CSC 3510 Computer Organization

The page describes how to install the environment needed for programming in ARM AARCH64 assembly language.

Windows Demo

Our platform - emulated Cortex-A53's

The emulated machines will be command-line only. A good development environment would be to edit on your native machine and sftp code down to the emulated machine where you run and debug. VS Code has a nice SFTP plug-in (search SFTP by Natizyskunk) that can do this for you every time you save a file.

Additionally, VSCode supports editing via ssh by downloading the *Remote Development-SSH* extension (which can be found here).

The official getting started docs can be found here. This has the advantage of giving you direct access to the VM (including terminal), not just syncing files.

Installing QEMU on MacOS

Get QEMU

This assumes you already have `brew`. If you don't you will need to install that first.

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.s
```

After `brew` is installed, install `qemu`.

```
brew install qemu
```

or, if QEMU is already installed - check for an upgrade:

```
brew upgrade qemu
```

`brew` may ask you to change the ownership of certain directories. Do as it suggests and rerun the `qemu` installation.

Ensure you have QEMU

From a terminal, enter `qemu-system-aarch64` and hit return. The program should be found.

Installing debian

I have built a working system for us to use directly.

Getting the Distro

Go to here.

Download 3510.zip.

Unzipping the distro

The Mac can unzip automatically by double clicking on the zip file. Some browsers support this internally. If you unzip in this way, you can skip the following steps...

1. Create the folder you wish to use for the distro.
2. Change directory into that directory.
3. Unzip the zip file in this directory.
4. Confirm files are present.

Here is a sample:

```
[Donnager] /tmp $ mkdir aarch64
[Donnager] /tmp $ cd aarch64
[Donnager] /tmp/aarch64 $ unzip ~/Downloads/3510.zip
Archive:  /Users/perrykivolowitz/Downloads/3510.zip
  inflating: hda.qcow2
  inflating: initrd.img-4.9.0-8-arm64
  inflating: vmlinuz-4.9.0-8-arm64
[Donnager] /tmp/aarch64 $ ls
hda.qcow2           initrd.img-4.9.0-8-arm64    vmlinuz-4.9.0-8-arm64
[Donnager] /tmp/aarch64 $
```

Adding a QEMU alias

Edit the file that your terminal runs at launch. This could be a different file depending upon your Mac. On the most recent machines, this will be `~/.zprofile`. It may be that you have `~/.profile` instead. If you have neither, make one or the other. Do `ps`. If you see `bash` make `~/.profile`. If you see `zsh` make `~/.zprofile`.

Add this line to the bottom of the file.

```
alias ARM='qemu-system-aarch64 -M virt -m 2048 -cpu cortex-a53 -kernel vmlinuz-4.9.0-8-arm64
```

Save the file and exit.

Ensure the alias works

You can use this alias when you are in the directory where you installed the distro. It will not work anywhere else. Ensure you are in the right directory now.

Run:

```
source ~/.zprofile
```

`~/.zprofile` may be a different file depending upon your Mac.

Now run `alias`.

You should get this plus some other text potentially:

```
[Donnager] /tmp/aarch64 $ alias
ARM='qemu-system-aarch64 -M virt -m 2048 -cpu cortex-a53   -kernel vmlinuz-4.9.0-8-arm64 -in
```

Ensure the distro boots

Run:

```
ARM
```

when *in* the directory where you have unzipped the distro.

Exit the distro by entering `root` as the user and `a` as the password. When the shell prompt is shown, type `shutdown now`. Always do this to exit the distro.

Installing QEMU on Windows

Don't.

Installing QEMU on WSL

1. Install WSL.

This means you may have to enable the Windows Subsystem for Linux in the old Add Remove Programs -> Windows Features settings.

Enable WSL

1. Completely update your Windows machine.
2. Open Settings app.
3. Click on Apps.
4. On right hand side, click on Programs and Features. Yet another settings dialog appears following a completely different standard. Microsoft is wonderful.
5. On the left of the new dialog click `Turn Windows features on or off`.
6. Scroll down to `Windows Subsystem for Linux`.
7. Ensure this is checked. If it is not, checking it will enable WSL.
8. Hit OK and close settings related windows.

There is a tiny chance your computer BIOS might need to enable Virtualization Technology. You won't find out until much later. Modern machine need this less and less.

Install Ubuntu on Windows

From a command prompt type

