Timing specifications may also be incomplete. Manufacturers do not always guarantee minimum or maximum parameters, depending on the specific type of device and the particular specification. As with DC voltages, worst-case parameters should always be specified. When a minimum or maximum delay is not specified, it is generally because that parameter is of secondary importance, and the manufacturer was unable to control its process to a sufficient level of detail to guarantee that value. In many situations where incomplete specifications are given, there are acceptable reasons for doing so, and the lack of information does not hurt the quality of the design.

Typical timing numbers are not useful in many circumstances, because they do not represent a limit of the device's operation. A thorough design must take into account the best and worst performance of each IC in the circuit so that one can guarantee that the circuit will function under all conditions. Therefore, worst-case timing parameters are usually the most important to consider first, because they are the dominant limit of a digital system's performance in most cases. In more advanced digital systems, minimum parameters can become equally as important because of the need to meet hold time and thereby ensure that a signal does not disappear too quickly before the driven IC can properly sense the signal's logic level.

Output timing specifications are often specified with an assumed set of loading conditions, because the current drawn by the load has an impact on the output driver's ability to establish a valid logic level. A small load will enable the IC to switch its output faster, because less current is demanded of the output. A heavier load has the opposite effect, because it draws more current, which places a greater strain on the output driver.

CHAPTER 3
Basic Computer Architecture

Microprocessors are central components of almost all digital systems, because combinations of hardware and software are used to solve design problems. A computer is formed by combining a microprocessor with a mix of certain basic elements and customized logic. Software runs on a microprocessor and provides a flexible framework that orchestrates the behavior of hardware that has been customized to fit the application. When many people think about computers, images of desktop PCs and laptops come to their minds. Computers are much more diverse than the stereotypical image and permeate everyday life in increasing numbers. Small computers control microwave ovens, telephones, and CD players.

Computer architecture is fundamental to the design of digital systems. Understanding how a basic computer is designed enables a digital system to take shape by using a microprocessor as a central control element. The microprocessor becomes a programmable platform upon which the major components of an algorithm can be implemented. Digital logic can then be designed to surround the microprocessor and assist the software in carrying out a specific set of tasks.

The first portion of this chapter explains the basic elements of a computer, including the microprocessor, memory, and input/output devices. Basic microprocessor operation is presented from a hardware perspective to show how instructions are executed and how interaction with other system components is handled. Interrupts, registers, and stacks are introduced as well to provide an overall picture of how computers function. Following this basic introduction is a complete example of how an actual eight-bit computer might be designed, with detailed descriptions of bus operation and address decoding.

Once basic computer architecture has been discussed, common techniques for improving and augmenting microprocessor capabilities are covered, including direct memory access and bus expansion. These techniques are not relegated to high-end computing but are found in many smaller digital systems in which it is more economical to add a little extra hardware to achieve feature and performance goals instead of having to use a microprocessor that may be too complex and more expensive than desired.

The chapter closes with an introduction to assembly language and microprocessor addressing modes. Writing software is not a primary topic of this book, but basic software design is an inseparable part of digital systems design. Without software, a computer performs no useful function. Assembly language basics are presented in a general manner, because each microprocessor has its own instruction set and assembly language, requiring specific reading focused on that particular device. Basic concepts, however, are universal across different microprocessor implementations and serve to further explain how microprocessors actually function.