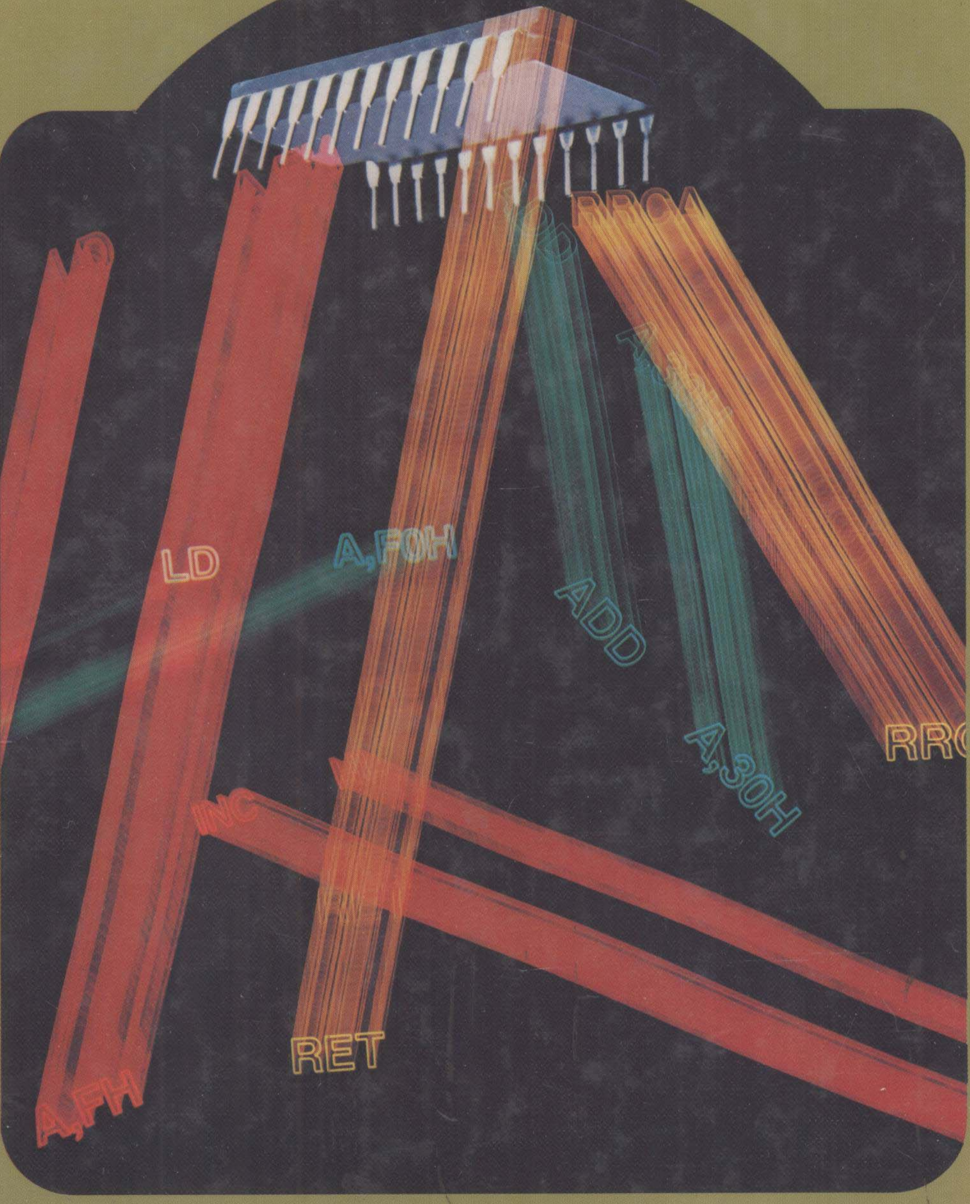
# MICROPROCESSING FUNDAMENTALS
## HARDWARE AND SOFTWARE



Ramirez/Weiss

# Microprocessing Fundamentals

## Hardware and Software

Edward V. Ramirez
Grumman Aerospace Corporation

Melvyn Weiss
Hughes Aircraft Company

McGRAW-HILL BOOK COMPANY
GREGG DIVISION

New York Atlanta Dallas St. Louis San Francisco
Auckland Bogotá Düsseldorf Johannesburg
London Madrid Mexico Montreal
New Delhi Panama Paris
São Paulo Singapore
Sydney Tokyo
Toronto

Sponsors: George Horesta and Mark Haas
Editing supervisor: Alice V. Manning
Design supervisor: Karen T. Mino
Art supervisor: Howard Brotman
Production supervisor: Kathleen Morrissey
Cover photographer: Martin Bough/Studios, Inc.

