FORTH ENGINES
WORKING GROUP REPORT

Chairman: Philip Koopman
Secretary: Dave Angel

Attendees:

H. Braams, D. Brumm, D. Care, S. Carter, D. Colburn, B. Davis, R. Dixon, J. Fahey, R. Fritsch, D. Full, A. Furman, J. Garst, P. Ginn, H. Glass, J. Guglielmo, T. Hand, G. Haydon, J. Hayes, N. Hennenfent, D. Hooley, M. Hugelshofer, C. Johnson, H. Kamo, K. Kelly, A. Kobylkin, P. Lambrix, B. Lenz, D. Lindbergh, F. MacIntyre, T. McNay, G. Nicol, J. Park, P. Plumbo, E. Rather, J. Rible, A. Rose, S. Rose, J. Ross, D. Schrader, N. Smith, C. Ting, H. Vera, T. Vold, N. Winer, S. Zepp

INTRODUCTION

The purpose of the Forth Machines Working Group was to discuss the issues of performance measurement, application areas, and Forth engines of the future. The points of the discussion given here are meant to stimulate the reader to investigate some of the questions that must be resolved before Forth engines can mature. The results of any such investigations would likely be a worthwhile contribution to next year's Rochester Forth Conference.

PERFORMANCE MEASUREMENT

The concept of "Million Instructions Per Second" (MIPS) for Complex Instruction Set Computers (CISCs) and Reduced Instruction Set Computers (RISCs) and the concept of "Million Operations Per Second" (MOPS) for Forth and stack-oriented computers are not directly comparable. Since the response seen by the user is the ultimate goal, "real-life" benchmarks should be developed. No one in the working group knew of any major benchmarks that would be appropriate for Forth engines.

COMMENTS:

A normalization factor is needed to insulate architectural efficiency measurements from the speed of the hardware technology used for various implementations.

Get the architecture right first, then worry about the technology.

If the user doesn't see a noticeable improvement with a new architecture, then differences don't matter.

When RAM bandwidth is a problem, then CISC makes sense. RISC is very dependent on RAM caching. Forth engines can combine the best of both worlds.

A dual-memory-speed architecture would allow the

programmer to assign heavily used programs/data to a limited amount of high speed memory without the complexity of dynamic caching.

Task switching hurts the concept of using cache.

Some users need low power and robustness, not large MIPS/MOPS figures.

Sorting text could be a possible benchmark.

New architectures aren't that expensive to explore with currently available chips. Getting a commercial-grade implementation to run quickly and/or onto a chip is very expensive.

Some people aren't going to be using Forth benchmarks; they'll be using C. The efficiency of non-Forth high level language implementations may decide the fate of Forth engines.

APPLICATION AREAS

In some respects, Forth engines are solutions looking for problems. The working group listed several application areas that should be well-suited to Forth engine implementations: real time process simulation, signal processing, simulators, large scale models (floating point math bound -- e.g. Atmospheric and Oceanographic models), 3-D graphics, real time animation, artificial intelligence inference engines, robotics servos/real time path planning, voice recognition, sensor fusion, encryption/code breaking, real time video input and output, adaptive data compression, Hypertext(tm), distributed systems, centralized transaction processing, general purpose co-processors for PCs.

The general consensus of this working group was that most current interest is in developing new applications that are made possible by high-speed Forth engines, rather than porting existing personal computer applications over to high-speed hardware.

COMMENTS:

More work needs to be done in addressing whether a high-level language cross-compiled onto or implemented with Forth is sufficient, or whether general-purpose stack-oriented hardware should be used with a different instruction set for each high level language.

New software implementation areas don't have to worry about compatibility with existing hardware.

Tackling new application areas first reduces the perceived threat to present big corporation competition.

Seymore Cray produced his hardware without any high level language compilers available. The users wrote the first compilers. Perhaps Forth engine vendors should not worry about writing compilers for C, Pascal, Modula-2, FORTRAN, etc. themselves.

Most of the listed application areas seem to require a 32-bit architecture for large memory space addressing.

FORTH ENGINES OF THE FUTURE

Even though current Forth engine architectures are still in their first generation, Forth engines can now easily equal the performance of conventional mini-computers. Future stack-oriented architectures could yield mainframe performance at micro-computer prices. Some of the architectural features that may be included in the next generation of Forth engines include: wider data paths, interrupts, parallel processing capabilities, improved context switching hardware support, more than two hardware stacks, Forth engine bit-slices, floating point coprocessor support, and tag bits for data typing.

After much discussion, the Working Group came to the conclusion that even though many exciting application areas (collectively described as a work-station approach) require a 32-bit architecture, the 16-bit machines will always be around for lower-cost, simple applications (such as embedded controllers).

COMMENTS:

32-bit machines give greater integer dynamic range -- 16-bit scaling is painful.

Robotics applications run fine on 16-bit machines.

The 8-bit Z80 microprocessor isn't dead yet! 16-bit Forth engines will not quickly become obsolete.

Since Forth architectures are unconventional, speed is the most important selling point.

"Everyone" would buy a Forth chip if it were only $5, even if it were not the best possible architecture.

Give away the chips for a very low price. Make the profit on software sales.

Parallel processing is in the near future. Forth is well suited to multiple processor applications.

Hardware development will have to follow Software trends.

Current conventional language software applications are too costly and full of errors. Software may end up being thrown out and re-written for significantly more powerful hardware.

Interrupt handling is essential on future machines. Multitasking hardware support is not as critical, since software multitasking seems to get the job done.

Multiple Forth chips in a single box would reduce the need for context switching hardware support.

Future Forth engines should be extensible, just like the Forth language. Coprocessor support (especially for floating point math) is very desirable.

24-bit architectures have historically presented a big marketing problem. 16-bit and 32-bit machines are likely to be the only ones that can be sold.