高等职业技术学院教材

微机原理

卢秉娟　张华鹏　陈文清　李　涛　编

Weiji Yuanli

武汉理工大学出版社

微 机 原 理

卢秉娟 张华鹏
陈文清 李 涛 编

武汉理工大学出版社

内 容 简 介

本书用对比的方法详细介绍了 Intel8086/8088 微处理器、MCS-51 系列、MCS-96 系列单片机组成原理，8086/8088 和 MCS-51 系列单片机的指令系统和汇编语言程序设计方法，MCS-51 的定时器/计数器及应用。

全书共分六章，适用于大专院校学生，同时也可供广大科技人员阅读参考。

图书在版编目(CIP)数据

微机原理/卢秉娟等编．—武汉：武汉理工大学出版社，2001.8
ISBN 7-5629-1745-0

Ⅰ．微…
Ⅱ．卢…
Ⅲ．微机-原理-教材
Ⅳ．TP301

武汉理工大学出版社出版发行
（武昌珞狮路 122 号　邮编：430070）
各地新华书店经销
荆州市鸿盛印刷厂印刷

*

开本：787×1092　1/16　印张：12.25　字数：300 千字
2001 年 8 月第 1 版　2001 年 8 月第 1 次印刷
印数：1—2000 册　定价：22.00 元

前　言

本书是配合国家第四批教改专业“工厂计算机集中控制”专业教改方案编写的。主要介绍了 Intel8086/8088 微处理器、MCS-51 系列、MCS-96 系列单片机组成原理，8086/8088 和 MCS-51 系列单片机的指令系统和汇编语言程序设计方法，MCS-51 的定时器/计数器及应用。

在编写方法上采用将 8086 微处理器和 MCS-51 基本原理、指令系统进行对比的方法展开。每章最后均附有习题，每单元内容后备有自测题一套，共三套自测题，以便使学生更好地掌握所学知识及检测对知识的掌握情况。

全书共分六章，其中第一章、第二章由张华鹏编写，第三章由李涛编写，第四章由卢秉娟编写，第五章、第六章由陈文清编写。

由于编者学识浅薄，书中定有不少缺点和错误，恳切希望读者给予指正。

编　者

2000 年 12 月

目　录

1 计算机基础知识

1.1 序言

学习本书的读者，应学习过与本教材最为密切的两门课程：计算机算法语言程序设计和数字电路。

通过这两门课程的学习，不仅应了解计算机的发展状况、计算机的工作特点及计算机基本工作原理和简单的操作方法，而且还应掌握程序设计的基本方法、基本技能，并且能够利用计算机解决一定的问题。但作为工厂计算机集中控制、电气自动化专业的学生而言，仅仅掌握这些知识是远远不够的，还需要在此基础上进一步深入地学习计算机的内部工作原理，熟练掌握汇编语言程序设计及微机接口等技术。

本教材采用对比方法，主要阐述了 8086 微处理器、MCS-51、96 系列单片机的内部结构及工作原理，并深入浅出地讲解了 8086/8088 和 MCS-51 系列单片机存储器的构造、指令系统以及程序设计等有关问题，关于微机接口技术将在另外一门课程中详细论述。

希望通过本课程的学习，读者能更多更好的把计算机应用在工业控制、仪器仪表及日常生活当中，让计算机更多更好的为人类服务，这也是编写本教材的宗旨。

1.2 常用进制数及其相互转换

本节主要介绍常用十进制数、二进制数、十六进制数的特点和它们之间的相互转换。

1.2.1 常用进制数

1.2.1.1 十进制数(Decimal)

其主要特点是：

(1)有十个不同的数字符号：0、1、2、…9。

(2)它是逢“十”进一。

因此，同一数码处在不同的位置，其代表的数值是不相同的。

如：55.55，个位上的 5 代表的值为 5，十分位上的 5 代表的值为 5×10^{-1}，依次类推，因此 55.55 可以写成：

$55.55=5\times10^{1}+5\times10^{0}+5\times10^{-1}+5\times10^{-2}$

同样，对于任意一个十进制数，均可采用上述的形式进行表示。

1.2.1.2 二进制数(Binary)

