二进制数及其他

0　　题记

在学习计算机基础或者是数字电子技术、微机原理、单片机、C 语言等课程的时候，都会讲到一个概念：二进制数。就是这个二进制数难倒了很多的英雄汉，上面罗列的这些课程我都教过，这些课程有一个共同的知识点就是二进制数，十进制数，十六进制数，还有八进制数以及他们的相互转换，而对于电子类专业的一门比较重要的课程－－单片机，更是要用到二进制数和十六进制数。故而每次我讲课时都会尽量详细的给学生讲解这些内容。今天上午刚刚结束了两个班的第一堂课，又勾起了我的回忆，让我有一种把这部分知识点写下来的冲动，给那些刚刚开始学习数字电子技术、单片机、C 语言的学生们。以上是为这篇文章的来历。

1　　引子

随着电的使用，电器应用越来越广泛，人们对于电器的要求也越来越高，要求功能强大，还要智能化，使用简单化，这些要求让数字电子技术应用范围越来越广泛，原先很多采用模拟电路的地方都被数字电路取代了，特别是对于信号处理方面，随着计算机科学与技术突飞猛进地发展，用数字电路进行信号处理的优势也更加突出。信号处理的一般方法都是先将模拟信号按比例转换成数字信号，然后送到数字电路进行处理，最后再将处理结果根据需要转换为相应的模拟信号输出。从一般的模拟信号到数字信号，要经过采样、量化、编码，最终一个连续的模拟信号波形就变成了一串离散的、只有高低电平之分“０１０１…”变化的数字信号。自然界来的，或者通过传感器转化的主要是模拟信号，那么为什么要多此一举把它们变为数字信号呢？原因有以下几点：

一、模拟信号有无穷多种可能的波形，同一个波形稍微变化就成了另一种波形，而数字信号只有两种波形（高电平和低电平），这就为信号的接收与处理提供了方便。即，数字信号易于传输，抗干扰能力强。

二、模拟信号由于它的多变性极容易受到干扰，其中包括来自信道的和电子器件的干扰，模拟器件难以保证高的精度（如放大器有饱和失真、截止失真、交越失真，集成电路难免有零点漂移）。而数字电路中有限的波形种类保证了它具有极强的抗干扰性，受扰动的波形只要不超过一定门限总能够通过一些整形电路（如斯密特门）恢复出来，从而保证了极高的准确性和可信性，而且基于门电路、集成芯片所组成的数字电路也简单可靠、维护调度方便，很适合于信息的处理。特别是计算机科学技术发展后，很多模拟电路无法实现的功能都可以在采用数字电路来实现。

而电子计算机的出现，让数字电子技术有了更广阔的发展空间，也让我们的生活更加的丰富。手机带给我们联络的方便，电脑带给我们工作和娱乐以及学习的便利，天上的飞机，路上的汽车，让我们出行更加方便，家中的空调冰箱洗衣机微波炉等让我们生活更加舒适。在手机、电脑、飞机、汽车以及家电中都有一块或者多块的微处理器在工作，而这些微处理器就是由数字电路构成的。

2　　０和１的舞蹈

2.1　二进制数的来历

电，发明出来是为人类服务的，那么电路同样的是为人们服务的，也就是说我们需要在电路中能够帮助我们做平时生活中的事情，这些事情应该是我们不用电也可以来做的。举个例子，譬如我们生活中的数，我们采用的是十进制数，今年是 2010 年，就是指从耶稣诞生之年到现在有 2010 年了，班上有 45 个人，买了一条裤子花费 368 元等等，那么在这些数字的背后隐藏着什么呢？我们都知道 1+1＝2，但是陈景润花费了很大的功夫才证明，这说明即使我们看起来很平常的的事情也隐藏着一些我们不知道的因素，在这里我们不是来讨论 1+1＝2 的，因为我们不是顶极的数学家，但关于十进制数，我们可以讨论一下一些普通人都有能力理解的东西。2010，45，368 这些数字给我们提供了 2 个信息，数码和数位，2010 由三个数码 0，1，2 构成，45 由两个数码 4，5 构成，368 由三个数码 3，6，8 构成，而且这些数码的位置不一样，那么他们所代表的大小不一样的，如图 1：

