数字系统逻辑设计

- 邹逸飞
- yfzou@sdu.edu.cn

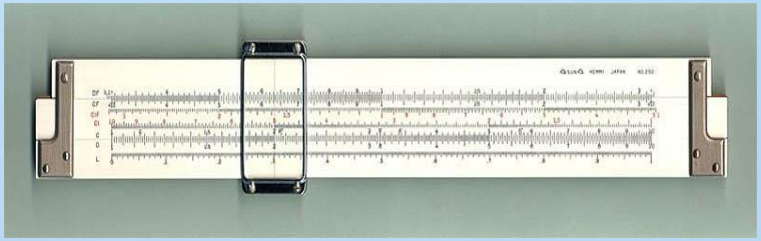
数字电路与模拟电路

- 所有电子电路中的工作信号，只有数字信号与模拟信号两种类型
- 数字信号是一种二值信号，一般用两个电平（高低电平）表示这两个值（逻辑0和逻辑1）
  - 两种逻辑体制：
    - 正逻辑：高电平为1，低电平为0
    - 负逻辑：低电平为1，高电平为0
- 工作于数字信号下的电子电路为数字电路
- 数字电路的特点
  - 用二进制数字信号表示自然界的物理量
- 数字电路的应用
  - 控制、测量、通讯、计算等

数字技术的由来

- 计算和器件的历史发展
  - 数字的出现
    - 数字在各个古代文明中都独立的存在
    - 数字都采用十进制数
    - 阿拉伯数字
  - 早期的计算用具
    - 最早的“计算机”-算盘
    - 计算尺（可实现对数）

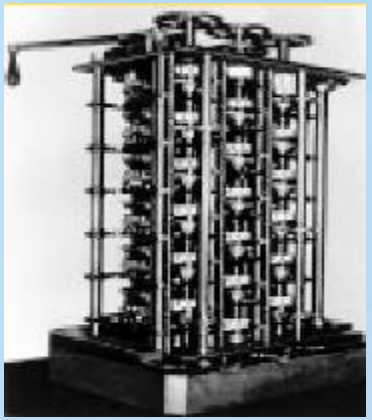
公元190年出现算盘

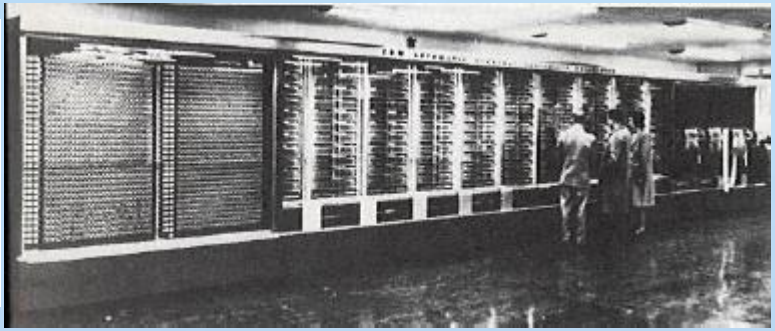


17世纪出现纳皮尔计算尺

数字技术的由来

- 计算和器件的历史发展
  - 早期的计算用具
    - 机械计算器具
      - 加法机、差分机（齿轮）
    - 机电计算机
      - 使用继电器设计自动计算器

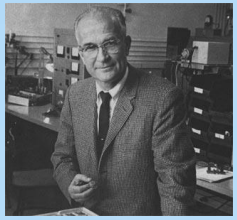




数字技术的由来

- 计算和器件的历史发展发展
  - 数字电子计算机
    - 第一代：约1946-1957

      电子真空管



威廉·肖克利，晶体管之父
1956年诺贝尔物理学奖

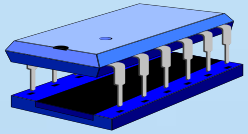
    - 第二代：约1957-1964

      晶体管

杰克·基尔比，集成电路之父
2000年诺贝尔物理学奖

    - 第三代：约1965-1972

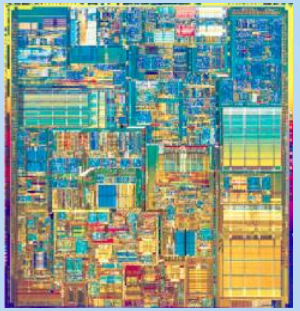
      中小规模集成电路



发明人：罗伯特·诺伊斯、杰克·基尔比

    - 第四代：约1972-现在？

      大规模、超大规模集成电路



摩尔定律

- Moore‘s Law：

  The number of transistors per integrated circuit would double every 18 month.（集成电路芯片上所集成的晶体管的数目，每隔18个月就翻一番。 即微处理器的性能每隔18个月提高一倍，而价格下降一倍）

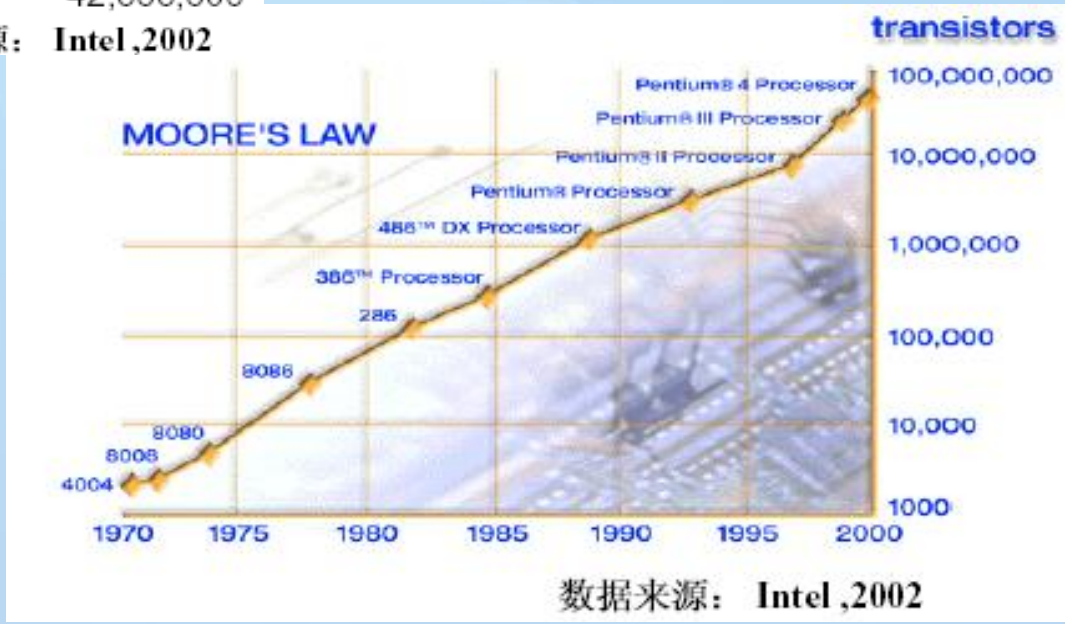
- 这个论断是在第一块平面集成电路产生4年以后的1965年做出的。
- 当时认为这个发展趋势将持续到1975年
  - 事实上，这个发展规律在目前仍是正确的。

Intel公司创建人之一戈登·摩尔

Intel微处理器的发展

微处理器	生产日期（年）	晶体管数量
4004	1971	2,250
8008	1972	2,500
8080	1974	5,000
8086	1978	29,000
286	1982	120,000
386™ processor	1985	275,000
486™ DX processor	1989	1,180,000
Pentium® processor	1993	3,100,000
Pentium II processor	1997	7,500,000
Pentium III processor	1999	24,000,000
Pentium 4 processor	2000	42,000,000

