CS33: Introduction to Computer Organization

The course describes how computers actually work, from a mid level hardware architecture and Linux internals point of view.

The hardware is described using the Intel IA32 as an example but the description applies to most digital computers, past and present. A broad overview of the hardware of a complete computer system is presented.

The course covers memory architecture: registers, caches, main memory, disks, etc. The data representation in these memories is described: binary integers, floating point, characters, with all of their different C programming language implementations dealing mainly with their sizes and whether or not they are signed.

Machine and assembly language are taught with the aim of being able to understand, not necessarily to write assembly language code. The Linux implementation of the memory stack and how it implements procedure calls is investigated. The translation of C programming language statements into assembly/machine code is explored to see how C code can be optimized to run faster.

Various modes of caching are explored to see how they are implemented and how they affect the performance of programs with loops, in particular. Virtual memory, another mode of caching is explained in detail. A hybrid software/hardware memory model which enables virtual memory in Linux is explained. Dynamic memory allocation is investigated.

The Linux memory model is further explored to see how processes are managed, primarily in the area of exception control flow. The same path leads to understanding how concurrency is implemented. We also learn the differences between processes and threads and how they are best used. Control of shared resources is examined.

The course makes extensive use of “lab” assignments which are programming tasks intended to support the lectures: and expose how a computer works on a hardware and software level.

The text is:

Randal E. Bryant and David R. O’Hallaron
Computer Systems: A Programmer’s Perspective (CSPP)