高·等·学·校·计·算·机·教·材

微型计算机原理及应用

（第3版）

朱定华　主　编
刘福珍　蔡　勤　副主编

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

高等学校计算机教材

微型计算机原理及应用

（第3版）

電子工業出版社
Publishing House of Electronics Industry
北京 · BEIJING

内 容 简 介

《微型计算机原理及应用》系统地介绍了80x86 PC机的原理、汇编语言程序设计及接口技术，主要内容包括计算机基础知识；汇编语言与汇编程序；程序设计技术；总线；半导体存储器；输入与输出技术；中断技术；常用可编程接口芯片等。本书内容精练、实例丰富，其中大量的接口电路和程序是作者多年来在科研和教学中反复提炼得来的，因而本书应用性很强，可作为大专院校和高职高专成人高等教育“汇编语言程序设计”、“微机原理及应用”、“接口技术”等课程的教学用书。也可以供从事电子技术、计算机应用与开发的科研人员和工程技术人员学习参考，还适于初学者自学使用。

图书在版编目(CIP)数据

微型计算机原理及应用 / 朱定华主编. —3版. —北京：电子工业出版社，2010.4
高等学校计算机教材
ISBN 978-7-121-08935-0

Ⅰ.微… Ⅱ.朱… Ⅲ.微型计算机—高等学校—教材 Ⅳ.TP36

中国版本图书馆CIP数据核字（2009）第089126号

策　　划：陈晓明
责任编辑：陈晓明　　特约编辑：张晓雪
印　　刷：北京市海淀区四季青印刷厂
装　　订：涿州市桃园装订有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路173信箱　邮编 100036
开　　本：787×1 092　1/16　印张：21　字数：538千字
印　　次：2010年4月第1次印刷
印　　数：4 000册　　定价：33.00元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至zlts@phei.com.cn，盗版侵权举报请发邮件至dbqq@phei.com.cn。

服务热线：（010）88258888。

第 3 版前言

本书第 2 版自 2005 年出版以来，受到诸多兄弟院校师生及广大读者的关注，我们深表感谢。

通过多年来的教学实践，尤其是近 3 年来课程改革的经验，我们对教材内容和课程体系进行了深入的研究，并作了修改和更新。随着电子技术和微型计算机技术的迅猛发展，从 8086 开始，80286、80386、80486、Pentium 等系列微处理器不断推出，本书在第 2 版的基础上对原章节加宽加深，既保持了多年形成的比较成熟的课程体系，又适当地介绍了微型计算机中的新器件、新技术和新方法。

本书通过 80x86 到 Pentium 微处理器和 PC 机的硬件和软件分析，阐明微型计算机的组成原理、汇编语言程序设计以及存储器、输入/输出接口芯片与微型计算机的接口方法，为学习者在微处理器和微型计算机的应用上打下坚实的基础。

本书包括汇编语言程序设计和接口技术两部分内容。汇编语言程序设计是微机应用系统的系统软件和应用软件的设计基础，接口技术是微机应用系统硬件组成的设计基础。本书内容较全面，实例丰富。书中的程序和接口电路的设计包含了作者多年来在科研和教学中积累的经验和技巧。学习微型计算机的汇编语言程序设计和接口技术必须理论联系实际。本书在介绍基本概念的同时，列举了大量典型而有意义的例题和习题。这些例题和习题，无论是汇编程序还是接口电路都在 80x86 和 Pentium 系列微机系统上调试通过。

80x86 为用户提供了实地址方式、虚地址保护方式和虚拟 80x86 方式 3 种工作方式，但从编程角度看，仅提供了实地址方式和虚地址保护方式 2 种工作方式。就编程而言，这 2 种工作方式并无实质上的区别，而且使用实地址方式已可解决应用程序所面向的大量问题，所以本书有关汇编语言程序设计的讨论只限于 DOS 环境下（MASM 5.0）的实地址方式。

本书内容精练，实用性强。每章后均附有思考题与习题。编写本书时，注意了理论和实践相结合，力求做到既有一定的理论基础，又能运用理论解决实际问题；既掌握一定的先进技术，又着眼于当前的应用服务。

