, false otherwise. If the input-stream source is a text file, **REFILL** attempts to read the next line from the text-input file, making the result the current input stream and returning true if the read succeeded, and returning false otherwise.

REGISTER **IFORTH**

(u "name" --)

Parse *name* delimited by a space, ignoring leading delimiters. Create a temporary dictionary entry for *name* with the execution semantics defined below. The high speed variable with an *offset* of *u* cells into the high speed variable area is associated with *name*. No more than sixteen of such variables can be used. Note that an important drawback of using these high speed variables is that they are global, and different names can be made to refer to the same physical location. The iForth system and utilities never use registers. *name* is referred to as a "register".

Execution: ("name" -- x)

Place *x* on the stack. The value of *x* is unspecified until the phrase *x* **TO** *name* is executed, causing *x* to be associated with *name*.

See also: to-concept (?)

RENAME-FILE **FILE EXT**

(c-addr1 u1 c-addr2 u2 -- ior)

Rename the file named by the first character string specified by *c-addr1 u1* to the name in the second character string. *ior* is the implementation-defined **I/O** result code.

REPEAT **C** **CORE**

Compilation: (orig dest --)

Append the execution semantics given below to the current definition, resolving the backward reference *dest*. Resolve the forward reference *orig* using the location following the appended execution semantics.

Execution: (--)

Continue execution at the location given by *dest*.

See also: **BEGIN WHILE REPEATED**

REPEAT, "repeat-comma" **IFORTH-ASSEMBLER**

Assembling: (addr1 addr2 --)

Assemble a backward branch to the memory location *addr1* as generated by BEGIN, . Resolve the conditional forward branch at *addr2* as generated by WHILE, .

Execution: (--)

Take the branch.

See also: **BEGIN, WHILE,**

REPEATED **C** **IFORTH**

*Assembling: (n*orgs dest --)*

Append the execution semantics given below to the current definition, resolving the backward reference *dest*. Resolve the *n* forward referenced *orgs* using the location following the appended execution semantics. **REPEATED** aborts with an error message if **SECURE** is **OFF** . This word allows the standard ANS phrase

```
BEGIN ... WHILE ... WHILE ... WHILE ... REPEAT THEN THEN
```

to be written as

```
BEGIN ... WHILE ... WHILE ... WHILE ... REPEATED .
```

Execution: (--)

Continue execution at the location given by *dest*.

See also: **BEGIN WHILE REPEAT**

REPOSITION-FILE **FILE**

(ud fileid -- ior)

Reposition the file identified by *fileid* to *ud*. *ior* is the implementation-defined **I/O** result code. If the file is positioned outside the file boundaries any read there will return a count of 0 bytes. If the file is (also) open for writing, it is made larger as soon as a write operation is attempted. The contents of the file between the previous end of file and the file pointer before the write are garbage. Note that other ANS systems may react different to this condition. At the conclusion of the operation, **FILE-POSITION** returns the value *ud*.

REPRESENT **FLOAT**

(c-addr u -- n flag1 flag2) (F: r --)

Place a character-string representation of the significand of the floating-point number *r* at *c-addr*, return the decimal-base exponent as *n*, the sign as *flag1* and 'valid result' as *flag2*. The character string consists of the *u* most signifant digits of the significand represented as a decimal fraction with the implied decimal point to the left of the first digit, and the first digit zero only if all digits are zero. The significand is rounded to *u* digits following the round to nearest rule; *n* is adjusted, if necessary, to correspond to the rounded magnitude of the significand. If *flag2* is true then *r* was in the implementation-defined range of floating-point numbers. If *flag1* is true then *r* is negative. The ANS Forth standard specifies an ambiguous condition if the value of **BASE** is not decimal ten. When *flag2* is false, *n* and *flag1* are implementation defined, but the string at *c-addr* is printable and conveys some information about *r*.

RESIZE **MEMORY**

(a-addr1 u -- a-addr2 ior)

Change the allocation of the contiguous data space starting at the address *a-addr1* allocated by **ALLOCATE** or **RESIZE** to *u* address units. *u* may be either larger or smaller than the current size of the region. The address of the next available data space location is unaffected by this operation. If the operation succeeds, *a-addr2* is the aligned starting address of *u* address units of allocated memory and *ior* is zero. *a-addr2* may or may not be the same as *a-addr1*. If they are not the same, the values contained in the region at *a-addr1* are copied to *a-addr2*, up to the minimum size of either of the two regions. If they are the same, the values contained in the region are preserved to the minimum of *u* or the original size. If *a-addr2* is not the same as *a-addr1*, the region of memory at *a-addr1* is returned to the system according to the operation of **FREE** . If the operation fails, *a-addr2* does not represent a valid address and *ior* is the implementation-defined **I/O** result code.

See also: **ALLOCATE AVAILABLE FREE INIT-MEM**

RESIZE-FILE **FILE**

(ud fileid -- ior)

Set the size of the file identified by *fileid* to *ud*. *ior* is the implementation-defined **I/O** result code. If the resultant file is larger than the file before the operation, the portion of the file added as a result of the operation may not have been written. At the conclusion of the operation, **FILE-SIZE** returns the value *ud* and **FILE-POSITION** returns an unspecified value. Note that this word performs an operation that may not be supported directly by all operating systems.

See also: **READ-FILE READ-LINE**

RESIZE-MEMORY **IFORTH**

(u --)

Resizes the total amount of dynamic memory available to iForth to *u* bytes. This word should not be executed when dynamic memory is in use already. iForth handles all memory allocations by itself, using a single contiguous block requested from the **OS** at boot-up. The default size of this block is 100 KBytes.

See also: **AVAILABLE INIT-MEM**

RESTORE-FFORMAT **IFORTH**

(x1 x2 x3 x4 x5 x6 --)

Place the values *x1* ... x6 in the 6 variables that control the appearance of a floating-point number when printing that number. The values *x1* ... x6 should be in the same order as the routine **SAVE-FFORMAT** has placed them on the stack.

See also: **FECHAR FDEC FELEN FESIGN FMSIGN SAVE-FFORMAT**

RESTORE-INPUT **FILE EXT**

(x1 x2 ... xn n -- flag)

Attempt to restore the current input stream to the position described by *x1* ... *xn*. *flag* is true if the input stream cannot be positioned to the place described by *x1* ... *xn*. For iForth *n* is five. The ANS Forth standard does not require to specify exactly what *x1* ... *xn* mean, nor does it prescribe the number of parameters. It follows that a standard program may not use the fact that iForth returns five parameters.

See also: **SAVE-INPUT**

RETURN-STACK-CELLS **IFORTH**

RETURN-STACK-CELLS is an environment query.

See also: **ENVIRONMENT?**

RETURNCODE **IFORTH**

(-- a-addr)

The word at *a-addr* contains an implementation-defined result code, valid directly after executing the **SYSTEM** command.

REVISION **D** **IFORTH**

("name" ""explanation"" --)

Parse *name* delimited with a space, ignoring leading delimiters. If *name* exists in the current search order, execute that word (in effect forgetting it). Create a dictionary entry for *name* with the execution semantics defined below. Parse explanation surrounded by " (double

quote). Add the resulting text to the definition of *name*. Display the text 'Creating: explanation'. It is advised that the contents of explanation is the name of the current software package and a version number. *name* is referred to as a "revision control word".

Execution: ("name" --)

Display the text 'Removing: explanation'. Remove *name* and all definitions that were defined after *name*.

ROL **IFORTH**

(x1 x2 -- x3)

Rotate *x1* over *x2* bits to the left, giving the result *x3*. Bits shifted out at the left end will be inserted at the right end.

ROLL **CORE EXT**

(x(u) x(u-1) ... x(0) u -- x(u-1) ... x(0) x(u))

Remove *u*. Rotate *u*+1 items on the top of the stack. An ambiguous condition exists if there are less than *u*+2 entries on the stack before **ROLL** is executed.

ROOM **IFORTH**

(-- addr)

A user variable that returns the *offset* above **PAD** that is guaranteed safe for use by a user buffer. In an ANS Forth **ROOM @ PAD** + returns the address of the first byte after the **PAD** (**PAD** /PAD +). The idea is that every program that needs a floating multi-programming resistant buffer manipulates **ROOM** as in the following example:

```
\ Concatenate two strings at PAD2 , using PAD3 .
\ Note: Both strings can be (somewhere) at pad2, no problem.

: PAD$     CREATE  ROOM @ , $100 ROOM +!
           DOES>   @ PAD + ;

           PAD$ pad2  PRIVATE
           PAD$ pad3  PRIVATE
```

See also: **PAD**

ROR **IFORTH**

(x1 x2 -- x3)

Rotate *x1* over *x2* bits to the right, giving the result *x3*. Bits shifted out at the right end will be inserted at the left end.

ROT "rote" **CORE**

(x1 x2 x3 -- x2 x3 x1)

Rotate the top three stack entries.

RP! "r-p-store" **C** **IFORTH**

(a-addr --)

Set the return stack pointer to *a-addr*.

RP0 "r-p-zero" **IFORTH**

(-- a-addr)

a-addr is the address of **RP0** . **RP0** contains the pointer to the base of the return stack. The phrase **RP0 @ RP!** removes all elements from the return stack.

RP@ "r-p-fetch" **IFORTH**

(-- a-addr)

Get the return stack pointer. Note that iForth uses stacks that grow towards lower addresses. The top element of the return stack is at **RP@** , however you should never access stacks in this way.

RSHIFT **CORE**

(x1 n -- x2)

Perform a logical shift of *n* bit-places on *x1*, giving *x2*. Shift toward the least significant bits. Put zero into the places "uncovered" by the shift.

RUN **IFORTH**

(xt -- xt')

Convert execution token to something that a **FORTH-PROCESS** can understand.

S **IFORTH**

(-- x) (S: x -- x)

Copy *x* from the system stack to the data stack.

S! **IFORTH**

(n --)

Overwrites the value on top of the system stack with *n*. Equivalent to **S> DROP >S** .

See also: **R!**

S” "s-quote" **CORE**, **FILE**

Compilation: ("ccc<">" --)

Parse characters *ccc* delimited by " (double quote). Append the execution semantics given below to the current definition.

Execution: (-- c-addr u)

Return *c-addr* and *u* that describe a string consisting of the characters *ccc*.

