| Word | Vocab | New Word | Type | Stack Diagram | Usage | Description |
|---|---|---|---|---|---|---|
| **?EF** | FORTH | yes | 1802 | --- nibble | - | Put the state of the four 1802 EF pins into lowest nibble of number on top of stack |
| **DI** | FORTH | Yes | 1802 | --- | - | Disables interrupts |
| **EI** | FORTH | yes | 1802 | --- | - | Enable interrupt processing |
| **INP** | FORTH | yes | 1802 | N -- byte | - | inputs a byte of data from a port selected by the 1802's N line to stack. Top byte of resulting word set to zero. |
| **OUTP** | FORTH | yes | 1802 | n port# --- | - | outputs value n to 1802 N-line port# |
| **QOFF** | FORTH | yes | 1802 | --- | - | Set the state of the 1802 Q pin to zero ( gnd ) |
| **QON** | FORTH | yes | 1802 | --- | - | Set the state of the 1802 Q pin to zero ( +5 V) |
| **!CODE** | FORTH | no | Compile | --- | - | Used internally to create new word headers |
| **#** | FORTH | no | Compile | d1 --- d2 | See #S | Generate from a double number d1, the next ascii character which is placed in an output string. Result d2 is the quotient after division by BASE, and is maintained for further processing. Used between <# and #>. |
| **#>** | FORTH | no | Compile | d --- addr count | - | Terminates numeric output conversion by dropping d, leaving the text address and character count suitable for TYPE. |
| **#S** | FORTH | no | Compile | d1 --- d2 | - | Generates ascii text in the text output buffer, by the use of #, until a zero double number n2 results. Used between <# and #>. |
| **(** | FORTH | no | Compile | --- | ( comment text) | Ignore a comment that will be delimited by a right parenthesis on the same line. May occur during execution or in a colon-definition. A blank after the leading parenthesis is required. |
| **(NUMBER)** | FORTH | no | Compile | d1 addr1 --- d2 addr2 | see NUMBER | Convert the ascii text beginning at addr1+l with regard to BASE. The new value is accumulated into double number d1, being left as d2. Addr2 is the address of the first unconvertable digit. |
| **:** | FORTH | no | Compile | --- | : cccc ... ; | Creates a dictionary entry defining cccc as equivalent to the following sequence of Forth word definitions '...' until the next ';' or ';CODE'. The compiling process is done by the text interpreter as long as STATE is non-zero. Other details are that the CONTEXT vocabulary is set to the CURRENT vocabulary and that words with the precedence bit set (P) are executed rather than being compiled. |
| **;** | FORTH | no | Compile | --- | - | Terminate a colon-definition and stop further compilation. Compiles the run-time ;S. |
| **;CODE** | FORTH | no | Compile | --- | : cccc .... ;CODE | Stop compilation and terminate a new defining word cccc by compiling (;CODE). Set the CONTEXT vocabulary to ASSEMBLER, assembling to machine code the following mnemonics. When cccc later executes in the form: cccc nnnn the word nnnn will be created with its execution procedure given by the machine code following cccc. That is, when nnnn is executed, it does so by jumping to the code after nnnn. An existing defining word must exist in cc prior to ;CODE |
| **;S** | FORTH | no | Compile | --- | - | Stop interpretation of a screen. ;S is also the run-time word compiled at the end of a colon-definition which returns execution to the calling procedure |
| **[** | FORTH | no | Compile | --- | : xxx [ words ] more ; | Suspend compilation. The words after [ are executed, not compiled. This allows calculation or compilation exceptions before resuming compilation with ] . See LITERAL, ] |
| **[COMPILE]** | FORTH | no | Compile | --- | : xxx [COMPILE] FORTH ; | [COMPILE] will force the compilation of an immediate definitions, that would otherwise execute during compilation. The above example will select the FORTH vocabulary then xxx executes, rather than at compile time. |
| **]** | FORTH | no | Compile | --- | See [ | Resume compilation, to the completion of a colon-definition |
| **<BUILDS** | FORTH | no | Compile | --- | : cccc <BUILDS ... DOES> ... ; | Each time cccc is executed, <BUILDS defines a new word with a high-level execution procedure. Executing cccc in the form sp?<br>cccc nnnn<br>uses <BUILDS to create a dictionary entry for nnnn with a call to the DOES> part for nnnn. When nnnn is later executed, it has the address of its parameter area on the stack and executes the words after DOES><br>in cccc. <BUILDS and DOES> allow runtime procedures to written in high-level rather than in assembler code (as required by ;CODE). |

| Word | Vocab | New Word | Type | Stack Diagram | Usage | Description |
|---|---|---|---|---|---|---|
| --> | FORTH | no | Compile | --- | | Continue interpretation with the next disc screen. (pronounced next-screen). |
| **BACK** | FORTH | no | Compile | addr -- | - | Calculate the backward branch offset from HERE to addr and compile into the next available dictionary memory address |
| **BEGIN** | FORTH | no | Compile | --- addr n (compiling) | BEGIN ... UNTIL<br>BEGIN ... AGAIN<br>BEGIN ... WHILE ... REPEAT | At run-time, BEGIN marks the start of a sequence that may be repetitively executed. It serves as a return point from the corresponding UNTIL, AGAIN or REPEAT.<br>When executing UNTIL, a return to BEGIN will occur if the top of the stack is false; for AGAIN and REPEAT a return to BEGIN always occurs.<br>At compile time BEGIN leaves its return address and n for compiler error checking. |
| **CFA** | FORTH | no | Compile | pfa --- cfa | - | Convert the parameter field address of a definition to its code field address. |
| **COMPILE** | FORTH | no | Compile | --- | - | When the word containing COMPILE executes, the execution address of the word following COMPILE is copied (compiled) into the dictionary. This allows specific compilation situations to be handled in addition to simply compiling an execution address (which the interpreter already does). |
| **CONSTANT** | FORTH | no | Compile | n -- | n CONSTANT cccc | creates word cccc, with its parameter field containing n. When cccc is later executed, it will push the value of n to the stack |
| **CONTEXT** | FORTH | no | Compile | -- addr U,L0 | - | A user variable containing a pointer to the vocabulary within which dictionary searches will first begin |
| **CREATE** | FORTH | no | Compile | --- | CREATE cccc | Used by such words as CODE and CONSTANT to create a dictionary header for a Forth definition. The code field contains the address of the words parameter field. The new word is created in the CURRENT vocabulary. |
| **CSP** | FORTH | no | Compile | ---- addr | - | A user variable temporarily storing the stack pointer position, for compilation error checking. |
| **DEFINITIONS** | FORTH | no | Compile | --- | cccc DEFINITIONS | Set the CURRENT vocabulary to the CONTEXT vocabulary. In the example, executing vocabulary name cccc made it the CONTEXT vocabulary and executing DEFINITIONS made both specify vocabulary cccc. |
| **DLITERAL** | FORTH | no | Compile | d --- d (executing)<br>d --- (compiling) | - | If compiling, compile a stack double number into a literal. Later execution of the definition containing the literal will push it to the stack. If executing, the number will remain on the stack. |
