第 2 章

数制转换与运算

本章介绍的“数制”（数据制式）及其相关知识，之所以要把它放在本书正式介绍网络技术之前，是因为它是我们学习网络技术，甚至今后要从事程序开发工作的基础和必备知识。“数制”其实是数据结构中内容之一，看似与网络关系不大，但是它却实实在在地影响了我们日常的网络管理工作。

在我们日常的网络管理中，与数制关系最密切的要数 IP 地址的表示形式了。我们知道 IP 地址其实都是二进制的（包括最新的 IPv6 地址），所以才有说 IPv4 地址是 32 位的，而 IPv6 地址是 128 位的。但是在我们进行的日常网络管理中，IPv4 地址是以十进制表示的，而 IPv6 则是以十六进制表示的。这就涉及到二进制与十进制和十六进制之间的相互转换问题。

与数制另一个关系密切的是注册表。我们在编辑注册表时，经常要选择数制格式，有二进制的、有十进制的，有十六进制的，还有八进制的。如果不了解这些数制之间的相互转换，也就不可能清楚如何在注册表中使用这些不同数制的数据了。

另外与数制关系密切的一个方面就是网络管理工具（如 Sniffer）中的抓包分析了。在这些网络管理所抓到的包中，基本上都是以十六进制格式显示的，如果不懂十六进制数据格式，也就无从谈起对数据包内容的深入分析了。

所以，作为网络管理工作者，了解一些数制基础知识还是很有必要的。如果将来想要从事程序开发工作，那更是必不可少的了。也正因如此，现在网络类考试中，数制虽然所占分数不多，但总还是会有的。

教学（自学）课时安排

课时安排	本章老师共需安排 3 个授课课时：其中安排 2 个课时教授本章内容，1 个课时讲解训练题（学员课前预习 4 个小时，课后安排 2 个小时做训练题）	
授课课时	主要内容	重点
1	①主要数制分类 ②二进制、八进制、十进制和十六进制的相互转换 ③二进制的算术四则运算	①各种进制的标注 ②二进制与十进制的相互转换 ③二进制与十六进制的相互转换 ④十进制与十六进制的相互转换 ⑤二进制整数四则运算与小数四则运算的区别
2	①二进制的逻辑运算 ②二进制的 BCD 编码 ③二进制的原码、反码和补码表示形式及相互转换 ④补码的加减法运算	①二进制与、或、异或运算 ②二进制原码、反码和补码相互转换规则 ③二进制补码加减法运算规则
3	讲解实战训练中的主要练习题	

2.1 数制的分类

“数制”的全称就是“数据制式”，是指数据的进位计数规则，所以又称为“进位计数制”，简称“进制”。在日常生活中经常要用到数制，我们日常所使用的数都是十进制的，如我们拿的 3000 元工资，1.5 元/斤的菜价等。

除了十进制计数以外，还有许多非十进制的计数方法。在计算机中常见的还有二进制、八进制、十六进制等制式。其实数据制式远不止这么几种，如我们常以 60 分钟为 1 小时，60 秒为 1 分钟，用的就是 60 进制计数法；一天之中有 24 小时，用的是 24 进制计数法；而一星期有 7 天，用的是 7 进制计数法；一年中有 12 个月，用的是 12 进制计数法等。

虽然数据制式可以有很多种，但在计算机通信中通常遇到的仍是以上提到的二进制、八进制、十进制和十六进制。在一种数制中所能使用的数码的个数称为该数制的“基数”，如二进制的基数为“2”，八进制的基数为“8”，十进制的基数为“10”，十六进制的基数也就是“16”。

既然有不同的数制，那么在计算机程序中给出一个数时就必须指明它属于哪一种数制，否则计算机程序就不知道该把它看成是哪种数了。不同数制中的数可以用下标或后缀来标识。下面分别予以介绍。

- 十进制（Decimal）

十进制是我们日常生活中常用的数制类型，基数是 10，也就是它有 10 个数字符号，即 0、l、2、3、4、5、6、7、8、9。其中最大数码是“基数”减 1，即 10-1=9，最小数码是 0。

十进制数的标志为：D，如（1250）D，表示这个数是十进制数，也可用下标“10”来表示，如（1250）$_{10}$（注意是下标）。

- 二进制（Binary）

二进制是计算机运算所采用的数制，基数是 2，也就是说它只有两个数字符号，即 0 和 1（=2-1）。如果在给定的数中，除 0 和 1 外还有其他数，例如 1061，那它就绝不会是一个二进制数了。

二进制数的标志为：B，如（1001010）B，表示这个数是二进制数，也可用下标“2”来表示，如（1001010）$_2$（注意是下标）。

- 八进制（Octal）

八进制的基数是 8，也就是说它有 8 个数字符号，即 0、l、2、3、4、5、6、7（= 8-1）。对比十进制可以看出，它比十进制少了两个数“8”和“9”，这样当一个数中出现“8”和（或）“9”时，如 23459，那它也就绝不是八进制数了。八进制数的最大数码也是基数减 1，即 8-1=7，最小数码也是 0。

八进制数的标志为：O 或 Q（注意它特别一些，可以有两种标志），如（4603）O（注意是字母 O，不是数字 0）、（4603）Q，表示这个数是八进制数，也可用下标“8”来表示，如（4603）$_8$（注意是下标）。

- 十六进制（Hexadecimal）

十六进制数用得比较多，最新的 IPv6 地址通常也是采用十六进制来表示的（IPv4 地址通常采用的是十进制表示），在注册表中也有用到，所以了解十六进制还是非常重要的。

十六进制的基数是 16，也就是说它有 16 个数字符号，除了十进制中的 10 个数可用外，还使用了 6 个英文字母，这 16 个数字和字母依次是 0、l、2、3、4、5、6、7、8、9、

A、B、C、D、E、F。其中 A 至 F 分别代表十进制数的 10 至 15。如果数据中出现了字母之类符号，如 63AB，则一定不会是八进制，或十进制，而是十六进制了。它的最大的数码也是“基数”减 1，即 16-1=15（为 F），最小数码也是 0。

十六进制数的标志为：H，如（4603）H，表示这个数是十六进制数，也可用下标“16”来表示，如（4603）$_{16}$（注意是下标）。十六进制数也常常用前缀 0x 或者 0X 来表示（注意是数字 0，而不是字母 O），如 0x250 表示的是十六进制的 250，而不是八进制，或者十进制的 250。

表 2-1 是以上介绍的几种数制对应关系。注意，八进制没有 8 和 9，二进制 1000 对应的八进制是 10，而不是想象中的 8，二进制 1001 对应的八进制是 11，而不是想象中的 9。

表 2-1　不同数制的对应关系

二进制数	十进制数	八进制数	十六进制数
0	0	0	0
1	1	1	1
10	2	2	2
11	3	3	3
100	4	4	4
101	5	5	5
110	6	6	6
111	7	7	7
1000	8	10	8
1001	9	11	9
1010	10	12	A
1011	11	13	B
1100	12	14	C
1101	13	15	D
1110	14	16	E
1111	15	17	F

2.2　数制转换

同一个数在一些环境中，可能要用不同数制形式来表示，这就涉及到数制间的转换问题了。下面是常见的十进制、二进制、八进制、十六进制之间的转换方法。

2.2.1　非十进制数转换成十进制数

非十进制数（常见的是指二进制、八进制和十六进制）转换成十进制数的方法是将非十进制数按位权（位权是指对应数值位，对应制式的幂次方）展开求和。

1．二进制转换为十进制

说到二进制到十进制的转换，大家可能早就有所了解了，如在 IPv4 地址计算时就经常进行这样的操作。转换的方法比较简单，只需按它的位权展开即可。展开的方式是把二进制数首先写成加权系数展开式，然后按十进制加法规则求和。这种做法称为“按权相加”法。

二进制整数部分的一般表现形式为：$b_{n-1}\cdots b_1b_0$（共 n 位），按位权相加展开后的展开格式为（注意，展开式中各项的幂次是从高到低下降的）：

$$b_{n-1}\times2^{n-1}+b_{n-2}\times2^{n-2}\cdots+b_1\times2^1+b_0\times2^0$$

如二进制数（11010）$_2$的按位权相加展开格式为：$1\times2^4+1\times2^3+0\times2^2+1\times2^1+0\times2^0=16+8+0+2+0=$（26）$_{10}$。

二进制小数部分的幂次则是反序排列的（也就是与整数部分的幂次序列相反，其绝对值是从低到高上升的），且为负值，最低幂次（也就是最靠近小数点的第一个小数位的幂次）为“-1”。如二进制小数部分的格式为：$0.b_{n-1}\cdots b_1b_0$，则按位权相加后的展开格式为：

$$b_{n-1}\times2^{-1}+b_{n-1}\times2^{-2}\cdots+b_1\times2^{-(n-1)}+b_0\times2^{-n}$$

如（0.1011）$_2$ 的按权相加展开格式为：$1\times2^{-1}+0\times2^{-2}+1\times2^{-3}+1\times2^{-4}=0.5+0+0.125+0.0625=$（0.6875）$_{10}$。

2．八进制转换为十进制

八进制转换成十进制也是采取“按权相加”法，只是这里的位权值是 8 的相应幂次方。如八进制整数部分的格式为：$b_{n-1}\cdots b_1b_0$，则按位权相加展开后的格式就为（幂次是从高到低下降的）：

$$b_{n-1}\times8^{n-1}+b_{n-2}\times8^{n-2}\cdots+b_1\times8^1+b_0\times8^0$$

如八进制数（26356）$_8$的按位权相加展开格式为：$2\times8^4+6\times8^3+3\times8^2+5\times8^1+6\times8^0=4096+0+192+32+5=$（4425）$_{10}$。

八进制小数部分的幂次是反序排列的（也就是与整数部分的幂次序列相反，其绝对值是从低到高上升的），且为负值，最低幂次（也就是最靠近小数点的第一个小数位的幂次）为“-1”。如八进制小数部分的格式为：$0.b_{n-1}\cdots b_1b_0$，则按位权相加后的展开格式为：

$$b_{n-1}\times8^{-1}+b_{n-2}\times8^{-2}\cdots+b_1\times8^{-(n-1)}+b_0\times8^{-n}$$

如（0.257）$_8$ 按位权相加展开格式为：$2\times8^{-1}+5\times8^{-2}+7\times8^{-3}=0.25+0.078125+0.013671875=$（0.341796875）$_{10}$。

3．十六进制转换为十进制

十六进制转换成十进制的方法也是采取“按权相加”法，只是这里的位权值是 16 的相应幂次方。如十六进制整数部分的格式为：$b_{n-1}b_{n-2}\cdots b_1b_0$，则按位权相加展开后的格式就为（幂次是从高到低下降的）：

$$b_{n-1}\ 16^{n-1}+b_{n-2}\times16^{n-2}\cdots+b_1\times16^1+b_0\times16^0$$

如十六进制数（26345）$_{16}$的按位权相加展开格式为：$2\times16^4+6\times16^3+3\times16^2+4\times16^1+5\times16^0=131072+24576+768+64+5=$（156485）$_{10}$。

十六进制小数部分的幂次也是反序排列的（也就是与整数部分的幂次序列相反，其绝对值是从低到高上升的），且为负值，最低幂次（也就是最靠近小数点的第一个小数位的幂次）为“-1”。如十六进制小数部分的格式为：$0.b_{n-1}\cdots b_1b_0$，则按位权相加后的展开格式为：

$$b_{n-1}\times16^{-1}+b_{n-2}\times16^{-2}\cdots+b_1\times16^{-(n-1)}+b_0\times16^{-n}$$

如（0.25A）$_{16}$ 按位权相加展开格式为：$2\times16^{-1}+5\times16^{-2}+10\times16^{-3}=0.125+0.0234375+0.00244140625=$（0.15087890625）$_{10}$。

2.2.2　十进制数转换成非十进制数

十进制数转换成非十进制数的方法是：整数之间的转换用“除基取余法”，也就是用基

数相除，然后反序（由后向前取）取余数；小数之间的转换用“乘基取整法”，也就是用基数相乘，然后正序（由前向后取）取得到的整数。

1．十进制转换为二进制

要将十进制整数转换为二进制整数可以采用“除 2 取余”法。也就是先将十进制数除以 2，得到一个商数和余数，再将商数除以 2，又得到一个商数和余数，以此类推，直到商数小于 2 为止。也就是把十进制不断除 2，直到结果的值小于 2 为止。

如图 2-1 左、右图分别为十进制数 48 和 250 通过除 2 的方法得到的二进制数值。然后从最后的小于 2 的商数开始向上与所得的余数连起来即为对应二进制数的各位数字，也就是说把先得到的余数作为最后二进制数的低位，后得到的余数作为最后二进制数的高位，依次排列起来（俗称“逆序排列”）。如图 2-1 左图所示的十进制数 48 转换成二进制后就为（110000）$_2$，右图所示的十进制数 250 转换成二进制后就为（11111010）$_2$。

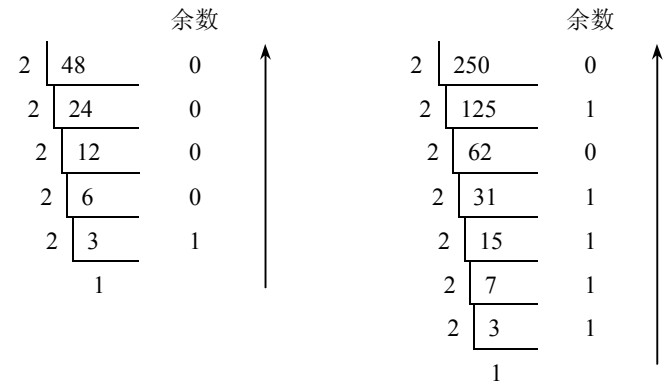

图 2-1　两个十进制整数转换成二进制整数的步骤

要将十进制纯小数转换为二进制纯小数可以采用“乘 2 取整”法。具体做法是：用 2 乘十进制小数，可以得到积，将积的整数部分取出，再用 2 乘余下的小数部分，又得到一个积，再将积的整数部分取出，如此进行，直到积中的小数部分为零，或者达到所要求的精度为止。然后把取出的整数部分按正序排列起来，即先取的整数作为二进制小数的高位，后取的整数作为低位。

如图 2-2 左、右图分别是十进制小数 0.125 和 0.625 转换成二进制的过程，最后得到的二进制值就是从最开始得到的整数值开始，一直到最后得到的整数值（也就是自上而下的顺序，与整数转换中取余的顺序相反）。0.125 和 0.625 最后的二进制值分别为（0.001）$_2$和（0.101）$_2$（注意一定要记得在整数部分加上“0.”，因为转换成二进制后仍是小数）。

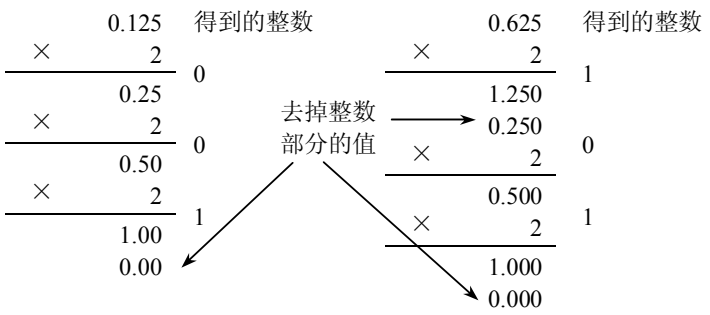

图 2-2　两个十进制小数转换成二进制小数的示例

【注意】有些十进制小数乘以 2 后是个无穷循环数，永远不会有完整的整数，此时就要看所需的精度如何了，按所需位数精度取值即可。如 0.825 就是这样一个数，如果仅要求小数点后 3 位，则相应的二进制数为（0.110）$_2$，如果要求为 4 位，则对应的二进制数为（0.1101）$_2$。

2．十进制转换成八进制

八进制数中的基数为 8，因此在八进制数中的数码有 0、1、2、3、4、5、6、7 这八个。与二进制数转换成十进制方法类似，将十进制整数转换为八进制整数也可以采用“除 8 取余”的方法，直到所得的商小于 8，然后把余数按反序排列即可；十进制小数转换为八进制小数是采用“乘 8 取整”法，直到所得到的积小数部分为 0，或者在规定的精度范围内，然后把所得到的整数正序排列起来即可。

如整数（65）$_{10}$ 和（2467）$_{10}$ 按“除 8 取余”的方法转换成八进制的步骤分别如图 2-3 左、右图所示。（65）$_{10}$ 和（2467）$_{10}$ 转换成八进制后的结果分别是（101）$_8$ 和（463）$_8$（为逆序排列）。

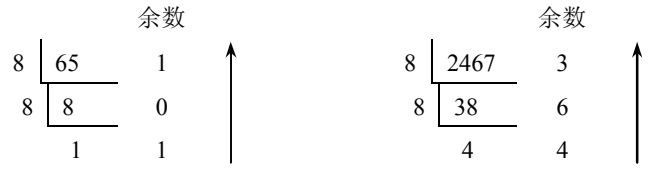

图 2-3　两个十进制整数转换成八进制整数的步骤

下面是两个十进制小数通过“乘 8 取整”的方法转换成八进制的示例。如要把（0.125）$_{10}$ 和（0.8125）$_{10}$ 转换成八进制，其步骤分别如图 2-4 左、右图所示。（0.125）$_{10}$ 和（0.8125）$_{10}$ 转换成八进制后的结果分别是（0.1）$_8$ 和（0.64）$_8$（是正序排列，也一定要记得在整数部分加上“0.”，仍是小数）。

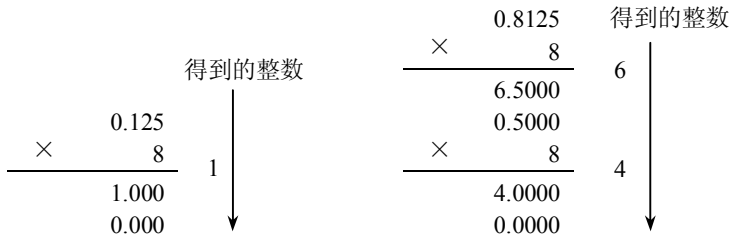

图 2-4　两个十进制小数转换成八进制小数的示例

3．十进制转换成十六进制

十六进制数的基数为 16，因此，在十六进制数中的数码有 0、1、2、3、4、5、6、7、8、9、A、B、C、D、E、F 共十六个。

也与二进制数类似，将十进制整数转换为十六进制整数采用“除 16 取余”法，直到所得的商小于 16，然后把余数按反序排列即可；十进制小数转换为十六进制小数采用“乘 16 取整”法，直到所得到的积小数部分为 0，或者在规定的精度范围内，然后把得到的整数正序排列起来即可。

如整数（45）$_{10}$ 和（3456）$_{10}$ 按“除 16 取余”的方法转换成十六进制的步骤分别如图 2-5 左、右图所示。（45）$_{10}$ 和（3456）$_{10}$ 转换成十六进制的结果分别是（2D）$_{16}$ 和（D80）$_{16}$（注意，其中的 13 用十六进制的 D 表示了）。

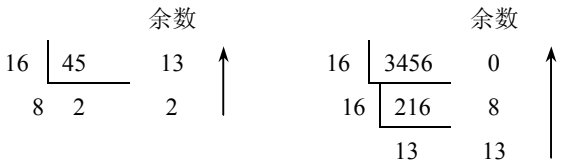

图 2-5　两个十进制整数转换成十六进制整数的示例

下面是两个十进制小数通过“乘 16 取整”的方法转换成十六进制的示例。如要把（0.125）转换成十六进制，其步骤如图 2-6 所示，结果是（0.2）$_{16}$（是正序排列，也一定要记得在整数部分加上“0.”，仍是小数）。

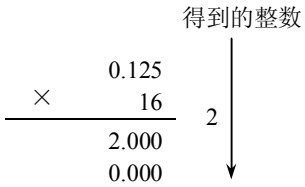

图 2-6　十进制小数转换成十六进制小数的示例

2.2.3　非十进制数之间的相互转换

从表 2-1 可以得出这样一个规律：1 位八进制数对应 3 位二进制数，而 1 位十六进制数对应 4 位二进制数。因此，二进制数与八进制数之间、二进制数与十六进制数之间的相互转换十分容易。

八进制数转换成二进制数的方法是：将每 1 位八进制数直接写成相应的 3 位二进制数。

二进制数转换成八进制数的方法是：以小数点为界，向左或向右将每 3 位二进制数分成一组，若不足 3 位，则用 0 补足 3 位。然后，将每一组二进制数直接写成相应的 1 位八进制数。

十六进制数转换成二进制数的方法是：将每一位十六进制数直接写成相应的 4 位二进制数。

二进制数转换成十六进制数的方法是：以小数点为界，向左或向右将每 4 位二进制数分成一组，若不足 4 位，则用 0 补足 4 位。然后，将每一组二进制数直接写成相应的 1 位十六进制数。

以上二进制、八进制、十六进制的对应关系参见表 2-1 即可，直接进行对应转换即可。

1．二进制转换成八进制

通过前面的学习，我们知道，最多只需要 3 位二进制数即可表示 1 位八进制数，由此可以得出，二进制数转换成八进制数的方法是：以小数点为界，向左或向右将每 3 位二进制数分成一组，在最后一组中若不足 3 位，则用 0 补足。然后，将每 3 位一组的二进制数直接写成相应的 1 位八进制数即可。它们之间的对照关系可参见表 2-1。

例如要将（1101011.10111）$_2$转化为八进制数，可以把整数部分从右向左每 3 位分为一组，最后不足 3 位时加 0 补上，然后把小数部分从左向右（与整数部分的划分顺序相反）同样以每 3 位分为一组，最后不足 3 位时加 0 补上。如图 2-7 所示。最后的结果为（153.56）$_8$。

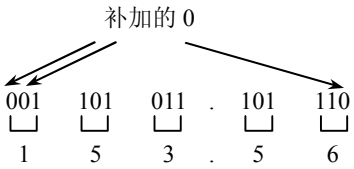

图 2-7　二进制转换成八进制示例

2．八进制转换成二进制

知道了二进制数转换成八进制数的方法，现要将八进制数转换成二进制，则非常简单了。只需把每 1 位八进制数直接写成相应的 3 位二进制数即可（前导 0 可以省略）。

例如，将（3456.2262）$_8$ 转换为二进制数的方法是依次把八进制的每 1 位用 3 位二进制数转换，如图 2-8 所示。最后的结果为（11100101110.010010101010）$_2$。

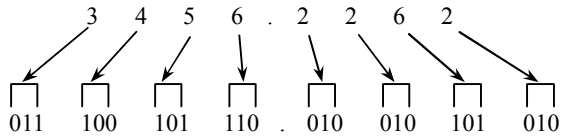

图 2-8　八进制转换成二进制示例

3．二进制转换成十六进制

二进制数转换成十六进制数的方法与二进制转换成八进制类似，不同的只是 1 位十六进制数要用 4 位二进制数来表示（而不是转换成八进制时的 3 位）。方法是：以小数点为界，向左或向右将每 4 位二进制数分成一组，若不足 4 位，则用 0 补足 4 位。然后，将每一组二进制数直接写成相应的 1 位十六进制数。

例如将二进制数（11 0110 1110.000 1 0101 0101）$_2$ 转换为十六进制数的最后结果为（36E.155）$_{16}$。

将十六进制数转换成二进制数的方法是上述二进制数转换成十六进制数的逆过程。方法是：将每 1 位十六进制数直接写成相应的 4 位二进制数。例如将（4AF.51）$_{16}$ 转换成二进制数的最后结果为：（100 1010 1111.0101 0001）$_2$。

4．八进制转换成十六进制

八进制与十六进制的相互转换最好的方法就是以二进制为桥梁，先把其中一个转换成二进制，然后再把得到的二进制转换成另一个进制的数。如（6237.431）$_8$ 转换成十六进制的步骤是分为以下两步完成的：

（1）将（6237.431）$_8$ 转换成二进制，得到（110010011111.100011001）$_2$。

（2）再将（110010011111.100011001）$_2$ 转换成十六进制，最终的结果就是（C9F.99）$_{16}$。

将十六进制转换成八进制的步骤类似，在此以（3AB.11）$_{16}$ 为例进行介绍。具体的转换过程也是分两步：

（1）先将十六进制数（3AB.11）16 转换成二进制，得到（001110101011.00010001）$_2$。

（2）再将（001110101011.00010001）$_2$ 转换成八进制，最后的结果为（1653.042）$_8$。

2.3 二进制运算

二进制数在计算机中是应用最广的，因为它最简单，数码仅两个：1 和 0，也可以代表电平的高和低，或者电压的正和负，或者电路的开与关等。

对于二进制数，除了与十进制数一样可以进行四则算术运算外，还可以进行逻辑运算，因为它只有两个数码，可以代表两种截然相反的状态。

2.3.1 二进制的四则算术运算

二进制数与十进制数一样，同样可以进行加、减、乘、除四则算术运算。其算法规则如下：

- 加运算：0+0=0，0+1=1，1+0=1，1+1=10 # 逢 2 进 1
- 减运算：1-1=0，1-0=1，0-0=0，0-1=1 # 向高位借 1 当 2
- 乘运算：0*0=0，0*1=0，1*0=0，1*1=1 # 只有同时为“1”时结果才为“1”
- 除运算：二进制只有两个数（0，1），因此它的商是 1 或 0

1．加、减法运算示例

在进行二进制加减法运算时，最关键的一点就是逢 2 进 1，进 1 当 1，而借 1 当 2。

例如：求（10010）$_2$+（11010）$_2$ 之和；求（111010）$_2$-（101011）$_2$ 之差，这两个计算过程分别如图 2-9 的左、右图所示。结果分别为 101100 和 1111。

	被加数	10010		被减数	111010
	加数	11010		减数	101011
+	进位	1 1	−	借位	1111
		101100			001111

图 2-9 二进制加、减法计算示例

下面对以上两个示例的加、减法运算步骤详细阐述一下。

图 2-9 左图所示的加法运算步骤如下：

（1）首先是最右位相加。这里加数和被加数的最后一位都为“0”，根据加法原则可以知道，相加后为“0”。

（2）再进行倒数第二位相加。这里加数和被加数的倒数第二位都为“1”，根据加法原则可以知道，相加后为“（10）$_2$”，此时把后面的“0”留下，而把第一位的“1”向高一位进“1”。

（3）再进行倒数第三位相加。这里加数和被加数的倒数第三位都为“0”，根据加法原则可以知道，本来结果应为“0”，但倒数第二位已向这位进“1”了，相当于要加“被加数”、“加数”和“进位”这三个数，所以结果应为 0+0+1=1。

（4）再进行第四位的相加。这里加数和被加数的倒数第四位分别为“1”和“0”，根据加法原则可以知道，相加后为“1”。

（5）最后是最高位相加。这里加数和被加数的最高位都为“1”，根据加法原则可以知道，相加后为“（10）$_2$”。但一位只能有一个数字，所以需要再向前进“1”，本身位留下

“0”，这样该位相加后就得到“0”，同时产生新的最高位，值为“1”（如果超出了字长的限制，则新产生的最高位将溢出）。

$(10010)_2$+$(11010)_2$的最后运算结果为101100。

图2-8右图所示的减法运算步骤如下：

（1）首先是最后一位相减。被减数的最后为“0”，而减数的最后为“1”，所以不能直接相减，需要向倒数第二位借“1”，这样相当于得到了十进制中的“2”，用2减1结果得1。

（2）再计算倒数第二位的减法运算。本来被减数和减数的该位都为“1”，但是被减数的该位被最后一位在上一步中借走了，所以为“0”了。这时也就不能直接与减数的倒数第二位相减了，需要再向倒数第三位借位。同样是借 1 当 2。这样两数的倒数第二位相减后的结果仍是2-1=1。

（3）用上一步同样的方法计算倒数第三位和倒数第四位的减法运算。结果都为1。

（4）再计算被减数和减数的倒数第五位减法运算。在被减数上的该位原来为“1”，可是已被倒数第四位借走了，所以成了“0”，而减数的倒数第五位也为“0”，可以直接相减，得到的结果为“0”。

（5）最后是最高位的相减了，被减数和减数的该位都是“1”，可以直接相减，得到的结果为“0”。

这样一来，$(111010)_2$-$(101011)_2$的结果是$(001111)_2$，前面的两个“0”可以不写，所以最后结果就是$(1111)_2$。

【经验之谈】在二进制的加、减法运算中一定要联系上十进制的加、减法运算方法，其实它们的道理是一样的，也是一一对应的。在十进制的加法中，进“1”仍旧当“1”，在二进制中也是进“1”当“1”。在十进制减法中我们向高位借“1”当“10”，在二进制中就是借“1”当“2”。而被借的数仍然只是减少了“1”，这与十进制一样。

2．乘、除法运算示例

二进制的乘、除法运算规则在本节前面已有介绍，那就是“0”乘“1”或者“0”结果都为“0”，只有“1”与“1”相乘等于“1”。乘法运算被乘数也是需要由乘数一位位地去乘（同样需要对齐积的数位，与十进制的乘法一样）。除法运算时，当被除数大于除数时，商是“1”；当被除数小于除数时不够除，商只能是“0”，这与十进制的除法也类似。

例如求$(1010)_2$×$(101)_2$和$(11001)_2$÷$(101)_2$的结果，计算过程分别如图2-10的左、右图所示。

```
    被乘数      1010                 商    101
×    乘数        101              101 √ 11001     被除数
  ----------------          除数   -    101
                1010              ------------
               0000                      101
 +            1010                 -     101
  ----------------                ------------
      积     110010                      000
```

图2-10　二进制乘、除法运算示例

图2-10左图所示的乘法运算步骤如下：

（1）首先是乘数的最低位与被乘数的所有位相乘，因为乘数的最低位为“1”，根据乘法原则可以得出，结果其实就是被乘数本身，直接复制下来即可。

（2）接着进行的是乘数的倒数第二位与被乘数的所有位相乘，因为乘数的这一位为“0”，根据乘法原则可以得出，它与被乘数的所有位相乘的结果都为“0”，也就是结果是

与被乘数一样位数的“0”。

（3）然后是乘数的最高位与被乘数的所有位相乘，同样乘数的这一位为“1”，结果就是被乘数本身。

（4）最后再按照前面介绍的二进制加法原则对以上三步所得的结果按位相加（与十进制的乘法运算方法一样），结果得到（1010）$_2$×（101）$_2$=（110010）$_2$。

图 2-9 右图所示的除法运算步骤如下：

（1）因为除数为“100”，有 3 位，所以在被除数中也至少要有 3 位（从高位数起）。被除数的高 3 位为“110”，恰好比除数“101”大，可以直接相除，商只能是 1，然后把被除数与用商“1”与除数相乘后的结果进行相减，得到的值为“1”。

（2）再从被除数中取下一位“0”下来，与上一步的差“1”值组成新的被除数，为“10”，显然它比除数“101”小，不够除。于是在商的对应位置上输入“0”。

（3）继续从被除数中取下一位“1”下来，与上一步的余数“10”组成新的被除数，为“101”，此数恰好与除数“101”相等，所以此时的商取“1”时正好除尽。

这样一来，（11001）$_2$÷（101）$_2$所得的商就是（101）$_2$了。

2.3.2 二进制的逻辑运算

逻辑运算是指对因果关系进行分析的一种运算。逻辑运算的结果并不表示数值大小，而是表示一种逻辑概念，若成立则为“真”，用“1”表示，若不成立则为“假”，用“0”表示。二进制的逻辑运算有“与”、“或”、“非”和“异或”四种。

1.“与”运算（AND）

“与”运算又称逻辑乘，用符号“.”或“∧”来表示。运算规则如下：

0∧0=0　0∧1=0　1∧0=0　1∧1=1

也就是说，在“与”运算中，只要两个参加“与”运算的数的对应码位中有一个数为 0，则运算结果肯定为 0，只有两数对应码位都为 1，“与”运算的结果才为 1。这与前面介绍的二进制乘法运算是一样的。图 2-11 是两个“与”逻辑运算的示例。

【注意】图 2-11 左图所示的是两个位数不一样的二进制数进行“与”运算，这时要求两个数从最低位开始对齐，在位数少的二进制的最高位前面加上“0”补齐，使得它与位数多的二进制数有一样的位数。

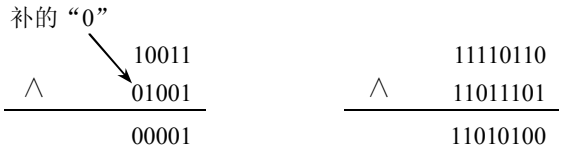

图 2-11　两个逻辑“与”运算示例

2.“或”运算（OR）

“或”运算又称逻辑加，用符号“+”或“∨”表示。运算规则如下：

0∨0=0　0∨1=1　1∨0=1　1∨1=1

也就是说，在“或”运算中，只要两个参加“或”运算的数的相应码位中有一个为 1，则运算结果为 1，只有两码位对应的数均为 0，结果才为 0。如图 2-12 是两个“或”逻

辑运算的示例。同样，进行“或”运算时要求两数从最低位开始对齐，少位数的数在最高位前面加“0”补齐，最终使两个二进制数的位数相同。

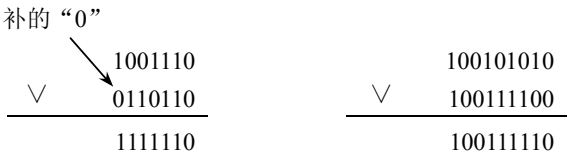

图 2-12　两个逻辑“或”运算示例

3.“非”运算（NOT）

“非”运算就是逐位求反的运算。非运算规则：“0”的反为“1”，“1”的反为“0”，也就是“0”与“1”互为反。注意“非”运算只是针对一个数所进行的“运算”，这与前面的“与”和“或”运算不一样。如“101110101”进行“非”运算后就得到“010001010”（可简写为“10001010”）。

4.“异或”运算（XOR）

“异或”运算用符号“⊕”来表示。其运算规则如下：

$0 \oplus 0 = 0$　　$0 \oplus 1 = 1$　　$1 \oplus 0 = 1$　　$1 \oplus 1 = 0$

也就是说，当两个参加“异或”运算的数对应码位相同时运算结果为 0，不同时运算结果为 1。如图 2-13 是两个“异或“逻辑运算示例。同样，进行“异或”运算时要求两数从最低位开始对齐，少位数的数在最高位前面加“0”补齐，最终使两个二进制数的位数相同。

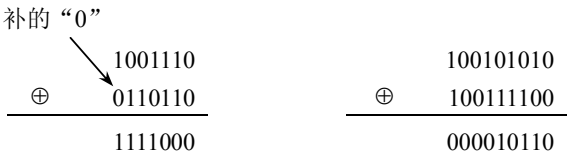

图 2-13　两个逻辑“异或”运算示例

2.4　二进制数的编码

通常，我们习惯用十进制表示数据，但在计算机中是用二进制来表示数据并进行运算的，这就需要进行数制之间的转换。我们把每位十进制数转换二进制的编码，称之为 BCD 码（Binary Coded Decimal，也称“二－十进制编码”）。二进制数的运算以及非数值信息表示都是基于 BCD 编码格式上的。

2.4.1　BCD 编码

我们知道，一个字节有 8 个位，而根据前面学习到的十进制与二进制的转换规则（参见表 2-1）可以得出，十进制中的数字 0～9 最多只需要使用 4 个位（8 和 9 分别用 1000 和 1001 表示），如果用一个字节来存储一个数字，这样就会有一半空间被浪费。于是人们就想出了压缩的办法，这就是前面介绍的 BCD 编码。

BCD 编码将一个字节的 8 个位拆分成高 4 位和低 4 位两个部分，这样一来，一个字节能存储两位数字。BCD 编码是用 4 位二进制编码来表示 1 位十进制数（与十六进制类似）。这种编码方法有多种，常见的有以下几种：

