Introduction

Forth on Large Machines

The power of small computers is still increasing rapidly. Already many personal computers have well over 64K of memory. Larger address spaces, faster processors, and new architectures will dramatically alter the nature of software systems. Changes will come to the way we solve problems, and to which problems we can solve.

Forth originated at a time when 8K was considered a large amount of memory. It is extremely well suited to the kinds of control tasks which are still done with small machines. Forth systems on small machines will remain useful for a wide class of problems.

Forth is very adaptable. Forth systems can take advantage of the capabilities provided by new hardware. Forth systems have already been implemented on 32-bit machines. Many variations on stack widths, threading mechanisms, and dictionary structures have been tried.

The articles in this issue discuss several approaches to the problems of porting Forth to machines with large address spaces. Mitch Bradley tackles some significant and popular philosophical issues. William Sebok describes a subroutine-threaded VAX implementation. Terry Holmes discusses token threading for the Motorola 68000. Robert Berkey addresses the special problems of the segmented addressing of the Intel 8086, and a technique which he calls "segment threading". There is also a paper by Bradley and Sebok concerned with making code portable between Forths of different sizes.

Debate will certainly continue over what Forth should be and how it should adapt to large machines. Forth programmers will continue to do what they have always done – get the job done. We will all benefit from maintaining some uniformity of approach, if only so we can continue to talk.

Michael Perry
Even/Odd Designs
Berkeley, California