Computer architecture has been a very dynamic field for over four decades. Every year several manufacturers design and implement new architectures for machines. Many trends, such as the RISC approach to simplify decoding and facilitate pipelining, effective use of memory hierarchies, and exploitation of instruction-level parallelism (ILP), data-level parallelism (DLP), and thread-level parallelism (TLP), have spurred current computer architecture research over the last few decades.

In this course we will concentrate on the study of memory-hierarchy design, ILP, DLP, and TLP. The text, "Computer Architecture: A Quantitative Approach. Fifth Edition" by Hennessy and Patterson, is regarded as the definitive text for computer architecture and design. Reading this text will help reinforce your understanding of these concepts. We will also examine a number of research papers that will allow a closer inspection of specific topics.

You will be assigned three programming projects and a few other smaller projects. The first programming assignment will be to write a memory hierarchy simulator. Not only will this assignment help you to understand how memory hierarchies operate, but it should also help you grasp the impact that design alternatives for a memory hierarchy can have on performance. The second programming assignment will be to write a pipeline simulator that determines the exact cycles at which each instruction proceeds through various stages of an in-order pipeline. This assignment will help you understand pipelining and when potential pipelining hazards can occur. The third programming assignment will be to exploit streaming SIMD extensions. This assignment will help you understand the benefits of exploiting SIMD features available on a current processor. Finally, you will be assigned some small supplementary assignments on other topics that will be discussed in this class during the semester.

While most of you will not have careers designing the architecture of computers, it is important to understand concepts in computer architecture. These concepts will be useful for determining what types of systems to purchase, how to write programs so that they execute efficiently on processors, and understanding related concepts that can be applied at higher levels to develop more efficient software systems.