1．8421BCD 编码

这是一种使用最广的 BCD 码，是一种有位权的编码，其各位的位权分别是（从最高有效位开始到最低有效位）8、4、2、1（对应二进制中的权值：2^3、2^2、2^1、2^0），因而称之为“8421BCD 编码”。其实这与二进制转换成十进制的算法一样，只是这里仅有 4 位二进制。

在使用 8421BCD 码时一定要注意其有效的编码仅十个（因为十进制中仅 0～9 这十个码数），对应的二进制为：0000～1001。四位二进制数的其余六个编码 1010，1011，1100，1101，1110，1111 不是有效编码，因为在 8421BCD 编码中，每 4 位仅表示 1 位十进制数，而 1 位十进制数最大就是 9，转换成二进制时就是 1001。8421BCD 编码如表 2-2 所示。

表 2-2　8421BCD 编码

十进制数字	8421BCD 码	十进制数字	8421BCD 码
0	0000	5	0101
1	0001	6	0110
2	0010	7	0111
3	0011	8	1000
4	0100	9	1001

2．2421BCD 编码

2421BCD 码也是 BCD 编码的一种，也是一种有位权的编码，其从高位到低位的位权分别为 2、4、2、1，即 2^1、2^2、2^1、2^0。也可以用 4 位二进制数来表示 1 位十进制数。如表 2-3 所示。

表 2-3　2421BCD 编码

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

从上表可以看出，当最高位（第 4 位）为 0 时（小于等于十进制 4 时），2421BCD 编码与 8421BCD 编码是一样的，因为从名称上可以看出，权值不同仅在最高位（8421BCD 编码中的最高位权值为 8，而 2421BCD 编码中的最高位权值为 2）。当最高位为 0 时，则该位无权值了，0 与任何数相乘都是 0；而当最高位不为 0 时（大于十进制 4 时），就不同了。8421BCD 编码最高位权值为 2^3=8，而 2421BCD 编码的最高位权值为 2^1=2，这样一来得到的数字结果自然就不一样了。

下面以十进制 5 为例，它的 2421BCD 编码为 1011，根据以上权值展开原则可以得出：$1\times2^1+0\times2^2+1\times2^1+1\times2^0=2+0+2+1=5$。其他十进制数的计算方法一样，大家可以试一

下。要注意的是，这与普通的二进制转换成十进制方法是不一样的。

3．余 3 码

余 3 码也是一种 BCD 码，但它是无位权的编码，是通过每一个码与对应的 8421BCD 码值相差 3（其实就是比对应的 8421BCD 码大 3）而得名（如表 2-4 所示），故称为“余 3 码”。

表 2-4　余 3 码

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

其实计算余 3 码的方法很简单，就是求比对应的 8421BCD 码大 3 的十进制数字所对应的 8421BCD 码。如要求 5 的余 3 码，则只要求 8 的 8421BCD 码即可，因为这两者的值是一样的。这一点可以将表 2-4 中的 5 的余 3 码（1000）与对应的 8 的 8421BCD 码（1000）比较一下就可知道了，值是相等的。

2.4.2　二进制数的非数值数据表示方法

计算机除了具有数值运算能力之外，还具有非数值信息表示能力。现在，后者的应用领域已远远超过了前者的应用领域，如文字处理、图形图像处理、信息检索、日常的办公管理等。所以，对非数值信息的编码就显得越发重要。

1．基本的数据类型

汇编语言所用到的基本数据类型有字节、字、双字节等。下面对它们进行最基本的描述。

- 字节

一个字节由 8 位二进制组成（0～7 位）。在表示有符号数时，最高位就是符号位。通常情况下，存储器按字节编址，读写存储器的最小信息单位就是一个字节。

- 字

由 2 个字节组成一个字，共 16 位（0～15 位）。高 8 位称为高字节，低 8 位称为低字节。字节和字是汇编语言程序中最常用的两种数据类型，也是最容易出错的数据类型。

- 双字节

用 2 个字（4 个字节）来组成一个双字节，共有 32 位（0～31 位），其中高 16 位称为高字，低 16 位称为低字。双字节有较大的数据表示范围，它通常是为了满足数据的表示范围而选用的数据类型，也可用于存储远指针（可以指向其他 64KB 代码段的指针）。

字节、字和双字节是汇编语言最常用的三种数据类型。

- 四字节

由 4 个字（8 个字节）组成一个四字类型，共有 64 位（0～63 位）。它有更大的数据表示范围，但在汇编语言中很少使用该数据类型。

- 十字节

由 10 个字节组成一个十字节类型，共有 80 位（0～79）。在汇编语言中也很少使用该数据类型。

- 字符串

字符串是由若干个字节组成的，字节数不定，通常每个字节存储一个字符。该数据形式是汇编语言程序中另一种经常使用的数据形式。

2．字母与汉字编码

通信的目的是交换信息，信息的载体可以是数字、文字、语音、图形或图像。计算机产生的信息一般是字母、数字、语音、图形或图像的组合。为了传送这些信息，首先要将字母、数字、语音、图形或图像用二进制代码的数据来表示。而为了传输二进制代码的数据，必须将它们用模拟或数字信号编码的方式表示。在非数值数据中主要有以下几种编码类型：

（1）ASCII 码。

由于计算机是数字设备，内部用二进制数，这样对于从外部输入到计算机的所有信息必须用二进制数表示，并且要对各种命令、字符等进行二进制转换。这样就牵涉到信息符号转换成二进制数所要采用的编码问题，于是也就有了国际上统一的美国标准信息编码（ASCII，American Standard Code for Information Interchange，美国信息交换标准码）。

ASCII 码被国际标准化组织（ISO）接受，成为国际标准 ISO646，又称为国际 5 号码。它用于计算机内码，也用做数据通信中的编码标准。

在 ASCII 码标准中规定，一个字节为 8 位二进制，一个 ASCII 码占一个字节的低 7 位，最高位为校验位，用于在传输过程中检验数据正确性。也就是说 ASCII 码采用 7 位二进制比特编码。ASCII 码占的低 7 位二进制数表示一个字符，这样一个字节可表示 2 的 7 次方即 128 种状态（从 00000000～01111111）。每种状态与一个 ASCII 码字符唯一对应，即可表示 128 个字符。这 128 个字符的编码规则如表 2-5 所示。例如大写字母 C 的 ASCII 码，只需在图中对应于字符 C 的位置，找出其对应的行 $D_6D_5D_4$ 和列 $D_3D_2D_1D_0$，依次按 $D_6D_5D_4D_3D_2D_1D_0$ 的顺序排列出来，再在最高位补以 0，即得 C 的 ASCII 码为 01000011。

表 2-5　ASCII 码编码表

$D_6D_5D_4$ / $D_3D_2D_1D_0$	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	、	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	“	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	’	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z

续表

$D_6D_5D_4$ / $D_3D_2D_1D_0$	000	001	010	011	100	101	110	111
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

在这 128 个字符中，包括 26 个英文大写字符、26 个英文小写字符、10 个数字字符、33 个标点符号和 33 个控制字符。可以表示 128 个字符。在整个 ASCII 码字符中又分为“图形字符”与“控制字符”两类：图形字符包括数字、字母、运算符号、商用符号等。例如大写 A 的 ASCII 码是 65，小写 a 则是 97。为便于书写和记忆，有时，也将 ASCII 码写作十六进制形式，即将某字符的 ASCII 码二进制数形式，转换成十六进制数的形式，再标以 H 表示这是一个十六进制的数。例如字母 A 的 ASCII 码为 01000001（注意第 8 位是用“0”补齐的，下同），写成十六进制即 41H；字母 C 的 ASCII 码为 01000011，写成十六进制即 43H。

（2）汉字编码（内码）。

计算机处理汉字信息的前提条件是对每个汉字进行编码，这些编码统称为汉字编码。汉字信息在系统内传送的过程就是汉字编码转换的过程。在这其中又有几种编码形式：

- 汉字交换码

ASCII 码是针对英文的字母、数字和其他特殊字符进行编码的，它不能用于对汉字的编码（其能编码的个数也远远不够）。要想用计算机来处理汉字，就必须先对汉字进行适当的编码，这就是“汉字交换码”。我国在 1981 年 5 月对 6000 多个常用的汉字制定了交换码的国家标准，即 GB2314-80，又称为“国标码”。该标准规定了汉字交换用的基本汉字字符和一些图形字符，共计 7445 个，其中汉字有 6763 个。在这 6763 个汉字中，一级汉字（常用字）3755 个按汉语拼音字母顺序排列，二级汉字 3008 个按部首笔画顺序排列。该标准给定每个字符的二进制编码，即国标码。

- 区位码

它是将 GB 2314-80 的全部字符集组成一个 94×94 的方阵，每一行称为一个“区”的编码方式。在这种编码中的编号为 01～94：每一列称为一个“位”，编号也为 01～94，这样得到 GB 2314-80 标准中汉字的区位图，用区位图的位置来表示的汉字编码，称为区位码。

- 机内码

为了避免 ASCII 码和国标码同时使用时产生二义性问题，大部分汉字系统都采用将国标码每个字节高位置“1”作为汉字机内码。这样既解决了汉字机内码与西文机内码之间的二义性，又使汉字机内码与国标码具有极简单的对应关系。