本教材的参考学时数为 80 学时（不含实验）。学时数较少的学校或专业可以不讲授第 2 章和第 3 章中的以下内容：地址传送指令、查表转换指令、BIOS、串处理程序设计和宏功能程序设计等，本书后面没有使用这些内容。为了适应非电子信息类的教学要求，本书的第 1 章中还补充了二进制数的逻辑运算与逻辑电路以及逻辑单元与逻辑部件等内容。

本书由朱定华、刘福珍和蔡勤编写。参加本书编写工作的人员还有蔡苗、黄松、周斌、翟晟、蔡红娟、吕建才、程萍、张德芳、林卫、李志文、林威等。

计算机的发展日新月异，教学改革任重道远。限于我们的水平和能力，不妥之处在所难免，恳请读者批评指正，以便我们今后不断改进。

朱定华

2010 年 1 月于武昌

目　录

第 1 章　微型计算机的基础知识

1.1　计算机中的数和编码

1.1.1　计算机中的数制

计算机最早是作为一种计算工具出现的，所以它的最基本的功能是对数进行加工和处理。数在机器中是以器件的物理状态来表示的。一个具有两种不同的稳定状态且能相互转换的器件就可以用来表示 1 位（bit）二进制数。二进制数有运算简单，便于物理实现，节省设备等优点，所以目前在计算机中数几乎全采用二进制表示。但是二进制数书写起来太长，且不便于阅读和记忆；而 4 位二进制数有 16 个不同的状态 0000～1111，即 1 位十六进制数；所以微型计算机中的二进制数都采用十六进制数来缩写。十六进制数用 0～9 和 A～F 等 16 个数码表示 4 位二进制数 0000～1111，这 16 个二进制数 0000～1111 的大小就是十进制数 0～15。1 个 8 位的二进制数用 2 位十六进制数表示，1 个 16 位的二进制数用 4 位十六进制数表示等。这样书写方便，又便于阅读和记忆，且转换方便，因此常用十六进制数来缩写二进制数。然而人们最熟悉、最常用的是十进制数。为此，要熟练地掌握十进制数、二进制数和十六进制数间的相互转换。它们之间的关系如表 1-1 所列。

表 1-1　十进制数、二进制数及十六进制数对照表

十进制	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
二进制	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
十六进制	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

为了区别十进制数、二进制数及十六进制数 3 种数制，可在数的右下角注明数制，或者在数的后面加一字母。如 B（binary）表示二进制数制；D（decimal）或不带字母表示十进制数制；H（hexadecimal）表示十六进制数制。

1．二进制数和十六进制数整数间的相互转换

根据表 1-1 所示的对应关系即可实现它们之间的转换。

二进制整数转换为十六进制数，其方法是从右（最低位）向左将二进制数分组：每 4 位为 1 组，最后一组若不足 4 位则在其左边添加 0，以凑成 4 位 1 组，每组用 1 位十六进制数表示。如：

1111111000111B →1 1111 1100 0111B → 0001 1111 1100 0111B=1FC7H

十六进制数转换为二进制数，只需用 4 位二进制数代替 1 位十六进制数即可。如：

3AB9H=0011 1010 1011 1001B

2．十六进制数和十进制数间的相互转换

十六进制数转换为十进制数十分简单，只需将十六进制数按权展开相加即可。如：

$1F3DH=16^3\times 1+16^2\times 15+16^1\times 3+16^0\times 13=4096\times 1+256\times 15+16\times 3+1\times 13=4096+3840+48+13=7997$

十进制整数转换为十六进制数可用除 16 取余法，即用 16 不断地去除待转换的十进制数，直至商等于 0 为止。将所得的各次余数，依倒序排列，即可得到所转换的十六进制数。如将 38947 转换为十六进制数，其方法及算式如下：

```
16 | 38947   3
16 |  2434   2
16 |   152   8
16 |     9   9
     0
```