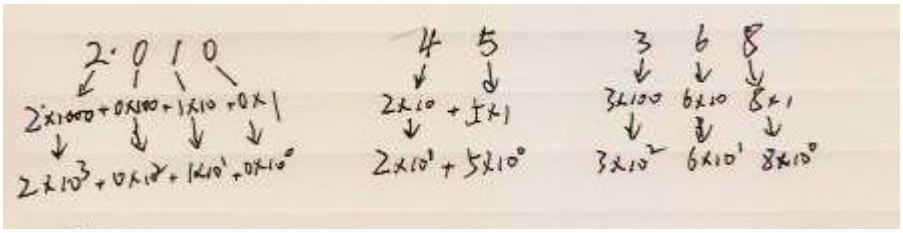


图 1 十进制数的数位、基数和权

图中 10 就是基数，而 10^3、10^2、10^1、10^0 也就是 1000，100，10，1 就是权。所谓的权，就是在这个数中占的数值大小。也就是说 2010 中的“2"代表了 2 个“千”，45 中的”4"代表了 4 个“十”，而 368 中的”8“代表了 8 个“一”，而且同一个数码放在不同的位置上就代表了不同数值，如 555 中，三个 5 的权分别 100，10，1，那么第一个 5 代表的数值就是 5X100，第二个 5 代表的数值是 5X10，的三个 5 代表的数值是 5X 1。采用这种方法，我们就可以用有限的数码来表示无限的数据了。

总结一下，十进制采用了 0，1，2，3，4，5，6，7，8，9 共 10 个数码，基数是 10，进行运算的时候，我们采用逢十进一。

这是我们现实生活中需要用到的十进制的一些情况，那么我们在数字电路中必然也要采用这种计数方法，电路中传输的就是电压和电流，我们要用 10 种不同的状态来表示这 10 个数码有点困难。我们举例来说吧，譬如有一个电压，0～5V ，那么我们就可以这样来表示 0～9 这 10 个数码，如表 1。

表 1 电压和数码之间的对应关系

电压	十进制数码	电压	十进制数码
0V	0	2.5V	5
0.5V	1	3V	6
1V	2	3.5V	7
1.5V	3	4V	8
2V	4	4.5V	9

接下来就是要制造一个能够精确的实现 0V，0.5V，1V，1.5V……4.5V 等各种电平的基本电路，但这一件是非常困难的事情。两个相邻的电平只有 0.5V，电路受到干扰，电平偏移 0.5V，那么就变成另外一个数据了，而要保证电平完全没有漂移是不可能的，所以，十进制数在电路中很难直接实现了。即使勉强实现了，数据传输的时候又遇到了更大的数据准确性的问题，因为电平经过导线传输的时候会变化，相邻的两个电平很容易混淆。这种十进制数在数字电路中是没法直接实现，更别说是在微处理器这种高频电路中实现了。这样必然要另外想办法了。而戈特弗里德·威廉·凡·莱布尼茨（Gottfried Wilhelm von Leibniz，1646 年 7 月 1 日～1716 年 11 月 14 日）在 18 世纪初提出的二进制帮助人们解决了问题，虽然莱布尼茨受中国的易经八卦启发而发明的二进制数最初不是用来设计电路的，因为那个时候人们才开始研究电的现象，电灯，电池等都还没有出现。但 20 世纪初人们制造出二极管、三极管、集成电路等的时候，却把二进制拿来用于电路的设计。二进制数因为只有两个数 0 和 1，状态也只有两种，在电路中实现起来就方便的多了，只要一个高电平和低电平就可以，甚至说有电流和无电流、有电荷和无电荷都可以表示，这样的话电路的实现非常简单，而且这种电路也不容易受到干扰，抗干扰性好的多。还是以上面 0～5V 的一个电平来说明，看图 2。

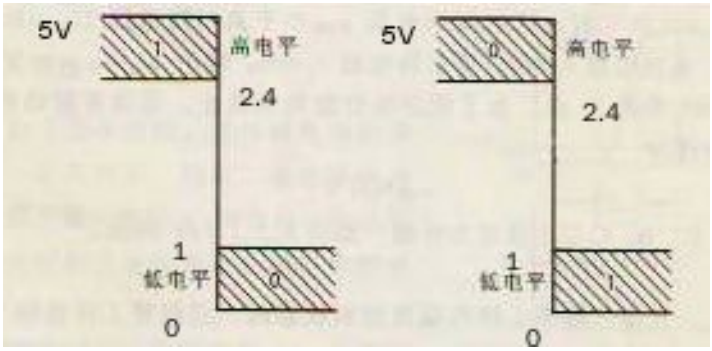


图 2　0 和 1 的电平实现

从图 2 中可以看到，我们可以认为 0～1V 都是低电平，2.4V～5V 都是高电平，若假设低电平代表 0，高电平代表 1，那么我们就实现了二进制数了，这个电路简单，而且易与实现，电平允许有一定的漂移，提高了抗干扰能力，数据传输可靠性高的多。所以数字电路中采用了二进制数。

假若以高电平代表 1，低电平代表 0，则称为正逻辑系统，反之，以高电平代表 0，低电平代表 1，则称为负逻辑系统，一般来说，我们采用正逻辑系统。

2.2 二进制数与十进制数

接下来我们就研究一下二进制数，注意了，下面我们纯粹的研究二进制数，跟二进制的数字电路实现没有任何的关系了。

借助于十进制数的思路，我们的二进制数有两个数码：0 和 1，基数是 2，进行运算的时候是逢二进一。举例来说明，比如二进制数 10110（注意，读这个数据的时候只需要把每一位数据读出来就可以了，千万不要采用十进制数的读法。即这个数读作：一 零 一 一 零，而不是一万零一百一 十，若按照十进制数的读法，会让别人笑话的。切记切记）。对于这个数，我们知道它的每一位都有权，而且权是 2 的幂，即

$10110 = 1X2^4+0X2^3+1X2^2+1X2^1+0X2^0$ 若我们把这些数字相加计算出数值来，就会发现它是一个十进制数 22，这样我们就把一个二进制数转换为十进制数了。我们接下来就讲二进制数和十进制数的相互转换问题。

随便拿出一本教材来，关于二进制数和十进制数的相互转换，都讲了一个方法：二进制数转换为十进制数采用加权法，就是上面说的例子。而十进制数转换为二进制数则分为整数部分和小数部分分别转换，整数部分用除 2 取余法，小数部分采用乘 2 取整法，然后要列竖式来求解。一般来说，我们在进行应用的时候，譬如数字电路，单片机中使用的数字都是整数，而且只需要我们快速的计算出这个数据即可，若按照除 2 取余法来求解，则太费时间，这里我讲一种方法，命名为“8421”法，可以快速的求解 255 以内的数据（超过 255 的数据建议大家用计算器来求解，手算或者心算就太费劲了。）。这个方法就是利用权，一个 4 位的二进制数，它的每一位的权恰好是 8421，如图 3。

128	64	32	16	8	4	2	1

图 3　二进制数每一位的权

接下来我们就以一个具体的例子来说明这种方法的使用。先看二进制数转换为十进制数的例子，就是上面说的 10110 吧，把它的每一位的权都标出来，如图 4。

16	8	4	2	1
1	0	1	1	0

图 4　二进制数转换为十进制数的例子

我们只要把数值是 1 的位的权加起来就可以得到对应的十进制数，即 16+4+2＝22，完全一样。但是比列式子快速的多了，如果熟悉了每一位的权之后我们都可以心算，快速的算出结果为 22。

接下来讲十进制数转换为二进制数的例子，把十进制数 55 转换为二进制数。开始运算之前先把图 3 画在草稿纸上，然后开始填 1，首先，55 在 64 和 32 之间，所以 64 处不能是 1，我们在 32 处写 1，这个 1 的权是 32，那么我们还剩下 55-32＝23，比 16 大，我们在 16 的位置上写 1，这时候我们还剩下 23-16＝7，接着我们就可以在 4、2 和 1 的位置上分别写一个 1，32+16+4+2+1 恰好等于 55，所以我们在其他的位置上写 0，把这个数写出来 110111，就得到了转换后的二进制数了。整个过程如图 5 所示。

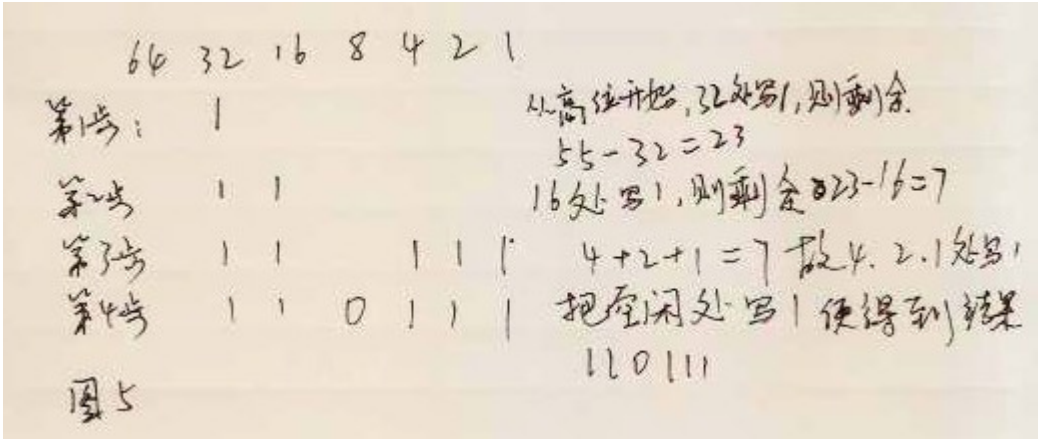


图 5　十进制数转换为二进制数

采用这种方法可以快速的实现二进制数和十进制数的相互转换，这里要提醒大家一点了，我们只需要练习十进制数 255 以内的数据和二进制数之间的相互转换就可以了，太大的数据交给计算器来运算就好了，千万不要为难自己，非要去计算 52369 的二进制数，那将让你失去许多乐趣的。相对于二进制数来说，我们只要能计算 8 位以内的二进制数转换为十进制数就可，超过 8 位的还是交给计算器吧。当然了，每个人都有自己的自由，如果某人要手工计算 32 位二进制数转换为十进制数或者把一个上亿的十进制数转换为二进制数，我也没有办法。

