微型计算机系统与应用基础

孙家启　吴国凤　编著
胡正家　于之硕　主审

兵器工业出版社

（京）新登字049号

本书是根据国家教委制订的《微型计算机系统与应用基础》课程教学要求而组织编写的试用教科书。

本书从实际出发较全面介绍微型计算机系统知识和应用软件开发的手段。全书共八章，即微型计算机系统概览，程序设计基础——PASCAL语言，汉字信息处理与操作系统，汉字编码输入技术——王码体系，本文编辑，表格处理，汉字dBASEⅢ数据库管理系统，微型计算机绘图软件——Auto CAD。书中还提供了较详细的PC TOOLS、CCBIOS 2.13和DEBUD等开发与应用资料。

本书文字通俗易懂，深入浅出，内容丰富。既可作为高等院校试用教材，又可供提高工程、管理技术人员应用微型机能力而编写的一本专著。

微型计算机系统与应用基础

孙家启　吴国凤　编著

*

兵器工业出版社　出版社发行

（北京市海淀区车道沟10号）

新华书店总店北京科技发行所经销

建新印刷厂印装

*

开本787×1092 1/16　印张：21　字数：518千字

1993年3月第1版。1993年3月第1次印刷

印数：00，001—5600　·定价13.50元

ISBN 7-80038-559-0/TP·40

前　言

本书是根据国家教育委员会“全国高等学校工科计算机基础课程教学指导委员会”制订的“微型计算机系统与应用基础”课程教学要求，以及编著者多年来的教学实践、应用开发中的经验和科技咨询中了解到社会需要而编写的一本专著。

本书既是高等学校非计算机专业大学生继高级语言程序设计课之后需更进一步提高微型计算机系统中硬件知识和软件开发能力而开设的“微型计算机系统与应用基础”课程的试用教材，又是行政管理人员、办公自动化人员、科技人员在本行业内进行微型计算机系统应用和开发的有益的参考书。

本书编写过程中，得到了国家教育委员会“全国高等学校工科计算机基础课程教学指导委员会”、机械电子工业部“高等院校计算机基础教育究研会”的支持；并得到合肥工业大学教材科长郑象鹄工程师对本书出版给予关心和支持；全国高等学校工科计算机基课程度数学指导委员会主任委员西安交通大学胡正家教授和哈尔滨电工学院计算机系于之硕副教授对本书进行了主审，并提出了有益的建议和修改意见，在此一并表示衷心的感谢。

由于编著者的水平和收集资料有限，书中的错误和不妥之处在所难免，迎欢专家和读者提出批评指正。

编著者

1992年9月

目　录

第一章　微型计算机系统概览

§1-1　微型计算机系统的基本概念

近年来，我国的计算机应用事业迅速发展，大批科技人员、大中学生、管理人员，以及各行各业的在职人员都迫切要求学习微型计算机知识，他们已经认识到，微型计算机知识是当代知识分子的知识结构中不可缺少的重要组成部分。

学习微型计算机的目的，对大多数人来说，是为了自己在工作中应用它。为了用好微型计算机，充分发挥微型计算机的作用，有必要了解整个系统及其组成部分的功能和基本工作原理以及整个系统的特点。

一、微型计算机系统

微型计算机系统是一个很复杂的系统。为此，首先要问什么是系统呢？我们把为了达到某些特定的目标（或功能），或为了完成特定的任务而把若干组成部分有机地联系起来的一个整体称之为系统。组成部分可以是机器、人、工作程序、方法或规程等。

为了弄清楚什么是微型计算机系统，下面首先从组织角度看，看它是由哪些部分组成的？微型计算机系统一般认为是由硬件系统和软件系统两大部分组成。硬件系统是指哪些与组成微型计算机系统而有机联系起来的电子的、电磁的、机械的、光学的元件、部件或装置的总和，它们一般是有形的物理实体。软件系统是相对硬件系统而言的。软件系统实质上是微型计算机程序、方法、规则、相关的文档以及在微型机上运行时所必需的数据。硬件是微型计算机系统的物质基础。没有硬件，谈不上应用微型计算机。但是，光有硬件而没有合适的软件，微型计算机也只是毫无用处的摆设。所以硬件和软件是相辅相成的，它们配合起来才能完成系统的主人交给的任务。

硬件系统和软件系统本身还可细分成更多的子系统。见图1-1所示。

为了便于学习和掌握这样复杂的系统，我们采用分而治之的办法，把整个系统分成许多小部分，逐个学习。

从功能看，微型计算机系统是处理信息的系统，见图 1-2 所示。它的输入以某种形式表示的信息，经加工处理后得到所需要的信息，并将其以外界所能接受的形式输出。在一定意义上，可把微型计算机系统比喻为信息加工厂。只是普通的加工厂用的原材料是有形的物质，如煤、原油等；加工方法是物理和化学处理方法；加工目的是改变物理和化学性能，使产品有使用价值；所获得的产品是物质的实体，如煤气、石油等。而微型计算机的输入输出则是以某种编码形式表示的信息，加工只是对输入的信息进行分析、运算、处理。因此微型机和一般工厂的差别主要是送进工厂的原材料，加工方法和得到的产品不一样，但从处理概念上看却是相似的。