Interpretation:("ccc<">" -- c-addr u)

Parse characters *ccc* delimited by " (double quote). Store the resulting string at a temporary location described by *c-addr* and *u*. The maximum length of the temporary buffer is 255 characters but other implementations of the standard may limit this to 80 characters. iForth allows for the creation of 4 such strings before new strings start to overwrite the string buffer. Note that other implementations of the standard may limit this to just one string. A standard program may not alter the returned string. For systems that do not implement the FILE wordset **S"** may be a compile only (**C**) word.

S+! **IFORTH**

(n --)

Adds *n* to the value on top of the system stack. Equivalent to **S>** + **>S** .

See also: **R+!**

S> "from-s" **IFORTH**

(-- x) (S: x --)

Move *x* from the system stack to the data stack.

S>D "s-to-d" **DOUBLE**

(n -- d)

d is the equivalent of *n*.

S>F "s-to-f" **IFORTH**

(x --) (F: -- r)

r is the floating-point equivalent of *x*. An ambiguous situation exists if *x* cannot be precisely represented as a floating-point number.

SAVE-BUFFERS **BLOCK**, **FILE**

(--)

Transfer the contents of each **UPDATE** -d block buffer to mass storage. Mark all buffers as unmodified. An **UPDATE** -d block that came from a file must be transferred back to the same file when the block buffer is needed for another block.

See also: **FILE-BLOCK FILE-BUFFER**

SAVE-FFORMAT **IFORTH**

(-- x1 x2 x3 x4 x5 x6)

Stack the values of the 6 variables that control the appearance of a floating-point number when printing that number. This appearance is restored by calling **RESTORE-FFORMAT** with the values *x1* ... x6 on the stack.

See also: **FECHAR FDEC FELEN FESIGN FMSIGN RESTORE-FFORMAT**

SAVE-INPUT **FILE EXT**

(-- x1 x2 ... xn n)

x1 ... *xn* describe the current position in the input stream or text input file for later use by **RESTORE-INPUT** . The common trick: { **>IN @** ... **>IN** ! } can only be used within the current input line and may not be sufficient for other ANS Forth systems. For iForth *n* is five. The ANS Forth standard does not require to specify exactly what *x1* ... *xn* mean, nor does it prescribe the number of parameters. It follows that a standard program may not use the fact that iForth returns five parameters.

See also: **RESTORE-INPUT**

SAVE-SYSTEM **IFORTH**

("fn" --)

Save the system to image file *fn*. The MS-DOS iForth needs the batch file MAKESYS.BAT to turn this image into an executable. For any other **OS** you must make the new image findable to the "**C**" server by copying it to the ./bin directory, or by setting the environment variable IFORTHBIN .

SCAN **IFORTH**

(c-addr1 u1 delimiter -- c-addr2 u2)

Scan the string identified by *c-addr1 u1* for *delimiter*. If this character is found, leave its address in *c-addr2* and the count remaining in *u2*, otherwise *c-addr2* points after the string and *u2* is 0. When the *delimiter* is not a <tab>, <lf> or <cr>, all three of these characters are considered equivalent to a space. This means **BL SCAN** is a special case.

See also: **SKIP**

SCAN-$ "scan-string" **IFORTH**

(c-addr u --)

Read a blank delimited word from the input stream and compare it to the string described by *c-addr u*. If these are not equal, keep reading until the input stream is exhausted or until a matching word is found. In this case the input stream points after the space at the end of the matching word. **SCAN-$** invokes **REFILL** automatically. This word treats the characters <cr>, <lf> and <tab> as equivalent to a blank.

See also: **SCAN-ANY SCAN-CHAR**

SCAN-ANY **IFORTH**

(-- xt) or (-- 0) or (-- -1)

Read a blank delimited word from the input stream and look it up using the current search order. Returns -1 if the input stream is exhausted. If the word cannot be found, return 0, otherwise return the execution token. This word executes **REFILL** internally. This word treats the characters <cr>, <lf> and <tab> as equivalent to a blank.

See also: **WORD FIND REFILL**

SCAN-CHAR **IFORTH**

(delimiter --)

Read characters of the input stream until the first one not equal to *delimiter*, or until the input stream is exhausted. If found, the input stream points just after the *delimiter*. (Note that this is different from how **SCAN** works). **REFILL** is invoked automatically. This word treats the characters BL, <cr>, <lf> and <tab> in the way of **SCAN** .

See also: **SCAN-ANY SCAN-$**

SCR "s-c-r" **BLOCK EXT**

(-- a-addr)

a-addr is the address of **SCR** . **SCR** contains the block number of the block most recently LISTed. (**SCR** stands for screen).

SDEPTH "s-depth" **IFORTH**

(-- +n) (S: --)

+n is the number of one cell values contained on the system stack.

SEARCH **STRING**

(c-addr1 u1 c-addr2 u2 -- c-addr3 u3 flag)

Search the string specified by *c-addr1* and *u1* for the string specified by *c-addr2* and *u2*. If *flag* is true, a match was found at *c-addr3* with *u3* characters remaining. If *flag* is false there was no match and *c-addr3* is *c-addr1* and *u3* is *u1*.

SEARCH-ENV$ **IFORTH**

(c-addr1 u1 -- c-addr2 u2 flag)

Search the string specified by the tag *c-addr1* and *u1* in the **OS** environment string space. When the tag is found, return the accompanying data identified by *c-addr2* and *u2*. If *flag* is true, the tag was found and *c-addr2* with *u2* defines a valid string. If *flag* is false the tag was not found and the null string is returned.

See also: **'PARAM #PARAMS WHERE-I-AM**

SEARCH-ORDER **IFORTH**

SEARCH-ORDER is an environment query.

See also: **ENVIRONMENT?**

SEARCH-ORDER-EXT **IFORTH**

SEARCH-ORDER-EXT is an environment query.

See also: **ENVIRONMENT?**

SEARCH-WORDLIST **SEARCH**

(c-addr u wid -- 0) if not found, *(c-addr u wid -- xt 1)* if immediate

found or (*c-addr u wid -- xt* -1) if non-immediate found Find the Forth word identified by the string *c-addr u* in the word list identified by *wid*. If the word is not found, return zero. If the word is found, return its execution token *xt* and 1 if the word is immediate, -1 otherwise.

SECURE **IFORTH**

(-- a-addr)

a-addr is the address of **SECURE** . **SECURE** contains a flag that determines whether flow control words are checked for correct nesting. When this variable is set to true, it is an error not to nest flow control words properly.

SEE **TOOLKIT EXT**

("name" --)

Display a human-readable representation of the named word's definition. The particular form of the display is implementation-dependent. Since iForth compiles definitions to native processor assembler code, this word cannot do much more than provide a disassembly of the word. Therefore not much of the original structure is left. Where possible, addresses in the listing are changed to alphanumeric labels. Code generated by some macros is recognized and the original label *name* is listed instead of the code fragment. Due to the size of this word this word is available only after loading the file see.frt. To do this, just type **INCLUDE** see.frt at the Forth prompt.

SEEK-FILE **IFORTH**

(doffs begin|current|end fid -- dpos ior)

Seek the file identified by *fid* to the (double-precision) position *doffs*, relative to file *begin* (-1), *current* position (0), or file *end* (1). The actual file pointer position, relative to file start, is returned as the double precision number *dpos*. *ior* is the OS-dependent IO error code.

See also: **REPOSITION-FILE**

SEMIT "s-emit" **IFORTH**

(char --)

If *char* is a printable ASCII character in the range { 32 .. 126 }, use **EMIT** to print this character. Otherwise use **EMIT** to display a '.' (full stop).

See also: **EMIT**

SERVER **IFORTH**

SERVER is an environment query.

See also: **ENVIRONMENT?**

SET-CURRENT **SEARCH**

(wid --)

Set the compilation word list to the word list identified by *wid*.

SET-ORDER **SEARCH**

(wid1 ... widn n --)

Set the search order to the word lists identified by *wid1 ... widn*. Subsequently, word list *widn* will be searched first, followed by word list *widn*-1, and so on, with word list *wid1* searched last. If *n* is zero, empty the search order. If *n* is minus one, set the search order to the implementation-defined minimum search order. The minimum search order must include the ability to interpret the words **FORTH-WORDLIST** and **SET-ORDER** .

See also: **GET-ORDER**

SET-PRECISION **FLOAT**

(u --)

Set the number of decimal places (digits to the right of the radix point) displayed by **E.** and **F.** .

See also: **PRECISION**

SETPRIORITY **IFORTH**

(0|1 -- oldpri)

Set the number of the queue that the current process will be stored in when it becomes inactive. Return the previous queue number. 0 is the high-priority queue, 1 is the low-priority queue. Note that it is bad programming practice to change the priority of a process other than at the start of a sub-process as created in PAR ... ENDPAR .

SF! "s-f-store" **IFORTH**

(a-addr1 --) (F: r --)

Store the floating-point number *r* as a 32-bit IEEE single precision number at *a-addr1*. Note that in other implementations of the ANS standard *r* may have more digits of precision or may be too large for representation as a 32-bit IEEE number, so rounding or overflow might occur.

See also: **SF! DF! XF!**

SF!+ "s-f-store-plus" **IFORTH**

(a-addr1 -- a-addr2) (F: r --)

Store the floating-point number *r* as a 32-bit IEEE single precision number at *a-addr1*, and leave the incremented pointer as *a-addr2*. Note that in other implementations of the ANS standard *r* may have more digits of precision or may be too large for representation as a 32-bit IEEE number, so rounding or overflow might occur.

See also: **SF! DF!+ XF!+**

SF+! "s-f-plus-store" **IFORTH**

(F: r --) (a-addr --)

Add the IEEE single *r* to the single at *a-addr*.

SF+!+ "s-f-plus-store-plus" **IFORTH**

(F: r --) (a-addr1 -- a-addr2)

Add the IEEE single *r* to the single at *a-addr1* and leave the incremented pointer as *a-addr2*.

See also: **SF+!**

SF, "s-f-comma" **IFORTH**

(--) (F: r --)

Reserve the size of a single precision IEEE floating-point number in the data-space and store *r* in that space. An ambiguous condition exists if the address of the next available data space location is not aligned.

See also: **F, DF,**

SF-! "s-f-minus-store" **IFORTH**

(F: r --) (a-addr --)

Subtracts the IEEE single 32-bit number *r* from the single at *a-addr*.

SF@ "s-f-fetch" **FLOAT EXT**

(a-addr --) (F: -- r)

Fetch the 32-bit IEEE single precision number stored at *a-addr* to the floating-point stack as *r* in the internal representation. If the IEEE single precision significand has more precision than the internal representation it will be rounded to the internal representation using the "round to nearest" rule.

