课程学习问题

- 课程与计划（电路/磁路；模拟电子/数字电子-数字逻辑；强电/弱电、功率电子/微电子；有限/无线、高频/低频；…）（电场/磁场/…；电子/光子/…）（计算机科学与技术？软件学院/硬件学院/…）
- 学习（方法；行业领域/学科专业；应用问题/解决方案/；信息/知识/技术/技能/能力…）
- 教材（数字逻辑，第四版，华中科技大学）
- 结课（期末60+实验20+作业20）

第　一　章

基　本　知　识

注意：

- 定义（准确、认同、…）
- 公理、定理、性质、原理、规则、类别、…

本章知识要点

- ★ 数字系统的基本概念 ；
- ★ 常用计数制及其转换 ；
- ★ 带符号二进制数的代码表示 ；
- ★ 常用的几种编码 。

掌握（数制体系/数值表示/数值转换：一般化！）

- 不同计数制之间的数值相互转换。
- 不同数值编码之间的相互转换。

1.1　概　述

1.1.1　数字系统

一、信息与数字

信息的概念：人们站在不同的角度，对“信息”给出了不同的解释。诸如，“信息是表征物理量数值特征的量”，“信息是物质的反映”，“信息是人类交流的依据”，…，

广义的说，“信息是对客观世界所存在的各种差异的描述”。

数字及符号/信息/知识

二、数字系统

什么是数字系统?（独立存在？/非数字系统关系？）

数字系统是一个能对数字信号进行存储、传递和加工的实体，它由实现各种功能的数字逻辑电路相互连接而成。例如，数字计算机。

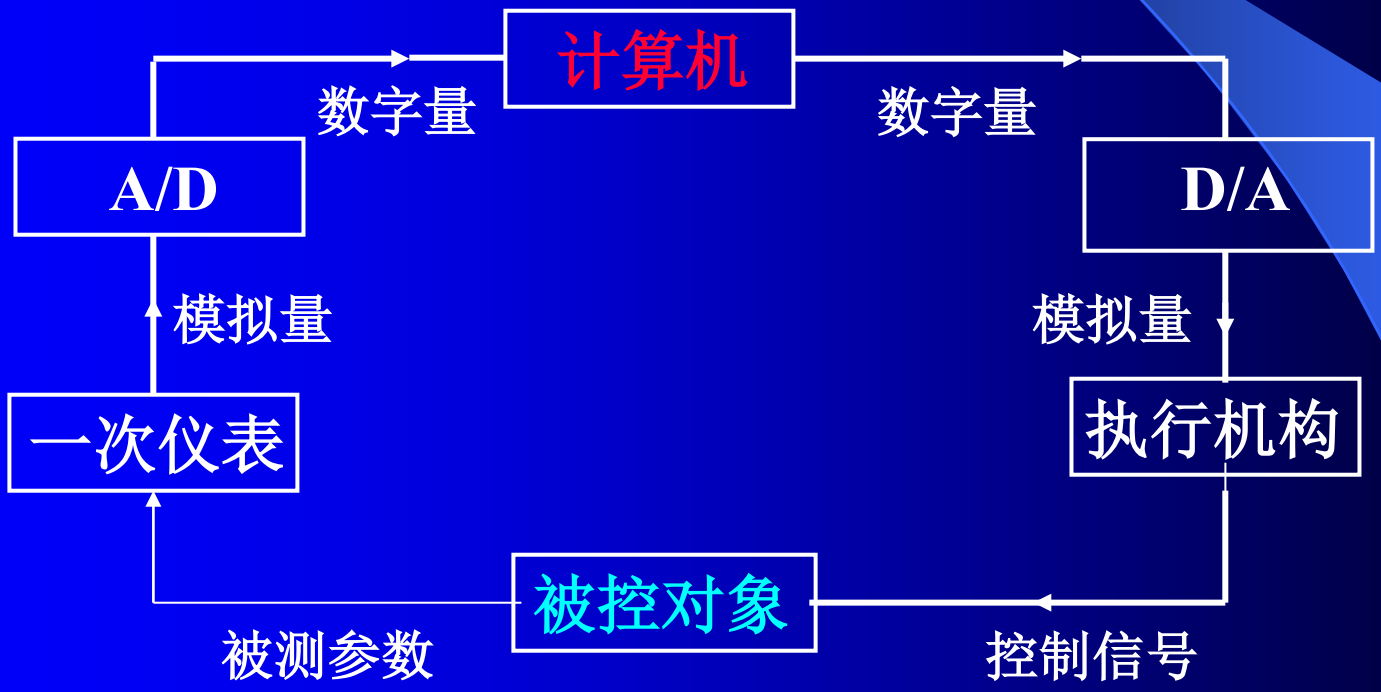
1. 数字信号

若信号的变化在时间上和数值上都是离散的，或者说断续的，则称为离散信号。离散信号的变化可以用不同的数字反映，所以又称为数字信号，简称为数字量。

例如，学生成绩记录，工厂产品统计，电路开关的状态等。

数字系统中处理的是数字信号，当数字系统要与模拟信号发生联系时，必须经过模/数(A/D)转换和数/模(D/A)转换电路，对信号类型进行变换。

例如,某控制系统框图如下图所示。



2. 数字逻辑电路（物理系统中数字信号存在？模拟信号！）

用来处理数字信号的电子线路称为**数字电路**。由于数字电路的各种功能是通过逻辑运算和逻辑判断来实现的，所以数字电路又称为**数字逻辑电路**或者**逻辑电路**。

