1.2 数制与编码

1.2.1 数制及其相互转换

任意数制的特点、表示法、运算规则及相互关系

各种进位计数制的相互转换

多项式替代法，基数乘除法，桥梁法

直接转换法，小数位置的确定

1.2.2 数的表示及其运算

小数点的表示，带符号数的表示及其运算

1.2.3 十进制数的代码表示及其运算

1.2.4 可靠性编码

1.2.1 数制及其相互转换

所谓“数制”，即各种进位计数制

- 在日常生活中通常采用的是十进制计数制，
  计数规则“逢十进一”，
  例：0,1,2,3,4,5,6,7,8,9,10,11, 12,⋯,99,100, ⋯,;

- 在计算机中多用的是二进制计数制，
  计数规则“逢二进一”

例：0,1,10,11,100,101,110,111,⋯。

1. 进位计数制---十进位计数制的特点、表示法

1）特点：(1) 10个有序的数字符号：0,1,2,3,4,5,6,7,8,9

(2) 小数点符号：“.”，“+”，“-”

(3)“逢十进一”的计数规则

其中：“十”为进位基数，

简称基数。

2）表示法：并列表示法

多项式表示

①并列表示法

	万位	千位	百位	十位	个位		十分位	百分位	千分位	万分位	十万分位
	10^4	10^3	10^2	10^1	10^0		10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
例：十进制数	1	2	3	4	5	.	6	7	8	0	9

小数点

如上所示，处在不同**位置**的数字具有不同的“*权(Weight)*”，**并列表示法**，也称**位置计数法**。

②多项式表示法

将并列式按“**权**”展开为**按权展开式**，称为多项式表示法。如下例：

$$12345.67809 = 1\times10^4 + 2\times10^3 + 3\times10^2 + 4\times10^1 + 5\times10^0$$
$$+ 6\times10^{-1} + 7\times10^{-2} + 8\times10^{-3} + 0\times10^{-4} + 9\times10^{-5}$$

由此推出，任意一个十进制数 N 可以表示成：

①并列表示法：

$$(K_{n-1}K_{n-2}\cdots K_{1}K_{0}.K_{-1}K_{-2}\cdots K_{-m})_{10} \quad (0 \leqslant K_i \leqslant 9)$$

②多项式表示法

$$\begin{aligned}(K_{n-1}\times 10^{n-1} + K_{n-2}\times 10^{n-2} + \cdots + K_{1}\times 10^{1} \\ + K_{0}\times 10^{0} + K_{-1}\times 10^{-1} + K_{-2}\times 10^{-2} + \cdots \\ + K_{-m}\times 10^{-m})_{10} = (\sum_{i=-m}^{n-1} K_i \times 10^{i})_{10}\end{aligned}$$

$$(0 \leqslant K_i \leqslant 9)$$

我们将这两种表示法记作： N_{10}，读作：N的十进制表示。

如：N=8 时，8_{10}读作：8 的十进制表示，即：$(8)_{10}$ 。同时：

$(N)_{10}$，读作：十进制的N(表示)。$(8)_{10}$读作：十进制的8(表示)。

进位计数制---任意进位计数制的特点、表示法和运算规则

对于一个R进制的数，R为任意的十进制数，有：

1）特点：

①R个有序的数字符号：0、1、 …、R-1；

② 小数点符号：“.”，“+”，“-”

③ “逢R进一”的计数规则

其中：“R”为进位基数或基数。

例：R=2，二进制，数字符号有0、1，逢二进一；

R=10，十进制，数字符号有0,1,2,3,4,5,6,7,8,9，逢十进一；

… …

2）表示法

①并列表示法

$$(A_{n-1}A_{n-2}\cdots A_1A_0.A_{-1}A_{-2}\cdots A_{-m})_R \quad (0\leqslant A_i\leqslant R-1)$$

②多项式表示法

$$(A_{n-1}\times 10^{n-1}+A_{n-2}\times 10^{n-2}+\cdots+A_1\times 10^1+A_0\times 10^0$$

$$+A_{-1}\times 10^{-1}+A_{-2}\times 10^{-2}+\cdots+A_{-m}\times 10^{-m})_R=(\sum_{i=-m}^{n-1}A_i\times 10^i)_R$$

其中：n 整数位数， m 小数位数，$0\leqslant A_i\leqslant R-1$， R为进位基数。用十进制表示，当R=10时，则括号及括号外的基数R可以省略。

我们将这两种表示法记作： N_R，读作：N的R进制表示。

如：N=8，R=2 时，8_2 读作：8 的二进制表示，即：$(1000)_2$

不同进位计数制的数值具有等值关系：

$$N = N_{10} = (N)_{10} = N_{(R)} = (N)_R$$

（N为十进制数）

参见书 P5表1.1（下页）。如可查表得：

$$13 = (13)_{10} = 13_8 = (15)_8 = (1101)_2$$

3） 运算规则：

在R进制中，相应的运算（+，-，X，/）规则与十进制一样。 必须特别注意：逢R进一。

R=10	R=2	R=3	R=4	R=8	R=16
0	0	0	0	0	0
1	1	1	1	1	1
2	10	2	2	2	2
3	11	10	3	3	3
4	100	11	10	4	4
5	101	12	11	5	5
6	110	20	12	6	6
7	111	21	13	7	7
8	1000	22	20	10	8
9	1001	100	21	11	9
10	1010	101	22	12	A
11	1011	102	23	13	B
12	1100	110	30	14	C
13	1101	111	31	15	D
14	1110	112	32	16	E
15	1111	120	33	17	F
16	10000	121	100	20	10
17	10001	122	101	21	11
…	…	…	…	…	…

特例：二进制数为计算机运算的基础，特予以关注：

① 运算规则：＋、－、×、÷

（×、÷运算可以由+、－运算来实现）

加法规则：0＋0＝0　0＋1＝1＋0＝1　1＋1＝10

乘法规则：0×0＝0　0×1＝1×0＝0　1×1＝1

减法规则：0－0＝0　1－0＝1　1－1＝0　10－1＝1(借位)

除法规则：0÷1＝0　1÷1＝1　（0不能作除数）

举例：

(1) 1010＋0110＝10000：

$$\begin{array}{r} 1010 \\ +\ 0110 \\ \hline 10000 \end{array}$$

(2) 1010 － 0110＝0100　练习

$$\begin{array}{r} 1010 \\ -\ 0110 \\ \hline 0100 \end{array}$$

练习

(3) $1010 \times 11 = 1010 \times (10+1) = 1010 \times 10 + 1010 \times 1$

$= 10100 + 1010 = 11110$

1. 乘法用加法实现

2. 长乘法：

$$\begin{array}{r} 1010 \\ \times \quad 11 \\ \hline 1010 \\ 1010 \\ \hline 11110 \end{array}$$

② 常用的二进制常数要记住。

i	R^i	i	R^i	i	R^i
-7	0.0078125	0	1	7	128
-6	0.015625	1	2	8	256
-5	0.03125	2	4	9	512
-4	0.0625	3	8	10	1024
-3	0.125	4	16	11	2048
-2	0.25	5	32	12	4096
-1	0.5	6	64	13	8192

2. 进位计数制的相互转换

数值转换： $N_\alpha \rightarrow N_\beta$

转换是等值的：即 $N_\alpha = N_\beta$

1） 多项式替代法

要点：在β进制下完成 $N_\alpha \rightarrow N_\beta$ 的转换

多项式替代法的转换步骤，归纳如下，

$$N_\alpha = (A_{n-1}A_{n-2}\cdots A_1A_0.A_{-1}A_{-2}\cdots A_{-m})_\alpha$$

$$= (A_{n-1}\times 10^{n-1} + A_{n-2}\times 10^{n-2} + \cdots + A_1\times 10^1 + A_0 \times 10^0 + A_{-1}\times 10^{-1} + A_{-2}\times 10^{-2} + \cdots + A_{-m}\times 10^{-m})_\alpha$$

$$= (B_{n-1}\times \gamma^{n-1} + B_{n-2}\times \gamma^{n-2} + \cdots + B_1\times \gamma^1 + B_0\times \gamma^0 + B_{-1}\times \gamma^{-1} + B_{-2}\times \gamma^{-2} + \cdots + B_{-m}\times \gamma^{-m})_\beta = N_\beta$$

将α进制下的A_i、10转换成β进制下的数

其中：$(A_i)_\alpha=(B_i)_\beta$，$(10)_\alpha=(\gamma)_\beta$，则：$\gamma=\alpha_\beta$，$B_i=?$

注意：多项式替代法是在β进制下完成N_α到N_β的转换的，因此，要求熟悉β进制的算术运算规则。

例1：将$(1CE8)_{16}$转换为十进制。

$$(1CE8)_{16} = (1\times10^{3}+C\times10^{2}+E\times10^{1}+8\times10^{0})_{16}$$

$$= (1\times16^{3}+12\times16^{2}+14\times16^{1}+8\times16^{0})_{10}$$

$$= (4096+3072+224+8)_{10}$$

$= (7400)_{10}$ (计算是按十进制计算规则进行的)

