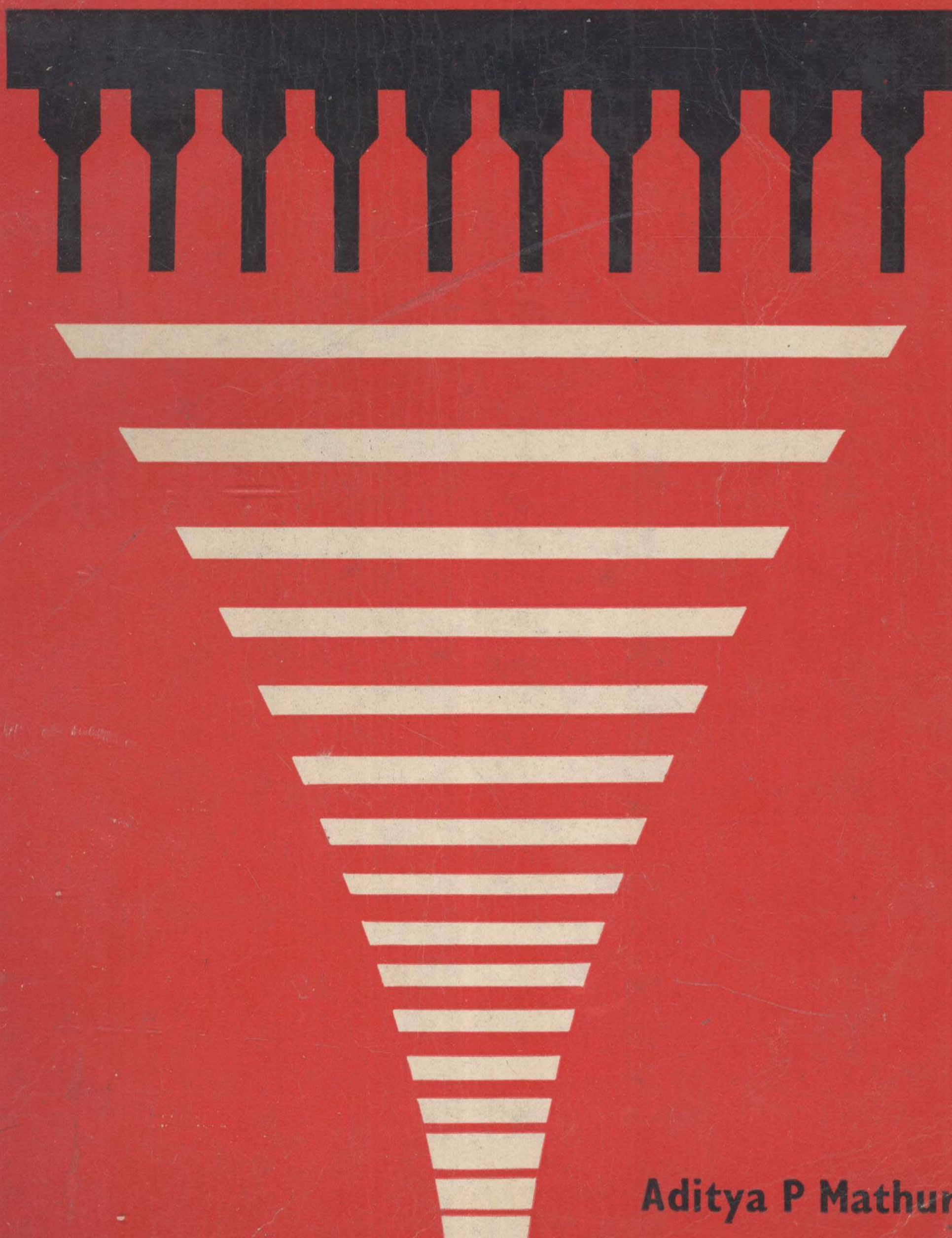
INTRODUCTION TO MICROPROCESSORS



Aditya P Mathur

Introduction to MICROPROCESSORS

ADITYA P MATHUR
Birla Institute of Technology and Science
Pilani

Tata McGraw-Hill Publishing Company Limited
NEW DELHI

Sponsoring Editor: Rajiv Beri
Editorial Supervisor: Usha Tankha
Production Supervisor: M.S. Phogat

Published by Tata McGraw-Hill Publishing Company Limited,
12/4 Asaf Ali Road, New Delhi 110002, and printed at
Jay Print Pack, 8/39 Kirti Nagar Industrial Area, New Delhi 110015

PREFACE

The objective of this book is to teach the basic principles and techniques underlying the design of microprocessor based systems. The emphasis is on presenting in an integrated manner the general principles and techniques and not merely describing a set of microprocessors.