即 38947=9823H

1.1.2 符号数的表示法

1．机器数与真值

二进制数与十进制数一样有正负之分。在计算机中，常用数的符号和数值部分一起编码的方法表示符号数。常用的有原码、反码和补码表示法。这几种表示法都将数的符号数码化。通常正号用“0”表示，负号用“1”表示。为了区分一般书写时表示的数和机器中编码表示的数，我们称前者为真值，后者为机器数，即数值连同符号数码“0”或“1”一起作为一个数就称为机器数，而它的数值连同符号“+”或“－”称为机器数的真值。把机器数的符号位也作为数值的数，就是无符号数。

为了表示方便，常把 8 位二进制数称为字节，把 16 位二进制数称为字，把 32 位二进制数称为双字。对于机器数应将其用字节、字或双字表示，所以只有 8 位、16 位或 32 位机器数的最高位才是符号位。

2．原码

按上所述，数值用其绝对值，正数的符号位用 0 表示，负数的符号位用 1 表示，这样表示的数就称为原码。如：

X_1= 105=+01101001B　　$[X_1]_{原}$=01101001B

X_2=−105= −01101001B　　$[X_2]_{原}$=11101001B

其中最高位为符号，后面 7 位是数值。用原码表示时，+105 和－105 的数值部分相同而符号位相反。

原码表示简单易懂，而且与真值的转换方便。但若是两个异号数相加，或两个同号数相减，就要做减法。为了把减运算转换为加运算，从而简化计算机的结构，就引进了反码和补码。

3．反码

正数的反码与原码一样，符号位为 0，其余位为其数值；负数的反码为它的绝对值（即与其绝对值相等的正数）按位 （连同符号位） 取反。如：

X_1= 105=+01101001B　　$[X_1]_{反}$=01101001B

X_2= −105= 01101001B　　$[X_2]_{反}$=10010110B

4．补码

正数的补码与原码一样，符号位为 0，其余位为其数值；负数的补码为它的绝对值（即与该负数的绝对值相等的正数）的补数。把一个数连同符号位按位取反再加 1，可以得到该数的补数。如：

X_1= 105=+01101001B　　$[X_1]_{补}$=01101001B

X_2=−105=−01101001B　　$[X_2]_{补}$=10010111B

求补数还可以直接求，方法是从最低位向最高位扫描，保留直至第一个“1”的所有位，以后各位按位取反。负数的补码可以由与其绝对值相等的正数求补得到。根据两数互为补数的原理，对补码表示的负数求补就可以得到该负数的绝对值。如：

$[-105]_{补}$=10010111B=97H

对其求补，从右向左扫描，第一位就是 1，故只保留该位，对其左面的七位均求反得：01101001，即补码表示的机器数 97H 的真值是−69H（=−105）。

一个用补码表示的机器数，若最高位为 0，则其余几位即为此数的绝对值；若最高位为 1，则其余几位不是此数的绝对值，把该数（连同符号位）求补，才得到它的绝对值。

当数采用补码表示时，就可以把减法转换为加法。例如：

64−10=64+(−10)

$[64]_{补}$=40H=0100 0000B

$[10]_{补}$=0AH=0000 1010B

$[-10]_{补}$=1111 0110B

做减法运算过程如下：

```
  0100 0000
- 0000 1010
  0011 0110
```

用补码相加过程如下：

```
  0100 0000
+ 0000 1010
1 0011 0110
↑
└进位自然丢失
```

结果相同，其真值为：54（36H=48+6）。

最高位的进位是自然丢失的，故做减法与用补码相加的结果是相同的。因此，在微型计算机中，凡是符号数一律是用补码表示的。一定要记住运算的结果也是用补码表示的。如：

34−68=34+(−68)

