



清华大学  
Tsinghua University

# 第一单元 第三讲

## 数据表示及检错纠错

刘卫东

计算机科学与技术系

# 内容提要



- ❖ 数据表示的需求
- ❖ 逻辑型数据表示
- ❖ 字符的表示
- ❖ 整数的表示
- ❖ 检错纠错码

# 计算机是什么？



- ✚ 一种高速运行的电子设备
- ✚ 用于进行数据的计算
- ✚ 可接受输入信息
- ✚ 根据用户要求对信息进行加工
- ✚ 输出结果

❏ A calculating machine, esp. an automatic electronic device for performing mathematic or logical operations; freq. with defining word prefixed, as analogue, digital, electronic computer.

— Oxford English Dictionary

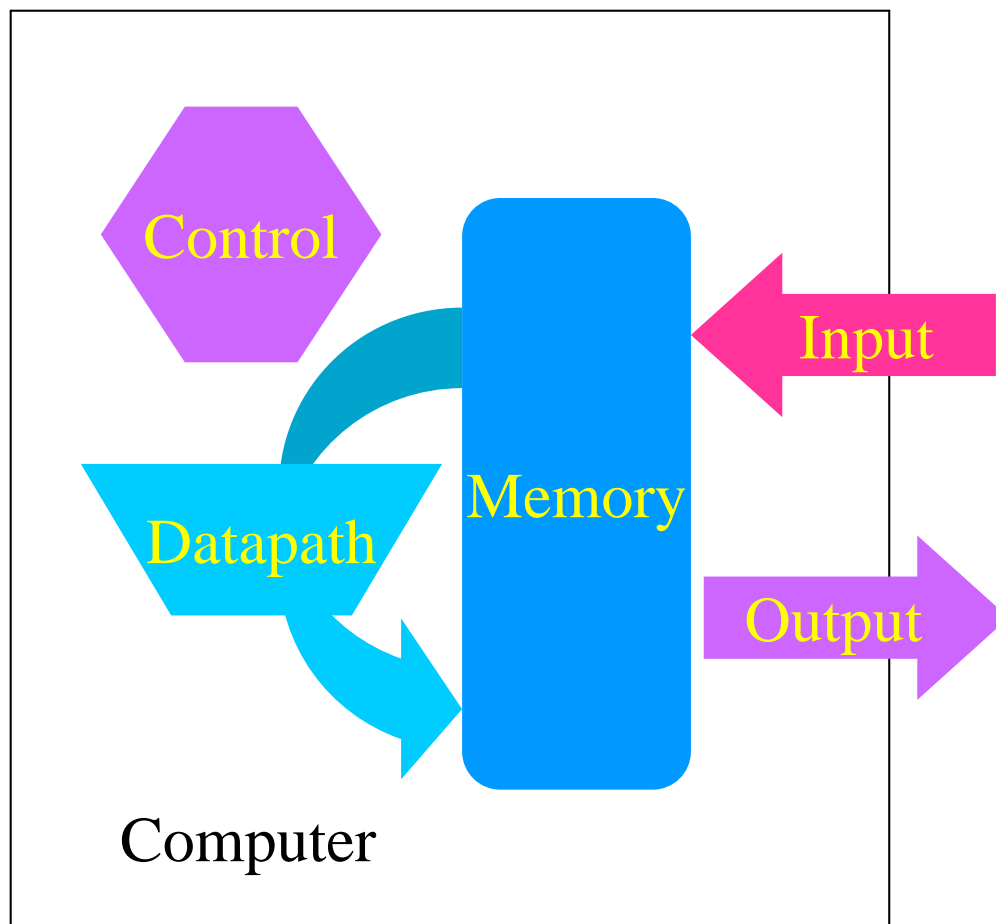
- ❖ Computer programs (also **software programs**, or just **programs**) are **instructions** for a **computer**. A computer requires programs to function, and a computer program does nothing unless its instructions are executed by a **central processor**. Computer programs are either **executable** programs or the **source code** from which executable programs are derived (e.g., **compiled**).

