

The Secret of Forth

Dave Wyland
Asilomar 2009

What Is Forth?

- A dictionary-oriented extensible language
 - ~100 core words
 - Each word does an action
- Program by creating new words from old
 - Or by writing words in Assembly
- Uses a data stack to pass parameters
 - Also for calculations

What Good is It?

- Gives *You* ~5X productivity increase
 - You get more work done per hour
 - You do in a day what usually takes a week
- The ultimate benchmark: Your experience
 - Benchmarks are infinitely arguable
 - Your own, personal experience is not

Forth is Like Cold Fusion

- The lab data is persistent and consistent
- But the theorists cannot explain it
- So they deny it and hope it goes away

Previous Explanations

- Forth encourages small, deeply nested code
 - Always recommended by theorists
 - Forth does feature low overhead calls
 - But why 5X in Forth? Why not C code, too?
- Forth has a tight code-debug-recode loop
 - So does Turbo Pascal. But it does not do 5X

A Revealing Experiment

- In 1978, I implemented a Forth-like system
 - It had a data stack and stack routines
 - But it was written in pure assembly language
 - There was no Forth outer interpreter
 - There was no tight code-debug loop
- But I got the 5X speed up!
 - I thought I implemented Forth – but had not!

Experimental Results

- I got the 5X speed up!
 - I thought I implemented Forth – but had not!
 - Did it again in 2007 in an 8051
- Other results: Almost no variables used!
 - Defined 2, used one. Did not need others.
 - No restrictions on definition or use
 - No conscious intent not to use variables

Discussion

- Forth encourages a “workbench” model
 - The data stack is the workbench
- Data elements on the stack are not named
 - Like a calculator: working data is not named
- Data elements are passed along on the stack
 - They are put there and consumed in place
 - Equations are not used, not needed

Equations Considered Harmful

- Almost all other languages use equations
- Equations require named variables
- Therefore, equations breed variables
- The variables exist because of equations
 - Most are transient, not long term state
 - They disappear when the program ends
 - We get used to them, so we justify them

Disadvantages of Variables

- Named Variables must be understood
- They are a significant abstraction load.
 - OOP exists to deal with variables and names
- Only use them for long term state
 - Data base management, cookies

The Secret of Forth

- Eliminating named variables speeds code
 - THIS is where the 5X came from!
- The experiments eliminated all but the stack
- The 5X speed up was achieved in brute force assembly language coding.
- It works because you experience it
 - It is hard to argue with yourself

That's All, Folks!