其主要特点是：

(1)有两个不同的数码：0 和 1。

(2)它是逢"二"进一。

因此,同十进制数类似,数码处在不同的位置,其代表的值也是不相同的。

如:$(11.101)_2$ 可以写成:

$$(11.101)_2=1\times2^1+1\times2^0+1\times2^{-1}+0\times2^{-2}+1\times2^{-3}$$

1.2.2 常用数码的表示方法

通常用数码下标或用结尾标志来表示不同的进制数。

二进制数可表示为:$(10.11)_2$、$(1101.011)_2$ 或 1011.011B。

十进制数可以带下标或省略下标如:$(129)_{10}$或 129D 或 129 均可以。

八进制数可表示为:$(377)_8$ 或 377Q。

十六进制数可表示为:$(12A)_{16}$或 12AH。

关于八进制数、十六进制数表示方法及其特点可参考二进制数或十进制数的方法推出,在此不再详述。为了熟悉十进制数、二进制数、十六进制数的表示,现就几个简单的数字列出它们的对照表 1.1。

表 1.1 二、十、十六进制间转换表

十进制数	二进制数	十六进制数
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

1.2.3 常用计数制间的相互转换

1.2.3.1 二进制数转换为十进制数

方法:只要将二进制数仿照十进制数形式逐位展开,然后求和。

[例 1.1] $(111.101)_2=1\times2^2+1\times2^1+1\times2^0+1\times2^{-1}+0\times2^{-2}+1\times2^{-3}$

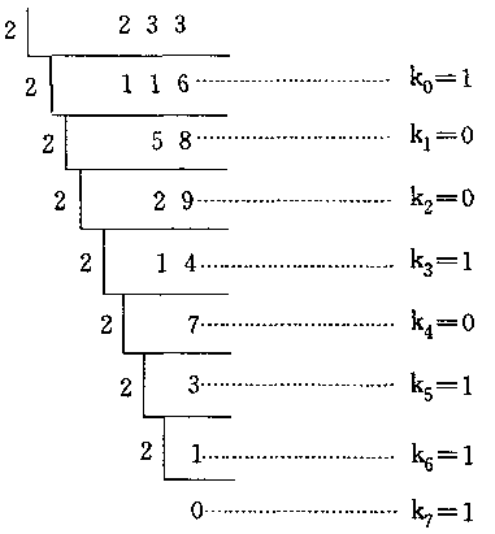
$=(7.625)_{10}$

1.2.3.2 十进制数转换为二进制数

(1)十进制整数转换为二进制数

方法：用 2 不断地去除要转换的十进制数，直至商为 0。每次所得的余数即为二进制数码，最初得到的为整数的最低有效数位记为 k_0，最后得到的为最高有效数位记为 k_{n-1}，则 $k_{n-1}k_{n-2}\cdots k_2k_1k_0$ 即为所得到的结果。

[例 1.2] 求：$(233)_{10}=(\quad)_2$



即 $(233)_{10}=(11101001)_2$

(2)十进制小数转换为二进制小数

方法：不断用 2 去乘被转换的十进制小数部分，将每次所得的整数(0 或 1)，依次用 k_{-1}、k_{-2}、…保留下来。若乘积的小数部分最后能为 0，那么最后一次乘积的整数部分记为 k_{-m}。则：$0.k_{-1}k_{-2}\cdots k_{-m}$ 即为十进制小数的二进制表达式。

若乘积的小数部分不能为 0，这时可以根据精度的要求，保留适当的位数。

[例 1.3] 求：$(0.8125)_{10}=(\quad)_2$

0.8125
× 2
1.6250　　取整数部分 $k_{-1}=1$
0.625
× 2
1.250　　取整数部分 $k_{-2}=1$
0.25
× 2
0.50　　取整数部分 $k_{-3}=0$
0.5
× 2
1.0　　取整数部分 $k_{-4}=1$

即 $(0.8125)_{10}=(0.1101)_2$

(3) 既有整数又有小数的十进制数转换为二进制数

方法：将整数和小数分别进行转换，然后求两者的和，即可得到结果。

[例 1.4] 求：$(233.8125)_{10}=(\quad)_2$

因为$(233)_{10}=(11101001)_2$

$(0.8125)_{10}=(0.1101)_2$

所以$(233.8125)_{10}=(11101001.1101)_2$

1.2.3.3 任意进制数与十进制之间的转换

其方法跟十进制和二进制的相互转换类似。这里以最常用的十六进制数与十进制的转换为例加以说明。

(1)十六进制数转换为十进制数

方法：仿照十进制展开方法，逐位展开再求和。

[例 1.5] 求：$(5A86B.0C)_{16}=(\quad)_{10}$

$(5A86B.0C)_{16}$

$=5\times16^4+10\times16^3+8\times16^2+6\times16^1+11\times16^0+0\times16^{-1}+12\times16^{-2}$