$= 7400$ (十进制的R可以省略)

例2：将$(121.2)_3$转换为二进制。

$$(121.2)_3 = (1\times10^{2}+2\times10^{1}+1\times10^{0}+2\times10^{-1})_3$$

$$= (1\times11^{2}+10\times11^{1}+1\times11^{0}+10\times11^{-1})_2$$

$$= (1001+110+1+0.101010\cdots)_2$$

$$= (10000.101010\cdots)_2$$

(计算是按二进制计算规则进行的。)

2) 基数乘除法

要点： 在α进制下完成 $N_\alpha \to N_\beta$ 的转换

(1) 整数部分转换用基数除法

(2) 小数部分转换用基数乘法

① 整数转换(基数除法)

设: $N_\alpha = N_\beta = (b_{n-1} b_{n-2} \cdots b_1 b_0)_\beta \quad (0 \leqslant b_i \leqslant \beta - 1)$

$$= (b_{n-1} \times 10^{n-1} + b_{n-2} \times 10^{n-2} + \cdots + b_1 \times 10^1 + b_0 \times 10^0)_\beta$$

将b_i、10转换成α进制下的数，则

$$N_\alpha = (c_{n-1} \times \gamma^{n-1} + c_{n-2} \times \gamma^{n-2} + \cdots + c_1 \times \gamma^1 + c_0 \times \gamma^0)_\alpha$$

$(0 \leqslant c_i \leqslant \beta - 1)$

其中：$(b_i)_\beta = (c_i)_\alpha$，$(10)_\beta = (\gamma)_\alpha$，则: $\gamma = \beta_\alpha$

$$N_{\alpha} = (((\cdots((c_{n-1})\gamma + c_{n-2})\gamma + \cdots)\gamma + c_1)\gamma + c_0)_{\alpha}$$

在α进制下，将该数除以γ，则余数分别为: $c_0, c_1, \cdots, c_{n-1}$

再转化为β进制下的: $b_0, b_1, \cdots, b_{n-1}$

例1 将十进制的179 转换成二进制数。 $\alpha=10$，$\beta=2$，$\gamma=2$

$179 \div 2 = 89$	余1$\cdots\cdots c_0 \rightarrow b_0$
$89 \div 2 = 44$	余1$\cdots\cdots c_1 \rightarrow b_1$
$44 \div 2 = 22$	余0$\cdots\cdots c_2 \rightarrow b_2$
$22 \div 2 = 11$	余0$\cdots\cdots c_3 \rightarrow b_3$
$11 \div 2 = 5$	余1$\cdots\cdots c_4 \rightarrow b_4$
$5 \div 2 = 2$	余1$\cdots\cdots c_5 \rightarrow b_5$
$2 \div 2 = 1$	余0$\cdots\cdots c_6 \rightarrow b_6$
$1 \div 2 = 0$	余1$\cdots\cdots c_7 \rightarrow b_7$

即 $(179)_{10} = (10110011)_2$

练习

例2 将十进制的3417 转换成十六进制数。 α =10， β=16，

$\gamma=16$

$$16\,\underline{|3417}\quad 余9 \cdots\cdots c_0 \to b_0$$

$$16\,\underline{|213}\quad 余5 \cdots\cdots c_1 \to b_1$$

$$16\,\underline{|13}\quad 余13 \cdots c_2 \to b_2 \cdots\cdots D$$

即 $(3417)_{10} = (D59)_{16}$ 注意： 次序

例3 将二进制的1011 转换成三进制数。 α =2， β=3， γ=11

$$11\,\underline{|1011}\quad 余10 \cdots c_0 \to b_0 \cdots\cdots 2$$

$$11\,\underline{|11}\quad 余0 \cdots\cdots c_1 \to b_1$$

$$11\,\underline{|1}\quad 余1 \cdots\cdots c_2 \to b_2$$

即 $(1011)_2 = (102)_3$

②小数转换(基数乘法)

设：$N_{\alpha} = N_{\beta}$

$$= (0 . C_{-1} C_{-2} \cdots C_{-m})_{\beta}$$

$$= (C_{-1} \times 10^{-1} + C_{-2} \times 10^{-2} + \cdots + C_{-m} \times 10^{-m})_{\beta}$$

$$(0 \leqslant C_i \leqslant \beta - 1)$$

将 C_i、10转换成α进制下的数，则

$$N_{\alpha} = (D_{-1} \times \gamma^{-1} + D_{-2} \times \gamma^{-2} + \cdots + D_{-m} \times \gamma^{-m})_{\alpha}$$

其中：$(C_i)_{\beta} = (D_i)_{\alpha}$，$(10)_{\beta} = (\gamma)_{\alpha}$，则：$\gamma = \beta_{\alpha}$

$$(0 \leqslant D_i \leqslant \beta - 1)$$

$$N_\alpha = (D_{-1} + (D_{-2} + (\cdots((D_{-m+1} + ((D_{-m})\gamma^{-1}))\gamma^{-1})\cdots)\gamma^{-1})\gamma^{-1})_\alpha$$

在α进制下，将该数乘以γ, 则整数部分分别为: $D_{-1}, D_{-2}, \cdots, D_{-m}$

再转化为β进制下的: $C_{-1}, C_{-2}, \cdots, C_{-m}$

例1 将$(0.4321)_{10}$转换成十六进制数。 $\alpha=10$，$\beta=16$，$\gamma=16$

$N = 0.4321$

$0.4321 \times 16 = 6.9136$ 得： $D_{-1} = 6\ (6)$

$0.9136 \times 16 = 14.6176$ $D_{-2} = 14(E)$

$0.6176 \times 16 = 9.8816$ $D_{-3} = 9\ (9)$

$0.8816 \times 16 = 14.1056$ $D_{-4} = 14(E)$

即 $(0.4321)_{10} \approx (0.6E9E)_{16}$

练习

例2 将 $(0.375)_{10}$ 转换成二进制数。 $\alpha=10$，$\beta=2$，$\gamma=2$

$$
\begin{array}{l}
\quad 0\ .\ 3\ 7\ 5 \\
\times \qquad\quad 2 \\
\hline
[0]\,.\,7\ 5\ 0 \\
\times \qquad\quad 2 \\
\hline
[1]\,.\,5\ 0\ 0 \\
\times \qquad\quad 2 \\
\hline
[1]\,.\,0\ 0\ 0
\end{array}
$$

…… …… $D_{-1}=0$ 转化为β进制下的 C_{-1}

…… …… $D_{-2}=1$转化为β进制下的 C_{-2}

…… …… $D_{-3}=1$转化为β进制下的 C_{-3}

即 $(0.375)_{10}=(0.011)_2$ 注意：次序

3.任意两种进制之间的转换

$$N_{\alpha} \longrightarrow N_{\beta}$$

① 若熟悉β进制的运算规则，则采用多项式替代法完成转换。(要求熟悉单个数字字符的转换)

② 若熟悉α进制的运算规则，则采用基数乘除法完成转换;

注意：

a. $\gamma = \beta_{\alpha;}$

b. 整数（部分）除以γ得余数，小数（部分）乘以γ取整数;

c. 再从α进制转换为β进制，同时注意次序。

③ 若不熟悉α、β进制的运算规则，则可利用十进制作为转换桥梁。

例3 将$(1023.231)_4$转换成五进制数。

第一步： $(1023.231)_4 \xrightarrow[\text{替代法}]{\text{多项式}} (\quad ? \quad)_{10}$

$(1023.231)_4$

$= (1\times 10^3 + 0\times 10^2 + 2\times 10^1 + 3\times 10^0 + 2\times 10^{-1} + 3\times 10^{-2}$

$+ 1\times 10^{-3})_4$

$= (1\times 4^3 + 0\times 4^2 + 2\times 4^1 + 3\times 4^0 + 2\times 4^{-1} + 3\times 4^{-2}$

$+ 1\times 4^{-3})_{10}$

$= 64 + 0 + 8 + 3 + 0.5 + 0.1875 + 0.015625$

$= 75.703125$

第二步： $(75.703125)_{10} \xrightarrow[\text{除法}]{\text{基数乘}} (\quad ? \quad)_5$

整数部分与小数部分分别转换。

整数部分

$$
\begin{array}{r|l l}
5 & 75 & \cdots\cdots\ b_0 = 0 \\
\hline
5 & 15 & \cdots\cdots\ b_1 = 0 \\
\hline
5 & 3 & \cdots\cdots\ b_2 = 3 \\
\hline
 & 0 &
\end{array}
$$

小数部分

$$
\begin{array}{r l}
 & 0\,.\,703125 \\
\times & 5 \\
\hline
 & [3]\,.\,515626 \quad \cdots\cdots\ C_{-1} = 3 \\
\times & 5 \\
\hline
 & [2]\,.\,578125 \quad \cdots\cdots\ C_{-2} = 2 \\
\times & 5 \\
\hline
 & [2]\,.\,890625 \quad \cdots\cdots\ C_{-3} = 2 \\
\times & 5 \\
\hline
 & [4]\,.\,453125 \quad \cdots\cdots\ C_{-4} = 4
\end{array}
$$

