21

高等学校计算机专业“十三五”规划教材

微型计算机原理及应用

（第三版）

李伯成　侯伯亨　张毅坤　编著◎

WEIXING JISUANJI YUANLI JI YINGYONG

西安电子科技大学出版社
http://www.xduph.com

高等学校计算机专业“十三五”规划教材

微型计算机原理及应用

(第 三 版)

李伯成　侯伯亨　张毅坤　编著

西安电子科技大学出版社

内 容 简 介

本书以 8086(8088)为对象，描述微型计算机的组成和接口技术。全书共分 8 章，主要内容包括微型计算机的基本结构、指令系统及汇编语言程序设计、存储系统、输入/输出技术、总线及典型接口芯片的应用等。此外，书中还对 Pentium 处理器及 SOC 作了简要介绍。

本书既强调基本概念，又强调工程上分析问题和解决问题的方法。书中内容简明扼要、重点突出，融入了作者的教学经验和体会。本书既可作为高等院校各相关专业的教材，也可作为相关技术人员的参考书。

★本书配有电子教案，有需要者可登录出版社网站，免费下载。

图书在版编目(CIP)数据

微型计算机原理及应用 / 李伯成，侯伯亨，张毅坤编著. —3 版.
—西安：西安电子科技大学出版社，2017.6
ISBN 978-7-5606-4549-0

Ⅰ. ①微… Ⅱ. ①李… ②侯… ③张… Ⅲ. ①微型计算机 Ⅳ. ①TP36

中国版本图书馆 CIP 数据核字(2017)第 121211 号

责任编辑　阎　彬　臧延新
出版发行　西安电子科技大学出版社(西安市太白南路 2 号)
电　　话　(029)88242885　88201467　邮　　编　710071
网　　址　www.xduph.com　电子邮箱　xdupfxb001@163.com
经　　销　新华书店
印刷单位　陕西天意印务有限责任公司
版　　次　2017 年 6 月第 3 版　2017 年 6 月第 19 次印刷
开　　本　787 毫米×1092 毫米　1/16　印 张 22
字　　数　520 千字
印　　数　85 001～88 000 册
定　　价　42.00 元
ISBN 978 - 7 - 5606 - 4549 - 0/TP
XDUP 4841003-19

前　言

本书第二版出版已有几年，而计算机技术的发展日新月异，作为高等学校的教材也必须适应技术的发展。而且本书第二版经过几年的使用，也发现一些内容需要修订。为此，我们对第二版的内容进行修订，删去一些次要的、过时的内容，改正一些小的错误，增加一些新的知识，形成本书的第三版。

尽管计算机技术发展得非常快，有许多新的技术、新的器件不断涌现出来，但从另一角度可以看到，一些最基本的概念、最基本的解决问题的方法是不会改变的。认真学习并很好地掌握这些基本概念和基本方法，就能够在具体的工程实践中解决问题。因此，本书力图阐明微型计算机原理及应用中的基本概念和基本方法，并适当地介绍新的知识。

在这一版里，继续保持前两版中的核心知识，内容仍以适合于课堂教学的8086（8088）微处理器为对象，全面地介绍微型计算机的组成及工程应用。只要学生掌握了本书的基本概念和基本方法，今后再去学习和理解其他相关知识就不会有什么困难。

本书作为理工科专业的教材，以适应学生未来工作的需要为目的。在具体的教学实施过程中，前面六章是基本要求，需要仔细讲述；最后两章的内容，教师可在概要说明后让学生自行阅读，慢慢加以体会。

需再次强调的是，本书的目的在于培养学生的工程思维能力，其内容在描述清楚基本概念的基础上，侧重于解决具体工程应用问题的方法。要求读者能利用所学的基本概念，提出解决工程问题的思路和方法，提高分析和解决问题的能力。

在本书的编写过程中，除了书后的参考资料外，编者还参考了一些网上的资料，在此向这些资料的作者表示感谢！

还要感谢西安电子科技大学出版社的关心和支持，感谢家人的支持与帮助。

本次修订工作由李伯成完成。由于水平有限，书中不当之处在所难免，敬请读者批评指正。

编　者

2017 年 2 月

第二版前言

微型计算机已广泛应用于各行各业，促进了社会的发展和进步。作为今天的工程技术人员，必须很好地掌握微型计算机的概念与技术。本书是为高校理工科专业教学及一般工程技术人员学习微型计算机而编写的。