数据来源： Intel ,2002



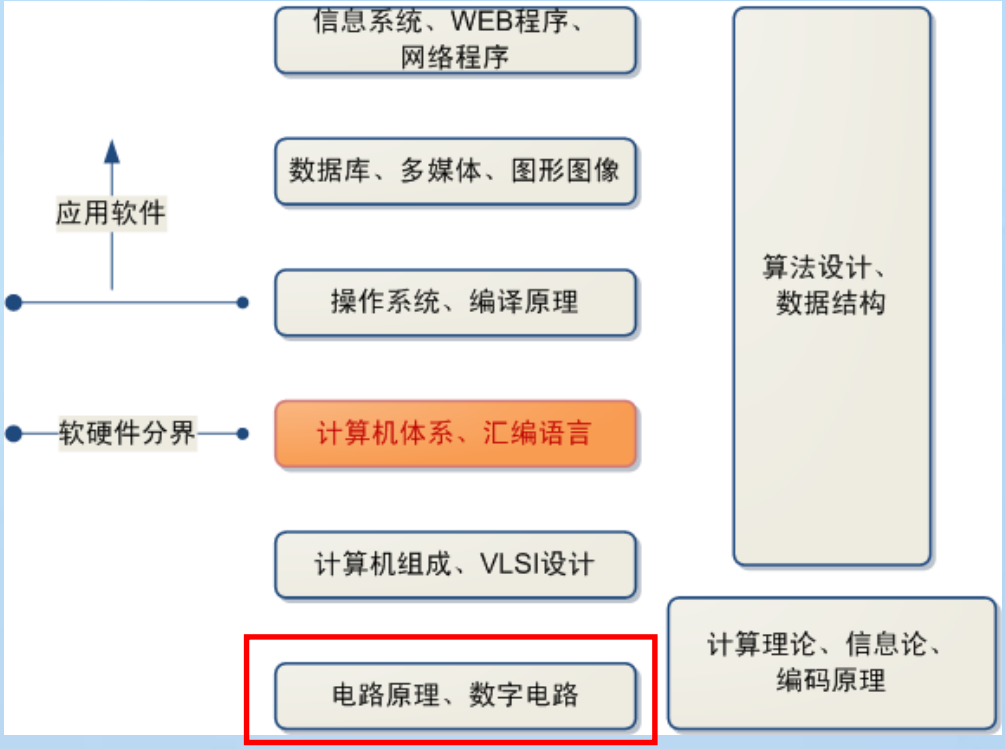
本书主要内容

- 数字逻辑和数字系统的基本概念
  - 数字逻辑：研究数值的逻辑加工和运算
  - 数字系统：对数字量进行加工、处理的系统
- 主要内容
  - 数制和编码
  - 逻辑代数基础
  - 组合逻辑和时序逻辑电路的分析和设计方法
  - 中大规模集成电路的应用

数字逻辑电路
- 组合逻辑
- 时序逻辑
  - 同步逻辑电路
  - 异步逻辑电路

课程性质与相关课程

- 数字逻辑 Digital Logic
  - 必修的专业基础课
- 后期课程
  - 计算机组成原理
  - 微机原理与接口技术
  - 嵌入式系统



主要教材：

《数字系统逻辑设计》

主编：王维华、曲兆瑞，山东大学出版社

参考教材：

1. 《数字逻辑与数字系统》

   白中英 主编，科学出版社

2. 《数字逻辑电路设计》

   王诚、刘卫东、宋佳兴，清华大学出版社

3. 数字电子技术基础

   阎石主编 人民教育出版社

- 第一章 数据信息的二进制表示

内容提要

- 1.1 进位记数制
- 1.2 带符号的二进制数表示
- 1.3 定点数和浮点数
- 1.4 编码
- 1.5 校验码

1.1 进位记数制

- 进位记数制是按一定的规则和符号表示数量的方法
- 生活中的数制
  - 六十进制：1小时=60分，1分=60秒
  - 十二进制：1英尺=12英尺，1年=12月
  - 十进制：符合人们的习惯

× 10 =

1.1.1 进位计数制的基本概念

- 进位记数制三要数：数码、基数、位权

  数码：每个数位上允许的数的集合

  基数：进制中允许每个数位上选用基本数码的个数

  位权：数码“1”在不同数位上代表的数值

例如：十进制

  数码：0 – 9十个数码

  基数：10，逢十进一，借一当十

  位权：第i位的权值为10^i

$(143.75)_{10}=1\times10^2+4\times10^1+3\times10^0+7\times10^{-1}+5\times10^{-2}$

任意十进制数 $N=\sum d_i\times10^i$

1.1.2 数字系统中常用的数制

1 二进制 Binary

例如：$(1001)_2$ 1001B

2 八进制 Octal

例如：$(317)_8$ 317Q

3 十六进制 Hexadecimal

例如：$(9A1)_{16}$ 9A1H

4 十进制 Decimal

例如：$(531)_{10}$ 531D

1.1.2.1 二进制

(1)二进制 Binary

数码：0和1两个数码

进位规则：逢二进一

例如：$(101.11)_2$

$=1\times2^2+0\times2^1+1\times2^0+1\times2^{-1}+1\times2^{-2}$

$=(5.75)_{10}$

任何二进制数 $D=\sum K_i\times 2^i$

表示范围：？

$0—(2^n-1)$

- **运算过程：**

	(被加数)	1	0	1	0	1	1	0	1
+	(加数)	0	0	1	1	1	0	0	1
	(进位)		1	1	1			1	
	(和)	1	1	1	0	0	1	1	0

	(被减数)	1	0	1	0	1	1	0	1
–	(减数)	0	0	1	1	1	0	0	1
	(借位)	1	1	1					
	(差)	0	1	1	1	0	1	0	0

(2) 八进制 Octal

数码：0, 1, 2, 3, 4, 5, 6, 7

进位规则：逢八进一

例如：

$$(23.71)_8 = 2\times8^1+3\times8^0+7\times8^{-1}+1\times8^{-2}$$

$$= (19.890625)_{10}$$

- 运算过程：

	(被加数)	2	5 5
+	(加数)	0	7 3
	(进位)	1	1
(和)		3	5 0

	(被减数)	2	5 5
-	(减数)	0	7 3
	(借位)	1	0
(差)		1	6 2

（3）十六进制数

- 特点：基数为16，有0-9和A(10)、B(11)、C(12)、D(13)、E(14)、F(15)共16个数码

  逢16进一，借一当16

  举例： $BF3CH=11\times16^3+15\times16^2+3\times16^1+12\times16^0$

  $=11\times4096+15\times256+3\times16+12\times1$

  $=48956D$

- 对于任意十六进制数

  $H=h_nh_{n-1}\cdots h_0.h_{-1}h_{-2}\cdots h_{-m}$可以表示为：

  $$(H)_{16}=h_n16^n+h_{n-1}16^{n-1}+\cdots+h_016^0+h_{-1}16^{-1}+\cdots+h_{-m}16^{-m}$$

  $$=\sum_{i=n}^{-m}b_i16^i$$

（4）不同进制表示16以内的数

十进制	二进制	八进制	十六进制
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

1.1.3 数制之间的转换

- 非十进制数转换为十进制数

  按权相加法：先将各位数码与权值相乘，再将各位的乘积值相加，得到十进制数

- 十进制数转换为任意进制数
  - 整数小数分别转换
    - 整数部分：*除基取余法*
    - 小数部分：*乘基取整法*
- 转换举例
  - 任意进制转为十进制
  - 十进制转为其他进制
  - 二进制、八进制与十六进制互转

(1) 任意进制转换为十进制

转换方法：

①按权相加法(掌握)

特点：比较直观，适于手算

②逐次乘基相加法

特点：适合于编程实现

例：按权相加法

$(1011)_2=1\times2^3+0\times2^2+1\times2^1+1\times2^0$

$=(11)_{10}$

例：$(0.1011)_2=1\times2^{-1}+0\times2^{-2}+1\times2^{-3}+1\times2^{-4}$

$=0.5+0.125+0.0625$

$=(0.6875)_{10}$

(2a)十进制整数转换为二进制整数

十进制数转换成等值的二进制数。

转换方法：

①除2取余法 P4(掌握)

