Letters

Forth Coprocessor

It seems a rather simple method of creating a Forth engine was not directly mentioned in the Forth Machines issue: using coprocessors to implement the Forth run-time inner interpreters. This idea is now practical with the latest 32 bit microprocessors, such as the MC68020 which contains a coprocessor interface and virtual memory capability. Combined with a Forth coprocessor as a sequencer, a Forth Operating System in external ROM, and a modern user interface, such a chip set can be the basis for a multipurpose affordable system.

I'm assuming, of course, that the development costs of designing a coprocessor are smaller than that for a custom CMOS general purpose machine.

Jose Betancourt
Corona, NY

Thoughts on the Future of Number Crunching in FORTH

The work described in my paper "Number Crunching with 8087 FQUANs: The Mie Equations" (this Journal issue) was my first venture into FORTH-based number crunching and was undertaken as a challenge, to see if it could be done. I am more than pleased with the results. Nevertheless, if FORTH is to become a generally useful language for numerical computations, several things need to happen. The most obvious are, in order of increasing difficulty:

1. It must develop a standard floating-point vocabulary;
2. It must be shown to be enough better than other languages to encourage researchers to learn a new language;
3. It must take advantage of numerical co-processors;
4. It must have the support of a useful subroutine library.

In more detail:

1. A proposed floating-point standard is now in review for this journal (MacIntyre 1985a).

2. It is difficult for me to make a direct comparison between FORTH and FORTRAN because I once watched someone using FORTRAN on a micro, and it never occurred to me to try it thereafter. MacIntyre (1985b) reports an interesting bubble-sort benchmark (no floating-point operations) on the IBM PC, showing MMSFORTH beating Microsoft FORTRAN, Microsoft Compiling BASIC, and Pascal by factors of 1.5 to 500, at compilation time, execution time, and compiled code size. Yet I am encouraged by colleagues attempting the Mie equations who write, "Our programs are too slow to be practical. Which FORTRAN compiler are you using?"

3. As long as we implement FORTH with conventional hardware, we can use conventional co-processors. This convenience does not at present exist for the 124-pin 16-bit Novix

4000 designed by Charles Moore and Robert Murphy, which was scheduled for prototyping in silicon in January 1985, by Mostek. Preliminary data (Golden, 1984) mention a 10 MHz clock and one multiple operation like " DUP @ SWAP nn + " per clock cycle, pointing out that even though the prototype is made with standard gate-array techniques, it is expected to run FORTH 100 times faster than today's microprocessors, or about one tenth the speed of the CRAY-1. In a few years, when this approach is implemented in high-speed CMOS or ECL, we will have portable FORTH machines with the speed, if not the memory capacity, of today's supercomputers. Supporting these machines with, or interfacing them to, a numerical co-processor is an area of extraordinary opportunity and need.

4. The only reason I can see for FORTRAN's longevity is its extensive collection of scientific subroutine libraries, with acronyms such as SSP, BMDP, SAS, and IMSL. At least IMSL is available on micros, and all can be reached by attaching to a mainframe. I have made abortive attempts to use the campus PRIME in this manner, but the problems of interfacing to alien software have always made it simpler to re-invent the wheel and write the needed routine. Last month it was a general-purpose matrix-equation solver (now on an MMSFORTH Utilities disc as a precursor to the promised matrix-arithmetic routines); this month it is a taut spline for plotting the chromaticity diagrams, to go into FortHPlot, which drives the HP 7470A plotter. Unfortunately, the source (de Boor 1978) for the taut spline "conveniently" presents its logic in the form of a FORTRAN program, returning us immediately to the starting point: the scientific world is heavily committed to FORTRAN. We can neither lick FORTRAN nor ignore it, so we'd better be able to join it, and that means either a simple method for calling FORTRAN subroutines, or a program which can translate FORTRAN into FORTH.

The first two problems, a floating-point standard and letting the world know what FORTH can do, will more or less take care of themselves. The others, interfacing upcoming FORTH engines to fast co-processors and interfacing FORTH programs to the vast FORTRAN libraries, are undertakings which seem to call for some sort of community action.

Ferren MacIntyre
Narragansett, RI

De Boor, C., 1978. "A Practical Guide to Splines." *Appl. Math. Sci.* 27: Springer-Verlag, Berlin.

Golden, J., 1984. Questions FORTH programmers ask about the Novix 4000. Preliminary data sheet: Novix, 981 University Ave., Suite A, Los Gatos, CA 95030.

MacIntyre, F., 1985b. FORTH, the language microcomputers were invented to run. I, The language. *American Laboratory*, February 1985.

MacIntyre, F., 1985a. Toward a floating-point standard for FORTH. *J. Forth Appl. Res.* (submitted).