Forth#

Table of Contents

- Forth
- Background
- Forth Standards
- Forth Systems for the Atari
- Forth Books
- Forth Articles
- Tutorials
- Videos and Screencasts

Background#

Forth is an interpreted programming language that uses a stack-based metaphor in an effort to reduce memory requirements as much as possible. It is perhaps the only successful example of such a language; the PostScript system was derived from it and saw much widespread use, but this was hidden inside the printers and not seen by end-users or programmers.

Stack-based languages simplify the interpreter's parser considerably because the data for an instruction always appears in the source code before the instructions that will use it. To see why this helps, consider this typical line of Basic:

```
A = 10 + 20 * B
```

To perform this line, the interpreter has to read the entire line, look up the value of B (let's say 30), realize that the * has to be performed before + and order the instructions correctly, and then finally convert those into instructions something like:

`get(B,temp1)` - get the value in B and store it in temp1
`multiply(20,temp1,temp2)` - multiply that value by 20 and store the result in temp2
`add(10,temp2,temp3)` - add 10 to temp2 and store the result in temp3
`put(temp3,A)` - store the value of temp3 into the variable A

In contrast, in a stack-based system, the programmer organizes the code in the fashion it will ultimately be performed. The equivalent would be something like:

```
B 20 mul
10 add
```

When this code is performed, the interpreter pushes the value of B on the stack, then 20. It then encounters the mul, which removes the last two items, the 30 and 20, multiplies them, and puts the result back on the stack. Next, it pushes 10 on the stack, leaving the top two locations containing 60 and 10. It then encounters add, taking the two values, adding them, and putting the result back on the stack. The top of the stack now contains the result, 70.

Notice that the stack-based version *has no temporary values*, and only reads a single instruction at a time, not an entire line of code. As a result, the parser is much simpler, smaller and requires less memory to run. This, in turn, generally makes it much faster, comparable to compiled programs.

Another key aspect of the language was Forth's inherently multitasking design. The program could set up separate stacks and feed different code into each one. The Forth kernel would run each of these stacks in turn, so all Forth programs had access to these features. This made writing multithreaded code very easy, so one could, for instance, have a thread reading the joystick as it moved, and then read that value in a game loop in another stack.

The downside to the stack-based approach is that it makes the language difficult to understand by mere mortals. Even tutorials purporting to show how simple it was often ended in an unreadable mess. As a result, Forth was subject to perhaps one of the longest running fanboi wars since APL

was invented. Constantly derided by practically everyone in the industry, it saw some interest in spite of this, but little commercial software emerged. The singular exception is the PostScript system, which is essentially a version of Forth modified to produce graphics output.

The Atari 8-bits were being sold right in the middle of this battle, and as a result there was a fair amount of support on the platform and some interest in the press.

Forth Standards#

- Forth79 (1979)
- Forth83 (1983)
- ANSI Forth (1994)
- Forth 200x (2009)

(Family tree)

Forth Systems for the Atari#

- FOCO65 a Forth Cross-Compiler written in Python that translates into XASM assembly language
- SPL (Simple Programming Language) a Forth-ish compiler written in Python that translates into Assembly language
- X-FORTH - a FIG Forth variant, currently maintained
- VolksForth - a powerful Forth83 standards Forth for Atari 8bit, Atari ST, MS-DOS, CP/M, C=64, C=16/116/Plus4, still maintained
- ANTIC Forth
- valFORTH
- English Software Company FORTH
    - Page 6 Review of ES Forth
- Extended Atari FIG-Forth APX20029
- Mesa Forth
- QS Forth
- Graphic Forth - A ANTIC / Fig-FORTH 1.4s Version with special Graphics Extensions.
- FIG Forth 1.1
- FIG Forth 1.0D
- fig-FORTH1.4S-1.atr
- fig-FORTH1.4S-2.atr
- ProForth Apple II (6502 Source)
- SNAUT
- Forth Compiler from Frank Ostrowski
- CoinOp FORTH

Forth Books#

- Using Fig FORTH On The Atari 800 By Stephen A. Cohen
- APX 20157 FORTH Turtle Graphics Plus Manual by William D. Volk

Forth Articles#

- What is Forth?
- Converting FIG-Forth Programs to Forth-83
- Forth Code Size
- 6502 Assembler in Forth
- A FORTH ASSEMBLER FOR THE 6502 by William F. Ragdale, FOURTH DIMENSIONS Vol 3, 5p, 143ff

- FigForth Source Listing - FIG Forth for the BBC Micro in 6502 Assembler
- Some Debugging Sourcecode found on a BBC Micro Fig-Forth Disk
- 6502 DISASSEMBLER in Forth
- Kermit Protocol in Forth
- 6502 Forth like tiny Operating System
- APPLE II QForth
- GNU Forth EC for 6502
- Henry Laxen on Slashdot 2002
- Yet another target compiler
- Forth Macros
- Local Variables
- Freedom of Assembly by Julian V. Noble
- Forth sorting routines
- Forth memory allocator
- Forth Database design "ELEMENTS OF DATA BASE DESIGN" by Glen B. Haydon
- Signed Integer Division by Robert L. Smith
- From PASCAL to FORTH by Leonard Morgenstern
- Implementations of NEXT on 6502
- Ultimate CASE Statement by Wil Baden, VD 2 1987

Tutorials#

- Einfuehrung in Forth 83

Videos and Screencasts#

- Introducing FIG-Forth on the Atari 8-bit
- Atari fig-FORTH: Showing the Stack
- Atari FIG-FORTH: About Screens
- Atari FIG-FORTH - Editing Screens
- Atari fig-FORTH: Manipulating the Display List - Part 1
- Atari fig-FORTH: Manipulating the Display List - Part 2
- Atari Fig-Forth - Full Screen Editor progress
- Making Atari fig-FORTH Tools - ANTIC Disassembler 1 of 4
- Making Atari fig-FORTH Tools - ANTIC Disassembler 1 of 4
- Making Atari fig-FORTH Tools - ANTIC Disassembler 1 of 4
- Making Atari fig-FORTH Tools - ANTIC Disassembler 1 of 4