②减权定位法

例：$(117)_{10}=(?)_2$

∴ $(117)_{10} = (1110101)_2$

或 117D = 1110101B

- 原理：二进制的基数为2，所以二进制中各个数字符号所表示的数值表示该数字符号乘以位权，而位权的大小是以基数为底，数字符号所处的位置序号为指数的整数次幂

(2b)十进制小数转换为二进制小数

转换方法：

①乘2取整法 P26(掌握)

②减权定位法

例：$(0.625)_{10}=(?)_2$

小数部分乘以2	整数部分	小数部分
0.625×2=1.25	1	0.25
0.25×2=0.5	0	0.5
0.5×2=1	1	0

注：1. *若出现乘积的小数部分一直不为“0”，根据计算精度的要求截取一定的位数即可；*

2. *一个十进制数不一定有对应的二进制数。*

- 十进制小数转换二进制小数的具体步骤：
  - 用2乘以十进制小数，将积的整数部分取出，再用2乘以余下的小数部分，再将积的整数部分取出，直至积的小数部分为0，或达到所要求的精度为止。
    - 如一个十进制小数B转换为二进制小数0.ab，按权展开应为$B=a2^{-1}+b2^{-2}$
    - 等式两边×2，得 $2B=a+b2^{-1}$
    - 由此，a就变成了整数部分.....

(3) 二/八/十六进制数的互换

转换方法：分组转换(掌握)

- 二进制->十六进制

  原则：四位二进制对应一位十六进制

  0011 0101 1011 1111

  ↓ ↓ ↓ ↓

  3 5 B F

  ∴ 0011010110111111B = 35BFH

- 十六进制->二进制

  A 1 9 C

  ↓ ↓ ↓ ↓

  1010 0001 1001 1100

  ∴ A19CH = 1010000110011100B

作业

- P26练习一
  - 1、2、3、4、9

内容提要

1.2 带符号数的二进制表示

- 真值
  - 用正、负符号加绝对值表示的数

例如：　+9的真值:+1001

　　-9的真值:-1001

- 机器数
  - 在计算机内部使用的、符号数码化的定长二进制数
  - 计算机硬件能够直接识别、处理

- 机器数的实现需要解决三个问题
  - 进制：只能采用二进制 Why?
  - 将符号为数字化
  - 采用什么编码方法表示数值

- 带符号数的编码方式
  - 原码表示(掌握)
  - 反码表示
  - 补码表示(重点)

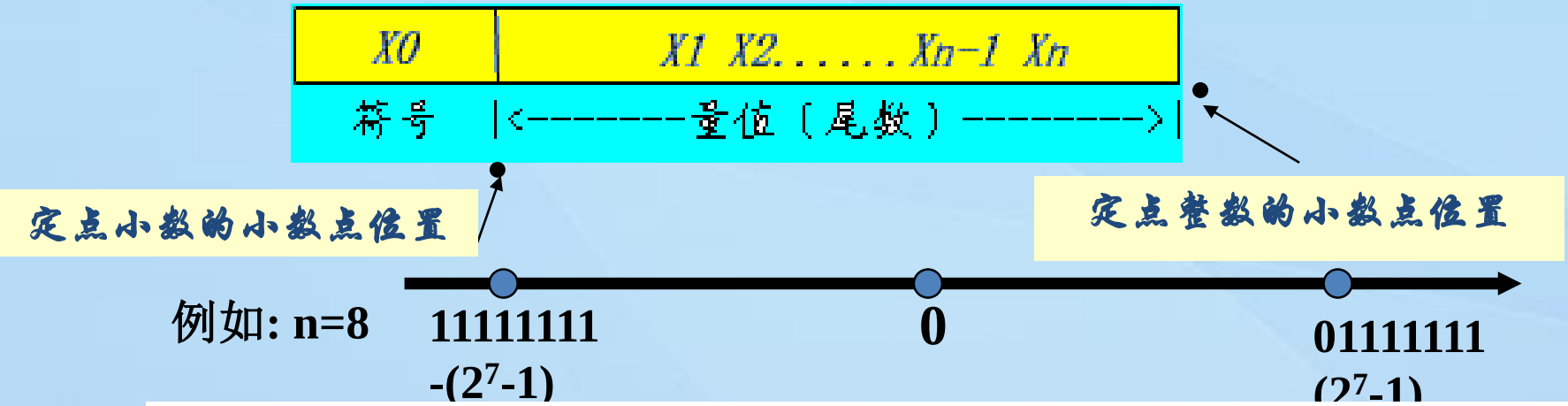
对于正数，三种表示方式一样，其区别在于负数的表示

1.2.2 原码表示法

- 原码(true code)表示法：符号位 + 数值

(1)表示定点整数 如下图

- n位原码表示范围： $-2^{(n-1)}-1<X<2^{(n-1)}-1$



定点小数的小数点位置

定点整数的小数点位置

例如: n=8

11111111
$-(2^7-1)$

0

01111111
(2^7-1)

例：n = 8bit

$[+3]_{原码}$ = 0 0000011 = 03H

$[-3]_{原码}$ = 1 0000011 = 83H

$[+0]_{原码}$ = 0 0000000 = 00H

$[-0]_{原码}$ = 1 0000000 = 80H

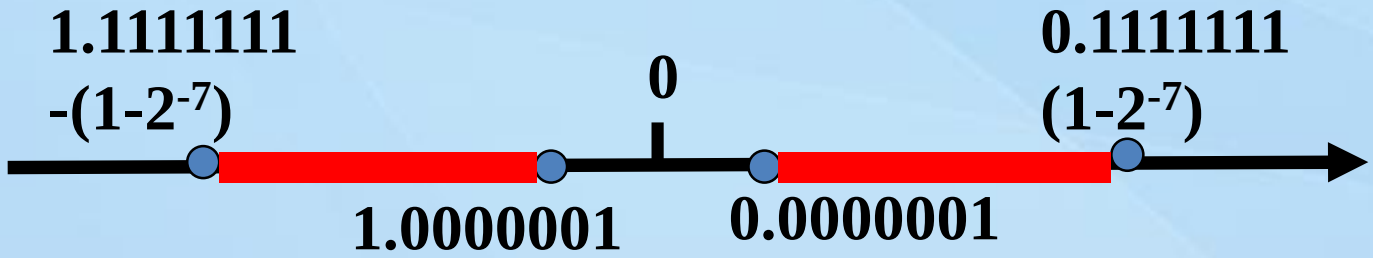
∴ 0 的表示不惟一

1.2.2 原码表示法

(2)表示定点小数（如上图）

- 范围：　正数部分 $-(1-2^{-(n-1)}) \sim -(2^{-(n-1)})$
　　负数部分 $(2^{-(n-1)}) \sim (1-2^{-(n-1)})$