在现实生活中，对于十进制数，我们自动的根据数据的大小调整数位，15 有两位有效数字，那么我们写 15 就好，那么没有人会写成 00015 的，同样的 369 有三位有效数字，也没人会写成 00369，因为在数字的前面加 0 不改变大小，所以我们通常是省略前面的 0。但在数字电路中有另外一种情况，譬如我们制造好了一个电路后，能表示 8 位二进制数，那么就必须制造 8 个基本元件，每个基本元件存储一个二进制数，那么表示任何一个数，都是这 8 个基本元件作为一个整体来表示的，这样就会遇到多余的 0，如表示十进制数 30，那么就是 00011110，前面的 0 你不能省略，因为你不能说最前面的 3 个元件不存储数据了，再者，电路造好之后你也不能随便的用刀砍掉一部分。所以，在我们数字电路以及单片机课程中，一般遇到的二进制都是位数固定的，我们在写这些数据的时候一定不要省略前面的 0,那么这个固定的位数是多少呢？8 的倍数，也就是说，一般来说都是 8 位数一组，或者是 16 位，32 位，64 位，128 位等。

针对单片机中二进制位数固定这一特点，这里有几个名词：位（bit），字节（Byte），字（Word）。其中位就是二进制位，1 位就是一个二进制位，称为 1bit，简写 1b，1 字节代表 8 个二进制的位，1Byte = 8bit ，1 字代表 2 个字节，1Word = 2Byte。Byte 可以简写作 B，我们可以得到如下公式：

1B = 8b，1Word = 2B = 16b

随着计算机技术的发展，数据越来越多，我们还有几个单位，KB,MB,GB,TB,其关系为：

$$1KB = 1024B = 2^{10}B$$
$$1MB = 1024KB = 2^{20}B$$
$$1GB = 1024MB = 2^{30}B$$
$$1TB = 1024GB = 2^{40}B$$

2.3 十六进制数出世

数字电路中都用二进制数，计算机中当然也用二进制数，而我们要与这些电路打交道，必然要会二进制数，大家看看以下这几个二进制数，然后抄一遍：

第一个数：00001101

第二个数：0101001110011010

的三个数：11100101011100110011001011101111

第四个数：

1101010001111000001110110110111011100110000011100100010010011100

第一个数是 8 位，写下来没什么太大的关系，第二个数是 16 位，仔细的看一下，抄写也可以，第三个是 32

位，我想可能要非常吃力的才能写下来，也许还要多次才能正确的抄写下来，那么最后一个64位的，有人有勇气面对它吗？如果是写满了0和1的20张A4的纸呢，任是谁也会崩溃的，太苦恼了，如果每天都是看到的都是这些数字，也只有神仙才可以做得到了。这还不算，怕的就是出错了，满目都是0和1，稍微错了一位，面目全非了，所有的工作就要重新来过。有人会说，我直接转换成10进制数来读写好了。但是二进制数转换为十进制数太繁琐，谁能告诉我最后一个数对应的十进制数是多少呢？

二进制数难读，难写，数据位数多，写和读都不方便，而我们却不能不看，不能不用，因为我们不能不用数字电路，也不能不用计算机。当问题出现了，我们就要解决它，于是出现了十六进制。

十六进制有十六个数码：0，1，2，3，4，5，6，7，8，9，A，B，C，D，E，F，基数是16，运算时逢十六进一。为什么说十六进制数解决了二进制数读写困难，也没有十进制数转换繁琐的困局呢？因为十六进制和二进制数相互转换非常简单，4位二进制数对应于一位十六进制数，这样就可以把上述冗长的二进制数转换为十六进制数。上面四个二进制数转换为十六进制数为：

第一个数：0D

第二个数：539A

的三个数：E57332EF

第四个数：D4783B6EE60E449C

当你看到上面这组数据的时候，读和写的时候要轻松的多了吧，这样我们被前面二进制数打击的信心又回来了。那么你一定迫切的想知道二进制数和十六进制数是如何转换的吧。好吧，我们就来讲二进制数和十六进制数的转换。

二进制数转换为十六进制数：4位一组，分别转换；

十六进制数转换为二进制数：1位转换为4位，原序排列。

在进行学习二进制数和十六进制数转换之前，先看一个表格，

表2 十进制数、二进制数、十六进制数的对应关系

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

我们只要对照这个表格，就可以很轻松的进行二进制数和十六进制数的转换了。下面用具体的例子来说明。

例 1　把二进制数 1011 0110 转换为十六进制数

首先把二进制数分组

1011　　0110

B　　6

则二进制数 10110110 转换为十六进制数就是 B6 了。更多的位数一样的转换。

例 2 把二进制数 1110 1100 0111 0010 转换为十六进制数

把二进制数分组

1110　1100　　0111　　0010

E　　C　　7　　2

转换的结果为十六进制数 EC72

反过来，十六进制数转换为二进制数则反过来，直接一位变为 4 位就可以了。例如把十六进制数 A157 转换为二进制数，则

A　　1　　5　　7

1010　0001　0101　0111

转换后的结果就是 1010 0001 0101 0111。

正因为十六进制数和二进制数的相互转换不需要进行计算，只是简单的替换就可以，所以我们在很多场合下经常用十六进制数来代替二进制数，在学习单片机课程的时候，经常遇到十六进制数，所以必须掌握十六进制数和二进制数的相互转换，而且要能熟练的转换。针对表 2，我建议大家就用 8421 法来记忆，数字都很小，即使记不住，临时来计算也很快的。