- ❖ 程序员和计算机进行交互的语言

- ❖ 计算机程序分类

- ❑ 高级语言
- ❑ 汇编语言
- ❑ 机器语言

# 计算机运行机制



- ✿ Datapath: 完成算术和逻辑运算，通常包括其中的寄存器。
- ✿ Control: CPU的组成部分，它根据程序指令来指挥datapath，memory以及I/O运行，共同完成程序功能。
- ✿ Memory: 存放运行时程序及其所需要的数据的场所。
- ✿ Input: 信息进入计算机的设备，如键盘、鼠标等。
- ✿ Output: 将计算结果展示给用户的设备，如显示器、磁盘、打印机、喇叭等。

# 程序设计举例(1)



汇 **0110100100000001**

LI R1 1 ;将R1寄存器赋值为1  
LI R2 1 ;将R2寄存器赋值为1  
LI R3 80 ;将R3寄存器赋值为80h  
SLL R3 R3 0 ;R3逻辑左移8位为8000h  
LI R4 9 ;将R4寄存器赋值为9, 规定循环次数为9  
SW R3 R1 0 ;将R1的值写入[R3+0]内存处  
SW R3 R2 1 ;将R2的值写入[R3+1]内存处  
ADDU R1 R2 R1 ;R1=R1+R2  
ADDI **0100110011111111**  
ADDIU R4 FF ;R4=R4-1  
BNEZ R4 F9 ;跳转到指令 (SW R3 R1 0) 处, F9为偏移量-7

# 程序设计举例(2)



;WELCOME

MFPC R7

ADDIU R7 0x0003

NOP

B TESTW

LI R6 0x00BF

SLL R6 R6 0x0000

LI R0 0x004F

SW R6 R0 0x0000

NOP

MFPC R7

ADDIU R7 0x0003

NOP

B TESTW

LI R6 0x00BF

SLL R6 R6 0x0000

LI R0 0x004B

SW R6 R0 0x0000

NOP

MFPC R7

ADDIU R7 0x0003

NOP

B TESTW

LI R6 0x00BF

SLL R6 R6 0x0000

LI R0 0x000A

SW R6 R0 0x0000

NOP

MFPC R7

ADDIU R7 0x0003

NOP

B TESTW

LI R6 0x00BF

SLL R6 R6 0x0000

LI R0 0x000D

SW R6 R0 0x0000

NOP

TESTW:

NOP

LI R6 0x00BF

SLL R6 R6 0x0000

ADDIU R6 0x0001

LW R6 R0 0x0000

LI R6 0x0001

AND R0 R6

BEQZ R0 TESTW

NOP

JR R7

NOP

# 数据编码与表示



## ❖ 需要在计算机中表示的对象

- ❑ 程序、整数、浮点数、字符（串）、逻辑值
- ❑ 通过编码表示

## ❖ 表示方式

- ❑ 用数字电路的两个状态表示，存放在机器字中
- ❑ 由上一层的抽象计算机来识别不同的内容

## ❖ 编码原则

- ❑ 少量简单的基本符号
- ❑ 一定的规则
- ❑ 表示大量复杂的信息
- ❑ 计算性能/存储空间



# 编码表示



## ❖ 基本元素

- ❖ 0、1 两个基本符号

## ❖ 字符

- ❖ 26 字母  $\Rightarrow$  5 位

- ❖ 大/小写 + 其它符号  $\Rightarrow$  7 bits (in 8)

- ❖ 世界上其他语言 的文字  $\Rightarrow$  16 bits (unicode)

## ❖ 无符号整数 ( $0, 1, \dots, 2^{n-1}$ )

## ❖ 逻辑值

- ❖  $0 \rightarrow \text{False}$ ,  $1 \Rightarrow \text{True}$

## ❖ 颜色

## ❖ 位置 / 地址 / 指令

## ❖ 但 $n$ 位只能代表 $2^n$ 个不同的对象

# 逻辑型数据



## 逻辑型数据

True、真

False、假

## 数据表示

1

0

## 数据运算

与、或、非

X	Y	X与Y	X或Y	X的非
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

# 字符型数据



## ❖ 重要的人机界面

- ❑ 由符号组成
- ❑ 为每个符号进行编码，由输入/输出设备进行转换
- ❑ 一般以字符串的形式在计算机存储器中存放

## ❖ 字符集编码标准

- ❑ 主机和设备、主机之间进行信息交互的基础
  - ◆ ASCII
  - ◆ UNICODE
  - ◆ UTF-8

# ASCII字符编码



- ❖ American Standard Code for Information Interchange
- ❖ 采用7位二进制编码，占用一个字节
- ❖ 表示128个西文字符