$=370795.046875$

(2)十进制数转换为十六进制数

方法：与十进制数转换为二进制数的方法基本相似。对整数部分用 16 去除，依次取得相应的余数，直至商为 0，对小数部分分别用 16 去乘，保留相应的整数，直至小数部分为零，或根据精度要求选取适当小数位数，然后将两部分相加即可。

[例 1.6] 求：$(370795)_{10}=(\quad)_{16}$

16 | 370795		
16 | 23174		$k_0=11=(B)_{16}$
16 | 1448		$k_1=6=(6)_{16}$
16 | 90		$k_2=8=(8)_{16}$
16 | 5		$k_3=10=(A)_{16}$
0		$k_4=5=(5)_{16}$

即$(370795)_{10}=(5A86B)_{16}$

[例 1.7] 求：$(0.046875)_{10}=(\quad)_{16}$

$$
\begin{array}{rl}
 & 0.046875 \\
\times & \quad 16 \\
\hline
 & 0.75 \qquad \text{取整数部分 } k_{-1}=0 \\
\times & \quad 16 \\
\hline
 & 12.0 \qquad \text{取整数部分 } k_{-2}=C
\end{array}
$$

即 $0.046875=(0.0C)_{16}$

[例 1.8] 求：$(370795.046875)_{10}=(\quad)_{16}$

因为$(370795.046875)_{10}=5A86B+0.0C=5A86B.0C$

所以$(370795.046875)_{10}=(5A86B.0C)_{16}$

1.2.3.4 二进制数与十六进制数的相互转换

在计算机中，数均以二进制形式表示，但二进制数写起来太长，且容易出错，因此书写

时，常采用八进制或十六进制形式表示。在计算机中，由于目前通用的字长为 8 的倍数，如字长为 8 位、16 位或 32 位等，而它们用十六制数表示起来比其他进制数表示更为简洁、方便，因此十六进制数在微型机中的应用更为普通，掌握两者间的相互转换更具有实用意义。

(1)十六进制数转换为二进制数

方法：不论是整数还是小数，只要把每一位 16 进制数用相应的四位二进制数代替，就可以直接将其转换为二进制数。

[例 1.9] 求：$(6A8C)_{16}=(\quad)_2$

因为

6	A	8	C
0110	1010	1000	1100

故$(6A8C)_{16}=(110101010001100)_2$

[例 1.10] 求：$(0.5DCE)_{16}=(\quad)_2$

因为

5	D	C	E
0101	1101	1100	1110

所以$(0.5DCE)_{16}=(0.0101110111001110)_2$

[例 1.11] 求：$(6A8C.5DCE)_{16}=(\quad)_2$

$(6A8C.5DCE)_{16}=(110101010001100.0101110111001110)_2$

(2)二进制数转换为十六进制数

方法：二进制整数部分由小数点向左，每四位一组，最后不足四位的前面补 0，小数部分由小数点向右，每四位一组，最后不足四位的后面补 0，然后把每四位二进制数用相应 16 进制数符代替，即可转换为 16 进制数。

[例 1.12] 求：$(1101101010.010011011)_2=(\quad)_{16}$

0011，0110，1010.0100，1101，1000

3 6 A. 4 D 8

即$(1101101010.010011011)_2=(36A.4D8)_{16}$

1.3 带符号数的表示方法——原码、补码和反码

本节主要介绍有符号数在机内常用的表示方法，以及它们之间的相互关系。

1.3.1 机器数与真值

在前面提到的二进制数，没有提到数的符号，故是一种无符号数的表示方法。但是在实际使用中，数显然有正负之分，那么正负号在机内如何表示呢？通常利用最高位表示符号位，当数为正数时，该位用 0 表示，当数为负数时，该位用 1 表示。这种连同符号位一起表示的数称为机器数，该数的实际值即为其真值。