微型计算机系统的输入或输出信息的形式是多种多样的。例如，它们可能是数字信息、文字信息、图象信息、声音信息、符号信息或是以某种形式表示的温度、压力、位移、速度、

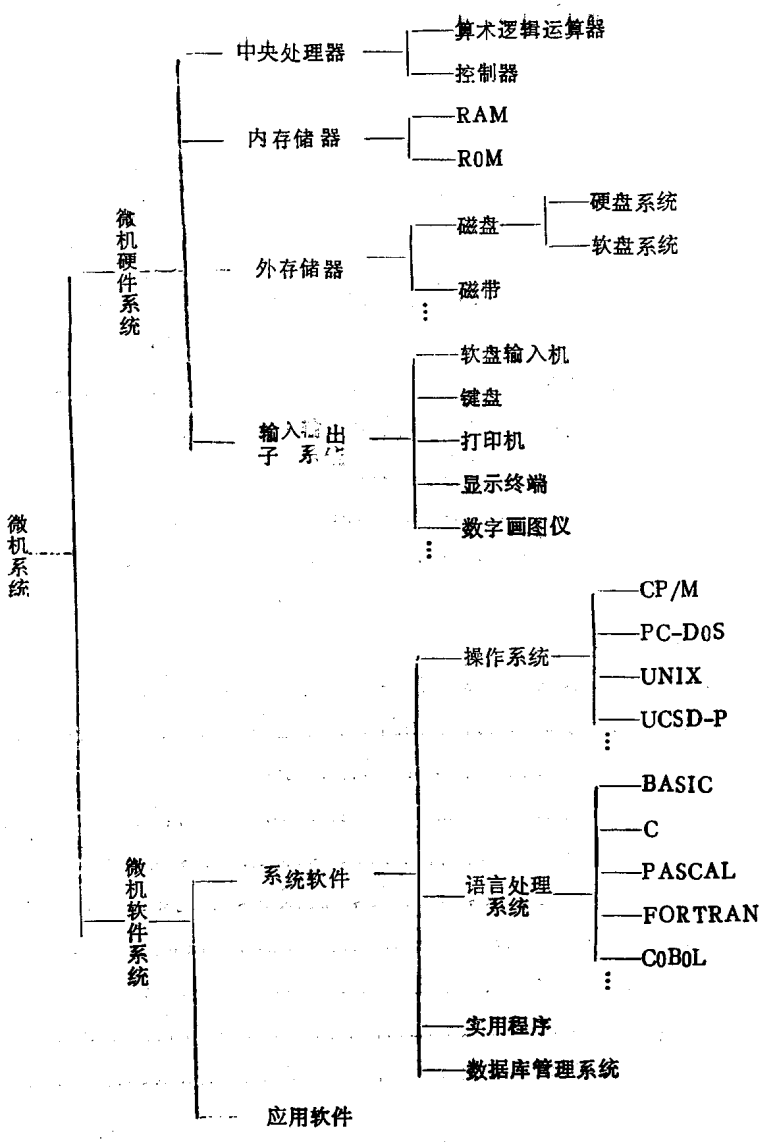

图1-1 微机系统的组成

转角或电压等各种物理量的信息。微型机系统输入和输出信息的多样化正反映了微型计算机用途的广泛性：例如，当微型计算机用于财务部门时其输入信息主要是数字或文字，而输出信息则是由数字和文字组成的帐单和报表；当微型计算机用于办公室自动化时，其输入／输出的信息可能是声音，图像和文字综合；当微型计算机用于生产过程的自动控制时，除了上述文字、数字和图像信息外，还必须有生产过程的各种参数（如温度、压力、流量、组分等）的输入和各种起控制作用的信息输出，如开启某个控制阀，设置某参数为新的要求值等。

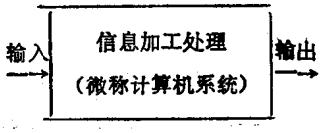

图1-2 微型计算机系统的功能

顾名思义，微型计算机是一种先进的计算工具，它能迅速地计算和解决复杂的数学问题，得出精确的计算结果，广泛应用于科学技术和工程的计算中。但是，随着计算技术的发

展，目前微型计算机的应用已大大超出数值计算的范围，而广泛地应用在非数值计算机的信息处理或其它形形色色的应用领域中。

总之，从一般的意义上讲，微型计算机是用于处理信息的机器。但是用户在购买微型计算机时，应当有明确而具体的目标和用途，当您选购微型机时，应根据用途选择适当硬件配置和软件，以充分发挥微型机系统的效能。

二、微型计算机的数据信息表示法

数据信息是微型计算机加工和处理的对象。它一般可分为数值数据（如定点数、浮点数）和非数值数据（如字符串）两大类。

数据信息的表示法将直接影响微型机的结构和性能。所以，确定微型计算机的数据信息的表示方法，是设计一个微型机系统必须解决的首要问题。它包括选用何种进位计数制；带符号数的表示方法；小数点的表示方法；字符的编码方法等。

（一）数的进位系统

1.常用的进位制

人们常常采用由低位向高位的进位来进行计数，这种表示数的方法称为进位计数制。在进位计数制中，所用到的数字符号的个数叫做基本数。一个数字符号在不同的数位，所表示的数值是不同的；每个数字符号所表示的数值等于它本身的数值乘以一个与它所在数位对应的常数，这个常数叫做位权，简称为权。常用的进位制有十进制、二进制、八进制、十六进制等。

(1)十进制表示

十进制是人们最熟悉的一种进位计数制。它有0～9十个数字符号，即基数为10；每个数位计满10就向高位进位，即逢十进一。任何一个十进制数都可以用一个多项式来表示。

例如：$1505.95=1\times10^3+5\times10^2+0\times10^1+5\times10^0+9\times10^{-1}+5\times10^{-2}$

显然，上式各位的权分别为：10^3、10^2、10^1、10^0、10^{-1}、10^{-2}。任意一个十进制数$(N)_十$的多项式表示为：

$$(N)_十=K_n\cdot10^n+K_{n-1}\cdot10^{n-1}+\cdots+K_0\cdot10^0+K_{-1}\cdot10^{-1}+\cdots+K_{-m}\cdot10^{-m}=\sum_{i=n}^{-m}K_i\cdot10^i$$

式中，K_i是0～9中一个数字符号，由$(N)_十$决定，m和n为正整数。

(2) 二进制表示

目前微型机普遍采用二进制表示。因为二进制表示便于实现，具有两种稳定状态的物理器件很多；二进制数的运算简单，数据的传送和存储可靠，运算线路易于实现，而且十分经济，故得到广泛应用。

任意一个二进制数可用多项式表示为：

$$(N)_二=K_n\cdot2^n+K_{n-1}\cdot2^{n-1}+\cdots+K_0\cdot2^0+K_{-1}\cdot2^{-1}+\cdots+K_{-m}\cdot2^{-m}=\sum_{i=n}^{-m}K_i\cdot2^i$$

（m，n为正整数）

式中，K_i为0或1，故基数为2，且逢二进一。因为二进制表示数的容量最小，往往一个较大的数值要用许多位二进制数表示，书写不便。为此，人们常常采用八进制，十六进制表示，

多用于机器语言程序的书写。

(3) 八进制表示

任意一个八进制数可用多项式表示为：

$$(N)_{八}=K_n\cdot 8^n+K_{n-1}\cdot 8^{n-1}+\cdots K_0\cdot 8^0+K_{-1}\cdot 8^{-1}+\cdots+K_{-m}\cdot 8^{-m}=\sum_{i=n}^{-m}K_i\cdot 8^i$$

(m、n为正整数)

式中，K_i可取0～7八个数字符号之一，由 (N)八决定；其基数为8；且逢八进一。

因为$2^3=8^1$故可把三位二进制数用一位八进制数表示，以缩短二进制表示的位数。

(4) 十六进制表示

任意一个十六进制数可用多项式表示为：

$$(N)_{十六}=K_n\cdot 16^n+K_{n-1}\cdot 16^{n-1}+\cdots+K^0\cdot 16^0+K_{-1}\cdot 16^{-1}+\cdots+K_{-m}\cdot 16^{-m}=\sum_{i=n}^{-m}K_i\cdot 16^i$$

(m、n为正整数)

式中，K_i可取0，1…，9，A，B，C，D，E，F十六个数字符号之一，由 (N)十六决定；其基数为16，且逢十六进一。

因为$2^4=16^1$，故可把四位二进制数用一位十六进制数表示，大大缩短二进制表示的位数。在微型机中信息常用字节（即8位二进制数）表示，那么一字节可表示为二位十六进制数。

综上所述，任意一个r进制数 (N)r可以表示为：

$$(N)_r=K_nr^n+K_{n-1}r^{n-1}+\cdots\cdots K_0r^0+K_{-1}r^{-1}+\cdots+K_{-m}r^{-m}$$

$$=\sum_{i=n}^{-m}K_ir^i=\sum_{i=n}^{0}K_ir^i+\sum_{i=-1}^{-m}K_jr^i$$

(整数部分) (小数部分)

式中，K_i可以取0至r－1中一个数字符号，由 (N)r来决定，m和n为正整数。

2.各种进位制的相互转换

(1) 按“权”展开法

任意一个r进制数转换为相应的十进制数，只要把各位数字与它的权相乘，再把乘积相加，就得到十进制数。

例如：

$$(100111.1011)_{二}=1\times 2^5+0\times 2^4+0\times 2^3+1\times 2^2+1\times 2^1+1\times 2^0$$
$$+1\times 2^{-1}+0\times 2^{-2}+1\times 2^{-3}+1\times 2^{-4}$$
$$=(35.6875)_{十}$$

$$(37.2)_{八}=3\times 8^1+7\times 8^0+2\times 8^{-1}=(31.25)_{十}$$

$$(4E6.C)_{十六}=4\times 16^2+14\times 16^1+6\times 16^0+12\times 16^{-1}=(1254.75)_{十}$$

(2) 基数乘除法

要把十进制数转换为相应的r进制数，需要对它的整数部分和小数部分分别进行转换，

再把它们拼接起来实现。

1)整数部分的转换　把十进制整数部分连续除以所要转换进制数的基数，直到除完（商为0）为止，依次求出各次除法的余数，即为所要转换的整数。

现以十进制整数转换为二进制整数加以说明。因为

$$\frac{N}{2}=(K_n\cdot 2^{n-1}+K_{n-1}\cdot 2^{n-2}+\cdots+K_1\cdot 2^0)+\frac{K_0}{2}$$

上式中括号内为商，K_0为余数。若余数为0，则$K_0=0$；若余数为1，则$K_0=1$。依此类推；不断除2可得到$K_1\sim K_n$，直到商为0止。

例如：

2 | 35	……余数1	K_0	↑ 读余数方向
2 | 17	……余数1	K_1	
2 | 8	……余数0	K_2	
2 | 4	……余数0	K_3	
2 | 2	……余数0	K_4	
2 | 1	……余数1	K_5	
0			

$$\therefore\quad (35)_{十}=(K_5K_4K_3K_2K_1K_0)_{二}=(100011)_{二}$$

2)小数部分的转换　把十进制小数部分连续乘以所要转换进制的基数，依次取出各次乘积的整数部分，直到小数部分为0或达到所要求的精度为止。

现在以十进制小数转换为二进制小数加以说明。因为

$$2N=K_{-1}+(K_{-2}\cdot 2^{-1}+K_{-3}\cdot 2^{-2}+\cdots+K_{-m}\cdot 2^{-m+1})$$

上式括号内为小数部分，K_{-1}为整数部分，再将括号内的内容乘以2，取出整数部分，依此类推，不断乘2，就可以得到$K_{-2}\sim K_{-m}$。　例如：

提出整数		0.6875	
		× 2	
读整数方向	K_{-1}1……	|1|.3750	
		× 2	
	K_{-2}0……	|0|.7500	
		× 2	
	K_{-3}1……	|1|.5000	
		× 2	
	K_{-4}1……	|1|.0000	
		× 2	
↓	K_{-5}0……	0.0000	

$$\therefore\quad (0.6875)_{十}=(0.1011)_{二}$$

$$故\quad (35.6875)_{十}=(100011.1011)_{二}$$

以上讨论了十进制与其他进制数的互换，其他进制数之间的互换也可按照上述方法进行，在此就不详细介绍了。

（二）数的编码表示

人们通常在数据的前面加上“+”、“−”号来表示数的正、负。但在微型机中所表

示的数或信息均已数码化了，所以数的正、负号也要用0、1来表示。这种在机器中使用的连同数符一起数码化的数称为机器数，它所对应的原来书写形式表示的数称为机器数的真值。

在微型机中使用的定点数有二种：一种是定点小数，即小数点固定在数值最高位之前；另一种是定点整数，即小数点固定在数值最低位之后。下面就来讨论几种常见的定点数编码表示法。

1.原码表示法

原码表示法是一种较简单的机器数表示法。用最高位表示符号，符号位为0表示该数为正；符号位为1表示该数为负，其有效数值部分则用二进制数的绝对值表示。由此可得出原码表示定义：

设定点小数的原码形式为$x_0, x_1x_2\cdots x_n$，则

$$[x]_{原}=\begin{cases}x & 0\leqslant x<1\\ 1-x=1+|x| & -1<x\leqslant 0\end{cases}$$

设定点整数的原码形式为$x_0, x_1x_2\cdots\cdots x_n$，则

$$[x]_{原}=\begin{cases}x & 0\leqslant x<2^n\\ 2^n-x=2^n+|x| & -2^n<x\leqslant 0\end{cases}$$

式中，x表示真值，$[x]_{原}$表示原码。当x为正时，$[x]_{原}$与真值在表示形式上完全一样；当x为负时，$[x]_{原}$与真值在表示形式上的区别是符号位为1，真值0在原码中有两种表示形式，即+0和−0之分。

在定点小数中：$[+0]_{原}=0.00\cdots0$；$[-0]_{原}=1.00\cdots0$；

在定点整数中：$[+0]_{原}=000\cdots0$；$[-0]_{原}=100\cdots0$。

真值x与$[x]_{原}$关系见图1-3所示。

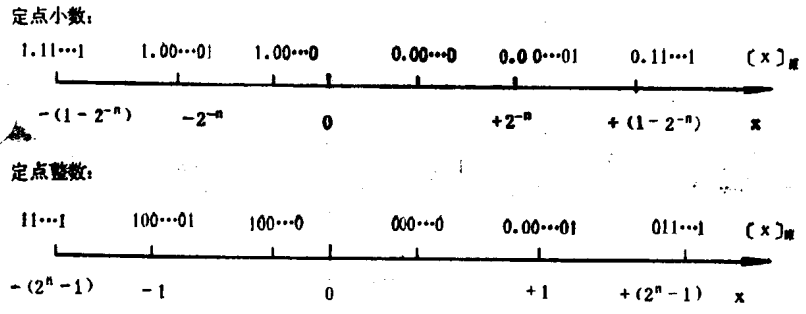

图1-3 真值与原码关系图

原码表示比较简单，运算结果直观。在原码乘除法中，数值部分相乘，符号按异或逻辑处理，还比较方便。但用原码进行加减运算时，需要比较操作数的符号和绝对值的大小，再确定加减运算，并要判定结果的符号，使用很不方便。