数字逻辑电路具有如下特点：

(1) 电路的基本工作信号是二值信号。它表现为电路中电压的“高”或“低”、开关的“接通”或“断开”、晶体管的“导通”或“截止”等两种稳定的物理状态。

(2) 电路中的半导体器件一般都工作在开、关状态。

(3) 电路结构简单、功耗低、便于集成制造和系列化生产；产品价格低廉、使用方便、通用性好。

(4) 由数字逻辑电路构成的数字系统工作速度快、精度高、功能强、可靠性好。

由于数字逻辑电路具有上述特点，所以，数字逻辑电路的应用十分广泛。

随着半导体技术和工艺的发展，出现了数字集成电路，集成电路发展十分迅速。

数字集成电路按照集成度的高低可分为小规模（SSI）、中规模（MSI）、大规模（LSI）和超大规模（VLSI）几种类型。

三、 数字计算机及其发展

1. 数字计算机（模拟计算机！）

数字计算机是一种能够自动、高速、精确地完成数值计算、数据加工和控制、管理等功能的数字系统。

2. 计算机的发展

数字计算机从1946年问世以来，其发展速度是惊人的。根据组成计算机的主要元器件的不同，至今已经历了四代。具体如下表所示。

数字计算机总的发展**趋势是：速度↑、功能↑、可靠性↑、体积↓、价格↓、功耗↓。**

1.1.2 数字逻辑电路的类型和研究方法（吃过梨/种过梨吗？在吃梨和种梨的过程中慢慢体会！）（综述：述而不综=史料）

一、数字逻辑电路的类型

根据一个电路是否具有记忆功能，可将数字逻辑电路分为**组合逻辑电路**和**时序逻辑电路**两种类型。

组合逻辑电路：如果一个逻辑电路在任何时刻的稳定输出仅取决于该时刻的输入，而与电路过去的输入无关，则称为组合逻辑(Combinational Logic)电路。

由于这类电路的输出与过去的输入信号无关，所以不需要有记忆功能。

时序逻辑电路：如果一个逻辑电路在任何时刻的稳定输出不仅取决于该时刻的输入，而且与过去的输入相关，则称为时序逻辑(Sequential Logic)电路。

由于这类电路的输出与过去的输入信号有关，所以需要有记忆功能。

时序逻辑电路按照是否有统一的时钟信号进行同步，又可进一步分为**同步时序逻辑电路**和**异步时序逻辑电路。**

二、数字逻辑电路的研究方法

对数字系统中逻辑电路的研究有**两个主要任务：一是分析，二是设计**。

对一个已有的数字逻辑电路，研究它的工作性能和逻辑功能称为**逻辑分析**；

根据提出的逻辑功能，在给定条件下构造出实现预定功能的逻辑电路称为**逻辑设计**，或者**逻辑综合**。

逻辑电路分析与设计的方法随着集成电路的迅速发展在不断发生变化，最成熟的方法是**传统的方法**。

1. 逻辑电路分析和设计的传统方法

传统方法：传统方法是建立在小规模集成电路**基础**之上的，它以技术经济指标作为**评价**一个设计方案优劣的主要性能指标，设计时追求的**目标**是如何使一个电路达到**最简**。

如何达到最简呢？在组合逻辑电路设计时，尽可能使电路中的逻辑门和连线数目达到最少。而在时序逻辑电路设计时，则尽可能使电路中的触发器、逻辑门和连线数目达到最少。

注意：一个最简的方案并不等于一个最佳的方案！

最佳方案应满足全面的性能指标和实际应用要求。所以，在用传统方法求出一个实现预定功能的最简结构之后，往往要根据实际情况进行相应调整。

2．用中、大规模集成组件进行逻辑设计的方法

用中、大规模集成组件去构造满足各种功能的逻辑电路时，**如何寻求经济合理的方案呢？**要求设计人员**必须注意：**

▲ 充分了解各种器件的逻辑结构和外部特性，做到合理选择器件；

▲ 充分利用每一个已选器件的功能，用灵活多变的方法完成各类电路或功能模块的设计；

▲ 尽可能减少芯片之间的相互连线。

3．用可编程逻辑器件(PLD)进行逻辑设计的方法

各类可编程逻辑器件(PLD)的出现，给逻辑设计带来了一种全新的方法。人们不再用常规硬线连接的方法去构造电路，而是**借助丰富的计算机软件对器件进行编程烧录来实现各种逻辑功能，这给逻辑设计带来了极大的方便。**

