

1.2 数字技术基础

1.2.1 比特

1.2.2 比特与二进制数

1.2.3 信息在计算机中的表示

1.2.4 比特的运算

1.2.5 小结

1.2.1 信息的基本单位 ——比特 (bit)

- (1) 什么是比特
- (2) 比特的存储
- (3) 比特的传输

什么是比特？

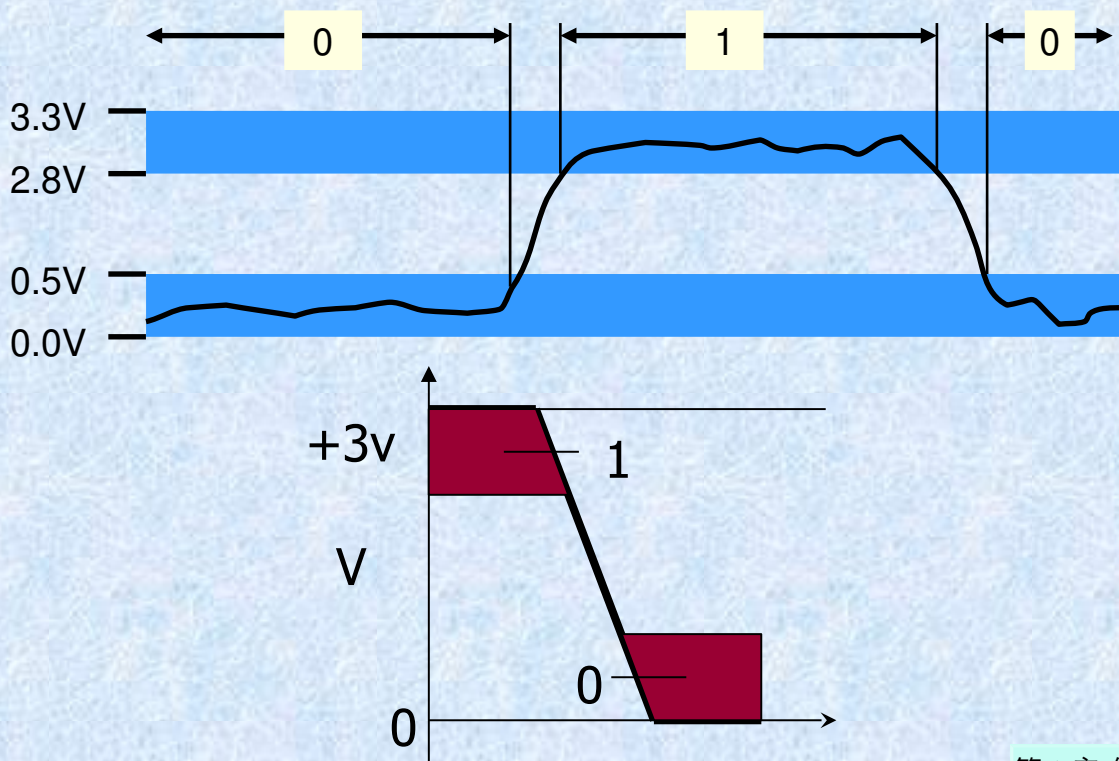
- 比特（ bit ， **binary digit** 的缩写 ）中文翻译为“二进制数字”、“二进制”或简称为“位”
- 比特只有 2 种取值：0 和 1 ，一般无大小之分
- 如同 DNA 是人体组织的最小单位、原子是物质的最小组成单位一样，**比特是组成数字信息的最小单位**
- 数值、文字、符号、图像、声音、命令 …… 都可以使用比特来表示

比特在计算机中如何表示？

- 在计算机中表示与存储二进位的方法：
 - 电路的高电平状态或低电平状态 (CPU)
 - 电容的充电状态或放电状态 (RAM)
 - 两种不同的磁化状态 (磁盘)
 - 光盘面上的凹凸状态 (光盘)
 - ...