2.补码表示法

为了克服原码在加减运算中的缺点，又适应微型机本身结构的特点，通常采用数的补码表示法。

(1) 模和补码

现在以校正时钟为例来说明补码的概念。假设时钟停在8点，而正确的时间为6点，要拨准时钟可以使用以下二种方法：

倒拨2点，即8－2＝6（做减法）；

顺拨10点，即8＋10＝6＋12＝6（做加法，因钟面上12＝0）。

这里能用加法来代替减法，是因为钟面的容量有限，时针转一圈又回到原来位置，自然丢失了12，12是溢出量，又称为模（mod）。

在数学上可以用“同余”来描述这种关系，设A、B两个整数用同一个正数M去除而余数相等，则称A、B对M同余，记为A＝B(mod M)。在上例中以12为模，A＝－2，B＝10则

$$\frac{10}{12}=\frac{12-2}{12}=1+\frac{-2}{12}$$

故－2与＋10同余；或者说－2与＋10对模12互补，即－2的补码是＋10（以12为模）。

微型机中的运算器、寄存器、计数器都有一定的位数，也会产生溢出，所产生的溢出量就是模，在定点小数（x_0、$x_1x_2\cdots x_n$）中，溢出量为2，即以2为模；在定点整数（$x_0x_1x_2\cdots x_n$）中，溢出量为2^{n+1}，即以2^{n+1}为模。因此，在微型机中的数可以采用补码表示法，这样可以变减法为加法，把符号位也看成数一起参加运算。

(2) 补码的定义

由上式可知：　$[x]_{补}=M+x\ (\mathrm{mod}\ M)$

当$x\geqslant 0$时，模M丢失，所以正数的补码就是它本身；当$x<0$时，它的补码可以由该负数加上模数得到，即$[x]_{补}=M+x=M-|x|$。由此可以得出补码的定义：

设定点小数的补码形式为$x_0x_1x_2\cdots x_n$，则

$$[x]_{补}=\begin{cases}x & 0\leqslant x<1\\ 2+x=2-|x| & -0\leqslant x<0\end{cases}\quad(\mathrm{mod}\ 2)$$

设定点整数的补码形式为$x_0x_1x_2\cdots x_n$，则

$$[x]_{补}=\begin{cases}x & 0\leqslant x<2^n\\ 2^{n+1}+x=2^{n+1}-|x| & -2^n\leqslant x<0\end{cases}\quad(\mathrm{mod}\ 2^{n+1})$$

式中，x为真值，$[x]_{补}$为其补码。

例如：　$x=+1011$　　则$[x]_{补}=0,1101(\mathrm{mod}\ 2^5)$

　　$x=-1101$　　则$[x]_{补}=2^5+(-1101)=1,0011\ (\mathrm{mod}\ 2^5)$

(3) 由补码求真值的公式

设$[x]_{补}=x_0x_1x_2\cdots x_n$，则真值

$$x=-2^nx_0+\sum_{i=1}^{n}x_i\cdot 2^{n-i}$$

证：当$x_0=0$时，$[x]_{补}=x=2^{n+1}x_0+x$

　　当$x_0=1$时，$[x]_{补}=2^{n+1}x_0+x$

故 $$[x]_{补}=2^{n+1}x_s+x$$

$$x=[x]_{补}-2^{n+1}x_s=x_sx_1x_2\cdots x_n-2^{n+1}x_s$$

$$=2^nx_s+x_1x_2\cdots x_n-2\cdot2^nx_s=-2^nx_s+\sum_{i=1}^{n}x_i2^{n-i}$$

设$[x]_{补}=x_sx_1x_2\cdots x_n$，则真值

$$x=-x_s+\sum_{i=1}^{n}x_i\cdot2^{-i}$$，反之亦然

此公式请读者自己证明。

(4)补码与原码的转换关系

当x为正数时，$[x]_{补}=[x]_{原}=x$。

当x为负的定点整数时，则

$$[x]_{原}=1,\ x_1x_2\cdots x_n$$

$$[x]_{补}=2^{n+1}+x=2^{n+1}-x_1x_2\cdots x_n=2^n+(2^n-x_1x_2\cdots x_n)$$

$$=2^n+(\sum_{i=1}^{n}2^{n-i}-\sum_{i=1}^{n}x_i\cdot2^{n-i}+1)$$

$$=2^n+\sum_{i=1}^{n}(1-x_i)\cdot2^{n-i}+1=1,\quad \bar{x}_1\bar{x}_2\cdots\bar{x}_n+1$$

式中，$\bar{x}_i=1-x_i=\begin{cases}1 & x_i=0\\ 0 & x=1\end{cases}$

由上式可知，当 $x<0$时，把$[x]_{原}$的符号位保持不变，其他各位取反，末位加1，即得$[x]_{补}$。同样，把$[x]_{补}$的符号位不变，其他各位取反，末位加1，即得$[x]_{原}$。

