Engines of Prosperity

M. Simon

By M. Simon

In my last post I discussed Forth as a language [1].

A language that is based on a virtual machine. What if that virtual machine was turned into a real machine? Good things. For one operations can be done in parallel. Returns can be automatically initiated at the end of an instruction cycle. And except for a few special cases Forth machines are two stack zero operand machines. Thus instruction bits that would otherwise need to be used to designate registers are freed up for other uses. The two stacks are the return stack and the data stack. This means data does not need to be flushed from the return stack on a return. Which means you can nest subroutines easily and upon return the data required for the next operation is at the top of the stack. The process is not totally automatic. But it is nearly so. As you can imagine, eliminating a stack thrash on return from a subroutine is a very good idea. And having the data right where you need it for the next operation is a time saver too. Another time saver is that because "registers" are actually stack items you can have as many "registers" as you need in a machine just by making the stacks deeper. At least if you are designing with an FPGA.

But first a nod to the man who kicked all this off Charles Moore.

Masterminds of Programming: Conversations with the Creators of Major Programming Languages (Theory in Practice) [2] Is a book about a number of programming language designers and how they made the decisions they did. One reviewer had this to say about the interview with Moore, "The interview with charles moore is completely insane, in a good way."

Forth machines come in many flavors. Phil Koopman in his book, Stack Computers: the new wave [3], discusses the design issues in building stack machines. In addition there are a number of examples of machines that have been built. You can also read Phil's book for free on line at Stack Computers - Phil Koopman's Page [4]. In addition you can download a copy from Phil's page.

From: A very short bio of Charles Moore
[5]
In 1983 Moore founded Novix, Inc., where he developed the NC4000 processor. This design was licensed to Harris Semiconductor which marketed it as the RTX2000, a radiation hardened stack processor which has been used in numerous NASA missions. The RTX2000 patent number is 4,980,821. The patent was filed on March 24, 1987 and issued on December 25, 1990. So by any measure the patent has expired. You can look up the patent at the US Patent Office [6].

Here is a link to the RTX2010 [7] data page. The device is no longer in production.

To get the pluses and minuses of such a design Phil Koopman among others compares the RTX2000 [8] to other architectures of its day (1992).

So what do you do if you want a Forth processor these days? You get out an FPGA and program it. Because the design of the processor is so simple they are easy to impliment and test and they don't use a lot of gates. With stacks internal to the machine there is no waiting to get stack data. And since internal stacks can be circular there is no need to flush a stack (change the stack pointer) if you have no further need for the stored data.

So where do you go for such a machine? I have worked with a 16 bit Forth machine that John Rible designed in about 1998 that was very nice. John did the architecture and Cadence did the implimentation. John currently does Forth chips in Verilog for FPGAs. His www site is: Sandpipers [9]. Another place to get a Forth chip is at opencores.com [10]. There is a Forth core [11] that is available for download.

Of course he can help with architecture as I can. I did a few tweaks on the processor John designed. I also have a few ideas of my own for a 32 bit machine.

So how about an assembler/Forth for such a machine? It is pretty easy to write one in Forth. Or Forth Inc. will do the job for you. You can contact them at Forth Inc. [12] Or you could ask me. And think about it: For many of the basic instructions the assembly code maps directly to the machine code. Pretty slick and runs fast too.

If you need help with your Forth Chip design, or even tech writing you can contact M. Simon by getting his e-mail from the sidebar at IEC Fusion Technology [13].

Links:
[1] http://www.ecnmag.com/blog-go-forth-102609.aspx
[2] http://www.amazon.com/gp/product/0596515170?ie=UTF8&tag=poweandcont-20&linkCode=as2&camp=1789&creative=390957&creativeASIN=0596515170
[3] http://www.amazon.com/gp/product/B002ACXDNW?ie=UTF8&tag=poweand

cont-20&linkCode=as2&camp=1789&creative=390957&creativeASIN=B002ACXDNW
[4] http://www.ece.cmu.edu/~koopman/stack.html
[5] http:///
[6] http://www.uspto.gov/
[7] http://www.intersil.com/data/fn/fn3961.pdf
[8] http://portal.acm.org/citation.cfm?id=141868.141872
[9] http://www.sandpipers.com/
[10] http://www.opencores.com/
[11] http://www.opencores.com/project,fcpu
[12] http://forth.com/
[13] http://iecfusiontech.blogspot.com/