| **DOES>** | FORTH | no | Compile | --- | - | A word which defines the run-time action within a high-level defining word. DOES> alters the code field and first parameter of the new word to execute the sequence of compiled word addresses following DOES>. Used in combination with <BUILDS. When the DOES> part executes it begins with the address of the first parameter of the new word on the stack. This allows interpretation using this area or its contents. Typical uses include the Forth assembler, multidimensional arrays, and compiler generation. |
| **IMMEDIATE** | FORTH | no | Compile | --- | - | Mark the most recently made definition so that when encountered at compile time, it will be executed rather than being compiled. i.e. the precedence bit in its header is set. This method allows definitions to handle unusual compiling situations, rather. than build them into the fundamental compiler. The user may force compilation of an immediate definition by preceeding it with [COMPILE], |
| **INTERPRET** | FORTH | no | Compile | --- | See NUMBER | The outer text interpreter which sequentially executes or compiles text from the input stream (terminal or disc) depending on STATE. If the word name cannot be found after a search of CONTEXT and then CURRENT it is converted to a number according to the current base. That also failing, an error message echoing the name with a " ?" will be given.<br>Text input will be taken according to the convention for WORD. If a decimal point is found as part of a number, a double number value will be left. The decimal point has no other purpose than to force this action. |

| Word | Vocab | New Word | Type | Stack Diagram | Usage | Description |
|---|---|---|---|---|---|---|
| **LITERAL** | FORTH | no | Compile | n --- (compiling) | - | If compiling, then compile the stack value n as a 16 bit literal. This definition is immediate so that it will execute during a colon definition. The intended use is: : xxx [ calculate ] LITERAL ; Compilation is suspended for the compile time calculation of m value. Compilation is resumed and LITERAL compiles this value. |
| **NFA** | FORTH | no | Compile | pfa --- nfa | - | Convert the parameter. field address of a definition to its name field |
| **PFA** | FORTH | no | Compile | --- pfa | - | Convert the name field address of a compiled definition to its parameter field address |
| **SMUDGE** | FORTH | no | Compile | --- | - | Used during word definition to toggle the "smudge bit" in a definitions' name field. This prevents an uncompleted definition from being found during dictionary searches, until compiling is completed without error. |
| **STATE** | FORTH | no | Compile | --- addr | - | A user variable containing the compilation state. A non-zero value indicates compilation |
| **TASK** | FORTH | no | Compile | --- | - | A no-operation word which can mark the boundary between applications |
| **TRAVERSE** | FORTH | no | Compile | addr1 n --- addr2 | - | Move across the name field of a fig-FORTH variable length name field. addr1 is the address of either the length byte or the last letter. If n=1, the motion is toward hi memory; if n=-l, the motion is toward low memory. The addr2 resulting is address of the other end of the name. |
| **USER** | FORTH | no | Compile | n --- | n USER cccc | creates a user variable cccc. The parameter field of cccc contains n as a fixed offset relative to the user pointer register UP for this user variable. When cccc is later executed, it places the sum of its offset and the user area base address on the stack as the storage address of that particular variable |
| **VARIABLE** | FORTH | no | Compile | n --- | n VARIABLE cccc | creates the definition cccc with its parameter field initialized to n. When cccc is later executed, the address of its parameter field (containing n) is left on the stack, so that a fetch or store may access this location |
| **VOCABULARY** | FORTH | no | Compile | --- | VOCABULARY cccc | creates a vocabulary definition cccc. Subsequent use of cccc will make it the CONTEXT vocabulary which is searched first by INTERPRET. The sequence "cccc DEFINITIONS" will also make cccc the CURRENT vocabulary into which new definitions are placed. In fig-FORTH, cccc will be so chained as to include all definitions of the vocabulary in which cccc is itself defined. All vocabularies ultimately chain to Forth. By convention, vocabulary names are to be declared IMMEDIATE. See VOC-LINK. |
| **VOC-LINK** | FORTH | no | Compile | --- addr U | - | A user variable containing the address of a field in the definition of the most recently created vocabulary. All vocabulary names are linked by these fields to allow control for FORGET'ing thru multiple vocabularies. |
| **WIDTH** | FORTH | no | Compile | --- addr | - | In fig-FORTH, a user variable containing the maximum number of letters saved in the compilation of a definitions' name. It must be 1 thru 31, with a default value of 31. The name character count and its natural characters are saved, up to the value in WIDTH. The value may be changed at any time within the above limits |
| **.** | FORTH | no | Console | n --- | - | Print a number from a signed l6 bit two's complement value, converted according to the numeric BASE. A trailing blanks follows. Pronounced "dot" |
| **."** | FORTH | no | Console | --- | ." cccc" | Compiles an in-line string cccc (delimited by the trailing ") with an execution procedure to transmit the text to the selected output device. If executed outside a definition, ." will immediately print the text until the final ',. The maximum number of characters may be an installation dependent value. See (.") |
| **.LINE** | FORTH | no | Console | line scr -- | - | Print on the terminal device, a line of text from the disc by its line and screen number. Trailing blanks are suppressed. |
| **.R** | FORTH | no | Console | n1 n2 --- | - | Print the number n1 right aligned in a field whose width is n2. No following blank is printed. |
| **.S** | FORTH | yes | Console | --- | - | dump contents for stack to console output formatted as 16 bit hex numbers |
| **?** | FORTH | no | Console | addr --- | - | Print the value contained at the address in free format according to the current base. |
| **?COMP** | FORTH | no | Console | --- | - | Issue error message if not compiling |
| **?CSP** | FORTH | no | Console | --- | - | Issue error message if stack position differs from value saved in CSP. |
| **?ERROR** | FORTH | no | Console | f n -- | - | Issue an error message number n, if the boolean flag is true |

| Word | Vocab | New Word | Type | Stack Diagram | Usage | Description |
|---|---|---|---|---|---|---|
| **?EXEC** | FORTH | no | Console | --- | - | Issue an error message if not executing |
| **?LOADING** | FORTH | no | Console | --- | - | Issue an error message if not loading |
| **?PAIRS** | FORTH | no | Console | n1 n2 -- | - | Issue an error message if n1 does not equal n2. The message indicates that compiled conditionals do not match |
| **?STACK** | FORTH | no | Console | --- | - | Issue an error message is the stack is out of bounds. This definition may be installation dependent. |
