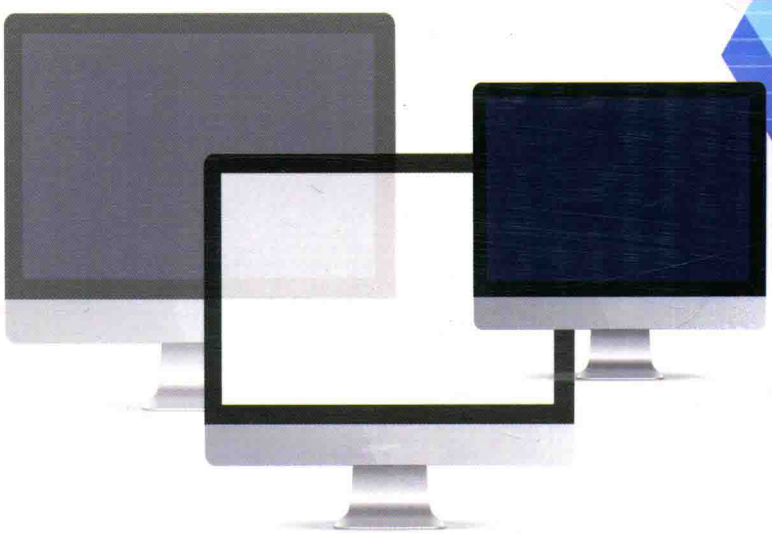
The Principle and Application of Microcomputer

微机原理及应用

■ 丁艳 编著



国防工业出版社
National Defense Industry Press

微机原理及应用

The Principle and Application of Microcomputer

丁艳　编著

国防工业出版社

·北京·

内容简介

本书以16位微处理器为核心，全面讲述了微型计算机的基本组成、工作原理以及硬件接口技术。全书共8章，逐一讲述了计算机基础知识、微型计算机的基本组成及工作原理、16位微处理器8086/8088CPU的寻址方式、指令系统、汇编程序的设计、输入/输出接口、存储器系统和可编程定时/计数控制器等内容。本书是作者多年教学经验的结晶，内容编排合理，由浅入深，体系完整，重点突出并配有丰富的例题及详尽的注释。

本书采用全英文编写，并配有部分汉语注释，适合用作普通高等院校非计算机类各专业学生“微型计算机原理及应用”课程的教材，也可用作成人高等教育的培训教材及广大科技工作者的参考书。

图书在版编目(CIP)数据

微机原理及应用 = The Principle and Application of Microcomputer：英文/丁艳编著. —北京：国防工业出版社，2016.1

ISBN 978-7-118-10410-3

Ⅰ. ①微… Ⅱ. ①丁… Ⅲ. ①微型计算机-英文 Ⅳ. ①TP36

中国版本图书馆CIP数据核字(2015)第234161号

※

国防工业出版社出版发行

(北京市海淀区紫竹院南路23号 邮政编码100048)

天利华印刷装订有限公司印刷

新华书店经售

*

开本 787×1092 1/16 印张 21¼ 字数 490千字

2016年1月第1版第1次印刷 印数 1—2500册 定价 58.00元

国防书店：(010)88540777 发行邮购：(010)88540776

发行传真：(010)88540755 发行业务：(010)88540717

前　言

从第一台计算机诞生至今，微型计算机技术得到了迅速发展。为了满足高素质人才培养的需求，在高等院校中，很多专业都开设了微型计算机原理及应用的相关课程。

本书编写的主要目的是为高等院校非计算机专业“微型计算机原理及应用”课程提供教材，为了满足双语教学的需求，本书采用全英文编写。考虑到非计算机专业学时的限制、本门课程的定位以及学生对语言理解的精准度，本书仍然以 IBM PC/XT 或 AT 机型为背景机，以 16 位微处理器为核心，全面讲述微机系统的组成、工作原理、硬件接口技术和典型应用。在此基础上，帮助学生系统掌握汇编语言程序设计基本方法和微机硬件接口技术，建立微机系统的整体概念，以使其具有一定的微机软件及硬件初步开发、设计的能力。