34=22H=0010 0010B

68=44H=0100 0100B

−68=1011 1100B

做减运算过程如下：

```
  0010 0010
- 0100 0100
-----------
1 0100 0100
↑
└借位自然丢失
```

用补码相加过程如下：

```
  0010 0010
+ 1011 1100
-----------
  1101 1110
```

结果相同。因为符号位为 1，所以结果为负数。对其求补，得其真值：-00100010B，即为−34（−22H）。

由上面两个例子还可以看出，当数采用补码表示后，两个正数相减，若无借位，化为补码相加就会有进位；若有借位，化为补码相加就不会有进位。

5．8 位二进制数的范围

8 位二进制数，将其看成无符号数和符号数，它所表示的数的大小是不同的。为加深读者的印象，将其列于表 1-2 中。

表 1-2　8 位二进制数（2 位十六进制数）的大小

8 位二进制数	2 位十六进制数	无符号数	原码数	反码数	补码数
0000 0000	00	0	0	0	0
0000 0001	01	1	1	1	1
0000 0010	02	2	2	2	2
⋮	⋮	⋮	⋮	⋮	⋮
0111 1101	7D	125	125	125	125
0111 1110	7E	126	126	126	126
0111 1111	7F	127	127	127	127
1000 0000	80	128	-0	-127	-128
1000 0001	81	129	-1	-126	-127
1000 0010	82	130	-2	-125	-126
⋮	⋮	⋮	⋮	⋮	⋮
1111 1101	FD	253	-125	-2	-3
1111 1110	FE	254	-126	-1	-2
1111 1111	FF	255	-127	-0	-1

由表 1-2 可知，8 位无符号数的数值范围为 00H～FFH（0～255）。8 位反码数的数值范围为 80H～7FH（−127～127）。8 位原码数的数值范围为 FFH～7FH（−127～127）。8 位补码数的数值范围为 80H～7FH（−128～127）。原码数 80H 和 00H 的数值部分相同、符号位相反，它们分别为−0 和+0；补码数 80H 的最高位既代表了符号为负又代表了数值为 1，80H 的真值是−128(−80H)。

对于一个 16 位的二进制数，若把它看成无符号数，则其数值范围为 0000H～FFFFH（0～65535）；若把它看成反码数，则其数值范围为 8000H～7FFFH（–32767～32767）；若把它看成原码数，则其数值范围为 FFFFH～7FFFH（–32767～32767）；若把它看成补码数，则其数值范围为 8000H～7FFFH（–32768～32767）。

上述分析表明，若 8 位二进制补码数运算结果超出–128～127，16 位二进制补码数运算结果超出–32768～32767，则产生溢出。小于–128 或小于–32768 的运算结果称为下溢出，大于 127 或大于 32767 的运算结果称为上溢出。产生溢出的原因是数据的位数少了，使得结果的数值部分挤占了符号位的位置，为了避免产生溢出，可以将数位扩展。

6．二进制数的扩展

二进制数的扩展是指一个数据从位数较少扩展到位数较多，如从 8 位（字节）扩展到 16 位（字），或从 16 位扩展到 32 位（双字）。一个二进制数扩展后，其数的符号和大小应保持不变。

无符号数的扩展是将其左边添加 0。如 8 位无符号二进制数 F8H 扩展为 16 位无符号二进制数，则为 00F8H。

对于用原码表示的二进制数，它的正数和负数仅 1 位符号位相反，数值位都相同。所以，原码二进制数的扩展是将其符号位向左移至最高位，符号位即最高位与原来的数值位间的所有位都填入 0。例如：68 用 8 位二进制数表示的原码为 44H，用 16 位二进制数表示的原码为 0044H；–68 用 8 位二进制数表示的原码为 C4H，用 16 位二进制数表示的原码为 8044H。

补码表示的二进制数的符号位向左扩展若干位后，所得到的补码数的真值不变。所以，对于用补码表示的二进制数，正数的扩展应该在其前面补 0，而负数的扩展，则应该在前面补 1。例如：68 用 8 位二进制数表示的补码为 44H，用 16 位二进制数表示的补码为 0044H；-68 用 8 位二进制数表示的补码为 BCH，用 16 位二进制数的补码表示为 FFBCH。

补码表示的二进制数的扩展与补码相同。

1.1.3 二进制数的加减运算

计算机把机器数均当成无符号数进行运算，即符号位也参与运算。运算的结果要根据运算结果的符号，运算有无进（借）位和溢出等来判别。计算机中设置有这些标志位，标志位的值由运算结果自动设定。