SF@+ "s-f-fetch-plus" **IFORTH**

(a-addr1 -- a-addr2) (F: -- r)

Fetch the 32-bit IEEE number stored at *a-addr1* to the floating-point stack as *r* in the internal representation. Also leave the incremented pointer as *a-addr2*.

See also: **SF@**

SFALIGN **FLOAT**

(--)

If the address of the next available data space location is not aligned, reserve enough space to align it to a location which is suitable to hold a IEEE single precision floating point number.

SFALIGNED **FLOAT**

(addr -- a-addr)

a-addr is the first aligned address greater than or equal to *addr* and which is suitable to reference a IEEE single precision floating point number.

SFLOAT+ "s-float-plus" **FLOAT**

(a-addr1 -- a-addr2)

Add the size of an IEEE single precision floating point number, specified in address units, to *a-addr1*, giving *a-addr2*.

SFLOAT- "s-float-min" **IFORTH**

(a-addr1 -- a-addr2)

Subtract the size of an IEEE single precision floating point number, specified in address units, from *a-addr1*, giving *a-addr2*.

SFLOATS "s-floats" **FLOAT**

(n1 -- n2)

n2 is the size, in address units, of *n1* IEEE single precision floating point numbers.

SFLOAT[] **IFORTH**

(addr1 index -- addr2)

Equivalent to **SFLOATS** + .

See also: **[]SFLOAT**

SFVARIABLE "s-f-variable" **D** **FLOAT**

("name" --)

Parse *name* delimited by a space, ignoring leading delimiters. Create a dictionary entry for *name* with the execution semantics defined below. Reserve 1 **SFLOATS** address units of data space at an aligned address. *name* is referred to as an "s-f-variable."

Execution: ("name" -- a-addr)

a-addr is the address of the data space reserved by **SFVARIABLE** when it created *name*. The application is responsible for initializing the contents of the reserved space.

SIGN **CORE**

(n --)

If *n* is negative, add a minus sign to the beginning of the pictured numeric output string. Typically used between **<#** and **#>** .

SIZEOF "size-of" **IFORTH**

Usage: **SIZEOF** name

(-- u)

u is the size in bytes of the contents of the variable *name*. *name* must be created by **VALUE** or any other word that implements the to-concept.

SKIP **IFORTH**

(c-addr1 u1 delimiter -- c-addr2 u2)

Skip all leading *delimiter*s in the string. Leave the address of the first non-*delimiter* character in *c-addr2* and the count remaining in *u2*. If only *delimiter*s are found, *c-addr2* points directly after the string and *u2* is 0. When the *delimiter* is not a <tab>, <lf> or <cr>, all three of these characters are considered equivalent to a space. This means **BL SKIP** is a special case that skips spaces, tabs, <lf>'s and <cr>'s.

See also: **SCAN**

SLITERAL **STRING**

Interpretation:undefined.

Compilation:(c-addr1 u --)

Append the execution semantics given below to the current definition.

(-- *c-addr2 u*)

Return *c-addr2 u* describing a string consisting of the characters specified by *c-addr1 u* during compilation. A Standard program shall not alter the string. Example: : foo [**S"** bar"] **SLITERAL TYPE** ; foo<cr> bar ok

See also: **LITERAL**

SM/REM "s-m-slash-remainder" **CORE**

(d1 n1 -- n2 n3)

Divide *d1* by *n1*, giving the symmetric quotient *n3* and the remainder *n2*. Input and output stack arguments are signed. An ambiguous condition exists if *n1* is zero or if the quotient lies outside the range of a single-cell signed integer.

SOURCE **IFORTH**

(-- c-addr u)

Return the location and number of valid characters in the input buffer. Example: when inputting from the terminal **SOURCE** returns **TIB #TIB @** .

SOURCE-ID **FILE**

(-- 0) or (-- -1) or (-- fileid)

Identifies the source of the non-block input stream (*i.e.*, when **BLK** is zero) as follows:

```
SOURCE-ID       Input stream source
-----------     ------------------
0               Keyboard
-1              String (via EVALUATE )
fileid          Text file "fileid"
```

SP! "s-p-store" **IFORTH**

(a-addr --)

Set the data stack pointer to *a-addr*.

SP0 "s-p-zero" **IFORTH**

(-- a-addr)

a-addr is the address of **SP0** . **SP0** contains the pointer to the base of the data stack. The phrase **SP0 @ SP!** removes all elements from the data stack.

SP@ "s-p-fetch" **IFORTH**

(-- a-addr)

Get the data stack pointer. Note that iForth uses stacks that grow towards lower addresses. The top element of the data stack is at **SP@** , however you should never access stacks in this way.

SPACE **CORE**

(--)

Display one space.

SPACES **CORE**

(n --)

If *n* is greater than zero, display *n* spaces.

SPAN **CORE EXT**

(-- a-addr)

a-addr is the address of **SPAN** . **SPAN** contains the count of characters stored by the last execution of **EXPECT** . Note: This word is obsolescent and is included as a concession to existing implementations.

SPICK "s-pick" **IFORTH**

(u -- x(u)) (S: x(u) ... x(1) x(0) -- x(u) ... x(1) x(0))

Remove *u*. Copy the *u*th item of the system stack to the data stack. An ambiguous condition exists if there are less than *u*+2 items on the system stack.

SSP! "s-s-p-store" **IFORTH**

(a-addr --)

Set the system stack pointer to *a-addr*.

SSP0 "s-s-p-zero" **IFORTH**

(-- a-addr)

a-addr is the address of **SSP0** . **SSP0** contains the pointer to the base of the system stack. The phrase **SSP0 @ SSP!** removes all elements from the system stack.

SSP@ "s-s-p-fetch" **IFORTH**

(-- a-addr)

Get the system stack pointer. Note that iForth uses stacks that grow towards lower addresses. The top element of the system stack is at **SSP@** , however you should never access stacks in this way.

SSPPTR "s-s-p-pointer" **IFORTH**

(-- a-addr)

Get the memory address where the system stack pointer is kept. Note that iForth uses stacks that grow towards lower addresses. The top element of the system stack is at **SSPPTR @** (or **SSP@**) , however you should never access stacks in this way.

See also: **SSP@**

STACK-CELLS "stack-cells" **IFORTH**

STACK-CELLS is an environment query.

See also: **ENVIRONMENT?**

STACK-DUMP "stack-dump" **IFORTH**

(rp --)

Symbolic dump of the return stack, starting from rp, using a special format. May be useful for debugging.

STATE **CORE**, **TOOLKIT EXT**

(-- a-addr)

a-addr is the address of **STATE** . **STATE** contains the compilation state flag. **STATE** **@** is true when in compilation state, false otherwise. The true value in **STATE** is non-zero, but is otherwise implementation-defined. Only the following standard words alter the value in **STATE** : : (colon), ; (semicolon), **ABORT** , **QUIT** , **:NONAME** , [(left-bracket),] (right-bracket) and **;CODE** . Note: A Standard Program may not directly alter the contents of **STATE** .

See also: : ; **;CODE** **ABORT** **QUIT** [ ] **:NONAME**

STOP **IFORTH**

(--)

Stops a FORTH-PROCESS. It stops to run, but process memory stays allocated.

STRING **IFORTH**

STRING is an environment query.

See also: **ENVIRONMENT?**

STRING-EXT **IFORTH**

STRING-EXT is an environment query.

See also: **ENVIRONMENT?**

STYPE "s-type" **IFORTH**

(addr n --)

Display the character string specified by *addr n*. Any character in the string that is not inside the range { 32 .. 126 } is printed as a '.' (full stop).

See also: **TYPE**

SUBROUTINE **D** **IFORTH**

("name" -- sys)

Like CODE, but doesn't pop the machine return address to put it on the Forth return stack.

See also: **CODE** **END-CODE**

SWAP **CORE**

(x1 x2 -- x2 x1)

Exchange the top two stack items.

SYSCALL **IFORTH**

(n{u} n #service -- result error)*

Calls the statically linked "**C**" routine in the server which is located in jumptable slot *#service*. There are *n* unsigned arguments *u*. Prevents the **OS** from disrupting correct Forth operation by saving and restoring all Forth registers. The **C** (machine) and iForth (data) stacks are swapped for the duration of the call. This is essential as some **C** routines use enormous amounts of stack space. **C** sees the arguments in the same order as Forth (top Forth == last *u* in **C**'s argument list). The pre-compiled Forth kernel assumes **C** returns function *result*s in the EAX register. This is true for the Borland and gcc compilers. The *error* is the contents of the **C** errno variable, it may be relevant depending on the specific service being called.

See also: **FOREIGN**

SYSTEM **IFORTH**

(addr n --)

Send the character string specified by *addr n* to the host machine so that the string can be interpreted by a command interpreter running on the host. If *n* is 0 a system dependent command interpreter is called. By terminating the command interpreter, control is returned to iForth in the exact state in which it was left. In all currently supported OSes this is done by issuing the **EXIT** command on the shell command line.

SYSTEM-STACK-CELLS
IFORTH

SYSTEM-STACK-CELLS is an environment query.

See also: **ENVIRONMENT?**

T **IFORTH**

(-- x) (S: x y -- x y)

Copy the element that is second from the top of the system stack to the data stack.

TERMINATE **IFORTH**

(n --)

Disconnect the server from the processor. Remove the server program on the host computer and let the server return the implementation defined error code *n* to the operating system.

See also: **BYE**

TFORTH? **IFORTH**

(-- bool)

TRUE if this is tForth.

See also: **LINUX? MS-DOS? WINNT? XLINUX?**

THEN **C** **CORE**

Compilation: (orig --)

Resolve the forward reference *orig* using the location of the execution semantics.

Execution: (--)

Continue execution.

See also: **ELSE IF**

THEN, **IFORTH-ASSEMBLER**

Assembling: (addr --)

Resolve the forward branch as assembled by IF, or ELSE, . The forward branch is located at the memory location *addr*.

Execution: (--)

Do nothing.

THROW **ERROR**

*(k*x 0 -- k*x) not thrown or (k*x n -- i*x n) thrown.*

If the top of the stack is zero, **THROW** discards it. Otherwise, it pops the topmost error interception frame (see **CATCH**) from the return stack, along with everything on the return stack above that error frame. **THROW** then adjusts the stack depth so that the depth is the same as the depth saved in the error frame (*i* is the same number as the *i* in the input arguments to the corresponding **CATCH**), puts *n* on top of the stack, and transfers control to a point just after the **CATCH** that installed that error frame. If there is no error interception frame on the return stack, **THROW** performs the function of **ABORT** .