# ASCII 码字符集

<b>L \ H</b>	<b>000</b>	<b>001</b>	<b>010</b>	<b>011</b>	<b>100</b>	<b>101</b>	<b>110</b>	<b>111</b>
<b>0000</b>	<b>NUL</b>	<b>DLE</b>	<b>SP</b>	<b>0</b>	<b>@</b>	<b>P</b>	<b>`</b>	<b>p</b>
<b>0001</b>	<b>SOH</b>	<b>DC1</b>	<b>!</b>	<b>1</b>	<b>A</b>	<b>Q</b>	<b>a</b>	<b>q</b>
<b>0010</b>	<b>STX</b>	<b>DC2</b>	<b>"</b>	<b>2</b>	<b>B</b>	<b>R</b>	<b>b</b>	<b>r</b>
<b>0011</b>	<b>ETX</b>	<b>DC3</b>	<b>#</b>	<b>3</b>	<b>C</b>	<b>S</b>	<b>c</b>	<b>s</b>
<b>0100</b>	<b>EOT</b>	<b>DC4</b>	<b>\$</b>	<b>4</b>	<b>D</b>	<b>T</b>	<b>d</b>	<b>t</b>
<b>0101</b>	<b>ENG</b>	<b>NAK</b>	<b>%</b>	<b>5</b>	<b>E</b>	<b>U</b>	<b>e</b>	<b>u</b>
<b>0110</b>	<b>ACK</b>	<b>SYN</b>	<b>&amp;</b>	<b>6</b>	<b>F</b>	<b>V</b>	<b>f</b>	<b>v</b>
<b>0111</b>	<b>BEL</b>	<b>ETB</b>	<b>'</b>	<b>7</b>	<b>G</b>	<b>W</b>	<b>g</b>	<b>w</b>
<b>1000</b>	<b>BS</b>	<b>CAN</b>	<b>(</b>	<b>8</b>	<b>H</b>	<b>X</b>	<b>h</b>	<b>x</b>
<b>1001</b>	<b>HT</b>	<b>EM</b>	<b>)</b>	<b>9</b>	<b>I</b>	<b>Y</b>	<b>i</b>	<b>y</b>
<b>1010</b>	<b>LF</b>	<b>SUB</b>	<b>*</b>	<b>:</b>	<b>J</b>	<b>Z</b>	<b>j</b>	<b>z</b>
<b>1011</b>	<b>VT</b>	<b>ESC</b>	<b>+</b>	<b>;</b>	<b>K</b>	<b>[</b>	<b>k</b>	<b>{</b>
<b>1100</b>	<b>FF</b>	<b>FS</b>	<b>,</b>	<b>&lt;</b>	<b>L</b>	<b>\</b>	<b>l</b>	<b> </b>
<b>1101</b>	<b>CR</b>	<b>GS</b>	<b>-</b>	<b>=</b>	<b>M</b>	<b>]</b>	<b>m</b>	<b>}</b>
<b>1110</b>	<b>SO</b>	<b>RS</b>	<b>.</b>	<b>&gt;</b>	<b>N</b>	<b>↑</b>	<b>n</b>	<b>~</b>
<b>1111</b>	<b>SI</b>	<b>US</b>	<b>/</b>	<b>?</b>	<b>O</b>	<b>←</b>	<b>o</b>	<b>DEL</b>

注：H 表示高 3 位，L 表示低 4 位。

# UNICODE编码



- ❖ 使用16位表示一个字符，可以表示65536个字符
- ❖ 将整个编码空间划分为块，每块为16的整数倍，按块进行分配。
- ❖ 保留6400个码点供本地化使用。
- ❖ 依然无法覆盖所有字符。

# UTF-8编码



字符位数	字节1	字节2	字节3	字节4	字节5	字节6
7	0ddddddd					
11	110dddddd	10ddddddd				
16	1110dddd	10ddddddd	10ddddddd			
21	11110ddd	10ddddddd	10ddddddd	10ddddddd		
26	111110dd	10ddddddd	10ddddddd	10ddddddd	10ddddddd	
31	1111110d	10ddddddd	10ddddddd	10ddddddd	10ddddddd	10ddddddd

- ❖ 变长字符编码，提高存储空间利用率
- ❖ 字符长度由首字节确定
- ❖ 字符首字节外，均以“10”开始，可自同步
- ❖ 可扩展性强
- ❖ 成为互连网上占统治地位的字符集

# 数值型数据表示



## ❖ 定点数

- ❖ 小数点位置固定
- ❖ 整数
- ❖ 定点小数

## ❖ 浮点数

- ❖ 小数点位置浮动



# 数值范围和数据精度



## ❖ 数值范围

数值范围是指一种类型的数据所能表示的最大值和最小值；

## ❖ 数据精度

通常指实数所能给出的有效数字位数；对浮点数来说，精度不够会造成误差，误差大量积累会出问题。

## ❖ 机内处理

数值范围与数据精度概念不同。在计算机中，它们的值与用多少个二进制位表示某种类型的数据，以及怎么对这些位进行编码有关。

# 整数的二进制表示



- ❖ 二进制的两个状态0和1
- ❖  $n$ 位可得到 $2^n$ 种组合，可表示 $2^n$ 个整数
- ❖ 那么，如何来表示呢？

机内表示	真值
0	0
01	1
10	2
11	3
100	4
101	5
...	...
$1\{n\}$	$2^n-1$

# 评价标准



## ❖ 无符号数

- ❖ 表示范围
- ❖ 直观
- ❖ 便于算术运算的实现

## ❖ 有符号数

- ❖ 表示范围
- ❖ 直观
- ❖ 正、负数平衡
- ❖ 便于算术运算的实现

# 进位记数法



## 进位记数法

$$N = \sum_{i=m}^{-k} D_i r^i$$

- ❖ N表示某个数值
- ❖ r是这个数制的基
- ❖ i表示这些符号排列的位号
- ❖  $D_i$ 是位号为i的位上的一个符号
- ❖  $r^i$ 是位号为i的位上的一个1代表的值

