CS-301 Computer Architecture
Course Overview

Computer Architecture is a course on computer organization and architecture. The focus of the course is on the design and function of the various components necessary to process information digitally. Computing systems will be presented as a series of layers, starting with low-level hardware and progressing to higher-level software including assemblers and operating systems. These levels constitute a hierarchy of virtual machines.

The study of computer organization focuses on this hierarchy and the issues involved with how the levels are partitioned and how each level is implemented. The study of computer architecture focuses on the interface between hardware and software, and emphasizes the structure and behavior of the system. The majority of this course is devoted to computer hardware, and computer organization and architecture, and their relationship to software performance.

Students invariably ask, "Why, if I am a computer science major, must I learn about computer hardware? Isn't that for computer engineers? Why do I care what the inside of a computer looks like?" As computer users, we probably do not have to worry about this any more than we need to know what our car looks like under the hood in order to drive it. We can certainly write high-level language programs without understanding how these programs execute; we can use various application packages without understanding how they work. But what happens when the program we have written needs to be faster and more efficient, or the application we are using does not do precisely what we want? As computer scientists, we need a basic understanding of the computer system in order to rectify these problems.

There is a fundamental relationship between the computer hardware and the many aspects of programming and software components in computer systems. In order to write good software, it is very important to understand the computer system as a whole. Understanding hardware can help you explain the mysterious errors that sometimes creep into your programs, such as the infamous segmentation fault or bus error. The level of knowledge about computer organization and architecture that a high-level programmer must have depends on the task the high-level programmer is attempting to complete. For example, to write compilers, you must understand the particular hardware to which you are compiling. Also, to model large, complex, real-world systems, you must understand how floating-point arithmetic should, and does, work (which are not necessarily the same thing).

I hope that you find this course an enjoyable experience, and that you take time to delve deeper into some of the material that will not be presented. Although a substantial amount of information will be given, it is only a foundation upon which you can build throughout the remainder of your studies and your career. Successful computer professional continually add to their knowledge about how computers work. Welcome to the start of your journey.