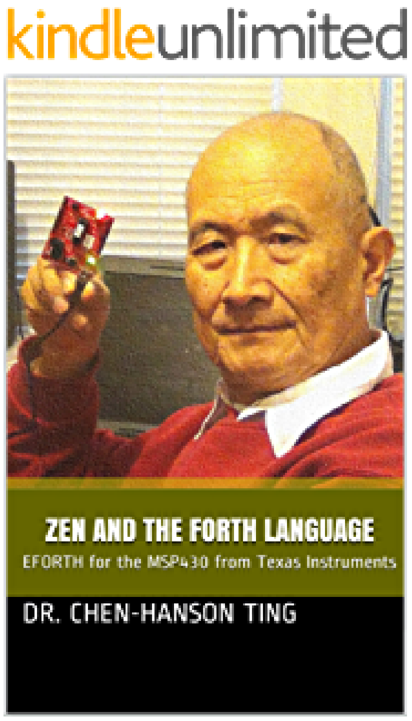
Thoughts from the Towpath

Reflections of a Narrowboat Owner

More Thoughts SIMPL and eForth

Posted on January 21, 2017 by monsonite


kindleunlimited
ZEN AND THE FORTH LANGUAGE
EFORTH for the MSP430 from Texas Instruments
DR. CHEN-HANSON TING

I recently downloaded Dr C.H. Ting's book via Kindle "Zen and the Forth Language" – published by fellow Forther, Juergen Pintaske.

In the book, Ting describes in detail his eForth model ported to the MSP430, and how a complete direct threaded Forth can be built using a limited set of primitives.

Ting's book is very comprehensive, giving details of how to use TI's Code Composer to assemble a program and create an output file in a form that can be used by the 4E4TH IDE. There is sufficient information in his book for any curious person to discover the inner workings of Forth and have a complete working system on the inexpensive MSP430 Launchpad.

Whist at Forth Day, I had the opportunity to talk with Ting and this was the motivation I needed to take the plunge with MSP430 assembly language. Ting's book, and it's detail of the eForth mode convinced me that I was on the right lines with my approach to SIMPL – indeed almost to the point where my implementation of SIMPL could be considered to be a sub-set of Ting's eForth.

version of eForth. This means that if I ever choose to extend SIMPL at a later date – there is a clear and proven route via Ting's eForth.

SIMPL removes the dictionary search overhead that otherwise exists in Forth. The "words" are just singe ASCII characters, and it is easy to jump to an address, calculated from that value. Whilst this might appear wasteful of memory, the ultimate MSP430 target has 256Kbytes of FRAM to play with. It massively reduces the overhead of decoding text strings.

Can we switch between SIMPL and Forth?

SIMPL source code can be written in such a way that it can be expanded into standard Forth, using the SIMPL interpreter to automatically generated this more verbose form. The SIMPL primitives can all be expanded out to their conventional Forth names, as can the internal words associated with letters a-z and user words A-Z.

Conversely, small eForth application programs could be analysed for their word content and mapped onto the primitives and user symbols used in SIMPL. Provided that only 52 user defined words have been used in writing the application – then the switch between Forth and SIMPL should be relatively straightforward.

SIMPL is essentially a shorthand or shortform of Forth. The two should be designed in such a way that they are interchangeable. SIMPL is just a subset of Forth – simplified!

MUP21 & eForth Primitives

From Ting, Dr. Chen-Hanson. Zen and the Forth Language: eFORTH for the MSP430 from Texas Instruments (Kindle Locations 406-413). Juergen Pintaske. Kindle Edition.

Here's a list of the instruction set of Chuck Moore's MUP21 Forth cpu. Ting compared his 31 eForth primitives with these and this confirmed that they were thinking along very similar lines.

Transfer Instructions JMP, JZ, JC, CALL, RET, LOOP

Math/ Logic Instructions AND, OR, XOR, NOT, SHR, ADDC

Memory Instructions LDR, STR, LIT

Stack Instructions PUSH, POP, DUP, DROP, SWAP, OVER, NOP

These are also very similar to the instruction set of James Bowman's J1 Forth processor, which is the intended FPGA target for my SIMPL language. So my plan is to have a maximum of 32 primitives – which can be represented by a 5 bit instruction, and three of these instructions can be pipelined into a 16 bit wide memory – again an idea borrowed from Chuck's MUP21.

instructions from the 5 bit instruction token. This is the price we pay for having a token set that has been chosen for human readability – it needs a further level of decoding to suit the ISA of the Forth processor.

We can of course model the instruction set and architecture of the J1 in MSP430 assembly language, and this might be a useful step to do. In effect we will have created a cross assembler, where SIMPL code is assembled into J1 instructions. Whilst slower than running SIMPL directly in MSP430 assembler, it would be useful to prove the development of the language on the J1 – all be it simulated in code. When run directly on the J1, there will be an order of magnitude speed up.

Uses of SIMPL.

Writing SIMPL has been an education for me, as I have learned about the inner workings of virtual stack based machines, and how they are created within a conventional register based processor. A learning exercise for me, can equally be a educational process for others, so I see one of the uses of SIMPL is as a learning tool to teach students about the instruction set and the architecture of simple stack processors. In the same theme as "From NAND to Tetris" "Building a SIMPL Stack Processor" could be offered in the form of an online study course, consisting of theory, implementation on the MSP430 Launchpad, or Juergen Pintaskes MicroBox. For some, learning electronic engineering and hardware description languages – the implementation in a dedicated soft core processor on an FPGA woud aso be a useful earning exercise.

At it's heart, SIMPL is a text interpreter, and can be used to process text files. Many of our modern manufacturing processes, such as pcb production, CNC machining, 3D printing and laser cutting transfer their information using fairly simple text file formats – such as Gerber and G code. These files consist of numbers and alpha characters which are translated into sequential machine operations. SIMPL can be used as a text processor to convert these files between standards, or to generated new compatible formats – which may be of interest to the open source and maker communities. Animating a walking robot, or fying a drone through a sequence of aerobatic manoeuvres are both things that could be translated into SIMPL text files.

In a later post I will describe how SIMPL source code can be used as a lingua franca for bi-directional exchange of information and program code between radically different classes of computing hardware – from the humble Arduino or Launchpad to the fastest laptop or Octa-core Android platform. Laptops and other mobile computing devices are convenient viewing platforms because of their high performance graphics capabilities, whist small microcontrollers generally struggle to produce anything other than a serial text output.

However if this serial text was effectively a list of drawing commands for graphics primitives – the mobile platform, running a SIMPL virtual machine (written in iForth or Processing for convenience and portability) could interpret the "display list" and produce the fully rendered graphics display from it. SIMPL can send serial text to a Laptop or tablet at some 200,000 bytes per second – perfectly fast enough to render images for datalogging, oscilloscope or IDE type applications.

Twitter Facebook

Like

One blogger likes this.

About monsonite

mostly human

View all posts by monsonite →

This entry was posted in Uncategorized. Bookmark the permalink.

Thoughts from the Towpath