即 $(75.703125)_{10} \approx (300.3224)_5$

$\therefore\ (1023.231)_4 \approx (300.3224)_5$

4.直接转换法

二进制数是计算机内部真正使用的数，但由于它的表示既易出错也不易交流，故常常用八进制或十六进制的形式表示。

二进制*Binary*，简称B，如 $(10)_2 = (10)_B$ ；

八进制*Octal*，简称O ，如 $(10)_8 = (10)_O$ ；

十六进制*Hexadecimal*，简称H ，如 $(10)_{16} = (10)_H$ 。

转换：　　$N_{\alpha} \longrightarrow N_{\beta}$

当基数α、β是2的幂次方时，可以进行直接转换。

基数为 2^k 的进位制是将一个**k位**二进制字符串用**一位**数字字符表示，如$(5)_8 = (101)_2$、 $(5)_{16} = (0101)_2$

因此，用**划分相应字符串**的方法实现基数为 2^k 的进位制与**二进制**之间的直接转换。

注意：从小数点位置向左右划分，不足K位补0。

例1 将$(11111010.0111)_2$转换为八进制数。

二进制数	011	111	010	.	011	100
八进制数	3	7	2	.	3	4

即 $(11111010.0111)_2 = (372.34)_8$

例2 将$(AF.16C)_{16}$转换为八进制数。

十六进制数	A	F	.	1	6	C	
二进制数	1010	1111	.	0001	0110	1100	

二进制数 1 0 1 0 1 1 1 1 . 0 0 0 1 0 1 1 0 1 1 0 0

八进制数	2	5	7	.	0	5	5	4

即 $(AF.16C)_{16} = (257.0554)_8$

5.数制转换时小数位数的确定

阅读

问题：十进制的0.001，转换为四进制时小数部分取几位?

例子：为保持十进制数0.326与0.327的精度（在千分位有差别），应至少转换为四进制的几位?

$$\begin{array}{rl} & 0.326 \\ \times & 4 \\ \hline [1].304 & \cdots\cdots C_{-1}=1 \\ \times & 4 \\ \hline [1].216 & \cdots\cdots C_{-2}=1 \\ \times & 4 \\ \hline [0].864 & \cdots\cdots C_{-3}=0 \\ \times & 4 \\ \hline [3].456 & \cdots\cdots C_{-4}=3 \\ \times & 4 \\ \hline [1].824 & \cdots\cdots C_{-5}=\underline{1} \end{array}$$

$$\begin{array}{rl} & 0.327 \\ \times & 4 \\ \hline [1].308 & \cdots\cdots C_{-1}=1 \\ \times & 4 \\ \hline [1].232 & \cdots\cdots C_{-2}=1 \\ \times & 4 \\ \hline [0].928 & \cdots\cdots C_{-3}=0 \\ \times & 4 \\ \hline [3].712 & \cdots\cdots C_{-4}=3 \\ \times & 4 \\ \hline [2].848 & \cdots\cdots C_{-5}=\underline{2} \end{array}$$

∴小数部分取至少5位。

也就是说，应有 $(0.1)_{10}^{3} = (0.1)_{4}^{5} = (0.0009765625)_{10}$

阅读

设：α进制的小数有k位，转换成β进制后，至少具有相同精度的小数是 j 位，则 $(0.1)_{\alpha}^{k} >= (0.1)_{\beta}^{j}$

在十进制中可表示为：$(\frac{1}{\alpha})^{k} >= (\frac{1}{\beta})^{j}$

即 $\alpha^{k} =< \beta^{j}$

两边取对数 $\log \alpha^{k} =< \log \beta^{j}$

即 $k \log \alpha =< j \log \beta$

$$j >= k \frac{\log \alpha}{\log \beta}$$

(j 取最小的整数)

则 j 应满足不等式的整数：$k \frac{\log \alpha}{\log \beta} \leqslant j < k \frac{\log \alpha}{\log \beta} + 1$

阅读

例：将$(0.4321)_{10}$转换成十六进制时，小数位数应取几位？

j 应满足下列不等式：

$$4\frac{\log 10}{\log 16} \leqslant j < 4\frac{\log 10}{\log 16}+1$$

即 $$3.320 \leqslant j < 4.320$$

所以，小数位数应取4位。

1.2.2 数的表示及其运算

1. 数的小数点的表示:

- 小数点的位置在计算机中是怎样表示的呢?实际上，小数点在机器中并不存在，它只是人们约定的一个位置。
- 计算机中表示小数点位置的方法通常有两种:

一种是定点表示法；另一种是浮点表示法。

① 数的定点表示法

所谓定点表示法，在计算机中数的小数点位置是固定的，一般固定在数的最高位之前或数的最低位之后。

定点小数表示法：小数点均约定在最高位之前，符号位之后。

定点整数表示法：小数点均约定在最低位之后。

但实际处理的数可能既有整数部分，又有小数部分。这就需要选取比例因子。

假设某计算机字长8位，规定最高位用来表示数的正、负号，其余7位表示数值。现有一组数：1101.01，-100.111和1110.1，若用定点整数表示，可选取比例因子2^3，并用此比例因子乘上述各数，可得

$$1101.01 \times 2^3 = 1101010$$

$$-100.111 \times 2^3 = -0100111$$

$$1110.1 \times 2^3 = 1110100$$

它们在机器中的表示形式为：

0 1 1 0 1 0 1 0

1 0 1 0 0 1 1 1 (原码)

0 1 1 1 0 1 0 0

若要用定点小数表示，则需提取比例因子2^{-4}，可得

$$1101.01 \times 2^{-4} = 0.110101$$

$$-100.111 \times 2^{-4} = -0.0100111$$

$$1110.1 \times 2^{-4} = 0.11101$$

它们在机器中的表示形式为：

0	1	1	0	1	0	1	0
1	0	1	0	0	1	1	1
0	1	1	1	0	1	0	0

②数的浮点表示法

所谓浮点表示法，就是计算机中数的小数点位置不是固定的，而是浮动的。因而，计算机必须表示出小数点位置的浮动情况。

在普通数学中，有数的“记阶表示法”(也称“科学表示法”)。一般说来，十进制数N可表示为: $N=10^J \times S$

其中，J是整数，称为阶码；S为小数，称为尾数。显然，只要有J和S，即可表示出N的值。

例如，十进制数2.537可表示为：$2.537 = 10^1 \times 0.253\,7$

$$= 10^2 \times 0.025\,37$$

而数0.002 537也可表示为：$0.002\,537=10^{-1} \times 0.025\,37$

$$=10^{-2} \times 0.253\,7$$

同样，二进制数也可用这种方法表示，其中的基数“10”是十进制中的2，即$(10)_2=(2)_{10}$。

例如，二进制数101.1和10.11可表示为

$101.1=(10)^{11} \times 0.1011$ 和 $10.11=(10)^{10} \times 0.1011$

因此，这两个二进制数可用阶码和尾数表示为

101.1—>11， 0.1011

10.11—>10， 0.1011

不难看出，其尾数完全相同，仅阶码不同。

这种用阶码和尾数表示数的方法就是数的浮点表示法。

表示浮点数时，需将一个字长划分为两部分，其中一部分表示阶码，另一部分表示尾数，其高位都定为符号位。

例如，规定某一16位字长的前5位表示阶码的符号及数值，后11位表示尾数的符号及数值，如下所示：

按此规定，数101.1 和10.11 的实际表示形式为

0 0011 0 1011000000

0 0010 0 1011000000

2. 带符号数的代码表示及其运算

带符号数	真值	符号位	数值位
x	＋5	＋	5
y	－7	－	7

带符号（进位计数制）数的真值表示

如：$(-111)_2$是带符号（二进制）数的真值表示

带符号(二进制)数的代码表示是指（在计算机中）：

带符号数的数值位和符号位都用 统一的代码形式，即仅取 0 和 1 两种数字符号表示，并且数值位为二进制数。有三种代码表示：原码、反码和补码。

① 原码 *True form* (符号-数值表示法)

原码的形成规则：

真值 x	符号位 S	数值位 (二进制数)
原码 $[x]_{原}$	$+\rightarrow 0$ $-\rightarrow 1$	不变 不变

例1 用8位二进制代码表示的原码

$x=+5 \quad x=+101 \quad [x]_{原}=00000101$

$y=-7 \quad y=-111 \quad [y]_{原}=10000111$

例2 由原码写出所对应的十进制数值： 练习

$[x]_{原}=01010101 \quad x=+85;$ $[x]_{原}=11010101 \quad x=-85;$

$[x]_{原}=01111111 \quad x=+127;$ $[x]_{原}=11111111 \quad x=-127;$

$[x]_{原}=00000000 \quad x=+0;$ $[x]_{原}=10000000 \quad x=-0;$

原码的运算规则： $[z]_{原}=[x+y]_{原}=[x]_{原}+[y]_{原}$

➢ 当$[x]_{原}$、$[y]_{原}$的**符号S相同**时，则$[z]_{原}$的符号同$[x]_{原}$的符号，$[z]_{原}$的数值为两加数的和；

➢ 当$[x]_{原}$、$[y]_{原}$的**符号S相异时**，先判断两加数真值的绝对值的大小， $[z]_{原}$的符号同大数的原码符号， $[z]_{原}$的数值为用大数减去小数的差。