二进制数和十六进制数能够相互转换，那么十进制数和十六进制数的相互转换怎么做呢？十进制数和十六进制数可以直接相互转换，也可以用加权法，十六进制数的每一位的权是 1，16，256，4096……数据运算量比较大，所以我们就简单的计算一下 2 位的十六进制数和十进制数的相互转换，太大的数据就不要为难自己了，用计算器吧。我的方法是先转换为二进制数，然后再把二进制数转换为十进制数，这样手算的速度要快些。反过来，要把十进制数转换为十六进制数，也是先把十进制数转换为二进制数，然后在转换为十六进制数。

我们现实生活中使用十进制数，而计算机中使用二进制数，为了读写的方便，我们发明了十六进制数，并且通过上面的学习我们也知道了如何快速的在这三种进制数据之间相互转换，应该没有上面太大的问题了。但是还有一个问题，大家再看一看表 2，二进制、十进制、十六进制数据他们的数码，你会发现数码有重合的部分，这就有问题了，如果出现了一个数据，如何知道是那种进制的数据呢？如 1110，145，562。可能有人会说，第一个数是二进制数，第二个和第三个数据是十进制数。但这是错误的。就如一件衣服 200 元，如果在重庆，那么就是 200 人民币，如果是在香港买的，那么就是 200 港元，如果你跑到英国去买，那么可能会是 200 英镑或者 200 欧元了，这可是不一样的，而且差别很大。这个时候你再来看看 1110 和 145 这两个数，到底是什么进制的数据呢？不知道，除非做了说明。这就是我要讲的另外一个问题，对于任何一个数字，我们必须作出说明是什么进制数据才有意义，否则我们不知道它的真

实大小。那么如何来区分这三种进制的数据呢？我们采用在数字的末尾加一个字母来表示。

二进制的英文单词是 Binary，十进制的英文单词是 Decimal，十六进制的英文单词是 Hexadecimal，所以我们就在二进制数后面加字母 B ，在十进制数后面加字母 D ，在十六进制数后面加字母 H，这样就可以区分这三种进制的数据了。如 1010B，145D，562H 等等。因为我们现实生活中用的最多的是十进制数，所以十进制数后的字母 D 可以省略，直接写 145，就如我们在中国买东西，标价是 200 的话默认单位就是人民币了，但二进制数和十六进制数后的字母不能省略。

2.4 负号的解决之道

以上在讨论数值的时候都只考虑了正数的情况，其实我们还使用负数以及小数，鉴于小数在我们课程的学习阶段用的不是很多，特别是 9051 单片机，对于小数的运算非常不擅长，所以也就很少用到了，自然不会讲太多。接下来我们就仅讨论负数的问题。

在数学运算中，表示一个数的正负，我们在数据的前面加上一个 正号或者负号（+/-），但是在计算机中，对于这个正负号的表示就有点问题了，计算机中只能使用 0 和 1，没法使用+ -，那么我们如何表示一个数值的正负呢？方法是用 0 和 1 来表示正负号。正常的情况下，我们用 0 来表示正号，1 来表示负号。这样，我们对于一个数值就有两部分构成，符号位和数值位，符号位用 0 和 1 来表示正负，数值位表示大小。计算机中的数值有很多，为了防止符号位和数值位不对应，我们一般把符号位和数值位作为一个整体来处理。前面我们讲过，在计算机中经常用到的单位是 Byte，有 8bit，我们就把最高位作为符号位，其他的 7 位作为数值位。如图 6。

符号位　　数值位

0　正数

1　负数

图 6　符号位和数值位

这样我们就可以用二进制数来表示负数了。如

+10＝0000 1010B

-10＝1000 1010B

这样我们就不怕负数了。我们来计算一下+10 ＋ （－10）的结果。在计算机中，+10 和-10 我们已经转化为二进制数了，这里直接列竖式相加就可以了

　　0000 1010

＋　1000 1010

　　1001 0100

为什么结果不是０？难道+10 +（－10）不等于０？答案肯定是０的，绝对是二进制数运算出错，可是错误在哪里呢？大家思考一下数学中对于两个数据相加是如何做的？首先是比较两个数的符号，如果符号相同，那么两个数值相加，符号不变，而如果两个数值符号不同，则比较一下哪个数值大，用数值较大的减去数值较小的，符号用数值较大的符号。也就是说，我们数学上计算的时候是分情况的，而在上面的式子中，我们把符号也参与运算了，并没有比较两个数的数值大小。

