Microprocessors and Microcomputers

Hardware and Software

RONALD J. TOCCI
LESTER P. LASKOWSKI

Microprocessors and Microcomputers

Hardware and Software

RONALD J. TOCCI
Monroe Community College

LESTER P. LASKOWSKI
Monroe Community College

PRENTICE-HALL, INC., *Englewood Cliffs, New Jersey 07632*

Library of Congress Cataloging in Publication Data

TOCCI, RONALD J
Microprocessors and microcomputers.

Includes index.
1. Microprocessors. 2. Microcomputers.
I. Laskowski, Lester, joint author. II. Title.
QA76.5.T556 001.6'4'04 78-26831
ISBN 0-13-581330-1

Editorial/production supervision and interior
design by Don Rosanelli and Gary Samartino
Cover design by Edsal Enterprises
Manufacturing buyer: Gordon Osbourne

Printed in the United States of America

10 9 8 7 6 5 4 3 2

PRENTICE-HALL INTERNATIONAL, INC., *London*
PRENTICE-HALL OF AUSTRALIA PTY. LIMITED, *Sydney*
PRENTICE-HALL OF CANADA, LTD., *Toronto*
PRENTICE-HALL OF INDIA PRIVATE LIMITED, *New Delhi*
PRENTICE-HALL OF JAPAN, INC., *Tokyo*
PRENTICE-HALL OF SOUTHEAST ASIA PTE. LTD., *Singapore*
WHITEHALL BOOKS LIMITED, *Wellington, New Zealand*

Microprocessors
and
Microcomputers

Hardware and Software

Preface

This book was written to present a broad spectrum of readers with a practical introduction to the relatively new world of microprocessors and microcomputers. It should prove to be useful to the computer novice, technical student, and practicing engineer alike. A comprehensive review of digital principles and circuits is provided for readers with a minimum background.

As the title suggests, this book concentrates on the fundamentals of microprocessor-based systems and is *not* intended to be a survey of the numerous microprocessors and microprocessor applications. The authors have chosen to emphasize the ideas and principles common to all microprocessor systems. The only deviation from this approach is the chapter on programming, where the principles are best described using an actual microprocessor's instruction set. Once the reader understands these principles, it is a relatively easy task to extend this understanding to any microprocessor.

The text is divided into three sections: review material, hardware, and programming. Chapters One and Two provide a thorough review of terminology, number systems, and digital circuits from basic gates to memory chips. Chapters Three through Six deal principally with computer structure and hardware, with some programming concepts introduced to show how the software and hardware work together. These chapters cover microcomputer structure, internal microprocessor organization, memory interfacing, and input/output interfacing. Chapter Seven is a detailed treatment of microcomputer programming on a machine language level, with some elements of assembly language.

The text includes several valuable learning aids to facilitate the understanding of important concepts: (1) numerous thoroughly explained illustrative examples; (2) clear, uncomplicated diagrams and flowcharts; and (3) extensive glossaries of new terms at the end of each chapter. In addition, the interrelationship of hardware and software is emphasized throughout the text so that the reader can develop a solid understanding of the complete microprocessor system which he/she can easily build on.

The authors wish to extend their thanks to several special people for their valuable assistance in preparing the original manuscripts. Through the efforts of Sally Altobello and our wives, Cathy and Sandy, the manuscript was completed almost as fast as we could write it and it was essentially error-free. We also would like to thank each other for performing this formidable task with virtually no areas of disagreement and with a spirit of cooperation that made the task a pleasant one.

RONALD J. TOCCI
LESTER P. LASKOWSKI

Contents

PART 2: MICROCOMPUTER HARDWARE

PART 3: MICROCOMPUTER SOFTWARE

PART 1
Introductory Topics

1

Number Systems and Codes

Computers of all sizes have one thing in common—they handle *numbers.* In digital computers, these numbers are represented by binary digits. A *binary digit* is a digit that can only take on the values of 0 or 1, and no other value. The major reason why binary digits are used in computers is the simplicity with which electrical, magnetic, and mechanical devices can represent binary digits. Because the term "binary digit" is used so often in computer work, it is commonly abbreviated to *bit.* Henceforth, we shall use the latter form.

1.1 DIGITAL NUMBER SYSTEMS

Although actual computer operations use the binary number system, several other number systems are used to communicate with computers. The most common are the decimal, octal, and hexadecimal systems.