在一个 **n 位**原码系统中（包括一位符号位），除了符号位，其余 n-1 位数值位可表示的数值（绝对值）范围是$0\sim2^{n-1}-1$。因此带符号 n 位原码可表示的数值范围是$-(2^{n-1}-1)\sim(2^{n-1}-1)$。以n＝4为例：

−7，	−6，	−5，	−4，	−3，	−2，	−1，	0，	1，	2，	3，	4，	5，	6，	7
1111，	1110，	1101，	1100，	1011，	1010，	1001，	1000							
							0000，	0001，	0010，	0011，	0100，	0101，	0110，	0111

② 反码 *Negative Number* (对“1”)

反码的形成规则：

真值 X	符号位 S	数值位 (二进制数)
反码 $[x]_{反}$	$+ \rightarrow 0$ $- \rightarrow 1$	不变 按位变反

例 用8位二进制代码表示的原码、反码

$x = +5 \quad x = +101 \quad [x]_{原} = 00000101 \quad [x]_{反} = 00000101$

$y = -7 \quad y = -111 \quad [y]_{原} = 10000111 \quad [y]_{反} = 11111000$

$[z]_{反} = 00000000 \quad [z]_{反} = 11111111 \quad z = 0$

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

反码的运算规则： $[z]_{反}=[x+y]_{反}=[x]_{反}+[y]_{反}$

➢ 把符号位看作一位数值位，一律按加法规则来处理，所得结果的符号位也就是正确结果的符号。

➢ 当符号位产生进位时，将产生的进位要加到数值位的最低位。

在一个n位反码系统中（包括一位符号位），除了符号位，其余n-1位数值位可表示的数值（绝对值）范围是 $0 \sim 2^{n-1}-1$。因此带符号n位反码可表示的数值范围也是 $-(2^{n-1}-1) \sim (2^{n-1}-1)$。以n＝4为例：

−7，　−6，　−5，　−4，−3，−2，−1，0，1，2，3，4，5，6，7

1000，1001，1010，1011，1100，1101，1110，1111

0000，0001，0010，0011，0100，0101，0110，0111

③补码 two' s-complement Representation

补码的形成规则：

真值 x	符号位 S	数值位 (二进制数)
补码 $[x]_{补}$	＋→0 － →1	不变 按位变反+1

例 用8位二进制代码表示的原码、补码

x＝＋5 x=＋101 $[x]_{原}$ = 00000101 $[x]_{补}$ ＝00000101

y＝－7 y=－111 $[y]_{原}$ = 10000111 $[y]_{补}$ ＝11111001

z＝0 $[z]_{原}$ = 00000000 $[z]_{补}$ ＝00000000

z＝－128 z=－10000000 $[z]_{补}$ = 10000000

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

补码的运算规则： $[z]_{补}=[x+y]_{补}=[x]_{补}+[y]_{补}$

- 把符号位看作一位数值位，一律按加法规则来处理，所得结果的符号位也就是正确结果的符号。
- 当符号位产生进位时，将产生的进位丢掉。

n 位补码可表示的数值范围是 $-2^{n-1} \sim (2^{n-1}-1)$。

以n＝4为例

−8，−7，−6，−5，−4，−3，−2，−1，0，1，2，3，4，5，6，7

1000，1001，1010，1011，1100，1101，1110，1111

0000，0001，0010，0011，0100，0101，0110，0111

阅读

n=4时的原码、反码和补码：

-7，-6，-5，-4，-3，-2，-1，0，1，2，3，4，5，6，7

1111，1110，1101，1100，1011，1010，1001，1000

0000，0001，0010，0011，0100，0101，0110，0111

-7，-6，-5，-4，-3，-2，-1，0，1，2，3，4，5，6，7

1000，1001，1010，1011，1100，1101，1110，1111

0000，0001，0010，0011，0100，0101，0110，0111

-8，-7，-6，-5，-4，-3，-2，-1，0，1，2，3，4，5，6，7

1000，1001，1010，1011，1100，1101，1110，1111

0000，0001，0010，0011，0100，0101，0110，0111

试分析下表，你会有什么结论？

阅读

二进制代码	无符号数 x	$[x]_原$	$[x]_反$	$[x]_补$
0000	0	+0	+0	+0
0001	1	+1	+1	+1
0010	2	+2	+2	+2
0011	3	+3	+3	+3
0100	4	+4	+4	+4
0101	5	+5	+5	+5
0110	6	+6	+6	+6
0111	7	+7	+7	+7
1000	8	−0	−7	−8
1001	9	−1	−6	−7
1010	10	−2	−5	−6
1011	11	−3	−4	−5
1100	12	−4	−3	−4
1101	13	−5	−2	−3
1110	14	−6	−1	−2
1111	15	−7	−0	−1

阅读

补码的概念:

由于补码加、减运算的简便性，导致了它在计算机中被广泛应用,为了理解补码，有必要对补码的本质做进一步的说明。从数学上讲，补码与其真值构成了一个以某一值(由计算机字长或其他规定所决定)为模的“模数系统”，或者说构成了一个“同余”结构的代数系统。

在此基础上，不仅能容易地构成二进制数的各种不同模数的补码，而且能很容易地构成十进制或任何R进制中的补码。

阅读

“模”或称为“模数”，就好比一个计量器的容量，一个计量器就构成了一个模数系统。

例如：我们的手表上有12个小时，这12就是这个手表的模数，手表也构成一个模为12的模数系统。

在模数系统中，模数减去一个数等于该数的补，或称这两个数互补。

例如12-1=11，我们将11称为1的补，或称1与11互补。

在模数系统中，减去一个数就等于加上这个数的补。

例如在手表中，X-1=X+11，假如 X=3，则X-1=2而且X+11=14,在手表上14就是2。在这里我们也把14与2称为同余数。

阅读

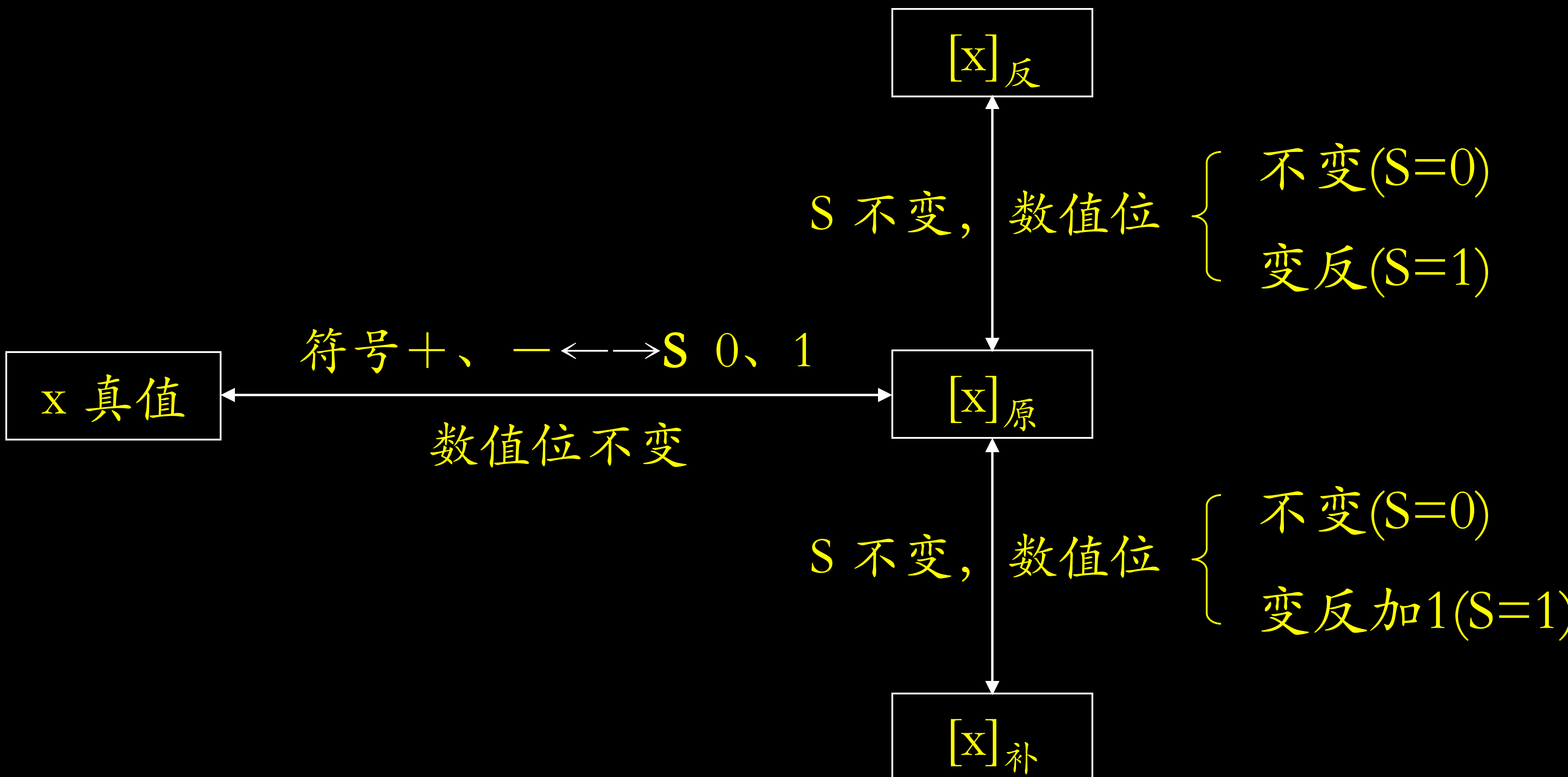
在二进制的计算机系统中，对于字长为8的机器，它的容量是256，因为减去1等于加上255，所以255与1互补，1是255的补，255是1的补，通常称-1的补码就是255，也就是说减去1等于加上-1的补码255。实际上，这是对256求补。如果对255求补，则1的补就是254，但是，这里往往不将254称为-1的补码。