## 常用的进制

- ❖ 十进制、二进制、八进制、十六进制

# 数制与进位记数法



## ❖ 二进制

❖  $r = 2$ , 基本符号: 0 1

## ❖ 八进制

❖  $r = 8$ , 基本符号: 0 1 2 3 4 5 6 7

## ❖ 十进制

❖  $r = 10$ , 基本符号: 0 1 2 3 4 5 6 7 8 9

## ❖ 十六进制

❖  $r = 16$ , 基本符号: 0 1 2 3 4 5 6 7 8 9 A B C D E F

## ❖ 计算机采用二进制

# 数制转换



把二进制数转换为十进制数，

累加二进制数中全部数值为 1 的那些位的位权

$$\begin{aligned}(1101.1100)_2 &= (1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0)_{10} \\ &\quad + (1 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 0 \times 2^{-4})_{10} = (13.75)_{10}\end{aligned}$$

把二进制数转换成八或十六进制数时，从小数点向左和向右把每 3 或者 4 个二进制位分成一组，直接写出每一组所代表的数值，小数点后不足位数补 0。

$$(1101.1001)_2 = (\text{D}.9)_{16} = (15.44)_8, \text{ 而不是 } (15.41)_8$$

# 数制转换



## 二进位数和十进制数之间的转换方法

**二进制**： $r = 2$ ，基本符号：**0 1**

**十进制**： $r = 10$ ，基本符号：**0 1 2 3 4 5 6 7 8 9**

求二进制数所对应的十进制数值，可通过进位记数公式来计算，即把取值为1的数位的位权累加。

把十进制数转换为二进制，**对整数部分通过除2取余数来完成**，**对小数部分通过乘2取整数来完成**。

2		13	-----	1	低位
2		6	-----	0	
2		3	-----	1	
2		1	-----	1	高位
		0			

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

		0.76	× 2	
1		0.52	× 2	高位
1		0.04	× 2	
0		0.08	× 2	
0		0.16		低位

$$(0.76)_{10} = (0.1100)_2$$

# 二进制整数的进位表示法



✿ 具体到n位无符号二进制整数，如

- ❏  $b_{n-1}b_{n-2}\cdots b_1b_0$

- ❏ 其中， $b_i$ 为0或者1

- ❏ 表示的值为：
$$N = \sum_{i=0}^{n-1} b_i 2^i$$

- ❏ 可表示的范围为 $0 \sim 2^n - 1$ ，共 $2^n$ 个数

✿ 如果要表示有符号的整数呢？

- ❏ 需要有1位来表示符号

  - ◆ 最高位

  - ◆ 0表示正数、1表示负数

- ❏ 其他位表示数据



# 原码、反码和补码



编码	原码	反码	补码
000	0	0	0
001	1	1	1
010	2	2	2
011	3	3	3
100	-0	-3	-4
101	-1	-2	-3
110	-2	-1	-2
111	-3	-0	-1

负数表示形式：

原码 (Sign Magnitude) : 符号位||数的绝对值

反码 (One's Complement) : 符号位||数值按位求反

补码 (Two's Complement) : 反码的最低位+1

# 整数编码的定义



$x$  为真值     $n$  为整数的位数

$$[x]_{\text{原}} = \begin{cases} x & 2^n > x \geq 0 \\ 2^n - x & 0 \geq x > -2^n \end{cases}$$

$$[x]_{\text{补}} = \begin{cases} x & 2^n > x \geq 0 \\ 2^{n+1} + x & 0 \geq x \geq -2^n \pmod{2^{n+1}} \end{cases}$$

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



# 补码的性质

## 补码与真值的对应

补码求真值

$$N = -b_{n-1} * 2^{n-1} + \sum_{i=0}^{n-2} b_i 2^i$$

真值求补码：

◆ 正数的补码是绝对值原码

◆ 负数的补码是绝对值原码按位求反后，再在最低位加1

## 补码的加法运算

加法运算：符号位和数据位同样计算

$$[x+y]_{\text{补}} = [x]_{\text{补}} + [y]_{\text{补}}$$

# 补码的性质



## ✚ $[x]_{\text{补}}$ 与 $[-x]_{\text{补}}$

▣  $[x]_{\text{补}}$  连同符号位在内，逐位求反，再在最低位加1，即可得  $[-x]_{\text{补}}$

▣ 当  $X \geq 0$  时，...