全书分为 8 章。第 1 章介绍计算机中的数制、逻辑电路等基础知识。第 2 章介绍微型计算机的系统组成及基本组成电路，包括算术逻辑部件（ALU）、触发器、寄存器及存储器的基本原理及其符号等，同时以一个简化计算机为例剖析了微型计算机的工作原理。第 3 章以 8086/8088CPU 为例重点介绍 16 位微处理器的结构、工作模式及操作功能。第 4 章介绍 8086/8088CPU 的寻址方式及指令系统。第 5 章介绍微型计算机的汇编语言及汇编程序。第 6 章介绍微型计算机的程序设计步骤及基本的程序形式。第 7 章介绍微型计算机存储器分类及存储器扩展技术。第 8 章介绍输入/输出接口及其编程方法，并对典型的串行通信接口芯片、并行接口芯片及可编程定时/计数控制器进行介绍。

在内容的取舍及编排上，我们从非计算机专业的特点出发，在保证体系完整的前提下，力求深入浅出，循序渐进，适用实用，系统与原理并重，既有利于教师组织课程教学、实验教学，又便于学生自学。为了保证学生的学习效果，本书对部分难理解的知识点给出了汉语注释。

本书的编写过程中，我们参阅了大量的资料，在此谨向引为本书内容和作为本书参考资料的各位作者、译者表示由衷的感谢。在本书的成稿过程中，薛斌、王建林、杨慧蕾、王晓熔、李浩、袁瑜键、胡雪梅、李佳等同学做了大量的文字校对工作，在此向他们致以诚挚的谢意。

由于编者水平有限，本书中错误和不当之处在所难免，敬请各位读者和专家批评指正。

编者

CONTENTS

Chapter 1 Fundamentals of Computer

1.1 Number System

Unlike human counting habits, computer memory does not store numbers in decimal (base 10). In order to make computers more reliable and easier to build, they are based on devices that can take on only two states, one of which is denoted by 0 and the other by 1. All information in computers is stored in a binary (base 2) format because it greatly simplifies the hardware.

This section is the basis of the follow-up study. It is helpful for those who are unfamiliar with number systems. Some basic concepts and conversions between decimal and binary, decimal and hexadecimal, and binary and hexadecimal are introduced.

1.1.1 Basic Number Systems

Nonnegative integers are normally represented by choosing a number **x**, called the **base**, and **x** different symbols, called **digits**, and then using a string of digits to indicate the number. Although a base of 10 is what is used in our everyday work, bases of 2 and 16 are commonly encountered when working with computers.

The number systems based of 10, 2, and 16, which are most common used, are **called decimal**, **binary**, and **hexadecimal** (base 16) **respectively**.

1. Decimal

When we began to learn to count, especially when we entered the primary school, we learned that a **decimal** (base 10) number was constructed with **10** digits: **0** through **9**. Note that the first digit in 10 numbering system is a 0 and the last one is a 9.

In a decimal number system, large numbers are constructed by using **positional notation**. For example, the number 19 is a compilation of two digits in a certain order. The digit 9 is the units position of this numeral and the 1 is the tens position. As shown in **Example 1-1**, the position to the left of the units position was the tens position, the position to the left of the tens position was the hundreds position, and so forth. It also uses the exponential value of each position to express. The units position has a weight of 10^0, or 1; the tens position has weight of 10^1, or 10; and the hundred position has a weight of 10^2, or 100.

【Example 1-1】

Number	3	2	8.	8	0	6
Power	10^2	10^1	10^0	10^{-1}	10^{-2}	10^{-3}
Weight	100	10	1	0.1	0.01	0.001
Numeric Value	300	20	8	0.8	0	0.006

Value $300+20+8+0.8+0+0.006=328.806$

2. Binary

Base 2 numbers are composed of **2** possible digits (**0 and 1**). Each digit of a number has a power of 2 associated with it based on its position in the number, just like **Example 1-2.**

【Example 1-2】

Number	1	1	1	0	0	1
Power	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}
Weight	4	2	1	0.5	0.25	0.125
Numeric Value	4	2	1	0	0	0.125

Value $4+2+1+0+0+0.125=7.125$

Example 1-2 shows a binary number. Its radix is 2. So its position to the left of the binary point is 2^0, or 1.