4．用计算机进行辅助逻辑设计的方法

面对日益复杂的集成电路芯片设计和数字系统设计，人们不得不越来越多地**借助计算机进行辅助逻辑设计**。目前，已有各种设计数字系统的软件在市场上出售。计算机辅助逻辑设计方法正在不断推广和应用。不少人认为计算机设计自动化已形成计算机科学中的一个独立的学科。

1.2　数制及其转换（一般化表示/转换体现能力！）

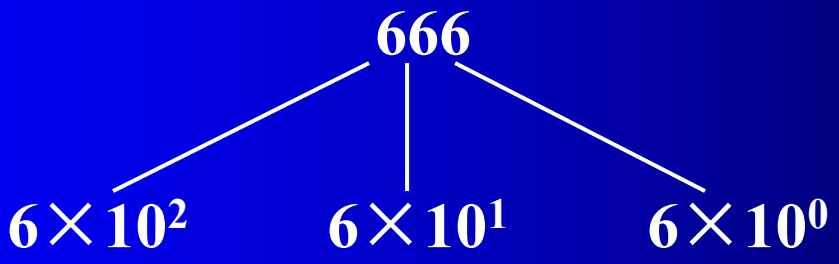
1.2.1 进位计数制

数制是人们对数量计数的一种统计规律。生活中广泛使用的是十进制，而数字系统中使用的是二进制。

一、十进制（半斤五两）

十进制中采用了0、1、…、9共十个基本数字符号，进位规律是“逢十进一”。当用若干个数字符号并在一起表示一个数时，处在不同位置的数字符号，其值的含意不同。

如



同一个字符6从左到右所代表的值依次为600、60、6。即

$$(666)_{10}=6\times10^2+6\times10^1+6\times10^0$$

二、　R进制（一般化-类）

广义地说，一种进位计数制包含着**基数**和**位权**两个基本的因素：

基数：指计数制中所用到的数字符号的个数。在基数为R计数制中，包含0、1、…、R-1共R个数字符号，进位规律是“逢R进一”。称为R进位计数制，简称R进制。

位权：是指在一种进位计数制表示的数中，用来表明不同数位上数值大小的一个固定常数。不同数位有不同的位权，某一个数位的数值等于这一位的数字符号乘上与该位对应的位权。**R进制数的位权是R的整数次幂。**

例如，十进制数的位权是10的整数次幂，其个位的位权是10^0，十位的位权是10^1……。

一个R进制数N可以有两种表示方法：

(1) 并列表示法(又称位置计数法)

$$(N)_R = (K_{n-1}K_{n-2}\cdots K_1K_0 \cdot K_{-1}K_{-2}\cdots K_{-m})_R$$

(2) 多项式表示法(又称按权展开法)

$$(N)_R = K_{n-1}\times R^{n-1} + K_{n-2}\times R^{n-2} + \ldots + K_1\times R^1 + K_0\times R^0 + K_{-1}\times R^{-1} + K_{-2}\times R^{-2} + \ldots + K_{-m}\times R^{-m}$$

$$= \sum_{i=-m}^{n-1} K_i R^i$$

其中：**R**—— 基数；　**n**——整数部分的位数；

m—— 小数部分的位数；

K_i —— R进制中的一个数字符号，其取值范围为　$0 \le K_i \le R-1$ $(-m \le i \le n-1)$。

R进制的特点可归纳如下：

(1) 有0、1、…、R-1共R个数字符号；

(2) “逢R进一”；

(3) 位权是R的整数次幂，第i位的权为R^i $(-m \le i \le n-1)$。

三、二进制（半斤一两）（具体化-实例/对象）

基数R=2的进位计数制称为二进制。二进制数中只有**0**和**1**两个基本数字符号，进位规律是“**逢二进一**”。二进制数的位权是2的整数次幂。

任意一个二进制数N可以表示成

$$(N)_2=(K_{n-1}K_{n-2}\cdots K_1K_0.K_{-1}K_{-2}\cdots K_{-m})_2$$

$$=K_{n-1}\times 2^{n-1}+K_{n-2}\times 2^{n-2}+\cdots+K_1\times 2^1+K_0\times 2^0$$

$$+K_{-1}\times 2^{-1}+K_{-2}\times 2^{-2}+\cdots+K_{-m}\times 2^{-m}$$

$$=\sum_{i=-m}^{n-1}K_i2^i$$

其中：n—整数位数；m—小数位数；

K_i—为0或者1，　$-m\le i\le n-1$。

例如，一个二进制数1011.01可以表示成：

$(1011.01)_2 = 1\times2^3+0\times2^2+1\times2^1+1\times2^0+0\times2^{-1}+1\times2^{-2}$

二进制数的运算规则如下：

加法规则	0+0=0	0+1=1
	1+0=1	1+1=0（进位为1）
减法规则	0−0=0	1−0=1
	1−1=0	0−1=1（借位为1）
乘法规则	0×0=0	0×1=0
	1×0=0	1×1=1
除法规则	0÷1=0	1÷1=1

