
This guide will outline the steps necessary to build a cross compilation of GCC for a target architecture of MIPS on a OS X host machine. GCC (The GNU Compiler Collection) is an open source compiler used for compiling computer code written in high level languages such as C and C++. The compiler is need to convert the high level / human readable language code into machine level / assembly language code.

As of 2014, Apple is natively shipping the LLVM compiler with OS X (10.9) / Mavericks. I was unable to compile binutils for a target architecture of MIPS using the native LLVM compiler provided by Apple. An alternative I chose was to install GCC (4.8.2) using the Homebrew package management system. In order to install GCC via Homebrew, Homebrew must be installed on the host system first. (duh)

The steps for installing homebrew can be found at brew.sh

Alright, so your still reading, so that probably means you have Homebrew installed on your system, your just bored, or your reading just for the sake of learning something new. In that case, lets continue.

In your favorite OS X terminal application run the following commands to get GCC installed via Homebrew.

```
$ brew tap homebrew/versions
$ brew install gcc48
```

Once GCC 4.8.x is installed the next step is to download the required packages to build a cross compile of GCC. From what I have read on the internet, in order to build a GCC toolchain the binutils package is required. So your probably gonna want to download it.

Recommendation: I would download the following packages to the following location,* `/opt/cross/src/` To download binutils run the following command in a shell.

```
$ wget ftp://sourceware.org/pub/binutils/releases/binutils-2.24.tar.gz
```

Note: *If you don't have* `wget` *installed on your system you can install it with following command,*

```
$ brew install wget
```

Next, download GCC source code,

```
$ wget http://www.netgull.com/gcc/releases/gcc-4.8.2/gcc-4.8.2.tar.gz
```

Note: This link may be broken in the future, so to get the latest version of GCC head on over to gcc.gnu.org

Now extract the following newly downloaded packages with the following command,

```
$ tar xzf binutils-2.24.tar.gz
$ tar xzf gcc-4.8.2.tar.gz
```

Next, download GCC dependencies / prerequisites,

```
$ cd gcc-4.8.2
$ ./contrib/download_prerequisites
$ cd ..
```

Note: *The GCC prerequisites are **mprf**, **gmp**, and **mpc*** Next, create the GCC binary folder / directory.

```
$ mkdir mips-gcc
```

Next, the following environment variables need to be set in order to use the homebrew version of GCC to cross compile GCC for the MIPS architecture.

```
$ export CC=/usr/local/bin/gcc-4.8
$ export CXX=/usr/local/bin/g++-4.8
$ export CPP=/usr/local/bin/cpp-4.8
$ export LD=/usr/local/bin/gcc-4.8
$ export PREFIX=/opt/cross/gcc-mips
$ export CFLAGS=-Wno-error=deprecated-declarations
```

Note: *To check if the environment variables are set correctly run the following command,*

```
$ echo $CC
```

The above command should print something like, `/usr/local/bin/gcc-4.8` Next, configure and build binutils.

```
$ mkdir binutils-build
$ cd binutils-build
$ ../binutils-2.24/configure --target=mips-netbsd-elf --prefix=$PREFIX
$ make all 2>&1 | tee make.log
$ make install
$ cd ..
```

Note: *binutils should now be installed in* `/opt/cross/gcc-mips/mips-gcc/bin` The newly created binaries should be added to your path when building GCC, so run the following command,

```
$ export PATH=${PREFIX}/bin:${PATH}
```

Next, download, unpack newlib into `/opt/cross/src`, then make a directory called,`newlib-build`

```
$ wget ftp://sourceware.org/pub/newlib/newlib-2.1.0.tar.gz
$ tar xzf newlib-2.1.0.tar.gz
$ mkdir newlib-build
```

Now, it's time to configure and build a bootstrap cross compile of GCC :)

```
$ mkdir gcc-build
$ cd gcc-build
$ ../gcc-4.8.2/configure --target=mips-netbsd-elf --prefix=$PREFIX --with-newlib --without-h
--with-gnu-as --with-gnu-ld --disable-shared --enable-languages=c
$ make all-gcc 2>&1 | tee make.log
$ make install-gcc
```

Now, at this point you might think you can compile a simple **Hello MIPS** program using the following code,

```
/* Simple C program. */
#include<stdio.h>

int main() {
printf("Hello MIPS! \n");

return 0;
}
```

However it won't compile . . . yet.

Next, build the newlib C runtime library

```
$ cd newlib-build
$ ../newlib-2.1.0/configure --target=mips-netbsd-elf --prefix=$PREFIX
$ make all 2>&1 | tee make.log
$ make install
```

Finally, it is time to build a complete cross compiler.

```
$ cd gcc-build && rm -rf *
$ ../gcc-4.8.2/configure --target=mips-netbsd-elf --prefix=$PREFIX --with-gnu-as \
 --with-gnu-ld --enable-languages=c,c++ --disable-multilib --with-newlib
$ make all 2>&1 | tee make.log
$ make install
$ cd ..
```

Now the simple **Hello MIPS** program can be compiled using the freshly built compiler.

```
$ mips-elf-gcc -Tidt.ld hello.c -o hello
```

If all goes well, you have just compiled **Hello MIPS** program with the above command and have a MIPS binary ready for testing. Thanks for reading, and happy building.

See Also

External Links

- apple.stackexchange.com thread on installing GCC via homebrew
- Cross compiling GCC for MIPS on Linux/Debian
- osdev.org GCC Cross Compiler article
- Compiling binutils on OS X Mavericks
- An Introduction to the GNU Compiler and Linker
- Toolchains - LinuxMIPS
- GCC Wiki FAQ
- Nacho's Cross Compile of GCC for MIPS
- GNU.org Installing GCC
- GNU.org Building a Cross Toolchain with GCC
- AVR/ARM Cross Toolchains for OS X
- GNU Host/Target specific installation notes