微型计算机原理与使用教程

郭 立 金大胜 编著

中国科学技术大学出版社

微型计算机原理与使用教程

郭　立　金大胜　编著

中国科学技术大学出版社
1998・合肥

微型计算机原理与使用教程

郭　立　金大胜　编著

中国科学技术大学出版社出版

（安徽省合肥市金寨路 96 号　邮编：230026）

安徽省金寨县印刷厂印刷

全国新华书店经销

*

开本：850mm×1168mm　1/32　印张：7.75　字数：202 千

1998 年 5 月第 1 版　　1998 年 5 月第 1 次印刷

印数：1—8000 册

ISBN 7-312-00950-6/TP・194　**定价：10.00 元**

内 容 简 介

本书为一本微机应用入门教材，为初学者而写，为初学者所用. 从数制和编码讲起，深入浅出地介绍了微型计算机的基本原理；汇编语言程序设计的基本方法和技巧；输入/输出与中断的概念；DOS、BIOS 功能调用的方法以及基本接口方法等.

本书内容丰富，言简意赅，实用性强，通俗易懂. 可作为高等院校非电子信息类专业学生教材，也可作为微机原理及汇编程序设计的培训教材.

前言

近几年来，我国的计算机技术发展很快，大批科技工作者、大学生以及各行业在职人员迫切需要学习、掌握计算机知识与应用技术. 在广为流行与普及的微型计算机中，比较流行的是以 Intel 的 8088/8086 系列以及 Motorola 的 6800 系列为 CPU 的微型计算机系列. 从学习的角度，8088/8086 系列和 6800 系列 CPU 是当前迅速发展的 CPU 系列的基础，只有掌握了 8088/8086 以及 6800 才能进一步掌握各种更为先进的 CPU 和计算机应用技术.

本书注重理实交融，给出了大量插图、例题，力图使读者较快掌握微型计算机的基本原理，并在系统地介绍微型计算机原理的基础上，突出对微型计算机应用技术的讲述，介绍了与微型计算机相关的外围部件和设备以及它们的接口知识. 内容深入浅出，简明扼要，重点突出.

为了充分利用计算机硬件特性，更加有效地直接控制计算机资源，以达到编制占用内存少、执行速度快，运行效率高的程序之目的，读者应掌握使用汇编语言进行编程的技术. 本书在介绍了汇编语言的基础知识后，通过丰富的实例，形象地介绍了汇编语言程序设计的思想、基本方法和编程技巧，除了系统介绍程序设计的基本方法

外，还对程序设计中难以理解，容易产生混淆的内容进行了深入的讨论.

本书的这些特点，使得本书具有较强的可读性和实用性. 因此，本书不是简单罗列技术的手册或使用说明，而是从教学需要的角度出发，按照人的认识规律，使之具有系统性、完整性，且由浅入深、循序渐进. 本书引入了大量实例、图表，正是为了实现这一既定目标服务.

本书就内容构成来说，主要由 3 部分组成，其中，基础知识部分和涉及 Motorola 的 6800 系列的内容由金大胜撰稿，涉及 Intel 的 8088/8086 系列的内容由郭立撰稿.

本书在编写过程中，得到了中国科学技术大学电子技术部、成人教育学院领导和老师们的大力支持，在此表示衷心的感谢.

编者尽力想将该书编写得简单明了，方便自学，但由于水平所限，疏漏之处，在所难免，敬请读者和同行批评指正.

编　者

1998 年 1 月

目　次

第一章　数制和编码

微处理机的基本语言是二进制数及编码，另外一些制数如八进制数、十六进制数采用二进制数表示和处理又极为方便. 因此，了解并熟悉二进制数及编码，对于学习微处理机来说是非常重要的. 二进制数与十进制数既有区别又有联系. 从大家熟悉的十进制数入手，将其基本概念引伸到二进制数、八进制数和十六进制数，这对于充分理解及掌握微处理机的基本语言是非常合适的.

1.1　十进制数制

区别一种数制的基本特征是底数. 底数表示所用的字符或数码的数目，这些字符和数码表示数制中量的大小. 十进制数制用 0～9 共 10 个数码表示量的大小，故底数为 10. 一般采用在数的右下角标注底数的方法来说明所用的数制，例如，2301_{10}表示以 10 为底数的数制.

1. 按位计数法

在十进制数制中，每位数均可表示成以 10 为底的 n 次幂，即个位数为 10^0，十位数为 10^1，百位数为 10^2，等等，依次类推. 因此，依照按位计数法，任意一个十进制数的数值可以用每位数字乘以它所在位的位数再相加来得到. 例如，十进制数 2301_{10}可表示为：

$(2\times10^3)+(3\times10^2)+(0\times10^1)+(1\times10^0)=2000+300+0+1=2301_{10}$.

2. 小数(分数)

十进制小数也可用同样的方法来表示,只是在小数点后面的位数都是 10 的负次幂,例如,$0.1=10^{-1}$,$0.01=10^{-2}$,等等.

所以,如果一个数是由小数点分成整数和小数两部分组成的话,那么小数点左边的整数部分其所在位数(自右向左依次是 $10^0,10^1,10^2,\cdots\cdots$;小数点右边的小数部分其所在位数是自左向右依次是 $10^{-1},10^{-2},10^{-3},\cdots\cdots$. 例如十进制数 278.94,用按位计数法表示为:$(2\times10^2)+(7\times10^1)+(8\times10^0)+(9\times10^{-1})+(4\times10^{-2})=(2\times100)+(7\times10)+(8\times1)+(9\times0.1)+(4\times0.01)=200+70+8+0.9+0.04=278.94$. 在这个例子中,最左边的数字 (2×10^2)是最大有效数字(MSB),因为在确定数值时,它的影响最大. 最右边的数字,称为最小有效数字(LSB),因为在确定数值时,它的影响最小.

1.2 二进制数制

二进制数制只包括两个元素或状态,即 1 和 0. 由于比较简单,微处理机就用它来处理数据.

1. 按位计数法

和十进制数制一样,二进制数的每一位所在的位置均具有确定数值大小的作用,一般用数制底数 2 的 n 次幂来表示,对于二进制小数则可用 2 的负 n 次幂来表示. 例如对于二进制数 1101_2,其按位计数为:$(1\times2^3)+(1\times2^2)+(0\times2^1)+(1\times2^0)$,具体计算出来的值为:$1\times8+1\times4+0\times2+1\times1=8+4+1=13$,即为二进制数 1101_2 所表示的十进制数. 而对于二进制小数 0.1101 按位计数表示为:$(1\times2^{-1})+(1\times2^{-2})+(0\times2^{-3})+(1\times2^{-4})$,计算出来的值:$0.5+0.25+0+0.0625=0.8125$,即为该二进制小数转化成的十进制数.

2 的 n 次幂简表为：

$2^0=1_{10}$　　$2^6=64_{10}$

$2^1=2_{10}$　　$2^7=128_{10}$

$2^2=4_{10}$　　$2^8=256_{10}$

$2^3=8_{10}$　　$2^9=512_{10}$

$2^4=16_{10}$　　$2^{10}=1024_{10}$

$2^5=32_{10}$　　$2^{11}=2048_{10}$

2 的负 n 次幂简表为：

$2^{-1}=\frac{1}{2}=0.5_{10}$　　$2^{-5}=\frac{1}{32}=0.03125_{10}$

$2^{-2}=\frac{1}{4}=0.25_{10}$　　$2^{-6}=\frac{1}{64}=0.015625_{10}$

$2^{-3}=\frac{1}{8}=0.125_{10}$　　$2^{-7}=\frac{1}{128}=0.0078125_{10}$

$2^{-4}=\frac{1}{16}=0.0625_{10}$　　$2^{-8}=\frac{1}{256}=0.00390625_{10}$

2、二进制数和十进制数的转换

在使用微机的过程中，经常需要进行二进制数和十进制数之间的转换，下面简单介绍转换的办法.

(1) 二进制转换成十进制

把二进制数转换成相应的十进制数，只要将二进制中出现 1 的所在位的位数值相加即可. 例如把二进制数 101101.11 转换成相应的十进制，则为：

二进制数	1	0	1	1	0	1	1	1	
各位位数	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	
十进制数	32+	0+	8+	4+	0+	1+	0.5+	0.25	=45.75

(2) 十进制转换成二进制

只要把十进制数依次除以 2，并记下每次所得的余数(余数总是 1 或 0)，这个余数即为相应的二进制数.

例如，把十进制数 25 转换成二进制数：

25÷2＝12　　余数为 1←LSB

12÷2＝6　　0

6÷2＝3　　0

3÷2＝1　　1

1÷2＝0　　1←MSB

把十进制数除以 2，并记下余数，所得的商再除以 2，并记下余数，如此继续下去，直到商得 0 为止. 然后，收集余数，从余数的末位(MSB)开始，直到余数的第一位(LSB)为止，所以 $25_{10}=11001_2$. 请注意，余数一定要按颠倒顺序收集，即第一位余数是最低位，而末位余数是最高位.

要将一个十进制小数转换成不同底数的数时，应把所需的底数连续不断地乘以该十进制小数，并且记录所得的整数部分，直至小数部分得 0 为止. 例如，将十进制数 0.3125 转换成相应的二进制数，则用 2 重复乘：

0.3125×2＝0.625⇒0.625　　整数为 0←MSB

0.625×2＝1.250 ⇒0.250　　1

0.2500×2＝0.500⇒0.500　　0

0.500×2＝1.000 ⇒0　　1←LSB

将每次乘得的结果中小数点左边的个位上的 0 或 1 记录下来，就组成了相应的二进制小数. 当 0.3125 乘以 2 时，个位数为 0，这个 0 就是相应的二进制小数的最高位. 然后把 0.625 再乘以 2，得到 1.250，个位数为 1，这个 1 就是相应的二进制小数的次最高位. 而在下一步乘法时，如须用乘积减去 1 得到的 0.25 去乘以 2. 依次继续下去，直到小数得 0 为止. 有时结果永不为 0，则只要转换到所要求的精度为止即可. 将一开始得到的小数点左边的整数值写在二进制小数点后的第一位，并继续写到最低位. 所以有结果：$0.3125_{10}=0.0101_2$.

如果十进制数包含整数和小数两部分，则必须将十进制小数

点两边的整数部分和小数部分分开，分别完成相应的转换，然后，再把二进制整数和小数部分组合在一起. 例如，将十进制数 14.375_{10} 转换成相应的二进制数：

$14.375_{10}=14_{10}+0.375_{10}$

$14\div2=7$　　余数为 0←LSB

$7\div2=3$　　1

$3\div2=1$　　1

$1\div2=0$　　1←MSB

所以　$14_{10}=1110_2$

$0.375\times2=0.75\Rightarrow0.75$　　整数为 0←MSB

$0.750\times2=1.50\Rightarrow0.50$　　1

$0.500\times2=1.00\Rightarrow0$　　1←LSB

所以　$0.375_{10}=0.011_2$

因而　$14.375_{10}=14_{10}+0.375_{10}=1110_2+0.011_2$

$=1110.011_2$

1.3　八进制数制

八进制是微处理机中使用的又一种数制，它的底数是 8，用数字 0～7 表示. 这些数具有和十进制 0～7 一样的数值. 和十进制数制一样，八进制数的每一位所在的位置均具有确定数值大小的作用，一般用数制底数 8 的 n 次幂来表示，对于八进制小数则可用 8 的负 n 次幂来表示. 例如，八进制数 372.01 用按位计数法可写成下列形式：