In fact, in any number systems, the position to the left of the radix (number base) point is always the units position and the position to the left of the units position is always the number base raised to the first power. As shown in **Example 1-1**, this is 10^1, or 10. In **Example 1-2**, it is 2^1, or 2.

3. Hexadecimal

Hexadecimal numbers use **base 16**. Hexadecimal (or hex for short) can be used as shorthand for binary numbers. Hex has **16** possible digits in which letters are used for the extra digits after 9. It means that the 16 hex digits are **0-9** then followed by **A** for 10, **B** for 11, **C** for 12, **D** for 13, **E** for 14 and **F** for15. Each digit of a hex number has a power of 16 associated with it.

4. Other number systems

Besides the common number systems introduced above, the number system corresponding to the base of 8 which is called **octal** is also used in computers sometimes.

The real numbers often encountered in many high-level languages using the Intel family of microprocessors. A real number, or a **floating-point number** as it is often called, contains two parts: a mantissa significant, or fraction; and an exponent. **Figure 1-1** illustrates both the 4-byte and 8-byte forms of real numbers stored in any Intel system. Note that the 4-byte real number is called **single-precision** and the 8-byte form is called **double-precision**.

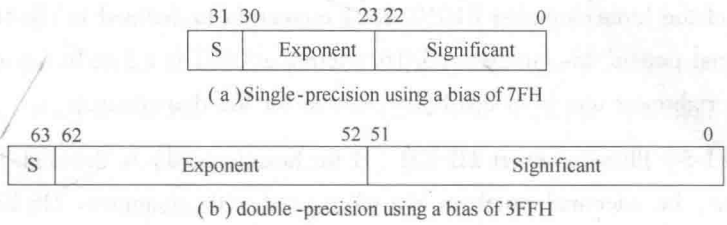


Figure 1-1 The floating-point numbers

1.1.2 Conversions between Different Number Systems

After understanding the basic concept of a number system, the conversions between different number base systems will be learned in this section.

1. Conversion to Decimal

To convert to decimal, we need to determine the weights of values of each position of the number, and then sum the weights to form the decimal equivalent. **Example 1-2** illustrates the conversion from binary number base to decimal, while **Example 1-3** gives the conversion from octal.

【Example 1-3】 Please convert 205.6_8 octal to decimal.

To accomplish this conversion, we need to write down the weight of each position of the number and then multiply them by the digit of each position.

Number	2	0	5.	6
Power	8^2	8^1	8^0	8^{-1}
Weight	64	8	1	0.125
Numeric Value	128	0	5	0.75

Value 128 + 0 + 5 + 0.75 = 133.75

As shown in expression $2\times64+0\times8+5\times1+6\times0.125$, the value of 205.6_8 is 133.75 decimal.

Note that the weights of position which immediately to the right of the octal point is 1/8, or 0.125. The next position is 1/64. So the number in **Example 1-3** can also be written as the decimal number $133^6/_8$.

【Example 1-4】 Please complete the conversion from a binary number 01010.0101 to decimal.

As we did in prior example, first we should write down the weights and powers of each position, and then sum the numeric values together.

Number	0	1	0	1	0.	0	1	0	1
Power	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
Weight	16	8	4	2	1	0.5	0.25	0.125	0.0625
Numeric Value	0	8	0	2	0	0	0.25	0	0.0625

Value 0 + 8 + 0 + 2 + 0 + 0 + 0.25 + 0 + 0.0625 = 10.3125

The value of the binary number 01010. 0101 converted to decimal is 10. 3125.

The fractional part of this number is 5/16. Note that 0101 is a 5 in binary code for the numerator and the rightmost one is in the 1/16 position for the denominator.

[Example 1-5] Please convert 2B. EH (H for hexadecimal) to decimal.

As we know, hexadecimal numbers are often used with computers 2B. EH is illustrated with its weights as follows.

Number	2	B.	E
Power	16^1	16^0	16^{-1}
Weight	16	1	0. 0625
Numeric Value	32	11	0. 875

Value 32 + 11 + 0. 875 = 43. 875

The sum of its digits is 43. 875, or $43^7/_8$. The whole number part is represented with 2 × 16 plus 11 (B) ×1. The fraction part is 14 (E) as a numerator and 16 (16^{-1}) as the denominator, or 14/16, which is reduced to 7/8.