二进制的优点：运算简单、物理实现容易、存储和传送方便、可靠。

因为二进制中只有0和1两个数字符号，可以用电子器件的两种不同状态来表示一位二进制数。例如，可以用晶体管的截止和导通表示1和0，或者用电平的高和低表示1和0等。所以，**在数字系统中普遍采用二进制。**

二进制的缺点：数的位数太长且字符单调，使得书写、记忆和阅读不方便。

因此，人们在进行指令书写、程序输入和输出等工作时，**通常采用八进制数和十六进制数作为二进制数的缩写。**

四、八进制（半斤四两）

基数R=8的进位计数制称为八进制。八进制数中有0、1、…、7共8个基本数字符号，进位规律是“逢八进一”。八进制数的位权是8的整数次幂。

任意一个八进制数N可以表示成

$$(N)_8=(K_{n-1}K_{n-2}\cdots K_1K_0\,.\,K_{-1}K_{-2}\cdots K_{-m})_8$$

$$=K_{n-1}\times 8^{n-1}+K_{n-2}\times 8^{n-2}+\cdots+K_1\times 8^1+K_0\times 8^0$$

$$+K_{-1}\times 8^{-1}+K_{-2}\times 8^{-2}+\cdots+K_{-m}\times 8^{-m}$$

$$=\sum_{i=-m}^{n-1}K_i8^i$$

其中：n—整数位数；m—小数位数；

K_i—0～7中的任何一个字符，$-m\leqslant i\leqslant n-1$

五、十六进制（半斤八两）

基数R=16的进位计数制称为十六进制。十六进制数中有0、1、…、9、A、B、C、D、E、F共16个数字符号，其中，A～F分别表示十进制数的10～15。进位规律为“逢十六进一”。十六进制数的位权是16的整数次幂。

任意一个十六进制数N可以表示成

$$(N)_{16} = (K_{n-1}K_{n-2}\ldots K_1K_0 . K_{-1}K_{-2}\ldots K_{-m})_{16}$$

$$= K_{n-1}\times 16^{n-1}+K_{n-2}\times 16^{n-2}+\ldots+K_1\times 16^1+K_0\times 16^0$$

$$+K_{-1}\times 16^{-1}+K_{-2}\times 16^{-2}+\ldots+K_{-m}\times 16^{-m}$$

$$= \sum_{i=-m}^{n-1} K_i 16^i$$

其中：n—整数位数；m—小数位数；K_i—表示0～9、A～F中的任何一个字符，$-m \leqslant i \leqslant n-1$。

1.2.2　数制转换（物理上的等价/等效）

数制转换是指将一个数从一种进位制转换成另一种进位制。从实际应用出发，要求掌握二进制数与十进制数、八进制数和十六进制数之间的相互转换。

一、二进制数与十进制数之间的转换

1．二进制数转换为十进制数

方法：多项式替代法

将二进制数表示成按权展开式，并按十进制运算法则进行计算，所得结果即为该数对应的十进制数。

例如，$(10110.101)_2=(?)_{10}$

$$(10110.101)_2=1\times2^4+1\times2^2+1\times2^1+1\times2^{-1}+1\times2^{-3}$$

$$=16+4+2+0.5+0.125$$

$$=(22.625)_{10}$$

2．十进制数转换为二进制数

方法：基数乘除法

十进制数转换成二进制数时，应对整数和小数分别进行处理。

整数转换——采用“**除2取余**”的方法；

小数转换——采用“**乘2取整**”的方法。

（1）整数转换

“**除2取余**”法：将十进制整数N除以2，取余数计为K_0；再将所得商除以2，取余数记为K_1；……。依此类推，直至商为0，取余数计为K_{n-1}为止。即可得到与N对应的n位二进制整数$K_{n-1}\ldots K_1K_0$。

例如，（35）$_{10}$ =（？）$_{2}$

2	3	5		余数	
2	1	7	.....	1 (K_0)	低位
2		8	.....	1 (K_1)	
2		4	.....	0 (K_2)	
2		2	.....	0 (K_3)	
2		1	.....	0 (K_4)	
		0	.....	1 (K_5)	高位

即 $(35)_{10}=(100011)_{2}$

(2) 小数转换

“乘2取整”法：将十进制小数 N 乘以2，取积的整数记为K_{-1}；再将积的小数乘以2，取整数记为K_{-2}；……。依此类推，直至其小数为0或达到**规定精度**要求，取整数记作K_{-m}为止。即可得到与 N 对应的m位二进制小数$0.K_{-1}K_{-2}\ldots K_{-m}$。

例如，$(0.6875)_{10}=(?)_{2}$

	整数部分	
		0.6875
		× 2
高位	1 (K_{-1})……	1.3750
		× 2
	0 (K_{-2})……	0.7500
		× 2
	1 (K_{-3})……	1.5000
		× 2
低位	1 (K_{-4})……	1.0000