▣ 当  $X < 0$  时，...

## ✚ 补码减法

▣  $[x-y]_{\text{补}} = [x+(-y)]_{\text{补}} = [x]_{\text{补}} + [-y]_{\text{补}}$

## ✚ 补码的乘法

# 补码表示中的符号位扩展



由  $[X]_{\text{补}}$  求  $[X/2]_{\text{补}}$  的方法

原符号位不变，且符号位与数值位均右移一位

例如，

$$[X]_{\text{补}} = \underline{10010} \quad \text{则} \quad [X/2]_{\text{补}} = \underline{110010}$$

不同位数的整数补码相加减时，要进行符号扩展

位数少的补码数的符号位向左扩展，

一直扩展到与另一数的符号位对齐。

$$\begin{array}{r} 0101010111000011 \\ + 111111110011100 \\ \hline 0101010101011111 \end{array}$$

$$\begin{array}{r} 0101010111000011 \\ + 0000000000011100 \\ \hline 0101010111011111 \end{array}$$



# 大端和小端

## 大端存储

- 数据的低位保存在内存的高地址字节中
- 数据的高位保存在内存的低地址字节中
- 例如：32位整数“12345678”保存在内存4000起始地址

内存地址	4000	4001	4002	4003
------	------	------	------	------

存放数据	12	34	56	78
------	----	----	----	----

## 小端存储

- 数据的高位保存在内存的高地址字节中
- 数据的低位保存在内存的低地址字节中
- 例如：32位整数“12345678”保存在内存4000起始地址

内存地址	4000	4001	4002	4003
------	------	------	------	------

存放数据	78	56	34	12
------	----	----	----	----

# 原反补码表示小结



正数的原码、反码、补码表示均相同，  
符号位为 0，数值位同数的真值。

零的原码和反码均有 2 个编码，补码只 1 个码

负数的原码、反码、补码表示均不同，  
符号位为 1，数值位：  
原码为数的绝对值  
反码为每一位均取反码  
补码为反码再在最低位+1

由  $[X]_{\text{补}}$  求  $[-X]_{\text{补}}$ ：每一位取反后再在最低位+1

# 检错纠错码



❏ 数据或编码在存储、传输等过程中可能出错

❏ 如何判断是否已经出错？

◆ 比较：与所有正确的编码进行比较

◆ 特征：检验是否存在某些特征

❏ 发现错误后能否自动纠正？

❏ 计算机中的数据如何进行检错？

❏ 纠错呢？



# 检错纠错码



- ❖ 使编码具有某种特征，通过检查这种特征是否存在来判断编码是否正确
- ❖ 出错时，如果还能指出是哪位出错，则可纠正错误
  - ❖ 编码
  - ❖ 检查
  - ❖ 出错后纠正

**码距（最小码距）的概念：**是指任意两个合法码之间至少有几个二进制位不相同。例如：

仅有一位不同，称最小码距为 1，例如用 4 位二进制表示 16 种状态，则 16 种编码都用到了，此时**码距为 1**，就是说，任何一个编码状态的四位码中的一位或几位出错，都会变成另一个合法码，此时**无检错能力**。

若用 4 个二进制位表示 8 种合法状态，就可以只用其中的 8 个编码来表示之，而把另 8 种编码作为非法编码，此时可以使合法码的**码距为 2**。**如果**一个码字中的任何一位出错后都会成为非法码，则它就**有了发现一位出错的能力**。

