Introduction

We close out Volume 5 of JFAR with an issue that offers some peeks at where we may be headed in the Forth world, particularly with respect to hardware improvements. There is much ado about new approaches to improving computer performance, but the extent to which it is occurring in Forth based approaches is astounding.

Debaere's paper, *Language Coprocessor Boosting the Execution Speed of Threaded Code Programs* examines some of the special difficulties associated with executing threaded code on traditional architecture machines and considers methods for improving performance in existing environments. He proposes the use of a coprocessor as a means to accelerate the execution of such programs in a standard environment with a relatively low cost. Such a solution is not meant to replace specialized Forth architectures but rather serve as an additional option.

The use of Forth to power a massively parallel architecture is the subject of Dorband's *Parallel FORTH.* The paper examines the unique problems of language design in a parallel environment and describes the features of the author's solution — MPP Parallel FORTH. The author illustrates the use of such an environment via a sample sort implementation. The underlying architecture here certainly represents another approach to improving program performance!

The next two papers describe processor architectures designed with Forth in mind. Goodman and McAuley's paper, *An Arithmetic-Stack Processor for High Level Language Execution* describes the design of the ASP, a 32-bit building block for a computer optimized for high level language execution. Of course, the native machine language for the ASP just happens to be FORTH. Combined with the fact that the processor is running at 25Mips, this adds up to a rather powerful Forth environment. Hayes and Lee describe the details of another powerful 32-bit microprocessor in *The Architecture of the SC32 Forth Engine.* The SC32 is also designed to run Forth as its native machine language. In addition, the processor couples RISC techniques with two stack caches to provide the potential for impressive performance executing Forth programs.

Zettel's paper, *Error-Free Statistics in Forth*, is a fine illustration of the use of Forth to produce statistical programs that minimize the loss of information during calculation. In particular, the author employs scaling techniques and an implied binary point to compute the mean and standard deviation for data captured with a fixed level of accuracy. The result is a set of methods for a one-pass, error-free calculation of these statistics.

The issue also continues to present information about what's going on around the world in Forth. The ANSI standardization effort for Forth continues to go forward and we report the minutes from Meeting 5 and Meeting 6 in this issue. The abstracts from 1988 EuroFORML keep us abreast of what is happening with Forth in Europe. We also have the abstracts from the first ASYST conference (held last fall in Rochester) to provide some insights into how that Forth-based environment is being utilized as well as what techniques are being developed for problem solving. Keep those cards and letters coming (I haven't received one yet!) and let us know what you think about the current state of the Forth world.

James D. Basile,
Editor-in-Chief