Hacking on RISC

Mohammad Yazdani

CS 850

2018

SPARC: Where to begin?

The process of booloading a toy kernel on SPARC is partly easy, partly interesting and mostly challenging.

- OpenFirmware: Boot sequence is determined by a 512 byte table in the start of memory space.
- A position-independent binary has to be loaded into memory.
- In the binary, the loader starts with the text section
- Our boot.S determines the position of a string we want to print, and the printing function.
- Sets string pointer and length as 1st and second arguments and then calls the function.

SPARC: Now: ld

Then we need to tell ld to:

- Kickout the headers
- Add a.out signature
- Size of our text section (comes after header)

ARM: Where to begin?

The process of booloading a toy kernel on ARM is much harder than SPARC, mostly because of QEMU.

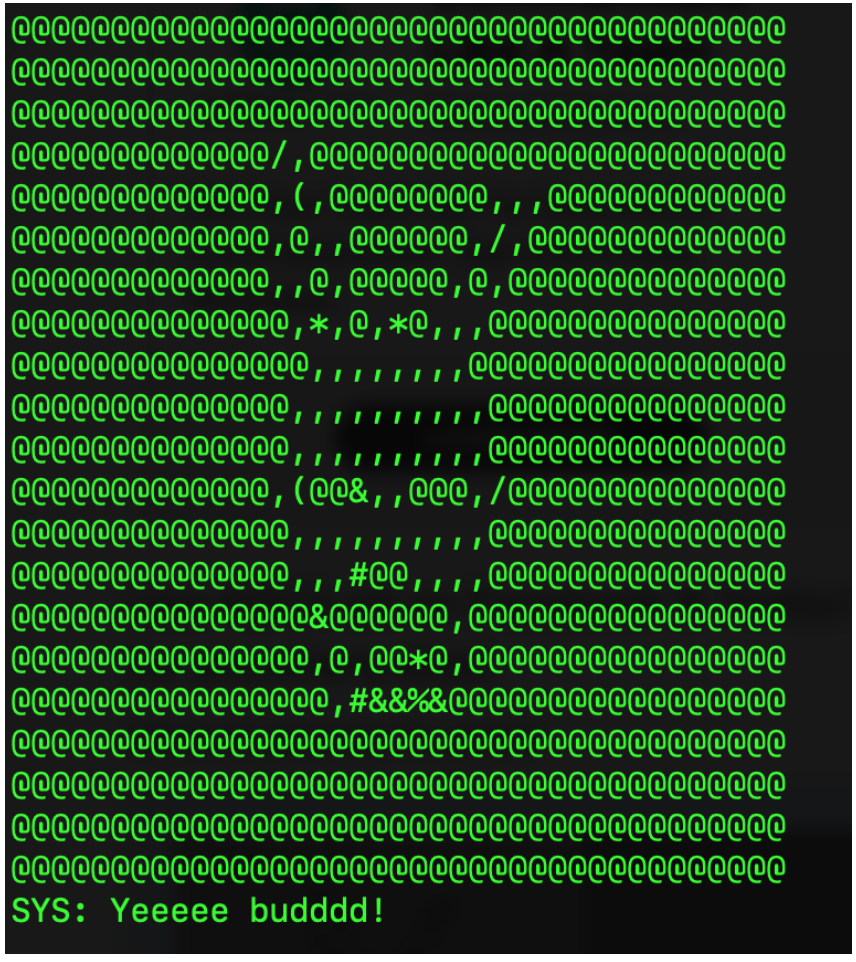
- Boot starts at reset, or any machine reset starts at reset. Reset is address 0.
- ARM has a very good feature and that is the Interrupt Vector Table that is located at reset.
- Interrupt and Exception handling is located in the beginning table and it makes the process of implementing them much easier, specially for embedded systems because of simplicity.
- Even though it was harder and took longer to boot ARM, the code looks cleaner and the linking process is more straight forward. This is mostly because of the Interrupt Vector Table.

After many hours

SPARC — ARM

```
Trying disk...
No valid state has been set by load

0 > boot cdrom
HelloSPARC
```



What I learned and what was interesting:

- Compilers matter: I faced a lot of problems being able to get the right assembler and the right compiler. In the process of figuring out the bootloader I started to get more and more interested in diving deeper on compilers as an integral part of systems dev.
- Compared to what I read on boot loaders for other architectures, the SPARC boot loader is rather more straight forward. The basic process is figuring out your entry point, pointing to something you want to do, return from it and loop.
- Even though writing the bootloader did not need as much assembly, you can still see (also after reading some instruction manuals) that ARM is much more complex than SPARC.