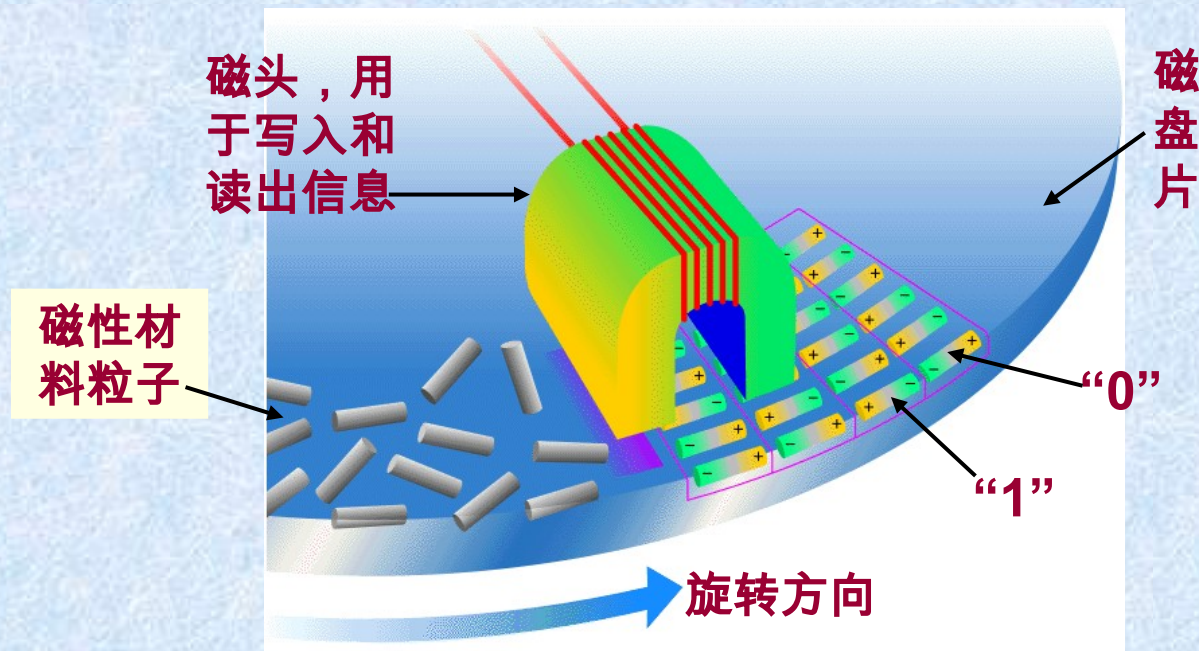
例 1 : CPU 内部比特的表示

- CPU 内部通常使用高电平表示 1，低电平表示 0



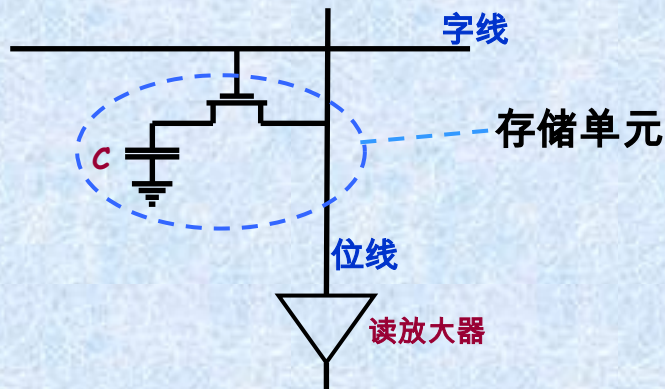
例 2：磁盘中比特的表示与存储

- 磁盘表面微小区域中，磁性材料粒子的两种不同的磁化状态分别表示 0 和 1



例 3：内存存储器中比特的存储

- 计算机存储器中用电容器存储二进位信息：当电容的两极被加上电压，它就被充电，电压去掉后，充电状态仍可保持一段时间，因而 1 个电容可用来存储 1 个比特



信息存储原理

- 电容 C 处于充电状态时，表示 1
- 电容 C 处于放电状态时，表示 0

- 集成电路技术可以在半导体芯片上制作出以亿计的微型电容器，从而构成了可存储大量二进位信息的半导体存储器芯片

断电后信息不再保持！

存储容量的计量单位

- 8 个比特 = 1 个字节 (byte , 用大写 B 表示)
- 计算机内存储器容量的计量单位 :
 - KB: $1\text{ KB} = 2^{10}\text{ 字节} = 1024\text{ B}$ (千字节)
 - MB: $1\text{ MB} = 2^{20}\text{ 字节} = 1024\text{ KB}$ (兆字节)
 - GB: $1\text{ GB} = 2^{30}\text{ 字节} = 1024\text{ MB}$ (吉字节、千兆字节)
 - TB: $1\text{ TB} = 2^{40}\text{ 字节} = 1024\text{ GB}$ (太字节、兆兆字节)
- 外存储器容量经常使用 10 的幂次来计算 :
 - $1\text{ MB} = 10^3\text{ KB} = 1\,000\text{ KB}$
 - $1\text{ GB} = 10^6\text{ KB} = 1\,000\,000\text{ KB}$
 - $1\text{ TB} = 10^9\text{ KB} = 1\,000\,000\,000\text{ KB}$

现象



160GB 的移动硬盘



为什么 160GB 的硬盘显示出来的容量只有 149.05GB ? 8GB 的 U 盘显示出来的容量只有 7.46GB ?