问题出现了，那么就要解决它。计算机的先驱者们发现，若按照我们数学上的处理方法，则实现的电路很复杂，便采取了另外一种方法，符号可以参与运算，而且结果也可以是正确的，这便是补码表示。讲补码的时候涉及到几个概念：原码，反码，补码。

原码

最高位表示符号，其他位表示数值，则这种表示方法就是原码。如

[+10]原＝0000 1010B

[-10]原＝1000 1010B

反码

对于正数，反码和原码一样。对于负数，反码就是把原码符号不变，数值为取反，如

[+10]反＝0000 1010B

[-10]反＝1111 0101B

补码

对于正数，补码和原码一样，对于负数，补码就是把反码加１，如

[+10]补＝0000 1010B

[-10]补＝1111 0110B

总结一下，对于正数，[X]补＝[X]原

负数　，[X]补＝[X]反 +1

补码费劲的得到了有水木好处呢？我们用补码的话就可以直接带符号参与运算了。还是上面的例子。

[+10]补＝0000 1010B

＋　[-10]补＝1111 0110B

---

1 0000 0000B

得到的结果是０，可能有人说了，不是最前面有个１的吗，怎么会是０呢。这里我们不要忘记了，在计算机中，所有的数据位数是固定的，我们这里举例为8bit 的例子，那么得到结果后我们也只能保存八位，你看看上面的结果，一共有８个０，计算机只能保存这８个０，最前面的１是不算在结果里的。所以，得到的结果就是正确的。

补码的运算中还有一个溢出的问题，大家可以试着用补码来计算一下 -98+（-50），你会发现得到了一个最高位是０的八位数，也就是说，变成了一个正数。这就是超出了数据范围，产生了溢出。关于溢出，因为不是重点，大家可以自己查书找到答案。

3　万物归于阴阳

《易传》记录“易有太极，始生两仪。两仪生四象，四象生八卦。"这里所说的两仪，就是阴和阳。这里所说的卦，是宇宙间的现象，是我们肉眼可以看见的现象，宇宙间共有八个基本的大现象，而宇宙间的万有、万事、万物，皆依这八个现象而变化，这就是八卦法则的起源。而八卦的来源就是阴阳。 我国古代人们发明的太极八卦用阴阳能够代表世间万物，那么由０和１组成的二进制数自然能够表示世间所有东西，而不仅仅是几个数字。也就是说我们现实生活中的图形、图像、声音、文字、色彩等等，都可以用二进制数表示，当然也可以在计算机中处理和显示出来。其实这个做到了的，否则我们今天也就不会有电脑里的图片、音乐、视频、文字等等，我们今天的世界将不会这么多姿多彩。

那么，单纯的０和１如何表示世间的万物呢？这里要讲到一个词：代码。代码，从字面意思来看，就是代替的码字，即我们找一组二进制数来代替，代替谁呢？代替世间的万物。到这里可能有人会有疑问了？既然是代替的，必然不是真的，有什么用呢？自然有用，要是没用的话我们不会随时随地的使用。其实我们就是生活在一个代码的世界里，如我们的名字就是一个代码，用汉字给我们每个人的一个代码，代表一个个体。在学校里，每个学生都有一个学号，而这个学号就是一个代码，用一组十进制数来代表一个学生。甚至我们所说的课桌，操场等等名词都是代码，用汉字来代表某个物体或者某种意义。代码到底有什么好处呢？方便于我们的沟通和交流。还是以我们的名字为例来说。如果一个人叫“张三”，那么我们有事情要找他，那么就喊“张三”，叫张三的人就答应了，于是你可以跟他交流了。合同上要双方签字，而就是签的名字，合同签完后就可以存档了，不管经过多久，其他人看到这个签字，就知道这是经过双方本人认可了的，而不需要双方两个大活人亲自告诉你说，这个合同我认可了的，因为名字就代表了其本人。如果我们不用代码，那么一个合同文本上必须有两个人站在那里，证明合同双方都同意的，这是一件和荒唐的事情，文件柜里站着两个大活人不是很滑稽和不可能的事情吗？所以，我们使用代码。注意的是代码就是代码，不是人本身，你的名字不等于你这个人本身，它仅仅代表你这个人，我们不能说几个汉字和活生生的人是一样的吧。每个人都有血有肉，有情感，但是汉字只是一些笔迹，不会有血肉。

代码，有任意性，就是我们可以用任何的东西来代表某个含义，如汉字里的“桌子”和英语里的“desk“都是代表了同一种东西。这也表示这我们可以用随意的什么来代表我们每个人，我们的名字是汉字，两个或者三个或者四个汉字，当然，我们也可以用数字来代表我们每个人，比如监狱里每个囚犯都有一个编号，这个编号就是用十进制数来给每个人的代码。虽然代码有随意性，但是我们一般不会随意的进行编写代码，而是按照某种规律来编码，因为有规律的代码使我们的维护更加方便。我们每个人的身份证就是一个代码，是很有规律的，不知道有没有注意到这个规律。

