Computer Corner

FORTH — A good programming environment for laboratory automation? I. Introduction to the language

Daniel Ph. Zollinger and M. Bos
Enschede, The Netherlands

The origin of FORTH[1]

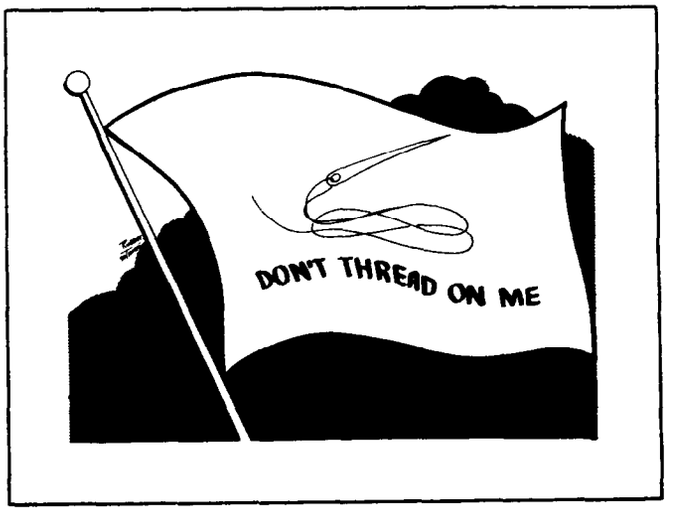
FORTH was created around 1970 by Charles H. Moore as a personal tool to increase his programming productivity. Over a number of years he developed the concepts that make this computer language to stand somewhat apart from earlier generations like FORTRAN, Algol etc.

F-O-R-T-H is an abbreviation of F-O-U-R-T-H (the machine it was coded for allowed only five characters for a name!) and stands for fourth generation computer language. It became more than a personal tool when it was adopted by the astronomy community for which he did a job at Kitt Peak National Observatory. The donation of the system coded for the PDP-11 to DECUS by Martin Ewing was the step that brought it to the attention of the computer community as a whole.

As the FORTH 'movement' gained more momentum and people started to add more and more extensions to the language (extensibility is one of its major features), the need to agree upon a standard kernel arose. This was accomplished in the FORTH-79 standard. However, FORTH evolved on and still does so; so the recent FORTH-83 standard probably will not be the last one[2]. This development certainly shows that the FORTH environment is not a static one, but can be adapted to almost any kind of need in computing.

The nature of FORTH[3,4]

FORTH is an interpreted language that is used fully interactively, which means that commands entered on the keyboard are acted upon immediately. In contrast to the general belief that interpreted languages are always slow, FORTH's execution speed is remarkably good. The reason for this is that interpretation is done on two levels. The text typed at the keyboard is interpreted by the so-called outer interpreter that parses the commands by looking them up in a dictionary. This dictionary, a linked list of commands that can be executed by the inner interpreter, plays a major role in the inner workings of FORTH. These commands are generally implemented in threaded code, which means that they are implemented as a list of adresses of subcommands. The inner interpreter steps through this list to execute these subcommands, which themselves can be lists. At the bottom line of this threaded code are the primaries: subroutines consisting of machine code.



Official Flag of Anti-FORTH Programmers

FORTH language is truly extendable: the kernel dictionary that comes with a standard FORTH system contains commands to define new commands or 'words' as they are called and to add them to the (existing) dictionary. Once defined, these new words are part of the language and can be used in the same way as the basic commands. In contrary to other high-level languages such as FORTRAN or Pascal, every word ('procedure') can be used independent from a 'MAIN' program, which facilitates the

0165-9936/85/$02.00.

top	—	4	5	9	6	7	13	117	117
Stack	—	—	4	—	9	6	9	—	—
	—	—	—	—	—	9	—	—	—
	—	—	—	—	—	—	—	—	—
	—	—	—	—	—	—	—	—	—
Operation performed		4	5	+	6	7	+	*	

*Fig. 1. Use of postfix notation and stack manipulation in the expression (4 5 + 6 7 + *). Numbers are pushed onto the stack at the top. Operators (+ and *) pop the top two entries off the stack and push the result of that operation back on it. For example, the first '+' (column 4) replaces the 4 and 5 on the stack with 9, the result of the addition operation.*

writing and debugging of new programs considerably.

Program design

Program construction in FORTH is done bottom up: starting with simple definitions, based on the contents of the system (kernel) dictionary and working up to higher levels in which these words are used in more complicated ones, one finally defines one word which is used to evoke the complete program (for example RUN or SCAN).

The design of the program, however, is best done top-down: the task to be done is divided in logical sub-tasks which are subdivided etc., until the level of commands available in the standard system dictionary is reached. The programs are well-structured due to the use of conditionals and repetitives such as if-then-else, do-loop, begin-until and begin-while-repeat statements. (FORTH does not even know any goto-statement!) Parameters are passed between the words using a last-in-first-out stack mechanism which implies the use of postfix notation (also called Reversed Polish Notation, RPN) (see Fig. 1). Like many other things in FORTH, the control of the stack is left to the vigilance of the user!

FORTH and the laboratory

FORTH possesses many of the properties a good laboratory computer language should have. (For a detailed discussion of this topic, the reader is referred to the column of R. E. Dessy[5].) Some strong and weak points of FORTH are given in Table I.

TABLE I. Evaluation of FORTH.

Strong points	Weak points
Low memory demand	Programs difficult to read (stack manipulations, postfix notation)
Compact, fastly executing code	Number crunching cumbersome
Flexible, structured programming	'Error trapping' is left to control of the user (array border limits, stack manipulations)
Extensibility	
Modularity	
System portability	

Applications of FORTH are manifold and can be found in such different environments such as process control (control of a robot motion–picture camera or of a fruit sorter in a cannery), medical applications (maintainance of a hospital data base) and data acquisition and handling (control of a complete observatory, including all math-operations, or of a single FT–IR spectrophotometer).

In the second part of this contribution an example of a microcomputer application of FORTH and the analytical laboratory will be given and some of the characteristics mentioned above will be explained in more detail.

References

1 J. S. James, *Byte,* 5 (8) (1980) 100.
2 C. K. McCabe, *Byte,* 9 (8) (1984) 137.
3 R. G. Loeliger, *Threaded Interpretive Languages,* Byte Books, Peterborough, 1981.
4 L. Brodie, *Starting FORTH,* Prentice-Hall, Englewood Cliffs, 1981.
5 R. E. Dessy, *Anal. Chem.,* 55 (1983) 650A.

Additional information

FORTH is already available on most brands of computer, *e.g.*

- On Apple II-like micro's: FysFORTH, developed from P.S.D., P.O. Box 82, 5271 SM St. Michielsgestel, The Netherlands
- On IBM-PC:
  - — PC/FORTH: Laboratory Microsystems Inc., 4147 Beethoven St., Los Angeles, CA 90060, U.S.A.
  - — polyFORTH II level 3: FORTH Inc., 2309 Pacific Coast Highway, Hermosa Beach, CA 90254, U.S.A.
- On PDP-11 system, FORTH is available from the users organisation DECUS.

Daniel Ph. Zollinger and M. Bos are at the Laboratory of Chemical Analysis, Technical University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands.