计算机原理

计算机原理

计 算 机 原 理

利 锐

教材编写组选编

目录

第一章　微型计算机的基础知识

计算机是一种能对数字进行处理和运算的电子设备。它只能识别和处理二进制代码，也就是说，所有需要计算机加以处理的数、字母、符号等都要用二进制代码来表示。本章前半部分将介绍二进制及其它与计算机密切相关的数制的特点，它们相互转换的方法；各种数、字符在计算机中的表示方法、编码方法等。这些都是计算机的运算基础。

微型计算机的电路尽管很复杂，但它们主要是由一些基本的逻辑电路组成的。在本章的后半部分，我们将介绍微机中常用的数字逻辑电路的原理和逻辑功能，最后介绍微机的重要部件——存储器的组成及原理。掌握这一部分电路知识，将有助于我们进一步学习微机的硬件系统。

第一节　数制的概念

一、进位计数制

按进位的原则进行计数的方法称为进位计数制。在日常生活中，我们用到各种进位计数制，如 60 秒为 1 分钟，60 分钟为 1 小时，这就是六十进位计数制；24 小时为 1 天，这是二十四进位计数制……不过人们最常用、最熟悉的还是十进位计数制，简称十进制。而计算机中使用的数制是二进制，此外还有八进制、十六进制等。本节主要介绍这些数制的特点及相

互转换的方法。

1. 十进制

在十进制中，每一位用 0～9 共 10 个数符来表示。当数大于 9 时，用“逢十进一”的原则向高位进位。用这 10 个数符和计数法可以表示任何大小的数。如 709.62 这个数，其小数点左边的第一位是个位，余者依次为十位、百位，而小数点右边的各位依次为 1/10 位、1/100 位。因而这个数可写为：

$$(709.62)_{10}=7\times10^{2}+0\times10^{1}+9\times10^{0}+6\times10^{-1}+2\times10^{-2}$$

其中 7、0、9、6、2 为各位的数符；10 是该计数制中数符状态的个数，称为基数；10^{2}、10^{1}、10^{0}、10^{-1}、10^{-2}即基数的 n 次幂，称为各位的权。各位数符和权值的乘积表示了该位数值的大小。

基数和权是进位计数制中的两个要素。

2. 二进制

在二进制中，只有 0、1 两个不同的数符，所以基数为 2。低位和相邻高位之间的进位关系是“逢二进一”。

任何一个二进制数，都可写成按权展开的形式，这里权值是 2 的 n 次幂。如

$$(1011.001)_{2}=1\times2^{3}+0\times2^{2}+1\times2^{1}+1\times2^{0}+0\times2^{-1}+0\times2^{-2}+1\times2^{-3}$$

3. 八进制

在八进制中，有 0～7 共 8 个不同的数符，所以基数为 8。低位和相邻高位之间的进位关系为“逢八进一”。

任何一个八进制数都可以写成按权展开的形式。如

$$(356.71)_{8}=3\times8^{2}+5\times8^{1}+6\times8^{0}+7\times8^{-1}+1\times8^{-2}$$

4. 十六进制

十六进制的基数为 16，有 16 个不同的数符，分别用 0～9，A(10)、B(11)、C(12)、D(13)、E(14)、F(15)来表示。低位

和相邻高位的进位关系是“逢十六进一”。

十六进制数同样也可以按权展开，例如

$$(A30.7D)_{16}=10\times16^{2}+3\times16^{1}+0\times16^{0}$$
$$+7\times16^{-1}+13\times16^{-2}$$

不同进制数可以像上面那样用在数后面加下标(10、2、8、16 等)的办法来区别。也可以在数后面加字母 D(十进制)、B(二进制)、Q(八进制)、H(十六进制)来表示，如 709.62D、1011.011B、356.71Q、A30.71DH 等。十进制数通常可以既不加下标，也不加字母。

常用的几种进位计数制表示数的方法见表 1-1。

表 1-1 常用计数制表示数的方法

十 进 制	二 进 制	八 进 制	十六进制
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10

二、计算机中采用二进制数

在计算机中，并不采用人们熟悉的十进制，而是采用二进制来表示数值和进行运算。这是因为二进制数只有两个不同的数符，容易用器件的物理状态（如电平的高低、晶体管的导通和截止等）来表示。同时，二进制数的运算比较简单，便于用电路来实现。例如二进制加法只有 4 种情况：

$$0+0=0 \qquad 1+0=1$$

$$0+1=1 \qquad 1+1=10$$

二进制乘法也只有 4 种情况：

$$0\times0=0 \qquad 1\times0=0$$

$$0\times1=0 \qquad 1\times1=1$$

因此，目前计算机中都采用二进制数。但是，用二进制表示的数往往位数很长，不便于书写和记忆。为了弥补这个缺陷，常使用八进制和十六进制。因为表示同一个数用八进制或十六进制要比用二进制位数少得多，使用很方便。并且我们在后面会看到这两种进制和二进制的转换非常容易，因此它们常作为二进制的缩写形式用于书写、输入和显示。

三、不同进制数的相互转换

1. 二进制数和十进制数的相互转换

(1) 二进制数转换为十进制数

将二进制数按权展开求和就可得到相应的十进制数。如

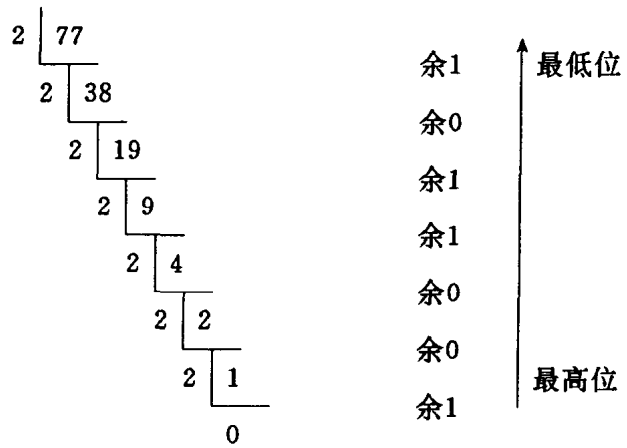
$(1101.1)_2=1\times2^3+1\times2^2+0\times2^1+1\times2^0+1\times2^{-1}=(13.5)_{10}$

(2) 十进制整数转换为二进制数

十进制整数转换为二进制整数，可用“除 2 取余”法。即用 2 去除十进制整数，得到一个商和余数，再用 2 去除商，又得

到新的商和余数，如此继续下去，直到商等于 0 为止。将最初得到的余数作为最低位(权值为 2^0)，最后得到的余数作为最高位，依次排列，就是所求的二进制数。

例 将十进制数 77 转换成二进制数。



所以 $(77)_{10}=(1001101)_2$

(3)十进制小数转换成二进制数

十进制小数转换成二进制小数，可用“乘 2 取整”法。即用 2 去乘十进制小数，然后去掉乘积中的整数部分，再用 2 去乘剩下的小数部分，如此继续下去，直到小数部分等于 0，或者达到所要求的精度为止。将最初得到的整数作为小数点后的最高位，最后得到的整数作为小数点后的最低位，依次排列就可得到十进制小数所对应的二进制小数。

例 将十进制数 0.6875 转换为二进制数

	0.6875		
×	2		
	1.3750	整数部分＝1	最高位
	0.375		
×	2		
	0.750	整数部分＝0	
	0.75		
×	2		
	1.50	整数部分＝1	
	0.5		
×	2		
	1.0	整数部分＝1	最低位