$(3\times8^2)+(7\times8^1)+(2\times8^0)+(0\times8^{-1})+(1\times8^{-2})$

所得结果即为十进制数，即：

$(3\times64)+(7\times8)+(2\times1)+(0\times0.125)+(1\times0.015625)$

$=192+56+2+0+0.015625=250.015625_{10}$

8的n次幂简表为：

$8^0=1_{10}$　　$8^{-1}=0.125_{10}$

$8^1=8_{10}$　　$8^{-2}=0.015625_{10}$

$8^2=64_{10}$　　$8^{-3}=0.001953125_{10}$

$8^3=512_{10}$　　$8^{-4}=0.000244140625_{10}$

$8^4=4096_{10}$

$8^5=32768_{10}$

$8^6=262144_{10}$

1. 十进制数转换成八进制数

转换方法与把十进制数转换成二进制数的方法相同，只是底数是8而不是2. 例如，将十进制数194转换成相应的八进制数，可按下述步骤来进行：

194÷8=24　　余数为 2←LSB

24÷8=3　　0

3÷8=0　　3←MSB

将十进制数除以8，记下其余数，余数可以是0～7中的任意数，然后将商数再除以8，记下其余数，连续除到结果得0为止. 最后，从最高位(MSB)开始到最低位(LSB)为止，收集余数即为八进制数. 所以，该例中的十进制数194_{10}转换成的八进制数为302_8.

若将十进制小数转换成八进制小数，则需将十进制小数逐次乘8，并且记录所得的整数部分，直到乘积的小数部分结果得0为止. 如果永远得不到0，则只需继续到所需的精度为止. 例如，把十进制小数0.46875转换成相应的八进制数：

0.46875×8=3.75　　整数为 3 ← MSB

0.25000×8=6.00　　整数为 6 ← LSB

将十进制数乘以8，当乘积超过1时，需减去乘积的整数部分，用剩下的小数部分再乘以8.

再举一例. 将十进制小数的0.136转换成相应的八进制小数，其精确度到小数点后四位：

$0.136\times8=1.088$　　整数为 1←MSB

$0.088\times8=0.704$　　0

$0.704\times8=5.632$　　5

$0.632\times8=5.056$　　5←LSB

$0.136_{10}\doteq0.1055_8$

十进制转换成八进制应包括整数和小数两部分. 所以，必须将整数和小数分开，分别进行相应的转换，然后，再将八进制整数和小数部分合并. 例如，将十进制数 124.78125 转换成八进制数：

$124.78125_{10}=124_{10}+0.78125_{10}$

$124\div8=15$　　余数　4← LSB

$15\div8=1$　　7

$1\div8=0$　　1←MSB

$124_{10}=174_8$

$0.78125\times8=6.25$　　整数为 6←MSB

$0.25000\times8=2.00$　　2←LSB

$0.78125_{10}=0.62_8$

所以，结果为：

$124.78125_{10}=124_{10}+0.78125_{10}=174_8+0.62_8=174.62_8$

2. 八进制数与二进制数的转换

微处理机是用二进制来处理数据的. 但是，当数据量较大时，为了加速并简化数据的输入和显示，常常用其它的数制作为二进制的速写形式，八进制数制是其中之一.

从十进制、八进制和二进制数的比较表上可明显看出，最多用 3 位二进制数就可以表示八进制的八个数字符. 所以，能够用 3 位二进制数表示 1 位八进制数. 例如：

$101_2=(1\times2^2)+(0\times2^1)+(1\times2^0)=4+0+1=5_8$

由于这个关系，二进制数转换成八进制数是很简单的. 例如，把二进制数 101001 转换成八进制数：

101001

改写成

MSB LSB

↓ ↓

101 001

得到

51_8

由此可见，将二进制数转换成八进制数，只要将数分开，从最低位开始分成3位一组，然后将每组转换成相应的八进制数即可.

用同样方法也能将二进制小数转换成八进制小数，只是二进制数必须从小数点右边最高位开始分成3位一组. 例如，将二进制小数0.011101转换成八进制小数：

0.011101_2

改写成

MSB LSB

↓ ↓

0.011 101

得到

0.35_8

在把二进制数分成3位一组位数不够时，则需补0，直至达到可分成3位一组的数为止. 例如，将二进制数10010101.1011转换成八进制数：

10010101.1011_2

改写成

MSB LSB

↓ ↓

010 010 101.101 100

得到

225.54_8