本书于1998年出版第一版，后经十几次印刷使用至今。现在，第一版中的部分内容有些陈旧，同时由于技术发展很快，更需要补充一些新的内容。为此，本书在第一版的基础上进行修订，删去了一些陈旧的、不适合于课堂教学的内容，并增加了一些新的内容，以进一步提高内容的完整性和实用性。

本书编写的目的在于培养学生的工程思维能力，其内容在描述清楚基本概念的基础上，侧重于解决具体的工程应用问题，要求读者能利用所学的基本概念，提出解决工程问题的思路和方法，从而培养分析具体的工程问题和解决问题的能力。

全书共分8章。第1章为本书所用到的预备知识；第2章介绍8086(8088)微处理器；第3章讲述8086指令系统汇编语言及程序设计的基本方法；第4章讨论内部存储器；第5章论述微型机中常用的I/O技术；第6章描述一些最常用的典型接口芯片及应用；第7章介绍微机中常用的总线；第8章给出一典型的SOC（片上系统）芯片。

在教学工作中，前6章是基本要求，应仔细向学生描述清楚。通过这几章的学习，读者可掌握有关微型机的最基本的概念、基本原理和基本方法。后面两章有新增加的内容，为的是拓展学生的知识面。对这两章可作简要介绍，亦可留给学生自己阅读学习。

本次的修订工作由李伯成完成。在编写本书的过程中，力求以简明扼要的语言，重点突出基本原理、基本概念和基本方法，并且在内容中融入作者以往的教学和科研工作的经验。尽管作者做了努力，但由于水平及时间上的限制，书中疏漏或不当之处在所难免，敬请读者批评指正。

编　者

2007年8月

第一版前言

随着技术的发展和进步，微型计算机的应用在各行各业中迅猛发展，它已成为每个专业技术人员必备的基础。本书是为高校各专业师生及一般科技人员学习微型计算机的需要而编写的。

如何学习微型计算机，使自己很快入门是经常困扰初学者的问题。而微机的发展，不管是硬件还是软件，真可谓日新月异，使人眼花缭乱。从通用的CPU到单片机、数字信号处理器(DSP)、位片机、专用处理器等等，它们均由许多厂家生产，本身又有许多系列。我们认为可以从特殊到一般进行学习，即选择国内比较流行的某种型号的微型机(或单片机)，认真仔细地学好，建立正确的概念。只要进了门，再遇到其他类型的微型机你将很容易掌握它们。因为，尽管型号不同，但它们共性的东西是大量的。为此，我们以8086(8088)为对象，为读者做深入分析和描述。

本书偏重于工程应用，因此，对于各种芯片（包括CPU），我们强调读者抓住其外部特性，能将它们用好就达到了目的。至于芯片内部的东西，以工程上够用即可，读者也没有必要搞清楚那些大（或超大）规模集成电路芯片的内部细节。

微型计算机能够不知疲倦地一条指令接一条指令地执行程序，从而实现你所要求的各种功能。在学习本书时，请读者注意这门课的一些特点。

全书共分8章，从最基本的概念入手，引导读者逐步掌握微型机从硬件组成到软件编程的基本知识。通过本书的学习，使读者能初步掌握微型计算机组成原理和简单的应用。编写过程中力求重点突出，通俗易懂。在内容上做到简明扼要，深入浅出，便于各类人员阅读和学习。

本书的第1、3章由张毅坤编写，第2、5、6章由侯伯亨编写，第4、7、8章由李伯成编写。全书由李伯成主编并统稿。此书在编写过程中得到陕西省教委有关同志的支持和帮助，在此表示衷心的感谢。由于水平所限，加之时间匆促，错误和不当之处在所难免，敬请读者批评指正。

编　者

1997年8月于西安

目　录

第1章 预备知识

本章介绍书中所要用到的一些预备知识。如果读者已经熟悉了这些知识，则可以跳过这一章。

1.1 数与数制

1. 十进制计数法

在十进制计数中，用 0，1，2，…，9 这 10 个符号来表示数量，无论多大的数，都用这 10 个符号的组合来表示。正是由于表示数量的符号有 10 个，因此称之为十进制计数法。