| **?TERMINAL** | FORTH | no | Console | --- | - | Perform a test of the terminal keyboard for actuation of the break key. A true flag indicates actuation. This definition is installation dependent. |
| **<#** | FORTH | no | Console | --- | - | Setup for pictured numeric output formatting using the words:<br><# # #S SIGN #><br>The conversion is done on a double number producing text at PAD. |
| **>IH** | FORTH | yes | Console | --- | - | Receives data from the console formatted as Intel Hex and stores in memory address from the intel hex file. |
| **2.R** | FORTH | yes | Console | n --- | - | outputs lower byte of number on top of stack as two digit hex |
| **4.R** | FORTH | yes | Console | n --- | - | outputs number on top of stack as four digit hex |
| **A2H1** | FORTH | yes | Console | n1 n2 --- b | - | combines two nibbles on TOS into one byte - used by Intel Hex Loader |
| **BASE** | FORTH | no | Console | --- addr | - | A user variable containing the current number base used for input and output conversion |
| **BL** | FORTH | no | Console | --- c | - | A constant that leaves the ascii value for "blank". |
| **BLANKS** | FORTH | no | Console | addr count -- | - | Fill an area of memory beginning at addr with blanks |
| **C/L** | FORTH | yes | Console | --- n | - | Store the number of characters available per line on the console |
| **CAPS** | FORTH | yes | Console | --- addr | - | Leave address of caps lock variable |
| **COUNT** | FORTH | no | Console | addr1 --- addr2 n L0 | - | Leave the byte address addr2 and byte count n of a message text beginning at address addr1. It is presumed that the first byte at addr1 contains the text byte count and the actual text starts with the second byte. Typically COUNT is followed by TYPE. |
| **CR** | FORTH | no | Console | --- | - | Transmit a carriage return and line feed to the selected output device. |
| **D.** | FORTH | no | Console | d --- | - | Print a signed double number from a 32 bit two's complement value. The high-order l6 bits are most accessible on the stack. Conversion is performed according to the current BASE. A blank follows. Pronounced D-dot |
| **D.R** | FORTH | no | Console | d n --- | - | Print a signed double number d right aligned in a field n characters wide. |
| **DECIMAL** | FORTH | no | Console | --- | - | Set the numeric conversion BASE for decimal input-output. |
| **DIGIT** | FORTH | no | Console | c n1 --- n2 tf (ok)<br>c n1 --- ff (bad) | - | Converts the ascii character c (using base n1) to its binary equivalent n2, accompanied by a true flag. If the conversion isinvalid, leaves only a false flag. |
| **DPL** | FORTH | no | Console | ---- addr | - | A user variable containing the number of digits to the right of the decimal on double integer input. It may also be used hold output column location of a decimal point, in user generated formatting. The default value is -1 |
| **DUMP** | FORTH | no | Console | addr n --- | - | Print the contents of n memory locations beginning at addr. Both addresses and contents are shown in the current numeric base |
| **EMIT** | FORTH | no | Console | c --- | - | Transmit ascii character c to the selected output device. OUT is incremented for each character output. |
| **ENCLOSE** | FORTH | no | Console | addr1 c -- addr1 n1 n2 n3 | - | The text scanning primitive used by WORD. From the text address addr1 and an ascii delimiting character c, is determined the byte offset to the first non-delimiter character n1, the offset to the first delimiter after the text n2, and the offset to the first character not included. This procedure will not process past an ascii 'null', treating it as an unconditional delimiter. |

| Word | Vocab | New Word | Type | Stack Diagram | Usage | Description |
|---|---|---|---|---|---|---|
| **ERROR** | FORTH | no | Console | line --- in blk | - | Execute error notification and restart of system. WARNING is first examined. If 1, the text of line n, relative to screen 4 of drive O is printed. This line number may be positive or negative, and beyond just screen 4. If WARNING=O, n is just printed as a message number (non disc installation). If WARNING is -l, the definition (ABORT) is executed, which executes the system ABORT. The user may cautiously modify this execution by altering (ABORT). fig-FORTH saves the contents of IN and BLK to assist in determining the location of the error. Final action is execution of QUIT |
| **ERRS** | FORTH | yes | Console | --- addr | - | Returns the address of the variable where a count of serial I/O communication errors are saved |
| **EXPECT** | FORTH | no | Console | addr count --- | - | Transfer characters from the terminal to address, until a "return" or the count of characters have been received. One or more nulls are added at the end of the text |
| **FLD** | FORTH | no | Console | --- addr | - | A user variable for control of number output field width. Presently unused in fig-FORTH. |
| **HEX** | FORTH | no | Console | --- | - | Set the numeric conversion base to sixteen (hexadecimal). |
| **HLD** | FORTH | no | Console | --- addr | - | A user variable that holds the address of the latest character of text during numeric output conversion. |
| **HOLD** | FORTH | no | Console | c --- | - | Used between <# and #> to insert an ascii character into a pictured numeric output string. e.g. 2E HOLD will place a decimal point. |
| **ID.** | FORTH | no | Console | addr -- | - | Print a definition's name from its name field address |
| **IH>** | FORTH | yes | Console | addr count --- | - | outputs n bytes of memory data in Intel Hex format from memory starting at addr. |
| **IN** | FORTH | no | Console | --- addr | - | A user variable containing the byte offset within the current input text buffer (terminal or disc) from which the next text will be accepted. WORD uses and moves the value of IN. |
| **KEY** | FORTH | no | Console | --- c | - | Leave the ascii value of the next terminal key struck |
| **MESSAGE** | FORTH | no | Console | n -- | - | Print on the selected output device the text of line n relative to screen 4 of drive O. n may be positive or negative. MESSAGE may be used to print incidental text such as report headers. If WARNING is zero, the message will simply be printed as a number (disc unavailable) |
| **NEXTBYTE** | FORTH | yes | Console | --- | - | reads in two ASCII Hex character and converts to binary byte |
| **NUMBER** | FORTH | no | Console | addr --- d | - | Convert a character string left at addr with a preceeding count, to a signed .double number, using the current numeric base. If a decimal point is encountered in the text, its position will be given in DPL, but no other effect occurs. If numeric conversion is not possible, an error message will be given |
| **OUT** | FORTH | no | Console | --- addr | - | A user variable that contains a value incremented by EMIT. The user may alter and examine OUT to control display formatting |
| **PAD** | FORTH | no | Console | --- addr | - | Leave the address of the text output buffer, which is a fixed offset above HERE |
| **QKEY** | FORTH | yes | Console | --- ch | - | Used by internal intel hex loader to get the next character from console and abort if it's an ESC ($1B) |
