# 第二部分 数理逻辑基础（逻辑思维）

逻辑思维创新发展故事

进制的选取

1. 数的表示、存储与运算

### 计算机数据组织简介

计算机的应用领域及其广泛，但不论其应用在什么地方，信息在机器内部的形式都是一致的，即均位0和1组成的各种编码。这些二进制数字也被称为位(bit)，形成了第三次工业革命的基础。如今大家熟悉并使用了1000多年的十进制（以10为基数）起源于印度，在12世纪被阿拉伯数学家改进，并于13世纪被意大利数学家Leonardo Pisano（大约公元1170—1250，更为大家所熟知的名字是Fibonacci）带到西方。正常人类拥有10个手指头，所以使用十进制表示法是很自然的事情。但是当选择在机器上处理信息的数制时，使用二进制来工作的效果更好。二进制信号能够很容易地被表示、存储和传输，例如，在早期计算机上使用的穿孔卡片上用有洞和无洞来表示1和0、导线上的高电压或低电压，或者顺时针或逆时针的磁场。对二进制信号进行存储和执行计算的电子电路非常简单和可靠，并且经过集成电路的不断发展，制造商能够在一个单独的硅片上集成数百万甚至数十亿个这样的电路。

然而单个位不是非常有用，但是把这些位组合在一起，再人为赋予这些不同的可能位的组合以意义，我们就能够表示任何有限集合的元素。例如，使用一个二进制数字系统，我们可以用不同位的组合来对负数进行编码。我们还可通过使用标准定义的字符码来对文档中的字母和其他符号进行编码。在本章中，我们将讨论这两种编码，以及带符号数和其他音视频的编码等。

我们研究了三种最重要的数字表示。无符号（unsigned）编码基于传统的二进制表示法，能够表示大于或者等于0的数字。补码（two’s-complement）编码是表示有符号整数的最常见方式，有符号整数就是正数或者负数。浮点数（floating-point）编码是表示实数的以2为基数来表示的科学计数法版本。计算机用这些不同的数的表示方法来实现算术运算，例如加法和乘法，类似于对应的整数和实数来运算。

计算机的表示法是用有限数量的位来对一个数字编码，因此，当结果太大以至于不能表示时，某些运算就会产生溢出(overflow)。溢出会造成错误的结果。例如，在今天的大多数计算机上使用32位来表示数据类型int，计算表达式200\*300\*400\*500会得出结果为-884 901888。这显然违背了整数运算的特性，一些正数的乘积，结果不会得到负数。

另一方面，计算机在运算整数时满足我们熟知的真正整数运算的许多性质。例如，利用乘法的结合律和交换律，计算下面任何一个C表达式，都会得出结果-884 901888:

( 500 \* 400 ) \* ( 300 \* 200 )

( ( 500 \* 400 ) \* 300 ) \* 200

( ( 200 \* 500) \* 300 ) \* 400

400 \* ( 200 \* ( 300 \* 500 ) )

虽然上述表达式得到的结果都是错的，但是它们的结果至少是一致的！

浮点运算有完全不同的数学属性。虽然在溢出时会产生特殊的值，但是一组正数的乘积总是正的。由于表示的精度有限，浮点运算是不可结合的。比如在大多数机器上，表达式(3.14+1e20)-1e20求得的值会是0.0，而 3.14+(le20-le20)求得的值会是3.14。整数运算和浮点数运算会有不同的数学属性是因为它们处理数字表示有限性的方式不同——整数的表示虽然只能编码一个相对较小的数值范围，但是这种表示是精确的；而浮点数虽然可以编码一个较大的数值范围，但是这种表示只是近似的。

通过研究数字的实际表示情况，我们能够了解可以表示的值的范围和不同算术运算的属性。为了使编写的程序能在全部数值范围内正确工作，而且具有跨平台的可移植性，了解这种属性是非常重要的。

计算机用几种不同的二进制表示形式来编码数值。在本章中，我们会介绍这些编码，并且教你如何推出这些数字的表示。通过直接操作的数字的位级表示，我们得到了几种算术运算的方式。理解这些技术对于理解编译器产生的机器码是很重要的，并且编译器会试图优化算术表达式求值的性能。

我们对这部分内容的处理是基于一些核心的数学原理。从编码的基本定义开始，然后得出一些属性，例如可表示的数字的形式以及算术运算的属性。我们相信从这样一个抽象的观点来分析这些内容，对你来说是很重要的，因为作为计算机专业学生需要对计算机运算与更为人熟悉的整数和实数运算之间的关系有清晰的理解。

### 数的表示及进制转换

### 3.2.1数的表示

1. 十进制数的表示方法

十进制计数法的特点如下：

1. 使用10个数字符号0，1，2，……，9的不同组合来表示一个十进制数。这些符号称为数码，数码的个数称为基数，十进制的基数是10。
2. 一个数中，每个数码表示的值不仅仅取决于数码本身，还取决于其所处的位置(对每一个数码赋以不同的“权重”)。十进制中，每个数码上的权是10的某次幂。从个位，十位，百位开始，权重依次为100，101，和102，例如678=6×103＋7×10'+8×10°。每个数位上的数字所表示的量是该位数字和该数位上的权的乘积。
3. 逢十进一。任何一个十进制数可以用以下公式来表示：

上式(3.1)中，m表示小数位的位数，n表示整数位的位数，a表示第i位上的数码(可以是0-9中的任意一个)

1. 二进制数

式(3.1)可以推广到任意进制数。设其基数为R，则任意数N为

而对于二进制，R=2，ai为0或1，逢二进一。

例如，-1101.01012 = -13.312510，对于计算机存储和处理，负号和小数点是不方便的，因为只能用二进制数字(0和1)来表示数。如果只使用非负整数，那么其表示是直截了当的。一个8位的二进制数能表示从0~255的数、例如：

00000000 = 0

00000001 = 1

00101001 = 41

10000000 = 128

11111111 = 255

1. 八进制数

对于八进制，R=8，ai为0~7中的任何一个，逢八进一。

1. 十六进制数表示法

对于十六进制，R=16，ai为0~9，A，B，C，D，E，F中的任何一个，逢十六进一。



一个字节由8位组成。在二进制表示法中，它的值域是000000002~111111112。如果看成十进制数，它的值域就是010~25510.两种符号表示法对于描述位模式都不是非常方便。二进制表示法太冗长，而十六进制表示法与位模式的互相转化很麻烦、替代的方法是，以16为基数，或者叫十六进制数(hexadecimal)数，来表示位模式。十六进制(简写为hex)使用数字0~9以及字符A，B，C，D，E，F来表示16个可能的值。下表3.1展示了16个十六进制数字对应的十进制值和二进制值。用十六进制书写，一个字节(8位二进制数)的值域为0016~FF16。

表3.1

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 十六进制 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 十进制值 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 二进制值 | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 |
| 十六进制 | 8 | 9 | A | B | C | D | E | F |
| 十进制值 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 二进制值 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |

在C语言中，以0x或0X开头的数字常量被认为是十六进制的值。字符A~F可以是大写也可以是小写。例如可以将FA1D37B16写作0Xfa1437b，或者，0xfa1d37b，甚至是可以大小写混合，比如0xFa1D37b。

### 3.2.2进制转换

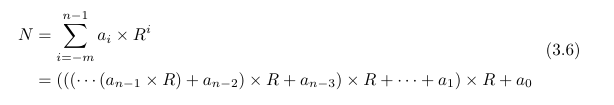
1. 任意进制数转化为十进制数

二进制，八进制和十六进制以至任意进制数转换为十进制数的方法很简单，可先将其按定义展开为多项式，再将系数及权均用十进制表示，按十进制进行乘法与加法运算，所得结果即为该数对应的十进制数。例如：

(101.01)2 = 1 × 22 + 0 × 21 + 1 × 20 + 0 ×2-1 + 1 × 1-2 = 5.25

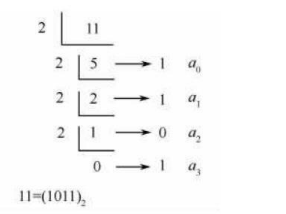
(AC7)16 = 10 × 162 + 12 × 161 + 7 × 160 = 2759

1. 十进制数转化为任意进制数

设N为任一十进制整数，如果要把它转换成n位R进制整数，则有

从式3.6可以看出，等式右边除了最后一项a0以外，其余各项都是包含基数R的因子，都能被R除尽。所以等式两边同除以基数R取其余数的方法得到ai。首先得到的是a0，如此一直进行下去，直到商等于0为止，就得到一系列余数a0，a1，…，an-1，他们正是要求的R进制数的各位。

十进制整数转换为任意进制整数的方法总结为：除以基数R取余数，先为低位后为高位。

例如将十进制数11转换为二进制数。

除了这种方法，如果对2的倍数比较熟悉的话，还可以直接看出，例如11 = 8+2+1，所以其二进制表示为1011。再比如592=512+64+16，其二进制数表示为100101000。

而十进制数转换为十六进制数需要使用乘法或者除法来处理一般情况。将一个十进制数字x转换为十六进制，可以反复地使用16除x，得到一个商q和一个余数r，也就是x = q × 16 + r。然后我们用十六进制数字表示的r作为最低为数字，并且对q反复进行这个过程得到剩下的数字。例如，考虑十进制314156的转换：

314156 = 19634 × 16 + 12 (C)

19634 = 1227 × 16 + 2 (2)

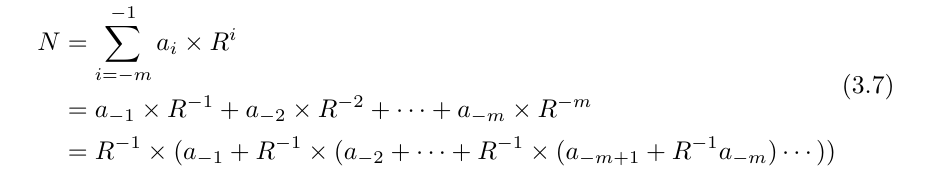
1227 = 76 × 16 + 11 (B)

76 = 4 × 16 + 12 (C)

4 = 0 × 16 + 4 (4)

通过上述步骤，我们能得出其十六进制表示为0x4CB2C。

对十进制小数转换成非十进制小数则使用如下方法。设N为任一十进制小数，若要把它转换为m位R进制小数，则有：



因此，可以将十进制小数不断乘以R，再取其乘积的整数作为ai，直到小数部分为零时停止。首先得到的是a-1，然后依次得到a-2，a-3，…，a-m。若乘积的小数部分始终不为0，说明相对应的R进制小数为不尽小数。这时可以乘到能满足计算机精度要求为止。综上所述，可以把十进制小数转换为相应R进制小数的方法总结为：乘以基数R取整数，现为高位后为低位。

例如，将0.625分别转换为二进制小数，其具体过程如下：

0.625 × 2 = 0.25 a-1 = 1

0.25 ×2 = 0.5 a-2 = 0

0.5×2 = 1 a-3 = 1

0.625 = (0.101)2

1. 二进制数与十六进制数之间的转换

因为24=16，即可用4位二进制数表示1位十六进制数，所以二进制和十六进制之间的转换比较简单直接。数字的转换可以参考上表3.1。一个简单的方法是：以小数点为界，向左(整数部分)每4位为一组，高位不足4位时补0；向右(小数部分)每4位为一组，低位不足4位时补0.然后分别用一个十六进制数表示每一组中的4位二进制数。

将十六进制数转换为二进制数的方法是：直接将每1位的十六进制数写成其对应的4位二进制数(参考表3.1)。

例如，假设一个十六进制数0x173A4C，可以通过展开每个十六进制数字，将它转换为二进制格式，如下所示：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 十六进制 | 1 | 7 | 3 | A | 4 | C |
| 二进制 | 0001 | 0111 | 0011 | 1010 | 0100 | 1100 |

这样就得到了该十六进制数的二进制表示：000101110011101001001100

反过来，如果给定了一个二进制数字1111001010110110110011，可以先把分为4位一组，每一组再转换成对应的十六进制数，再拼接起来。但是最左边的一组可能不足4位，这就需要在其前面补0，以满足4位。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 二进制 | (00)11 | 1100 | 1010 | 1101 | 1011 | 0011 |
| 十六进制 | 3 | C | A | D | B | 3 |

括号中为位数不足4位是补的0。

当值x是2的非负整数n次幂时，也就是x=2n，我们可以很容易地将x写成十六进制形式，只要记住x的二进制表示就是1后面跟n个0。十六进制数字0代表4个二进制0。所以，当n表示成i＋4j的形式，其中0≤i≤3，我们可以把x写成开头的十六进制数字为1(i=0)、2(i=1)、4(i=2)或者8(i=3)，后面跟随着j个十六进制的0。比如，x=2048=211，我们有n=11=3＋4·2，从而得到十六进制表示0x800。