2. Conversion from Decimal

Conversions from decimal to other number systems are more difficult to accomplish than that conversions to decimal. To convert the whole number portion of a number from decimal, we should divide them by the radix. To convert the fractional portion, we should multiply them by the radix.

1) Whole Number Conversion from Decimal

To convert a decimal whole number to another number system, we should divide it by the radix and save the remainders as significant digits of the result. An algorithm for this conversion is as follows:

(1) Divide the decimal number by the radix (number base).

(2) Save the remainder (first remainder is the least significant digit).

(3) Repeat steps 1 and 2 until the quotient is zero.

That is to say, if we want to convert decimal number to any other number systems, what we only need to do is dividing the decimal number by the corresponding base. **For example**, to convert a 15 decimal to binary, divide it by 2. To convert a decimal into base 8, divide it by 8. Conversion from decimal to hexadecimal is accomplished by dividing it by 16.

[Example 1-6] Please convert a 15 decimal to a binary.

2)15 remainder = 1 ↑

2) 7 remainder = 1 ↑

2) 3 remainder = 1 ↑

2) 1 remainder = 1 ↑

0 result = 1111

Above Example shows the conversion process from a 15 decimal to binary. Dividing 15 by 2, the quotient is 7 with a remainder of 1. The first remainder 1 is the unit position of the conversion result. Then divide the 7 by 2 with a quotient of 3 and a remainder of 1. The 1 is the value of the two's (2^1) position. Continue the division until the quotient is a zero. The result is written as 1111_2, from the bottom to the top.

【Example 1-7】 Please convert a 10 decimal to an octal.

$$\begin{array}{rll} 8)\underline{10} & \text{remainder} = 2 & \uparrow \\ 8)\underline{1} & \text{remainder} = 1 & \uparrow \\ 0 & \text{result} = 12 & \end{array}$$

The base of an octal is 8, so divide 10 by 8 and follow the rules. As shown in **Example 1-7**, a 10 decimal is a 12 octal.

【Example 1-8】 Please convert a 109 decimal to a hexadecimal.

Conversion from 109 decimal to hexadecimal is accomplished by dividing it by 16.

$$\begin{array}{rll} 16)\underline{109} & \text{remainder} = 13\ (\text{D}) & \uparrow \\ 16)\underline{6} & \text{remainder} = 6 & \uparrow \\ 0 & \text{result} = 6\text{D} & \end{array}$$

The remainder will range from 0 through 15. Notice that any remainder of 10 to 15 is then converted to the letter A to F as the hexadecimal number. **Example 1-8** shows the decimal number 109 converted to a 6DH.

2) Conversion from a Decimal Fraction

Conversion from a decimal fraction to another number base is accomplished with multiplication by the radix. **For example**, to convert a decimal fraction to binary, multiply it by 2. After the multiplication, the whole number portion of the result is saved as a significant digit of the result, and the fractional remainder is again multiplied by the radix. When the fraction remainder is zero, multiplication ends. Note that some numbers are never ending. That is, a zero is never a remainder. An algorithm for conversion from a decimal fraction is as follows:

(1) Multiply the decimal fraction by the radix (number base).

(2) Save the whole number portion of the result (even if zero) as a digit. Note that the first result is written immediately to the right of the radix point.

(3) Repeat step 1 and 2, using the fractional part of step 2 until the fractional part of step 2 is zero.

【Example 1-9】 Please convert a 0.125 decimal to binary.

Because the radix of binary is 2, the conversion is accomplished with multiplications by 2 and the multiplication continues until the fractional remainder is zero.

$$
\begin{array}{rll}
 & 0.125 & \\
\times & \quad 2 & \\
\hline
 & 0.25 & \text{digit is } 0 \downarrow \\
\times & \quad 2 & \\
\hline
 & 0.5 & \text{digit is } 0 \downarrow \\
\times & \quad 2 & \\
\hline
 & 1.0 & \text{digit is } 1 \downarrow
\end{array}
$$

The result is written as 0. 001 binary.