汉字机内码、国标码和区位码三者之间的关系为：区位码（十进制）的两个字节分别转换为十六进制后加 20H 得到对应的国标码；机内码是汉字交换码（国标码）两个字节的最高位分别加 1，即汉字交换码（国标码）的两个字节分别加 80H 得到对应的机内码；区位码（十进制）的两个字节分别转换为十六进制后加 A0H 即得到对应的机内码。

（3）汉字输入码（外码）。

目前，汉字输入法主要有键盘输入、文字识别和语音识别。键盘输入法是当前汉字输入的主要方法。它大体可以分为：

- 流水码：如区位码、电报码、通信密码，优点是重码律少，缺点是难于记忆。
- 音码：以汉语拼音为基准输入汉字，优点是容易掌握，但重码律高。
- 形码：根据汉字的字型进行编码，优点是重码少，但不容易掌握。
- 音形码：将音码和形码结合起来，能减少重码律同时提高汉字输入速度。

（4）汉字字模。

供计算机输出汉字（显示和打印）用的二进制信息叫汉字字形信息，也称字模。通用汉字字模点阵规格有 16×16，24×24，32×32，48×48，64×64。每个点在存储器中用一个二进制位存储，如一个 16×16 点阵汉字需要 32 个字节的存储空间。

2.5 二进制数的编码表示形式及运算

计算机中的二进制数称之为“机器数”。机器数是带符号的，它是用一个数的最高位存放符号（0 为正，1 为负），其余位用来表示数本身，这样就将机器数的符号数值化了。尽管上节介绍了二进制数的编码类型，但并没有说到采用的编码表示形式。二进制数的编码可以有多种表示形式，最重要的就是原码、反码和补码这三种。

2.5.1 二进制数的编码表示形式

符号数值化后，为了方便对机器数进行算术运算（因为二进制数并不是每位都代表数值本身，最高位仅代表符号），提高运算速度，人们设计了符号数的各种编码方法，最常见的有原码、反码和补码。

1．真值、字长和模

在正式介绍机器数的三种主要编码方式前，先要明白几个与机器数密切相关的概念，那就是真值、字长和模。

- 真值

因为在机器数中，符号占了 1 位，这样一来机器数的形式值就不等于真正的数值。例如有符号数 10000101，其最高位 1 代表负，故其真正数值是-5，而不是形式值 133（10000101 转换成十进制等于 133）。所以，为区别起见，带符号位的机器数对应的真正数值称机器数的“真值”。例如：00100001 的真值 = 0 0100001 = +33，10100011 的真值 = 10100011 = -35。

- 字长

在机器数中还有一个概念需要清楚，那就是“字长”。计算的字长是指它一次可处理的二进制数字的数目，是计算机进行数据存储和数据处理的运算单位。如我们通常所指的 32 位处理器，就是指该处理器的字长为 32 位，也就是一次能处理 32 位二进制数。通常称 16 位是一个字，32 位是一个双字，64 位是两个双字。

数值的转换结果是与字长有关的。十进制中的数+5，如果计算机字长为 8 位，转换成二进制就是 00000101。如果是-5，转换成二进制就是 10000101，但如果字长是 16 位，+5 转换的结果就是 00000000 00000101 了。也就是对应的机器数要转换为字长所代表的位数。

字长越长当然所能处理的数就越大。在 8 位字长中，因为最高位要用于符号位，所以实际可用于数值表示的只有低 7 位，也就是可以处理的数值大小范围为-127～-0～0～127（2 的 7 次方），共 256 个数了；但是如果字长是 16 位，可以处理的数值大小就可以是-32512～-0～

0～32512（2 的 15 次方）。

【注意】以上的“-0”与“0”的机器数是不一样的，在 8 位字长中，-0 为 1 0000000，而+0 为 0 0000000；在 16 位字长中，-0 为 1 0000000 00000000，而+0 为 0 0000000 00000000。

- 模

在机器数中，还有一个很重要的概念，那就是“模”。“模”是指一个计量系统的计数范围。如时钟中的一个小时就是 60 分钟，这个 60 分钟就是“模”。计算机也可以看成一个计量机器，它也有一个计量范围，即都存在一个“模”。

“模”实质上是计量器产生“溢出”（也就是超过指定的范围）时的阈值。超过这个阈值的数在对应的计量器上是表示不出来的。例如我们的时钟是以 12 小时为一个周期的，也就是时钟计算的“模”为 12。如果时间超过 12 点，在时钟指针上并没有大于 12 的值，而是指在“0”和“1”之间。

对于机器数，模也就是相应位数寄存器所能表示的最大数（也就是 2 的 n 次方-1）再加 1（其实就是因为还有“0”这个数值），实际上结果就是 2 的 n 次方。如 8 位寄存器，模就等于 2^8=256，而 16 位寄存器所能存储的模为 2^{16}=65536。

2．原码

原码就是符号位加上真值的绝对值，其实就是数据值的机器数表示形式本身。比如，+3（以 8 位字长为例，下同），符号位为 0，3 转化为二进制就是 11，那么 +3 的原码就是 00000011（最高位为符号位，正数的符号位为“0”，其余数值位不足部分补 0）。同理，-3，符号位为 1，3 转化为二进制就是 11，那么 -3 的原码就是 10000011（不足 8 位时在前面用 0 补足）。即：$[+3]_{原}$= 00000011，$[-3]_{原}$= 10000011。

再来计算+127 和-127 各自的原码。+127 中符号位为“0”，127 的二进制为“1111111”，这样$[+127]_{原}$=01111111。而-127 的符号位为“1”，所以最终$[-127]_{原}$=11111111。

另外，要注意，原码表示形式中，0 有“+0”和“-0”之分。对应的原码分别是 0 0000000 和 1 0000000。

3．反码

仅用原码（也就是机器数本身）很快就发现：用带符号位的原码进行乘除运算时结果是正确的，而在进行加减运算时就可能会出现问题。下面的示例仍是以 8 位字长为例：

如最简单的 1-1，本来要等于 0，但如果用原码进行减法运算时却得到：

$(0\ 0000001)_{原}+(1\ 0000001)_{原}=(1\ 0000010)_{原}=(-2)_{原}$，这显然不正确。

【说明】以上计算方法是把原码的减法运算转换为加法运算方法进行的。就是把后面的减数变为负数的原码。如上例的“1-1”后面的“-1”就可以直接认为是“-1”的原码，这样就变成了“1+（-1）”这样的加法运算了。

因为在两个无符号整数的加法运算中是没有问题的，问题仅出现在带符号位的负数（其实就是减法运算）上。于是就出现了“反码”的概念，相对原码来说，它仅对负数进行了重新编码，就是正数的反码仍是原码本身，而负数的反码是对原码除符号位外的其余各位逐位取反产生的。

【注意】反码其实只是为后面介绍的原码转换为补码作过渡，并不能直接通过它把减法运算变为加法运算，因为许多情况下这样计算的结果是不正确的（但不是全部不正确）。如 2-3，用反码进行加法运算就得到：

$(0\ 0000010)_{反}+(1\ 1111100)_{反}=(1\ 1111110)_{反}$=-1，结果正确。

但再如十进制数 123-121，用反码加法运算就得到：

（0 1111011）$_{反}$+（1 0000110）$_{反}$=（0 0000001）$_{反}$=1_{10}，显然不正确。

与原码一样，在反码表示形式中，0 也有“+0”和“-0”之分，对应的反码分别为 0 0000000 和 1 1111111。

4．补码

在反码之后又引入了“补码”的概念，那是因为前面说了，仅通过反码的方式，在减法运算中并不能全面解决原码运算中存在的问题。在计算机中，整数的运算都是以补码形式进行的。但为什么有了“补码”，还要“反码”呢？原因是“补码”是通过“反码”进行一定的运算法则后得出的。那就是：负数的补码就是对反码加 1，而正数的补码不变。这样一来，正数的原码、反码、补码都是一样的了。

下面仍以 1-1、2-3 和 123-121 这三个例子为例介绍通过补码方式运算的结果。

1-1=（0 0000001）$_{补}$ +（1 1111111）$_{补}$ =（0 0000000）$_{补}$ = 0，是正确的。

十进制数 2-3=（0 0000010）$_{补}$ +（1 1111101）$_{补}$ =（11111111）$_{补}$ = -1_{10}，也是正确的。

十进制数 123-121=（0 1111011）$_{补}$+（1 0000111）$_{补}$=（0 0000010）$_{补}$=2_{10}，也是正确的

在补码表示形式中，0 仅有一种表示形式，因为无论是“+0”，还是“-0”的补码均为 0 0000000。

从以上运算结果可以得出设计“补码”的以下几个目的：

- 使符号位能与有效值部分一起参加运算，从而简化运算规则。补码机器数中的符号位，并不是强加上去的，而是数据本身的自然组成部分，可以正常地参与运算。
- 补码的用途是让机器学会减法运算。因为所有的处理器都是电路做的，电路其实只是加法器，只能做加法。如何能让电脑做减法呢，这就要用到补码。减去一个数就等于加上它的补码。于是减法就转换为机器能执行的加法了，这样电脑就能算减法了，进一步简化计算机中运算器的线路设计。