Decimal System

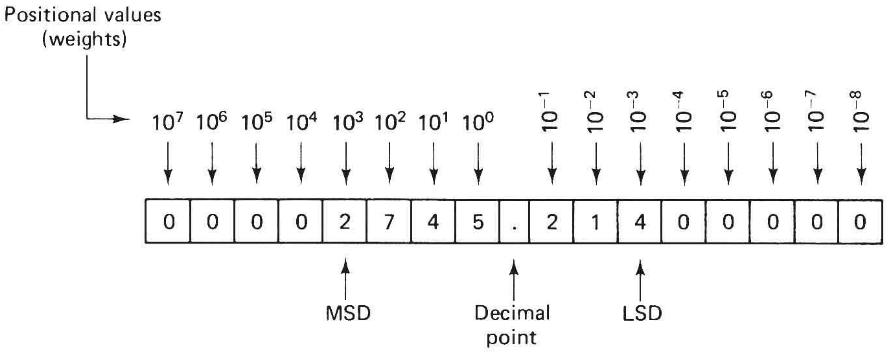
The *decimal system* is composed of the 10 symbols or digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9; using these symbols, we can express any quantity. The decimal system, also called the *base 10* system, because it has 10 digits, has evolved naturally as a result of the fact that man has 10 fingers. In fact, the word "digit" is the Latin word for "finger."

The decimal system is a *positional-value system*, in which the value of a digit depends on its position. For example, consider the decimal number 453. We know that the digit 4 actually represents 4 *hundreds*, the 5 represents 5 *tens*, and the 3 represents 3 *units*. In essence, the 4 carries the most weight of the three digits; it is referred to as the *most significant digit* (MSD). The 3 carries the least weight and is called the *least significant digit* (LSD).

The various positions relative to the decimal point carry weights that can be expressed as powers of 10. This is illustrated below, where the number 2745.214 is represented. The decimal point separates the positive powers of 10 from the negative powers. The number 2745.214 is thus equal to

$$(2 \times 10^{+3}) + (7 \times 10^{+2}) + (4 \times 10^{+1}) + (5 \times 10^{+0}) + (2 \times 10^{-1}) + (1 \times 10^{-2}) + (4 \times 10^{-3})$$

In general, any number is simply the sum of the products of each digit value times its positional value:



Decimal Counting

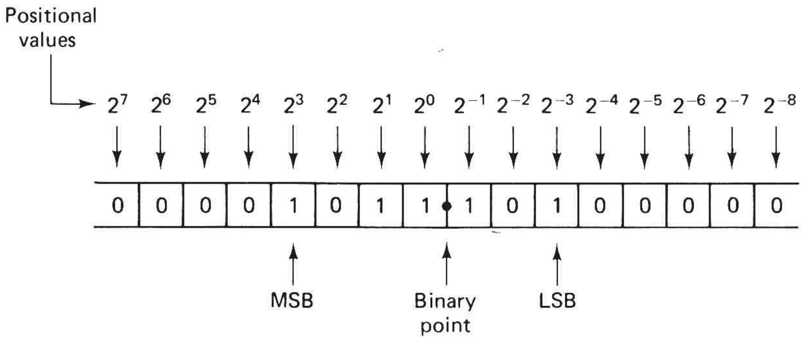
The number 9 is the largest digit value in the decimal system. Thus, as we are counting in decimal, a given digit will progress upward from 0 to 9. After 9, it goes back to 0 and the next higher digit position is incremented (goes up by 1). For example, note the digit changes in the following counting sequences: 25, 26, 27, 28, 29, 30; 196, 197, 198, 199, 200.

For a given number of digits, N, we can count decimal numbers from zero up to $10^N - 1$. In other words, with N digits we can have 10^N different numbers, including zero. To illustrate, with three decimal digits, we can count from 000 to 999, a total of 1000 different numbers.

Binary System

In the *binary system* there are only two symbols or possible digit values, 0 and 1. Even so, this *base 2 system* can be used to represent any quantity that can be represented in decimal or other number systems. In general, though, it will take a greater number of binary digits to express a given quantity.

All the statements made earlier concerning the decimal system are equally applicable to the binary system. The binary system is also a positional-value system, wherein each bit has its own value or weight expressed as powers of 2, as follows:



In the number expressed above, the positions to the left of the *binary point* (counterpart of the decimal point) are positive powers of 2 and the positions to the right of the binary point are negative powers of 2. The binary number 1011.101 is represented above, and its equivalent decimal value can be found by taking the sum of the products of each bit value (0 or 1) times its positional value.

$$\begin{aligned}1011.101_2 &= (1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0)\\ &\quad + (1 \times 2^{-1}) + (0 \times 2^{-2}) + (1 \times 2^{-3})\\ &= 8 + 0 + 2 + 1 + 0.5 + 0 + 0.125\\ &= 11.625_{10}\end{aligned}$$

Notice in the preceding operation that subscripts (2 and 10) were used to indicate the base in which the particular number is expressed. This convention is used to avoid confusion whenever more than one number system is being employed.