例如：$x=-101100$ $[x]_{原}=1,101100$

$[x]_{补}=1,010011+1=1,010100$

仔细分析，可以找出转换的规律：自低位向高位，尾数的第一个“1”及其右部的“0”保持不变，左部的各位取反，符号位保持不变。

3.反码表示法

反码表示法与补码表示法相类似，正数的反码就是真值本身；负数的反码，是对除符号位以外的各位按位取反。它与补码的区别是末位少加一个“1”，这样就可以由补码的定义推出反码的定义。

设定点小数的反码形式为$x_s\cdot x_1x_2\cdots x_n$，则

$$[x]_{反}=\begin{cases}x & 0\leqslant x<1 \quad (\text{mod } \underbrace{1.11\cdots1}_{n+1位})\\ (2-2^{-n})+x & -1<x\leqslant0\end{cases}$$

设定点整数的反码形式为$x_0x_1x_2\cdots x_n$，则

$$[x]_{反}=\begin{cases}x & 0\leqslant x<2^n \quad (\text{mod } \underbrace{111\cdots\cdots}_{n+1\text{位}})\\ (2^{n+1}-1)+x & -2^n<x\leqslant 0\end{cases}$$

式中，x为真值，$[x]_{反}$为其反码。反码在定点小数、定点整数中分别以1.11…1和111…1为模，故有时也把反码称为“1”的补码。真值0的反码中有两种表示形式：

在定点小数中，$[+0]_{反}=0.00\cdots0$；$[-0]_{反}=1.11\cdots1$；

在定点整数中，$[+0]_{反}=000\cdots0$；$[-0]_{反}=111\cdots1$；

真值x与$[x]_{反}$的关系见图1-4所示。

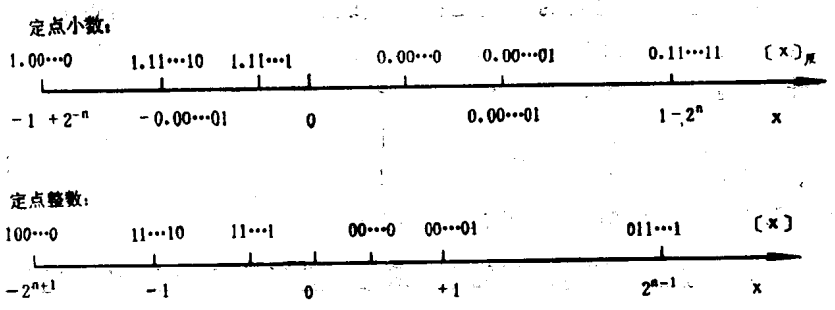

图1-4 真值与反码关系图

在反码运算中，如果最高位产生进位，不能象补码相加那样把进位丢掉，而要把进位加到最低位上，称为循环进位。这是因为反码是以$(2^{n+1}-1)$为模，而不是象补码以2^{n+1}为模。

(三) 二——十进制编码及运算规则

十进制是人们最熟悉、最习惯的计数系统，而在微型机中的数是二进制表示的。为了解决这一矛盾，可把十进制数的各位数字变成一组对应的二进制编码。一般是用四位二进制数来表示一位十进制数，称为二——十进制(BCD)编码。它既具有二进制数的形式，又保持十进制数的特点，可以作为人机联系时的一种中间表示，也有时用它直接进行运算。

由于使用4位二进制数可以进行16种不同的编码，而从中取出10个分别表示十进制数0～9，这样就可产生多种BCD编码，较常用的BCD码有8421码，2421码，余3码等等，见表1-1所示。

表1-1 常用的几种二——十进制编码

十进制数	8421码	2421码	余3码
0	0000	0000	0011
1	0001	0001	0100
2	0010	0010	0101
3	0011	0011	0110
4	0100	0100	0111
5	0101	1011	1000
6	0110	1100	1001
7	0111	1101	1010
8	1000	1110	1011
9	1001	1111	1100

8421码是一种有权码，从最高位起，权分别为8、4、2、1，它与十进制数的二进制表示相对应，故称为自然BCD码（即NBCD码），它的应用十分广泛。2421码也是一种有权码，从最高位起，权分别为2、4、2、1；它的另一特点是当任意两个十进制数相加等于9时，其对应的二进制编码相加也必然各位都为1，这种特点对减法很有用，它说明十进制中按9取补（即9的补码）对应于二进制中按1取补（即1的补码）余3码是根据8421码加上0011（即3）而形成的，它是一种无权码，它可以避免出现0000状态，也具有2421码的后一特点。