**合理增大码距，能提高发现错误的能力，但表示一定数量的合法码所使用的二进制位数要变多，增加了电子线路的复杂性和数据存储、数据传送的数量。**

# 常用检错纠错码



## 三种常用的检错纠错码：

奇偶校验码：用于并行数据传送中

汉明校验码：用于并行数据传送中

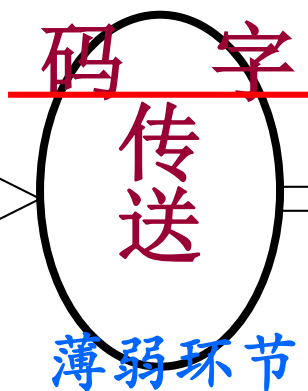
循环冗余校验码：用于串行数据传送中

检错纠错过程：

原始数据

编码过程

形成校验位的值  
加进特征



结果数据

译码过程

检查收到的码字  
发现 / 改正错误

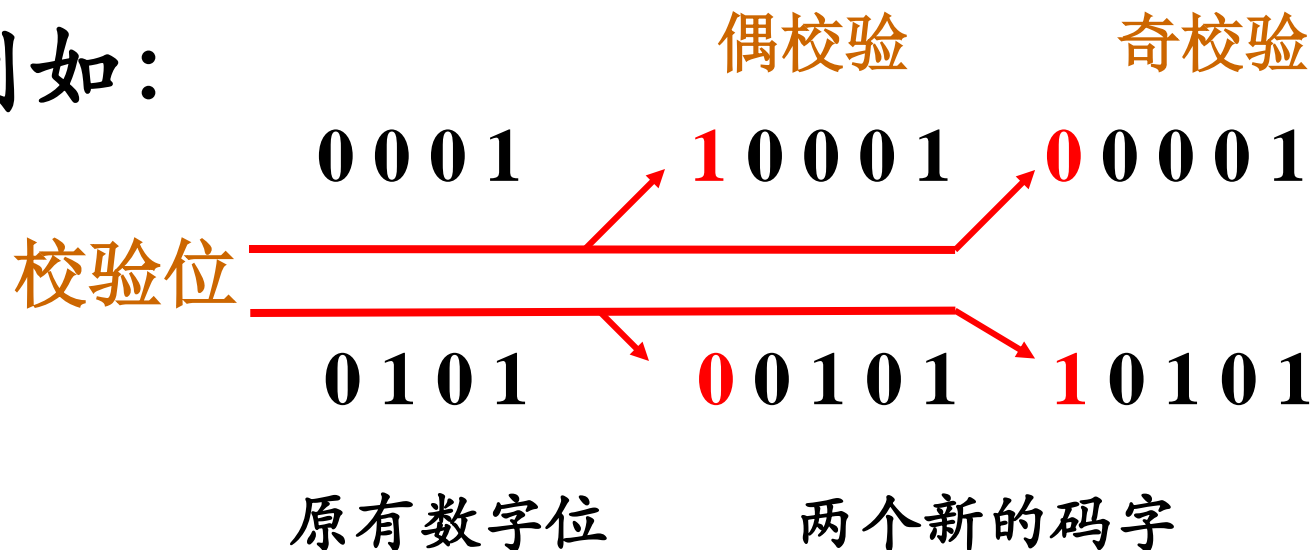
# 奇偶校验码



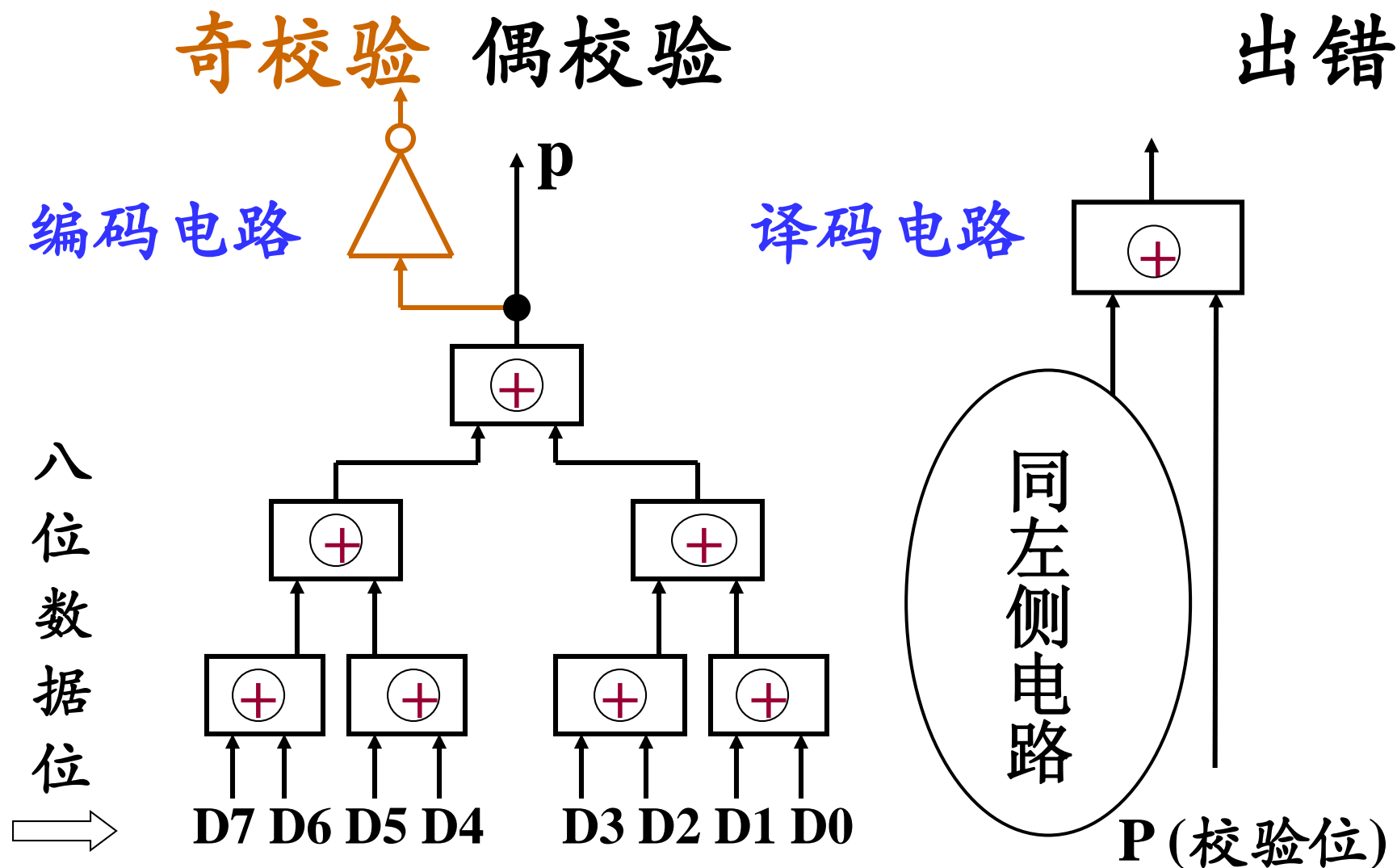
## 用于并行码检错

原理：在  $k$  位数据码之外增加 1 位校验位，  
使  $K+1$  位码字中取值为 1 的位数总保持  
为 偶数（偶校验）或 奇数（奇校验）。

例如：



# 奇偶校验码的实现电路





# 汉明校验码

用于多位并行数据检错纠错处理

实现：为  $k$  个数据位设立  $r$  个校验位，使  $k+r$  位的码字同时具有这样两个特性：

- ① 能发现并改正  $k+r$  位中任何一位出错；
- ② 能发现  $k+r$  位中任何二位同时出错，  
但已无法改正。

$k$  与  $r$  之间应该满足什么样的关系？

# 汉明码的编码方法



合理地用  $k$  位数据位形成  $r$  个校验位的值，即保证用  $k$  个数据位中不同的数据位组合来形成每个校验位的值，使任何一个数据位出错时，将影响  $r$  个校验位中不同的校验位组合起变化。这样一来，就可以通过检查是哪种校验位组合起了变化，来推断是哪个数据位错造成的，对该位求反则实现纠错。

有时两位出错与某种情况的一位出错对校验位组合的影响相同，必须加以区分与解决。

位数  $r$  和  $k$  的关系： $2^r \geq k+r+1$ ，即用  $2^r$  个编码分别表示  $k$  个数据位、 $r$  个校验位中哪一位错，都不错