例如: n=8



例：

$X=+0.1011$　　$[X]_{原}=0.1011$

$X=-0.1011$　　$[X]_{原}=1.1011$

1.2.2 原码性质

- 原码为符号位加数的绝对值，0正1负
  - 符号和数值无关
- 0可分+0和-0

  +0 为 00…0　　-0为 10…0
- 用原码做加减运算比较复杂，但乘除方便
  - 比较直观
  - 电路设计时，需要对最高位和其他位分别处理

1.2.3 补码表示法

补码（Two’s Complement）

正数的补码： 同原码

负数的补码： （1）写出与该负数相对应的正数的补码

（2）按位求反

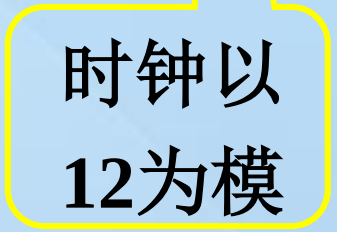
（3）末位加一

补的概念

- 时钟

逆时针 6 - 3 = 3 减法

顺时针 6 + 9 = 15 - 12 = 3 加法



可见 – 3 可用 + 9 代替

称 + 9 是 – 3 以 12 为模的 补数

记作 – 3 ≡ + 9 （mod 12）

同理 – 4 ≡ + 8 （mod 12）

– 5 ≡ + 7 （mod 12）

2.①由真值、原码转化为补码

- 求补规则：
  - 正数的补码符号位为0，数值部分就是真值。
  - 负数的补码符号位为1，数值部分可由真值的数值部分按位取反，末位加一得到。

例： 机器字长8位，$[-46]_{补码}$ = ？

$[46]_{补码}$ = 0 0 1 0 1 1 1 0

按位求反 → 1 1 0 1 0 0 0 1

末位加一 → 1 1 0 1 0 0 1 0 = D2H

当机器字长16位，$[-46]_{补码}$ = FFD2H

$[+1]_{补}$=00000001B　　$[-1]_{补}$=11111111B

0的补码

例：　$[+0]_{补码}$ = 0 0 0 0　0 0 0 0

$[-0]_{补码}$ = ?

0 0 0 0　0 0 0 0

取反　1 1 1 1　1 1 1 1

\+　0 0 0 0　0 0 0 1

1 0 0 0 0　0 0 0 0

进位

$[-0]_{补码}$ = 0 0 0 0　0 0 0 0 = $[+0]_{补码}$

∴ 补码中0 的表示惟一

n位二进制补码整数的表数范围： $-2^{n-1} \le N \le 2^{n-1}-1$

十进制	二进制	十六进制	十进制	十六进制
	n=8			n=16
+127	0111 1111	7F	+32767	7FFF
+126	0111 1110	7E	+32766	7FFE
…	…	…	…	…
+2	0000 0010	02	+2	0002
+1	0000 0001	01	+1	0001
0	0000 0000	00	0	0000
-1	1111 1111	FF	-1	FFFF
-2	1111 1110	FE	-2	FFFE
…	…	…	…	…
-126	1000 0010	82	-32766	8002
-127	1000 0001	81	-32767	8001
-128	1000 0000	80	-32768	8000

补码比原码多表示一个数！

② 由补码求真值

- **规则：**
  - 若补码的符号位为0，则真值为正，真值的数值部分等于补码的数值部分；
  - 若补码的符号位为1，则真值为负，真值的数值部分由补码的数值部分取反加一得到。

例：$X_{补}$ = 1.0110，求$X_{原}$与真值

$$
\begin{array}{ll}
X_{补} = & 1.0110 \\
\text{尾数变反} & 1.1001 \\
\text{末位加1} & +\quad\ \ 1 \\
\hline
X\text{原}= & 1.1010 \\
\text{真值} & -0.1010 = (-0.625)_{10}
\end{array}
$$

3. 由x的补码求-x的补码

求补运算：对一个补码表示的机器数（可以是正数或负数），连同符号位一起按位变反后，在最低位加1.

例：

$[117]_{补}$=01110101

对117求补：①取反得：10001010

②加一得：10001011

$[-117]_{补}$= 10001011

对-117求补：①取反得：01110100

②加一得：01110101

性质：$[X]_{补}$ —求补→ $[-X]_{补}$ —求补→ $[X]_{补}$

补码的运算：加法和减法

加法规则：$[X+Y]_{补码} = [X]_{补码} + [Y]_{补码}$

减法规则：$[X-Y]_{补码} = [X]_{补码} + [-Y]_{补码}$

例： X=64D, Y=46, 求X-Y

64	0100 0000
+ (-46)	+ 1101 0010
18	0001 0010

补码减法可转换为补码加法,因此加减法可以使用同一个电路实现

补码的优势

- 满足（−x）+（+x）=0

  (−6)+(+6)=0110+1010=0000

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

- 反码(one's complement code)表示法

正数的反码同原码，负数的反码数值位与原码相反

例：n = 8bit

$[+5]_{反码}$ = 0 0000101 = 05H

$[-5]_{反码}$ = 1 1111010 = FAH

$[+0]_{反码}$ = 0 0000000 = 00H

$[-0]_{反码}$ = 1 1111111 = FFH

∴ 0 的表示不惟一

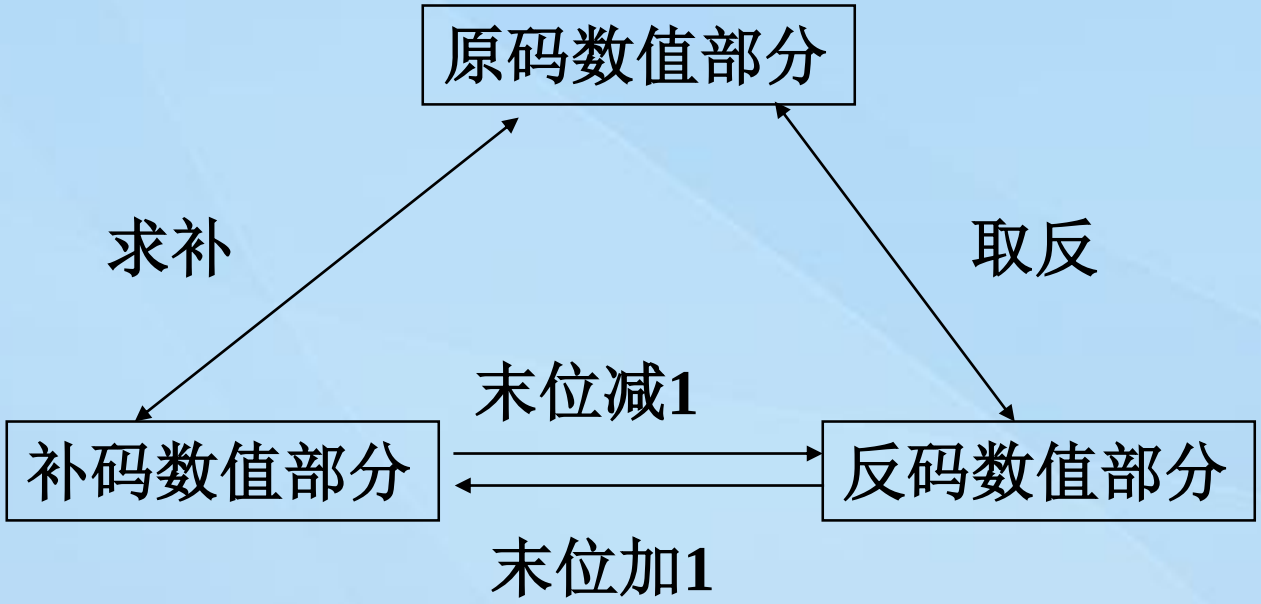
***反码不能直接进行两数的加减运算**

小结

- 原码
  - +0与-0不归一；
  - 8位原码表示数的范围为：-127 ～ +127；
  - 原码不能直接进行两数的加减运算
- 反码
  - +0与-0不归一；
  - 8位原码表示数的范围为：-127 ～ +127；
  - 反码不能直接进行两数的加减运算
- 补码
  - 补码+0与-0归一；
  - 数的范围为：-128 ～ +127；
  - 可以直接进行两数的加减运算