1．无符号数的运算

无符号数实际上是指参加运算的数均为正数，且整个数位全部用于表示数值。n 位无符号二进制数的范围为 0～（2^n–1）。

（1）两个无符号数相加，由于两个加数均为正数，因此其和也是正数。当和超过其位数所允许的范围时，就向更高位进位。如：

127+160 = 7FH+A0H

```
  0111 1111
+ 1010 0000
-----------
1 0001 1111=11FH=256+16+15=287
↑
└ 进位
```

（2）两个无符号数相减，被减数大于或等于减数，无借位，结果为正；被减数小于减数，有借位，结果为负。如：

192−10=C0H−0AH

```
  1100 0000
- 0000 1010
  1011 0110=B6H=176+6=182
```

反过来相减，即 10−192，运算过程如下：

```
  0000 1010
- 1100 0000
1 0100 1010=-10110110B=-B6H=-182
↑
└ 借位
```

由此可见，对无符号数进行减法运算，其结果的符号用进位来判别：CF=0（无借位），结果为正；CF=1（有借位）结果为负（对 8 位数值位求补得到它的绝对值）。

2．补码数的运算

n 位二进制补码数，除去一位符号位，还有 $n-1$ 位表示数值，所能表示的补码的范围为：$-2^{n-1}\sim(2^{n-1}-1)$。如果运算结果超过此范围就会产生溢出。如：

105+50=69H+32H

```
  0110 1001
+ 0011 0010
  1001 1011=9BH=155或=-65H=-101
```

若把结果视为无符号数，为 155，结果是正确的。若将此结果视为符号数，其符号位为 1，结果为-101，这显然是错误的。其原因是和数 155 大于 8 位符号数所能表示的补码数的最大值 127，使数值部分占据了符号位的位置，产生了溢出，从而导致结果错误。又如：

−105−50=−155

```
  1001 0111
+ 1100 1110
1 0110 0101
↑
└ 进位
```

两个负数相加，和应为负数，而结果 01100101B 却为正数，这显然是错误的。其原因是和数−155 小于 8 位符号数所能表示的补码数的最小值−128，也产生了溢出。若不将第 7 位（第 7 位～第 0 位）0 看成符号，也看成数值而将进位看作数的符号，结果为−0 1001 1011B=−155，结果就是正确的。

因此，应当注意溢出与进位及补码运算中的进位或借位丢失间的区别：

（1）进位或借位是指无符号数运算结果的最高位向更高位进位或借位。通常多位二进制数将其拆成二部分或三部分或更多部分进行运算时，数的低位部分均无符号位，只有最高部分的最高位才为符号位。运算时，低位部分向高位部分进位或借位。由此可知，进位主要用于无符号数的运算，这与溢出主要用于符号数的运算是有区别的。

（2）溢出与补码运算中的进位丢失也应加以区别，如：

−50−5=−55

```
  1100 1110
+ 1111 1011
-------------
1 1100 1001= -00110111B=-55
↑
└进位丢失
```

两个负数相加，结果为负数是正确的。这里虽然出现了补码运算中产生的进位，但由于和数并未超出 8 位二进制补码数-128～127 的范围，因此无溢出。那么如何来判别有无溢出呢？

设符号位向进位位的进位为 C_Y，数值部分向符号位的进位为 C_S，则溢出

$OF=C_Y \oplus C_S$

OF=1，有溢出；OF=0，无溢出。

下面用 M、N 两数相加来证明。设 M_S 和 N_S 为两个加数的符号位，R_S 为结果的符号位，则有如表 1-3 所列的真值表。由真值表得逻辑表达式：

$OF=\overline{C_S}C_Y + C_S\overline{C_Y} = C_S \oplus C_Y$

表 1-3 符号、进位、溢出的真值表

M_S	N_S	R_S	C_S	C_Y	OF
0	0	0	0	0	0
0	0	1	1	0	1
0	1	0	1	1	0
0	1	1	0	0	0
1	0	0	1	1	0
1	0	1	0	0	0
1	1	0	0	1	1
1	1	1	1	1	0

再来看 105+50、−105−50 和−50−5 三个运算有无溢出：

```
  0110 1001          1001 0111          1100 1110
+ 0011 0010        + 1100 1110        + 1111 1011
-------------      -------------      -------------
  1001 1011        1 0110 0101        1 1100 1001
```

