Tools Used in Experiments

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

Installing on Windows

We strongly discourage students from using WSL for experiments because it slows down the tests a lot, leading to unexpected timeouts on some labs. Students running Windows are encouraged to install Linux on their local machine.

First make sure you have the Windows Subsystem for Linux installed. Then add Ubuntu 20.04 from the Microsoft Store. Afterwards you should be able to launch Ubuntu and interact with the machine. To install all the software you need for this class, run:

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

From Windows, you can access all of your WSL files under the *"\wsl$"* directory. For instance, the home directory for an Ubuntu 20.04 installation should be at *"\wsl$\Ubuntu-20.04\home<username>"*.

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
```

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
```

Debian or Ubuntu

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

Arch Linux

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

Running a Linux VM

If the other options listed don't work, you can also try running a virtual machine with one of the other operating systems listed above. With platform virtualization, Linux can run alongside 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-20.04.3-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 shouldn't be too difficult.

Testing your Installation

To test your installation, you should be able to compile and run xv6 (to quit qemu type Ctrl-a x):

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

If that doesn't work, you can double check individual components. Which include QEMU:

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

And at least one RISC-V version of GCC:

```
$ riscv64-linux-gnu-gcc --version
riscv64-linux-gnu-gcc (Debian 10.3.0-8) 10.3.0
...
$ riscv64-unknown-elf-gcc --version
riscv64-unknown-elf-gcc (GCC) 10.1.0
...
$ riscv64-unknown-linux-gnu-gcc --version
riscv64-unknown-linux-gnu-gcc (GCC) 10.1.0
...
```