**Library of Congress Cataloging in Publication Data**

Ramirez, Edward V
Microprocessing fundamentals.

Includes index.
1. Microprocessors. I. Weiss, Melvyn, joint author.
II. Title.
QA76.5.R299 1980 001.6′4′04 79-17922
ISBN 0-07-051172-1

**Microprocessing Fundamentals: Hardware and Software**

Copyright © 1980 by McGraw-Hill, Inc. All rights reserved. Printed in the United States of America. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher.

1 2 3 4 5 6 7 8 9 0 DODO 7 8 6 5 4 3 2 1 0 9

# Preface

This book provides an introductory treatment of the microprocessor. Major topics covered are the basic elements of a microprocessor, programming fundamentals, the microprocessor interface, and microprocessor applications. With the growing number of microprocessors entering the field, an attempt was made to use a generic microprocessor rather than emphasizing any particular one. While this book does not contain any specification sheets or analyses of present-day microprocessors, it will provide you with a sound background in microprocessor fundamentals. The subject matter does not require any background in digital technology; however, a basic knowledge of this subject may be beneficial.

Chapter 1 deals with the basics of digital technology and should be helpful to readers who are unfamiliar with digital techniques. Chapter 2 reviews the basic elements of digital computer systems, stressing the fundamental operations that must take place at the functional level. In Chap. 3 you are introduced to microprocessor concepts. Chapter 4 is devoted to technology and the functional operation of solid-state memories.

Chapter 5 covers programming fundamentals with the concepts of instructions and addressing modes being presented. The "generic instruction set," a group of 30 basic instructions, is described, with illustrative examples to help you to understand the concepts of addressing modes and instructions. An understanding of the generic instruction set will provide you with the knowledge to program any available microprocessor.

Chapter 6 is devoted to software, with machine, assembly, and high-level languages described. The procedure describing how software is developed and the tools for accomplishing this are included. Chapter 7 introduces flowcharts and describes, step by step, the program instructions required to implement the numerous flowcharting examples.

An insight into microcomputers and bit-slice microprocessor units is presented in Chap. 8. Chapter 9 discusses interfaces, an important aspect in understanding microprocessors. Data transfer techniques and input-output

# Contents

此为试读,需要完整PDF请访问: www.ertongbook.com

# *Introduction*
# Evolution of Microprocessors

The development of microprocessors in the 1970s represents a significant advance in electronics. The microprocessor's popularity stems from many factors, such as its small size, its low power needs, its small parts count compared with hard-wired logic, and, perhaps most important of all, its low cost. Because of the microprocessor's small size and low cost, consumers and small businesses can now have in a handful of chips as much computational power as computers that in 1970 were affordable only to big users of data processing systems.

In the early 1960s a single transistor used for digital logic circuitry sold for an average price of about 5 dollars. In the late 1970s the same 5 dollars could purchase a single microprocessor chip; however, this chip had 10,000 transistors and included all the necessary resistors and intracomponent connections—a truly remarkable technological achievement.

To a large extent we owe the successful development of microprocessors to the needs of the military and the space program, which led manufacturers to develop miniaturized electronic circuits known as *microelectronics*. The need was for systems with reliable performance, low power dissipation, light weight, and small volume. These goals required small circuits. Semiconductor manufacturers were able to increase the circuit density of a basic chip by using simpler processes and by reducing the pattern mask sizes. This trend toward higher-density packaging continued through the 1960s. However, the success of microprocessors is really attributable to the commercial market, whose demands have created a high level of production and a subsequent reduction in the selling price of microprocessors.

A microprocessor can be defined as a single integrated circuit consisting of thousands of digital gates which perform the arithmetic, logic, and control functions of a general-purpose computer. It is a member of the family of large-scale integrated circuits, which reflect the present state of a trend toward miniaturization that began with the development of the transistor in the late 1940s.

As the semiconductor industry continues to make lower-cost chips with more functions, it is only natural for it to attempt higher levels of circuit integration. Microprocessors require external input-output circuits and external memory to functionally operate as computers. The trend is to incorporate these circuits into the same chip as the microprocessor, thereby forming a computer on a single chip. This configuration is called a *microcomputer.* Integrated into an area of less than 40,000 square mils ($\text{mil}^2$) are a memory, input-output interface circuits, and the arithmetic, logic, and control functions of a microprocessor. As new classes of these devices become available, the use of microprocessors and microcomputers will broaden.