In order that the presentation be coherent, one microprocessor had to be chosen which could act as a vehicle to carry various concepts to the minds of the readers. Our choice has fallen on Intel's 8085A — one of the recent, popular, third generation microprocessors. Apart from its popularity, we chose 8085 because of the ease with which several concepts can be illustrated. However, we hasten to add that several other devices may have proved to be as good as the 8085 for our task.

This book is ideal for those readers who have undergone one course each in Computer Programming and Basic Digital Electronics. However, for those who do not have this background, Chapter 2 and Chapter 4 have been included to provide the necessary background material.

The book starts by introducing microprocessors/microcomputers at a very high level in Chapter 1. Chapter 2 introduces the binary number system. It explains how numbers — integer and real — are represented in a microcomputer memory and the rudiments of binary arithmetic.

Chapter 3 reviews some of the commonly used semiconductor technologies. Semiconductor memories are also described in this chapter.

Chapter 4 introduces the subject of programming. Only assembly language programming is introduced. All those in the field of computer science know that higher level languages are much easier to use than the assembly languages. However, for quite sometime to come, microprocessor users will program in assembly languages due to various reasons, mainly the high cost of operating systems that support compilers. Though the assembly language introduced is the one for the 8085, the concepts involved are general. Thus, having learnt it, one has to merely scan the assembly language manual of any other microprocessor to learn its assembly language.

Chapter 5 presents the organization of the microprocessor itself. This chapter also introduces and explains timing diagrams an understanding of which is essential for any system designer.

Chapter 6 is, in our opinion, the most crucial of all for a reader who is contemplating design of systems involving I/O devices like card readers, floppy disk drives, etc. The chapter introduces general techniques for I/O device interfacing. All the commonly used techniques of data transfer have been described in sufficient detail. For the sake of completeness, interfacing of memories has also been introduced in this chapter.

A large number of special purpose peripheral chips are now offered by several manufacturers to aid a system designer in the task of interfacing. Chapter 7 describes the most commonly used of such devices, like the Peripheral Interface and the Interrupt controller. We strongly feel that any system designer should at least know about the existence of these devices so that he does not waste his time designing discrete versions of these devices when the need arises.

Chapter 8 presents applications of microprocessors. One application — an open loop temperature control system is presented in detail. The complete hardware and software has been developed for this system. Several other applications are described in brief merely to acquaint the reader with the wide applications domain of microprocessors. The design example presented at the beginning of this chapter is intended to familiarize the reader with the system design methodology.

Chapter 9 cursorily describes the hardware and software subsystems that are available to the user of microprocessors. These subsystems are useful development aids for a system designer.

The instruction sets of 8085 and 4004/4040 microprocessors have been included in the Appendices. Those who plan to work with the 8080 microprocessor should also find the instruction set useful as the 8080 is 100 per cent upward compatible with the 8085 in software. The 8085 has only two extra instructions (the RIM and SIM) which are not recognized by the 8080.

We would like to mention that this book is not intended to replace any manual. Complete technical specifications have not been given for any microprocessor. Thus, when designing an actual system this book will have to be supported by relevant manuals related to the microprocessor being used.

Aditya P. Mathur

ACKNOWLEDGEMENT

Several people have directly or indirectly contributed to the writing of this book. It is a pleasure acknowledging the help of all these individuals.

To begin with, I would like to thank Mr. R.P. Jain whose contribution to this book is a full chapter on Semiconductor Memories. It has been a great pleasure working with him while preparing this book.

I cannot forget to mention my friend Mr. R.S. Chauhan with whom I have had valuable discussions which influenced a major part of this book. Thanks are due to Mr. R. Umakishore who was associated with me during the initial stages of writing this book.