厂商标注的容量
使用十进制前缀

原因：

操作系统显示的容
量使用二进制前缀

相同的符号，有两种不同的含义！

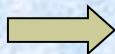
前缀名称	前缀符号	十进制前缀	二进制前缀	比值
kilo	k/K	10^3	$2^{10} = 1,024$	0.976
mega	M	10^6	$2^{20} = 1,048,576$	0.954
giga	G	10^9	$2^{30} = 1,073,741,824$	0.931
tera	T	10^{12}	$2^{40} = 1,099,511,627,776$	0.909
peta	P	10^{15}	$2^{50} = 1,125,899,906,842,624$	0.888
exa	E	10^{18}	$2^{60} = 1,152,921,504,606,846,976$	0.867
zetta	Z	10^{21}	$2^{70} = 1,180,591,620,717,411,303,424$	0.847
yotta	Y	10^{24}	$2^{80} = 1,208,925,819,614,629,174,706,176$	0.827

不同进位制前缀的使用场合

- 内存、cache、半导体存储器芯片的容量均使用二进制前缀：

- 512MB 的内存条 (其中 $1\text{M} = 2^{20}$)
- 256KB 的 cache (其中 $1\text{K} = 2^{10}$)

- 文件和文件夹的大小使用二进制前缀

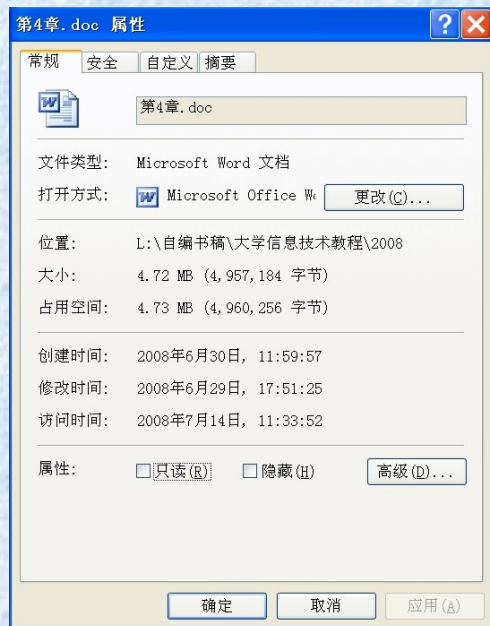


- 频率、传输速率等使用十进制前缀：

- 主频 1GHz ($1\text{G} = 10^9$)
- 传输速率 100Mbps (其中 $1\text{M} = 10^6$)

- 外存储器 (硬盘、DVD 光盘、U 盘、存储卡等) 容量：

- 厂商标注的容量使用十进制前缀
- 操作系统显示的容量使用二进制前缀



解决方案：使用两种不同的前缀符号

前缀名称	前缀符号	十进制值	二进制值	比值	IEC 建议 二进制前缀符号	
kilo	k/K	10^3	$2^{10} = 1,024$	0.976	kibi-	Ki
mega	M	10^6	$2^{20} = 1,048,576$	0.954	mebi-	Mi
giga	G	10^9	$2^{30} = 1,073,741,824$	0.931	gibi-	Gi
tera	T	10^{12}	$2^{40} = 1,099,511,627,776$	0.909	tebi-	Ti
peta	P	10^{15}	$2^{50} = 1,125,899,906,842,624$	0.888	pebi-	Pi
exa	E	10^{18}	$2^{60} =$	0.867	exbi-	Ei
zetta	Z	10^{21}	$2^{70} =$	0.847	zebi-	Zi
yotta	Y	10^{24}	$2^{80} =$	0.827	yobi-	Yi

- 已经采用 IEC 建议符号的有：
 - Mozilla Firefox , BitTornado , Linux , 以及其他一些 GNU 自由软件
- 尚未采用 IEC 建议符号的有：微软公司等

比特的传输

- 信息是可以传输的，信息只有通过传输和交流才能发挥它的作用
- 在数字通信技术中，信息的传输是通过比特的传输来实现的
- 近距离传输时：直接将用于表示“0/1”的电信号或光信号（称为**基带信号**）进行传输（称为**基带传输**），例如：
 - 计算机读/写移动硬盘中的文件；
 - 打印机打印文档的内容； 计算机通过投影仪进行显示
- 远距离传输或者无线传输时：需要使用**调制技术**（参见第4章第1节）