而对于二进制与八进制的转换方式与十六进制的转换方式类似，只是在分组时每组只有3位。例如(725)8的转换，

|  |  |  |  |
| --- | --- | --- | --- |
| 八进制 | 7 | 2 | 5 |
| 二进制 | 111 | 010 | 101 |

那么得到其二进制表示为111010101，反过来也是一样，这里不在举例。

对于八进制与十六进制的转换可以借助二进制这个中间变量来转换，先将某数转换为二进制数，再转换成其他进制数。

### 数的码制

码制即编码体制，在数字电路中主要是指用二进制数来表示非二进制数字以及字符的编码方法和规则。日常生活中遇到的数，除了上述的无符号数(上面各进制数我们都是以正数为例)，还有带符号的二进制数，在计算机中通常用二进制数的最高位来表示数的符号。对于一个字节型(8位)二进制数来说，其最高位(最左边的位)表示符号位，剩下位为数值位。在带符号数中，规定用0表示正，用1表示负，而数值为表示该数的数值大小。

把一个数及其符号位在机器中的一组二进制数表示形式，称为“机器数”。机器数所表示的值称为该机器数的“真值”。例如：

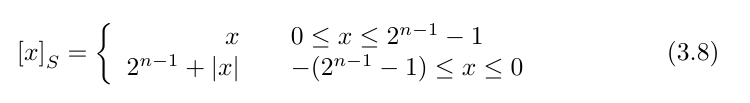
+32 = 0 0100000

-32 = 1 0100000

空格用来分割符号位与数值位。

### 3.3.1原码

原码(Sign-Magnitude)：最高有效位是符号位，用来确定剩下的位应该取负权还是正权。数x的原码记为[x]S，机器字长为n(二进制位数为n)，定义如下：



在原码的表示法中，最高位为符号位(正数为0，负数为1)，其余数字位表示数的绝对值，例如：

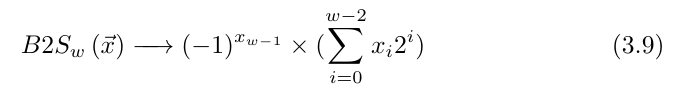
x1 = +1010101， 则[x1] S = 01010101，最高位的0表示它是正数

x2 = -1011101，则[x2] S = 11011101，最高位的1表示它是负数

可以看出，8位二进制原码表示数的范围为-127 ~ +127，16位二进制原码表示数的范围为-32767 ~ +32767。值得注意的是，0的原码表示有两种方式：

[+0] S = 00000000 或 [-0] S = 10000000

另外，式(3.9)给出了一个二进制原码序列到整数的映射公式。



其中B2SW(Binary to Unsigned)是一个函数，下标w为二进制原码序列的位数，该函数描述了将一个长度为w的二进制序列映射到一个整数。

总的来说，原码表示法简单直观，且与真值的转换很方便，但不便于在计算机中进行加减运算。如进行两数相加，必须先判断两个数的符号是否相同。如果相同，则进行加法运算，否则进行减法运算。如进行两数相减﹐必须比较两数的绝对值大小，再由大数减小数﹐结果的符号要和绝对值大的数的符号一致。按上述运算方法设计的算术运算电路很复杂。为此引入了反码和补码表示法，它们可以使正、负数的加法和减法运算简化为单一相加运算。

### 3.3.2反码

### 3.3.4补码

### 定点数和浮点数

### 3.4.1定点数

小数点再数中的位置是固定的

### 3.2.2浮点数

IEEE754

### 十进制数的编码

### 3.5.1有权码

8421

### 3.5.2无权码

### 数的存储（数字存储、文本存储、音频存储、图像存储）

### 3.6.1数字存储

Long int double

### 3.6.2文本存储

Unicode，gbk，ascii等等

### 3.6.3音频存储

Mp3等等

### 3.6.4图像存储

Jpg,jif等等，ffmpeg，MP4,h264

### 数据运算（逻辑运算、移位运算、算术运算）

### 3.7.1逻辑运算

与或非，异或同或。。。

### 3.7.2移位运算

左移右移，循环

### 3.7.3算术运算

加减乘，

1. 数字逻辑与数字系统

### 4.1基本逻辑关系

真值表，离散数学，

### 4.2逻辑门

与或非，及其组合

### 4.3逻辑代数与逻辑函数