| **QUERY** | FORTH | no | Console | --- | - | Input 80 characters of text (or until a "return") from the operators terminal. Text is positioned at the address contained in TIB with IN set to zero. |
| **R#** | FORTH | no | Console | -- addr | - | A user variable which may contain the location of an editing cursor, or other file related function |
| **SIGN** | FORTH | no | Console | n d --- d | - | Stores an ascii "-" sign just before a converted numeric output string in the text output buffer when n is negative. n is discarded but double number d is maintained. Must be used between <# and #>. |
| **SPACE** | FORTH | no | Console | --- | - | Transmit an ascii blank to the output device. |
| **SPACES** | FORTH | no | Console | n --- | - | Transmit n ascii blanks to the output device. |
| **TIB** | FORTH | no | Console | --- addr | - | A user variable containing the address of the terminal input buffer. |
| **-TRAILING** | FORTH | no | Console | addr n1 --- addr n2 | - | Adjusts the character count n1 of a text string beginning address to suppress the output of trailing blanks. i.e. the characters at addr+n1 to addr+n2 are blanks. |
| **TYPE** | FORTH | no | Console | addr count --- | - | Transmit count characters from addr to the selected output device |
| **U.** | FORTH | no | Console | u -- | - | Output the value on top of stack as an unsigned number |

| Word | Vocab | New Word | Type | Stack Diagram | Usage | Description |
|---|---|---|---|---|---|---|
| **VLIST** | FORTH | no | Console | --- | - | List the names of the definitions in the context vocabulary. "Break" will terminate the listing. |
| **WARNING** | FORTH | no | Console | --- addr | See MESSAGE, ERROR | A user variable containing a value controlling messages. If = 1 disc is present, and screen 4 of drive 0 is the base location for messages. If = 0, no disc is present and messages will be presented by number. If = -1, execute (ABORT) for a user specified procedure. |
| **WORD** | FORTH | no | Console | c --- | - | Read the next text characters from the input stream being interpreted, until a delimiter c is found, storing the packed character string beginning at the dictionary buffer HERE. WORD leaves the character count in the first byte, the characters, and ends with two or more blanks. Leading occurrences of c are ignored. If BLK is zero text is taken from the terminal input buffer, otherwise from the disc block stored in BLK. |
| **X.** | FORTH | no | Console | n --- | - | Outputs top of stack formated in hexidecimal |
| **'** | FORTH | no | Dictionary | --- addr | ' nnnn | Leaves the parameter field address of dictionary word nnnn. As a compiler directive, executes in a colon-definition to compile the address as a literal. If the word is not found after a search of CONTEXT and CURRENT, an appropriate error message is given. Pronounced "tick". |
| **FENCE** | FORTH | no | DIctionary | --- addr | - | A user variable containing an address below which FORGETting is trapped. To forget below this point the user must alter the contents of FENCE |
| **-FIND** | FORTH | no | Dictionary | --- pfa b tf (found)<br>--- ff (not found) | - | Accepts the next text word (delimited by blanks) in the input stream to HERE, and searches the CONTEXT and then CURRENT vocabularies for a matching entry. If found, the dictionary entry's parameter field address, its length byte, and a boolean true is left. Oherwise, only a boolean false is left |
| **FORGET** | FORTH | no | Dictionary | --- | FORGET cccc | Deletes definition named cccc from the dictionary with all entries physically following it. In fig-FORTH, an error message will occur if the CURRENT and CONTEXT vocabularies are not currently the same |
| **FORTH** | FORTH | no | Dictionary | --- | | The name of the primary vocabulary. Execution makes FORTH the CONTEXT vocabulary. Until additional user vocabularies are defined, new user definitions become a part of FORTH. FORTH is immediate, so it will execute during the creation of a colon-definition, to select this vocabulary at compile time |
| **HERE** | FORTH | no | Dictionary | --- addr | - | Leave the address of the next available dictionary location |
| **B/BUF** | FORTH | no | Disk | --- n | - | This constant leaves the number of bytes per disc buffer, the byte count read from disc by BLOCK |
| **B/SCR** | FORTH | no | Disk | --- n | - | This constant leaves the number of blocks per editing screen. By convention, an editing screen is 1024 bytes organized as 16 lines of 64 characters each. |
| **BLK** | FORTH | no | Disk | --- addr | - | A user variable containing the block number being interpreted. If zero, input is being taken from the terminal input buffer. |
| **BLOCK** | FORTH | no | Disk | n --- addr | See also BUFFER, R/W UPDATE, FLUSH | Leave the memory address of the block buffer containing block n. If the block is not already in memory, it is transferred from disc to which ever buffer Was least recently written. If the block occupying that buffer has been marked as updated, it is rewritten to disc before block n is read into the buffer. |
| **FIRST** | FORTH | no | Disk | --- n | - | A constant that leaves the address of the first (lowest) block buffer |
| **OFFSET** | FORTH | no | Disk | --- addr | - | A user variable which may contain a block offset to disc drives. The contents of OFFSET is added to the stack number by BLOCK. Messages by MESSAGE are independent of OFFSET. |
| **>SCR** | FORTH | yes | Editor | screen# --- | - | Receives incoming characters from the console to screen#. Input text files should be delimited by a ~ (tilde). Lines are padded to 64 characters with spaces if necessary. |
| **LIST** | FORTH | no | Editor | --- | - | Display the ascii text of screen n on the selected output device. SCR contains the screen number during and after this process. |
| **LOAD** | FORTH | no | Editor | n --- | See ;S and --> | Begin interpretation of screen n. Loading will terminate at the end of the screen or at ;S |
| **SCR** | FORTH | no | Editor | --- addr | - | A user variable containing the screen number most recently reference by LIST |
| **SCR>** | FORTH | yes | Editor | n screen# --- | - | Transmits n screens, starting at screen#, to the console. Lines are truncated at the last non-space character and a CR inserted. |

| Word | Vocab | New Word | Type | Stack Diagram | Usage | Description |
|---|---|---|---|---|---|---|
| **+LOOP** | FORTH | no | Execution Flow | n1 --- (run)<br>addr n2 --- (compile) | DO ... n1 +LOOP | At run-time, +LOOP selectively controls branching back to the corresponding DO based on n1, the loop index and the loop limit. The signed increment n1 is added to the index and the total compared to the limit. The branch back to DO occurs until the new index is equal to or greater than the limit (n1>0), or until the new index is equal to or less than the limit (n1<0). Upon exiting the loop, the parameters are discarded and execution continues ahead.<br>At compile time, +LOOP compiles the run-time word (+LOOP) and the branch offset computed from HERE to the address left on the stack by DO. n2 is used for compile tine error checking. |