代码就是用码字来代替，我们编写代码的过程叫做编码，有时候也称代码为编码。我们可以用０和１的二进制数按照某种规律排列起来代表任何一个事物，下面讲几种常用的代码。

- 二——十进制代码

二——十进制代码就是用二进制数对十进制数编写代码，也就是说用０和１来给十进制数的10个数码０～９进行编码，也称为BCD码。接下来我们就看代码是如何进行编写的，需要多少位二进制数来进行编码。表３列举了１～４位二进制数所能进行的编码个数，从中我们可以 知道，最少需要４位二进制数来进行

编码。

表３　１～４位二进制数所能进行的编码个数

位数	１位二进制数	２位二进制数	３位二进制数	４位二进制数
代码	0 1	00 01 10 11	000 001 010 011 100 101 110 111	0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
代码数目	2　2^1	4　2^2	8　2^3	16　2^4

从表３中可以看到，有Ｎ位二进制数，那么代码的数量就是 2^N ，我们这里有０～９共计10个数，需要多少位呢？３位二进制数有８个代码，10个数不够分，４位二进制数有16个代码，还多了６个呢，我们怎么办？我们可以想，如果有10个人来你家作客，如果你恰好有10张椅子还算好说，可是如果我们的椅子不是恰好10把呢，你是提供８把椅子让２个客人站着还是提供16把椅子让椅子有空余呢？自然是提供16把椅子。多出来的６把椅子就让他空着吧。

我们在前面讲了，代码的编写具有随意性，也就是说你可以随意的编写你自己的代码，我们有16个代码，给10个数进行编码，那么有多少种编码的方案呢？数学上问题就是从16个数里面取出10个数进行全排列，计算的结果是大约有10亿种。这10亿种方案都是二——十进制代码，不过我们不可能用那么多，代码的编写虽然有随意性，但我们进行编码不是自己一个人用的，还需要和别人交流，那么编写一个有规律的和通用性的代码是必须的。理论上有无限种可能，但实际我们只使用其中的几种。那么我们常用的都是哪种代码呢？最常用的就是8421BCD码了。这种编码的每位都有一个权值，恰好与自然二进制数的前10个数据相同，即用0000（0）～1001（9）来表示十进制数的０～９，从高位到低位的权值分别是８，４，２，１，所以就称作8421BCD码。在8421BCD码中，每组二进制数各位按照加权系数展开便是它

所对应的十进制数。如 8421BCD 码的 0110 安权展开为

0110＝0X8+1X4+1X2+0X1＝6

所以 8421BCD 码 0110 表示十进制数 6。

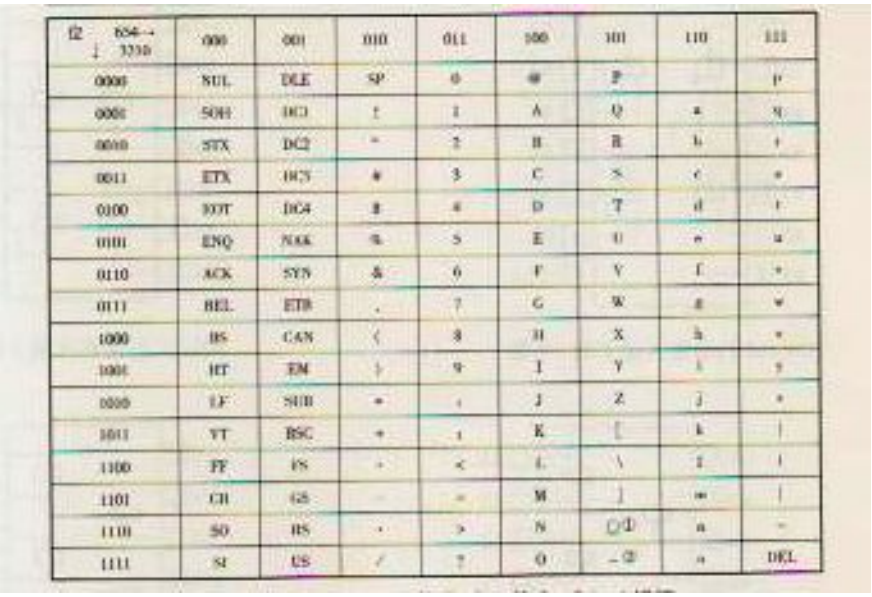
这里一定要注意代码和我们前面讲的十进制数转换为二进制数相区别，对于同一个数，两种运算结果是不一样的，例如十进制数 12，如果转换为对应的二进制数，那么结果是 1100 ，而如果转换为 8421BCD 码，那么结果为 0001 0010，也就是说，8421BCD 码就是严格的按照一位十进制数对应着 4 位二进制数来写，2 位十进制数，必然对应着 8 位二进制数，他们之间只有我们在进行 8421BCD 码编写的时候给的对应关系，12 和 0001 0010 没有数值上的任何关系。