补码是二进制数字系统中带符号数的一种表示。

在书中，常常说，0对1求补是1,1对1求补是0，实际上是求反；对于自补码，2对9求补是7,7对9求补是2，也是求反。所以书中常把求反称为求补就是这个原因。求补是针对某个数来说的。补码针对的数是二进制系统字长所表示的模。

阅读

原码、反码和补码之间的关系如下图



阅读

④带符号数的加、减运算

- 原码 加减法有不同的规则，关键是要判大小；
- 反码 $[x+y]_{反}=[x]_{反}+[y]_{反}$；

  $[x-y]_{反}=[x]_{反}+[-y]_{反}$;
- 补码 $[x+y]_{补}=[x]_{补}+[y]_{补}$；

  $[x-y]_{补}=[x]_{补}+[-y]_{补}$;

在反码和补码运算中，

① 减法运算也按加法运算完成;

② 符号位S也被看成一位数码，并与数值位一样，按同样的加法规则进行处理，所得结果的符号位即是正确结果的符号位。

③ 反码运算时符号位产生的进位要加到数的最低位上去，补码运算时符号位产生的进位要丢掉。

④带符号数的加、减运算

例1　求 $z = x - y$，其中 $x = +1010$，$y = +0011$

(1) 原码运算：　$[x]_{原} = 01010$　　$[y]_{原} = 00011$

∵ x 绝对值 > y 绝对值

∴ $[z]_{原} = [01010 - 00011]_{原} = 00111$　　$z = +0111$

(2) 补码运算：　$[x]_{补} = 01010$

$[-y]_{补} = [-0011]_{补} = 11101$

∵
$$
\begin{array}{r}
01010 \\
+11101 \\
\hline
\boxed{1}\,00111
\end{array}
$$
丢掉←

∴ $[z]_{补} = 00111$

$z = +0111$

例1 求 $z = x - y$，其中 $x = +1010$，$y = +0011$

(3)反码运算： $[x]_{反} = 01010$

$[-y]_{反} = [-0011]_{反} = 11100$

$$\because \quad \begin{array}{r} 01010 \\ +11100 \\ \hline \boxed{1}\,00110 \\ +\quad 1 \\ \hline 00111 \end{array}$$

$\therefore\ [z]_{反} = 00111$

$z = +0111$

"模2和" 的运算规则如下

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

$0 \oplus A = A \qquad 1 \oplus A = \overline{A} \qquad A \oplus A = 0 \qquad \overline{A} \oplus A = 1$

用真值表验证：

A	$0 \oplus A$	$1 \oplus A$	$A \oplus A$	$A \oplus \overline{A}$
0	0	1	0	1
1	1	0	0	1

1.2.3 十进制数的常用编码（代码）表示及其运算

十进制数的编码表示：BCD码 *Binary coded decimal*，二-十进制码

- 既具有二进制数的形式，又具有十进制数的特点，即用四位二进制数表示一位十进制数;
- 可按位直接相互转换;
- 可按位直接运算（要修正）。

主要有：“8421”码

“2421”码

余3码 *(Excess-3)*

表1.3 三种十进制数的代码表示法

十进制整数	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
无效码区 Unused code wrds	1010、1011、1100、1101、1110、1111	0101、0110、0111、1000、1001、1010	0000、0001、0010、1101、1110、1111

三种十进制数代码的分布图

四位二进制代码	8421码	2421码	余3码
0000	0000 0	0000 0	0000 非码区
0001	0001 1	0001 1	0001
0010	0010 2	0010 2	0010
0011	0011 3	0011 3	0011 0
0100	0100 4	0100 4	0100 1
0101	0101 5	0101 非码区	0101 2
0110	0110 6	0110	0110 3
0111	0111 7	0111	0111 4
1000	1000 8	1000	1000 5
1001	1001 9	1001	1001 6
1010	1010 非码区	1010	1010 7
1011	1011	1011 5	1011 8
1100	1100	1100 6	1100 9
1101	1101	1101 7	1101 非码区
1110	1110	1110 8	1110
1111	1111	1111 9	1111

设：代码表示为 $A_3A_2A_1A_0$

代码	对应的十进制数值	代码直接按位转换
8421码	有权码 (*Weighted code*) $8A_3+4A_2+2A_1+1A_0$	$(13)_{10}=(00010011)_{BCD}$ $(1011101010000)_{BCD}=(1750)_{10}$
2421码	有权码、对9自补码 $2A_3+4A_2+2A_1+1A_0$	$(13)_{10}=(00010011)_{2421}$ $(1110110110000)_{2421}=(1750)_{10}$
余3码	无权码、对9自补码 $8A_3+4A_2+2A_1+1A_0$ -0011	$(13)_{10}=(01000110)_{余3}$ $(100101010000011)_{余3}=(1750)_{10}$

代码运算若直接按二进制数运算，则运算结果要修正。

练习

2421码是一种对9的自补代码：

- 2421码$A_3\ A_2\ A_1\ A_0$对应十进制数值$2A_3+4A_2+2A_1+1A_0$

  其9补码为

  $$9-（2A_3+4A_2+2A_1+1A_0）$$
  $$=2(1-A_3)+4(1-A_2)+2(1-A_1)+1(1-A_0)$$

  例如1011（5）按位取反为0100（4）。

余3码是一种对9的自补代码：

- 余3码$A_3\ A_2\ A_1\ A_0$对应十进制数值$8A_3+4A_2+2A_1+1A_0-3$

  其9补码为

  $$9-（8A_3+4A_2+2A_1+1A_0）+3$$
  $$=15-（8A_3+4A_2+2A_1+1A_0）-3$$
  $$=8(1-A_3)+4(1-A_2)+2(1-A_1)+1(1-A_0)-3$$

  例如0110（3）按位取反为1001（6）。

代码运算: $C = A + B$

若按二进制数直接运算，则运算结果要修正。

例：一位代码的加法运算，如下：

8421码 $C_8C_4C_2C_1 = A_8A_4A_2A_1 + B_8B_4B_2B_1$	2421码 $C_2C_4C_2C_1 = A_2A_4A_2A_1 + B_2B_4B_2B_1$	余3码 $C_4C_3C_2C_1 = A_4A_3A_2A_1 + B_4B_3B_2B_1$
1. 当0000≤C≤1001，不需修正 例：0011＋0100 2. 当1010≤C≤1111，C＋0110 例：0111＋0100 3. 当C≥10000，C＋0110 例：1001＋1001	修正的算法较复杂（略）	1. 当C ≤ 1111，C－0011 例：0011＋0100 2. 当C≥10000，C＋0011 例：1100＋0100

- 8421码

练习

1. 当0000≤C≤1001， 不需修正
2. 当1010≤C≤1111，C＋0110

十进制数	8421码C	修正8421码	
10	1010	10000	（16）
11	1011	10001	（17）
:	:	:	:
15	1111	10101	（21）

3. 当C≥10000，C＋0110

十进制数	8421码C	修正8421码
16	10000	10110
17	10001	10111
:	:	:

1.2.4. 可靠性编码

目的：解决代码在形成或传输过程中可能会发生的错误，提高系统的安全性

方法：使代码自身具有一种特征或能力

作用：1. 不易出错

2. 若出错时易发现错误

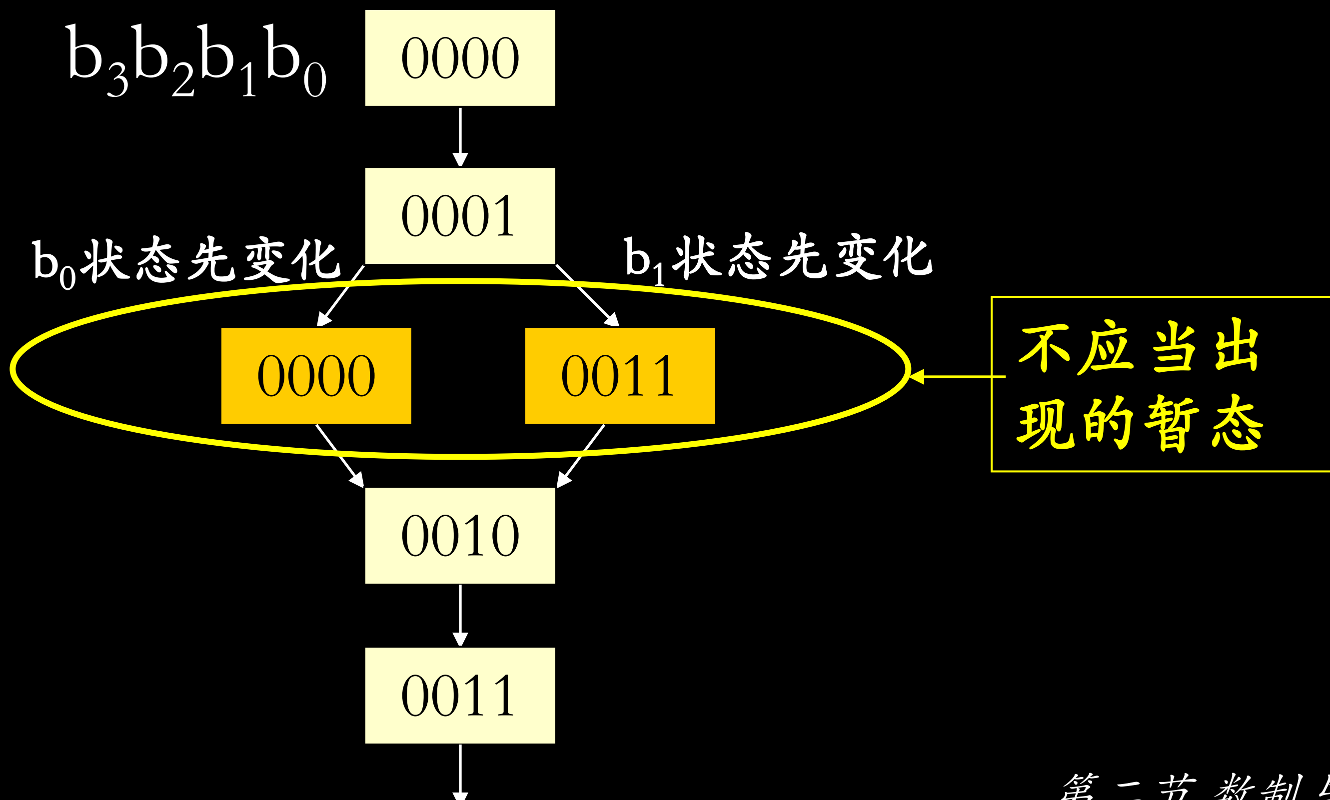
3. 出错时易查错且易纠错

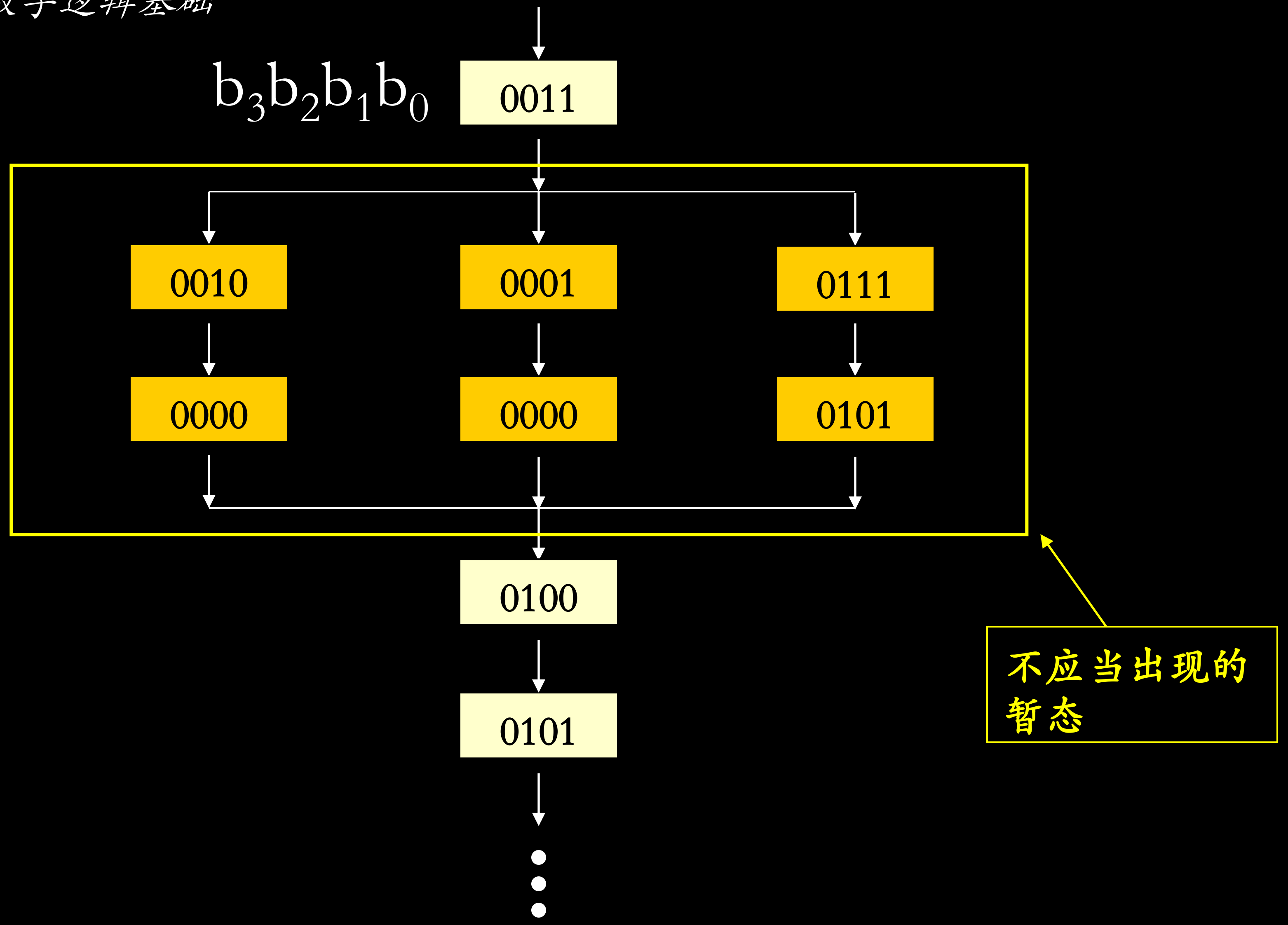
1. 格雷码（Gray)