比特的传输速率

- **传输速率**表示每秒钟可传输的二进位数目，常用单位是：
 - 比特 / 秒 (b/s)，也称“bps”。如 2400 bps(2400b/s)
 - 千比特 / 秒 (kb/s)， $1\text{kb/s}=10^3$ 比特 / 秒 =1 000 b/s
 - 兆比特 / 秒 (Mb/s)， $1\text{Mb/s}=10^6$ 比特 / 秒 =1 000 kb/s
 - 吉比特 / 秒 (Gb/s)， $1\text{Gb/s}=10^9$ 比特 / 秒 =1 000 Mb/s
 - 太比特 / 秒 (Tb/s)， $1\text{Tb/s}=10^{12}$ 比特 / 秒 =1 000 Gb/s

1.2.2 比特与二进制数

- (1) 不同进位制数的表示和含义
- (2) 不同进位制数的相互转换
- (3) 二进制数的算术运算

不同进位制数的表示和含义

“数”是一种信息，它有大小（数值），可以进行四则运算

“数”有不同的表示方法。日常生活中人们使用的是十进制数，但计算机使用的是二进制数，程序员还使用八进制和十六进制数。

二进制数，八进制和十六进制数怎样表示？其数值如何计算？

十进制数

- 每一位可使用十个不同数字表示 (0、1、2、3、4、5、6、7、8、9)
- 低位与高位的关系是：逢 10 进 1
- 各位的权值是 10 的整数次幂 (基数是 10)
- 标志：尾部加“D”或缺省

例：

$$264.96 = 200 + 60 + 4 + 0.9 + 0.06 = 264.96$$

Weights indicated by brackets:

- 2×10^2
- 6×10^1
- 4×10^0
- 9×10^{-1}
- 6×10^{-2}

二进制数

- 每一位使用两个不同数字表示 (0 、 1) ，即每一位使用 1 个“比特”表示
- 低位与高位的 relationship 是：逢 2 进 1
- 各位的权值是 2 的整数次幂 (基数是 2)
- 标志：尾部加 B

例：

$$101.01 \text{ B} = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} = 4 + 0 + 1 + 0 + 1/4 = 5.25$$

八进制数

- 每一位使用八个不同数字表示 (0 、 1 、 2 、 3 、 4 、 5 、 6 、 7)
- 低位与高位的 关系是：逢 8 进 1
- 各位的权值是 8 的整数次幂 (基数是 8)
- 标志：尾部加 Q

例：

$$\begin{array}{rcll} & \overbrace{\hspace{1.5cm}} & 2 \times 8^{-1} & \\ 365.2Q & = & 192 & + \quad 48 & + \quad 5 & + \quad 2/8 & = & 245.25 \\ & \underbrace{\hspace{1.5cm}} & 5 \times 8^0 & \\ & \underbrace{\hspace{1.5cm}} & 6 \times 8^1 & \\ & \underbrace{\hspace{1.5cm}} & 3 \times 8^2 & \end{array}$$

十六进制数

- 每一位使用十六个数字和符号表示 (0、1、2、3、4、5、6、7、8、9、A、B、C、D、E、F)
- 逢 16 进 1, 基数为 16
- 各位的权值是 16 的整数次幂 (基数是 16)
- 标志 : 尾部加 H

例 :

$$\begin{array}{l} \text{F5.4H} = \begin{array}{l} \overbrace{15}^{4 \times 16^{-1}} \overbrace{5}^{5 \times 16^0} \overbrace{4}^{4 \times 16^{-1}} \\ 240 + 5 + 4/16 = 245.25 \end{array} \end{array}$$

不同进位制数的比较

	十进制	二进制	八进制	十六进制
零	0	0000	0	0
壹	1	0001	1	1
贰	2	0010	2	2
叁	3	0011	3	3
肆	4	0100	4	4
伍	5	0101	5	5
陆	6	0110	6	6
柒	7	0111	7	7
捌	8	1000	10	8
玖	9	1001	11	9
拾	10	1010	12	A
拾壹	11	1011	13	B
拾贰	12	1100	14	C
拾叁	13	1101	15	D
拾肆	14	1110	16	E
拾伍	15	1111	17	F