若字长为 8 位，其表示方法如下所示：

如：$[X]_{真}=+45$　　则：$[X]_{机}=00101101$

$[X]_{真}=-45$　　则：$[X]_{机}=10101101$

此外，为便于运算（把减法运算变为加法运算）和简化机器内部结构，又引入了原码、反码和补码三种机器数代码，下面将以字长为8位介绍各种代码的特点及其相互转换。

1.3.2 原码

按上所述，正数的符号位用0表示，负数的符号位用1表示。这种表示法就称为原码。

如：$[X]_{真}=+107$　　则：$[X]_{原}=01101011$

$[X]_{真}=-107$　　则：$[X]_{原}=11101011$

8位二进制数的原码表示如表1.2所示。它有以下特点：

(1)"0"有两种表示方法

$[X]_{真}=-0$　　$[X]_{原}=10000000$

$[X]_{真}=+0$　　$[X]_{原}=00000000$

这两种表示在机内均按零处理。

(2)8位二进制原码所能表示的数值范围

因为 $[X]_{原}=01111111$　$[X]_{真}=+127$

$[X]_{原}=11111111$　$[X]_{真}=-127$

故8位二进制原码所能表示的数值范围为：+127～-127。

原码表示简单易懂，而且与真值的转换方便。但在运算时，尤其是碰到两个异号数相加（或两个同号数相减）时，就要涉及到减法运算。为简化运算，变减法为加法，又引入了更易于机内运算的反码和补码两种表示方法。

表1.2　八位有符号数原码、补码、反码表

二进制数码表示	无符号二进制数	原　码	补　码	反　码
00000000	0	+0	+0	+0
00000001	1	+1	+1	+1
00000010	2	+2	+2	+2
⋮	⋮	⋮	⋮	⋮
01111100	124	+124	+124	+124
01111101	125	+125	+125	+125
01111110	126	+126	+126	+126
01111111	127	+127	+127	+127
10000000	128	−0	−128	−127
10000001	129	−1	−127	−126
10000010	130	−2	−126	−125
⋮	⋮	⋮	⋮	⋮
11111100	252	−124	−4	−3
11111101	253	−125	−3	−2
11111110	254	−126	−2	−1
11111111	255	−127	−1	−0

1.3.3 反码

正数的反码表示与原码相同，最高位为符号位，用"0"表示，其余位为数值位。

如：$[X]_{真}=+27$　　则：$[X]_{反}=00011011$

$[X]_{真}=+127$　　则：$[X]_{反}=01111111$

负数的反码表示即为它的正数的原码连同符号位一起逐位求反所得。

如：$[X]_{真}=-27$　　则：$[X]_{反}=11100100$

$[X]_{真}=-127$　　则：$[X]_{反}=10000000$

8位二进制数的反码如表1.2所示。它有以下特点：

(1)"0"有两种表示法

$[X]_{真}=+0$　　则：$[X]_{反}=00000000$

$[X]_{真}=-0$　　则：$[X]_{反}=11111111$

(2)8位二进制反码所能表示的数值范围为：+127～-127

注意：当一个带符号数由反码表示时，最高位为符号位。当符号位为0(即正数)时，后面的七位为数值部分，但当符号位为1(即负数)时，一定要注意后面的几位表示的不是此负数的数值，一定要把它们按位取反，才表示它的二进制值。

如：$[X]_{反}=11111011$，其真值：$[X]_{真}=-4$，而$[X]_{真}\neq-123$。

1.3.4 补码

正数的补码表示与原码相同，即最高位为符号位，用"0"表示，其余位为数值位。

如：$[+5]_{补}=00000101$

$[+25]_{补}=00011001$

$[+127]_{补}=01111111$

而负数的补码表示即为其原码的符号位保持不变，剩余位逐位求反加1，或等于其反码，且在最低位加1所形成。

如：$[-25]_{原}=10011001$

$[-25]_{反}=11100110$

$[-25]_{补}=11100111$

$[-127]_{原}=11111111$

$[-127]_{反}=10000000$

$[-127]_{补}=10000001$

$[+0]_{原}=00000000$

$[-0]_{反}=11111111$

$[-0]_{补}=00000000$

8位带符号位数的补码表示也列在表1.2中。它有以下特点：

(1)$[+0]_{补}=[-0]_{补}=00000000$

(2)8位二进制补码所能表示的数值范围为：+127～-128

注意：同反码类似，一个用补码表示的二进制数，当符号位为"1"(即负数)时，其余几

位不是此数的二进制值，应把它们按位取反，且在最低位加 1，才是它的二进制值。

如：$[X]_{补}=10010110$　　$[X]_{真}\neq -22$

因为$[X]_{反}=10010101$　　$[X]_{原}=11101010$

故$[X]_{真}=-(1\times 2^6+1\times 2^5+1\times 2^3+1\times 2^1)=-106$

当负数采用补码表示时，便可以把减法运算转换为加法运算，从而使运算过程和计算机的内部结构得到简化。

[例 1.13]　计算：69－5＝？

因：$69-5=[69]_{补}+[-5]_{补}$

而：$[+69]_{补}=01000101$，$[-5]_{原}=10000101$，$[-5]_{补}=11111011$

于是：

$$\begin{array}{r} 01000101 \\ -00000101 \\ \hline 01000000 \end{array} \qquad \begin{array}{r} 01000101 \\ +11111011 \\ \hline \boxed{1}\,01000000 \end{array}$$