| **0BRANCH** | FORTH | no | Execution Flow | f --- | - | The run-time procedure to conditionally branch. If f is false (zero), the following in-line parameter is added to the interpretive pointer to branch ahead or back. Compiled by IF, UNTIL, and WHILE. |
| **ABORT** | FORTH | no | Execution Flow | --- | - | Clear the stacks and enter the execution state. Return control to the operators terminal, printing a message appropriate to the installation. |
| **AGAIN** | FORTH | no | Execution Flow | addr n --- (compiling) | BEGIN ... AGAIN | At run-time, AGAIN forces execution to return to corresponding BEGIN. There is no effect on the stack. Execution cannot leave this loop (unless R> DROP is executed one level below).<br>At compile time, AGAIN compiles BRANCH with an offset from HERE to addr. n is used for compile-time error checking. |
| **BRANCH** | FORTH | no | Execution Flow | --- | - | The run-time procedure to unconditionally branch. An in-line offset is added to the interpretive pointer IP to branch ahead or back. BRANCH is compiled by ELSE, AGAIN, REPEAT. |
| **BYE** | FORTH | no | Execution Flow | --- | - | Exits FORTH to an installation dependent address |
| **COLD** | FORTH | no | Execution Flow | --- | - | The cold start procedure to adjust the dictionary pointer to the minimum standard and restart via ABORT. May be called from the terminal to remove application programs and restart. |
| **DO** | FORTH | no | Execution Flow | n1 n2 --- (execute)<br>addr n --- (compile) | DO ... LOOP | At run time, DO begins a sequence with repetitive execution controlled by a loop limit n1 and an index with initial value n2. DO removes these from the stack. Upon reaching LOOP the index is incremented by one. Until the new index equals or exceeds the limit, execution loops back to just after DO; otherwise the loop parameters are discarded and execution continues ahead. Both n1 and n2 are determined at run-time and may be the result of other operations.  Within a loop 'I' will copy the current value of the index to the stack. See I, LOOP, +LOOP, LEAVE.<br>When compiling within the colon definition, DO compiles (DO), leaves the following address addr and n for later error checking. |
| **ELSE** | FORTH | no | Execution Flow | addr1 n1 --- addr2 n2 | IF ... ELSE ... ENDIF | At run-time, ELSE executes after the true part following IF. ELSE forces execution to skip over the following false part and resumes execution after the ENDIF. It has no stack effect.<br>At compile-time ELSE emplaces BRANCH reserving a branch offset, leaves the address addr2 and n2 for error testing. ELSE also resolves the pending forward branch from IF by calculating the offset from addr1 to HERE and storing at addr1. |
| **END** | FORTH | no | Execution Flow | --- | - | This is an 'alias' or duplicate definition for UNTIL. |
| **ENDIF** | FORTH | no | Execution Flow | addr n --- | IF ... ENDIF<br>IF ... ELSE ... ENDIF | At run-time, ENDIF serves only as the destination of a forward branch from IF or ELSE. It marks the conclusion of the conditional structure. THEN is another name for ENDIF. Both names are supported in fig-FORTH. See also IF and ELSE.<br>At compile-time, ENDIF computes the forward branch offset from addr to HERE and stores it at addr. n is used for error tests. |
| **EXECUTE** | FORTH | no | Execution Flow | addr --- | - | Execute the definition whose code field address is on the stack. The code field address is also called the compilation address. |
| **GO** | FORTH | yes | Execution Flow | --- addr | - | Transfers execution control to addr |

| Word | Vocab | New Word | Type | Stack Diagram | Usage | Description |
|---|---|---|---|---|---|---|
| **IF** | FORTH | no | Execution Flow | --- addr n (run-time)<br>--- P,C2,L0 (compile) | IF (tp) ... ENDIF<br>IF (tp) ... ELSE (fp) ... ENDIF | At run-time, IF selects execution based on a boolean flag. If f is true (non-zero), execution continues ahead thru the true part. If f is false (zero), execution skips till just after ELSE to execute the false part. After either part, execution resumes after ENDIF. ELSE and its false part are optional.; if missing, false execution skips to just after ENDIF<br>At compile-time IF compiles 0BRANCH and reserves space for an offset at addr. addr and n are used later for resolution of the offset and error testing. |
| **LEAVE** | FORTH | no | Execution Flow | --- | - | Force termination of a DO-LOOP at the next opportunity by setting the loop limit equal to the current value of the index. The index itself remains unchanged, and execution proceeds normally until LOOP or +LOOP is encountered |
| **LFA** | FORTH | no | Execution Flow | pfa --- lfa | - | Convert the parameter field address of a dictionary definition to its link field address |
| **LOOP** | FORTH | no | Execution Flow | addr n --- (compiling) | DO ... LOOP | At run-time, LOOP selectively controls branching back to the corresponding DO based on the loop index and limit. The loop index is incremented by one and compared to the limit. The branch back to DO occurs until the index equals or exceeds the limit; at that time, the parameters are discarded and execution continues ahead.<br>At compile-time. LOOP compiles (LOOP) and uses addr to calculate an offset to DO. n is used for error testing. |
| **NEXT** | FORTH | no | Execution Flow | --- |  | This is the inner interpreter that uses the interpretive pointer IP to execute compiled Forth definitions. It is not directly executed but is ff the return point for all code procedures. It acts by fetching the address pointed by IP, storing this value in register W. It then jumps to the address pointed to by the address pointed to by W. W points to the code field of a definition which contains the address of the code which executes for that definition. This usage of indirect threaded code is a major contributor to the power, portability, and extensibility of Forth. Locations of IP and W are computer specific |
| **QUIT** | FORTH | no | Execution Flow | --- | - | clear the return stack, stop compilation, and return control to the operators terminal. No message is given |
| **REPEAT** | FORTH | no | Execution Flow | addr n --- (compiling) | BEGIN ... WHILE ... REPEAT | At run-time, REPEAT forces an unconditional branch back to just after the corresponding BEGIN.<br>At compile-time, REPEAT compiles BRANCH and the offset from HERE to addr. n is used for error testing. |
| **THEN** | FORTH | no | Execution Flow | --- | - | exAn alias for ENDIF |
| **UNTIL** | FORTH | no | Execution Flow | f --- (run-time)<br>addr n --- (compile) | BEGIN ... UNTIL | At run-time, UNTIL controls the conditional branch back to the corresponding BEGIN. If f is false, execution returns to just after BEGIN, if true, execution continues ahead.<br>At compile-time, UNTIL compiles (0BRANCH) and an offset from HERE to addr. n is used for error tests. |
| **WARM** | FORTH | no | Execution Flow | --- | - | Warm starts the system while maintaining words added to the base dictionary |
| **WHILE** | FORTH | no | Execution Flow | f --- (run-time)<br>ad1 nl --- ad1 n1 ad2 n2 (compile) | BEGIN ... WHILE ... REPEAT | At run-time, WHILE selects conditional execution based on Boolean flag f. If f is true (non-zero), WHILE continues execution of the true part thru to REPEAT, which then branches back to BEGIN. If f is false (zero), execution skips to just after REPEAT, exiting the structure<br>At compile time, WHILE emplaces (0BRANCH) and leaves ad2 of the reserved offset. The stack values will be resolved by REPEAT. |