下面以8421码（NBCD）为例，介绍二——十进制编码的加减运算规则。

1．8421BCD码的加法运算

8421 BCD码与十进制数之间的转换是按位（或按组）直接进行转换的。

例如：

十进制权：	百位	十位	个位
十进制数：	5	3	8
	↑ ↓	↑ ↓	↑ ↓
8421 BCD码：	0101	0011	1000

进行8421 BCD码加法运算时，不能直接采用二进制运算规则，需要进行某些修正，其规则为：

（1）如果8421 BCD码相加结果等于或小于1001（即十进制数9），则无需修正。

（2）如果8421 BCD码相加结果等于或大于1010（即十进制数10），或按二进制相加产生进位，则需要加0110修正（即加6修正）。

加6修正原因是由于四位二进制数是逢十六进一，而十进制逢十进一，相差为6(即0110)的缘故。

2．8421 BCD码的减法运算

在BCD码运算中，为了变减法为加法，也需要采用补码运算。在十进制中是按9取补，它的模为（10^n-1）。例如3对9取补是9－3＝6。8421 BCD码按9取补的方法是每位取反，再加上1010即可。例如0011对9取补为

$$\begin{array}{rr} & 0011 \\ \text{每位相反} & 1100 \\ +) & 1010 \\ \hline \text{丢失}\ \boxed{1} & 0110 \end{array}$$

8421 BCD码减法运算规则：

（1）先把减数按9取补，再与被减数按8421BCD码加法运算规则相加。

（2）如果相加结果最高位有进位不能丢失，应加到最低位，称为循环进位（因为模为10^n-1）。有循环进位的结果为正；无循环进位的结果为负，其真正结果为它对9的补码。

（四）非数值数据的表示

不表示数值含义的数据称为非数值数据，它在机器内部也用二进制代码串表示，但是一种非结构化的二进制数。非数值数据包括逻辑数据、字符数据等。

1．逻辑数据表示

逻辑数据是由若干位二进制代码串组成，它无符号，位权等概念，也没有数值的大小，仅有真与假二种逻辑值。

逻辑数据只能参加逻辑运算，逻辑运算包括逻辑与、逻辑或、逻辑非等，以及它们的各种组合。参加逻辑运算的逻辑数据是按位进行的，每位之间彼此独立，没有算术运算的进位和借位问题。机器本身无法识别逻辑数据和数值数据，必须根据程序中的指令来识别它，逻辑运算指令指定的操作数是逻辑数据。

2.字符的编码表示

现代微型机不仅用于进行数值计算，还广泛用于信息处理，就必然要涉及到各种文字符号。而微型机只能识别和处理二进制代码，这就需要用二进制编码来表示各种字符。

目前，国际上广泛采用的一种字符编码系统是七单位的ASCII码（美国国家信息交换标准字符码）。它包括10个十进制数码、26个英文大小写字母、一些专用符号（如$、%、&、#等）和32个控制符号，共计128个字符。这就需要7位二进制编码，再加上一个偶校验位，刚好一个字节（8位），它对应2个BCD码。七单位ASCII码字符编码，见表1-2所示。

表1-2 七单位ASCII码字符编码表

位数 / 位数	W_7	0	0	0	0	1	1	1	1
	W_6	0	0	1	1	0	0	1	1
	W_5	0	1	0	1	0	1	0	1
W_4 W_3 W_2 W_1	行 / 列	0	1	2	3	4	5	6	7
0 0 0 0	0	空白(NUL)	转义(DLE)	SP	0	@	P	'or'	p
0 0 0 1	1	序始(SOH)	机控1(DC1)	!	1	A	Q	a	q
0 0 1 0	2	文始(STX)	机控2(DC2)	"	2	B	R	b	r
0 0 1 1	3	文终(ETX)	机控3(DC3)	#	3	C	S	c	s
0 1 0 0	4	送毕(EOT)	机控4(DC4)	$	4	D	T	d	t
0 1 0 1	5	询问(ENQ)	否认(NAK)	%	5	E	U	e	u
0 1 1 0	6	承认(ACK)	同步(SYN)	&	6	F	V	f	v
0 1 1 1	7	告警(BEL)	组终(ETB)	'Or'	7	G	W	g	w
1 0 0 0	8	退格(BS)	作废(CAN)	(	8	H	X	h	x
1 0 0 1	9	横表(HT)	载终(EM)	)	9	I	Y	i	y
1 0 1 0	10	换行(LF)	取代(SUB)	*	:	J	Z	j	z
1 0 1 1	11	纵表(VT)	扩展(ESC)	+	;	K	[	k	{
1 1 0 0	12	换页(FF)	卷隙(FS)	,	<	L	\	l	|
1 1 0 1	13	回车(CR)	群隙(GS)	-	=	M	]	m	}
1 1 1 0	14	移出(SO)	录隙(RS)	.	>	N	↑	n	~
1 1 1 1	15	移入(SI)	无隙(US)	/	?	O	←	o	DEL

此外，还有EBCDIC编码，它是扩充型二——十进制交换码，编码方法与ASCII码类似用8位表示信息，不设校验位，因而可以表示更多的符字信息。我国也制定了标准7单位码（GB1988-80），见表1-3所示。

