Advanced Homework 1
A Different Kind of Virtual Machine

Assigned: Friday, January 8, 11:00AM

Due: Friday, January 15, 11:00AM (Hard Deadline)

Not all of the Advanced Homeworks are the same difficulty. I would rate this as one of the hardest ones (but cool!) in no small part because you have not been taught a lot of what you need to know to do this assignemt. I have asked the course staff not to help out with this assignment, but I am curious to see if there are some folks who are capable of doing it.

Submission Instructions

To receive credit for this assignment you will need to stop by someone's office hours and demo your running code. You will need to explain all of the differences in the output of running `file` on the native code versus the ARM code to the instructor.

1 Emulating an ARM

Using VirtualBox, we were able to simulate a whole computer, running a computer as a program inside a running computer. Part of what helped us out was that the simulated computer we were running was very similar to the actual, underlying hardware. Critically, the guest operating system was built for the same ISA as the host machine. What if that's not the case, however? What if we wanted to create a virtual machine for something like a smartphone, that uses an ARM architecture, and run that simulation on our x64 machine?

It turns out we can do this! In fact, the Android development environment ships with a tool that does exactly that. For this assignment we aren't going to simulate a whole phone, but we will use the same underlying tool, QEMU. The Quick Emulator (QEMU) is capable of on-the-fly recompilation, or binary translation, which allows it to run a program compiled for an ARM machine on an x64 machine by rewriting ARM instructions as x64 instructions as they're executed. Today, we're going to build a simple Hello World program for an ARM but run it on an x64 machine.

1. Write a simple Hello World program and save it as `main.c`
2. Look at the output of `cc main.c -o main_x64 && file main_x64`. Copy this down somewhere and save it for later for comparison.
3. Install an ARM cross compiler
4. Compile `main.c` using the cross compiler. Run `file` on the resulting binary. How does it compare to the version that used the regular compiler?
5. Install qemu-arm in your virtual machine (yes, a VM can run a VM, though binary translation is technically not creating a virtual machine)
6. Run `qemu-arm main_arm` to print "Hello World" by running translated ARM code as a native x64 process in a virtual machine!