ForthImplementation

There are many ways to implement Forth.

Brad Rodriguez wrote the MovingForth series and other papers on Forth implementations on a variety of microprocessors, describing various threading techniques and other tradeoffs.

FIG UK: Build Your Own Forth

threading techniques

Each Forth implementation uses one of these threading techniques:

- Subroutine Threaded Code:
  - Pseudo code
  - Slide show
- Direct Threaded Code:
  - DirectThreadedCodePseudoCode
  - Slide show
- Indirect Threaded Code:
  - IndirectThreadedCodePseudoCode
  - IndirectThreadedCode Slide show
- Token Threaded Code:
  - Pseudo code
  - Slide show
- Return Threaded Code:
  - Pseudo code
  - Slide show
- StringThreadedCode:
  - Pseudo code
  - Slide show

''*Should we describe all these threading techniques on one page, rather than trying to flip back and forth between several nearly-identical pages trying to puzzle out the subtle differences between them ?*''

TU Wien: an overview of various threading techniques

Wikipedia: threaded code is a good introduction to various threading techniques for people that have never heard of threaded code.

stack implementation

Many Forth implementations have only 2 stacks, the return stack (used for subroutine return addresses and subroutine-local data such as the loop index) and the data stack.

Some implementers find it more convenient to split "the" return stack into a dedicated stack only for subroutine return addresses, and a "rack" for subroutine-local data such as the loop index.

Most Forth implementations cache the stack pointers and the value at the Top of each stack in internal CPU registers.

A few Forth implementations have a completely separate stack for floating-point numbers.

Some Forth implementations have an additional data stack used only by interrupt code. A few have several complete sets of stacks (data, rack, return), one for each parallel "process".

other implementation details

While implementing a new version of Forth, you might want to consider where your new Forth fits in ClassifyForthCode .

Reuben Thomas is writing

- Mite: intended to be a good target for all compiled languages.
- Beetle: a virtual processor designed for the Forth language. It uses a byte-stream code designed for efficient execution which is binary portable between implementations.

.