THRU **BLOCK EXT**

(u1 u2 --)

Sequentially **LOAD** the mass storage blocks numbered *u1* through *u2*.

TIB "t-i-b" **CORE**

(-- c-addr)

c-addr is the address of the text input buffer. Note: A Standard Program may not directly alter the contents of the text input buffer.

TID **IFORTH**

(-- n)

Returns a number that is related to the processor the currently running binary is compiled for. i3FE5M__.EXE returns 386.

TIME **IFORTH**

(-- +n1 +n2 +n3)

Return the current time. +*n1* is the second (0-59), +*n2* is the minute (0-59), +*n3* is the hour (0-23).

TIME&DATE "time-and-date" **IFORTH**

(-- +n1 +n2 +n3 +n4 +n5 +n6)

Return the current time and date. +*n1* is the second (0-59), +*n2* is the minute (0-59), +*n3* is the hour (0-23), +*n4* is the day (1-31), +*n5* is the month (1-12), and +*n6* is the year (*e*.g., 1991).

TO **CORE EXT**, **LOCAL**

Execution: (x --)

Store *x* in *name*. An ambiguous condition exists if *name* was not defined by **VALUE** or **(LOCAL)** .

See also: **VALUE (LOCAL)**

TOOLS **IFORTH**

TOOLS is an environment query.

See also: **ENVIRONMENT?**

TOOLS-EXT **IFORTH**

TOOLS-EXT is an environment query.

See also: **ENVIRONMENT?**

TRUE **CORE EXT**

(-- true)

Return a true flag.

TSCREEN> "text-screen-from" **STRING**

(c-addr1 c-addr2 u --)

If *u* is greater than zero, copy *u* consecutive characters from *c-addr1* to *c-addr2*. Equivalent to **CMOVE** , **CMOVE>** or **MOVE** if *c-addr1* is not in screen memory. Only use **TSCREEN>** to do block moves from the screen to normal memory. Block moves across screen memory are not supported. They will hang the machine.

See also: **>TSCREEN**

TUCK **CORE EXT**

(x1 x2 -- x2 x1 x2)

Copy the first (top) stack item below the second stack item.

TYPE **CORE**

(c-addr u --)

If *u* is greater than zero, display the character string specified by *c-addr* and *u*.

See also: **EMIT**

U **IFORTH**

(-- x) (S: x y z -- x y z)

Copy the element that is third from the top of the system stack to the data stack.

U+cy "u-plus-carry" **IFORTH**

(u1 u2 -- ud)

Add *u2* to *u1*, giving the result *ud*. Alternatively, this can be thought of as (*u1 u2* -- *u1+u2* carry), with carry 0 or 1.

See also: **UM+ UM- UD+c**

U. "u-dot" **CORE**

(u --)

Display *u* in free field format.

U.R "u-dot-r" **CORE EXT**

(u n --)

Display *u* right aligned in a field *n* characters wide. If the number of characters required to display *u* is greater than *n*, all digits are displayed with no leading spaces in a field as wide as necessary.

U1 **IFORTH**

(-- x) (S: x y z u -- x y z u)

Copy the element that is fourth from the top of the system stack to the data stack.

U2 **IFORTH**

(-- x) (S: x y z u v -- x y z u v)

Copy the element that is fifth from the top of the system stack to the data stack.

U3 **IFORTH**

(-- x) (S: x y z u v w -- x y z u v w)

Copy the element that is sixth from the top of the system stack to the data stack.

U< "u-less-than" **CORE**

(u1 u2 -- flag)

flag is true if *u1* is less than *u2*.

U<= "unsigned-less-equal" **IFORTH**

(u1 u2 -- flag)

flag is true if *u1* is unsigned less then, or equal to *u2*.

U> "u-greater-than" **CORE EXT**

(u1 u2 -- flag)

flag is true if *u1* is greater than *u2*.

U>= "unsigned-greater-equal" **IFORTH**

(u1 u2 -- flag)

flag is true if *u1* is unsigned greater then, or equal to *u2*.

U>D "u-to-d" **IFORTH**

(u -- ud)

ud is the double number equivalent to *u*.

UD+c "u-d-plus-c" **IFORTH**

(d1 d2 carry1 -- d3 carry2)

Adds two doubles, taking into count a previous *carry1*. The *carry2* is 1 when the result *d3* overflows, 0 elseway. Allowed values for *carry1* are 0 or a value with the LSB set.

UD. "u-d-dot" **IFORTH**

(ud --)

Display *ud* in free field format.

UD.R "u-d-dot-r" **IFORTH**

(ud n --)

Display *ud* right aligned in a field |*n*| characters wide. If the number of characters required to display *ud* is greater than |*n*|, all digits are displayed with no leading spaces in a field as wide as necessary. (In **UD.R** , R stands for RIGHT). When *n* is negative the fill character will be '0', not **BL** .

UDEC. "u-dec-dot" **IFORTH**

(x --)

Print *x* as an unsigned decimal number. The current number base is not changed.

UM* "u-m-star" **CORE**

(u1 u2 -- ud)

Multiply *u1* by *u2*, giving the unsigned double-cell product *ud*. All values and arithmetic are unsigned.

UM*/ "u-m-star-slash" **IFORTH**

(ud1 u1 u2 -- ud2)

Multiply *ud1* by *u1* producing the triple-cell intermediate result t. Divide t by *u2* giving the double-cell quotient *ud2*. An ambiguous condition exists if *u2* is zero, or the quotient lies outside of the range of a double-precision integer.

UM+ "u-m-plus" **IFORTH**

(ud1 u -- ud2)

Add *u* to *ud1*, giving the result *ud2*.

See also: **U+cy UM- UD+c**

UM- "u-m-minus" **IFORTH**

(borrow n1 n2 -- d)

Subtract *n2* from *n1*, taking the *borrow* (when LSB set) into account. Alternatively, this can be thought of as (*borrow n1 n2 -- n1-n2 borrow*), with *borrow* 0 or -1.

See also: **UM+ UD+c**

UM/MOD "u-m-slash-mod" **CORE**

(ud u1 -- u2 u3)

Divide *ud* by *u1*, giving the quotient *u3* and the remainder *u2*. All values and arithmetic are unsigned. An ambiguous condition exists if *u1* is zero or if the quotient lies outside the range of a single-cell unsigned integer.

UMAX "u-max" **IFORTH**

(u1 u2 -- u)

u is the unsigned maximum of the numbers *u1* and *u2*.

UMIN "u-min" **IFORTH**

(u1 u2 -- u)

u is the unsigned minimum of the numbers *u1* and *u2*.

UNBUFFERED **IFORTH**

(fam2 -- fam1)

fam1 is a file access method such as **R/W** **R/O** or **W/O** **BIN** . *fam2* is a file access method with the same properties as *fam1* but for which all buffering is turned off. This is necessary when, *e*.g., reading from a pipe or socket.

See also: **CREATE-FILE** **OPEN-FILE** **READ-FILE** **WRITE-FILE** **BIN**

UNDER+ "under-plus" **IFORTH**

(a b c -- a+c b)

Add TOS to the third on stack. Equivalent to **ROT** + **SWAP** .

UNLOOP **C** **CORE**

(--) (R: sys --)

Discard the loop control parameters. The loop control parameters must have been available.

See also: **LEAVE**

UNNEXT **C** **IFORTH**

(--)

Discard the loop control parameter of a **FOR** **NEXT** loop. The control parameter must have been available.

See also: **UNLOOP**

UNTIL **C** **CORE**

Compilation: (dest --)

Append the execution semantics given below to the current definition, resolving the backward reference *dest*.

Execution: (x --)

If all bits of *x* are zero, continue execution at the location specified by *dest*.

See also: **BEGIN**

UNTIL, **IFORTH-ASSEMBLER**

Assembling: (addr --)

Assemble a backward conditional branch to memory location *addr*. *addr* is the location marked by BEGIN, .

Execution: (--)

If flag is false, take the branch. Otherwise do not take the branch.

See also: **BEGIN,**

UNUSED **CORE EXT**

(-- u)

u is the amount of space remaining in the region addressed by **HERE** , in address units.

UP "u-p" **IFORTH**

(-- a-addr)

a-addr is the address of the table of user (and internal) variables.

See also: **USERBASE**

UP! "u-p-store" **IFORTH**

(a-addr --)

Set the address of the table of user (and internal) variables.

UPDATE **BLOCK**

(--)

Mark the current block buffer as modified. **UPDATE** does not immediately cause I/O. If there is no current block buffer, **UPDATE** does nothing. Other standard implementations can have an ambiguous condition if there is no current block buffer.

See also: **BLOCK BUFFER FLUSH SAVE-BUFFERS**

USE **IFORTH**

("file" --)

Parse file delimited by a space, ignoring leading delimiters. Open file. Use the open file as the storage for blocks by storing the file identifier in the variable **BLOCK-FID** . The current file is closed before opening a new one is attempted.

USE-BLOCKS **IFORTH**

(c-addr u --)

Open the file identified by *c-addr* and *u*. Use the open file as the storage for blocks by storing the file identifier in the variable **BLOCK-FID** . The current file is closed before opening a new one is attempted.

USER **D** **IFORTH**

Compilation: ("name" --)

Parse *name* delimited by a space, ignoring leading delimiters. Create a dictionary entry for *name* with the execution semantics defined below. Reserve one cell of memory at an aligned address in the so-called "user-area". Variables in the user-area are inherited by sub-processes. Modifying a variable from the user-variable space does not change the corresponding variable for either the parent (or any ancestors) or children (or any siblings). The application is responsible for initializing the contents of the reserved cell. The total number of user variables per process is limited to 64. *name* is referred to as a "user-variable".

Execution: (-- a-addr)

a-addr is the address of the reserved cell.

USERBASE "userbase" **IFORTH**

(-- a-addr)

a-addr is the address of a variable that holds the address of the current task's **UP** .

See also: **UP**

UT* "u-t-star" **IFORTH**

(ulo uhi u -- utlo utmid uthi)

Unsigned multiplication of a double *uhi*:*ulo* with *u*, returning a triple result.

See also: **MT* UT/**

UT/ "u-t-slash" **IFORTH**

(utlo utmid uthi u -- ud)

Unsigned division of a triple length *uthi*:*utmid*:*utlo* by *u*, returning a double length result *ud*. Overflow is ignored.