```
wsl --install -d Ubuntu
```

Follow the rest of the prompts and you'll have a brand spanking new Ubuntu Jammy.

Install QEMU

Once in Ubuntu / WSL:

1. `sudo -i`
2. Enter password.
3. `apt update` If this is your first time, there may be hundreds of packages to upgrade. Don't. Rather do it at home. See below.
4. `apt install qemu-system`
5. Enter
6. Wait
7. `exit` This will leave the super user shell.

QEMU is now installed.

Once you are **at home**, this is how you would update the rest of WSL:

1. `sudo -i`
2. `apt update`
3. `apt upgrade`
4. `exit`

Adding a QEMU alias (Windows)

In Ubuntu / WSL, edit the file that your terminal runs at launch. This will be `~/.bashrc`.

Add this line to the bottom of the file.

```
alias ARM='qemu-system-aarch64 -M virt -m 2048 -cpu cortex-a53 -kernel vmlinuz-4.9.0-8-arm64
```

Save the file and exit.

How? vi, of course.

0. Copy the above line into your copy / paste buffer.
1. Inside Ubuntu: `cd`
2. `vi .bashrc`
3. `G`
4. `A<enter>`
5. Paste (use right click)
6. `ESC`
7. `:wq<enter>`

The **next** time you log in, you should have the alias. You can confirm this by:

```
> alias
```

You should see the ARM alias. If you don't something is wrong.

Ensure the alias works (repeat)

You can use this alias when you are in the directory where you installed the distro. It will not work anywhere else. Ensure you are in the right directory now.

Run:

```
source ~/.bashrc
```

Now run `alias`.

You should get this plus some other text potentially:

```
[Donnager] /tmp/aarch64 $ alias
ARM='qemu-system-aarch64 -M virt -m 2048 -cpu cortex-a53   -kernel vmlinuz-4.9.0-8-arm64 -ir
```

Get the distro

Go to here

1. Download 3510.zip.
2. Choose to `Open with`.
3. Create the folder where your distro will live.
4. Drag all the files to that folder.

Change directory to where your distro lives

This is not as trivial as it sounds. Your home directory in `Ubuntu` is not your Windows home directory.

1. Change directory to here: `/mnt/c/Users`
2. Then `cd` to your user account name.
3. **THIS** is your Windows home directory.
4. `cd` to the distro.

Ensure the distro boots (repeat)

Run:

```
ARM
```

Exit the distro by entering `root` as the user and `a` as the password. When the shell prompt is shown, type `shutdown now`. Always do this to exit the distro.

BOTH PLATFORMS - NEVER USE THE `CONSOLE` WINDOW

Once you have launched the virtual machine, `ssh` into it.

```
ssh user@localhost -p 2222
```

Don't use the console directly. From time to time, the system burbles text to this console, potentially messing up your work. ssh in instead. You have been warned.

For the FIRST VM Project

You, by now, have installed the VS Code plug-ins specified above.

To initialize the SSH plug-in:

- Hit F1
- Select Remote-SSH: Add New SSH Host...
- Type in and hit enter: user@localhost -p 2222
- Hit F1
- Select Remote-SSH: Connect to Host
- Type in and enter: user@localhost
- It will ask you for a password. It is the letter "a"
- It will open another VS Code Window
- It will think for a bit and the first time, will install some stuff
- You can close this window as you have completed the set up for VS Code access to the VM

For EVERY VM Project After The Above

If you did the above, you have saved a connection to the ARM VM.

To edit, build, etc. do the following:

- Start up the VM.
- Start VS Code
- Hit F1
- Select Remove-SSH Connect to Host...
- Select Localhost
- Enter the password - it is the letter "a"
- You will now see the "Open Folder" - click it

- Select “/home/user”
- Enter the password again
- Operate VS Code normally