BCD 码还有 5421 码、余 3 码等等，大家可以看看数字电子技术的教材，我不一一的讲解了。

- ASCII 码

ASCII 码（美国标准信息交换码），适用于所有的拉丁文字母，被国际标准化组织（ISO）批准为国际标准，称为 ISO646 标准。我国相应的国家标准是 GB1988-80（即《信息处理交换用的七位编码字符集》）。这里的 GB 读作“guo biao”（国标）而不是两个英文字母 ”G“ ”B“。ASCII 码 规定了信息交换用的 128 个字符。每个字符用 b7b6b5b4b3b2b1 七位来标识，通常最高位用 0 表示，使用 7 位二进制数来表示所有的大写和小写字母，数字 0 到 9、标点符号， 以及在美式英语中使用的特殊控制字符。表 4 是 7 位的 ASCII 码表。

表 4　7 位的 ASCII 码表



位 654→ ↓ 3210	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
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	□①	n	~
1111	SI	US	/	?	O	←②	o	DEL

查看上面的表，我们可以看到字母“A”的 ASCII 码为 100 0001B，最高位为 0，即 0100 0001B（41H，十进制数是 65）。

对于 ASCII 码，我们不要去记忆什么，只需要知道如何查看就好。

- 汉字编码

GB 2312 是一个简体中文字符集的中国国家标准，全称为《信息交换用汉字编码字符集基本集》，又称为 GB0，由中国国家标准总局发布，1981 年 5 月 1 日实施。GB2312 编码通行于中国大陆；新加坡等地也采用此编码。中国大陆几乎所有的中文系统和国际化的软件都支持 GB 2312。

GB 2312 标准共收录 6763 个汉字，其中一级汉字 3755 个，二级汉字 3008 个；同时，GB 2312 收录了包括拉丁字母、希腊字母、日文平假名及片假名字母、俄语西里尔字母在内的 682 个全角字符。

GB 2312 的出现，基本满足了汉字的计算机处理需要，它所收录的汉字已经覆盖中国大陆 99.75%的使用频率。

对于人名、古汉语等方面出现的罕用字，GB 2312 不能处理，这导致了后来 GBK 及 GB 18030 汉字字符集的出现。

GB 2312 中对所收汉字进行了“分区”处理，每区含有 94 个汉字/符号。这种表示方式也称为区位码。

01-09 区为特殊符号。

16-55 区为一级汉字，按拼音排序。

56-87 区为二级汉字，按部首/笔画排序。

10-15 区及 88-94 区则未有编码。

举例来说，“啊”字是 GB2312 之中的第一个汉字，它的区位码就是 1601。

对于汉字编码，我们也不需要去管它，自然有计算机帮我们处理与之相关的问题，从信息处理的角度来看，汉字处理也是非数值处理，和英文字母一样，需进行编码才能被计算机处理。

同样的，今天我们在计算机中所看到的每一样东西，包括图片、声音、视频等等都需要编码，也只有进行了编码，我们才能在计算机中进行处理。我们的计算机不仅处理数值数据，还要处理大量的非数值数据，而实际上，处理非数值数据要多的多。关于图片、声音、视频等的编码不是我今天的主题，请查阅相关的专业书籍。

后记

上周有单片机课，讲到了二进制数，课堂上比较激动，下课后考虑到 09 级的学生在这个学期数电、C 语言、单片机同时上课，对于二进制数可能会有理解上的难题，便决定把课堂上的讲课思路写下来，于是就有了这篇文章。从上周四到现在，除开中间有其他的杂事，一共用了 2 天的时间写完。主要讲解了二进制、十进制、十六进制的相互转换，带符号数的补码表示方法，编码的概念以及 8421BCD 码，ASCII 码，简单的介绍了汉字的编码。对于八进制数，进制转换时小数的处理，因为我觉得这些不是最主要的，知道了

十六进制数那么八进制数也就没什么困难了，至于小数的进制转换，原理和整数一样，在后续的学习中很少使用，所以不讲，上课的时候跟着老师听一遍就会的。补码运算，溢出没有过多的讲解，因为这部分理解上有些困难，等这个学期结束了，再来看这两个问题比较好。

我在写的时候就在不断的思考如何讲才能让学生逐步的，递进的理解，我是尽力的按照上课的时候讲课思路来写的，语言也差不多是平时的用词，没有使用很正规的语法。有时候我发觉自己的思维有些跳跃，也不知道学生们能不能看懂这篇文章，如果有任何的问题，请告诉我，我的 site: http://www.51hei.com ,欢迎提出疑问和建议。也欢迎你们把它分发到电子系其他班级同学那里，让大 家得到方便。

阿州

2010 年 9 月 12 日星期日