Helena Huang  
Dr. Nelson  
P7 ATCS: Programming Languages  
1/13/14

Forth

In 1970, Charles H. Moore pooled together what he considered the strengths of various programming languages to build a more direct and therefore efficient interface between humans and machines. Moore considered it a fourth generation programming language, but was limited to five-character file names by the IBM 1130 on which he designed it, giving birth to the shortened name Forth.

Forth did not receive its name until near the end of development, and was notated in all upper-case (FORTH) until the late 1970s, but was later changed to its modern form when lower-case characters became popular and widely available. It was originally designed to control appliances – specifically, astronomy telescopes, which reflects its origins.

In the 1950s, Charles H. Moore, a Physics major at MIT, had already begun developing the basics of his not-yet-conceived language while he worked in the Smithsonian Astrophysical Observatory. He created a simple interpreter with a free-form input in order to deal with bugs arising from Fortran’s rigid structure. After graduating from MIT and subsequently Stanford, as he worked as a free-lance programmer in various languages, he continued using the simple interpreter as much as possible, developing it little by little as technology advanced. Because he preferred it over Fortran, in which the IBM 1130 was developed, he extended Forth into a 1130 compiler, adding features such as looping commands, 1024-byte blocks of memory, memory management tools, and many key compiler features.

Most importantly, he added a dictionary. Procedures were now given names, and the interpreter would search a linked list of names for a match. Each dictionary entry – each node of the linked list – containing a count and three characters, a “code field” consisting of an address to the code to be executed, and parameters, drawing upon ideas he learned from compiler writers at Stanford. The dictionary entries could either point to high level routines or machine instructions. Finally, a second stack, the “return stack,” was added such that the other stack could freely pass parameters without needing to worry about being balanced. However, when Moore ported Forth onto the new Univac 1108 in 1970 to work on a project, he further developed Forth, but the project was cancelled. Moore then wrote a book about the importance of simplicity and innovation and how to develop Forth software.

The first independent implementation of Forth was developed by Moore in 1971 to automate a telescope operate by the National Radio Astronomy Observatory in Arizona using a multiprogrammed and multiprocessor system. His contributions to NRAO enabled unprecedented progress, allowing them to control the telescope and graphically analyze live data. When Elizabeth Rather, a systems analyst, joined Moore at NRAO, she became amazed by the interactivity and resulting efficiency of Forth. She wrote the first Forth manual and published many papers and colloquia. Rather and Moore’s success at NRAO inspired major astronomers from around the world to pick up Forth.

In 1973, Moore, Rather, and another colleague, Edward Conklin, founded FORTH, Inc. to explore the language’s commercial potential. Because of the memory limitations of the minicomputers at the time, Forth’s efficiency gave it a boost, allowing it to become widely accepted. Several application areas in which it excelled were: commercial/business data base systems, image processing, and instrumentation and control. In 1976, FORTH, Inc. developed a new product specifically for a new 8-bit microprocessor; the language became known as “microFORTH,” and became especially popular among hobbiests, such as the Forth Interest Group, as the “Computer Revolution” spread quickly across Northern California. In the 1990s, Moore also developed colorForth, which used colors to simplify and speed up coding and compiling. Several standardizations, like FORTH-79, FORTH-83, and ANS Forth have been created. Additionally, some tried writing object-based systems based on Forth, like Charles Duff’s Neon, introduced in 1984.

Forth was designed to maximize efficiency by minimizing the number of languages through which normal high-level worked. Whereas another language might need to be processed through an application, a compiler, a supervisor, an assembler, and so on, Forth replaced what Moore called this “vast hierarchy” with a single language. Some argue that this therefore greatly improved the efficiency exponentially because it sped up coding and reduced the cost, bugs, maintenance, and commenting necessary. The language was supposed to be as easy to code and debug as BASIC yet as fast and powerful as Assembly, because it is not an interpreted language. Everything is stack-based, and the user must handle stack management by himself, which further increases efficiency by reducing necessary processing. A strange feature of the language is its ability to metacompile. For example, Forth has been written in Forth, and the core colon compiler can be coded using the colon compiler.

Ultimately, though usage of it has declined over the years, the hyper-minimalist programming language Forth succeeded in its goal to facilitate great efficiency.

References

The evolution of Forth. (n.d.). Retrieved January 13, 2014, from Forth website: http://www.forth.com/resources/evolution/evolve\_0.html

Forth language. (2010, June 15). Retrieved January 13, 2014, from Cunningham & Cunningham, Inc. website: http://c2.com/cgi/wiki?ForthLanguage

The Forth programming language. (1999, 11 13). Retrieved January 13, 2014, from University of Michigan-Dearborn Engineering website: http://groups.engin.umd.umich.edu/CIS/course.des/cis400/forth/forth.html

Meyers, C. (2009). FORTH - A simple stack oriented language. Retrieved January 13, 2014, from Open Book Project website: http://openbookproject.net/py4fun/forth/forth.html

Moore, C. (1991). Forth - the early years. Retrieved January 13, 2014, from Color Forth website: http://www.colorforth.com/HOPL.html