See also: **MT* UT***

V>HASH "v-to-hash" **IFORTH**

(vfa -- hfa)

hfa is address of the hash table of the vocabulary whose associated descriptor table has start address *vfa*.

VALUE **D** **CORE EXT**

("name" --)

Parse *name* delimited by a space, ignoring leading delimiters. Create a dictionary entry for *name* with the execution semantics defined below. *name* is referred to as a "value".

Execution: ("name" -- x)

Place *x* on the stack. The value of *x* is unspecified until the phrase *x* **TO** *name* is executed, causing *x* to be associated with *name*.

VARIABLE **D** **CORE**

("name" --)

Parse *name* delimited by a space, ignoring leading delimiters. Create a dictionary entry for *name* with the execution semantics defined below. Reserve one cell of data space at an aligned address. *name* is referred to as a "variable."

Execution: ("name" -- a-addr)

a-addr is the address of the reserved cell. The application is responsible for initializing the contents of the reserved cell.

VER **IFORTH**

VER is an environment query.

See also: **ENVIRONMENT?**

VFOREIGN **IFORTH**

(n{u} n 'service --)*

Calls a dynamically linked "**C**" library routine at address *'service*, with *n* unsigned arguments *u*. There is no result.

See also: **FOREIGN DFOREIGN FFOREIGN CALLBACK**

VISIBLE **IFORTH**

(dea --)

Make the dictionary entry identified by *dea* visible again.

See also: **HIDE PRIVATE**

VOCABULARY **IFORTH**

("name" --)

Parse *name* delimited by a space, ignoring leading delimiters. Create a dictionary entry for *name* with the execution semantics defined below. Create a new wordlist and store the wordlist identifier with the new word. *name* is referred to as a "vocabulary".

Execution: ("name" --)

Make the above created wordlist the current wordlist.

W-LINK "w-link" **IFORTH**

(-- a-addr)

a-addr is the address of **W-LINK** . **W-LINK** contains the address of the associated data structure of the most recently defined wordlist. Each of the associated data structures contain a similar pointer to their predecessor, thus forming a list of all available wordlists. Note that by traversing this list all wordlists are found, even those that are not currently available via the search order.

W/O "w-o" **FILE**

(-- x)

x is the implementation-dependent value for selecting the "write only" file access method.

See also: **CREATE-FILE OPEN-FILE**

WAIT? **IFORTH**

(-- flag)

Check whether a key has been pressed on the keyboard. If no key has been pressed return false. If the **ESC** key has been pressed read that key from the input stream and discard it, then return true. Otherwise wait for another key to be pressed. If the **ESC** key has been pressed return true, otherwise return false. The above sequence pauses a program when a key is pressed and the program continues when a second key is read. If any of the keys is the **ESC** key then return true. This word can be used in a loop as follows:

```
: do-it 100000 0 DO ... WAIT? IF LEAVE THEN ... LOOP ;
```

See also: **BREAK?**

WARNING **IFORTH**

(-- a-addr)

a-addr is the address of **WARNING** . **WARNING** contains a flag that is checked whenever a redefinition message is to be printed. If the flag is false, the message is not printed, otherwise the message is printed.

WHERE-I-AM **IFORTH**

(-- c-addr u)

This tells the MS-DOS iForth where its bin directory is. In the result '/' has been converted back to '\' so that a 'cd ' or 'dir ' can be done with it. This string is actually the zero-th

command line parameter, so 0 **'PARAM** returns the same string, without the '/' conversion. Only useful for MS-DOS.

See also: **'PARAM**

WHILE **C** **CORE**

Compilation: (dest -- orig dest)

Put the location of a new unresolved forward reference *orig* onto the control flow stack, under the existing *dest*. Append the execution semantics given below to the current definition. The semantics are incomplete until *orig* and *dest* are resolved (*e*.g., by **REPEAT**).

Execution: (x --)

If all bits of *x* are zero, continue execution at the location specified by the resolution of *orig*.

WHILE, **IFORTH-ASSEMBLER**

Assembling: (-- addr)

Place the value of the current code space pointer on the data stack. Assemble a conditional forward branch. The forward branch is to be resolved by REPEAT, .

Execution: (--)

If flag is false, take the branch. Otherwise do not take the branch.

See also: **BEGIN, REPEAT,**

WIN2K? **IFORTH**

(-- bool)

TRUE if this is iForth for Windows.

See also: **LINUX? MS-DOS? TFORTH? XLINUX?**

WIN95? **IFORTH**

(-- bool)

TRUE if this is iForth for Windows.

See also: **LINUX? MS-DOS? TFORTH? XLINUX?**

WIN98? **IFORTH**

(-- bool)

TRUE if this is iForth for Windows.

See also: **LINUX? MS-DOS? TFORTH? XLINUX?**

WINME? **IFORTH**

(-- bool)

TRUE if this is iForth for Windows.

See also: **LINUX? MS-DOS? TFORTH? XLINUX?**

WINNT? **IFORTH**

(-- bool)

TRUE if this is iForth for Windows.

See also: **LINUX? MS-DOS? TFORTH? XLINUX?**

WINXP? **IFORTH**

(-- bool)

TRUE if this is iForth for Windows.

See also: **LINUX? MS-DOS? TFORTH? XLINUX?**

WITHIN **CORE EXT**

(n1|u1 n2|u2 n3|u3 -- flag)

flag is true if *n1|u1* is less than *n3|u3* and not less than *n2|u2*. All comparisons are performed in a circular number space. An ambiguous condition exists if *n1|u1*, *n2|u2*, and *n3|u3* are not all the same type. Equivalent to: **OVER - >R - R> U<**

WORD **CORE**

(char "ccc<char>" -- c-addr)

char is a single character delimiter. Parse characters *ccc* delimited by *char*, ignoring leading delimiters. An ambiguous condition exists if the length of the parsed string is greater than the implementation-defined length of a counted string. *c-addr* is the address of a transient region containing the parsed word as a counted string. If the current input stream was empty or contained no characters other than the delimiter, the resulting string has a zero count. A space, not included in the count, follows the string. Note: The requirement to follow the string with a space is obsolescent and is included as a concession to existing programs that use **CONVERT** . A standard program may not depend on the existence of the space.

WORDLIST **SEARCH**

(-- wid)

Creates a new empty word list, returning its word list identifier *wid*. The new word list is dynamically allocated in data space. Note that other ANS systems may create the new word list in another place.

WORDLISTS **CORE**

WORDLISTS is an environment query.

See also: **ENVIRONMENT?**

WORDS **TOOLKIT EXT**

(--)

List the word names in the first word list of the search order. The format of the display is implementation-dependent. iForth prints all words in as many columns as will fit on the screen. The display can be halted and restarted with any key not equal to the **ESC** key. Pressing **ESC** at any time will abort **WORDS** . The variable **WORDS-DELAY** controls scrolling speed.

See also: **WORDS: WORDS? WORDS-DELAY**

WORDS-DELAY **IFORTH**

(-- a-addr)

Contains the number of milliseconds to wait per line of output in **WORDS** **WORDS:** and **WORDS?**

See also: **WORDS** **WORDS:** and **WORDS?**

WORDS: **IFORTH**

("substring" --)

List those word names in the first word list of the search order that partly match *name*, a blank-delimited string that must be parsed off the current input stream. The format of the display is implementation-dependent, see **WORDS** . The string *name* is matched when it encompasses the *substring*. The search is case-insensitive. The variable **WORDS-DELAY** controls scrolling speed.

See also: **WORDS** **WORDS?** **WORDS-DELAY**

WORDS? **IFORTH**

("name" --)

List those word names in the first word list of the search order that match *name*, a blank-delimited string that must be parsed off the current input stream. The format of the display is implementation-dependent, see **WORDS** . The string *name* is matched exactly unless it contains any of the characters '*' or '?'. The '?' matches any character, the '*' matches any character string. Because a lot of Forth words contain '?' or '*' characters, this is not an ideal solution. The variable **WORDS-DELAY** controls scrolling speed.

See also: **WORDS** **WORDS:** **WORDS-DELAY**

WRITE-FILE **FILE**

(c-addr u1 fileid -- ior)

Write *u1* characters from *c-addr* to the file identified by *fileid* starting at its current position. *ior* is the implementation-defined **I/O** result code. At the conclusion of the operation, **FILE-POSITION** returns a value past the characters written to the file and **FILE-SIZE** returns a value greater than or equal to the value returned by **FILE-POSITION** . **REPOSITION-FILE** will let you move the file pointer behind the end of file marker. This is no error. As soon as a write operation is attempted, the file is made larger. The contents of the file between the previous end of file and the file pointer before the write are garbage. There is a portability problem with the end-of-line sequence if this word is used to write text files.

See also: **READ-FILE** **READ-LINE**

WRITE-LINE **FILE**

(c-addr u1 fileid -- ior)

Write *u1* characters from *c-addr* followed by the implementation-dependent line terminator to the file identified by *fileid* starting at its current position. *ior* is the implementation-defined **I/O** result code. The iForth server translates a line feed ($0A) to the OS-dependent end of line sequence. See also **$CR** . Note that the Standard requires us to disclose this fact. You will need to know the end of line sequence only when using **READ-FILE** and **WRITE-FILE** for text. We do not recommend this practice. At the conclusion of the operation, **FILE-POSITION** returns a value past the characters written to the file and **FILE-SIZE** returns a value greater than or equal to the value returned by **FILE-POSITION** .

See also: **READ-FILE** **READ-LINE**

XEXIT **IFORTH**

(--)

De-initialize the assembler, *e*.g. fixup labels. Normally handled behind the scene.

XF! "x-f-store" **IFORTH**

(a-addr1 --) (F: r --)

Store the floating-point number *r* as an 80-bit float number at *a-addr1*. Note that in other implementations of the ANS standard *r* may have more digits of precision or may be too large for representation as a 80-bit IEEE number, so rounding or overflow might occur.

See also: **F!**

XF!+ "x-f-store-plus" **IFORTH**

(a-addr1 -- a-addr2) (F: r --)

Store the floating-point number *r* as a 80-bit IEEE extended precision number at *a-addr1*, and leave the incremented pointer as *a-addr2*. Note that in other implementations of the ANS standard *r* may have more digits of precision or may be too large for representation as a 80-bit IEEE number, so rounding or overflow might occur.

See also: **F!+**

XF+! "x-f-plus-store" **IFORTH**

(F: r --) (a-addr --)

Add the IEEE extended *r* to the extended at *a-addr*.