自然丢失

由于在字长为 8 位的机器中，从第七位（最高位）的进位是自然丢失的，故做减法与补码相加的结果是相同的。

[例 1.14]　$5-69=5+(-69)$

$=[5]_{补}+[-69]_{补}$

$[+5]_{补}=00000101$　　$[-69]_{原}=11000101$

$[-69]_{补}=10111011$

于是：

$$\begin{array}{r} 00000101 \\ +10111011 \\ \hline 11000000 \end{array}$$

其和为补码，符号位为 1，肯定和为负数，数值部分应由后七位按位取反再加 1，即为 1000000，则结果等于－64。

补码是人们从实践中总结出来的。日常生活中存在许多相“补”的事例，我们以时钟对准为例。

假若现在是北京时间 8 点整，而时钟却指着 10 点，快两小时，为了将时钟校准，可以用两种拨法，一种是从 10 点钟倒拨两小时，相当于做减法，即 10－2＝8。但是如果这台钟由于设计上的原因规定只能顺拨，即相当于做加法，那么，如何校准呢？显然我们可以从 10 点开始顺拨 10 小时，即：10－2＝10＋10。结果为什么正确呢？因为

$$10+10=\boxed{12}+8$$

自然丢失

从钟面上我们看到，当时钟顺拨时，经过 12 点然后从 0 重新开始，相当于自动丢掉了一个 12。这个自动丢掉的数 12 称为时钟的模（modulus），用符号 mod 表示，它是一个计量系统的量程，即是此系统所能表示的数的极限，它是自动丢失的。时钟以 12 为模，凡是 12 的整数倍都可丢失，因此以 12 为模时，12 等价于 0。

我们称 10－2 与 10＋10 对模 12 是同余或称（－2）与（＋10）对模 12 互为补数。

上例的情况可以扩充到一般情形。对于某一确定的模，某数减去一个正数，总可以用加上该数的负数的补码来代替，因而相当于变减法为加法。

对字长为 8 位的二进制数字系统，其模为 $2^8=256$。若有

$$\begin{aligned}64-20&=64+(-20)\\&=64+[256-20]\\&=64+236\\&=\boxed{256}+44\end{aligned}$$

自然丢失

故 $64-20=44$

可见在字长为 8 位(模为 2^8)的情况下，$64-20$ 与 $64+236$ 的结果是相同的，所以 (-20)与 236 互为补数，而 236 就是上述的(-20)的补码。所以在上述负数的补码表示中，实际上我们是利用了补码的概念，把减法转换为加法，而所有负数 X 的补码都可由模 2^8-X 来得到。但我们是利用了把正数连同符号位一起求反，然后再加 1 这样的简便方法来得到的，避免了求补码过程中的减法，使补码的运算具有实用价值。而此，在微型机中，凡是带符号的数一律是用补码表示的，所以计算机内给出的结果也是以补码形式给出的。这一点务必引起注意。

由于计算机的字长是有限制的，所以无论是用原码、反码或补码，所能表示的数据范围均是有限的，当表示的数超出此范围时，就无法表示，尤其当结果超过此范围时，结果就会出现错误，这种现象称为溢出。解决的唯一途径是改变模的大小，即用二字节(字长相当于 16 位，模为 2^{16})、四字节(字长相当于 32 位，模为 2^{32})或多字节来表示数值，即变相地改变字长，扩大所表示的数据范围，满足实际需要。

如字长为 16 位时，原码、反码、补码所能表示的数据范围为：

原码：$+(2^{15}-1)\sim-(2^{15}-1)$

反码：$+(2^{15}-1)\sim-(2^{15}-1)$

补码：$+(2^{15}-1)\sim-(2^{15})$

1.3.5 补码的运算规则

(1)一个用补码表示的负数，如将$[X]_{补}$再求一次补，即将$[X]_{补}$除符号位外逐位取反加 1，就可以得到$[X]_{原}$。可用下式表示为：

$$[[X]_{补}]_{补}=[X]_{原}$$

[例 1.15] $[X]_{原}=10010101$

$[X]_{补}=11101011$

$[[X]_{补}]_{补}=10010101$

(2)两个 n 位二进制数之和的补码等于该两数的补码之和，即

$$[X+Y]_{补}=[X]_{补}+[Y]_{补}$$

同理：

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