所以$(0.6875)_{10}=(0.1011)_2$

(4)十进制混合小数转换为二进制数

混合小数由整数和小数复合而成。按上述规则将整数和小数分别转换，然后将转换结果组合在一起即可。

例 将$(215.6531)_{10}$转换为二进制数。

其中 $(215)_{10}=(11010111)_2$

$(0.6531)_{10}\approx(0.101001)_2$

所以　$(215.6531)_{10}\approx(11010111.101001)_2$

2. 八进制数、十六进制数和十进制数的相互转换

八进制数、十六进制数与十进制数之间的转换的方法，和二进制数与十进制数之间的转换方法类似。

将八进制数、十六进制数按权展开求和，就可得到相应的十进制数。如

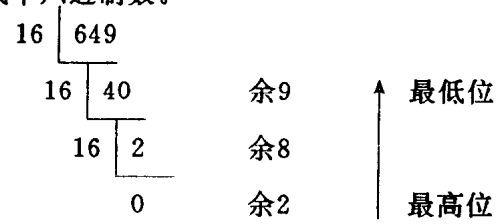
$$(372.1)_8=3\times8^2+7\times8^1+2\times8^0+1\times8^{-1}=(250.125)_{10}$$

$$(3AB.6)_{16}=3\times16^2+10\times16^1+11\times16^0+6\times16^{-1}$$
$$=(939.375)_{10}$$

将十进制数转换成为八进制数或十六进制数，也要把整数和小数分别进行转换。以转换为十六进制数为例，十进制整数转换为十六进制整数，采用“除 16 取余”法，即不断用 16 去除十进制整数，每次所得的余数就是十六进制的数符；最先得到的是十六进制数的最低位，最后得到的是最高位。十进制小数转换成十六进制小数，采用“乘 16 取整”法，即不断地用 16 去乘十进制小数，每次所得的整数部分，即为十六进制小数的数符；最先得到的是十六进制小数的最高位，最后得到的是最低位。

例　将$(649.75)_{10}$转换成十六进制数。

首先将$(649.75)_{10}$的整数部分和小数部分分开，分别转换成十六进制数。



得 $(649)_{10}=(289)_{16}$

$$
\begin{array}{rr}
 & 0.75 \\
\times & 16 \\
\hline
 & 450 \\
+ & 75 \\
\hline
 & 12.00
\end{array}
$$

整数部分＝12

得 $(0.75)_{10}=(0.C)_{16}$

所以 $(649.75)_{10}=(289.C)_{16}$

用类似的方法，可以实现任意进制数和十进制数之间的相互转换。

3. 二进制数和八进制数的相互转换

3 位二进制数一共有 8 种状态，它们的值正好分别和 1 位八进制数的各数符相对应：

二进制数	000	001	010	011	100	101	110	111
	↕	↕	↕	↕	↕	↕	↕	↕
八进制数	0	1	2	3	4	5	6	7

因此八进制数和二进制数相互转换十分方便。

(1)八进制数转换为二进制数

每位八进制数，用 3 位二进制数表示。

例 将$(360.734)_8$转换为二进制数。

3	6	0.	7	3	4
↓	↓	↓	↓	↓	↓
011	110	000.	111	011	100

即 $(360.734)_8=(011110000.111011100)_2$

$=(11110000.1110111)_2$

(2)二进制数转换成为八进制数

以小数点为界，向左向右每3位分为一组，如果最后整数部分不够3位在右边添0，小数部分不够3位在奋边添0，补足3位。然后将每一组二进制数用相应的八进制数表示即可。

例 将$(1101001.0100111)_2$转换成八进制数。

001	101	001.	010	011	100
↓	↓	↓	↓	↓	↓
1	5	1.	2	3	4

即 $(1101001.0100111)_2=(151.234)_8$

4. 二进制数和十六进制数的相互转换

4位二进制数一共有十六种状态，它们的值正好分别和1位十六进制数的各数符相对应。因此，相互转换十分方便。

(1)十六进制数转换为二进制数

每位十六进制数，用4位二进制数表示。

例 将$(3A0.7B8)_{16}$转换为二进制数。

3	A	0.	7	B	8
↓	↓	↓	↓	↓	↓
0011	1010	0000.	0111	1011	1000

即 $(3A0.7B8)_{16}=(001110100000.011110111000)_2$

$=(1110100000.011110111)_2$

(2)二进制数转换为十六进制数

以小数点为界，向左向右每4位分为一组，如果最后整数部分不够4位的在左边添0，小数部分不够4位的在右边添0，补足4位。然后将每一组二进制数用相应的十六进制数表示即可。

