2.7 The Editor. As described above, FORTH has the capability of accepting its input from block storage. Such stored input commands are the normal means for "compiling" the standard FORTH system as well as application programs. The FORTH user may put FORTH commands, or any text material, into block storage by means of the FORTH editor.

In the Caltech versions of FORTH, the editor is run as an "application program"; i.e. it must be loaded explicitly before use. The constant EDIT is defined as the address of the block that contains the editor; thus the command EDIT LOAD is sufficient to obtain the editor.

The integer BLK is used to specify the FORTH block to be edited; thus to edit block 35, we type 35 BLK =.

A FORTH block is conventionally defined to have a fixed format with 16 lines each of 64 characters.* The 64th character of a line immediately precedes the first character of the next line.

The FORTH word T (defined in the editor) may be used to type any line. Thus 12 T causes the 12th line of block BLK to be typed.

A line may be deleted through the D word. Typing 6 D will cause line 6 to be deleted. All of the lines following line 6 are automatically moved up one line. The last line (line 20_8) either remains as before or is filled with blanks - depending on which version of FORTH is used.

*This format applies only to blocks which are to be used as text. Any block may also be used to hold binary data, in which case the user may choose any format.

New text may be entered into a block with the R (replace) or I (insert) words. The special words " or (are needed to enter the next text line into an internal buffer. Quote (") enters all of the text following up to the next quote into the buffer. Left parenthesis (() does the same except that the text is delimited by a right parenthesis ()). Thus

" THIS IS A TEXT STRING" and (THIS IS A TEXT STRING)

both place THIS IS A TEXT STRING into the buffer. If needed, blanks are added to the right to make 64 characters. Note that, like any words, " and (must have a blank following in the input. The text string to go into the buffer begins after this necessary blank. The delimiting character, however, needs no preceding blank.

So, to replace line 3 of block 1Ø with SAMPLE-TEXT we type:

1Ø BLK = " SAMPLE-TEXT" 3 R .

To have quotes in the text string we would need to use the (-) method of specifying the text.

To insert a line of text after a given line we use the I word. Thus" : NULL ;" 2 I inserts ": NULL ;" after line 2. The new line becomes line 3, and the former lines 3 to 17_8 become lines 4 to $2Ø_8$. The old line 2Ø is lost.

After a T or a D operation the line that was typed or deleted is automatically copied into the buffer, ready for a possible R or I. Thus the commands 3 D 2 I leave the text unaltered.

After an editing session the user must be careful to have the updated text actually rewritten into block storage. The

last two blocks edited will still be in the block buffers in main memory until the FLUSH word is executed. DISCARD will have the same effect; but in this case the editor vocabulary will be discarded in anticipation of loading a new application vocabulary.

The standard vocabulary contains two words to type text blocks. LIST will type out any text block with the line number at the right of each line; e.g. 5 LIST to list block 5. SHOW lists a sequence of block with a number to indicate each block and line; thus 5 7 SHOW lists blocks 5 through 7.

The word COPY copies one block into another. Either text or binary data is allowed. Thus 14 17 COPY copies block 14 into 17. The old block is unchanged. The buffers must be flushed after the last COPY in a sequence.

2.8 How to Operate a FORTH System. There is some natural variance between implementations of FORTH in the details of how one interacts with the system. This Section will deal with how a user can interact with a typical FORTH system. A description of how to initiate the system will be followed by a sample dialog.

Loading programs into minicomputers is often a tricky process, not obvious to new users. The more basic machines have no permanently stored instructions; thus when the computer is first started up it is necessary to enter the first program through the console switches. Manual program entry in this way is tedious and unreliable. In practice one will normally enter only a very brief program called a "bootstrap loader". This program may be 10 - 15 instructions long; its purpose is to read in another program from some mechanical device - paper tape or disk, for example. The bootstrap loader requires its input to have a very simple format. Usually the format is one word for address followed by a word of data.

The bootstrap format requires two words in the input medium for every word of program; also there is normally no error checking. For these reasons it is conventional to use the bootstrap loader only to load a program called "the binary loader" ("absolute loader" or "relocating loader", sometimes). The binary loader, in turn, will load useful programs like FORTH. The format required by the binary loader is much more efficient; typically only one address is required for a block of some hundred words. In addition there is usually a checksum test that will detect most errors in transmission of data.