$2^{r-1} \geq k+r$ ，用  $r-1$  位校验码为出错位编码，再单独设一位用于区分 1 位还是 2 位同时出错，更实用



# 汉明码的实现方案

例如:  $k=3, r=4$

D3	D2	D1	P4	P3	P2	P1
1	1	1	1	1	1	1
1	1	0	0	1	0	0
1	0	1	0	0	1	0
0	1	1	0	0	0	1

$\oplus$  表示异或

$$P1 = D2 \oplus D1$$

$$P2 = D3 \oplus D1$$

$$P3 = D3 \oplus D2$$

$$P4 = P3 \oplus P2 \oplus P1 \oplus D3 \oplus D2 \oplus D1$$

编码方案

译码方案

$$S1 = P1 \oplus D2 \oplus D1$$

$$S2 = P2 \oplus D3 \oplus D1$$

$$S3 = P3 \oplus D3 \oplus D2$$

$$S4 = P4 \oplus P3 \oplus P2 \oplus P1 \oplus D3 \oplus D2 \oplus D1$$



# 汉明码的实现方案



如何分配不同的数据位组合来形成每个校验位的值

<b>P1</b>	<b>P2</b>	<b>D1</b>	<b>P3</b>	<b>D2</b>	<b>D3</b>	<b>P4</b>
<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	

编码方案

(一) 准备工作:

- (1) 从1~6按次序排列数据位、校验位,
- (2) 将校验位P1、P2、P3依次安排在2的幂次方位。
- (3) P4为总校验位, 暂不考虑。

# 汉明码的实现方案



如何分配不同的数据位组合来形成每个校验位的值

<b>P1</b>	<b>P2</b>	<b>D1</b>	<b>P3</b>	<b>D2</b>	<b>D3</b>	<b>P4</b>
1	2	3	4	5	6	