即：$(0.6875)_{10}=(0.1011)_{2}$

二、二进制数与八进制数、十六进制数之间的转换

1. 二进制数与八进制数之间的转换

二进制数转换成八进制数：以小数点为界，分别往高、往低每3位为一组，最后不足3位时用0补充，然后写出每组对应的八进制字符，即为相应八进制数。

例如，$(11100101.01)_2 = (?)_8$

011 100 101 . 010

↓ ↓ ↓ ↓

3 4 5 . 2

即 $(11100101.01)_2=(345.2)_8$

八进制数转换成二进制数时，只需将每位八进制数用3位二进制数表示，小数点位置保持不变。

例如，$(56.7)_8 = (?)_2$

5　6　.　7

↓　↓　↓

101　110　.111

即：$(56.7)_8 = (101110.111)_2$

2．二进制数与十六进制数之间的转换

二进制数转换成十六进制数：以小数点为界，分别往高、往低每4位为一组，最后不足4位时用0补充，然后写出每组对应的十六进制字符即可。

例如，$(101110.011)_2 = (?)_{16}$

0010　1110　.　0110

↓　　↓　　↓

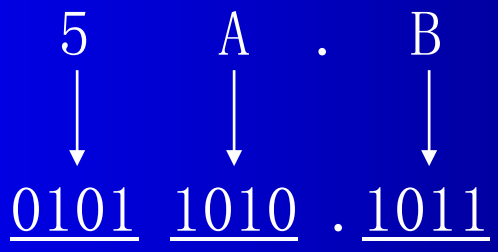
2　　E　.　6

即：

$(101110.011)_2 = (2E.6)_{16}$

十六进制数转换成二进制数时，只需将每位十六进制数用4位二进制数表示，小数点位置保持不变。

例如， $(5A.B)_{16} = (?)_2$



即：　$(5A.B)_{16}=(1011010.1011)_2$

1.3　带符号二进制数的代码表示（0/1表示）

为了标记一个数的正负，人们通常在一个数的前面用“+”号表示正数，用“–”号表示负数。在数字系统中，符号和数值一样是用0和1来表示的，一般将数的最高位作为符号位，用**0表示正，用1表示负**。其格式为

$$X_f \quad X_{n-1} \quad X_{n-2} \quad \cdots \quad X_1 \quad X_0$$

↑

符号位

通常将用“+”、“–”表示正、负的二进制数称为符号数的**真值**，而把将符号和数值一起编码表示的**二进制数**称为**机器数**或**机器码**。

常用的机器码有**原码**、**反码**和**补码**三种。

1.3.1 原码

原码：符号位用0表示正，1表示负；数值位保持不变。原码表示法又称为符号—数值表示法。

一、小数原码的定义

设二进制小数$X = \pm 0.x_{-1}x_{-2}\cdots x_{-m}$，则其原码定义为

$$[X]_{原} = \begin{cases} X & 0 \leqslant X < 1 \\ 1-X & -1 < X \leqslant 0 \end{cases}$$

即

- 符号位
  - 0　正
  - 1　负
- 数值位：不变

例如，若 $X_1 = +0.1011$ ， $X_2 = -0.1011$

则 $[X_1]_{原} = 0.1011$

$[X_2]_{原} = 1-(-0.1011)=1.1011$

根据定义，小数“0”的原码可以表示成0.0...0或1.0...0。

二、整数原码的定义

设二进制整数 $X = \pm x_{n-1}x_{n-2}\cdots x_0$，则其原码定义为

$$[X]_{原} = \begin{cases} X & 0 \leqslant X < 2^n \\ 2^n - X & -2^n < X \leqslant 0 \end{cases}$$

例如，若$X_1 = +1101$，$X_2 = -1101$，则X_1和X_2的原码为

$$[X_1]_{原} = 01101$$

$$[X_2]_{原} = 2^4 - (-1101) = 10000 + 1101 = 11101$$

同样，整数“0”的原码也有两种形式，即00...0和10...0。

原码的**优点：** 简单易懂,求取方便;

缺点： 加、减运算不方便。

当进行两数加、减运算时，要根据运算及参加运算的两个数的符号来确定是加还是减；如果是做减法，还需根据两数的大小确定被减数和减数，以及运算结果的符号。显然，这将增加运算的复杂性。

为了克服原码的缺点，引入了反码和补码。

1.3.2 反码

带符号二进制数的反码表示：

符号位———用0表示正，用1表示负；

数值位———正数反码的数值位和真值的数值位相同；而负数反码的数值位是真值的数值位按位变反。

一、小数反码的定义

设二进制小数$X = \pm 0.x_{-1}x_{-2}\ldots x_{-m}$，则其反码定义为

$$[X]_{反} = \begin{cases} X & 0 \leqslant X < 1 \\ (2-2^{-m})+X & -1 < X \leqslant 0 \end{cases}$$

