the Forth programming language

If "a language is a mapping from syntax to semantics" then FORTH is one of the most direct languages possible. It can define new words (unlike assembly (except via macros)) and the mapping is as clear one can imagine. Each "word" in forth (delimited only be spaces) is mapped directly to a collected sequence of other words or machine code.

Unlike machine code, where the paradigm is "data from source through operation to destination" FORTH is based on a stack, which is used to hold data for operations which then return new data to the stack. Assignments and parameters are consolidated into this one stack model.

For example, where we might say `A = B + C;` in C or Javascript, or `MOV B, A; ADD C, A;` in assembly, in FORTH we would say `B@ C@ + A!`. This loads the values at address B, and then A, from memory onto the stack, then + takes the top two items from the stack and adds them, replacing them on the top of the stack with the result, which we then bang into the address of A. Where we might implement a procedure called `int vectorDistance(vector A, point B)`which computes the distance between a line and a point, in FORTH we would `A B VECTOR_DISTANCE` (and be left with the distance remaining on the stack).

As a useful example, the following program converts F to C:

`: F2C (f -- c)` 32 - 5 * 9 / ;

The `(f -- c)` is a comment and is meant to show the values on the stack before and after the word is executed. We start with an expectation of a temperature in 'F on the stack, then we push a 32, subtract, push a 5, multiply, push a 9 and divide leaving the result on the stack in 'C.

There are several effects from this approach:

- **Postfix**: Lines of code in a FORTH program generally start with getting the data lined up for the "word" or procedure call. In that sense, FORTH is data oriented, and sometimes it is object oriented, although not often.
- **Stack ops**: There are commonly used words to rearrange the order of items left on the stack by prior words to match the order needed for the next word. If we want to add A and B and see how many times the result will go into 32, we would write `B@ C@ + 32 SWAP /`. The word "SWAP" reverses the order of the top two items on the stack so that we are dividing 32 by A+B instead of A+B / 32.
- FORTH compilers spend a lot less time jumping though hoops for the programmer and a lot more time getting work done. FORTH expects the programmer to be a partner, not a god. Programmers must, therefore, come to understand FORTH and work with it, vs applying, for example, standard math notation and letting the compiler work out the order of operation. You never have to override operator precedence in FORTH, but you do have to write the operations in order yourself.

FORTH, like Linux, is "very user friendly; it's just picky about who it is friends with."
- **Extensible**: You can dig into the "compiler" because there isn't really one; the FORTH compiler is just the set of words your implementation of FORTH came pre-defined with. You are always adding to the compiler (vs writing a separate program) and can, at any time, change the compiler by redefining words. When FORTH is implemented "bare metal" meaning that it is not running as a compiled program in an operating system, but instead *is* the operating system, drivers, language, and everything else on the machine, then you can go all the way down, and all the way back up, from the hardware devices to the applications, in one language.
- Because everything can be changed, everything is often different. "When you've seen one FORTH, you've seen one FORTH". And the lovely thing about FORTH language standards is that there are so many to choose from. For this reason, FORTH rarely rises above the level of bare metal, custom development for low level control systems. While a few high level applications have been written in FORTH, it is very difficult to find one programmer, or a team of programmers, who can work at all levels, from hardware device driver, to GUI implementation, to application programming. Applications programmers who must move from one system to the next often find that their code is not reusable because of differences between implementations.

FORTH in general

- The Forth Interest Group (FIG) http://www.forth.org has a list of local FIGs in Deutsch, Chinese, etc., information on ANS Forth, pointers to many different Forth implementations, articles on cross-compiling, implementing Forth new machines, etc.
- the Forth wiki http://forthfreak.net/wiki/ discusses all kinds of things related to the Forth language, various Forth implementations for Linux, Windows, and other platforms, implementing Forth on new machines, pointers to Forth discussion groups, etc.

Specific Forth implementations

See

- Z80 fig-FORTH v1.1g was featured on the Hackaday SuperConference badge computer under CPM (B: drive FORTH D:
- FORTH on the PIC processor
- https://www.gnu.org/software/gforth/ Gforth use GCC to compile to any target. It is fully ANS FORTH compliant.
- https://colorforth.github.io/ colorForth by Chuck Moore (father of FORTH) https://colorforth.github.io/forth.html Pentium colorForth a direct mapping of Forth Primitives to Pentium instructions. Brilliant! https://github.com/AshleyF/Color/blob/master/Docs/chuck_moores_creations.md
- TpForth http://tpforth.com/ "Supported target architectures are currently 8051, 16-bit 8086, 32-bit 8086 and the MIPS family. ... interactive symbolic debug

[running on] Windows" "free of charge for personal use" /* was http://www.technopoint.net/tpforth/eng/index.html */

- http://www.forthinc.com/Content/Products/SwForth/SwForth.htm Win32 Forth! (offline 2004-09-01 ?)
- http://win32forth.sourceforge.net/ Win32Forth for Windows
- http://sourceforge.net/projects/dragonforth/ DragonForth for PalmOS

Other languages / applications / operating systems implemented in FORTH:

- https://eclipseclp.org/index.html a Prolog constraint solving language in FORTH
- https://en.wikipedia.org/wiki/Open_Firmware A driver / boot system in FORTH
- https://github.com/pzembrod/cc64 A small C compiler, written on the C64, for 6502 processor.