In summary, to run a given program, one first loads a bootstrap loader, then uses the bootstrap to load the binary loader, and finally the binary loader to load the program of interest. This is the case for the most primitive minicomputers.

In the case of most of the Caltech computers, there is a permanently stored program (in read-only memory) that contains the bootstrap loader. Thus all that needs to be done is to start the computer running at the address of the bootstrap loader; the full binary loader will then be read in from paper tape or disk. For the purposes of FORTH, the binary loader is often combined with FORTH itself so that immediately after the loader is loaded, the FORTH system is loaded automatically.

Only the most fundamental part of the FORTH system is loaded by the process described above; this is the FORTH "object program". The object program has a very limited vocabulary, generally only large enough to allow itself to grow through CODE and : definitions. The rest of the "standard" FORTH system resides as text in block storage. To compile this into the dictionary one types FORTH LOAD. In a few seconds the dictionary expands to several hundred words including those described previously in Section 2. Applications programs such as interferometer control are compiled by further LOAD commands.

The following pages present a sample dialog between a user and the PDP-11 FORTH. The typing produced by FORTH is underlined. Text appearing between parentheses "(...)" is comment and is ignored by FORTH.

```
R FORTH                                  Command to RT-11 (run FORTH)
                                         (Underlining indicates FORTH's output.)
FORTH  19 JUL 74                         FORTH's initial message
HELLO ?                                  "FARTH" is not recognized, note "?"
FARTH LOAD FARTH ?                       Same mistake, corrected using "rubout"
FAAORTH LOAD OK                          Basic system loaded, now put no. on stack
123 OK                                   Type the current stack entry
. 123                                    Same thing on one line.
123 . 123                                If the stack is empty, you get error message
. ?                                      Unrecognized word is an error,too.
XXYZZ XXYZZ ?                            Put two numbers on the stack
123 234 OK                               Type the top one (it is popped)
. 234                                    Then type the second, stack now empty.
. 123                                    Correct a typing error with "rubout"
12993 . 123                              Delete a whole line with CTRL-U (^U)
THIS LINE AAAAIS ALL WRONTTG^U           Comment ignored.
( THIS IS A COMMENT. ) OK
( STACK MANIPULSATIONS) OK               "OK" is the standard response to
1 2 DROP OK                                 a successfully executed input line
. 1                                         if no typing was specified.
. ?
1 2 SWAP . 1
. 2
1 2 OVER . 1
. 2
. 1
. ?
1 DUP . 1
. 1
( CONSTANTS AND INTEGERS) OK
4 CONSTANT FOUR OK                       "FOUR" has the value 4.
FOUR . 4
30 INTEGER L OK                          Typing "L" will put the address of
L . 15720                                   the integer on the stack (15720₈)
L @ . 30                                 "@" retrieves the value of the integer
L ? 30                                   "?" combines "@" and "."
FOUR L = L ? 4                           "=" stores a value in an integer
L @ L @ + L = L ? 10                     Long-winded arithmetic example
4 L += L ? 14
L 1+=  L ? 15
L 0SET  L? L? ?                          A mistake
L 0SET L ? 0
( ARITHMETIC) OK
123 234 + . 357                          addition of integers
4 4 + . 10                               default is base 8, octal
DECIMAL 4 4 + . 8                        try decimal, base 10
5 5 + . 10
5 DUP * 4 - . 21
123 234 456 */ . 63                      123*234/456 if you will
232 10 MOD . 2
: ADD2 2 + ; OK                          A colon definition to add two.
CODE ADD4 S ) 4 # ADD, NEXT J, OK        Code definition to add four.
3 ADD2 . 5                                  (They work.)
2 ADD4 . 6
: PSQUARES 5 0 DO I . I I * . CR LOOP ; OK   Example of DO-LOOPs.
PSQUARES 0 0
1 1                                      The first output line would have
2 4                                      looked better had it started on
3 9                                      a new line.
4 16
                                         We redefine PSQUARES in terms of
: PSQUARES CR PSQUARES ; OK              itself, adding "CR" at beginning.
```

```
PSQUARES
0 0
1 1
2 4
3 9
4 16
```