※原码和反码有±0，补码只有一个0

负数原/反/补码关系



“0”的原码、反码、补码

$[+0.0000]_{原}=0.0000$　　$[-0.0000]_{原}=1.0000$

$[+0.0000]_{补}=0.0000$　　$[-0.0000]_{补}=0.0000$

$[+0.0000]_{反}=0.0000$　　$[-0.0000]_{反}=1.1111$

∵ “0”的原码和反码都有两个，补码只有一个

∴ 补码可以比原码和反码多表示一个负数

- 设$[x]_{补}$=1.0000 其真值应为‘1’，但是1.0000取反再加1，得1.1111+1=10.0000，则得到真值为‘0’，哪儿错了？
- 记住：已知$[x]_{补}$，求$[-x]_{补}$时，需要将$[x]_{补}$连同符号位一起取反，然后再在末位加1，即可得到$[-x]_{补}$

1.2.6 移码(Excess 2^{n-1})

- $[x]_{移}=2^{n-1}+x$
- 同一数值的移码与补码仅符号位不同
- 表示范围：-2^{n-1}～ $+2^{n-1}-1$

例:写出-0101100的补码和移码

[x]补=11010100　[x]移=01010100

无论正负数的移码，都是将其补码的符号位取反得到，因此，移码也称为符号位取反的补码，一般用作浮点数的阶码

移码

- 用补码表示阶码的时候，当阶码无限小，产生了下溢的时候，阶码变成了0，那么这个浮点数的值变为了1。
- 而实际上这个数是无限接近于零的。那么我们就需要取出其中的〞-0“ 值作为机器零，于是移码产生了。
- 移码（又叫增码）是符号位取反的补码，一般用做浮点数的阶码，引入的目的是为了保证浮点数的机器零为全0。

- ①移码的定义：设由1位符号位和n位数值位组成的阶码，则 [X]移= $2^n + X$ $-2^n \leqslant X \leqslant 2^n$
- 例如： X=+1011 [X]移=11011
- X=－1011 [X]移=00101
- ②移码与补码的关系： [X]移与[X]补的关系是符号位互为相反数（仅符号位不同），
- 例如： X=+1011 [X]补=01011 [X]移=11011
- X=－1011 [X]补=10101 [X]移=00101
- ③移码运算应注意的问题：
  - 对移码运算的结果需要加以修正，即对结果的符号位取反后才是移码形式的正确结果。
  - 移码表示中，0有唯一的编码——1000...00，当出现000...00时，属于浮点数下溢。

真值	原码	反码	补码	移码
-128			10000000	00000000
-127	11111111	10000000	10000001	00000001
…	…	…	…	…
-1	10000001	11111110	11111111	01111111
-0	10000000	11111111	00000000	10000000
+0	00000000	00000000	00000000	10000000
+1	00000001	00000001	00000001	10000001
…	…	…	…	…
+127	01111111	01111111	01111111	11111111

计算机中的数为什么用补码表示?

1、只有补码才能实现$[x]_{补}+[-x]_{补}=0$

2、只有补码“0”才是唯一的

作业

- P27练习一
  - 10、13、14

1.1 进位记数制

- **进位记数制三要数：数码、基数、位权**

数码：每个数位上允许的数的集合

基数：进制中允许每个数位上选用基本数码的个数

位权：数码“1”在不同数位上代表的数值

常用进制：

1	二进制	$(1001)_2$	1001B
2	八进制	$(317)_8$	317Q
3	十六进制	$(9A1)_{16}$	9A1H
4	十进制	$(531)_{10}$	531D

不同进制表示16以内的数

十进制	二进制	八进制	十六进制
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

1.1.3 数制之间的转换

- 非十进制数转换为十进制数

  按权相加法：先将各位数码与权值相乘，再将各位的乘积值相加，得到十进制数

- 十进制数转换为任意进制数
  - 整数小数分别转换
    - 整数部分：*除基取余法*
    - 小数部分：*乘基取整法*
- 转换举例
  - 十进制转为其他十进制
    - 79.45D=(　？　)B =(　？　)H

79.45D≈(1001111.0111)B≈(4F.7)B

1.2 带符号的二进制数的表示

- 真值:用正、负符号加绝对值表示的二进制数值
  - 例如：+9的真值:+1001 ， -9的真值:-1001
- 机器数:在计算机内部使用的、符号数码化的定长二进制数
  - 原码
    - +0与-0不归一;
    - 8位原码表示数的范围为: -127 ～ +127
    - 原码不能直接进行两数的加减运算
  - 反码
    - +0与-0不归一;
    - 8位原码表示数的范围为: -127 ～ +127
    - 反码不能直接进行两数的加减运算
  - 补码
    - +0与-0归一;
    - 数的范围为: -128 ～ +127;
    - 可以直接进行两数的加减运算

	+9	-9
真值	+1001	-1001
原码	01001	11001
反码	01001	10110
补码	01001	10111
移码	11001	00111

课堂练习

- 按8位字长写出原码、反码、补码和移码

	+30D	-30D	+0.0101B	-0.0101B
真值	+11110	-11110	+0.0101	-0.0101
原码	00011110	10011110	00101000	10101000
反码	00011110	11100001	00101000	11010111
补码	00011110	11100010	00101000	11011000
移码	10011110	01100010	10101000	01011000

内容提要

山东大学 计算机科学与技术学院

1.定点数(Fix point number)

小数点位置固定不变的数

(1)无符号整数

略去符号位的正整数

$$X_{n-1}\cdots X_1X_0$$

小数点隐含位置

(2)带符号整数

$$X = X_nX_{n-1}\cdots\cdots X_1X_0$$

(共n+1位)

符号位

小数点隐含位置

可以使用原码、反码、补码表示

定点整数表示范围

n 位（含符号位）原码

$$-2^{(n-1)}-1 \le X \le 2^{(n-1)}-1$$

例：n = 8bit

$[+127]_{原码}$ = 0 1111111 B = 7FH

$[-127]_{原码}$ = 1 1111111 B = FFH

$[+0]_{原码}$ = 0 0000000 = 00H

$[-0]_{原码}$ = 1 0000000 = 80H

定点整数表示范围

n 位（含符号位）补码

$$-2^{(n-1)} \le X \le 2^{(n-1)}-1$$

例：n = 8bit

$[+127]_{补码}$ = 0 1111111 B = 7FH

$[-127]_{补码}$ = 1 0000001 B = 81H

$[-128]_{补码}$ = 1 0000000 B = 80H

$[+0]_{补码} = [-0]_{补码}$ = 0 0000000 B= 0H

(3)带符号定点小数(纯小数)