My students, especially A.K. Senapati, Smarjit Dey, K.S. Arora and R.K. Gulabani, have helped me by giving parts of this book a careful reading and pointing out several errors and inconsistencies that have been removed in the final draft.

I must thank my wife Mrs. Jyoti Iyer Mathur who has been of tremendous help in the preparation of this book.

Finally, I would like to thank Mr. Banwari Lal Saini who meticulously typed various versions of the manuscript.

CONTENTS

Chapter 1

AN INTRODUCTION TO MICROPROCESSORS

1.1 OBJECTIVES OF THIS BOOK

The purpose of this book is to introduce the reader to the world of microprocessors. A microprocessor* is an electronic device which is of little use unless interfaced with several other peripheral devices. Thus, a study of microprocessors implies a study of a variety of useful peripheral devices and the techniques for interfacing them with a microprocessor. This book concentrates on both these aspects of the study of microprocessors, also known as hardware design.

Like any other digital computer, a system designed by using a microprocessor needs to be programmed, that is, a sequence of instructions has to be written down and then fed to the microprocessor-based system for an effective operation. A sequence of instructions designed to perform a particular task is known as a program. A set of programs written for a microprocessor-based system is known as the software for that system. This book also aims at teaching the reader the art of programming microprocessors. Those who are already familiar with the programming of digital computers (both machine language and high level language programing) will not find much new material except, perhaps, for the instruction set that we present for a few microprocessors. In general, both hardware and software design are of paramount importance in microprocessor-based system design, though, in particular cases, one may be more difficult or significant than the other.

There are many distinct approaches by which the subject of microprocessors can be introduced to a novice. One approach could be the general to specific, according to which general concepts regarding the architecture and programming of microprocessors are introduced without reference to any particular microprocessor. Specific microprocessors are considered only as examples. According to another approach a particular

*In this text we shall use the abbreviation μP for the word microprocessor wherever convenient. Similarly μC will be used for microcomputer.

microprocessor may be introduced first, followed by a generalization of the concepts and their illustration using other microprocessors. In this book, the second approach has been adopted.

The specific microprocessor that we have chosen for illustration of architectural and programming concepts is Intel's 8085 which is a comparatively recent device. Our choice has fallen on 8085 because of the popularity of its predecessor, the 8080, and the fact that both these are almost 100% software compatible and can be made hardware compatible also with the addition of a few extra devices.

In order that the reader gains a thorough knowledge of the techniques for interfacing microprocessors with the real world, detailed design examples have been presented. A laboratory available for experimentation would be an asset for the reader.

We have tried our best to introduce the latest developments in the field of microprocessors. However, as this is a rapidly developing field, the reader may find a few things missing because of the unavoidable delay in the printing of any book.

1.2 EVOLUTION OF MICROPROCESSORS

The first microprocessor was announced in 1971 by Intel Corporation. This was the Intel 4004. It was a processor on a single chip. It had the capability to perform simple arithmetic and logical operations, e.g. add, subtract, compare, AND and OR. It also had a control unit which could perform various control functions like fetching an instruction from the memory, decoding it and generating control pulses to execute it. It was a 4-bit microprocessor operating upon 4-bits of data at a time.

The first microprocessor was quite a success in industry. It found many applications and attracted much attention from both application engineers and semiconductor industry. Soon, other microprocessors were also announced. Intel itself followed by announcing an enhanced version of 4004, the 4040. Since then, many other 4-bit microprocessors have been announced; Rockwell International's PPS4 and Toshiba's T3472 are two examples.

The first 8-bit microprocessor, which could perform arithmetic and logic operations on 8-bit words, was announced in 1973 again by Intel. This was the 8008 which was followed by a better version - the 8080 from the same company. Today there is a variety of 8-bit processors, some examples being Motorola's M6800, National Semiconductors' SC/MP, Zilog Corporation's Z80, Fairchild's F8 and Intel's 8085.

The 8-bit processor was followed by microprocessors operating on 12- and 16-bit data words respectively. Intersil's IM6100 and Toshiba's T3190 are examples of 12-bit processors. Examples of 16-bit microprocessors are Fairchild's 9440, Data General's mN601 and Texas Instrument's TMS9900. Intel's 8086, Motorola's M68000 and Zilog's Z8000 are some of the most powerful 16-bit microprocessors available today.