| **-** | FORTH | no | Math | n1 n2 --- diff | - | Leave 16 bit the difference of n1-n2 |
| ***** | FORTH | no | Math | n1 n2 --- prod | - | Leave the signed product of two signed 16 bit numbers |
| ***/** | FORTH | no | Math | n1 n2 n3 --- n4 | - | Leave the ratio n4 = n1*n2/n3 where all are signed numbers. Retention of an intermediate 31 bit product permits greater accuracy than would. be available with the sequence:<br>n1 n2 * n3 / |
| ***/MOD** | FORTH | no | Math | --- | - | Leave the quotient n5 and remainder n4 of the operation :<br>n1*n2/n3<br>A 31 bit intermediate product is used as for */ |
| **/** | FORTH | no | Math | n1 n2 --- quot | - | Leave the signed quotient of n1/n2 |

| Word | Vocab | New Word | Type | Stack Diagram | Usage | Description |
|---|---|---|---|---|---|---|
| /MOD | FORTH | no | Math | n1 n2 --- rem quot | - | Leave the remainder and signed quotient of n1/n2. The remainder has the sign of the dividend |
| + | FORTH | no | Math | n1 n2 --- sum | - | Leave the 16 bit sum of n1+n2 |
| +- | FORTH | no | Math | n1 n2 --- n3 | - | Apply the sign of n2 to n1, which is left as n3 |
| +! | FORTH | no | Math | n addr --- | - | Add n to the value at the address. Pronounced "plus-store". |
| < | FORTH | no | Math | n1 n2 --- f | - | Leave a true flag if n1 is less than n2; otherwise leave a false flag. |
| = | FORTH | no | Math | n1 n2 --- f | - | Leave a true flag if n1=n2 ; otherwise leave a false flag |
| > | FORTH | no | Math | n1 n2 --- f | - | Leave a true flag if n1 is greater than n2 ; otherwise a false flag |
| 0 | FORTH | no | Math | --- 0 | - | puts a zero on top of stack - used so often that is attractive to define as a constant instead of a literal |
| 0< | FORTH | no | Math | n --- f | - | Leave a true flag if the number is less than zero (negative), otherwise leave a false flag. |
| 0= | FORTH | no | Math | n --- f | - | Leave a true flag is the number is equal to zero, otherwise leave a false flag |
| 1 | FORTH | no | Math | --- 1 | - | puts a one on top of stack -- used so often that is attractive to define as a constant instead of a literal |
| 1+ | FORTH | no | Math | n --- n+1 | - | Increment n by 1 |
| 2 | FORTH | no | Math | --- 2 | - | puts a two on top of stack -- used so often that is attractive to define as a constant instead of a literal |
| 2+ | FORTH | no | Math | n --- n+2 | - | Increment n by 2 |
| 3 | FORTH | yes | Math | --- 3 | - | puts a three on top of stack - used so often that is attractive to define as a constant instead of a literal |
| ABS | FORTH | no | Math | n --- u | - | Leave the absolute value of n as u. |
| AND | FORTH | no | Math | n1 n2 --- n2 | - | Leave the bitwise logical and of n1 and n2 as n3. |
| CURRENT | FORTH | no | Math | d1 d2 --- dsum | - | Leave the double number sum of two double numbers. |
| D+ | FORTH | no | Math | d1 d2 --- dsum | - | Leave the double number sum of two double numbers |
| D+- | FORTH | no | Math | d1 n --- d2 | - | Apply the sign of n to the double number d1, leaving it as d2 |
| DABS | FORTH | no | Math | d --- ud | - | Leave the absolute value ud of a double number |
| DMINUS | FORTH | no | Math | d1 --- d2 | - | Convert d1 to its double number two's complement |
| I | FORTH | no | Math | --- n | See R | Used within a DO-LOOP to copy the loop index to the stack. |
| J | FORTH | yes | Math | --- | - | Used within nested DO-LOOPs to copy the loop index of the outer loop to the stack. |
| LIT | FORTH | no | Math | --- n | - | Within a colon-definition, LIT is automatically compiled before each 16 bit literal number encountered in input text. Later execution of LIT causes the contents of the next dictionary address to be pushed to the stack |
| M* | FORTH | no | Math | n1 n2 --- d | - | A mixed magnitude math operation which leaves the double number signed product of two signed number. |
| M/ | FORTH | no | Math | d n1 --- n2 n3 | - | A mixed magnitude math operator which leaves the signed remainder n2 and signed quotient n3 from a double number dividend and divisor n1. The remainder takes its sign from the dividend. |
| M/MOD | FORTH | no | Math | ud1 u2 --- u3 ud4 | - | An unsigned mixed magnitude math operation which leaves a double quotient ud4 and remainder u3, from a double dividend ud1 and single divisor u2 |
| MAX | FORTH | no | Math | n1 n2 --- max | - | Leave the greater of two numbers |
| MIN | FORTH | no | Math | n1 n2 --- min | - | Leave the smaller of two numbers. |
| MINUS | FORTH | no | Math | n1 --- n2 | - | Leave the two's complement of a number |
| MOD | FORTH | no | Math | n1 n2 --- mod | - | Leave the remainder of n1/n2, with the same sign as n1 |
| OR | FORTH | no | Math | n1 n2 -- or | - | Leave the bit-wise logical or of two l6 bit values |
| S->D | FORTH | no | Math | n --- d | - | Sign extend a single number to form a double number |
| TOGGLE | FORTH | no | Math | addr b -- | - | Complement the contents of addr by the bit pattern b |
| U* | FORTH | no | Math | u1 u2 --- ud | - | Leave the unsigned double number product of two unsigned numbers |
| U/ | FORTH | no | Math | ud u1 --- u2 u3 | - | Leave the unsigned remainder u2 and unsigned quotient u3 from the unsigned double dividend ud and unsigned divisor u1. |

| Word | Vocab | New Word | Type | Stack Diagram | Usage | Description |
|---|---|---|---|---|---|---|
| **XOR** | Forth | no | Math | n1 n2 --- xor | - | Leave the bitwise logical exclusive or of two values |
| **!** | FORTH | no | Memory | n addr --- | - | Store 16 bits of n at address. Pronounced "store". |
| **,** | FORTH | no | Memory | n -- | - | Store n into the next available dictionary memory cell, advancing the dictionary pointer. (pronounced comma) |
| **+ORIGIN** | FORTH | no | Memory | n --- addr | - | Leave the memory address relative by n to the origin parameter area. n is the minimum address unit, either byte or word. This definition is used to access or modify the boot-up parameters at the origin area. |
| **ALLOT** | FORTH | no | Memory | n --- | - | Add the signed number to the dictionary pointer DP. May be used to reserve dictionary space or re-origin memory. n is with regard to computer address type (byte or word) |
| **C!** | FORTH | no | Memory | b addr --- | - | Store 8 bits at address. |
| **C,** | FORTH | no | Memory | b --- | - | Store 8 bits of b into the next available dictionary byte, advancing the dictionary pointer. |
| **C@** | FORTH | no | Memory | addr --- b | - | Leave the 8 bit contents of memory address. |
| **CMOVE** | FORTH | no | Memory | from to count -- | - | Move the specified quantity of bytes beginning at address from to address to. |
| **DP** | FORTH | no | Memory | ---- addr | - | A user variable, the dictionary pointer, which contains the address of the next free memory above the dictionary. The value may be read by HERE and altered by ALLOT |
| **ERASE** | FORTH | no | Memory | addr n -- | - | Clear a region of memory to zero from addr over n addresses |