符号位

$$X = X_n X_{n-1} X_{n-2} \cdots\cdots X_1$$

小数点隐含位置

数值部分

1.3.2 浮点数

浮点数(Floating point number)：小数点位置是浮动的

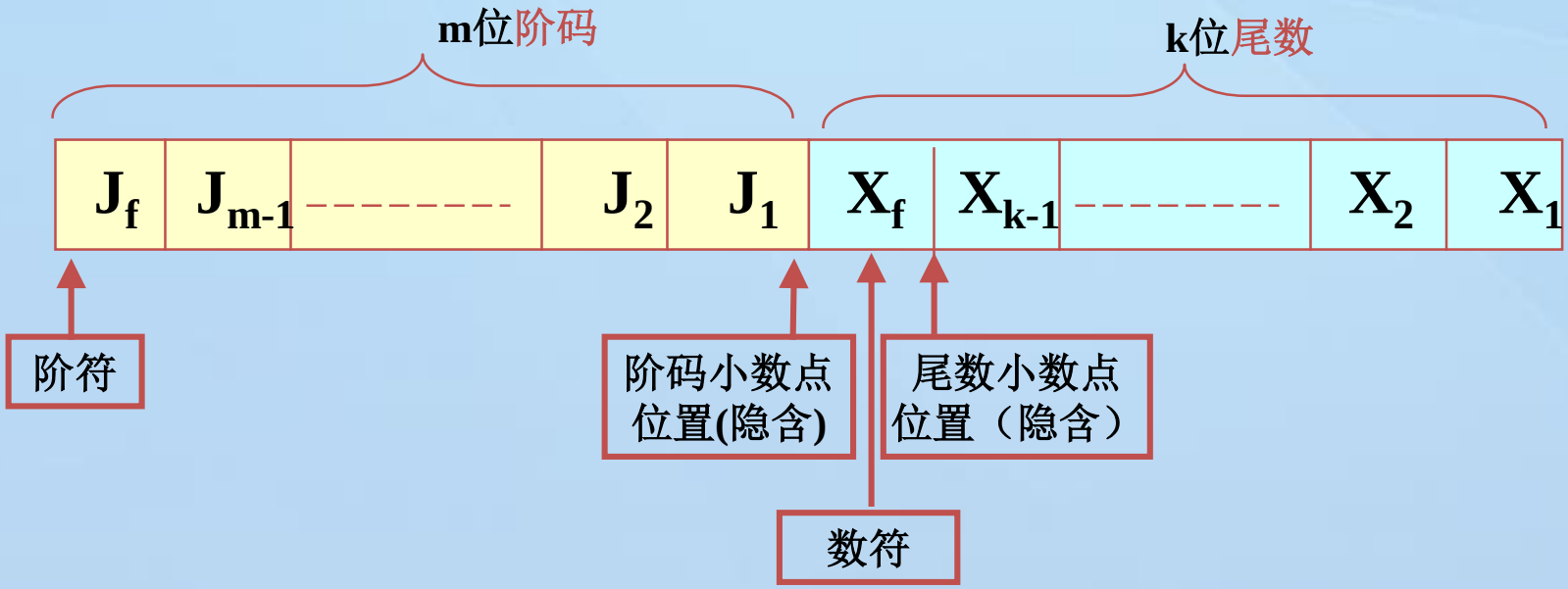
分两部分：

阶码J：用定点整数表示，它决定了浮点数的取值范围。

尾数X：常用定点小数表示，它决定了浮点数的表示精度。

底数R为2。

$$N = \pm R^{J} \times X$$



举例

⑩ 将二进制数x = -0.1010× 2^{-11}写成机器数形式。共占8位，J占3位，X占5位（各含1位符号位）。

⑩ 阶码和尾数都用原码表示

1	1	1	1	1	0	1	0

⑩ 阶码和尾数都用补码表示

1	0	1	1	0	1	1	0

实际应用中，阶码J常用移码表示，尾数X常用补码表示。

问题：表示不唯一。例如 $0.1010\times 2^{0}= 0.0101\times 2^{1}$

规格化的浮点数(尾数的规格化)：

① 尾数应为纯小数

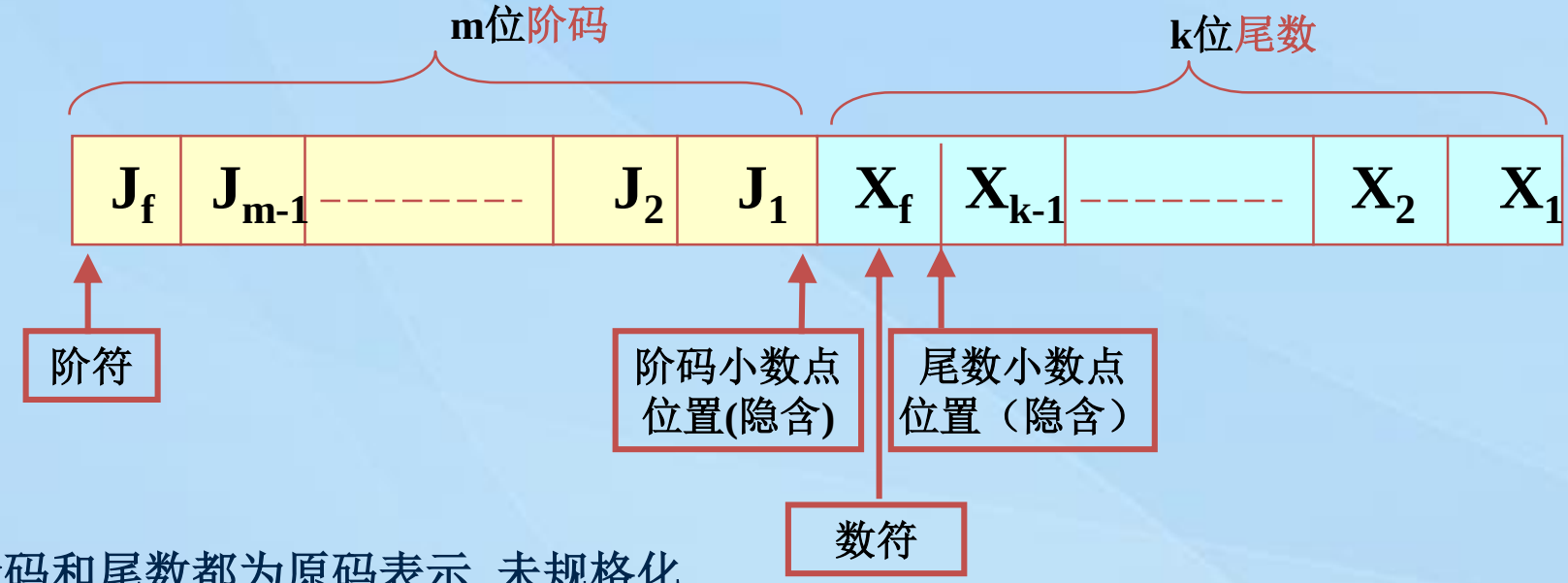
② 尾数的值不为0时，其绝对值应大于等于十进制的0.5而小于1，即$1/2\leqslant|X|<1$。

浮点表示法的特点：

① 优点：在有限位数(即不增加字长)内，既能保证有较大的取值范围，又能保证较高的精度。

② 缺点：实现浮点运算的硬件成本较高 。

4. 表示精度和范围



阶码和尾数都为原码表示, 未规格化

- 绝对值最大负数: 01…1 1. 11…1 $-(1-2^{-(k-1)})(2^{2^{m-1}-1})$
- 绝对值最小负数: 11…1 1. 00…1 $-2^{-(k-1)} \times 2^{-(2^{m-1}-1)}$
  - 最小正数: 11…1 0. 00…1 $2^{-(k-1)} \times 2^{-(2^{m-1}-1)}$
  - 最大正数: 01…1 0. 11…1 $(1-2^{-(k-1)}) \times 2^{(2^{m-1}-1)}$

例题：已知一个浮点数的格式为：阶码6位，其中阶符占1位，用补码表示；尾数10位，其中数符1位，用原码表示；基数为2，尾数采用规格化表示，问该浮点数所能表示的最大正数是多少？最小负数是多少？

阶码： 011111（正的最大值）

尾数的正的最大值：0.111111111

尾数的负的最小值：1.111111111

则：最大正数：$0.111111111 \times 2^{011111}$

最小负数：$1.111111111 \times 2^{011111}$

- 例:某浮点数字长32位,阶码8位,含1位阶符,补码表示;尾数24位,含1位数符,补码表示,规格化.若浮点数代码为$(A3680000)_{16}$,求其真值.

$$(A3680000)_{16}=(10100011\ \ 0110100000...00000)_2$$

$$E=-(1011101)_2=-(93)_{10}$$

$$M=(0.11010...0)_2=(0.8125)_{10}$$

$$N=2^{-93}\times 0.8125$$

作业

- P27
  - 浮点数表示16、17、18

内容提要

1.4 二进制编码

- 计算机只能识别二进制数
- 二进制编码
  - 数字:用二进制表示十进制BCD码
  - 字母：ASCII码
  - 符号
  - 声音
  - 图像

1.4.1 二进制编码的十进制

- **BCD码(Binary Code Decimal)**
  - **10个不同数字**
  - **逢十进位(十进制)**
  - **8421码**
    - 例如：**10D的$(0001\ 0000)_{BCD}$**