表1-3 部颁标准7单位码（GB1988-80）

列 / 行	0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
0 0000	KB空白	ZY转移	间隙	0	@	P	，	p
1 0001	XS序始	JK1机控1	！	1	A	Q	a	q
2 0010	WS文始	JK2机控2	"	2	B	R	b	r
3 0011	WZ文终	JK3机控3	#	3	C	S	c	s
4 0100	SB送毕	JK4机控4	¥	4	D	T	d	t
5 0101	XW询问	FR否认	%	5	E	U	e	u
6 0110	CR承认	TB同步	&	6	F	V	f	v
7 0111	GJ告警	ZZ组终	、	7	G	W	g	w
8 1000	TG退格	ZF作废	(	8	H	X	h	x
9 1001	HB横表	ZTZ载终	)	9	I	Y	i	y
A 1010	HH换行	QD取代	★	：	J	Z	j	z
B 1011	ZB组表	KZ扩展	+	；	K	[	k	{
C 1100	HY换页	JX卷隙	，	<	L	\	l	|
D 1101	HC回车	QX群隙	-	=	M	]	m	}
E 1110	YC移出	LX录隙	'	>	N	^	n	~
F 1111	YR移入	YX无隙	/	？	O	—	o	抹掉

三、微型计算机硬件系统

微型计算机硬件系统通常由下面五个部分组成：

内存储器　包括随机存取存储器（RAM）和只读存储器（ROM）。

中央处理器（CPU）　如同微型计算机的心脏，它的性能决定了整个微型计算机的各项关键指标。

外存储器　当前用得最多的磁盘存储器，包括软盘存储器和硬盘存储器。

输入／输出接口电路　是用来使外部设备和微型计算机相连的。

总线　为CPU和其它部件之间提供数据、地址和控制信息的传输通道。

图1-5所示是微型计算机系统的结构。

（一）内存储器

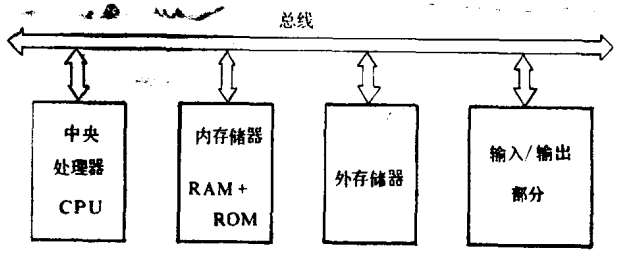

图1-5　一种微型机系统的结构

现代的内存储器基本上都是用大规模集成的存储器芯片堆砌而成的。它主要用来存放待加工的原始数据和结果，以及对其进行加工的方法和步骤(即微型机程序)。待加工的数据根据不同的用途可有不同的类型：如是数值计算方面的应用，则原始数据为一般的数；如是文字处理方面的应用，则原始数据为字符型的数据，若是图象处理方面的应用，则原始数据为图象类型的数据（如图素）等等。程序则依据用途、功能和数据类型的不同而不同。千变万化，正如没有两篇独立写的文章完全相同一样，对于一个问题的两个独立编写的程序也不会完全相同。

对内存储器的要求主要有三点：一是存取数据的速度要快，二是存储容量要大，三是成本要低。这三点要求也正是推动存储技术发展的动力。其中第三点要求是显而易见的，就不多说。对第一点要求，那是因为待处理的程序和数据都放在内存储器中，所以存取数据的速度应当和中央处理器的处理速度相匹配。如果存储器的速度跟不上，会严重影响整个系统的性能。微型计算机上用的存储器的存取周期一般在 200mμs 到 500mμs 范围之间（1mμs为10^{-9}s）。对第二点要求，那是因为使用微型计算机去解决的问题越来越复杂，也就是待加工的数据和程序的规模越来越大，相应的内存储器的容量应当越来越大（目前微机内存储器容量有1MB、2MB、4MB、10MB……等），保证容纳下待加工的数据和处理程序。

内存储器要保存成千上万的数据。因此如何把这些数据有规律地存放好，以便存（或写）取（或读）数据时方便、迅速，这是很重要的。其存放方法，见图1-6所示，是把内存储器分成一个个单元。每个单元内存放固定位数的二进制数据，通常称之为一个字(word)。每一个单元都有一个编号与之对应，称之为地址。只要指明地址，就可以到存储器的成千上万个单元中按所指地址存入或取出所需要的数据。这有点象是一幢学生宿舍楼，有许多房间，每间为一个单元，可住六个学生，每一个房间有一个房间号码("地址码")。要安排学生住宿或需找那房间的学生只要指定房间号码就可以了。

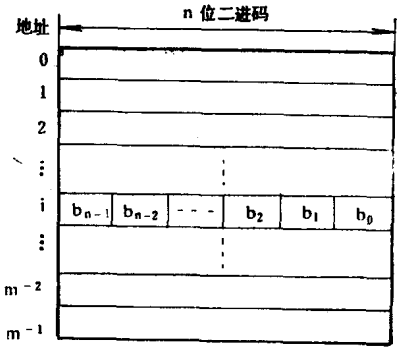

图 1-6　存储单元的组织及地址

内存储器的地址码是用二进制表示的。如果地证码共有二十位二进制位，则其地址码的可编码范围从0至$2^{20}-1$(即1024×1024)。由于二进制读起来不方便，所以我们在实际工作中，常用八进制。十六进制或十进制来表示地址。

图1-6所示，是一个有m个单元（字），每个单元为n位二进制位的存储器。地址从上到下顺序编号，由0单元至m-1单元，单元中的各位则从右到左顺序编号，由0位至n－1位。存