In the mid-1960s, when medium-scale integration (MSI, with 50 to 400 transistors per chip) technology appeared, it was widely used to make smaller computers than those that had previously existed. Minicomputers, as these computers were called, were intended for a specific, limited market. Minicomputers were designed to capture (as they in fact did) the low end of performance, such as controller functions, data acquisition, and displays formerly handled by large computers.

Minicomputer advantages included fast processing rates, relatively short word lengths, and versatile input-output structure. Also an important factor in the "mini" success was the small physical size achieved by the use of MSI. Large computers, called *main frame* units, using MSI parts soon followed. Cost was probably the most important factor in the success of the minicomputer. In the mid-sixties minicomputers were selling for about $25,000; by the late 1970s they were selling for under $10,000. This low price, coupled with modest performance, guaranteed minicomputers a place in the computer market.

The data processing spectrum is so wide that microprocessors and microcomputers have also captured part of the available market and even created new markets. The microprocessor will capture many of the low-end applications presently being performed by minicomputers.

Another trend being observed is in the increased level of complexity found in newer microprocessors. Circuits such as timer/counters and drivers, normally found external to the chip in older microprocessors, are now being integrated into the chip. When the first microprocessor was developed it was a 4-bit unit with modest density. Other units developed since have increased bit length and density. By the late 1970s møst microprocessors were using 8 bits per word, with newer units having 16 bits per word. It is reasonable to forecast that with the emphasis presently being placed on 8-bit microcomputers and 16-bit microprocessors, dramatic price reductions for single-chip 16-bit microcomputers will be a reality.

# Basic Digital Techniques

The microprocessor is a product of large-scale integration (LSI) technology, which uses digital techniques for its input, output, and internal structure. This chapter presents a review of number systems and digital integrated circuit (IC) building blocks.

The decimal number system which we use has as its base 10, since each digit position can contain any number from 0 to 9, a total of 10 different numbers. Early attempts to design decimal computers provided little fruitful result, as it is difficult to reliably represent 10 different states. A more satisfactory approach utilizing only two levels was adopted, with logic 1 representing one level and logic 0 representing the other level. In order to understand the binary system, it is first necessary to fully understand the familiar decimal (base 10) number system.

## 1-1 DECIMAL NUMBER SYSTEM

The number 4385 is read as "four thousand three hundred eighty-five"; we start with the most significant digit at the extreme left. A method of writing 4385 is

$$4385_{10} = 4 \times 10^3 + 3 \times 10^2 + 8 \times 10^1 + 5 \times 10^0$$

The digit position at the extreme right is the one of least value. The value of each digit position to the left increases by a power of 10.

When we add two numbers in the decimal number system, the rules are to first add the least-value digits and then the next higher-value digits, etc.

此为试读,需要完整PDF请访问: www.ertongbook.com

Should the sum of any two digits exceed 10, a carry is generated into the next higher digit position. For example,

$$\begin{array}{r} 086 \\ +\underline{057} \end{array}$$

The addition is performed as

$$\begin{aligned} 6 + 7 &= 3 + \text{“carry 1”} \\ 8 + 5 + 1 \text{ (previous carry)} &= 4 + \text{“carry 1”} \\ 0 + 0 + 1 \text{ (previous carry)} &= 1 + \text{“carry 0”} \\ \text{Answer} &= 143 \end{aligned}$$

In subtraction, the subtrahend is subtracted from the minuend. The subtraction rules state that if the minuend digit is less than the subtrahend digit then the base 10 is added to the minuend and the next subtrahend digit to the left has a "borrow 1" added to it. For example:

$$\begin{array}{rl} 86 & \text{minuend} \\ -\underline{57} & \text{subtrahend} \end{array}$$

The subtraction is performed as

$$\begin{aligned} &6 - 7 = 10 \text{ (1 borrow)} + 6 - 7 = 16 - 7 = 9 \\ &8 - 5 = 8 - (5 + \text{“borrow 1”}) = 8 - 6 = 2 \\ &\text{Answer} = 29 \end{aligned}$$

A more significant method of subtracting is the 9's or the 10's complement system.

## The 9's and 10's Complements

Nine's and ten's complements can be used to convert subtraction operations to additions.

To convert to the 9's complement, each decimal digit is subtracted from 9. For example, Find the 9's complement of 37, 45, and 29.

$$\begin{array}{r} 99 \\ -\underline{37} \\ 62 \end{array} \qquad \begin{array}{r} 99 \\ -\underline{45} \\ 54 \end{array} \qquad \begin{array}{r} 99 \\ -\underline{29} \\ 70 \end{array}$$

9's complement of 37 = 62

9's complement of 45 = 54

9's complement of 29 = 70

In 9's complement subtraction, the subtrahend is complemented and then added to the minuend. The carry generated from the highest digit position is not used to form a new digit position but instead is added to the sum under the original digit position.