See also: **F+!**

XF@ "x-f-fetch" **FLOAT EXT**

(a-addr --) (F: -- r)

Fetch the 80-bit IEEE extended precision number stored at *a-addr* to the floating-point stack as *r* in the internal representation. If the IEEE extended precision significand has more precision than the internal representation it will be rounded to the internal representation using the "round to nearest" rule.

See also: **F@**

XF@+ "x-f-fetch-plus" **IFORTH**

(a-addr1 -- a-addr2) (F: -- r)

Fetch the 80-bit IEEE number stored at *a-addr1* to the floating-point stack as *r* in the internal representation. Also leave the incremented pointer as *a-addr2*.

See also: **F@+**

XFALIGN **IFORTH**

(--)

If the address of the next available data space location is not aligned, reserve enough space to align it to a location which is suitable to hold a IEEE extended precision floating point number.

XFALIGNED **IFORTH**

(addr -- a-addr)

a-addr is the first aligned address greater than or equal to *addr* and which is suitable to reference a IEEE extended precision floating point number.

XFLOAT+ "x-float-plus" **FLOAT**

(a-addr1 -- a-addr2)

Add the size of an IEEE extended precision floating point number, specified in address units, to *a-addr1*, giving *a-addr2*.

XFLOAT- "x-float-min" **IFORTH**

(a-addr1 -- a-addr2)

Subtract the size of an IEEE extended precision floating point number, specified in address units, from *a-addr1*, giving *a-addr2*.

XFLOATS "x-floats" **FLOAT**

(n1 -- n2)

n2 is the size, in address units, of *n1* IEEE extended precision floating point numbers.

XFLOAT[] **IFORTH**

(addr1 index -- addr2)

Equivalent to **XFLOATS** + .

See also: **[]XFLOAT**

XFPUSHD "x-f-push-d" **IFORTH**

(-- ud) (F: r --)

Move a 64-bit item from the float to the parameter stack, in Intel format. To use in Forth add **SWAP** or use **FPUSHD** instead.

See also: **FPUSHS FPUSHD**

XINIT **IFORTH**

(--)

Initialize the assembler. Normally handled behind the scene.

XLINUX? **IFORTH**

(-- bool)

TRUE if this is iForth for Linux and is running under X.

See also: **LINUX? MS-DOS? TFORTH? WINNT?**

XOR "x-or" **CORE**

(x1 x2 -- x3)

x3 is the bit-by-bit exclusive-or of *x1* with *x2*.

Z” "z-quote” **IFORTH**

Compilation: ("ccc<">" --)

Parse characters *ccc* delimited by " (double quote). Append the execution semantics given below to the current definition.

Execution: (-- c-addr u)

Return *c-addr* and *u* that describe a string consisting of the characters *ccc*.

Interpretation:("ccc<">" -- c-addr u)

Parse characters *ccc* delimited by " (double quote). Store the resulting string at a temporary location described by *c-addr* and *u*. The maximum length of the temporary buffer is 255 characters but other implementations of the standard may limit this to 80 characters. A standard program may not alter the returned string. The difference with **S"** is that there is a 0 character appended to the returned string. The 0 is not included in the string count. These strings intention are unfortunately necessary to communicate with the WIN32 and X GUIs.

See also: **S" S~**

ZLOCAL "z-local” **C** **IFORTH**

Compilation: ("name" --)

Parse *name* delimited by a space, ignoring leading delimiters. Create a temporary dictionary entry for *name* with the execution semantics defined below. **ZLOCAL** can only be used inside a definition. *name* remains in the dictionary until the current definition is finished with ; **DOES>** or **;CODE** .

Execution: (F: r1 r2 --)

Initialize *name* with the floating-point value z = (*r1*,*r2*).

Execution: (--) (F: -- r1 r2)

Place z on the floating-point stack.

ZLOCALS| **C** **IFORTH**

*Compilation: ("name"*n "|" --)*

Parse names delimited by a blank, ignoring leading delimiters. The list of names is terminated by a '|' (bar). Create a temporary dictionary entry for each *name* with the execution semantics defined below. **ZLOCALS|** can only be used inside a definition. The names remain in the dictionary until the current definition is finished with ; **;CODE** or **DOES>** . iForth accepts an unlimited number of names. The ANS standard requires a minimum of eight.

Execution: (F: -- r1 r2)

Place the complex number z = (*r1*,*r2*) on the stack. The value of z is unspecified until the phrase z **TO** *name* is executed, causing z to be associated with *name*.

See also: **LOCALS| DLOCALS| FLOCALS|**

["left-bracket” **C** **CORE**

(--)

Enter interpretation state. [is an immediate word. Use of this word in interpret state is an error. Typical use: : X ... [4321] **LITERAL** ... ;

See also:]

['] "bracket-tick" **C** **CORE**

Compilation: ("name" --)

Parse *name* delimited by a space, ignoring leading delimiters. Find *name*. Compile *name*'s execution token as a literal. An ambiguous condition exists if *name* is not found in the search order or if *name* is a standard word with the **C** attribute.

Execution: (-- xt)

Place *name*'s execution token *xt* on the stack.

See also: '

[32]fil, **IFORTH-ASSEMBLER**

Assembling: (--)

Execution: (internal F: ---)

Assembler macro that generates code for the FILD dword ptr [EAX] opcode.

[32]fist, **IFORTH-ASSEMBLER**

Assembling: (--)

Execution: (internal F: a ---)

Assembler macro that generates code for the FISTP dword ptr [EAX] opcode.

[32]fld, **IFORTH-ASSEMBLER**

Assembling: (--)

Execution: (internal F: --- a)

Assembler macro that generates code for the FLDP dword ptr [EAX] opcode.

[32]fst, **IFORTH-ASSEMBLER**

Assembling: (--)

Execution: (internal F: a ---)

Assembler macro that generates code for the FSTP word ptr [EAX] opcode.

[64]fil, **IFORTH-ASSEMBLER**

Assembling: (--)

Execution: (internal F: ---)

Assembler macro that generates code for the FILD qword ptr [EAX] opcode.

[64]fist, **IFORTH-ASSEMBLER**

Assembling: (--)

Execution: (internal F: a ---)

Assembler macro that generates code for the FISTP qword ptr [EAX] opcode.

[64]fld, **IFORTH-ASSEMBLER**

Assembling: (--)

Execution: (internal F: --- a)

Assembler macro that generates code for the FLDP dword ptr [EAX] opcode.

[64]fst, IFORTH-ASSEMBLER

Assembling: (--)

Execution: (internal F: a ---)

Assembler macro that generates code for the FSTP qword ptr [EAX] opcode.

[80]fld, IFORTH-ASSEMBLER

Assembling: (--)

Execution: (internal F: --- a)

Assembler macro that generates code for the FLDP tword ptr [EAX] opcode.

[80]fst, IFORTH-ASSEMBLER

Assembling: (--)

Execution: (internal F: a ---)

Assembler macro that generates code for the FSTP tword ptr [EAX] opcode.

[CHAR] "bracket-char" **C CORE**

Compilation: ("ccc< >" --)

Parse characters *ccc* delimited by a space, ignoring leading delimiters. Compile *char*, the integer value of the first character of *ccc*, as a literal.

Execution: (-- char)

Place *char* on the stack.

[COMPILE] "bracket-compile" **C CORE EXT**

Compilation: ("name" --)

Parse *name* delimited by a space, ignoring leading delimiters. If *name* has default compilation semantics, append the execution semantics of *name* to the current definition; otherwise append the compilation semantics of *name*. An ambiguous condition exists if *name* is not found.

[CTRL] C IFORTH

Compilation: ("ccc" --)

Parse characters *ccc* delimited by a space, ignoring leading delimiters. Compile *char*, the lower 5 bits of the integer value of the first character of *ccc*, as a literal. *e*.g. **[CTRL]** E gives 5 which is also ^E.

Execution: (-- char)

Place *char* on the stack.

[ELSE] "bracket-else" **TOOLKIT-EXT**

(--)

Parse and discard blank-delimited words from the input stream, ignoring leading delimiters, until the word **[THEN]** has been parsed and discarded. If the input stream is exhausted while parsing words, it is refilled as with **REFILL** . Nested occurrences of **[IF]** ... **[THEN]** or **[IF]** ... **[ELSE]** ... **[THEN]** are discarded. This word is immediate.

[IF] "bracket-if" **TOOLKIT-EXT**

(flag --)

If the *flag* is true, do nothing. Otherwise parse and discard blank-delimited words from the input stream, ignoring leading delimiters, until either the word **[ELSE]** or the word **[THEN]** has been parsed and discarded. If the input stream is exhausted while parsing words, it is

refilled as with **REFILL** . Nested occurrences of **[IF]** ... **[THEN]** or **[IF]** ... **[ELSE]** ... **[THEN]** are discarded. This word is immediate. An ambiguous condition exists when **[IF]** is POSTPONEd, or if the end of the input stream is reached and cannot be refilled before the terminating **[ELSE]** or **[THEN]** is parsed. This word is immediate.

[THEN] "bracket-then" **TOOLKIT-EXT**

(--)

Terminates an **[IF]** construction. This word itself does nothing. This word is immediate.

See also: **[IF] [ELSE]**

[XCOMPILE] **IFORTH**

(--)

Undocumented experimental word.

See also: **:FAST FAST; FI/FO I/O**

[]CELL "cell-array" **IFORTH**

(u a-addr1 -- c-addr2)

c-addr2 is the address that has an *offset* of *u* cells into the cell array at *a-addr1*.

[]DFLOAT "dfloat-array" **IFORTH**

(index addr1 -- addr2)

Equivalent to **SWAP DFLOATS** + .

See also: **DFLOAT[] FLOAT[] SFLOAT[] XFLOAT[]**

[]DOUBLE "double-array" **IFORTH**

(u a-addr1 -- c-addr2)

c-addr2 is the address that has an *offset* of *u* double cells into the double cell array at *a-addr1*.

[]FLOAT "float-array" **IFORTH**

(u a-addr1 -- c-addr2)

c-addr2 is the address that has an *offset* of *u* floating point number elements into the floating-point array at *a-addr1*.

[]SFLOAT "sfloat-array" **IFORTH**

(u a-addr1 -- c-addr2)

c-addr2 is the address that has an *offset* of *u* floating point number elements into the single-precision floating-point array at *a-addr1*.

