VFX Forth for ARM Linux

Stephen Pelc
MicroProcessor Engineering
133 Hill Lane
Southampton SO15 5AF
England
t: +44 (0)23 8631 441
f: +44 (0)23 8033 9691
e: sfp@mpeforth.com
w: www.mpeforth.com

Abstract

All VFX Forth versions have been built from the same source tree. However, VFX Forth for ARM Linux is the first ARM port since 1999. This paper looks at how well the original VFX Forth source tree has stood up to the changes of the last 15 years.

Introduction

There is now a large number of ARM-based systems running Linux. These range from expensive to very low cost, e.g. Raspberry Pi (around EU 40) and Beaglebone Black. These devices are so cheap that many traditional embedded systems can be replaced at lower cost by these off-the-shelf systems.

These low-end ARM systems use CPUs that range from 450 MHz ARM11s to 1GHz Cortex-A8s. The CPUs all support the original ARM 32 bit instruction set. The systems provide several different flavours of Linux.

The project was thus to port VFX Forth for Linux from the x86 implementation to an ARM with minimal changes to the overall VFX Forth source tree. The source tree is implemented for a multiple stage build. Here we are concerned with the first two stages:

1. Production of the Forth kernel with a primitive interface to the operating system, but with a full assembler, disassembler and code generator,
2. Self-compilation by the kernel of the Linux O/S interface and development tools.

The first stage build is performed by the existing Forth cross compiler for the ARM. This only required a few minor changes to match creeping changes to internal data structures in the target Forth.

Operating System startup and coded definitions

The startup code is contained in a single file which covers the ELF headers, Forth startup from the operating system, primitive access to shared library access routines, and the callback interface. Most of this code is written in assembler, and is the largest assembler component of the whole VFX Forth system. Once the cross assembler, disassembler and code generator are working, this is one of the most critical files in the system.

Barring one or two, all the other code definitions are in a small file that contains words that are best coded. For example, the base ARM32 instruction set does not include a divide instruction. A version of **`CMOVE`** is available that provides four times the performance of a byte-by-byte **`CMOVE`** but is over 900 bytes in size. These routines were taken from the existing embedded ARM target for the cross compiler.

Other files contains the operating system specific routines required for the first-stage build, the default console (unchanged from x86 Linux) and the binary save utility (virtually unchanged from x86 Linux).s

Assembler, disassembler, and VFX code generator

The assembler is cross-compiled because it used by the code generator. The disassembler is cross-compiled because you need it to debug the code generator. The code generator is cross-compiled so that all the code in the system is optimised.

The code is taken from the cross-compiler's code tree. Changes are required for defining words. The notation used is from the ANS draft cross-compiler proposal. It's not pretty but it works. Changes are also needed because the MPE cross compilers and VFX Forth use different notations for connecting compilation semantics to word names. This could be improved. Additional minor changes were required because the cross compilers are focused on embedded systems with separate Flash and RAM, whereas hosted systems mainly have a single address space in RAM.

Library linkage

Linking the Forth to shared libraries is a fundamental part of making a Forth for a hosted system. The MPE **`Extern:`** notation emphasises being able to copy and paste a C prototype from the Linux documentation. The following example is taken from the GTK interface.

```
Extern: gboolean "C" g_file_set_contents(
 const gchar * filename,
   const gchar * contents, gsize length,
   GError ** error
);
```

This is, in many ways, the most critical file in the port. It is affected by the startup code and the interface into **dlopen()** and friends. Although MPE has VFX Forth for 32-bit x86 Linux, ARM Linux uses a rather different calling convention with the first four integer parameters passed in registers. Several other O/S interfaces use a similar convention. The choice of how to pass floating point values to Linux affects the floating point package and the parameter passing mechanisms may affect Forth stack layout.

The floating point options are such that we do not yet know how many ABIs must be supported! There are two main ones, for the VFP hardware and for floating point emulation. In many ARM9 implementations, there is no FP hardware and software FP is used. Software FP may well use a library API that passes FP numbers in the integer registers and/or the C stack. Hardware FP may either use the same API as the software FP or may use a faster API that uses the VFP registers. The choice of API is probably defined by the choice of Linux. There is no guarantee that two Linux implementations for the same hardware will use the same FP API.

Once all the choices have been made for the shared library interface, the same choices have to be implemented for the callback interface.

GTK

MPE uses GTK for cross-platform GUIs across Windows, Linux and even Mac OS X. For Linux, GTK is also our primary GUI environment. It has been extended with a simple graphics extension that works in a similar manner to the old Borland BGI interface from long ago.

Apart from the different shared library names on different operating systems, the GTK interface and the demo shown above uses the same source code unchanged.

Similarly the majority of the Forth examples and library interfaces compile unchanged.

GPIO

Using a Raspberry Pi as a base system, the speed of GPIO access varies hugely according to how it is done. The following link has the gory details:
http://codeandlife.com/2012/07/03/benchmarking-raspberry-pi-gpio-speed/

Depending on language and implementation technique, you can expect to see GPIO access in the range 40 kHz (Python) to 20 MHz (optimised C). In VFX Forth we expect a generalised routine to achieve about 7 MHz, while specific access should exceed 15 MHz. To achieve such speeds, the Forth application must be run with root permissions.

Conclusions

Once code generation is good enough, the vast majority of a Forth system can be written in high level Forth. The hard parts of the remainder are involved in the operating system interface. A very few routines are still best written in assembler, for example a high performance version of **CMOVE**.

Given that the last 15 years of VFX Forth development have all been for the Intel IA32 instruction set, the addition of ARM and allowance for multiple instruction sets has caused very few changes to existing files. At least for a Linux operating system, there have been no changes to the second stage build except to automate (by conditional compilation) the selection of shared library file names.

As the cost of hardware designed to run ARM Linux has plummeted, e.g. Raspberry Pi, Beaglebone Black and Olimex OlinuXino all fall in the EU 30 to EU60 range, Linux boards are becoming cheaper than conventional embedded hardware. We can expect to see many traditional embedded applications migrate to Linux boards. In particular, we already see the Raspberry Pi (2 million sold), being modified in the B+ form to be significantly more suitable for industrial use – more I/O, better mounting holes, more USB.

Where hard real-time is still important there's always a trade-off, but we are already seeing some migration to FPGA+ARM solutions, e.g. Xilinx Zynq, where the FPGA portion handles the heavy-lifting of the hard real-time requirements. The Zynq incorporates a dual-core Cortex-A9, all the standard peripherals including Gigabit Ethernet, plus an FPGA. Such devices will, in the long term, make the traditional embedded system an extremely niche product.

Acknowledgements

Our thanks to Vic Watson, Juergen Pintaske and several others for encouraging us to generate VFX Forth for ARM Linux.