The evolution of microprocessors from 1971 till today has been characterized by improvement in architecture and the instruction set. Both these aspects are covered in Chapters 3 and 5. Today, a variety of processors exist in the market and the choice of one for a particular application is not easy. Considerations that form the selection criteria are outlined in Chapter 8.

1.3 ORGANIZATION OF MICROCOMPUTERS - PRELIMINARY CONCEPTS

As the name implies, a microcomputer is essentially a small and inexpensive computer. However, another implication of this word is that the heart of the microcomputer is a microprocessor. In fact, both these connotations are true in today's microcomputer scene. A microcomputer is a device which must be capable of:

1. receiving input (data and instructions);
2. performing computations (arithmetic and logical);
3. storing data and instructions;
4. displaying the results of any computations; and
5. controlling all the devices that perform the above mentioned four tasks (directly or indirectly).

A device which enables a microcomputer to perform the first task is known as an input device. Typical input devices used are keyboards, paper tape readers and toggle switches. A device which performs the second task is known as an Arithmetic Logic Unit (ALU). In microcomputers, this task together with control (task 5 above) is performed, generally, by a single-chip microprocessor.

Storage of data and instructions is accomplished using memories. Several types of memories are in use. Semiconductor memories, which are fast, are the most commonly used ones. Casettes and floppy disks are used together with semiconductor memories in relatively large microcomputers. Details of semiconductor memories are given in Chapter 4.

The fourth task of displaying the results (or, in general, the contents of memory) is done by output devices. The commonly used output devices are character printers, cathode ray tubes (CRT) displays and light emitting diodes (LEDs).

Figure 1.1 gives the block diagram of a typical microcomputer. Various input/output devices and memories connected to the microprocessor by 'busses' are shown in this figure. A <u>bus</u> is a group of conducting wires or lines over which electrical signals are transmitted. In a typical system, there are three distinct types of busses as shown in Fig. 1.1, namely the <u>address bus</u>, the <u>data bus</u> and the <u>control bus</u>.

Over the address bus, the microprocessor transmits the address of that device (or a memory location) which it desires to access. For example, if the processor has to access the contents of a memory location having address 2900, then it would transmit this address, in binary form, on the address bus. This address is received by all devices connected to the processor. However, only the device which has been addressed responds.

The data bus is used by the processor to send and receive data to and from different devices. In the example mentioned above, the contents of the addressed memory location are transferred to the processor over the data bus. As another example, the processor may first address an output device and then transmit data on the data bus to the device. Notice that whereas the address bus is <u>unidirectional</u>, the data bus is <u>bidirectional</u>.