综上所述，所谓原码就是带正、负号的二进制数，即最高位为符号位，“0”表示正，“1”表示负，其余位表示数值的大小。反码表示法规定：正数的反码与其原码相同；负数的反码是对其原码除符号位外逐位取反。补码表示法规定：正数的补码与其原码、补码相同；负数的补码是在其反码的末位加 1。由此可见，这三种表示法中，关键是负数的表示方式不一样，正数的原码、反码、补码都是一样的。

2.5.2 补码的加减法运算

上节介绍了原码、反码和补码的转换方法，同时我们知道，在计算机的机器码中，实际上全是采用补码方式进行运算，特别是减法运算。因为原码和反码的减法运算有时结果是不正确的。所以本节仅介绍补码的加、减法运算方法。

1．补码的转换

在正式介绍补码的加减法运算前，还是先回顾一下补码的计算方法，这是补码运算的基础。通过前面的学习，我们知道，机器数的补码可由原码和反码得到。如果机器数是正数，则该机器数的补码与原码、补码一样；如果机器数是负数，则该机器数的补码是对它的反码在末位加 1 得到的。

例如，X= + 0.1011 时，根据以上规则可得到［X］$_{补}$= 0.1011（因为正数的补码与原码、补码一样）。

当 X = - 0.1011 时，则$[X]_{补}$ = 1.0101（负数的补码是符号位不变，真值是在它的反码基础上最末位加 1 进行二进制加法运算得到）。这里的“1011”的反码（也就是按位取反）为“0100”，再在末位加“1”后即得到了“0101”，再加上符号位“1”，所以最后的值为 1.0101（符号位是最高位）。

又例如，X = +1010 时，则［X］$_{补}$ = 01010（正数的补码与原码、补码一样，注意在最高位要体现数的正、负符号）。

当 X = -1010 时，则$[X]_{补}$ = 10110。计算方法一样：先计算“1010”的反码，为“0101”，再在末位进行加“1”二进制运算，得到“0110”，再加上符号位“1”，即得到“10110”。

【注意】整数“0”的补码只有一种表示形式，即 00…0，因为“-0”的补码最终结果也是“0”。现在来通过补码的转换规则计算一下“-0”的补码。首先根据“-0”的原码（1 0000000）求其反码，得到“-0”的反码为“1 1111111”（最高位 1 为符号位）；然后在真值部分末位加“1”，得到一个 9 位的值，1 00000000，如果是 8 位字长格式，则第 9 位会溢出，这样最后得到的补码值同样为“00000000”。如果是采用 16 位字长的，“-0”的补码同样为“00000000 00000000”，因为这将产生值为“1”的第 17 位，而这个第 17 位在 16 位字长中同样是溢出的。

表 2-6 是 8 位二进制的原码、反码、补码对照表。在进行编码转换时如果记得这个表，有时速率会快许多。

表 2-6　8 位二进制的原码、反码、补码对照表

二进制数码	无符号数	原码	反码	补码
00000000	0	+0	+0	+0
00000001	1	+1	+1	+1
00000010	2	+2	+2	+2
:	:	:	:	:
01111111	126	+126	+126	+126
01111111	127	+127	+127	+127
10000000	128	-0	-127	-128
10000001	129	-1	-126	-127
10000010	130	-2	-125	-126
:	:	:	:	:
11111110	254	-126	-1	-2
11111111	255	-127	-0	-1

2．补码的加减法运算

前面提到了，在计算机中，数值的运算和存储都是采用补码形式进行的，原因是补码形式的数字无论进行哪种运算，都可以直接以加法或者减法进行。而不像原码和反码那样有时在减法运算时会出现错误的结果。之所以补码数字在运算时不会出错，原因在于它把数字的符号位连同真值部分一起参与了运算。

- 补码的加法运算

补码的加法运算法则如下：

$$[X+Y]_{补}= [X]_{补}+[Y]_{补} \quad (1)$$

该式表明，如果要求两个有符号数和的补码，可以分别对两个数求补码，然后相加。在采用补码形式表示时，进行加法运算可以把符号位和数值位一起进行运算（若符号位有进位，则溢出不管），结果为两数之和的补码形式。

如要求下列两个算式的补码（按 8 位字长格式计算）：（+35）+（+18）；（+35）+（–18）。

计算这样一个补码数时，根据公式（1）的运算规则，可以分别求算式中各数的补码，然后再直接相加即可。但要记着的一点就是，正数的原码、反码和补码是一样的。所以像求（+35）+（+18）算式中的“+35”的补码，只需求得它的原码即可直接得到它的补码。而“+35”的原码是 0 0100011（记住最高位为符号位），因为正数的补码与原码一样，所以“+35”的补码也为 0 0100011。

同样在（+35）+（+18）算式中的“+18”的补码也与它的原码一样，均为 0 0010010。这样一来求（+35）+（+18）算式的补码，就直接把这两个数的补码相加即可。如图 2-14 左图所示，结果为 00110101。如果在最高位有溢出，则丢弃。

在（+35）+（–18）中，前面那个数“+35”与上一算式一样，为正数，可以直接得到它的补码 0 0100011；后面那个“-18”因为是负数，不能直接从它的原码得到补码了。需要先求它的原码，然后再根据反码计算规则得到它的反码，最后再根据补码计算规则得出它的补码。“-18”的原码为 1 0010010，根据反码与原码的转换原则（除符号位外，其他位按位取反）得到它的反码为 1 1101101，然后再根据补码与反码的转换原则（在反码的真值最低位加 1），得到其补码为 1 1101110。然后再与“+35”的补码进行加运算即可，如图 2-14 右图所示，结果为 00010001。同样如果在最高位有溢出，则丢弃。

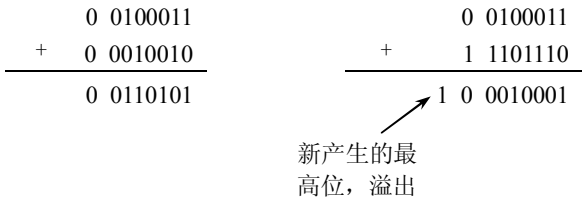

图 2-14　两个补码加法运算示例

- 补码的减法运算

补码的减法运算规则是：

$$[X-Y]_{补}=[X]_{补}+[-Y]_{补} \qquad (2)$$

该公式表明，求两个机器数的差值的补码（如$[X-Y]_{补}$），可以通过求被减数的补码（如$[X]_{补}$）与减数的负值的补码（$[-Y]_{补}$）的和得到。

$[-Y]_{补}$是对减数进行求负操作（一般称已知 $[Y]_{补}$求 $[-Y]_{补}$的过程为“变补”或“求负”）。求负的规则是全部位（含符号位）按位取反后再加 1。例如：已知$[15]_{补}$=00001111，则$[-15]_{补}$=11110000+1=11110001。

现假设 X=+35，Y=+18，要求$[X-Y]_{补}$。

先根据正数的补码与原码一样的规则，求得$[X]_{补}$= 00100011，$[Y]_{补}$ = 00010010；再根据以上介绍的补码求负操作规则，可以得到$[-Y]_{补}$=11101101；最后用$[X]_{补}+[-Y]_{补}$公式即可得到最终的$[X-Y]_{补}$ = 00010010。运算过程如图 2-15 左图所示，结果为 00010000。超出 8 位的最高位溢出。

如果是 X= –35，Y= –18，仍要求$[X-Y]_{补}$，则同样需要先求得$[-Y]_{补}$，也即$[-(-18)]_{补}$，实际上是要求[+18]的补码。因为正数的补码与原码一样，所以很快可以得到 $[+18]_{补}$=00010010。这样$[X-Y]_{补}$就等于$[-35]_{补}+[+18]_{补}$。“-35”因为是负数，要求其补码，则先要求其

原码。“-35”的原码为 1 0100011，它的反码为 1 1011100，由此得到它的补码为 1 1011101。最后$[-35]_{补}+[+18]_{补}$的运算过程如图 2-15 右图所示，结果为 11101111。

```
    0 0100011                    1 1011101
+   1 1101101                +   0 0010010
-------------                -------------
  1 0 0010000                  1 1101111
  ↗
新产生的最
高位，溢出
```

图 2-14　两个补码减法运算示例

2.6　实战训练

一、填空题

1．计算机数制一般有_____、_____、_____和_____四种，分别用_____、_____、_____和_____字符标识。计算机一般都采用_____数进行运算、存储和传送。

2．目前常见的机器编码有原码、反码和补码三种。原码的编码规则是：最高位代表_____，其余各位是该数的_____；反码的编码规则是：正数的反码_____，负数的反码是将二进制位_____；补码的编码规则是：正数的补码_____，负数的补码是将二进制位_____后在最低位加_____。

3．对 -0 和 +0 有不同表示方法的机器码是_____和_____，_____是一样的。8 位寄存器中存放二进制整数，内容全为 1，当它为原码、补码和反码时所对应的十进制真值分别是_____、_____、_____。

