原码（true form）是一种计算机中对数字的二进制定点的表示方法。原码是指一个[二进制数](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=268440&ss_c=ssc.citiao.link)左边加上符号位后所得到的码，且当二进制数大于0时，符号位为0；二进制数小于0时，符号位为1；在计算机内，[定点数](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=9001258)有3种表示法：[原码](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=3982703)、反码和[补码](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=3979855)。反码是数值存储的一种，但是由于补码更能有效表现数字在计算机中的形式，所以多数计算机一般都不采用反码表示数。

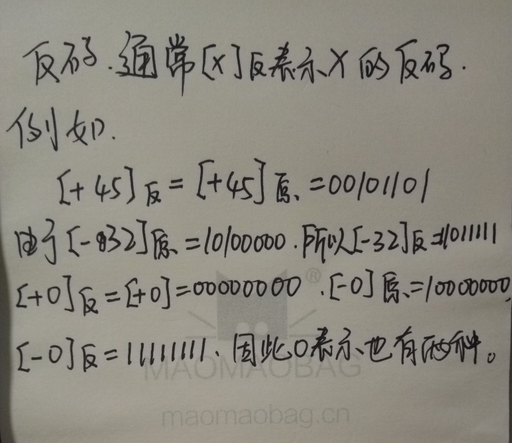
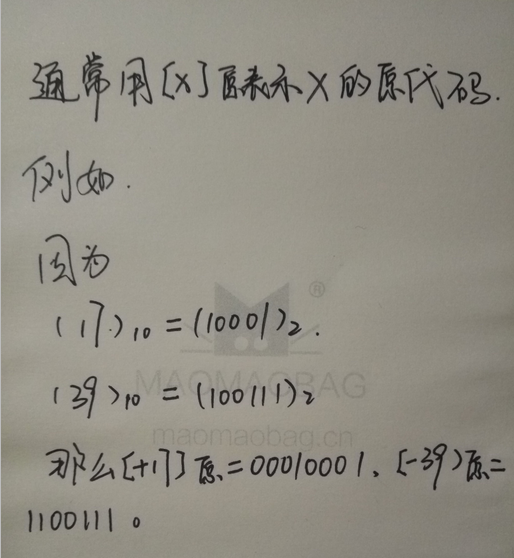
补码（two's complement），在[计算机系统](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=1695130)中，数值一律用补码来表示和存储。原因在于使用补码，可以将符号位和数值位统一处理；同时，加法和[减法](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=217486)也可以统一处理。此外，补码与[原码](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=3982703)的的相互转换，其运算过程是相同的，不需要额外的硬件电路。

一、正整数的原码、反码、补码完全一样，即符号位固定为0，数值位相同

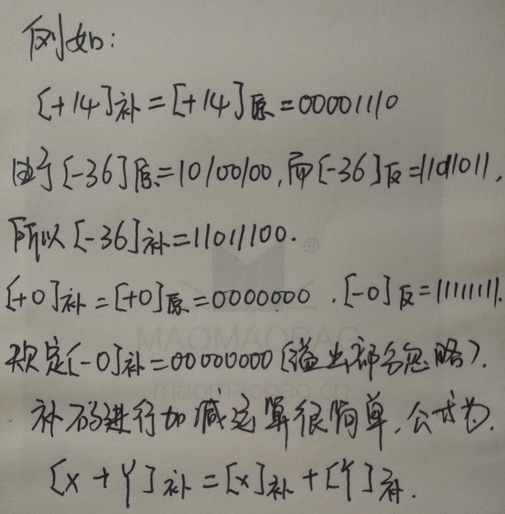
二、负整数的符号位固定为1，由原码变为补码时，规则如下：

      1、原码符号位1不变，整数的每一位二进制数位求反，得到反码

      2、反码符号位1不变，反码数值位最低位加1，得到补码



负数的补码是与其反码的最低有效位上加1。下面是举例。



**补码转化为原码**

已知一个数的补码，

⑴如果补码的符号位为“0”，表示是一个[正数](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=315382)，其原码就是补码。

⑵如果补码的符号位为“1”，表示是一个负数，那么求给定的这个补码的补码就是要求的原码。

【例4】已知一个补码为11111001，则原码是10000111（-7）。

因为符号位为“1”，表示是一个负数，所以该位不变，仍为“1”。

其余七位1111001取反后为0000110；

再加1，所以是10000111。

数值在计算机中表示形式为机器数,计算机只能识别0和1,使用的是二进制,而在日常生活中人们使用的是十进制,"正如亚里士多德早就指出的那样,今天十进制的广泛采用,只不过我们绝大多数人生来具有10个手指头这个解剖学事实的结果.尽管在历史上手指计数(5,10进制)的实践要比二或三进制计数出现的晚.

数值有正负之分,计算机就用一个数的最高位存放符号(0为正,1为负).这就是机器数的原码了.假设机器能处理的位数为8.即字长为1byte,原码能表示数值的范围为   
  
(-127~-0 +0~127)共256个.   
  
有了数值的表示方法就可以对数进行算术运算.但是很快就发现用带符号位的原码进行乘除运算时结果正确,而在加减运算的时候就出现了问题,如下: 假设字长为8bits   
  
( 1 ) 10- ( 1 )10 = ( 1 )10 + ( -1 )10 = ( 0 )10   
  
(00000001)原 + (10000001)原 = (10000010)原 = ( -2 ) 显然不正确.   
  
因为在两个整数的加法运算中是没有问题的,于是就发现问题出现在带符号位的负数身上,对除符号位外的其余各位逐位取反就产生了反码.反码的取值空间和原码相同且一一对应. 下面是反码的减法运算:   
  
( 1 )10 - ( 1 ) 10= ( 1 ) 10+ ( -1 ) 10= ( 0 )10   
  
(00000001) 反+ (11111110)反 = (11111111)反 = ( -0 ) 有问题.   
  
( 1 )10 - ( 2)10 = ( 1 )10 + ( -2 )10 = ( -1 )10   
  
(00000001) 反+ (11111101)反 = (11111110)反 = ( -1 ) 正确   
  
问题出现在(+0)和(-0)上,在人们的计算概念中零是没有正负之分的.(印度人首先将零作为标记并放入运算之中,包含有零号的印度数学和十进制计数对人类文明的贡献极大).   
  
于是就引入了补码概念. 负数的补码就是对反码加一,而正数不变,正数的原码反码补码是一样的.在补码中用(-128)代替了(-0),所以补码的表示范围为:   
  
(-128~0~127)共256个.   
  
注意:(-128)没有相对应的原码和反码, (-128) = (10000000)

补码的加减运算如下:   
( 1 ) 10- ( 1 ) 10= ( 1 )10 + ( -1 )10 = ( 0 )10   
  
(00000001)补 + (11111111)补 = (00000000)补 = ( 0 ) 正确   
10000001 11111110+1=11111111  
( 1 ) 10- ( 2) 10= ( 1 )10 + ( -2 )10 = ( -1 )10   
  
(00000001) 补+ (11111110) 补= (11111111)补 = ( -1 ) 正确   
  
所以补码的设计目的是:   
  
⑴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.   
  
⑵使减法运算转换为加法运算,进一步简化计算机中运算器的线路设计   
  
所有这些转换都是在计算机的最底层进行的，而在我们使用的汇编、C等其他高级语言中使用的都是原码。