Cross Compiling C Code for RISC-V in Linux

Ashton Johnson & Ian Swepston

May 1, 2017

1 Purpose

This document describes how to build and utilize the RISC-V toolchain for C/C++ development. For information about the toolchain, see the riscv-tools repository on Github.

2 Assumptions

This document assumes that you are using a linux machine with Git installed and that you have access to the fpga-zynq repository

3 Prerequisites

- Download the fpga-zynq repository by performing the following from the command line:

```
git clone https://github.com/ucb-bar/fpga-zynq
cd fpga-zynq
```

4 Install Dependencies

This can be done in parallel with following steps

- Run the following command to install them all:

```
sudo apt-get install autoconf automake autotools-dev curl libmpc-dev libmpfr-dev
libgmp-dev gawk build-essential bison flex texinfo gperf libtool patchutils bc
```

5 Build the Toolchain

NOTE: if installing in /opt (recommended) execute commands as root. To enter root mode use: `su root` OR `sudo su`

5.1 Go to the tools directory

```
cd <path_to_repo>/fpga-zynq/rocket-chip/riscv-tools
```

5.2 Checkout and update the remaining packages

This is slow, but independent of dependency installation. Do it in parallel with other things

```
git submodule update --init --recursive
```

5.3 Export location to install the toolchain (suggest /opt/riscv)

```
export RISCV=/path/to/install/riscv/toolchain
```

5.4 Execute the build script

```
./build.sh
```

6 Generated Files

New files have been generated in the path specified by `$RISCV`, and the executables are in `$RISCV/bin`. Also built at this time are a series of benchmarks and spike, the RISC-V simulator

7 Using The RISC-V Toolchain:

7.1 Setup

Note: Instructions assume installation in `/opt/riscv`

- Add the compiler to the path for ease of use:

Note: This could be put in .bashrc for seamless use with the terminal `PATH=$PATH:/opt/riscv/bin`

7.2 Compile on host PC

This functions like normal gcc/g++

- Execute the build command

`riscv64-unknown-elf-gcc main.c -o main.out`
or
`riscv64-unkown-elf-g++ main.cpp -o main.out`

7.3 Option 1: Execute on system

- Copy the binary via ssh:

`scp main.out root@192.168.1.5`
password is root

- Run the binary using the front-end server

`./fesvr-zynq pk main.out`

7.4 Option 2: Execute on simulator:

Execution in linux environment:
`spike pk main.out`

8 Conclusion

You have successfully installed the RISC-V toolchains and simulator on your computer. This should enable you to compile and run your programs as RISC-V binaries from any linux computer.