James Newton says:

> Warning: Assembly code and bad memory
>
> 1000 years ago, when I was 17, I wrote a FORTH for the Z80 on a Trash 80. It was unique for the time, because it used the machine language RET as NEXT for asm words. e.g. the machine stack pointed NOT at the data stack or return stack but at the list of word addresses being executed; the thread. That meant that asm words could be strung together even faster than a standard asm program because there was no asm CALL... the RET was both the return from the current asm word AND the call to the next one. The address of the data and return stacks were kept elsewhere... one of the other registers or a memory location (I don't remember, code is long lost). I think it was IX and IY. Every non-asm word had to start with an asm call to a routine that saved SP to the return stack pointer, set the SP to the value on the stack and RET. If I remember, that's called THREAD? Then at the end, the last address was to an asm word that pulled the return stack TOS back into SP and RET. I seem to remember that is called NEXT. So for asm words, NEXT is RET and for higher level words, NEXT was a few asm instructions.
>
> Advantages: /screaming/ fast low level words, no slower high level words.
>
> Disadvantages: low level words couldn't really use the stack, because they would overwrite a thread. Of course, you could save and restore SP... but I seem to remember doing most manipulation of the stack with other registers. e.g. LD (IX+0),SP; INC IX; INC IX; Or maybe it was DEC, I don't remember.
>
> And the big disadvantage: NO INTERRUPTS during asm words! LOL. Can't save PC to TOS because it would over write a thread. I got around that somewhat by adding an EI (enable interrupts) in the NEXT code while the SP was pointed at the return stack, doing a NOP and then DI. LOOP also had that in it because it was asm.

A comparison of Lisp and Forth by Gordon Charlton:

Forth and Lisp are mirror images. The opposites include postfix v. prefix, static allocation v. dynamic allocation, explicit v. Implicit stack.

The primary point of coincidence is that executing a Forth word and evaluating a Lisp function are both depth first tree traversal. They are both interactive and extensible.

Lisp comes from academia and the lambda calculus, a world where more abstract means more fundamental, and computing is mathematics made real with information processing technology.

Forth comes from pragmatism and electrical engineering, where more fundamental means closer to the physics, and software is hardware by other means.

In short, they are as different as Church and Turing, which is to say demonstrably equivalent.

See also:

- http://www.bradrodriguez.com/papers/moving1.htm Innards of FORTH.+
- http://home.iae.nl/users/mhx/crenshaw/tiny.html The Crenshaw language tutorial, adapted to FORTH.+
- https://arduino-forth.com/article/FORTH_exemples_convertInfixToPostfix Converting Infix (standard math notation) to Postfix (RPN as used by FORTH).+
- http://www.complang.tuwien.ac.at/forth/gforth/Docs-html/ Manual for GFORTH. Very nice reference for finding the right word.
- https://repl.it/languages/forth Online FORTH. Very nice. Click on the right panel for immediate response, and on the left side you can write a script and "Run" it.+
- http://home.hccnet.nl/anij/nof/noforth.html Forth for the MSP430+
- http://git.annexia.org/?p=jonesforth.git;a=tree An i386 / Linux version of FORTH. the .S file is the heavily documented asm, the .f files are in FORTH. Very instructive.+
- https://brendanator.github.io/jsForth/ FORTH in JavaScript... so you can use it in a browser.+
- http://www.physics.wisc.edu/~lmaurer/projects/FloatingAway/FloatingAway.html Forth on an Atmel AVR e.g. Arduino.+
- Postscript
- PIC uC 32-bit signed integer stack based math routines. add, subtract, multiply, divide, by Alan Cashin

file: /Techref/language/forths.htm, NaNKB (4 imgs) in 2.024s is NaNKBps, updated: 2020/6/20 18:39, local time: 2020/7/28 06:19,

```
<A HREF="http://www.piclist.com/techref/language/forths.htm"> the FORTH programming language </A>
```

After you find an appropriate page, you are invited to Post your

question
comment
link
preformatted text

to this *massmind* site! (posts will be visible only to you before review) Just type in the box and press the Post button. (HTML welcomed, but not the <A tag: Instead, use the link box to link to another page. A tutorial is available Members can login to post directly, become page editors, and be credited for their posts.

B *I* u

Link? Put it here:

if you want a response, please enter your email address:

Post

Attn spammers: All posts are reviewed before being made visible to anyone other than the poster.

Did you find what you needed? From: "*/microchip/language/forths.htm*"

- *"Not quite. Look for more pages like this one."*
- *"No. I'm looking for:* Fetch "
- *"No. Take me to the search page."*
- *"No. Take me to the top so I can drill down by catagory"*
- *"No. I'm willing to pay for help, please refer me to a qualified consultant"*
- *"No. But I'm interested.* Email *me at* *when this page is expanded."*

PICList 2020

contributors:

o List host: MIT, Site host massmind.org, Top posters @20200728 Bob Blick, RussellMc, Jason White, Neil, David C Brown, Harold Hallikainen, madscientistatlarge, Sean Breheny,

Welcome to www.piclist.com!

James Cameron, Allen Mulvey,
* Page Editors: James Newton, David Cary, and *YOU*!
* Roman Black of Black Robotics donates from sales of Linistep stepper controller kits.
* Ashley Roll of Digital Nemesis donates from sales of RCL-1 RS232 to TTL converters.
* Monthly Subscribers: Gregg Rew. *on-going support is MOST appreciated!*
* Contributors: Richard Seriani, Sr.

.