$C_Y=0, C_S=1$　　$C_Y=1, C_S=0$　　$C_Y=1, C_S=1$

OF=0⊕1=1，有溢出　　OF=1⊕0=1，有溢出　　OF=1⊕1=0，无溢出

1.1.4 二进制数的逻辑运算与逻辑电路

计算机除了可进行基本的算术运算外，还可对两个或一个无符号二进制数进行逻辑运算。计算机中的逻辑运算，主要是“逻辑非”、“逻辑乘”、“逻辑加”和“逻辑异或”等 4 种基本运算。下面介绍这 4 种基本逻辑运算及实现这些运算的逻辑电路。

1．逻辑非

逻辑非也称“求反”。对二进制数进行逻辑非运算，就是按位求它的反，常用变量上方加一横来表示。例如，

A=01100001B，B=11001011B

$\overline{A}$=10011110B，$\overline{B}$=00110100B

实现逻辑非运算的电路称为非门，又称反相器。它只有一个输入和一个输出。它的符号如图 1-1 所示。

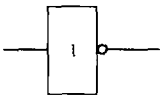

图 1-1 非门的符号表示

2．逻辑乘

对两个二进制数进行逻辑乘，就是按位求它们的“与”，所以逻辑乘又称“逻辑与”，常用记号“∧”或“·”来表示。1 位二进制数逻辑乘的规则为：

0∧0=0，0∧1=0，1∧0=0，1∧1=1

例如，01100001B∧11001011B，逻辑乘算式如下：

$$\begin{array}{r} 0110\ 0001 \\ \wedge\ 1100\ 1011 \\ \hline 0100\ 0001 \end{array}$$

即 01100001B∧11001011B=0100 0001B

实现逻辑乘运算的电路称为与门，2 输入与门的符号表示如图 1-2 所示。

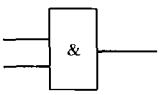

图 1-2　与门的符号表示

3．逻辑加

对两个二进制数进行逻辑加，就是按位求它们的“或”，所以逻辑加又称“逻辑或”，常用记号“∨”或“+”来表示。1 位二进制数逻辑加的规则为：

0∨0=0,0∨1=1,1∨0=1,1∨1=1。

例如，01100001B∨11001011B，逻辑加算式如下：

$$\begin{array}{r} 0110\ 0001 \\ \vee\ 1100\ 1011 \\ \hline 1110\ 1011 \end{array}$$

即 01100001B∨11001011B=11101011B

实现逻辑加运算的电路称为或门，2 输入或门的符号表示如图 1-3 所示。

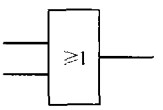

图 1-3　或门的符号表示

4．逻辑异或

对两个二进制数进行逻辑异或，就是按位求它们的模 2 和，所以逻辑异或又称“按位加”，常用符号“⊕”来表示。1 位二进制数的逻辑异或运算规则为：

0⊕0=0,0⊕1=1,1⊕0=1,1⊕1=0。

例如，01100001B⊕11001011B，逻辑异或算式如下：

$$\begin{array}{r} 0110\ 0001 \\ \oplus\ 1100\ 1011 \\ \hline 1010\ 1010 \end{array}$$

即 01100001B⊕11001011B=10101010B

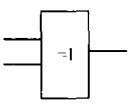

图 1-4　异或门的符号表示

注意：按位加与普通整数加法的区别是它仅按位相加，不产生进位。

实现逻辑异或运算的电路称为异或门，2 输入异或门的符号表示如图 1-4 所示。

异或门的特点是，只有当输入的两个变量相异时，输出为高（“1”），否则输出为低（“0”）。

5．正逻辑与负逻辑

逻辑电路实现的逻辑关系，可用高电平表示逻辑 1，用低电平表示逻辑 0，在这种规定下的逻辑关系称为正逻辑。事实上，还有另一种规定：用低电平表示逻辑 1，用高电平表示