4．十进制整数转换成二进制的方法是_____，十进制小数转换成二进制的方法是_____。7420.45Q 的十六进制数是_____。

5．采用 BCD 码，1 位十进制数要用_____位二进制数表示，1 个字节可存放_____个 BCD 码。36D 的 8421 码为_____。

6．逻辑操作有_____、_____、_____和_____四种类型。

二、选择题（可多选）

1．8 个二进制位至多可表示（　）个数据。

A．8　　B．64　　C．255　　D．256

2．与二进制数 100101.001101 等值的十进制数是（　）。

A．25.203125　　B．25.34　　C．37.203125　　D．37.34

3．与二进制数 100101.001101 等值的十六进制数是（　）。

A．25.203125　　B．25.34　　C．25.31　　D．91.0D

4．下列数中最小的数为（　）。

A．10101101B　　B．256Q　　C．ACH　　D．171D

5．将 -33 以单符号位补码形式存入 8 位寄存器中，寄存器中的内容为（ ）。
A．DFH　　B．A1H　　C．5FH　　D．DEH

6．对 +0 和 -0 表示形式唯一的机器码是（ ）。
A．原码　　B．补码　　C．反码　　D．真值

7．与十六进制数 AC.E 等值的十进制数是（ ）。
A．112.875　　B．162.875　　C．172.7　　D．172.875

8．与十六进制数 AC.E 等值的八进制数是（ ）。
A．254.16　　B．254.7　　C．530.07　　D．530.7

9．多项式 $2^{14}+2^{11}+2^4+2^1+2^0$ 表示为十六进制数为（ ）。
A．4813H　　B．8026H　　C．2410H　　D．EB410H

10．多项式 $2^{14}+2^{11}+2^4+2^1+2^0$ 表示为十进制数为（ ）。
A．18 448　　B．9 232　　C．18 451　　D．36 902

11．8 位补码可表示定点整数的范围是（ ）。
A．–127～+127　　B．–128～+128　　C．–128～+127　　D．–127～+128

12．原码 1.0101110 所表示的真值为（ ）。
A．-0.0101110　　B.+0.0101110　　C.-0.1010010　　D.+0.1010010

13．8 位反码可表示定点小数的范围是（ ）。
A．–1～1　　B．$-1\sim1-2^{-7}$　　C．$-1+2^{-7}\sim1$　　D．$-1+2^{-7}\sim1-2^{-7}$

14．在计算机加减法运算中，最常使用的是（ ）。
A．原码　　B．补码　　C．反码　　D．ASCII 码

15．每个字节中可存放（ ）个 BCD 码数码。
A．4　　B．3　　C．2　　D．1

16．计算机中字符的编码为（ ）。
A．原码　　B．补码　　C．反码　　D．ASCII 码

17．补码的作用是（ ）。
A．使机器数的码制简单　　B．使计算机的运算符合其物理性能
C．能将负数转换为正数　　D．能将减法转化为加法

18．对于 n 位二进制整数，（ ）的表示范围为：-（2^{n-1}–1）～+（2^{n-1}-1）。
A．原码　　B．补码　　C．反码　　D．都不是

19.（ ）的编码保持了数据原有的大小顺序。
A．原码　　B．补码　　C．反码　　D．移码

20．二进制整数采用机器码表示时，（ ）的表示范围最大。
A．原码　　B．补码　　C．反码　　D．BCD 码

21．设字长 8 位并用定点整数表示，模为 2^8，若$[X]_{补}$=11111010，则 X 的原码及真值 X 分别为（ ）。
A．$[X]_{原}$=00000110，X=+0000110　　B．$[X]_{原}$=10000110，X=-0000110
C．$[X]_{原}$=01111010，X=+1111010　　D．$[X]_{原}$=11111010，X=-0000110

22．对于 *R* 进制数，在每一位上的数字可以有（ ）种。
A．*R*/2　　B．*R*–1　　C．*R*　　D．*R*+1

23．十六进制数 1000 转换成十进制数是（ ）。
A．4096　　B．1024　　C．2048　　D．8192

24．假设用 12 个二进制位表示数据，它能表示的最大无符号整数为（ ）。
A．2047　　B．2048　　C．4095　　D．4096

25．与十进制数 28.625 等值的十六进制数为（　）。

A．112.10　　B．1C.A　　C．1C.5　　D．112.5

26．与二进制数 11101.010 等值的十进制数为（　）。

A．31.25　　B．29.75　　C．29.5　　D．29.25

27．与十六进制数 23.4 等值的十进制数为（　）。

A．35.5　　B．23.4　　C．35.75　　D．35.25

28．与十进制数 254 等值的二进制数是（　）。

A．11111110　　B．11101111　　C．11111011　　D．11101110

29．十进制数 125.24 对应的二进制数是（　）。

A．111101.1100　　B．1111101.0011

C．1111101.11　　D．1111001.0011

30．下面二、十、十六进制数之间转换正确的是（　）。

A．十进制数 200 转换为二进制无符号数是 11111000B

B．十进制数 122 转换成十六进制数是 7AH

C．十进制数 439 转换成 8421BCD 码是 010000011001

D．十六进制数 F2H 转换成十进制数是 234

31．真值 X=-127D，则其二进制真值及 8 位反码分别为（　）。

A．X=-1000000　$[X]_{反}$ =11111111　　B．X=-1000000　$[X]_{反}$ =10000000

C．X=-1111111　$[X]_{反}$ =11111111　　D．X=-1111111　$[X]_{反}$ =10000000

32．若 X=10111001，Y=11110011，则 X 和 Y“逻辑与”的值及“逻辑异或”的值分别为（　）。

A．110101100，000001101　　B．001010011，111110010

C．10110001，01001010　　D．01001110，11111011

33．若真值 X 为负小数，则用 n 位二进制数（含符号位）表示的原码定义为（　）。

A．$[X]_{原}=X$　　B．$[X]_{原}=1-X$　　C．$[X]_{原}=X-1$　　D．$[X]_{原}=2^{n-1}-X$

34．反码的作用是（　）。

A．作为求补码的中间手段　　B．作为求原码的中间手段

C．能将负数转换为正数　　D．能将减法转化为加法

35．某数在计算机中用 8421BCD 码表示为 0011 1001 1000，其真值为（　）。

A．398　　B．398H

C．1630Q　　D．1110011000B

36．在（　）表示中，数值 0 是唯一表示的。

A．原码　　B．反码　　C．补码　　D．原码或反码

37．若用 8 位机器码表示十进制数–101，则原码表示的形式为（　）。

A．11100101　　B．10011011　　C．11010101　　D．11100111

38．若用 8 位机器码表示十进制数–101，则补码表示的形式为（　）。

A．11100101　　B．10011011　　C．11010101　　D．11100111

39．已知 $x=-105/128$，若采用 8 位机器码表示，则 $[x]_{原}=$（　）。

A．10011010　　B．11010101　　C．11101001　　D．10100111

40．已知 $x=-105/128$，若采用 8 位机器码表示，则 $[x]_{补}=$（　）。

A．10010111　　B．11010101　　C．11101010　　D．10100111

三、判断题

1．正的二进制定点小数的真值与机器码相同。（ ）

2．十进制整数和十进制小数转换成二进制数的方法相同。（ ）

3．n 位二进制整数和小数补码的模都为 2^n。（ ）

4．n 位二进制负整数反码就是该数本身。（ ）

5．文字信息分为字符信息和汉字信息，都用 8 位 ASCII 码表示。其中字符信息的最高位为 0，用 2 个 8 位 ASCII 码表示，占 1 个字节。（ ）

6．BCD 码具有二进制的形式，又具有十进制的特点。（ ）

7．8421 码是一种有权码，用其表示的十进制数正好与相应的 4 位二进制数按权展开求和的值相等。（ ）

8．汉字输入码是指从键盘上输入的代表汉字的编码，简称外码。（ ）

9．逻辑运算的特点是只在对应的两个二进制位上进行，与相邻的高低位之间不发生关系，不存在进位、借位等。（ ）

10．在原码、补码、反码中，补码的表示范围最大。（ ）

四、计算题

1．将下列数转换成十进制数。

①（100110.101）$_2$；②（5675）$_8$；③（3B）$_{16}$。

2．将下列十进制数转换成二进制数，再转换成八进制数和十六进制数。

① 234D；②131.5D；③ 27/32。

3．写出下列二进制数的原码、反码和补码。

① 11010100B；② 0.1010000B；③ -10101100B；④ -0.0110000B。

4．将（143.65）$_8$ 转换成十进制；将（25.3125）$_{10}$ 转换为二进制；将（29.625）$_{10}$ 转换成八进制；将（10100101.01011101）$_2$ 转换成八进制；将（302.54）$_8$ 转换成二进制；将（1111111000111.100101011）$_2$ 转换成十六进制；将（3C.A6）$_{16}$ 转换成二进制。

5．（10100101）$_2$ 和（00111101）$_2$ 的“与”、“或”和“异或”逻辑运算结果是什么？（10100101）$_2$ 与（00111101）$_2$ 各自进行“非”逻辑运算后的结果又是什么？

6．计算（1001101）$_2$ 与（110）$_2$ 的加、减、乘法运算结果。计算（1001110）$_2$ 与（110）$_2$ 的除法运算结果。