编码 / 数码	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
权	8421	2421	

例：1011.01B=11.25D=(0001 0001.0010 0101)BCD

1.4.3 字符编码

●美国信息交换标准代码（ASCII码）

ASCII码（American Standard Code for Information Interchange）

由7位二进制数组成，可表示128种字符。

128个元素 包括：

- 0-9十个数字
- 52个大小写英文字母
- 34个专用符号
- 32个控制符号

非打印类（控制代码）：33个，如回车（0DH）、换行（0AH）等

打印类：95个，包括英文字符、数字和其他可打印的符号等。

A～Z,a～z及0～9的编码按顺序递增数据编码，便于检索。

1.4.3 ASCII码

- 数字0-9的ASCII码：30H-39H

  30H+数值

- A-Z的ASCII码：41H-5AH

- a-z的ASCII码：61H-7AH

小写字母的ASCII码=对应大写字母的ASCII码+20H

- 换行的ASCII码：0AH
- 回车的ASCII码：0DH
- 空格的ASCII码：20H

ASCII码表

ASCII 字符代码表 一

高四位 / 低四位		ASCII非打印控制字符										ASCII 打印字符												
		0000					0001					0010		0011		0100		0101		0110		0111		
		0					1					2		3		4		5		6		7		
		+进制	字符	ctrl	代码	字符解释	+进制	字符	ctrl	代码	字符解释	+进制	字符	+进制	字符	+进制	字符	+进制	字符	+进制	字符	+进制	字符	ctrl
0000	0	0	BLANK NULL	^@	NUL	空	16	▶	^P	DLE	数据链路转意	32		48	0	64	@	80	P	96	`	112	p	
0001	1	1	☺	^A	SOH	头标开始	17	◀	^Q	DC1	设备控制 1	33	!	49	1	65	A	81	Q	97	a	113	q	
0010	2	2	☻	^B	STX	正文开始	18	↕	^R	DC2	设备控制 2	34	"	50	2	66	B	82	R	98	b	114	r	
0011	3	3	♥	^C	ETX	正文结束	19	!!	^S	DC3	设备控制 3	35	#	51	3	67	C	83	S	99	c	115	s	
0100	4	4	◆	^D	EOT	传输结束	20	¶	^T	DC4	设备控制 4	36	$	52	4	68	D	84	T	100	d	116	t	
0101	5	5	♣	^E	ENQ	查询	21	§	^U	NAK	反确认	37	%	53	5	69	E	85	U	101	e	117	u	
0110	6	6	♠	^F	ACK	确认	22	■	^V	SYN	同步空闲	38	&	54	6	70	F	86	V	102	f	118	v	
0111	7	7	•	^G	BEL	震铃	23	↨	^W	ETB	传输块结束	39	'	55	7	71	G	87	W	103	g	119	w	
1000	8	8	◘	^H	BS	退格	24	↑	^X	CAN	取消	40	(	56	8	72	H	88	X	104	h	120	x	
1001	9	9	○	^I	TAB	水平制表符	25	↓	^Y	EM	媒体结束	41	)	57	9	73	I	89	Y	105	i	121	y	
1010	A	10	◙	^J	LF	换行/新行	26	→	^Z	SUB	替换	42	*	58	:	74	J	90	Z	106	j	122	z	
1011	B	11	♂	^K	VT	竖直制表符	27	←	^[	ESC	转意	43	+	59	;	75	K	91	[	107	k	123	{	
1100	C	12	♀	^L	FF	换页/新页	28	∟	^\	FS	文件分隔符	44	,	60	<	76	L	92	\	108	l	124	|	
1101	D	13	♪	^M	CR	回车	29	↔	^]	GS	组分隔符	45	-	61	=	77	M	93	]	109	m	125	}	
1110	E	14	♫	^N	SO	移出	30	▲	^6	RS	记录分隔符	46	.	62	>	78	N	94	^	110	n	126	~	
1111	F	15	☼	^O	SI	移入	31	▼	^-	US	单元分隔符	47	/	63	?	79	O	95	_	111	o	127	⌂	^Back space

注：表中的ASCII字符可以用:ALT + "小键盘上的数字键" 输入

2. 汉字编码(了解)

⑩输入码

即汉字输入方法，又称机外码。

⑩国标码：以数字代码来区别每一汉字的。

是指我国1981年公布的国家标准《信息交换用汉字编码字符集—基本集》，即GB2312-80码，简称国标码。是我国计算机系统必须遵循的基础性标准之一。

⑩ 机内码

是计算机内部用来表示汉字的编码，机内码的设计与具体的系统及使用要求有密切关系。两个字节代表一个汉字。

机内码=国标码+8080H。

中:11010110 11010000 国:10111001 11111010(0B9FAH)

作业

- P27练习一
  - 10、13、14

码距

- 反映两个码字不一样的程度，即把两个码字对齐后，不同的位数即为码距，如110和111，其码距为1；100和111，其码距为2；000和111，其码距为3，等等

1.4.2 单位间距码

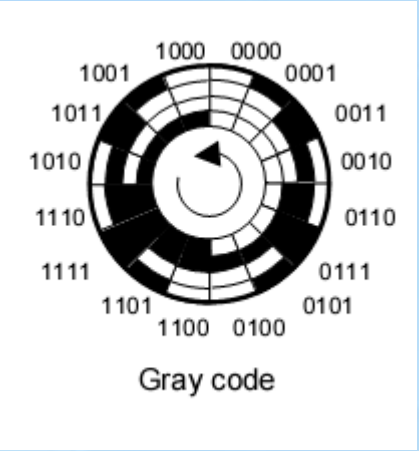
- 若表示相邻数字信息的代码只有一位不同，则该编码称为单位间距码(其间距为按位异或后1的个数)
  - 例如：二进制数的单位间距码称为Gray码

两位二进制数	两位Gray码
00	00
01	01
10	11
11	10

Gray Code是由贝尔实验室的Frank Gray在20世纪40年代提出的（是1880年由法国工程师Jean-Maurice-Emlle Baudot发明的），用来在使用PCM（Pusle Code Modulation）方法传送讯号时避免出错，并于1953年3月17日取得美国专利。Gray Code的编码方式不是唯一的，这里讨论的是最常用的一种。

1.4.2 单位间距码

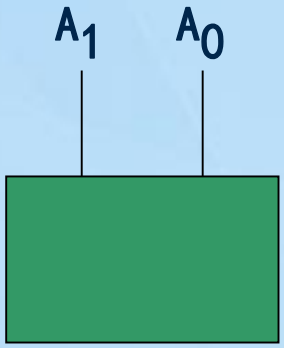
	四位二进制数	四位Gray码
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000



- 公式
  - $g_n=b_n$
  - $g_i=b_{i+1}\oplus b_i$ 异或

单位间距码的优势

- A_1A_0由01变为10



状态	A_1	A_0
开始	0	1
可能的中间状态1	1	1
可能的中间状态2	0	0
结束	1	0

- 可靠性编码：在状态转换时只有一位信号发生变化，不易出错。

格雷码的特点

- **是一种错误最小化的可靠性编码**（如从十进制的3（011/010）变为（100/110）4时二进制的变化巨大，可能使数字电路产生巨大的尖峰脉冲，但格雷码在相邻位间转换时只有一位产生变化，大大减少由一个状态到另一个状态时的逻辑混淆，因而在用于方向的转角位移量—数字量的转换中更为可靠）
- **是一种绝对编码方式，是一种具有反射性和循环性的单步自补码，可消除随机取数出现的重大误差，取反简单**
- **是变权码，每一码位没有固定的大小，难以直接比较大小和算术运算，需要经过一次变换转为自然二进制**
- **格雷码所对应十进制数的奇偶性与码字中1的个数的奇偶性相同。**

