Building a Coldfire Cross-Compiler for Darwin

Neil Klingensmith **Control Engineering Solutions** 5800 S Oxford Dr New Berlin, WI 53146 neil@controlengineeringsolutions.com

Introduction

This document describes the process by which a cross-compiler targeting the 68k/Coldfire architecture can be built on a Macintosh computer running OS X. The procedure assumes the user has installed the following software packages:

- gcc (available with Apple's X Code Installation)
- macports (www.macports.org)

Additionally, the source code for gcc, binutils, and newlib must be available. These can be obtained from the GNU website.

Procedure Overview

The procedure for bootstrapping a cross-compiling version of gcc is somewhat complex. As with all versions of gcc, the binutils package must be compiled and installed first. The binutils package includes an assembler, linker, and other tools necessary to build the gcc binaries. After binutils is installed, newlib and gcc can be compiled. Unfortunately, the two packages depend on each other. In order to build them from scratch, a minimalistic (bootstrap) version of the compiler (which does not depend on newlib) will be built first, and the newlib package will be built with the bootstrap compiler. Then, the freshly compiled newlib package can be used to build the full version of gcc.

This method for building a cross-compiler has been successfully tested on OS X versions 10.5 and 10.6.

Procedure

1. Use macports to install the gmp and mpfr libraries. To do this, you will need to be root[1] or use the `sudo` command. From this point on, it is assumed that the commands are being run as root. Some of the shell commands will fail if they are run as a normal user.

   ```
   port install mpfr
   ```

   or

   ```
   sudo port install mpfr
   ```

2. Extract binutils to a working directory (I used ~/Documents/binutils-2.19.1). From the directory ~/Documents, type:

   ```
   tar xvjf binutils-2.19.1.tar.bz2
   ```

   Use this procedure to extract the gcc and newlib packages as well. These packages may alternatively be extracted using the package manager bundled with Mac OS X. All three package directories (binutils, newlib, and gcc) should be subdirectories of ~/Documents.

[1] The root user is not enabled by default on OS X. To enable it, open the Directory Utility (Located in /Applications/Utilities in 10.5 and /System/Library/CoreServices in 10.6) and select Edit->Enable Root User. To make changes to the settings, you may have to undo the lock in the lower left corner of the window.

3. Create a new directory under binutils called build. This is the directory from which the compilation commands will be issued[2]. Also, make a directory for the binaries called m68k-elftools.

```
mkdir binutils-2.19.1/build

mkdir /opt/local/m68k-elftools

cd binutils-2.19.1/build
```

4. Configure and build the binutils package:

```
../configure --target=m68k-elf --prefix=/opt/local/m68k-elftools --enable-
interwork --enable-multilib --with-gnu-as --with-gnu-ld --disable-nls

make[3]

make install
```

5. Change to the gcc/build directory and build bootstrap gcc.

```
cd ~/Documents/gcc-4.3.4/

mkdir build

cd build

../configure --with-libiconv-prefix=/opt/local --target=m68k-elf --prefix=/opt/
local/m68k-elftools/ --enable-interwork --enable-multilib --enable-
languages="c" --with-newlib --without-headers --disable-shared --with-gnu-as --
with-gnu-ld --with-gmp=/opt/local --with-mpfr=/opt/local --disable-libssp

make -j3

make install
```

6. Change to the newlib directory and build newlib using the bootstrap version of gcc.

```
cd ~/Documents/newlib-1.17.0/

mkdir build

cd build

../configure --target=m68k-elf --prefix=/opt/local/m68k-elftools/ --enable-
interwork --enable-multilib --with-gnu-as --with-gnu-ld --disable-nls
```

---

[2] For some reason, the build will fail if the compilation commands are issued from the root directory of the source distribution—they must be issued from a subdirectory. This is part of the official method for building gcc and its associated tools, not my workaround.

[3] The make command may be issued with the -jx option where x is the number of concurrent jobs to run. It is usually chosen as the number of cores on the build system plus one. For example, in a dual core system, use the command

```
make -j3
```

```
make -j3

make install
```

7. Change back to the gcc directory and build gcc.

```
cd ~/Documents/gcc-4.3.4/build

rm -rf *

../configure --with-libiconv-prefix=/opt/local --target=m68k-elf --prefix=/opt/
local/m68k-elftools/ --enable-interwork --enable-multilib --enable-
languages="c" --with-newlib --disable-shared --with-gnu-as --with-gnu-ld --
with-gmp=/opt/local --with-mpfr=/opt/local --disable-libssp

make -j3

make install
```

8. Add the binary directory to the PATH environment variable:

```
export PATH=/opt/local/m68k-elftools/bin:$PATH
```

This will cause the terminal to automatically recognize commands like `m68k-elf-gcc`. The PATH environment variable is reset each time a user logs on, so the above command should be added to each user's bash profile script located in the home directory (~/.profile). This can be done using the command

```
nano -w ~/.profile
```