例如，十进制数 3758 可用下面的法则来表示：

$$(3758)_{10}=3\times10^3+7\times10^2+5\times10^1+8\times10^0$$

根据同样的法则，也可以表示十进制小数，小数点的右边各位的权依次为 10^{-1}，10^{-2}，10^{-3}，…。例如，十进制数 275.368 可以用上述法则写成：

$$(275.368)_{10}=2\times10^2+7\times10^1+5\times10^0+3\times10^{-1}+6\times10^{-2}+8\times10^{-3}$$

2. 二进制计数法

二进制计数法用来表示数量的符号只有两个，就是 0 和 1。二进制数中的任何一个 0 或 1 称为比特(bit)。

例如，二进制数 110101 可以表示为

$$(110101)_2=1\times2^5+1\times2^4+0\times2^3+1\times2^2+0\times2^1+1\times2^0$$

3. 二进制数与十进制数的相互转换

1) 二进制数转换成十进制数

如上所述，只要将二进制数的每一位乘上它的权，然后加起来就可以求得二进制数的十进制数值。例如，二进制数 101101.11 换算成十进制数为

$$\begin{aligned}(101101.11)_2&=1\times2^5+0\times2^4+1\times2^3+1\times2^2+0\times2^1+1\times2^0+1\times2^{-1}+1\times2^{-2}\\&=(45.75)_{10}\end{aligned}$$

2) 十进制数转换成二进制数

十进制数转换为二进制数的方法分两步进行，分别处理整数部分和小数部分。一个十进制整数的二进制转换方法是“除 2 取余”，而一个十进制小数的二进制转换方法是“乘 2 取整”。若一个十进制数既包含整数部分又包含小数部分，它的二进制转换就是将它的整数部分和小数部分用上述方法分别进行转换，最后将转换好的两部分结合在一起形成

要转换的二进制数。

4. 八进制计数法

在八进制记数中，用 0，1，2，…，7 这 8 个符号来表示数量，无论多大的数，都用这 8 个符号的组合来表示。现在，八进制计数用得很少，本书基本上不用。

5. 十六进制计数法

在十六进制计数中，用 0，1，2，…，9 和 A，B，C，…，F 等 16 个符号来表示数量，即表示数量的符号有 16 个。

例如，十六进制数 E5D7.A3 可以表示为

$$(E5D7.A3)_{16} = E \times 16^3 + 5 \times 16^2 + D \times 16^1 + 7 \times 16^0 + A \times 16^{-1} + 3 \times 16^{-2}$$

同前所述，一个十进制数可以转换成十六进制数，其方法为十进制数的整数部分“除 16 取余”，十进制数的小数部分则采用“乘 16 取整”。由于一位十六进制数可以用四位二进制数来表示，因此二进制数与十六进制数的相互转换就比较容易。二进制数到十六进制数的转换由小数点开始，每四位二进制数为一组，将每一组用相应的一位十六进制数来表示，即可得到正确的十六进制数。