[]XFLOAT "x-float-array" **IFORTH**

(u a-addr1 -- c-addr2)

c-addr2 is the address that has an *offset* of *u* floating point number elements into the extended-precision floating-point array at *a-addr1*.

**** "backslash" **CORE EXT**, **BLOCK EXT**, **FILE EXT**

("ccc<eol>" --)

Ignore the remainder of the current input stream. When **BLK** contains zero, ignore the remainder of the current input stream; otherwise ignore the portion of the current input stream

corresponding to the remainder of the current line. When SOURCE-FILE contains zero, ignore the remainder of the current input stream; otherwise ignore the remainder of the current line.

] "right-bracket" **CORE**

(--)

Enter compilation state.

See also: **[**

]OF **C** **IFORTH**

(--)

Tests for a flag, not for equality to the selector. If the branch is taken the selector is dropped. Example of use: **CASE** *x* 1 **OF ."** 1 " **ENDOF DUP** 9 19 **WITHIN]OF ."** 9 .. 19 " **ENDOF** 3333 **OF ."** 3333 " **ENDOF ."** huh?" **ENDCASE**

See also: **OF ENDOF**

^! **IFORTH**

(n addr --)

Xor *n* with the value at *addr* and store the result there.

See also: **+! |! -!**

_EMIT "underscore-emit" **IFORTH**

(char --)

Send *char* to the host, using the boot link. The boot link is acquired first.

_EMIT? "underscore-emit-query" **IFORTH**

(-- bool)

Test if a *char* can be sent to the host, using the boot link. The boot link is acquired first and released after the test.

_KEY **IFORTH**

(-- c)

Read a character from the link. If a character is not available yet, the current process is suspended for a fraction of a second and the operation is repeated until a character is available. This is the primitive for **EKEY** .

_KEY? **IFORTH**

(-- bool)

Test if a character is available on the link. If a character is not available yet, return false, else return true. This is the primitive for **EKEY?** .

_RX **IFORTH**

(-- c)

Read a byte from the link. If a byte is not available yet, the current process is suspended for a fraction of a second and the operation is repeated until a byte is available.

_RX$ **IFORTH**

(c-addr u -- c-addr' u')

Read a string from the link and place it in the buffer described by *c-addr* and *u*. This word assumes it owns the link, so it is not fit for use in a multi-process environment.

_RXW **IFORTH**

(-- w)

Read a word from the link. This word assumes it owns the boot link, so it is not fit for use in a multi-process environment.

_SEND-DATA **IFORTH**

(c-addr u --)

Send a string of data from the buffer at *c-addr*. The server should see this as transparant data and perform no interpretation of the byte stream. This word assumes it owns the link, so it is not fit for use in a multi-process environment.

_SEND-FORTH **IFORTH**

(c-addr u --)

Send the string at *c-addr* to the resident Forth interpreter in the server, to be interpreted in the host environment. The DFW **C**-server just ignores this string. This word assumes it owns the link, so it is not fit for use in a multi-process environment.

_TX **IFORTH**

(b --)

Write a byte to the link. This word assumes it owns the link, so it is not fit for use in a multi-process environment. Note that the server does not automatically know what to do with the item you sent it.

_TX$ **IFORTH**

(c-addr u --)

Write a string from the buffer described by *c-addr* and *u* to the link. This word assumes it owns the link, so it is not fit for use in a multi-process environment. Note that the server does not automatically know what to do with the item you sent it.

_TXW **IFORTH**

(w --)

Write a word to the link. This word assumes it owns the link, so it is not fit for use in a multi-process environment. Note that the server does not automatically know what to do with the item you sent it.

_TYPE **IFORTH**

(c-addr u --)

Write a string to the link, with the intention to display it on the current output device. This word assumes it owns the link, so it is not fit for use in a multi-process environment.

__debug__ **IFORTH**

(-- a-addr)

A variable to turn optimizer messages on and off. Normally off.

See also: **__verbose__** **.pstack**

__tokdmp__ **IFORTH**

(-- a-addr)

A variable that controls the verbosity of the optimizer messages turned on with __verbose__. Normally off. When ON, the tokens of inlined words are dumped to the terminal output device.

See also: **__verbose__**

__verbose__ **IFORTH**

(-- a-addr)

A variable that controls the verbosity of the optimizer messages turned on with __debug__. Normally off.

See also: **__debug__** **.pstack**

align **IFORTH**

(-- a-addr)

The address of an 8 byte table that controls the compiler's alignment use, depending also on the value in **_calign_** . The shown alignment values are optimal for AMD Athlon class CPUs and might well be detrimental on other machines. On older Pentiums better alignment values might be 0 or 4. Byte #8, the alignment of colon definitions and **CODE** , is absolutely critical for some architectures. For AMD's Athlon a slowndown of a factor four is possible on selected code when this byte is set to 0 (An I/**D** cache consistency probem). The result of setting byte #5 to 16 are sometimes useful, sometimes not. Setting **_align_** values only says what the alignment value should be, when it is selected. The selection process itself is controlled by setting bits in **_calign_** . This array is normally adjusted in ./iforth.prf.

```
Byte0: alignment for BEGIN , normally 16.
Byte1: alignment for ELSE , normally 16.
Byte2: alignment for ENDIF , normally 16.
Byte3: alignment for AHEAD , normally 0. Do NOT change.
Byte4: alignment for IF , normally 0.
Byte5: alignment for UNTIL , normally 0.
Byte6: alignment for AGAIN , normally 0.
Byte7: alignment of colon and CODE definitions, normally 4 (4*16=64 bytes).
```

See also: **_calign_**

calign **IFORTH**

(-- a-addr)

The address of a variable that controls the compiler's alignment usage, operating together with the values in the **_align_** array. The variable holds a bit array to control which alignment is used for 7 flow control constructs. This variable is normally set in ./iforth.prf.

```
Bit0: alignment for BEGIN , normally ON.
Bit1: alignment for ELSE , normally OFF.
Bit2: alignment for ENDIF , normally ON.
Bit3: alignment for AHEAD , OFF.
Bit4: alignment for IF , normally OFF.
Bit5: alignment for UNTIL , normally OFF.
Bit6: alignment for AGAIN , normally OFF.
```

See also: **_align_**

currentfile$ **IFORTH**

(-- addr)

The address of a counted string containing the name of the last top-level **INCLUDED** file. Note that **INCLUDE** uses **INCLUDED** . The name is exactly what the user specified and does not

necessarily show a fully-qualified path. "top-level" means that the user specified name is only stored when FLEVEL is 0, *e*.g. when the terminal input device is not a file. This feature is used in several utilities from ./include/os.frt (EDIT IN). Note that **currentfile$** complements **whatname** .

See also: **INCLUDED whatname**

cy, **IFORTH-ASSEMBLER**

Assembling: (--)

Execution: (internal F: ---)

Assembler macro that tells IF, UNTIL, et al to generate a conditional jump based on the processor carry flag being TRUE.

dDO **C** **IFORTH**

Compilation: (-- dodest)

Execution: (n1 n2 --) (R: -- sys)

Like **DO** , but for non-wrapping **+LOOPd** .

See also: **+LOOPd**

doWORDS **IFORTH**

(xt --)

The wordlist traversing factor of **WORDS WORDS:** and **WORDS?** Supply the *xt* of a word that can filter (*addr* -- bool) where *addr* is the address of a counted string (name of the next word in the list). For example, to realize a clone of **WORDS** that only prints short words you can do: : SWORDS-*XT* (*addr* -- bool) **C@** 3 **<** ; ' SWORDS-*XT* **doWORDS** .

fabs, **IFORTH-ASSEMBLER**

Assembling: (--)

Execution: (F: --) (internal F: a --- a')

Assembler macro that generates the code for **FABS** .

fadd, **IFORTH-ASSEMBLER**

Assembling: (--)

Execution: (internal F: a b --- c)

Assembler macro that generates code that will add the two numbers on the internal floating-point stack during run-time.

false, **IFORTH-ASSEMBLER**

Assembling: (--)

Execution: (internal F: ---)

Assembler macro that tells IF, UNTIL, et al to generate a conditional jump that will be never taken.

fchs, **IFORTH-ASSEMBLER**

Assembling: (--)

Execution: (internal F: a --- b)

Assembler macro that generates code for the FCHS instruction (change sign).

fcompp, **IFORTH-ASSEMBLER**

Assembling: (--)

Execution: (internal F: a b ---)

Assembler macro that generates code for the FCOMPP instruction (compare the top two numbers on the fp stack and remove them).

fcos, **IFORTH-ASSEMBLER**

Assembling: (--)

Execution: (internal F: a --- b)

Assembler macro that generates code for the **FCOS** instruction (cosine).

fdiv, **IFORTH-ASSEMBLER**

Assembling: (--)

Execution: (internal F: a b --- c)

Assembler macro that generates code for the FDIV instruction (divide the top two numbers on the fp stack, *c* = a / *b*).

fdivr, **IFORTH-ASSEMBLER**

Assembling: (--)

Execution: (internal F: a b --- c)

Assembler macro that generates code for the FDIVR instruction (divide the top two numbers on the fp stack, *c* = *b* / a).

fdropTOS, **IFORTH-ASSEMBLER**

Assembling: (--)

Assembler macro that generates code to drop the top number on the external fp stack. The internal FPU TOS is **NOT** reloaded. For use with external libraries, **FOREIGN** , **CALLBACK** etc.

finit, **IFORTH-ASSEMBLER**

Assembling: (--)

*Execution: (internal F: i*r ---)*

Assembler macro that generates code for the FINIT instruction.

fld1, **IFORTH-ASSEMBLER**

Assembling: (--)

Execution: (internal F: --- 1.0e)

Assembler macro that generates code for the FLD1 instruction.

fldl2e, **IFORTH-ASSEMBLER**

Assembling: (--)

Execution: (internal F: --- log_2(e))

Assembler macro that generates code for the FLDL2E instruction.

fldl2t, **IFORTH-ASSEMBLER**

Assembling: (--)

Execution: (internal F: --- log_2(10))

Assembler macro that generates code for the FLDL2T instruction.

fldlg2, **IFORTH-ASSEMBLER**

Assembling: (--)

Execution: (internal F: --- log_10(2))

Assembler macro that generates code for the FLDLG2 instruction.

fldln2, **IFORTH-ASSEMBLER**

Assembling: (--)

Execution: (internal F: --- log_e(2))

Assembler macro that generates code for the FLDLN2 instruction.

fldpi, **IFORTH-ASSEMBLER**

Assembling: (--)

Execution: (internal F: --- pi)

Assembler macro that generates code for the FLDPI instruction.

fldz, **IFORTH-ASSEMBLER**

Assembling: (--)

Execution: (internal F: --- 0.0e)

Assembler macro that generates code for the FLDZ instruction.

fmul, **IFORTH-ASSEMBLER**

Assembling: (--)

Execution: (internal F: a b --- c)

Assembler macro that generates code for the FMUL instruction (multiply the top two numbers on the fp stack, *c* = a * *b*).

fnclex, **IFORTH-ASSEMBLER**

Assembling: (--)

Execution: (internal F: ---)

Assembler macro that generates code for the FNCLEX instruction (clear exception flags).

fpatan, **IFORTH-ASSEMBLER**

Assembling: (--)

Execution: (internal F: a b --- c)

Assembler macro that generates code for the FPATAN instruction (*c* = arctangent(a) / *b*).

fpop, **IFORTH-ASSEMBLER**

Assembling: (--)

Execution: (external F: a ---) (internal F: --- a)

Assembler macro that generates code that will pop the topmost number from the external floating-point stack and push it on the internal fp-stack.