| **FILL** | FORTH | no | Memory | addr quan b - | - | Fill memory at the address with the specified quantity of bytes b. |
| **LIMIT** | FORTH | no | Memory | ---- n | - | A constant leaving the address just above the highest memory available for a (disc) buffer. Usually this is the highest system memory |
| **HALT** | FORTH | yes | Multitasking | --- | n HALT | Cause task n to stopt executing and go dormant |
| **PAUSE** | FORTH | yes | Multitasking | --- | - | used by a task to cooperatively release execution control for one pass of the multitasker |
| **RUN** | FORTH | yes | Multitasking | task# --- | - | Sets the state of task# to cause it to run when the mulittasker next gets to it |
| **START** | FORTH | yes | Multitasking | n --- | n START taskname | Finds taskname in current dictionary and adds to the task control table as task n |
| **TASK#** | FORTH | yes | Multitasking | --- n | - | returns the current task's number (to itself) |
| **TIC** | FORTH | yes | Multitasking | tics --- | - | pauses current task for tics intervals (actual time depends on system tic timer interrupt value) |
| **(.")** | FORTH | no | Run-time | --- | see ."" | The run-time procedure, compiled by ." which transmits the following in-line text to the selected output device. |
| **(;CODE)** | FORTH | no | Run-time | --- | see ;CODE | The run-time procedure, compiled by ;CODE, that rewrites the code field of the most recently defined word to point to the following machine code sequence. |
| **(+LOOP)** | FORTH | no | Run-time | --- | see +LOOP | The run-time procedure compiled by +LOOP, which increments the loop index by n and tests for loop completion. |
| **(ABORT)** | FORTH | no | Run-time | --- | - | Executes after an error when WARNING is -1. This word normally executes ABORT, but may be altered (with care) to a user's alternative procedure. |
| **(DO)** | FORTH | no | Run-time | --- | see DO | The run-time procedure compiled by DO which moves the loop control parameters to the return stack. |
| **(FIND)** | FORTH | no | Run-time | addr1 addr2 --- pfa b tf (ok)<br>addr1 addr2 --- ff (bad) | - | Searches the dictionary starting at the name field address addr2, matching to the text at addr1. Returns parameter field address, length byte of name field and boolean true for a good match. If no match is found, only a boolean false is left. |
| **(LINE)** | FORTH | no | Run-time | n1 n2 --- addr count | - | Convert the line number n1 and the screen n2 to the disc buffer address containing the data. A count of 64 indicates the full line text length. |
| **(LOOP)** | FORTH | no | Run-time | --- | - | The run-time procedure compiled by LOOP which increments the loop index and tests for loop completion |
| **@** | FORTH | no | Stack | addr --- n | - | Leave the 16 bit contents of address |
| **>R** | FORTH | no | Stack | n --- | - | Remove a number from the computation stack and place as the most accessible on the return stack. Use should be balanced with R> in the same definition. |
| **2DUP** | FORTH | yes | Stack | n1 n2 --- n1 n2 n1 n2 | - | duplicates top two number on the stack |
| **DROP** | FORTH | no | Stack | n --- | - | Drop the number from the stack |
| **DUP** | FORTH | no | Stack | n --- n n | - | Duplicate the value on the stack |

| Word | Vocab | New Word | Type | Stack Diagram | Usage | Description |
|---|---|---|---|---|---|---|
| **-DUP** | FORTH | no | Stack | n1 -- n1 (if zero)<br>n1 -- n1 n1 (non-zero) | - | Reproduce n1 only if it is non-zero. This is usually used to copy a value just before IF, to eliminate the need for an ELSE part to drop it. |
| **NIP** | FORTH | yes | Stack | n1 n2 --- n2 | - | Removes the number second from the top of stack. |
| **OVER** | FORTH | no | Stack | n1 n2 --- n1 n2 n1 | - | copy the second stack value, placing it as the new top |
| **PICK** | FORTH | yes | Stack | index --- n | - | Places the value of stack element index onto the top of stack |
| **R** | FORTH | no | Stack | --- n | - | Copy the top of the return stack to the computation stack. |
| **R>** | FORTH | no | Stack | --- n | See >R and R. | Remove the top value from the return stack and leave it on the computation stack. See >R and R. |
| **R0** | FORTH | no | Stack | --- addr | See >R and R | A user variable containing the initial location of the return stack. Pronounced R-zero. |
| **ROT** | FORTH | no | Stack | n1 n2 n3 --- n2 n3 n1 | - | Rotate the top three values on the stack, bringing the third to the top. |
| **-ROT** | FORTH | yes | Stack | n1 n2 n3 --- n3 n1 n2 | - | Counter rotate the top three values on the stack, bringing the second to the top |
| **RP!** | FORTH | no | Stack | --- | - | Initialize the return stack pointer from user variable R0. |
| **S0** | FORTH | no | Stack | --- addr | See SP! | A user variable that contains the initial value for the stack pointer. |
| **SP!** | FORTH | no | Stack | --- | - | initialize the stack pointer from SO. |
| **SP@** | FORTH | no | Stack | --- addr | - | Return the address of the stack position to the top of the stack, as it was before SP@ was executed |
| **SWAP** | FORTH | no | Stack | n1 n2 --- n2 n1 | - | Exchange the top two values On the stack. |
| **TUCK** | FORTH | yes | Stack | n1 n2 --- n2 n1 n 2 | - | copies top of stack and inserts as third item on stack |
| **!CSP** | FORTH | no | System | --- | - | Save the stack position in CSP. Used as part of the compiler security. |
| **LATEST** | FORTH | no | Vocabulary | --- addr | - | Leave the name field address of the topmost word in the CURRENT vocabulary |
| **ORIGIN** | FORTH | no | | --- addr | - | Constant value that returns the base address of USER variables |
| **#LAG** | EDITOR | Yes | Editor | --- | | editor internal use |
| **#LEAD** | EDITOR | Yes | Editor | --- | | editor internal use |
| **#LOCATE** | EDITOR | Yes | Editor | --- | | editor internal use |
| **1LINE** | EDITOR | Yes | Editor | --- | | editor internal use |
| **B** | EDITOR | Yes | Editor | --- | | back up over text found by F |
| **b** | EDITOR | Yes | Editor | --- | | editor internal use |
| **C** | EDITOR | Yes | Editor | --- | C text | spead and copy in text at cursor |
| **CLEAR** | EDITOR | Yes | Editor | n --- | | clear screen n to blanks |
| **COPY** | EDITOR | Yes | Editor | n1 n2 --- | | copy screen n1 to n2 |
| **D** | EDITOR | Yes | Editor | n --- | | delete line n to PAD ( i.e. cut) |
| **DELETE** | EDITOR | Yes | Editor | n --- | | deleted n characters before the current cursor position |
| **E** | EDITOR | Yes | Editor | n --- | | erase line n with blanks |
| **F** | EDITOR | Yes | Editor | --- | F text | move text to PAD and search forward |
| **f** | EDITOR | Yes | Editor | --- | | editor internal use |
| **find** | EDITOR | Yes | Editor | --- | | editor internal use |
| **H** | EDITOR | Yes | Editor | n --- | | hold line N at PAD ( i.e. copy) |
| **I** | EDITOR | Yes | Editor | n --- | | insert text from PAD at line n |
| **L** | EDITOR | Yes | Editor | --- | | lists the current screen |
| **LINE** | EDITOR | Yes | Editor | --- | | editor internal use |
| **M** | EDITOR | Yes | Editor | n --- | | moves the cursos by the signed about n |
| **MATCH** | EDITOR | Yes | Editor | --- | | editor internal use |