1.2 算术逻辑运算

1. 二进制加法

二进制加法与十进制加法类似，所不同的是，二进制加法是“逢二进一”，其法则为

$0 + 0 = 0$

$1 + 0 = 1$

$0 + 1 = 1$

$1 + 1 = 0$　　有进位

2. 二进制减法

在二进制减法中，同样有如下法则：

$0 - 0 = 0$

$1 - 0 = 1$

$1 - 1 = 0$

$0 - 1 = 1$　　有借位

当不够减时需要借位，高位的 1 等于下一位的 2，即“借一当二”。

3. 二进制乘法

二进制乘法与十进制乘法是一样的。但因为二进制数只由 0 和 1 构成，因此，二进制乘法更简单。其法则如下：

$0 \times 0 = 0$

$1 \times 0 = 0$

$0 \times 1 = 0$

$1 \times 1 = 1$

4. 二进制除法

二进制除法是乘法的逆运算，其方法与十进制除法是一样的，而且二进制数仅由0和1构成，做起来更简单。

5. 二进制与

二进制与又称为逻辑乘，其法则为

$$0\wedge 0=0,\ 0\wedge 1=0,\ 1\wedge 0=0,\ 1\wedge 1=1$$

6. 二进制或

二进制或又称为逻辑加，其法则为

$$0\vee 0=0,\ 0\vee 1=1,\ 1\vee 0=1,\ 1\vee 1=1$$

7. 二进制异或

二进制异或的法则为

$$0\forall 0=0,\ 0\forall 1=1,\ 1\forall 0=1,\ 1\forall 1=0$$

1.3 符号数的表示方法

表示一个带符号的二进制数通常有4种方法。

1. 原码法

原码法的规则就是符号与数值连续排列，符号放在最高位，且用0表示正数，用1表示负数，其后跟着数值。

例如，十进制数$(+45)_{10}$和$(-45)_{10}$，用8位二进制原码表示，它们的原码符号数如下：

$$(+45)_{10}=(0\quad 0101101)_2$$

↑ ↑

符号位 数值

$$(-45)_{10}=(1\quad 0101101)_2$$

↑ ↑

符号位 数值

2. 反码法

早期的计算机曾采用反码法来表示带符号的数。对于正数，其反码与其原码相同。例如：

$$(+45)_{10}=(00101101)_2$$

也就是说，正数用符号位与数值一起来表示。

对于负数，用相应正数的原码各位取反来表示，包括将符号位取反，取反的含义就是将0变为1，将1变为0。例如，$(-45)_{10}$的反码表示就是将上面$(+45)_{10}$的二进制数各位取反：

$$(-45)_{10}=(11010010)_2$$

3. 补码法

在计算机中，符号数最常用的是补码(对 2 的补码)形式。用补码法表示带符号数的法则是：正数的表示方法与原码法和反码法一样；负数的表示方法为该负数的反码加 1。

例如，$(+4)_{10}$ 的补码表示为$(00000100)_2$，而$(-4)_{10}$ 用补码表示时，可先求其反码表示$(11111011)_2$，而后再在其最低位加 1，变为$(11111100)_2$。这就是$(-4)_{10}$ 的补码表示，即$(-4)_{10}=(11111100)_2$。

4. 移码法

在计算机的浮点数表示中会用到移码。移码可以理解为在补码的基础上偏移多少数值。偏移的数值可以人为定义。例如，对 n 位整数来说，经常使用的偏移量为 2^{n-1}。若令 n 为 8，则偏移量为 2^7，即 128。也就是说在补码的基础上加上 128 便成为移码。

例如，+7 的补码为 00000111，则 +7 的移码为 10000111。同样，−7 的补码为 11111001，那么，−7 的移码便是 01111001。

可见，在偏移 2^{n-1} 的情况下，只要将补码的符号位取反便可获得相应的移码。

1.4 补码的运算

补码加减法的运算法则为

$$[X+Y]_{补}=[X]_{补}+[Y]_{补}$$

$$[-X]_{补}=-[X]_{补}$$

$$[X-Y]_{补}=[X]_{补}+[-Y]_{补}=[X]_{补}-[Y]_{补}$$

由这些法则可见，和的补码可用补码求和实现；而差的补码可通过将减数求补再与被减数相加实现。也就是说在补码情况下，利用加法器可完成减法运算。

在计算机中，一般都不设置专门的减法电路。遇到两个数相减时，处理器就自动地将减数取补，而后将被减数和减数的补码相加来完成减法运算。

例如，$(69)_{10}-(26)_{10}$ 可以写成$(69)_{10}+(-26)_{10}$。将$(69)_{10}$ 的原码和$(-26)_{10}$ 的补码相加，即可得到正确的结果。读者可以自行验证。

这里要强调的是，当两个同符号的数相加（或者是异符号数相减）时，若相加(或相减)的结果超出了所规定的数值范围，则会发生溢出。一旦发生溢出，其结果肯定是错误的。

例如，两个带符号数$(01000001)_2$(十进制数 + 65)与$(01000011)_2$(十进制数 + 67)相加：

$$\begin{array}{rr} & 01000001 \\ + & 01000011 \\ \hline & 10000100 \end{array}$$