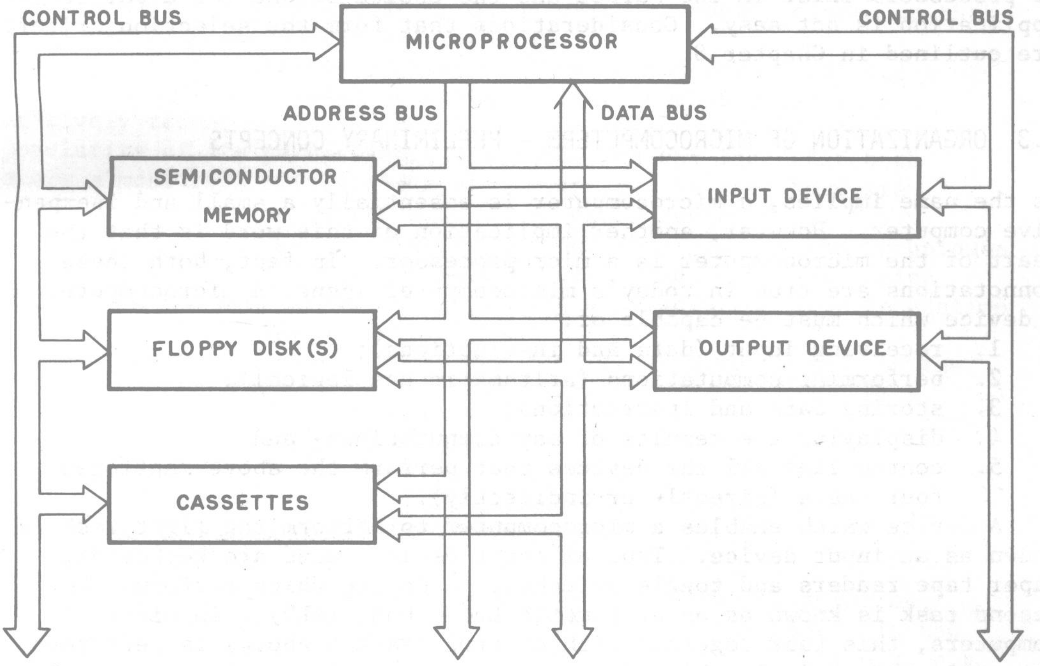


Fig. 1.1: Organization of a typical microcomputer system

The control bus is used for transmitting and receiving control signals between the processor and various devices. For example, after an output device has finished performing an output operation (say printing a character), it may signal the processor of this fact using the control bus. As another example, after an input device, say an analog to a digital convertor, has completed the process of conversion it may inform the processor of this fact using the control bus so that the processor may take further action (e.g. request the convertor to send data on the data bus).

The width of a bus is the number of signal lines that constitute the bus. The data bus for 8-bit microprocessors is 8-bit wide which implies that 8-bits (i.e. a byte) of information can be transmitted in parallel over the data bus. The width of the address bus determines the total amount of memory that can be directly accessed by the microprocessor. Most 8-bit and several 16-bit microprocessors have 16-bit wide address busses which implies that these processors can access a total of 2^{16} = 65536 bytes of memory (abbreviated as 64 K where K = 1024). In some microprocessors (e.g. Intel 8085) though the address bus is 16 bits wide, the 8 bits of data bus are used to transmit a complete 16-bit address. This is known as multiplexing of data and address busses. Details of this and other architectural features of microprocessors will be found in Chapters 3 and 5.

The microcomputer operates as directed by the user. The user writes a sequence of instruction known as a program, and requests the microcomputer to begin executing each instruction starting from the first one. The microprocessor fetches the instructions one by one from the memory and executes them. More details of this aspect of microcomputers will be found in the next section and Chapter 5.

1.4 PROGRAMMING OF MICROPROCESSORS - BASIC CONCEPTS

A program is a sequence of instructions that operate on certain data to produce desired results. Before a microprocessor can be made to perform a task (e.g. comparing two numbers and printing out, using a printer, the greater of the two) it has to be programmed. The program, written to perform a particular task, is stored in the semiconductor memory that is accessible to the microprocessor. The processor, having been instructed to execute the program fetches one instruction at a time from the memory and executes it.

Instructions

An instruction essentially consists of an operation code (abbreviated as opcode) and the address(es) of the data (also known as operand) on which it has to operate. We can thus view an instruction as an element having two fields as shown in Fig. 1.2(a). Most instructions specify directly the address of only one of the operands, the address of the other operand being implicitly known. Some instructions specify the data itself and not the address. The format of such an instruction appears in Fig. 1.2(b). In yet another type of format, only the opcode is specified, the operand address is implicit. An example given below explains all these three types.

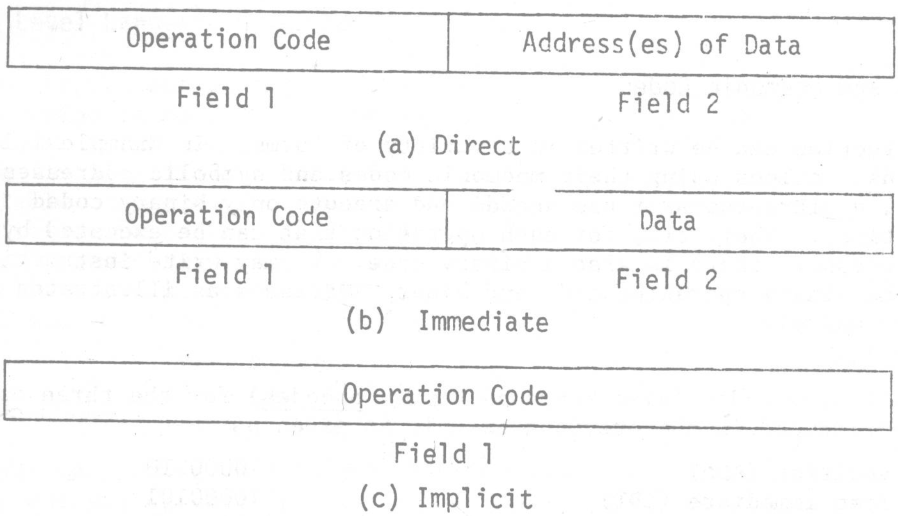


Fig. 1.2: Typical instruction formats

Example 1.1

Almost all microprocessors provide the following instructions in their instruction reportoire:

- Add the contents of a memory cell* (say a byte) to the contents of a particular register within the microprocessor (this register is usually the accumulator register which can hold, 4, 8, 12 or 16 bits of information depending on the microprocessor). The sum replaces the original contents of the accumulator.
- Transfer a given value stored in a memory location (say 17) to the accumulator.
- Shift the contents of the accumulator left by, say, 2 bits.

Each of these three instructions can be referred to by abbreviated codes also known as mnemonics. We assign the mnemonics ADD, LDI and SLA respectively to each of the above mentioned instructions. Now, if we desire to load the accumulator with the constant 7 and then add to it the value of A, where A designates some memory location, then the following sequence of instructions can be given to the microprocessor:

```
LDI   7
ADD   A
```

The first of these two instructions will cause the value 7 to be loaded into the accumulator. The second one will cause the contents of memory location designated by A to be added to the accumulator. Note that these two instructions are of the type shown in Fig. 1.2(b) and 1.2(a) respectively.

If the contents of the accumulator are to be shifted to the left by 2 bits, then the following instruction could be given:

```
SLA   2
```

This instruction is of the type shown in Fig. 1.2(c) where neither the operand nor the address of the operand appears directly in the instructions.

□

Machine and Mnemonic Codes

An instruction can be written in a variety of forms. In Example 1.1, we wrote instructions using their mnemonic codes and symbolic addresses. However, a microprocessor can decode and execute only binary coded instructions. Therefore, for each operation that can be executed by a microprocessor. there is also a binary code. We may write instructions using the binary operation code and binary addresses as illustrated in the next example.

Example 1.2

Let the binary codes (also known as binary opcodes) for the three operations introduced in the previous example as given as

addition (ADD)	10000110
load immediate (LDI)	10000101
shift left accumulator contents (SLA)	1100xxxx

*The phrases 'memory cell' and 'memory location' are used interchangeably to refer to a single byte in the memory.

In the opcode for the shift operation, the last four bits specify the shift count (the number of bits by which accumulator contents are to be shifted to the left).

We may now rewrite the three instructions written in Example 1.1 using the binary opcodes, as follows:

```
1000001    0000011                (LXI 7)
1000010    0000000    00001111    (ADD A)
1100000                           (SLA 2)
```

The first of these is a 2-byte instruction, in which the second byte is used to store the value (7, in our case). The second instruction is a 3-byte instruction in which the last 2-bytes are used for storing the address associated with A (in our case it is 15). The third instruction is only 1-byte long.

Machine and Assembly Language Programming

A machine language program is one which is written using binary opcodes and addresses. In an assembly language program the instructions are written using mnemonic codes and symbolic addresses. It is relatively easier to write an assembly language program for a microprocessor than to write a machine language program. However, a microprocessor can 'understand' only machine language programs. In Chapters 3 and 9, we shall mention tools that are available to a programmer for writing programs in assembly language and then getting them translated to machine language. Examples of programs written in machine and assembly languages are given in Chapter 3.

High Level Language Programming

Higher level languages like Fortran, Cobol and PL/M are available for programming on many microprocessors. Though programming in higher level languages is much easier than using machine or assembly languages, the former is possible only when a microcomputer with sufficient memory is available. Many people may possess only small microcomputers that they wish to use in certain applications. Such persons may be forced to write programs in machine or assembly language. However, the choice of a language for programming a microprocessor depends on many factors which are outlined in Chapter 9.

1.5 SUMMARY

In this chapter we have introduced microprocessors for the benefit of those who are already familiar with the very basis of computer programming and organization. In such an introductory chapter, detailed description of any aspect of a microprocessor is not possible and,