Forth: Background and History

CSE 410/510 ETH: Interactive Programming Environments

Ethan Blanton

Department of Computer Science and Engineering

University at Buffalo

Conception

Charles Moore was unhappy with the languages of the early ’60s. [5]

I figured that in 40 years a very good programmer could write forty programs. And I wanted to write more programs than that.

He particularly objected to rigid formatting constraints. [8]

He developed Forth to improve his personal efficiency.

Viewpoint

Moore's opinions on programming and languages are unorthodox.

It is difficult to argue they are not successful.

Forth represents a minimal language [5]:

I will say that if you take anything away from FORTH, it is not FORTH any longer—the basic components are all essential to the viability of the language.

Early Development

From 1960 to 1970, Moore developed an interpreted language.

In the early 1960s, it started to become a stack language.

In the late 1960s, it was freed from card punch development.

Around that time it was named FORTH on the IBM 1130.

The first FORTH

FORTH was originally Fourth, for 4th-generation computing. [4]

The IBM 1130 was limited to five characters per filename. [5]

Most I/O in that period was still uppercase-only.

Thus: Fourth $\rightarrow$ FORTH.

It later became Forth again, since it is not an acronym.

Forth on the 1130

Forth came of age on the IBM 1130.

There, it developed:

- Additional flow control
- The modern dictionary
- Indirect threaded, compiled words
- The second return stack

This forth was still not fully compiled.

1130 Forth was published to introduce Forth to the world. [6]

Portability

Early Forth, and Forth-likes, were ported many times.

Moore used them on:

- IBM 1130
- IBM 360
- Honeywell 316
- PDP-11
- …

…and implemented in PL/I, Algol, FORTRAN, assembly, *etc.*

In the 1970s, Forth was implemented on 18 CPUs. [8]

Science and Forth

Much of Moore's early Forth work was in the name of science:

Satellite tracking at the Smithsonian Astrophysical Observatory;

Particle acceleration and nuclear physics at Stanford;

Data-fitting at Stanford;

Telescope control at the National Radio Astronomy Observatory.

Simplicity

Remember that “if you take anything away from FORTH, it is not FORTH any longer”.

Moore called “Keep it simple!” his Basic Principle. [8]

He added the corollary “Do not speculate!”

Forth views generalization as something to be avoided.

The underlying machine is an important consideration.

All-level Language

Brodie describes forth as both low and high-level. [1]

From the 1970s on, Forth is typically implemented in Forth.

It provided a native assembler to accomplish this!

From 1971 onward, Forth was an operating system, as well.

Rather Discovers Forth

In 1971, Elizabeth Rather began working on Forth at NRAO.

She rapidly discovered that she liked the system.

Rather and Moore wrote a multi-user Forth OS for PDP-11.

It was in use from 1973 to 1991. [8]

This success led to Forth, Inc. and Forth as a universal solution.

Portability, Again

The mid-70s Forth system required about 60 assembly words.

Forth's self-hosting nature allowed it to cross-compile itself.

Moore took about two weeks to port Forth to a new system. [8]

This process included writing the assembler!

These Forth systems were very efficient and multi-user.

Diversity of Application

Forth applications in the 1970s were incredibly diverse. [8]

- Interactive timesharing
- Database systems
- Image processing
- Instrumentation and control

Elizabeth Rather would usually teach Forth to the customers.

The Forth Interest Group

In 1978, Forth, Inc. declined to produce Forth for the 6502.

Bill Ragsdale asked Army Major Robert Selzer to write a 6502 Forth.

Ragsdale *et al.* formed the Forth Interest Group in 1978.

The Ragsdale/Selzer Forth was published as open source.

Thus FIG Forth was born.

FIG Forth

FIG Forth was an independent reimplementation of Forth.

By 1979 it was available on seven or more architectures. [2]

At the time, there was no Forth standard.

FIG Forth did some things differently from Forth, Inc.

Thousands of microcomputer owners used and extended FIG Forth.

It effectively became a de facto standard for Forth.

A Complete Package

FIG Forth was a complete, single-tasking system.

It included:

- Forth
- Terminal I/O routines
- Disk I/O routines
- An editor (in Forth, run from Forth)
- An assembler for the current platform

The FIG journal (Forth Dimensions) published FIG Forth extensions.