可以看到，例中两个正数相加的结果为一负数，结果显然是荒谬的。产生错误的原因就是溢出。由于本例中是用 8 位二进制编码表示带符号的数，若用它表示整数，8 位补码所能表示的数值范围为 −128 ～ + 127。若结果超出这一范围就产生错误。在将来的编程中，可用增加表示数值编码的位数的方法来消除溢出的发生。上面例子中若用多于 8 位的二进制编码表示那两个带符号的数，再相加时肯定不会产生溢出。

再来看两个负数$(10001000)_2$和$(11101110)_2$的相加情况。

$$\begin{array}{r} 10001000 \\ +\quad 11101110 \\ \hline 01110110 \end{array}$$

两负数相加产生溢出的情况读者可以自行分析。

1.5 数的定点表示和浮点表示

1.5.1 数的定点表示法

所谓定点数就是小数点固定不变的数。小数点的位置通常有两种约定，即定点整数（相当于小数点在最低有效位之后）和定点小数（相当于小数点在最高有效位之前）。

如前所述，要表示带符号数，符号总是放在最高位。若表示带符号数的字长为 n 位，则定点整数原码、补码的表示范围分别为

定点整数原码的表示范围：$-(2^{n-1}-1) \sim +(2^{n-1}-1)$

定点整数补码的表示范围：$-(2^{n-1}) \sim +(2^{n-1}-1)$

若用 n 位字长表示小数，则定点小数原码、补码的表示范围分别为

定点小数原码的表示范围：$-(1-2^{-(n-1)}) \sim +(1-2^{-(n-1)})$

定点小数补码的表示范围：$-1 \sim +(1-2^{-(n-1)})$

1.5.2 数的浮点表示法

定点表示法比较简单，要么纯整数，要么纯小数，所能表示的数值范围也比较小，运算中很容易因超出范围而溢出。为克服这些缺点，引入了数的浮点表示法。

在十进制数中，一个数可以写成多种表示形式。例如，83.125 可写成 $10^2 \times 0.83125$，$10^3 \times 0.083125$，$10^4 \times 0.0083125$ 等。同样，一个二进制数也可以写成多种表示形式。例如，二进制数 1011.10101 可以写成 $2^4 \times 0.101110101$，$2^5 \times 0.0101110101$，$2^6 \times 0.00101110101$，等等。可以看出，一个二进制数能够用一种普遍的形式来表示：

$$2^E \times F$$

其中，E 称为阶码，F 叫做尾数。人们把用阶码和尾数表示的数叫做浮点数，这种表示数的方法称为浮点表示法。

在浮点表示法中，阶码通常为带符号的整数，尾数为带符号的纯小数。浮点数的表示格式如下：

数 符	阶 码	尾 数

很明显，浮点数的表示不是唯一的。当小数点的位置改变时，阶码也随着相应改变，可以用多种形式来表示同一个数。

浮点数所能表示的数值范围主要由阶码决定，所表示数值的精度则主要由尾数来决定。为了充分利用尾数来表示更多的有效数字，通常采用规格化浮点数。规格化就是将尾数限定在小于 1 且大于等于 0.5 之间。

当尾数用补码表示时，若尾数 $M \geq 0$，则尾数规格化应为 $M = 0.1\times\times\times\cdots\times$。其中×可为 0，也可为 1。

若尾数 $M < 0$，规格化应满足 $[-1/2]_{补} > [M]_{补} \geq [-1]_{补}$，则尾数规格化应为 $M = 1.0\times\times\times\cdots\times$。其中×可为 0，也可为 1。

1.5.3 工业标准 IEEE754

IEEE754 是由 IEEE 制定的有关浮点数的工业标准，被广泛采用。该标准的表示形式如下：

$$(-1)^S 2^E (b_0 \diamond b_1 b_2 b_3 \cdots b_{p-1})$$

其中：$(-1)^S$ 为该浮点数的数符，当 s 为 0 时表示正数，s 为 1 时表示负数；E 为指数，用移码表示；$(b_0 \diamond b_1 b_2 b_3 \cdots b_{P-1})$为尾数，共 P 位，用原码表示。

目前计算机中使用的三种形式的 IEEE754 浮点数格式列于表 1.1 中。

在 IEEE754 标准中，特别要说明的就是尾数在规格化时的处理，也就是说在规格化的过程中必须使 b_0 为 1，而且小数点应当在◇位置上，是隐含的。规格化时将 b_0 去掉，也是隐含的。这相当于使尾数增加了一位，在使用时应注意到这种情况。