特点：任意两个相邻数的代码只有一位二进制数不同

目的：解决代码生成时发生的错误

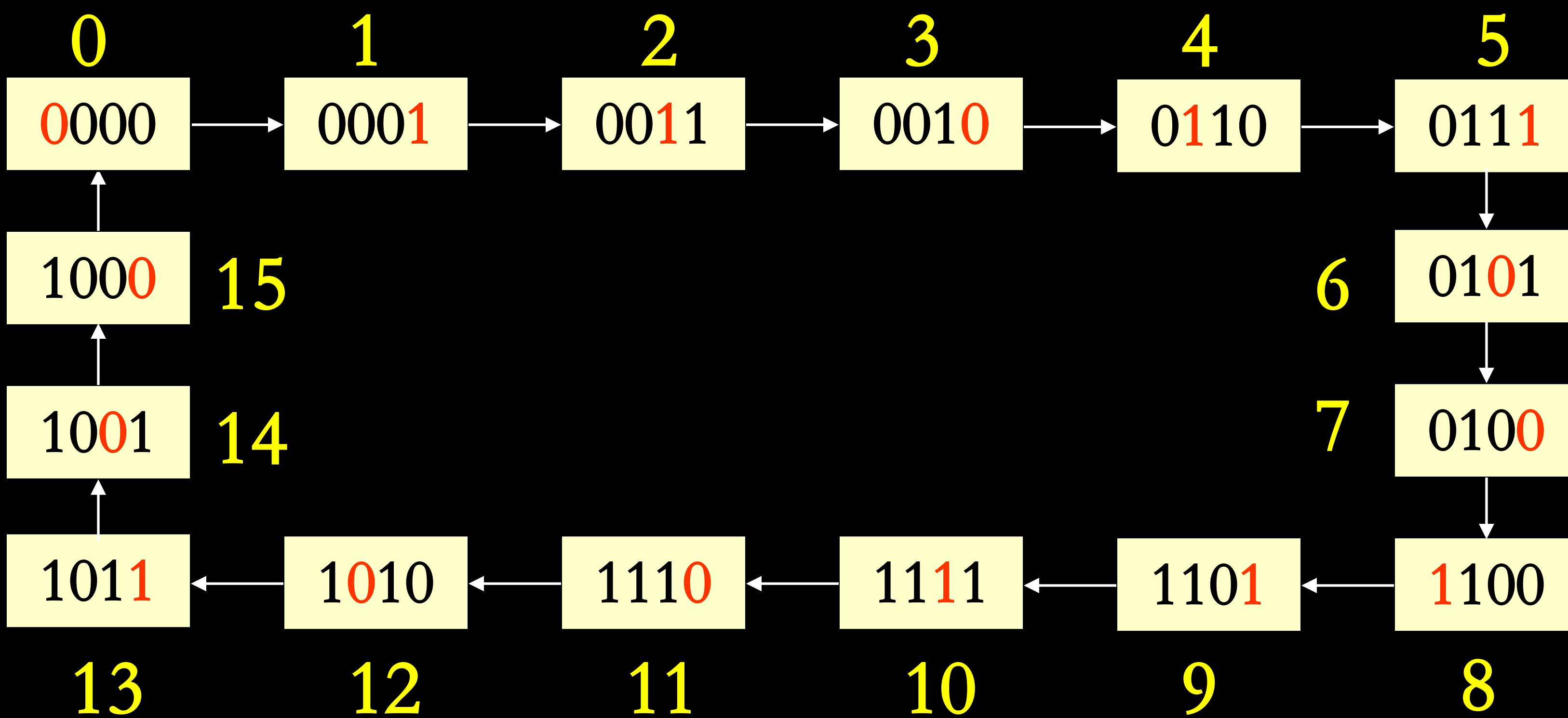
例：四位二进制加1计数器，工作时有如下情况出现





这种情况出现的最为严重的是当由1111加1计数到0000时，出现了所有14个可能的暂态。

但是，当选用典型Gray码设计加1计数器时，就不会出现上述情况。如下：



Gray码还包括步进码、十进制Gray 码等。参见书p15.表1.4 。

表1.4 几种Gray码、步进码和二进制码对照表

十进制数	二进制数	典型Gray	十进制 Gray码(1)	十进制 Gray码(2)	步进码
0	0000	0000	0000	0000	00000
1	0001	0001	0001	0001	00001
2	0010	0011	0011	0011	00011
3	0011	0010	0010	0010	00111
4	0100	0110	0110	0110	01111
5	0101	0111	1110	0111	11111
6	0110	0101	1010	0101	11110
7	0111	0100	1011	0100	11100
8	1000	1100	1001	1100	11000
9	1001	1101	1000	1000	10000
10	1010	1111			
11	1011	1110			
12	1100	1010			
13	1101	1011			
14	1110	1001			
15	1111	1000			

这种典型格雷码还有一个特点：

所有对应于十进制数 2^m-1 (m为正整数）的格雷码，都仅在m位上有 1，其他位都为0。

例：

m	十进制数 2^m-1	典型Gray码
1	1（0001）	0001
2	3（0011）	0010
3	7（0111）	0100
4	15（1111）	1000

这些数与 0 之间只有一位的差别，回到 0仍能保持一位差别的特点，所以称作循环码，特别适用做二进制码计数器。

阅读

十进制Gray码：代码特征应符合Gray码的要求，且十六选十。

使得9 回到 0 也只有一位差别，并保持循环码的特色，表中已列出了其中两种：

- 十进制Gray码(1)
- 十进制Gray码(2)

典型Gray码： 是由二进制码生成的。

通过异或运算 ⊕ 完成。 $G_i = B_{i+1} \oplus B_i$,

设：二进制码 B　　$0 \quad B_{n-1} \quad B_{n-2} \cdots B_1 \quad B_0$

$\oplus \quad \oplus \quad \oplus \quad \oplus$

典型Gray码G　　$G_{n-1} \quad G_{n-2} \cdots G_1 \quad G_0$

例：二进制码 B　　0　1　1　1　0

$\oplus \quad \oplus \quad \oplus \quad \oplus$

典型Gray码G　　1　0　0　1

阅读

∴ 由二进制码生成典型Gray码

$$G_0 = B_1 \oplus B_0$$

$$G_1 = B_2 \oplus B_1$$

$$\vdots$$

$$G_i = B_{i+1} \oplus B_i$$

$$\vdots$$

$$G_{n-2} = B_{n-1} \oplus B_{n-2}$$

$$G_{n-1} = 0 \oplus B_{n-1} = B_{n-1}$$

反之，由典型Gray码也可以得到二进制码，如下：

设：典型Gray码 $G_{n-1}\ G_{n-2}\cdots\ G_1\ G_0$

$\because\ G_{n-1}=B_{n-1}\oplus 0$ $\quad\therefore\ B_{n-1}=G_{n-1}$

$\because\ G_i=B_{i+1}\oplus B_i$

则等式两边同时 $\oplus\ B_{i+1}$：

$$G_i\oplus B_{i+1}=B_{i+1}\oplus B_i\oplus B_{i+1}$$

$\therefore\ B_i=G_i\oplus B_{i+1}$

故 $B_{n-2}=G_{n-2}\oplus B_{n-1}$

$\vdots$

$B_0=G_0\oplus B_1$

设：典型Gray码 G　$G_{n-1}G_{n-2} \cdots G_1 \quad G_0$　练习

$\oplus \qquad \oplus \quad \oplus$

二进制码 B　$B_{n-1} \quad B_{n-2} \cdots B_1 \quad B_0$

例：典型Gray码G　1　1　1　0

$\oplus \quad \oplus \quad \oplus$

二进制码 B　1　0　1　1

缺点：电路延时长

练习

$$\because \quad B_i = G_i \oplus B_{i+1}$$

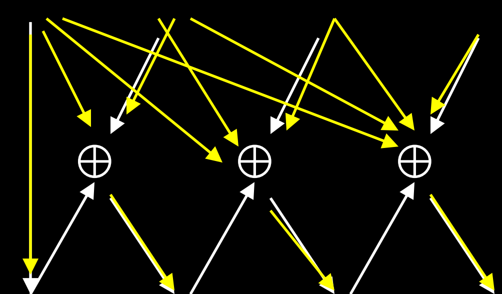
$$\therefore \quad B_{n-1} = G_{n-1} \oplus B_n = G_{n-1} \oplus 0 = G_{n-1}$$

$$B_{n-2} = G_{n-2} \oplus B_{n-1} = G_{n-2} \oplus G_{n-1}$$

$$\vdots$$

$$B_0 = G_{n-1} \oplus G_{n-2} \oplus \cdots \oplus G_1 \oplus G_0$$

例：典型Gray码G 1 1 1 0



二进制码 B 1 0 1 1

特点：电路延时短，但位数越长，异或关系越多

分析公式：$B_i = G_{n-1} \oplus G_{n-2} \oplus \cdots \oplus G_{i+1} \oplus G_i$ 练习

根据异或运算 ⊕ 的特点，

$B_i = 0$ 当（$G_{n-1}G_{n-2} \cdots G_i$ 中“1”的个数为偶数）

$B_i = 1$ 当（$G_{n-1}G_{n-2} \cdots G_i$ 中“1”的个数为奇数）

重新设计电路，改异或运算电路为判别电路（判“1”）

例：G = 1 1 0 1 0 0 1 0 1

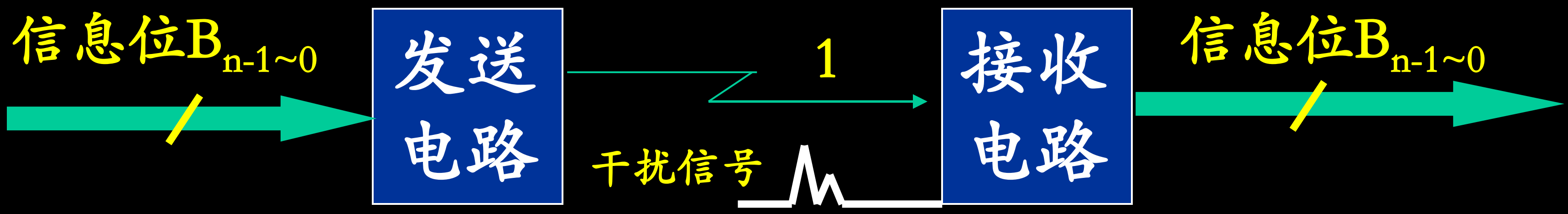
见1变反

B = 1 → 0 → 0 → 1 → 1 → 1 → 0 → 0 → 1

特点：电路实现简单。

➢ 校验码和纠错码 *Codes for detecting and Correcting Errors*

传输系统电路示意图:



问题：信息在传输过程中受外界干扰而出错，

且绝大多数为单错。

解决方法：① 增加校验位（P）；

② 通过异或运算⊕；

2. 奇偶校验码 *Parity Code*

校验码：

信息位$B_{n-1\sim 0}$	校验位 P

- 偶校验：

  校验码P的取值使校验码中“1”的个数是偶数；

  校验关系：$B_{n-1} \oplus B_{n-2} \oplus \cdots \oplus B_1 \oplus B_0 \oplus P_{偶} = 0$

  $P_{偶} = B_{n-1} \oplus B_{n-2} \oplus \cdots \oplus B_1 \oplus B_0$

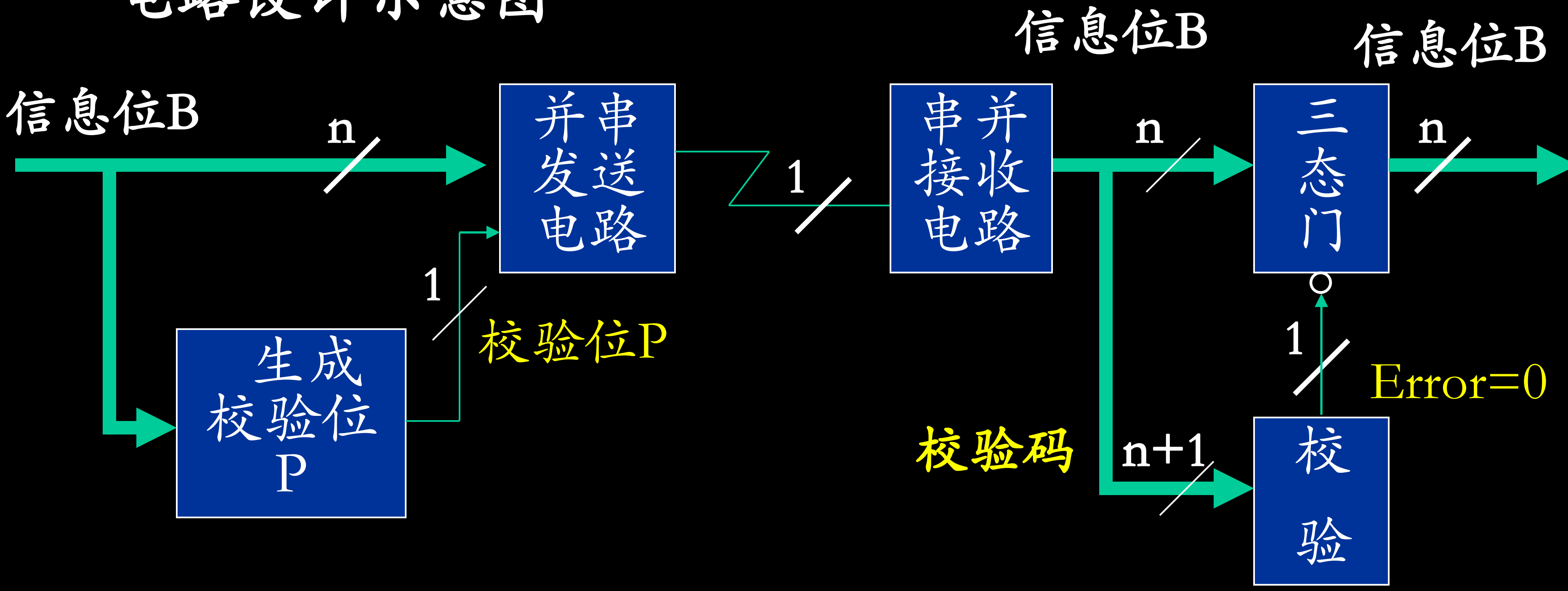
- 奇校验：

  校验码P的取值使校验码中“1”的个数是奇数；

  校验关系：$B_{n-1} \oplus B_{n-2} \oplus \cdots \oplus B_1 \oplus B_0 \oplus P_{奇} = 1$

  $P_{奇} = B_{n-1} \oplus B_{n-2} \oplus \cdots \oplus B_1 \oplus B_0 \oplus 1$

电路设计示意图



奇偶校验码：具有发现**一位错**的能力。

例：偶校验结果

$$Error_{偶} = B_{n-1} \oplus B_{n-2} \oplus \cdots \oplus B_1 \oplus B_0 \oplus P_{偶}$$

若 $Error_{偶}=0$ 则 传输正确

$Error_{偶}=1$ 传输出错

问题：① 奇偶校验码能发现单错，但不能对单错定位；
② 奇偶校验码不能发现双错；

对于问题①，或要求重新再发；
或在信息传送中采用横向及纵向奇偶校验，即可对单错定位。

对于问题②，试想：为什么奇偶校验码不能发现双错？

3. 海明校验码 *Hamming codes*

- 目的：不仅能检测出单错，还能校正单错
- 方法：增加校验位及相应的异或运算

以四位信息位$B_4B_3B_2B_1$为例，在传输前生成它的海明校验码：

(1) 位序： 7 6 5 4 3 2 1

$B_4\ B_3\ B_2\ P_3\ B_1\ P_2\ P_1$

(2) 校验位的生成公式：

$$P_3 = B_4 \oplus B_3 \oplus B_2$$

$$P_2 = B_4 \oplus B_3 \oplus B_1$$

$$P_1 = B_4 \oplus B_2 \oplus B_1$$

偶校验

(3) 校验位关系： $B_4 \oplus B_3 \oplus B_2 \oplus P_3 = 0$

（偶校验） $B_4 \oplus B_3 \oplus B_1 \oplus P_2 = 0$

$B_4 \oplus B_2 \oplus B_1 \oplus P_1 = 0$

无误时，校验关系成立；

有误时：校验关系不成立，有1个或1个以上个错；
也可能校验关系依然成立，但仍有1个以上的错误，如B_3、B_2、B_1同时出错。
但是出现1个以上的错的可能（概率）很小。

举例：误码率 P<1，例如$P=10^{-3}$（即传1000个码元时一个码元出错）。那么在一个码长为n=7的码字中刚好发生一个错误的概率是$P_7(1)=7\times10^{-3}$，发生2个码元错误的概率$P_7(2)=2.1\times10^{-5}$，发生3个码元错误的概率$P_7(3)=3.5\times10^{-8}$。

（$P_n(r)=C_n^r P^r(1-P)^{n-r}$）

表. “8421”海明码

位序	7	6	5	4	3	2	1
N .	B_4	B_3	B_2	P_3	B_1	P_2	P_1
0	0	0	0	0	0	0	0
1	0	0	0	0	1	1	1
2	0	0	1	1	0	0	1
3	0	0	1	1	1	1	0
4	0	1	0	1	0	1	0
5	0	1	0	1	1	0	1
6	0	1	1	0	0	1	1
7	0	1	1	0	1	0	0
8	1	0	0	1	0	1	1
9	1	0	0	1	1	0	0

对传输后的海明码进行检错和校错：

(3) 校验和：$S_3 = B_4 \oplus B_3 \oplus B_2 \oplus P_3$

$S_2 = B_4 \oplus B_3 \oplus B_1 \oplus P_2$

$S_1 = B_4 \oplus B_2 \oplus B_1 \oplus P_1$

① 当$S_3\,S_2\,S_1 = 0$时，接收到的信息是正确的；

② 当$1 \leqslant S_3\,S_2\,S_1 \leqslant 7$时，则$S_3\,S_2\,S_1$所表示的二进制值便是出错的那一位的位序值。

例：接收到的海明码为：

7	6	5	4	3	2	1
B_4	B_3	B_2	P_3	B_1	P_2	P_1
0	1	0	1	0	1	0

则 $S_3\,S_2\,S_1$=110，表示第6位(B_3)出错，改0为1。

表. 出错位的确定

$S_3 =$	$B_4 \oplus$	$B_3 \oplus$	$B_2 \oplus$	P_3				
$S_2 =$	$B_4 \oplus$	B_3			$\oplus B_1 \oplus$	P_2		
$S_1 =$	B_4		$\oplus B_2$		$\oplus B_1$		$\oplus P_1$	
$S_3 S_2 S_1$	111	110	101	100	011	010	001	000
出错位序列	7	6	5	4	3	2	1	
出错位	B_4	B_3	B_2	P_3	B_1	P_2	P_1	

1. 每个校验位P必分布在2^k位上，使其仅在一个校验和S中出现;
2. 信息位B分布在非2^k位上，使其在一个以上的校验和S中出现;
3. 若传送后海明码中的某一位出错，则将影响它所在的校验和S_i，故能得到它的位序值，即可实现其单错的定位和校错。

问题：① 海明码的位序可以改变吗？ 练习

② 若信息位是八位二进制代码，其海明校验码怎样编码？

③ 若信息位是n位二进制代码，则需要增加多少位校验位，才能组成能够检测和校正一位错的海明校验码呢？

设：信息位n位，校验位k位

则 $(2^k-1)-k \geqslant n$ 或 $(2^k-1) \geqslant n+k$

校验位数k	1	2	3	4	5	6	7	8
最大信息位数n	0	1	4	11	26	57	120	247
海明码位数 (2^k-1)	1	3	7	15	32	63	127	255