As illustrated in **Example 1-9**, the whole number portions are written as the binary fraction (0. 00).

Likewise, the same technique is used to convert a decimal fraction into any other number bases. The examples given below show that how to convert a decimal fraction to an octal or a hexadecimal fraction respectively.

【**Example 1-10**】 Please convert a 0. 125 decimal to octal.

$$
\begin{array}{rll}
 & 0.125 & \\
\times & \quad 8 & \\
\hline
 & 1.0 & \text{digit is } 1 \downarrow
\end{array}
$$

Multiplying with an 8, we can get 0. 125 decimal fraction is 0. 1 octal.

【**Example 1-11**】 Please convert a 0. 046875 decimal to hexadecimal.

Similarly, a decimal 0. 046875 is converted to hexadecimal by multiplying it by the radix 16. As shown in **Example 1-11**, the conversion of 0. 046875 is a 0. 0CH.

$$
\begin{array}{rll}
 & 0.046875 & \\
\times & \quad 16 & \\
\hline
 & 0.75 & \text{digit is } 0 \downarrow \\
\times & \quad 16 & \\
\hline
 & 12.0 & \text{digit is } 12\ (C) \downarrow
\end{array}
$$

The result is written as 0. 0C hexadecimal.

3. Conversion between Binary and Hexadecimal

Conversion of a binary number to a hexadecimal number is a simply matter of putting the binary digits in groups of four. Similarly to convert a hex number to binary, we can simply convert each hex digit to a 4-bit binary number.

【**Example 1-12**】 Please convert a hexadecimal number 14D to a binary number.

Hexadecimal Number	1	4	D
Binary Number	0001	0100	1101

Note that the leading zeroes of the 4-bits are important. Converting from binary to hex is just as easy. One does the process in reverse. Convert each 4-bit segments of the binary to a hex. Start from the right end of the binary number to ensure conversion correct. **For example**:

【**Example 1-13**】Please convert a binary number to a hexadecimal number.

Binary Number	0010	0000	0101	1010	0111	1110	0100
Hexadecimal Number	2	0	5	A	7	E	4

A 4-bit number is called a **nibble**. Thus each hex digit corresponds to a nibble. Two nibbles make a byte and so a 2-digit hex number can represent a byte. A byte's value ranges from 0 to 11111111 in binary, 0 to FF in hex and 0 to 255 in decimal.

Table 1-1 shows the decimal numbers 0 through 15 along with their equivalent binary and hexadecimal values

Table 1-1 Binary, decimal, and hexadecimal representation

Binary	Decimal	Hexadecimal	Binary	Decimal	Hexadecimal
0000	0	0	1000	8	8
0001	1	1	1001	9	9
0010	2	2	1010	10	A
0011	3	3	1011	11	B
0100	4	4	1100	12	C
0101	5	5	1101	13	D
0110	6	6	1110	14	E
0111	7	7	1111	15	F

1.1.3 Common Data Units

1. Byte-Sized Data

A group of nine **related** bits is called a **byte**. As shown in Figure 1-2, each byte consists of 8 bits for data and 1 bit for parity.

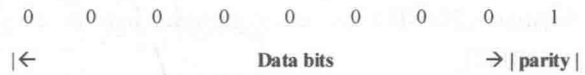


Figure 1-2 A byte with one bit parity

Byte-sized data is stored as unsigned or signed integers. **Figure 1-3** illustrates both the unsigned and signed forms of the byte-sized integer. The difference between those forms is the meaning of the leftmost bit position. In signed integer format, the leftmost bit represents the sign bit of the number. **That is to say**, an 80H represents a value of 128 as an unsigned number. While as a signed number, it represents a value of minus 128. The values of unsigned integers range from 00H to FFH (0-255). The values of signed integers range from −128 to 0 to +127.

Although negative signed numbers are represented in this way, they are stored in the two's complement form. The method of evaluating a signed number by using the weights of each bit position is much easier than the act of two's complementing a number to find its value. This is especially helpful to the programmers to design calculators.

Whenever a number is two's complemented, its sign changes from negative to positive or