例如，若 $X_1 = +0.1011$ ， $X_2 = -0.1011$，则X_1和X_2的反码为

$$[X_1]_{反} = 0.1011$$

$$[X_2]_{反} = 2-2^{-4}+X_2=10.0000-0.0001-0.1011=1.0100$$

即

−0	.	1	0	1	1
↓		↓	↓	↓	↓
1	.	0	1	0	0

根据定义，小数“0”的反码有两种表示形式，即0.0...0和1.1...1。

二、整数反码的定义

设二进制整数$X = \pm x_{n-1}x_{n-2}\cdots x_0$，则其反码定义为

$$[X]_{反} = \begin{cases} X & 0 \leqslant X < 2^n \\ (2^{n+1}-1)+X & -2^n < X \leqslant 0 \end{cases}$$

例如，若$X_1 = +1001$，$X_2 = -1001$，则X_1和X_2的反码为

$$[X_1]_{反} = 01001$$

$$[X_2]_{反} = (2^5-1)+X = (100000-1)+(-1001)$$

$$= 11111-1001 = 10110$$

即

−	1	0	0	1
↓	↓	↓	↓	↓
1	0	1	1	0

整数“0”的反码也有两种形式，即00...0和11...1。

采用反码进行加、减运算时，无论进行两数相加还是两数相减，均可通过加法实现。

加、减运算规则如下：

$$[X_1 + X_2]_{反} = [X_1]_{反} + [X_2]_{反}$$

$$[X_1 - X_2]_{反} = [X_1]_{反} + [-X_2]_{反}$$

运算时，符号位和数值位一样参加运算。当符号位有进位产生时，应将进位加到运算结果的最低位，才能得到最后结果。

1.3.3 补码

带符号二进制数的补码表示：

符号位——用0表示正，用1表示负；

数值位——正数补码的数值位与真值相同；负数补码的数值位是真值的数值位按位变反，并在最低位加1。

一、小数补码的定义

设二进制小数$X = \pm 0.x_{-1}x_{-2}\cdots x_{-m}$，则其补码定义为

$$[X]_{补} = \begin{cases} X & 0 \leqslant X < 1 \\ 2+X & -1 \leqslant X < 0 \end{cases}$$

例如，若$X_1 = +0.1011$，$X_2 = -0.1011$，则X_1和X_2的补码为

$$[X_1]_{补} = 0.1011$$

$$[X_2]_{补} = 2 + X = 10.0000 - 0.1011$$

$$= 1.0101$$

即

$$\begin{array}{rcccccc} & -0 & . & 1 & 0 & 1 & 1 \\ & \downarrow & & \downarrow & \downarrow & \downarrow & \downarrow \\ & 1 & . & 0 & 1 & 0 & 0 \\ + & & & & & & 1 \\ \hline & 1 & . & 0 & 1 & 0 & 1 \end{array}$$

注意：小数“0”的补码只有一种表示形式，即0.0...0。

二、整数补码的定义

设二进制整数X = ± $x_{n-1}x_{n-2}\cdots x_0$，则其补码定义为

$$[X]_{补} = \begin{cases} X & 0 \leqslant X < 2^n \\ 2^{n+1}+X & -2^n \leqslant X < 0 \end{cases}$$

例如，若X_1 = +1010 ， X_2 = -1010，则X_1和X_2的补码为

$[X_1]_{补}$= 01010（正数补码的数值位与真值相同。）

$[X_2]_{补}$= 2^5 + X = 100000-1010

= 10110（负数补码的数值位是真值的数值位按位变反，并在最低位加1。）

整数“0”的补码也只有一种表示形式，即00...0。

采用补码进行加、减运算时，可以将加、减运算均通过加法实现。

运算规则如下：

$$[X_1 + X_2]_{补} = [X_1]_{补} + [X_2]_{补}$$

$$[X_1 - X_2]_{补} = [X_1]_{补} + [-X_2]_{补}$$

运算时，符号位和数值位一样参加运算，若符号位有进位产生，则应将进位丢掉后才能得到正确结果。

+A-B=(+A)+(-B)；(+A-B)、(+A)、(-B)都用补码表示！

1.4　几种常用的编码（工程：协议/标准！）

1.4.1　十进制数的二进制编码（BCD码）

用4位二进制代码对十进制数字符号进行编码，简称为二–十进制代码，或称BCD(Binary Coded Decimal)码。

BCD码既有二进制的形式，又有十进制的特点。常用的BCD码有**8421码、2421码**和**余3码。**

十进制数字符号**0~9**与**8421码**、**2421码**和**余3码**的对应关系如下表所示。

常用的3种BCD码

十进制字符	8421码	2421码	余3码
0	0000	0000	0011
1	0001	0001	0100
2	0010	0010	0101
3	0011	0011	0110
4	0100	0100	0111
5	0101	1011	1000
6	0110	1100	1001
7	0111	1101	1010
8	1000	1110	1011
9	1001	1111	1100

一、8421码

8421码：是用4位二进制码表示一位十进制字符的一种**有权码**，4位二进制码从高位至低位的权依次为2^3、2^2、2^1、2^0，**即为8、4、2、1，故称为8421码。**