编码方案

(二) 为各校验位分配数据位组合:

(1) 看数据位的编号分别为3、5、6，它们是校验位编号的组合:

$$3=1+2、5=1+4、6=2+4$$

(2) 1出现在3和5中，则P1负责对D1和D2进行校验。

(3) 2出现在3和6中，则P2负责对D1和D3进行校验。

(4) 4出现在5和6中，则P3负责对D2和D3进行校验。



# 汉明码的实现方案

如何通过一张表分配不同的数据位组合来形成每个校验位的值

<b>P1</b>	<b>P2</b>	<b>D1</b>	<b>P3</b>	<b>D2</b>	<b>D3</b>	<b>P4</b>	编码方案
1	2	3	4	5	6		

(三) 写出各校验位的编码逻辑表达式:

(1) 结果是:

$$P1 = D2 \oplus D1; \quad P2 = D3 \oplus D1; \quad P3 = D3 \oplus D2$$

(2) 用其他各校验位及各数据位进行异或运算求校验位 P4 的值, 用于区分无错、奇数位错、偶数位错 3 种情况

$$\text{总校验位 } P4 = P3 \oplus P2 \oplus P1 \oplus D3 \oplus D2 \oplus D1$$

# 汉明码的实现方案



译码方案是：

对接收到数据位再次编码，用得到的结果和**传送过来的校验位**的值相比较，结果分别用  $S_4 \sim S_1$  表示，二者相同表明无错，不同是有 1 位错了。

排查是哪一位错了，看  $S_4 \sim S_1$  这 4 位的编码值。

$\oplus$  : 异或

$$P_1 = D_2 \oplus D_1$$

$$P_2 = D_3 \oplus D_1$$

$$P_3 = D_3 \oplus D_2$$

编码方案

$$P_4 = P_3 \oplus P_2 \oplus P_1 \oplus D_3 \oplus D_2 \oplus D_1$$

译码方案

$$S_1 = P_1 \oplus D_2 \oplus D_1$$

$$S_2 = P_2 \oplus D_3 \oplus D_1$$

$$S_3 = P_3 \oplus D_3 \oplus D_2$$

$$S_4 = P_4 \oplus P_3 \oplus P_2 \oplus P_1 \oplus D_3 \oplus D_2 \oplus D_1$$

# 汉明码的应用实例



如已有数据为 **110**，编码为: P1P2**1**P3**10**P4则有:

P1=0, P2=1 P3=1, P4=0

请看如下 3 种情况:

无错,                      单独 1 位错,                      2 位同时错

若无错, 则

S4 S3 S2 S1=0000

**4 位 S 全为 0**

若仅 D1 错, 则

S4 S3 S2 S1=1011

**S3S2S1 不为 000**

**其中 S4 必为 1**

若P2 D1错, 则

S4 S3 S2 S1=0001

**其中 S4 必为 0,**

**S3S2S1 不为 000**

# 检错纠错码小结



(1)  $k$  位码有  $2^k$  个编码状态，全用于表示合法码，则任何一位出错，均会变成另一个合法码，不具有检错能力。

(2) 从一个合法码变成另一个合法码，至少要改变几位码的值，称为最小码距 (码距)，码距和编码方案将决定其检错就错能力。

奇偶校验码的码距为 2，

汉明码的码距为 4。

# 检错纠错能力



(3)  $k+1$  位码，只用其  $2^k$  个状态，可以使码距为 2，  
如果一个合法码中的一位错了，就成为非法码，  
通过检查码字的合法性，就得到检错能力，这就是奇偶校验码，只能发现1位错，不具备纠错能力。

(4) 对  $k$  位数据位，当给出  $r$  位校验位时，  
要发现并改正一位错，须满足如下关系：

$$\underline{2^r} \geq k + r + 1$$

要发现并改正一位错，也能发现两位错，则应：

$$\underline{2^{r-1}} \geq k + r$$

# 小结



## ❖ 数据表示

- ❖ 通过二进制编码表示数据
- ❖ 逻辑型
- ❖ 字符型
- ❖ 整数

## ❖ 检错和纠错

- ❖ 通过冗余的编码,使之满足某些规则,来检查编码在传输中是否发生错误,并进行纠正
- ❖ 检错纠错能力



# 阅读及思考



## 阅读

教材第3章

## 思考

- 原、反、补码的定义及实现算术运算的难易比较。
- 试证明补码的性质。
- 试推导汉明码校验位 $r$ 和数据位 $k$ 的关系。
- 补码算术运算如何用硬件来实现？