EXAMPLES

Subtract 17 from 84

9's complement of 17 = 82

$$\begin{array}{lr} & 84 \\ + & \underline{82} \\ & 166 \\ \text{carry} & \underline{1} \\ & 67 \end{array}$$

Subtract 3 from 75

9's complement of 03 = 96

$$\begin{array}{lr} & 75 \\ + & \underline{96} \\ & 171 \\ \text{carry} & \underline{1} \\ & 72 \end{array}$$

To convert to the 10's complement system from the 9's complement, 1 is added to the 9's complement value.

EXAMPLE

Determine the 10's complement of 12, 44, and 73.

9's complement of 12 = 87, 10's complement of 12 = 88

9's complement of 44 = 55, 10's complement of 44 = 56

9's complement of 73 = 26, 10's complement of 73 = 27

In 10's complement subtraction, the subtrahend is first converted to its 10's complement and then added to the minuend. The resultant carry generated from the highest digit position does not form a new digit position but instead is ignored.

EXAMPLE 1

Subtract using 10's complement arithmetic

$$\begin{array}{r} 86 \\ \underline{-57} \end{array}$$

subtrahend = 57

9's complement = 42

10's complement = 42 + 1 = 43

$$\begin{array}{lr} & 86 \\ + & \underline{43} \\ & 129 \end{array}$$

Ignore carry

Answer = 29

EXAMPLE 2
Subtract using 10's complement arithmetic

$$\begin{array}{r} 456 \\ -\underline{308} \end{array}$$

subtrahend = 308
9's complement = 691
10's complement = 691 + 1 = 692

$$\begin{array}{r} 456 \\ +\underline{\ 692} \\ 1148 \end{array}$$

Ignore carry

Answer = 148

## 1-2 THE BINARY NUMBER SYSTEM

Digital systems, in contrast to analog systems have only two states, On and Off. These two conditions can be represented by the opening and closing of a switch or in common transistor-transistor logic (TTL) by the 1 state ($4\ V \pm 1\ V$) and the 0 state ($0.2\ V \pm 0.2\ V$). The logic 1 state is frequently specified as 2.4 V minimum and the logic 0 state as 0.8 V maximum. Since a digital system has only two states, it can be readily represented by the binary system, which has only two notations, 0 and 1. The binary system is the base 2 system, in contrast to the decimal system, which is the base 10 system. The principle behind representing any decimal number in binary is positional notation. That is, all digit positions to the left of the point, which in the binary number system is called the binary point, have ascending powers of 2. The first, or least significant, digit to the left of the binary point has a $2^0$ value. The next digit to the left has a $2^1$ value or weight assigned to its position. Each subsequent position to the left has its weight value, in powers of 2, increased by 1 over the previous digit. A four-digit number will have the digit weight assignment $2^3 2^2 2^1 2^0$, where any digit if it is 1 will assume the weight of its position. The binary number 1010110 can be converted to its decimal equivalent value as follows:

$$\begin{aligned} 1010110 &= 1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 \\ &= 1 \times 64 + 0 + 1 \times 16 + 0 + 1 \times 4 + 1 \times 2 + 0 \\ &= 86_{10} \end{aligned}$$

Table 1-1 represents the equivalent binary numbers of decimal numbers 0 through 15. A table of powers of 2 is found in Appendix A. Note that the

TABLE 1-1

| DECIMAL NUMBER | BINARY EQUIVALENT | DECIMAL NUMBER | BINARY EQUIVALENT |
|---|---|---|---|
| 0 | 0000 | 8 | 1000 |
| 1 | 0001 | 9 | 1001 |
| 2 | 0010 | 10 | 1010 |
| 3 | 0011 | 11 | 1011 |
| 4 | 0100 | 12 | 1100 |
| 5 | 0101 | 13 | 1101 |
| 6 | 0110 | 14 | 1110 |
| 7 | 0111 | 15 | 1111 |

number 15 has 1s in all digit positions. This implies that with four digits the highest value that can be represented is 15. This can be expressed mathematically as

$$\text{Highest decimal value} = 2^n - 1$$

where $n$ is the number of binary digits. With eight digits, the highest decimal number that can be expressed is 255; i.e., $2^8 - 1 = 256 - 1 = 255$.