例 将$(10111110110.00111)_2$转换成十六进制数。

0101　1111　0110.　0011　1000

↓　　↓　　↓　　↓　　↓

5　　F　　6.　　3　　8

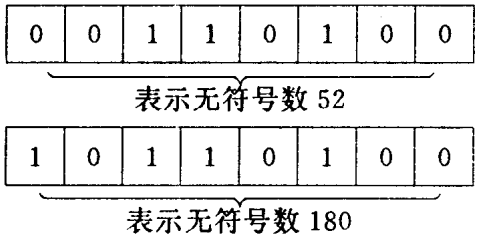
即　$(10111110110.00111)_2=(5F6.38)_{16}$

第二节　计算机中的数和编码

一、无符号数的表示方法

所谓无符号数，就是不考虑正、负号的数。

对于无符号数，机器的全部有效位都用来表示数的大小。在 8 位微机中，则 8 位全用来表示数值，如



二、带符号数的表示方法

平时，数的正、负是用符号“+”、“−”来区别的。由于计算机只能识别二进制符号，因此，在计算机中，带符号数的符号也只能用 0、1 来表示。通常规定数的最高位作为符号位，用 0 表示正号，用 1 表示负号。这种将符号数值化了，从而可以在计算机中使用的数，称为机器数。原来的带符号数称为相应机器数的真值。计算机中常用的机器数有 3 种——原码、反码和补码，下面分别加以介绍。

1. 原　码

在数值前加 1 位符号位，用 0 表示正号，1 表示负号，对原数值不产生任何影响，这种表示法称为原码表示法。

例如　$N_1=+52=+011\ 01\ 00B$

$N_2=-52=-011\ 01\ 00B$

则它们的原码为

$[N_1]_{原}=00\ 110\ 100$

$[N_2]_{原}=10\ 11\ 01\ 00$

0 的原码有两种：

$[+0]_{原}=00000000$

$[-0]_{原}=10000000$

若已知一个带符号数的原码，将其符号位还原成“+”号或“−”号，则可求得它的真值。

如　$[N_3]_{原}=11\ 10\ 11\ 01$

$[N_4]_{原}=011\ 0\ 11\ 01$

则它们的真值为

$N_3=-11\ 0\ 1101B$(或−109)

$N_4=+11\ 0\ 1101B$(或+109)

