Tools Used in 6.S081

For this class you'll need the RISC-V versions of a couple different tools: QEMU 5.1, GDB 8.3, GCC, and Binutils.

We highly recommend using a Debathena machine, such as athena.dialup.mit.edu, to work on the labs. If you use the MIT Athena machines that run Linux, then all of these tools are located in the 6.828 locker: just type '`add -f 6.828`' to get access to them. If you don't have access to a Debathena machine, you can install the tools directly or use virtual machine with Linux via the instructions below.

If you are having trouble getting things set up, please come by to office hours or post on Piazza. We're happy to help!

Using Athena

ssh into one of the Athena dialup machines and add the tools:

```
$ ssh {your kerberos}@athena.dialup.mit.edu
$ add -f 6.828
```

Installing on macOS

First, install developer tools:

```
$ xcode-select --install
```

Next, install Homebrew, a package manager for macOS:

```
$ /usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

Next, install the RISC-V compiler toolchain:

```
$ brew tap riscv/riscv
$ brew install riscv-tools
```

/opt/homebrew/Cellar/riscv-gnu-toolchain/main/bin

The brew formula may not link into `/usr/local`. You will need to update your shell's rc file (e.g. ~/.bashrc) to add the appropriate directory to $PATH.

```
PATH=$PATH:/usr/local/opt/riscv-gnu-toolchain/bin
```

Finally, install QEMU:

```
brew install qemu
```

Installing via APT (Debian/Ubuntu)

Make sure you are running either "bullseye" or "sid" for your debian version (on ubuntu this can be checked by running `cat /etc/debian_version`), then run:

```
sudo apt-get install git build-essential gdb-multiarch qemu-system-misc gcc-riscv64-linux-gnu binutils-riscv64-linux-gnu
```

(The version of QEMU on "buster" is too old, so you'd have to get that separately.)

qemu-system-misc fix

At this moment in time, it seems that the package `qemu-system-misc` has received an update that breaks its compatibility with our kernel. If you run `make qemu` and the script appears to hang after
`qemu-system-riscv64 -machine virt -bios none -kernel kernel/kernel -m 128M -smp 3 -nographic -drive file=fs.img,if=none,format=raw,id=x0 -device virtio-blk-device,drive=x0,bus=virtio-mmio-bus.0`
you'll need to uninstall that package and install an older version:

```
  $ sudo apt-get remove qemu-system-misc
  $ sudo apt-get install qemu-system-misc=1:4.2-3ubuntu6
```

Installing on Arch

```
sudo pacman -S riscv64-linux-gnu-binutils riscv64-linux-gnu-gcc riscv64-linux-gnu-gdb qemu-arch-extra
```

Other Linux distributions (i.e. compiling your own toolchain)

We assume that you are installing the toolchain into `/usr/local` on a modern Ubuntu installation. You will need a fair amount of disk space to compile the tools (around 9GiB). If you don't have that much space, consider using an MIT Athena machine.

First, clone the repository for the RISC-V GNU Compiler Toolchain:

```
$ git clone --recursive https://github.com/riscv/riscv-gnu-toolchain
```

Next, make sure you have the packages needed to compile the toolchain:

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

Configure and build the toolchain:

```
$ cd riscv-gnu-toolchain
$ ./configure --prefix=/usr/local
$ sudo make
$ cd ..
```

Next, retrieve and extract the source for QEMU 5.1.0:

```
$ wget https://download.qemu.org/qemu-5.1.0.tar.xz
$ tar xf qemu-5.1.0.tar.xz
```

Build QEMU for riscv64-softmmu:

```
$ cd qemu-5.1.0
$ ./configure --disable-kvm --disable-werror --prefix=/usr/local --target-list="riscv64-softmmu"
$ make
$ sudo make install
$ cd ..
```

Windows (i.e. running a Linux VM)

We haven't tested it, but it might be possible to get everything you need via the Windows Subsystem for Linux or otherwise compiling the tools yourself.

However, an easier option would probably be to run a virtual machine with one of the other operating systems listed above. With platform virtualization, Linux can cohabitate with your normal computing environment. Installing a Linux virtual machine is a two step process. First, you download the virtualization platform.

- **VirtualBox** (free for Mac, Linux, Windows) — Download page
- VMware Player (free for Linux and Windows, registration required)
- VMware Fusion (Downloadable from IS&T for free).

VirtualBox is a little slower and less flexible, but free!

Once the virtualization platform is installed, download a boot disk image for the Linux distribution of your choice.

- Ubuntu Desktop is one option.

This will download a file named something like `ubuntu-18.04.3-desktop-amd64.iso`. Start up your virtualization platform and create a new (64-bit) virtual machine. Use the downloaded Ubuntu image as a boot disk; the procedure differs among VMs but is pretty simple.

Testing your Installation

To test your installation, you should be able to check the following:

```
$ riscv64-unknown-elf-gcc --version
riscv64-unknown-elf-gcc (GCC) 10.1.0
...

$ qemu-system-riscv64 --version
QEMU emulator version 5.1.0
```

You should also be able to compile and run xv6:

```
# in the xv6 directory
$ make qemu
# ... lots of output ...
init: starting sh
$
```

To quit qemu type: Ctrl-a x.

Questions or comments regarding S6.081? Send e-mail to the course staff at *6S081@lists.csail.mit.edu*.

Top // **S6.081 home** // *Last updated Wednesday, 22-Sep-2021 12:14:49 EDT*