1.5 DESIGN OF MICROPROCESSOR SYSTEMS

The effective design of a microprocessor system requires expertise in three areas: hardware design, software design, and system synthesis (see Fig. 1.5-1). The purpose of this text is to present the fundamental concepts required to develop expertise in these areas. If the designer is also responsible for generating the system specification, a detailed knowledge of the specific application is also required.

It is the nature of microprocessor systems that knowledge in any one of the required areas cannot be pursued in isolation of the others. Thus, each of the following chapters involves varying amounts of hardware design, software design, and system synthesis. Early chapters deal primarily with one area, while later chapters essentially are balanced among two or all three.

Before attempting the hardware design of a microprocessor system, the designer must be familiar with the hardware available for use as system compo-

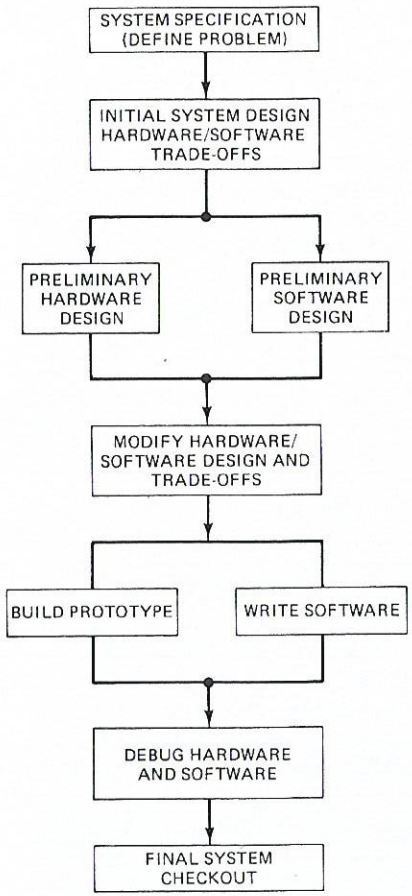

Figure 1.5-1 Steps in designing a microprocessor based system.

nents. Hardware components used range from gates and flip-flops to programmable LSI devices to completely packaged subsystems such as floppy disks. Familiarity with these components includes a knowledge and understanding of their function, logical organization, operation, performance characteristics, limitations, and costs.

Depending on the level at which the design is begun, the designer may be determining the interconnection of individual ICs or the interconnection of premanufactured subsystems. In any event, to be really effective, the designer's knowledge of hardware must extend down to the level of individual ICs.

It is assumed that the reader has a basic knowledge of digital system design using gates and flip-flops. The view of a microprocessor system as a collection of addressable registers is central in this text. Therefore, registers are considered in detail before microprocessors. In Chapter 2 the organization and operation of integrated circuit memory devices, including flip-flops, registers, and random access memories are discussed, as is the use of random access memory devices to form a memory system.

In Chapter 3 microprocessor organization and operation are presented using Intel's 8085A microprocessor as an instructional example. The 8085A is a general purpose 8-bit microprocessor that is an enhancement of the popular 8080A. Once a thorough understanding of the operation and application of this microprocessor is developed, it is relatively easy to understand other microprocessors from a study of the manufacturer's data sheets and application notes. Interconnection of a microprocessor to memory and simple I/O structures is also covered in this chapter.

To accomplish software design at the assembly language level, the designer must first have a knowledge of the microprocessor's instruction set and the function of each instruction. The designer must also know how to combine instructions to form a program that carries out a desired function using a particular hardware configuration. Once the program is written, procedures for translating it into machine language for loading into memory and techniques for testing and correcting any errors in the program must be understood.

The instruction set of the 8085A is introduced in Chapter 4. In particular, the instructions that implement data transfer, logical operations, and branching are presented in detail. These instructions are used in simple sequences and programs. The process of translating an assembly language program to machine code and subsequently testing that program is presented in Chapter 5.

A special memory structure common to microprocessors is the ***stack***. The operation of the stack and the instructions that control it are introduced in Chapter 6. Among its other uses, the stack provides the hardware mechanism to support subroutines. Subroutines and their associated instructions are covered, which leads to a discussion of the use of subroutines in modular program development and some initial considerations of software design.

Carrying out arithmetic functions in a microprocessor system is considered in detail in Chapter 7. Software approaches, which implement arithmetic computations in the microprocessor itself using its arithmetic instructions, are considered. Binary, two's complement, decimal, and floating point arithmetic are covered. Hardware implementation of arithmetic functions external to the microprocessor is also presented.