2. 反　码

正数的反码和原码相同。负数的反码等于其原码符号位不变，数值部分按位取反(0 变 1，1 变 0)。如

$[+52]_{反}=00\ 11\ 01\ 00$，　$[-52]_{反}=11\ 00\ 10\ 11$

$[+0]_{反}=00\ 00\ 00\ 00$，　$[-0]_{反}=11\ 11\ 11\ 11$

3. 补　码

正数的补码和原码相同。负数的补码为其反码加 1。

$[+52]_{补}=00\ 11\ 01\ 00$，　$[-52]_{补}=11\ 00\ 11\ 00$

$[+0]_{补}=00\ 00\ 00\ 00$，　$[-0]_{补}=1\ 00\ 00\ 00\ 00$

↓

丢失

上述求$[-0]_{补}$是用 11 11 11 11+1，最高位向前的进位超过了八位二进制系统的表示范围，自动丢失。

已知正数的补码，求它的真值是很方便的。

如　$[N_5]_{补}=00\ 10\ 11\ 10$

则　$[N_5]_{原}=[N^5]_{补}=00\ 10\ 11\ 10$

$N_5=+0\ 10\ 11\ 10B\ (+4b)$

将负数的补码再求一次补，则可得到该数的原码，从而求出真值。

如　$[N_6]_{补}=11110001$

则　$[N_6]_{原}=[[N_6]_{补}]_{补}=10001111$

$N_6=-0001111B\quad(-15)$

表 1－2 列出了用 8 位二进制代码表示无符号数、原码、补码、反码的相应关系。从表中可见，8 位二进制数码，用来表示无符号数为 0～255；表示原码为－127～＋127；表示补码为－128～＋127；表示反码为－127～＋127。

用原码表示带符号数具有简单、直观的优点，但是用原码进行减法运算很不方便。例如两个异号数相加（数值部分实际是相减），计算机要先比较两个数绝对值的大小，然后用大的绝对值减去小的绝对值，差值的符号与绝对值大的数的符号一致。这意味着计算机控制电路复杂化，运算速度降化。

表 1-2　数的表示方法小结

8位二进制数码表示	无符号数	原　码	补　码	反　码
00000000	0	+0	+0	+0
00000001	1	+1	+1	+1
00000010	2	+2	+2	+2
⋮	⋮	⋮	⋮	⋮
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
⋮	⋮	⋮	⋮	⋮
11111101	253	−125	−3	−2
11111110	254	−126	−2	−1
11111111	255	−127	−1	−0

引入补码的好处是可能将减法运算转换为加法运算，从而简化计算机的结构。而反码仅作为原码与补码相互转换过程的中间形式而已。

例 1　求十进制数 78−56=？

设 78−56=78+(−56)=x，将 78 和(−56)的补码相加，就可变减为加，得到用补码表示的结果 x。

$$\begin{array}{rl} & [+78]_{补}=01\ 00\ 11\ 10 \\ + & [-56]_{补}=11\ 00\ 10\ 00 \\ \hline & [x]_{补}=1\ 00010110 \end{array}$$

↓

丢失　　x=+22

例 2　求十进制数 23−78=？

设 23−78=23+(−78)=y

$$
\begin{array}{rl}
 & [+23]_{补}=00\ 01\ 01\ 11 \\
+ & [-78]_{补}=10\ 11\ 00\ 10 \\
\hline
 & [y]_{补}=11\ 00\ 10\ 01
\end{array}
\qquad y=-55
$$

在微型计算机中，带符号数都是用补码表示，得到的是补码表示的结果。但如果运算结果超过了计算机的表示范围（从表 1－2 中可知，在字长为 8 位时，如果超过了－128～＋127），则结果发生错误，这个现象叫“溢出”。

三、计算机中常用的编码

由于计算机只能识别二进制符号，所以在计算机中，所有的数、字母、符号等都要用特定的二进制编码来表示。

1. 二进制编码的十进制数

虽然二进制数实现容易、可靠，运算规律也十分简单，但二进制不直观，人们希望计算机的输出、输入仍使用十进制数。这就需要计算机先将输入的十进制数转换为二进制数。进行处理后再转换成十进制数输出。这种转换可由计算机实现，但是要花费较多的时间。为了简化电路，节省时间，同时又比较直观，人们采用一种特殊形式的编码——BCD 码来进行输入或输出。

BCD 码是用 4 位二进制编码表示 1 位十进制数。它的种类很多，最常用的是 8421 BCD 码。8421 BCD 码和通常的二进制数一样，从左到右各位的权值为 8、4、2、1，不同的是 4 位二进制数有 0000～1111 共 16 种状态，而 8421 BCD 码只取 0000～1001 共 10 种状态，详见表 1－3。

8421 BCD 码有 10 个不同的数字符号，它是“逢十进一”的，所以它是十进制数；但是它的每一位又是用 4 位二进制编码来表示的，因此称为二进制编码的十进制数。