## Bits and Other Terms

A *b*inary dig*it* (a 1 or a 0) is called a *bit.* A group of 4 consecutive bits within a computing system is called a *nibble,* while 8 consecutive bits make up a *byte.* Most microprocessing systems have an 8-bit internal structure, where transfers between the elements are done in parallel and with 8 bits. The term commonly used to describe a group of 8 bits that function together as a unit is the *word.* A word in a typical microprocessing system is a byte (8 bits); however, in larger systems a word can be 16 bits. Large data processing systems use words of 32 or 64 bits. This represents 4 to 8 bytes, or twice this number of nibbles.

In any data word organization, the bit furthest from the least significant bit (LSB) is called the most significant bit (MSB). In computer terminology both the LSB and the MSB digits are usually specifically referred to as such.

## Binary Code Applications

Suppose it is necessary to represent the numbers 0 through 9 (10 characters) and the uppercase and lowercase alphabet (52 characters) in binary codes; this would require a total of 62 characters. Since $2^5 < 62 < 2^6$, a minimum of 6 bits is required. These 6 bits form a word, with each word capable of

TABLE 1-2 Binary Representation of the Character Code

| B5 (MSB) | B4 | B3 | B2 | B1 | B0 (LSB) | WORD NUMBER | CHARACTER |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | A |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | B |
| 0 | 0 | 0 | 0 | 1 | 0 | 2 | C |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| 0 | 1 | 1 | 0 | 0 | 1 | 25 | Z |
| 0 | 1 | 1 | 0 | 1 | 0 | 26 | a |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| 1 | 1 | 0 | 0 | 1 | 1 | 51 | z |
| 1 | 1 | 0 | 1 | 0 | 0 | 52 | 0 |
| 1 | 1 | 0 | 1 | 0 | 1 | 53 | 1 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

defining a character. The binary format of these characters could be as shown in Table 1-2, with B0 through B5 representing the 6 binary bits. When this binary format is transmitted to an 8-bit microprocessor, each byte represents a character plus 2 spare bits, bits B6 and B7. These spare bits may be used to verify that no bits were lost in the transmission to the microprocessor and to tell the microprocessor that the transmitted character is valid.

A self-checking scheme which can enhance the validity of the transmitted data can be implemented by the use of *parity*. Parity is a method by which the total number of binary 1s (or 0s) is always even or always odd—no matter which character is transmitted. When the sum of the binary 1s in a word is odd, this is called *odd parity*. If the sum is even, this is called *even parity*. Table 1-3 shows a revised word structure using the entire 8 bits. The microprocessor using this word structure can test bit B6 to see if the character is valid, and after determining validity it can verify proper transmission.

## Positive Logic vs. Negative Logic

A common misconception is that negative logic refers to negative voltage. This is not true. Positive logic in TTL means simply 4 V ± 1 V = logic 1 and 0.2 V ± 0.2 V = logic 0. In negative logic 4 V ± 1 V = logic 0 and 0.2 V ± 0.2 V = logic 1. In positive logic the word True refers to 4 V ± 1 V and False 0.2 V ± 0.2 V, while in negative logic the word True means 0.2 V ± 0.2 V and False means 4 V ± 1 V. If an interface is TTL and uses negative logic, a True signal will be 0.2 V ± 0.2 V. Figure 1-1 shows positive-logic and negative-logic waveshapes.

TABLE 1-3 Binary Representation of the Character Code Using B7 as Parity (Odd) and B6 as Validity*

| PARITY (ODD) (MSB) B7 | VALIDITY B6 | B5 | B4 | B3 | B2 | B1 | (LSB) B0 | WORD NUMBER | CHARACTER |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | A |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | B |
| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 2 | C |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 25 | Z |
| 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 26 | a |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 51 | z |
| 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 52 | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 53 | 1 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 61 | 9 |

*1 = VALID
0 = INVALID

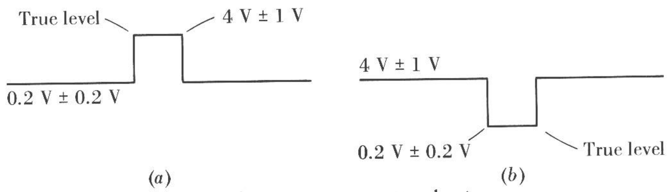


**Figure 1-1** Positive logic vs. negative logic. (*a*) Positive logic; (*b*) negative logic.

## Decimal-to-Binary Conversion

A method of determining the binary equivalent of a decimal number is to successively divide by 2 and use the remainders as the answer, the first remainder being the LSB, and so on.

EXAMPLE
Determine the binary equivalent of $57_{10}$.

$$2\overline{)57}\quad 28 \qquad R = 1\ (\text{LSB})$$

$$2\overline{)28}\quad 14 \qquad R = 0$$

此为试读，需要完整PDF请访问：www.ertongbook.com