

## 11.1. THE OPERATING SYSTEM OF FORTH

## 11.2. ENTERING THE TEXT INTERPRETER

: COLD ( --- )	High level cold start.
BOOT	Execute a user defined bootup definition.
QUIT	Jump to QUIT, a normal re-start point.

;

DEFER BOOT

Vectored to a user-selected initializing routine. The default boot routine is ABORT.

```

: QUIT      ( --- )      The main Forth loop.  Get more input from the console
                        terminal and interpret it.  Respond with "ok" if every thing is
                        well.

      SP0 @ 'TIB !      Initialize the terminal input buffer to address just above the
                        data stack.

      BLK OFF      Store a zero in BLK.  Force the interpreter to process input
                        from the console terminal.

      [COMPILE] [      Store a zero in the variable STATE, forcing the system into
                        the interpretive mode.

      BEGIN      Enter the main Forth loop.
        RP0 @ RP!      Initialize the return stack pointer.
        STATUS      Indicate status of the system. A deferred word normally
                        vectored to CR, doing a carriage return.      QUERY
        Prompt the user to enter a line of commands on the console
        and place this command line in the terminal input buffer.      RUN
        Process the command line.
        STATE @ NOT      If STATE is zero, the system is n the interpretive mode.
      IF ." ok" THEN      Then print the ok message.
      AGAIN      The Forth loop is an infinite loop.  After one command line is
                        processed, it goes back to ask for another line.  It goes on
                        this way forever.

      ;

: RUN      ( --- )      An enhanced INTERPRET.  It allows for multiline
                        compilation, enabling you to enter a colon definition that
                        spans over several lines.

      STATE @ IF      If STATE is not zero, the system must be in the compiling
                        dictionary.
        STATE @ NOT      After compiling one line of source codes, test STATE again.
        IF INTERPRET THEN      If the system left the compiling mode, then interpret
                                the rest of the line.  Otherwise, exit.

      ELSE      The state is zero,
        INTERPRET      interpret the command line.
      THEN ;

```

### 11.3. INTERPRET

INTERPRET is a beautiful piece of code, a classic example of the simplicity and power of Forth language in describing complicated computational processes using high level words. It is worthy of our time to read the code and do our best to gain the fullest understanding of it. The definition of INTERPRET reads:

```

: INTERPRET      ( --- )      The Forth interpreter loop.  It parses out a word from the
                        input stream.  If the word is defined execute it, otherwise
                        convert it to a number and push it on the stack.      BEGIN
      Begin the interpret loop.

```

	?STACK	Check for stack underflow or overflow.	DEFINED
		Get the next word from the input stream and return its cfa	
		and a flag.	
	IF EXECUTE	If the word is defined, execute it using the cfa left on stack.	ELSE
NUMBER		Otherwise, convert it to a number.	
	DOUBLE?	Is it a double precision integer?	

```

NOT IF DROP THEN No. Only a single precision number. Drop the upper half
                  of the double number, preserving only the lower half single
                  integer.
THEN
FALSE            Put up a false flag for DONE?.
DONE?            Is it the end of line?
UNTIL            If we reach end of line here, exit the loop. Otherwise, loop
                  back to interpret the next word.
;

```

DEFINED is a very big word. It first parses a word out of the input stream and places it in the word buffer on the top of the dictionary. It then searches through the dictionary for a command with the same name. If a command is found, its code field address is placed on the data stack followed by a true flag. A valid code field address is then turned over to EXECUTE. EXECUTE executes this command by invoking the appropriate inner interpreter, which we had discussed in the chapter on inner interpreters. DEFINED is discussed in the chapter on vocabulary.

Now, if DEFINED failed to find a command with a matching name, control is passed to NUMBER, which converts the parsed word to a double precision number on the data stack. If a period was embedded in the number string, which causes DPL to differ from -1, the word DOUBLE? returns a true flag and the double number remain on the stack. Otherwise, the higher half of the double number is dropped from the stack and only a single precision number is left on the stack.

At the beginning of the loop, the data stack is checked for overflow or underflow by ?STACK. If the stack is ok, control falls into DEFINED to process the next word in the input stream. If a stack error condition is encountered, the system is forced into ABORT to start all over again. If an error condition is encountered during the number conversion process, an abort is also forced. These are the two conditions for abnormal exit from the INTERPRET loop.

#### 11.4. DONE? AND X

At the end of the INTERPRET loop, DONE? is executed to test the end-of-buffer condition. If it has reached the end of the input buffer, the loop would be terminated and the control would fall into the outer Forth loop in QUIT. Otherwise, the interpreter will loop back to parse and execute the next word in the input buffer.

A flow chart of this chain of activities might be helpful in visualizing the sequence of events when the Forth system is cranking in full steam, as shown in Fig. 12.1.

A number of loose ends have to be patched before we finish this chapter.

```

: ?STACK    ( --- )      Check for data stack underflow or overflow. Abort if any of
                        the error conditions occurred.
    SP@      Get the current data stack pointer.
    SP0 @ SWAP U<      If the stack underflowed,
    ABORT" Stack Underflow"  Abort.
    SP@ PAD U<      If the stack grows too close to the top of the dictionary,  ABORT"

```

Stack Overflow"  
;

Abort also.

Otherwise, return normally.

```

: DONE?      ( n --- f )      Return a true flag if the inputstream is exhausted or the
                                STATE doesn't match with the current state.
                                STATE @ <>      Is the state flag left on stack the same as that in STATE?      END?
@ OR          Or the end of line? Leave the or'ed flag on stack.
END? OFF      Turn off the end-of-buffer flag to let the interpreter get a new
                                line of command and start over.
;

```

In F83 systems before Version 2.0, the end-of-buffer condition is detected and the END? flag is set by a word with a null string as its name. This null word was defined using a pseudo name of X and later patched to null. When the input stream is exhausted, the last word parsed out by WORD is this null word and it tells the text interpreter to stop processing the input buffer. This technique had been used in most Forth systems including fig-Forth. In F83 Version 2.0 and later, the end-of-buffer condition is detected and the END? flag is set in the word FIND; therefore, this mysterious null word is eliminated and the text interpreter is in much better shape. The discussion on the null word X is included here for completeness and for users with older versions of F83.

```

: X          ( --- )          The null word to flag end-of-buffer and to terminate the
                                interpreter loop.
                                END? ON      Turn on the end-of-buffer flag in the user variable END?. ;

```

HEX A080 LAST @ ! IMMEDIATE DECIMAL

The real name of X in the Forth dictionary is a null string, with a character count of 0 and a blank character. This null string is returned by WORD to the word buffer when the end of the input stream is reached. The contents of the name field of this null word is A080 in hex, with the MSB's in both bytes set as name field delimiters. As we reach the end of the input stream, this null word is returned by WORD and executed. It turns on the END? flag and terminates the interpret loop. Explicitly terminating the interpret loop at the end of input stream makes the definition of INTERPRET comprehensible. Another advantage is that the end-of-buffer condition does not have to be artificially synthesized by appending a NUL character at the end of the input line from the console or at the end of every disk buffer (as done in figForth), which can be easily corrupted and causes the Forth system to behave erratically.