This is the effect.

```
: TEST 0 < IF [ NEGATIVE!] ELSE [ POSITIVE!] THEN TYPE ; OK
1 TEST POSITIVE!
0 TEST POSITIVE!
-1 TEST NEGATIVE!
123456 TEST NEGATIVE!
12345 TEST POSITIVE!
```

Example of IF-ELSE-THEN

(This is a 16-bit machine, the number has overflowed.)

```
: EX3 CR 5 3 DO 3 1 DO 1 -1 DO I . J . K . CR LOOP LOOP LOOP ; OK
EX3
-1 1 3
0 1 3
-1 2 3
0 2 3
-1 1 4
0 1 4
-1 2 4
0 2 4
```

Example from text showing use of I, J, K.

For a neater output format, we can set F, the output field width.

```
4 F = EX3
  -1    1    3
   0    1    3
  -1    2    3
   0    2    3
  -1    1    4
   0    1    4
  -1    2    4
   0    2    4
```

```
( EDITOR EXAMPLES) OK
OCTAL OK
EIDIT^U
EDIT .    33
EDIT LIST
```

Use base 8 by convention.

EDIT is a constant

List the editor block.

```
    1 ( TEXT EDITOR)
    2 DISCARD REMEMBER DISCARD  : DISCARD FLUSH DISCARD ;
    3 BASE @ OCTAL   100 ARRAY TEXT
    4 : BLANKIT SPACES @ OVER = DUP 2+ 76 MOVE ;
    5 : STRING TEXT BLANKIT DELIM = WORD HERE COUNT TEXT SWAP MOVE ;
    6 : " 42 STRING ;
    7 : ( 51 STRING ;
   10 : HOLD    DUP LINE TEXT 100 MOVE ;
   11 : T       CR HOLD LINE SPACES? TYPE ;
   12 : R       LINE TEXT SWAP 100 MOVE UPDATE ;
   13 : D       HOLD DUP 20 < IF 20 SWAP DO I 1+ LINE DUP 100 - 100
   14           MOVE LOOP ELSE DROP THEN 20 LINE BLANKIT UPDATE ;
   15 : I       DUP 17 DO I LINE DUP 100 + 100 MOVE -1 +LOOP 1 + R ;
   16 : V BLK @ LIST ;
   17 : EMPTY   DUP BLOCK DROP PREV 2+ = ;
   20 BASE = ( RESET RADIX)  ;S END OF EDITOR
EDIT LOAD OK
501 LIST
    1 ;S
    2
    3
    4
    5
    6
```

Load this block.

We will edit block 501; this is the "before" picture.

```
   10
   11
   12
   13
   14
   15
   16
   17
   20
501 BLK = OK                          BLK determines which block is to be
1 T                                   edited.  Type first line.
;S
" INSERT" 0 I OK                      Insert after line 0, before line 1.
1 T
INSERT
2 T
;S
" REPLACE" 2 R 2 T                    Replace line 2, then type.
REPLACE
1 D 1 T                               Delete line 1, type new line 1.
REPLACE
2 T

" LINE 20" 20 R OK
20 T
LINE 20
1 D 1 T

17 T                                  A delete moves line 20 to line 17
LINE 20
20 T

" ( THIS IS A DEMONSTRATION OF THE EDITOR)" 1 R OK
" : MESSAGE [ WRITE A MESSAGE ON THE TYPEWRITER. ] TYPE ;" 2 R OK
" ;S" 3 R OK
" DISCARD" 1 I OK                      "DISCARD" will erase previous
" REMEMBER DISCARD" 2 I OK               application program, "REMEMBER
FLUSH OK                                DISCARD" permits this program to be
501 LIST                                erased later.
   1 ( THIS IS A DEMONSTRATION OF THE EDITOR)
   2 DISCARD
   3 REMEMBER DISCARD
   4 : MESSAGE [ WRITE A MESSAGE ON THE TYPEWRITER. ] TYPE ;
   5 ;S
   6
   7
  10
  11
  12
  13
  14
  15
  16
  17
  20
501 LOAD OK                           Load the sample program.
MESSAGE WRITE A MESSAGE ON THE TYPEWRITER.
DISCARD OK
GOODBY                                Return to RT-11.
```