Standardization

FIG and FIG Forth led to many discussions of Forth implementation.

Forth, Inc. was involved in many of the discussions!

Proposals were made for control flow, defining words, …

Meanwhile, commercial Forth implementations also improved.

A Forth standard became desirable.

Standards were produced in 1977, 1979, and 1983.

ANS Forth was standardized in 1994.

Threaded Interpreters

Forth code is compiled to an interpreted form.

This is normally implemented as a threaded interpreter.

Threaded interpreters produce very compact code.

Interpretation of threaded code is almost trivial.

This makes compiled Forth relatively fast and efficient.

The Dictionary

Forth words are compiled into the dictionary.

We discussed the dictionary when we learned Forth.

Dictionary entries contain:

- The word itself
- (optionally) Code implementing the word
- (optionally) Data

We saw empty entries, words with data, words with code, …
(Remember `CREATE ... DOES> ... ;`!)

Indirect Threading

Indirect threading is a common code format.

In indirect threaded Forth, the code for a word contains:

- The address of native code to interpret the word
- A list of addresses of words that this word invokes

Native code words include their native code directly.

Interpreted words link to the implementation of the interpreter.

[We'll look at this on the blackboard.]

Direct Threading

Some platforms use direct threading, instead.

Direct threaded code words contain:

- The actual native code to interpret the word
- A list of addresses of words that this word uses.

The difference is that:

- Indirect threading had a pointer to code
- Direct threading has the code

[We'll look at this on the blackboard, as well.]

Direct vs Indirect

Which threading method to use depends on the platform.

On some hardware it might not make a big difference.

On some hardware it might greatly affect either time or space.

In both cases, the interpreter word is quite simple.
(Frequently, four to ten instructions!)

Understanding Forth Implementations

I highly recommend reading through Jonesforth. [3]

You might find Moving Forth [9] helpful to understand it.

Once you have finished Jonesforth, planckforth [7] is a true joy.

Applications

Many diverse applications have been built in Forth:

- EasyWriter, an early word processor for the Apple II
- Boeing 777 avionics
- Many satellites
- Starflight, a space exploration video game

This is in addition to many process control applications.

OpenBoot

An interesting use of Forth is OpenBoot.

Open Boot was a Forth boot firmware.

It was developed at Sun Microsystems for SPARC systems.

Devices could include platform independent OpenBoot drivers in ROM.

It was standardized as IEEE 1275, called Open Firmware.

Apple and IBM subsequently adopted Open Firmware for PowerPC.

References I

Optional Readings

[1] Leo Brodie. “Thinking Forth: A Language and Philosophy for Solving Problems”. In: (1984).

[2] Forth Interest Group Editors. “The Forth Implementation Project”. In: *Forth Dimensions* 1.4 (July 1979).

[3] Richard WM Jones. *JONESFORTH*. URL: `https://rwmj.wordpress.com/2010/08/07/jonesforth-git-repository/`.

[4] Charles H. Moore. “Forth – The Early Years”. 1999. URL: `https://colorforth.github.io/HOPL.html`.

[5] Charles H. Moore. “The Evolution of FORTH, an Unusual Language”. In: *Byte* 5.8 (Aug. 1980), pp. 76–92. URL: `https://archive.org/details/byte-magazine-1980-08/page/n77/mode/1up`.

References II

[6] Charles H. Moore and Geoffrey C. Leach. *FORTH – A Language for Interactive Computing*. Tech. rep. Mohasco Industries, Inc., 1970.

[7] Koichi “nineties”Nakamura. *planckforth*. URL: https://github.com/nineties/planckforth.

[8] Elizabeth D. Rather, Donald R. Colburn, and Charles H. Moore. “The evolution of Forth”. In: *History of Programming Languages II*. Association for Computing Machinery, 1996, pp. 625–670. URL: https://dl.acm.org/doi/10.1145/234286.1057832.

[9] Brad Rodriguez. “Moving Forth”. In: *The Computer Journal* 59 (Jan. 1993), pp. 31–38. URL: http://www.bradrodriguez.com/papers/moving1.htm.

License