内容提要

1.5 校验码

- 传输或存储错误原因
  - 外界干扰
  - 电路故障
- 校验码目的
  - 检错：由信息本身判断信息是否出错
  - 纠错：发生错误时纠正错误
- 编码方法
  - 增加校验信息或校验位等冗余信息提高可靠性
- 校验码：将有效信息和校验信息按一定规律编制的代码
  - 奇偶校验
  - 水平垂直校验
  - 海明校验

1.5.1 奇偶校验

- 奇偶校验是对有效信息设置**一个校验位**来构成校验码，校验位的取值使得整个校验码中1的个数为奇数或偶数。
  - 奇校验
  - 偶校验

7 位数据	带有校验位的字节	
	偶校验	奇校验
0000000	00000000	00000001
1010001	10100011	10100010
1101001	11010010	11010011
1111111	11111111	11111110

1.5.1 奇偶校验

- 校验时如果检验码符合约定的规则，则认为正确
  - 在奇校验规则中0101 0011判为错误
  - 在偶校验规则中0101 0011判为正确
- 当有两位数字同时出错时，无法纠错
  - 偶校验规则中0101 0011->0100 1011
- 性能
  - 能发现奇数位错误，不能确定出错位置（纠错）
- 特点
  - 冗余较少，编码、校验方便
  - 在信息交换、处理中得到了广泛应用

1.5.2 水平垂直校验

- 水平垂直校验是一种具有纠错能力的多重奇偶校验
- 编码方法
  - 对每个有效信息代码配一个校验位，构成一个校验码
  - 将若干校验码作为一组，配一个校验字，构成校验组
- 示例：水平垂直校验（偶校验）

	7 位数据	校验位
校验码	0000000	0
	1010001	1
	1101001	0
	1111111	1
校验字	1000111	0

1.5.2 水平垂直校验

- 校验方法
  - 按奇偶校验规则，先检查水平方向是否有错
  - 再检查垂直方向是否有错
- 示例：水平垂直校验（偶校验）

00000000	00000000	00000000	00000000
10100011	10000011	10010011	10000011
11010010	11010010	11010010	11011010
11111111	11111111	11111111	11111111
10001110	10001110	10001110	10001110
正确	能确定错误位置	不能确定错误位置	不能确定错误位置

- 性能：能发现两位错误，纠正一位错误

- 奇偶校验是添加一个奇偶位用来指示之前的数据中包含有奇数还是偶数个1的检验方式。如果在传输的过程中，有奇数个位发生了改变，那么这个错误将被检测出来（注意奇偶位本身也可能改变）。原始数据和奇偶位组成的新数据中，将总共包含偶（奇）数个1.
- 奇偶校验并不总是有效，如果数据中有偶数个位发生变化，则奇偶位仍将是正确的，因此不能检测出错误。而且，即使奇偶校验检测出了错误，它也不能指出哪一位出现了错误，从而难以进行更正。导致已经接收到的数据必须整体丢弃并且重新传输。在一个噪音较大的介质中，成功传输数据可能需要很长时间甚至不可能完成。虽然奇偶校验的效果不佳，但是由于他只需要一位额外的空间开销，因此这是开销最小的检测方式。并且，如果知道了发生错误的位，奇偶校验还可以恢复数据。

如果一条信息中包含更多用于纠错的位，且通过妥善安排这些纠错位使得不同的出错位产生不同的错误结果，就可以找出出错位。

在一个7位的数据中，单个数据位出错有7种可能，因此3个错误控制位（2^3）就可以确定是否出错及哪一位出错。

1.5.3 海明校验(Hamming)

- 海明码也是一种具有纠错能力的多重奇偶校验，它将一个信息代码按一定规律分成若干组，每组配一个奇偶校验位。
  - 编码方法
    - 在位序是2的整数次幂的位置放置校验位，其他位置放置有效信息
    - 每个分组按偶校验规则确定校验位的取值

Richard Hamming
（1915-1998）

- 海明码也称为汉明码，是在电信领域的一种线性调试码，以发明者理查德·卫斯里·汉明的名字命名。汉明码在传输的消息流中插入验证码，以侦测并更正单一比特错误。由于汉明编码简单，它们被广泛应用于内存（RAM）。可以侦测两个或以下同时发生的比特错误，并能够更正单一比特的错误。

编码方法及示例

校验码		p1	p2	a1	p3	a2	a3	a4
位序		1	2	3	4	5	6	7
二进制位序码		001	010	011	100	101	110	111
分组覆盖	G3	X		X		X		X
	G2		X	X			X	X
	G1				X	X	X	X

- 海明(7, 4)码
  - 有效信息a1 a2 a3 a4
  - 校验信息p1 p2 p3
- 关键问题：如何分组

- 示例

有效信息1001

p1=a1⊕a2⊕a4=0

p2=a1⊕a3⊕a4=0

p3=a2⊕a3⊕a4=1

所以海明码为0011001

校验方法

- 分别检查各分组中1的个数是否为偶数
  - 当第i组中1的个数为偶数时，g_i=0;否则g_i=1
  - 错误字G(=$g_1g_2...g_r$)指明是否出错以及错误位置
    - 当G等于全0时，代码无错误
    - 当G不等于0时，代码有错，G指明了出错位置

示例1

有效信息1001的海明校验码0011001

若接收到的代码为0011001

p1	p2	a1	p3	a2	a3	a4	g1	g2	g3
0	0	1	1	0	0	1			
			X	X	X	X	0		
	X	X			X	X		0	
X		X		X		X			0

G=0，所以没有错误

校验方法

- 示例2

有效信息1001的海明校验码0011001

若接收到的代码为0001001

p1	p2	a1	p3	a2	a3	a4	g1	g2	g3
0	0	0	1	0	0	1			
			X	X	X	X	0		
	X	X			X	X		1	
X		X		X		X			1

$G=g_1g_2g_3=011$，所以第三位(即a1)出错

代码的正确值为0011001

- 性能：能发现并纠正一位错误，不能识别多位错误

- 若要海明码能够检测出多位的错误或能够纠正多位的错误，必须对海明编码进行改进。

若要检测出d位错误，码距至少满足d+1

若要能纠正d位错误，码距至少满足2d+1

汉明编码方案通用算法

- 一、给数字的数据位（从左向右）标上序号，1，2，3，4，5...
- 二、将这些数据位的位置序号转换为二进制，001，010，011，100，101，等。
- 三、数据位的位置序号中所有为二的幂次方的位（编号1，2，4，8，等，即数据位位置序号的二进制表示中只有一个1）是校验位
- 四、有其它位置的数据位（数据位位置序号的二进制表示中至少2个是1）是数据位
- 五、每一位的数据包含在特定的两个或两个以上的校验位中，这些校验位取决于这些数据位的位置数值的二进制表示

- 校验位1覆盖了所有数据位位置序号的二进制表示倒数第一位是1的数据：如1（校验位自身），11，101，111，1001，等
- 校验位2覆盖了所有数据位位置序号的二进制表示倒数第二位是1的数据：如10（校验位自身），11，110，111，1010，1011，等
- 校验位4覆盖了所有数据位位置序号的二进制表示倒数第三位是1的数据：如100（校验位自身），101，110，111，1100，1101，1110，1111，等
- 依此类推

请诸位为乌有国王出个主意：500桶酒，其中1桶是毒酒；48小时后要举行酒会；毒酒喝下去会在之后的第23-24小时内毒死人；国王决定用囚犯来试酒，不介意囚犯死多少，只要求用最少的囚犯来测试出哪一桶是毒酒，问最少需要多少囚犯才能保证找出毒酒？

作业

- 练习一
  - 22, 23, 24, 25