Binary Counting

The largest digit value in the binary system is 1. Thus, when counting in binary, a given digit will progress from 0 to 1. After it reaches 1, it recycles back to 0 and the next higher bit position is incremented:

Binary (Weights →)				Decimal equivalent
$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$	
			0	0
			1	1
		1	0	2
		1	1	3
	1	0	0	4
	1	0	1	5
	1	1	0	6
	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15
			↑ LSB	

Note in this example that the least-significant-bit (LSB) position changes value at each step in the counting sequence. The next higher bit (2^1) changes value every two counts, the 2^2 bit changes value every four counts, and so on.

In the binary system, using N bits, we can count through 2^N different numbers, including zero. For example, with 2 bits, we can count 00, 01, 10, 11 for four different numbers. Similarly, with 4 bits, we can count from 0000 up to 1111, a total of $2^4 = 16$ different numbers. The largest number that can be represented by N bits is always equal to $2^N - 1$ in decimal. Thus, with 4 bits, the largest binary number is 1111_2, which is equivalent to $2^4 - 1 = 15_{10}$.

Binary-to-Decimal Conversion

As explained earlier, the binary number system is a positional system where each bit carries a certain weight based on its position relative to the binary point. Any binary number can be converted to its decimal equivalent simply by summing together the weights of the various positions in the binary number which contain a 1. To illustrate:

$$\begin{aligned} &1 \quad 1 \quad 0 \quad 1 \quad 1 \ \text{(binary)} \\ &2^4 + 2^3 + 0 + 2^1 + 2^0 = 16 + 8 + 2 + 1 \\ &= 27_{10} \ \text{(decimal)} \end{aligned}$$

The same method is used for binary numbers which contain a fractional part.

$$\begin{aligned} 1\ 0\ 1\,.\,1\ 0\ 1 &= 2^2 + 2^0 + 2^{-1} + 2^{-3} \\ &= 4 + 1 + .5 + .125 \\ &= 5.625_{10} \end{aligned}$$

The following conversions should be performed and verified by the reader:

1. $1\ 0\ 0\ 1\ 1\ 0_2 = 38_{10}$.
2. $0\,.\,1\ 1\ 0\ 0\ 0\ 1_2 = 0.765625_{10}$.
3. $1\ 1\ 1\ 1\ 0\ 0\ 1\ 1\,.\,0\ 1\ 0\ 1_2 = 243.3125_{10}$.

There are several ways to convert a decimal number to its equivalent binary system representation. A method that is convenient for small numbers is just the reverse of the process described in the preceding section. The decimal number is simply expressed as a sum of powers of 2 and then 1s and 0s are written in the appropriate bit positions. To illustrate:

$$\begin{aligned} 13_{10} = 8 + 4 + 1 &= 2^3 + 2^2 + 0 + 2^0 \\ &= 1\quad 1\quad 0\quad 1_2 \end{aligned}$$

Another example:

$$\begin{aligned} 25.375_{10} &= 16 + 8 + 1 + .25 + .125 \\ &= 2^4 + 2^3 + 0 + 0 + 2^0 + 0 + 2^{-2} + 2^{-3} \\ &= 1\quad 1\quad 0\quad 0\quad 1\ .\ 0\quad 1\quad 1_2 \end{aligned}$$

For larger decimal numbers, the method above is laborious. A more convenient method entails separate conversion of the integer and fractional parts. For example, take the decimal number 25.375, which was converted above. The first step is to convert the integer portion, 25. This conversion is accomplished by repeatedly *dividing* 25 by 2 and writing down the remainders after each division until a quotient of zero is obtained.

$$\frac{25}{2} = 12 + \text{remainder of } 1$$

$$\frac{12}{2} = 6 + \text{remainder of } 0$$

$$\frac{6}{2} = 3 + \text{remainder of } 0$$

$$\frac{3}{2} = 1 + \text{remainder of } 1$$

$$\frac{1}{2} = 0 + \text{remainder of } 1_{\text{MSB}}$$

LSB

$$25_{10} = \boxed{1\quad 1\quad 0\quad 0\quad 1_2}$$