表 1.1 三种形式的 IEEE754 浮点数格式

参 数	单精度浮点数	双精度浮点数	扩充精度浮点数
浮点类字长	32	64	80
尾数长度 P	23	52	64
符号位 S	1	1	1
指数长度 E	8	11	15
最大指数	+127	+1023	+16383
最小指数	−126	−1022	−16382
指数偏移量	+127	+1023	+16383
可表示的实数范围	$10^{-38}\sim10^{38}$	$10^{-308}\sim10^{308}$	$10^{-4932}\sim10^{4932}$

为了说明 IEEE754 浮点数的应用，现举例如下：

利用 IEEE754 标准将数 176.0625 表示为单精度浮点数。首先将该十进制数转换成二进制数：

$$(176.0625)_{10} = (10110000.0001)_2$$

对上面的二进制数进行规格化：

$$10110000.0001 = 1.\diamond01100000001 \times 2^7$$

这就保证了使 b_0 为 1，而且小数点在◇位置上。将 b_0 去掉并扩展为单精度浮点数所规定的 23 位尾数：01100000001000000000000。

然后，再来求取阶码。现指数为 7，而单精度浮点数规定指数的偏移量为 127（请注意不是前面移码描述中所提到的 128），即在指数 7 上加 127。那么，E = 7 + 127 = 134，则指数的移码表示为 10000110。

最后，可得到$(176.0625)_{10}$的单精度浮点数表示：

0　10000110　01100000001000000000000

1.6 BCD 码

将十进制数转换为其等值的二进制数称为编码。前面所提到的二进制数称为纯二进制码。计算机只能识别用高低电平表示的 0 或 1，对计算机来说，用纯二进制码是十分方便的。但人们则更习惯使用十进制数。为此，人们发明了用二进制编码来表示的十进制数，它有多种表示形式，在此只介绍以下两种表示方法。

1. 8421 码

BCD 码中的 8421 码用 4 位二进制数表示一位十进制数，它们的对应关系如表 1.2 所示。

表 1.2　8421 码与十进制数的对应关系

十进制数	8421 码
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

由表 1.2 可以看到，这种形式的 BCD 码二进制数各位的权值分别是 8421。同时，在这种 BCD 码表示法中，剩下的 6 种四位编码，从 1010 到 1111 全都是非法的。

根据上述说明可以看到，一个十进制数能够很方便地用 BCD 码来表示。例如，十进制数 834 用 BCD 码表示为

$$(834)_{10}=(1000\quad 0011\quad 0100)_{BCD}$$

只要熟记十进制数 0～9 与 BCD 码的对应关系，就能方便地将它们进行转换。例如：

$$(0110\ 1001\ 0101\ .\ 0010\ 0111\ 1001)_{BCD} = (695.279)_{10}$$

2. 余 3 码

余 3 码也是用 4 位二进制编码来表示一个十进制数的。但这是一种无权码，其表示形式如表 1.3 所示。

表 1.3 余 3 码与十进制数的对应关系

十进制数	余 码
0	0011
1	0100
2	0101
3	0110
4	0111
5	1000
8	1001
7	1010
8	1011
9	1100

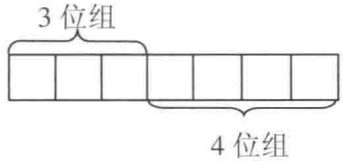
工程上还有多种 BCD 码的编码形式，此处不再介绍。

1.7 ASCII 码

ASCII 码是美国标准信息交换码的简称，现在为各国所广泛采用。

通常，ASCII 码由 7 位二进制编码来表示，用于微处理机与它的外部设备之间进行数据交换以及通过无线或有线手段进行数据传送。

代表上述字符或控制功能的 ASCII 码是由一个 4 位组和一个 3 位组构成的，形成 7 位二进制编码，其格式为



根据 ASCII 码的构成格式，可以很方便地从有关的 ASCII 表中查出每一个字符或特殊控制功能的编码。例如，大写英文字母 A，从表中查出其 3 位组为$(100)_2$，4 位组为$(0001)_2$，故构成字母 A 的 ASCII 编码为$(1000001)_2$或$(41)_{16}$。