See also: **`fget,`**

fpopTOS, **IFORTH-ASSEMBLER**

Assembling: (--)

Assembler macro that generates code to pop the external FPU stack to the internal FPU TOS. Although iForth caches the FPU TOS the old TOS is not saved (it is assumed invalid). For use with external libraries, **FOREIGN** , **CALLBACK** etc.

fptan, **IFORTH-ASSEMBLER**

Assembling: (--)

Execution: (internal F: a --- tan(a) 1.0e)

Assembler macro that generates code for the FPTAN instruction.

fpush, **IFORTH-ASSEMBLER**

Assembling: (--)

Execution: (external F: --- a) (internal F: a ---)

Assembler macro that generates code that will pop the topmost number from the internal floating-point stack and push it on the external fp-stack.

See also: **fput,**

fpushTOS, **IFORTH-ASSEMBLER**

Assembling: (--)

Assembler macro that generates code to push the internal FPU stack to the external FPU stack. Although iForth caches the FPU TOS the FPU TOS is not adjusted. For use with external libraries, **FOREIGN** , **CALLBACK** etc.

frndint, **IFORTH-ASSEMBLER**

Assembling: (--)

Execution: (internal F: a --- b)

Assembler macro that generates code for the FRNDINT instruction.

fsin, **IFORTH-ASSEMBLER**

Assembling: (--)

Execution: (internal F: a --- b)

Assembler macro that generates code for the **FSIN** instruction (sine).

fsincos, **IFORTH-ASSEMBLER**

Assembling: (--)

Execution: (internal F: a --- b c)

Assembler macro that generates code for the **FSINCOS** instruction (simultaneous sine and cosine generation).

fsqr, **IFORTH-ASSEMBLER**

Assembling: (--)

Execution: (internal F: a --- a^2)

Assembler macro to square FPU TOS.

fsqrt, **IFORTH-ASSEMBLER**

Assembling: (--)

Execution: (internal F: a --- b)

Assembler macro that generates code for the **FSQRT** instruction.

fsub, **IFORTH-ASSEMBLER**

Assembling: (--)

Execution: (internal F: a b --- c)

Assembler macro that generates code for the FSUB instruction (*c* = a - *b*).

fsubr, **IFORTH-ASSEMBLER**

Assembling: (--)

Execution: (internal F: a b --- c)

Assembler macro that generates code for the FSUBR instruction (*c* = *b* - a).

ftst, **IFORTH-ASSEMBLER**

Assembling: (--)

Execution: (internal F: a --- a)

Assembler macro that generates code for the FTST instruction (compare top of stack to 0.0e0).

logfile **IFORTH**

(-- c-addr)

Returns the address of a counted string buffer. The string at this address is the name of iForth's log file. The default name used is "forth.log". Change this with **S"** foobar.log" **logfile PACK DROP** . The buffer has room for 127 characters and a count.

lpop, **IFORTH-ASSEMBLER**

Assembling: (--)

Add a small code sequence to the code space which has the execution semantics defined below.

Execution: (--) (L: aa --)

Pop a value from the locals stack and place the value in the EBX register.

lpush, **IFORTH-ASSEMBLER**

Assembling: (--)

Add a small code sequence to the code space which has the execution semantics defined below.

Execution: (--) (L: -- a)

Push the value of the EBX register onto the locals stack.

ncy, **IFORTH-ASSEMBLER**

Assembling: (--)

Execution: (internal F: ---)

Assembler macro that tells IF, UNTIL, et al to generate a conditional jump based on the processor carry flag being FALSE.

nov, **IFORTH-ASSEMBLER**

Assembling: (--)

Execution: (internal F: ---)

Assembler macro that tells IF, UNTIL, et al to generate a conditional jump based on the processor flags indicating no integer overflow.

ov, **IFORTH-ASSEMBLER**

Assembling: (--)

Execution: (internal F: ---)

Assembler macro that tells IF, UNTIL, et al to generate a conditional jump based on the processor flags indicating integer overflow.

parse-hook **IFORTH**

(--)

A user-variable that contains the address of a routine that is executed by **PARSE** with a stack-diagram (*c-addr1 u1 -- c-addr2 u2*). The hook is intended to be used to implement a macro facility, see ./include/subst.frt .

See also: **PARSE**

pe, **IFORTH-ASSEMBLER**

Assembling: (--)

Execution: (internal F: ---)

Assembler macro that tells IF, UNTIL, et al to generate a conditional jump based on the processor flags indicating that the byte in the AL register has even parity.

po, **IFORTH-ASSEMBLER**

Assembling: (--)

Execution: (internal F: ---)

Assembler macro that tells IF, UNTIL, et al to generate a conditional jump based on the processor flags indicating that the byte in the AL register has odd parity.

pr0 **IFORTH-ASSEMBLER**

(-- offset)

offset is the *offset* in the user area of the first scratch register.

pr1 **IFORTH-ASSEMBLER**

(-- offset)

offset is the *offset* in the user area of the second scratch register.

pr2 **IFORTH-ASSEMBLER**

(-- offset)

offset is the *offset* in the user area of the third scratch register.

pr3 **IFORTH-ASSEMBLER**

(-- offset)

offset is the *offset* in the user area of the fourth scratch register.

resetwhat **IFORTH**

Set **FALSE** before calling **INCLUDED** or **INCLUDE-FILE** . If it is true afterwards, a user **THROW** routine may consult the string at **whatname** for the name of the file that caused the exception (note that files can be nested, so this is not necessarily the name of the file passed to INCLUDE)

rpop, **IFORTH-ASSEMBLER**

Assembling: (--)

Add a small code sequence to the code space which has the execution semantics defined below.

Execution: (--) (R: aa --)

Pop a value from the return stack and place the value into the EBX register.

rpush, **IFORTH-ASSEMBLER**

Assembling: (--)

Add a small code sequence to the code space which has the execution semantics defined below.

Execution: (--) (R: -- a)

Push the value of the EBX register onto the return stack.

spop, **IFORTH-ASSEMBLER**

Assembling: (--)

Add a small code sequence to the code space which has the execution semantics defined below.

Execution: (--) (S: aa --)

Pop a value from the system stack and place the value into the EBX register.

spush, **IFORTH-ASSEMBLER**

Assembling: (--)

Add a small code sequence to the code space which has the execution semantics defined below.

Execution: (--) (S: -- a)

Push the value of the EBX register onto the system stack.

text.bg **IFORTH**

(-- a-addr)

a-addr is the address of a variable that holds the text background color and/or attribute used when scrolling or clearing the screen. The default value stored here is 7. Storing arbitrary values in **text.bg** may result in an invisible cursor.

ticks/ms **IFORTH**

(-- a-addr)

On some operating systems, *a-addr* is the address of a variable that holds a special value after the execution of **?MS** .

See also: **useconds**

u<, **IFORTH-ASSEMBLER**

Assembling: (--)

Signals a following conditional jump mechanism like IF, or UNTIL, that the processor flags should be inspected for "U<" result.

u<=, **IFORTH-ASSEMBLER**

Assembling: (--)

Signals a following conditional jump mechanism like IF, or UNTIL, that the processor flags should be inspected for "U<=" result.

u>, **IFORTH-ASSEMBLER**

Assembling: (--)

Signals a following conditional jump mechanism like IF, or UNTIL, that the processor flags should be inspected for "U>" result.

u>=, **IFORTH-ASSEMBLER**

Assembling: (--)

Signals a following conditional jump mechanism like IF, or UNTIL, that the processor flags should be inspected for "U>=" result.

uDO **C** **IFORTH**

Compilation: (-- dodest)

Execution: (n1 n2 --) (R: -- sys)

Like **DO** , but for non-wrapping, up counting, **+LOOPu** .

See also: **+LOOPu**

useconds **IFORTH**

(-- a-addr)

On some operating systems, *a-addr* is the address of a variable that holds a special value after the execution of **?MS** .

See also: **ticks/ms**

whatline# **IFORTH**

(-- a-addr)

If iForth encounters an error when interpreting a source file, the line number where the error occurred is stored in *a-addr*. This is for the convenience of editing utilities.

See also: **whatname whatpos#**

whatname **IFORTH**

(-- c-addr)

If iForth encounters an error when interpreting a source file, the name of the disk file causing the problem is stored as a counted string at *c-addr*. This is for the convenience of editing utilities.

See also: **whatline# whatpos#**

whatpos# **IFORTH**

(-- a-addr)

If iForth encounters an error when interpreting a source file, the *offset* into the line that caused the trouble is stored in *a-addr*. This is for the convenience of editing utilities.

See also: **whatname whatline#**

|! **IFORTH**

(n addr --)

Or *n* with the value at *addr* and store the result there.

See also: **+! ^! -!**

}ASM **IFORTH**

(--)

Switch back from the **ASSEMBLER** vocabulary, in interpretative mode, to the regular Forth compiler. By default the compiler assumes empty stacks when the word **}ASM** executes. You can use **IN/OUT** and **ADJUST-STACK** to modify this. See the entry of Saturday, December 30, 2000, 12:52 AM in the ./bugs.txt file for more information.

See also: **}ASM IN/OUT ADJUST-STACK FIN/FOUT**

~MS **IFORTH**

(u --)

Wait *u* milliseconds. This word executes **PAUSE** to give other processes a chance to run. Therefore it can be (very) inaccurate.

See also: **MS**