| **-MOVE** | EDITOR | Yes | Editor | --- | | editor internal use |
| **N** | EDITOR | Yes | Editor | --- | | find next occurance of text moved to PAD by F |
| **P** | EDITOR | Yes | Editor | n --- | P text | overwrite line n with text |
| **R** | EDITOR | Yes | Editor | --- | | editor internal use |
| **S** | EDITOR | Yes | Editor | n --- | | spread at line n, moving subsequent lines down |
| **T** | EDITOR | Yes | Editor | n --- | | type line n and save in PAD |
| **TEXT** | EDITOR | Yes | Editor | --- | | editor internal use |

| Word | Vocab | New Word | Type | Stack Diagram | Usage | Description |
|---|---|---|---|---|---|---|
| TILL | EDITOR | Yes | Editor | --- | TILL text | deleted text from cursos to text |
| TOP | EDITOR | Yes | Editor | --- |  | positions the cursor at the top  of the current screen |
| X | EDITOR | Yes | Editor | --- | X text | find and delete text |
| LIST | EDITOR | No | Forth | n -- |  | lists screen n on the console |
| ?FAULT | ASSEMBLER | yes | 1802 Assembler | addr1 addr2 --- |  | used by compiler to detect off page short branches |
| AGAIN, | ASSEMBLER | yes | 1802 Assembler | --- | BEGIN,  code  AGAIN, | inserts a BR instruction back to where the IF, statement was |
| DF | ASSEMBLER | yes | 1802 Assembler | --- |  | inserts a BNF instruction |
| EF1 | ASSEMBLER | yes | 1802 Assembler | --- |  | inserts a BN1 instruction |
| EF2 | ASSEMBLER | yes | 1802 Assembler | --- |  | inserts a BN2 instruction |
| EF3 | ASSEMBLER | yes | 1802 Assembler | --- |  | inserts a BN3 instruction |
| EF4 | ASSEMBLER | yes | 1802 Assembler | --- |  | inserts a BN4 instruction |
| ELSE, | ASSEMBLER | yes | 1802 Assembler | --- |  | modifies the previous IF, instructions false code target address to current location |
| ENDIF, | ASSEMBLER | yes | 1802 Assembler | --- |  | modifies the previous IF, instructions true code target address to current location |
| IF, | ASSEMBLER | yes | 1802 Assembler | --- | test IF,  code ELSE,  code ENDIF, | 1802 op code |
| NEXT | ASSEMBLER | yes | 1802 Assembler | --- |  | 1802 op code |
| NOT | ASSEMBLER | yes | 1802 Assembler | --- |  | converts previous branch if false instruction to branch if true |
| Q | ASSEMBLER | yes | 1802 Assembler | --- |  | inserts a BNQ instruction |
| UNTIL, | ASSEMBLER | yes | 1802 Assembler | --- | BEGIN  code test UNTIL, | inserts a branch address back to BEGIN, for previous conditional instruction |
| WHILE, | ASSEMBLER | yes | 1802 Assembler | --- | BEGIN test WHILE,  code  REPEAT, | inserts a branch address for following UNTIL, |
| Z | ASSEMBLER | yes | 1802 Assembler | --- |  | 1802 op code |
| ADC, | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- |  | 1802 op code |
| ADCI, | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- |  | 1802 op code |
| ADD, | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- |  | 1802 op code |
| ADI, | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- |  | 1802 op code |
| AND, | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- |  | 1802 op code |
| ANI, | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- |  | 1802 op code |
| BEGIN, | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- |  | 1802 op code |
| BR, | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- |  | 1802 op code |
| DEC, | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- |  | 1802 op code |
| DIS, | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- |  | 1802 op code |
| GHI, | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- |  | 1802 op code |
| GLO, | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- |  | 1802 op code |
| IDL, | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- |  | 1802 op code |
| INC, | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- |  | 1802 op code |
| INP, | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- |  | 1802 op code |
| IRX, | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- |  | 1802 op code |
| LBR, | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- |  | 1802 op code |
| LDA, | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- |  | 1802 op code |
| LDI, | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- |  | 1802 op code |
| LDN, | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- |  | 1802 op code |
| LDX, | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- |  | 1802 op code |
| LDXA, | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- |  | 1802 op code |
| MARK, | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- |  | 1802 op code |
| NOP, | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- |  | 1802 op code |
| OR, | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- |  | 1802 op code |
| ORI, | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- |  | 1802 op code |
| OUT, | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- |  | 1802 op code |
| PHI, | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- |  | 1802 op code |
| PLO, | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- |  | 1802 op code |

| Word | Vocab | New Word | Type | Stack Diagram | Usage | Description |
|---|---|---|---|---|---|---|
| **REQ,** | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- | | 1802 op code |
| **RET,** | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- | | 1802 op code |
| **SAV,** | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- | | 1802 op code |
| **SD,** | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- | | 1802 op code |
| **SDB,** | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- | | 1802 op code |
| **SDBI,** | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- | | 1802 op code |
| **SDI,** | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- | | 1802 op code |
| **SEP,** | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- | | 1802 op code |
| **SEQ,** | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- | | 1802 op code |
| **SEX,** | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- | | 1802 op code |
| **SHL,** | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- | | 1802 op code |
| **SHLC,** | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- | | 1802 op code |
| **SHR,** | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- | | 1802 op code |
| **SHRC,** | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- | | 1802 op code |
| **SM,** | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- | | 1802 op code |
| **SMB,** | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- | | 1802 op code |
| **SMBI,** | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- | | 1802 op code |
| **SMI,** | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- | | 1802 op code |
| **STR,** | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- | | 1802 op code |
| **STXD,** | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- | | 1802 op code |
| **XOR,** | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- | | 1802 op code |
| **XRI,** | ASSEMBLER | yes | 1802 Assembler Mnemonic | --- | | 1802 op code |
| **CODE** | ASSEMBLER | no | Compiler | --- | | 1802 op code |
| **REPEAT,** | ASSEMBLER | yes | Execution control | --- | | inserts a BR instruction back to where the BEGIN, statement was |