按8421码编码的0～9与用4位二进制数表示的0～9完全一样。所以，8421码是一种人机联系时广泛使用的中间形式。

注意：

(1) 8421码中不允许出现1010～1111六种组合(因为没有十进制数字符号与其对应)。

(2) 十进制数字符号的8421码与相应ASCII码的低四位相同，这一特点有利于简化输入输出过程中BCD码与字符代码的转换。

1．8421码与十进制数之间的转换

8421码与十进制数之间的转换是**按位进行**的，即十进制数的每一位与4位二进制编码对应。例如，

$$(258)_{10} = (0010\ \ 0101\ \ 1000)_{8421码}$$

$$(0001\ \ 0010\ \ 0000\ \ 1000)_{8421码} = (1208)_{10}$$

2．8421码与二进制的区别

例如，

$$(28)_{10} = (11100)_{2} = (00101000)_{8421}$$

二、2421码

2421码：是用4位二进制码表示一位十进制字符的另一种有权码，**4位二进制码从高位至低位的权依次为2、4、2、1，故称为2421码。**

若一个十进制字符X的2421码为$a_3\ a_2\ a_1\ a_0$，则该字符的值为

$$X = 2a_3 + 4a_2 + 2a_1 + 1a_0$$

例如，$(1101)_{2421码} = (7)_{10}$

1．2421码与十进制数之间的转换

2421码与十进制数之间的转换同样是按位进行的，例如，

$$(258)_{10} = (0010\ \ 1011\ \ 1110)_{2421码}$$

$$(0010\ \ 0001\ \ 1110\ \ 1011)_{2421码} = (2185)_{10}$$

2．注意

（1）2421码不具备单值性。例如，0101和1011都对应十进制数字5。为了与十进制字符一一对应，2421码不允许出现0101～1010的6种状态。

（2）2421码是一种对9的自补代码。即一个数的2421码只要自身按位变反，便可得到该数对9的补数的2421码。例如，

$$(4)_{10} \longrightarrow (0100)_{2421}$$

$$\Downarrow$$

$$(1011)_{2421} \longrightarrow (5)_{10}$$

具有这一特征的BCD码可给运算带来方便，因为直接对BCD码进行运算时，可利用其对9的补数将减法运算转化为加法运算。

（3）应与二进制数进行区别！

三、余3码

余3码：是由8421码加上0011形成的一种无权码，由于它的每个字符编码比相应8421码多3，故称为余3码。

例如，十进制字符5的余3码等于5的8421码0101加上0011，即为1000。

注意：

1. 余3码中不允许出现0000、0001、0010、1101、1110和1111六种状态。

2. 余3码与十进制数进行转换时，每位十进制数字的编码都应余3。例如，

$$(256)_{10} = (0101\ \ 1000\ \ 1001)_{余3码}$$

$$(1000\ \ 1001\ \ 1001\ \ 1011)_{余3码} = (5668)_{10}$$

3. 余3码是一种对9的自补代码。

1.4.2 可靠性编码

作用：提高系统的可靠性。

为了减少或者发现代码在形成和传送过程中都可能发生的错误。形成了各种编码方法。下面，介绍两种常用的可靠性编码。

一、格雷(Gray)码

1. 特点：任意两个相邻的数，其格雷码仅有一位不同。

2. 作用：避免代码形成或者变换过程中产生的错误。

3. 典型格雷码与普通二进制码之间的转换。

转换规则如下：

设二进制码为 $B = B_{n-1}B_{n-2} \cdots B_{i+1}B_i \cdots B_1B_0$

对应格雷码为 $G = G_{n-1}G_{n-2} \cdots G_{i+1}G_i \cdots G_1G_0$

有：

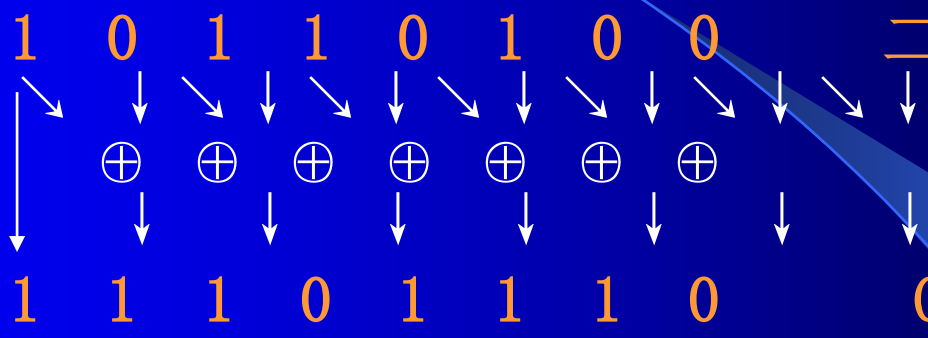
$$\begin{cases} G_{n-1} = B_{n-1} \\ G_i = B_{i+1} \oplus B_i \qquad 0 \leqslant i \leqslant n-2 \end{cases}$$

其中，运算“⊕”称为“异或”运算，运算规则是：

0⊕0=0；　0⊕1=1；

1⊕0=1；　1⊕1=0。

例如，



思考：如何将Gray码转换成二进制码？

二、奇偶检验码

奇偶检验码是一种用来检验代码在传送过程中是否产生错误的代码。

1．组成：

两部分组成：
- 信息位——位数不限的一组二进制代码
- 奇偶检验位——仅有一位。

2．编码方式：有两种编码方式.

奇检验：使信息位和检验位中“1”的个数共计为奇数；

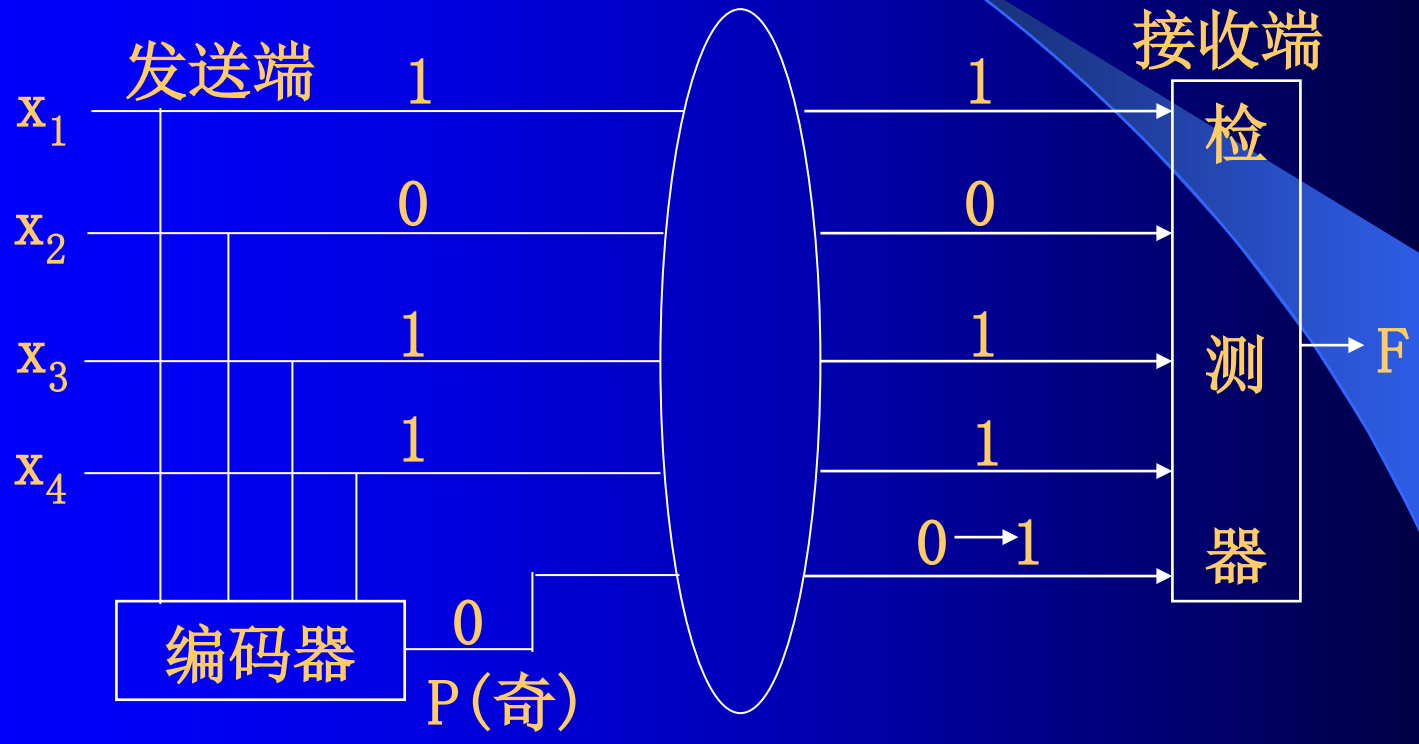
偶检验：使信息位和检验位中“1”的个数共计为偶数。

例如，

信息位（7位）	采用奇检验的检验位（1位）	采用偶检验的检验位（1位）
1001100	0	1

3．检验码的工作原理

奇偶检验码的工作原理如下图所示。



4．特点

(1) 编码简单、容易实现 ；

(2) 奇偶检验码只有检错能力，没有纠错能力 ；

(3) 只能发现单错，不能发现双错 。

1.4.3 字符编码

数字系统中处理的数据除了数字之外，还有字母、运算符号、标点符号以及其他特殊符号,人们将这些符号统称为字符。所有字符在数字系统中必须用二进制编码表示，通常将其称为字符编码。

最常用的字符编码是美国信息交换标准码，简称ASCII码(American Standard Code for Information Interchange)。ASCII码用7位二进制码表示128种字符，由于数字系统中实际是用一个字节表示一个字符，所以使用ASCII码时，通常在最左边增加一位奇偶检验位。