不同进制数的相互转换

熟练掌握不同进制数相互之间的转换，在编写程序和设计数字逻辑电路时很有用

只要学会二进制数与十进制数之间的转换，与八进制、十六进制数的转换就不在话下了

十进制数 → 二进制数

■ 转换方法：

整数和小数分开转换

整数部分：除以 2 逆序取余

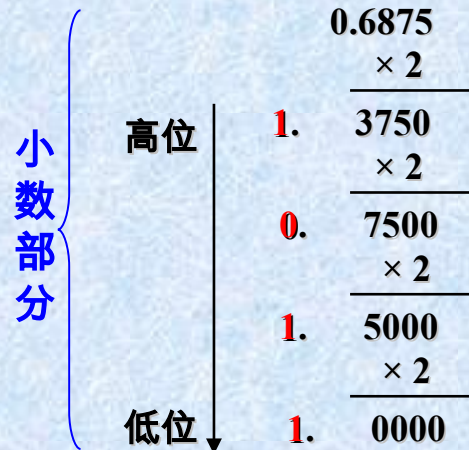
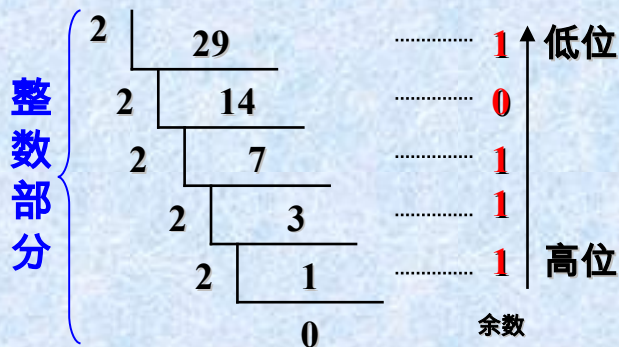
小数部分：乘以 2 顺序取整

■ 例如：29.6875 →

11101.1011 B

注意：十进制小数（如 0.63）在转换时会出现二进制无穷小数，这时只能取近似值

0.63 → 0.1010 1010 ...



二进制数 → 十进制数

■ 转换方法：

二进制数的每一位乘以其相应的权值，然后累加即可得到它的十进制数值

例：**11101.1011B**

$$\begin{aligned} &= 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \\ &\quad + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} + 1 \times 2^{-4} \\ &= 29.6875 \end{aligned}$$

记住 2^n 的值很有用！

■ $2^1=2$

■ $2^2=4$

■ $2^3=8$

■ $2^4=16$

■ $2^5=32$

■ $2^6=64$

■ $2^7=128$

■ $2^8=256$

■ $2^9=512$

■ $2^{10}=1024=1K$

■ $2^{11}=2048$

■ $2^{12}=4096$

■ $2^{13}=8192$

■ $2^{14}=16384$

■ $2^{15}=32768$

■ $2^{16}=65536$

...

■ $2^{20}=1M$

■ $2^{30}=1G$

■ $2^{40}=1T$

■ $2^{50}=1P$

■ $2^{60}=1E$

■ $2^{70}=1Z$

记住常用二进制小数的值！

二进制	十进制值
0.1	0.5
0.01	0.25
0.11	0.75
0.001	0.125
0.011	0.375
0.101	0.625
0.111	0.875

举例：二进制整数的表示范围

十进制数	可表示数的数目	可表示的最大数	二进制数	可表示数的数目	可表示的最大数
2 位	$10^2 = 100$	99	4 位	$2^4 = 16$	1111 =15
3 位	$10^3 = 1000$	999	8 位	$2^8 = 256$	11111111 =255
4 位	$10^4 = 10000$	9999	16 位	$2^{16} = 65536$	11...111 =65535
5 位	$10^5 = 100000$	99999	32 位	$2^{32} = 4294967296$	11...111 = $2^{32} - 1$
6 位	$10^6 = 1000000$	999999	64 位	$2^{64} = 18446744073709551616$	11.....111 = $2^{64} - 1$

八进制数与二进制数的互换

1 位八进制
数与 3 位二
进制数的对
应关系：

八进制数	二进制数	八进制数	二进制数
0	000	4	100
1	001	5	101
2	010	6	110
3	011		

- 八进制→二进制：把每个八进制数字改写成等值的 3 位二进制数，且保持高低位的次序不变

例：2467.32Q → 010 100 110 111 . 011 010 B

- 二进制→八进制：整数部分从低位向高位每 3 位用一个等值的八进制数来替换，不足 3 位时在高位补 0 凑满 3 位；小数部分从高位向低位每 3 位用一个等值八进制数来替换，不足 3 位时在低位补 0 凑满三位

例：1 101 001 110.110 01 B → 001 101 001 110.110 010 B
→ 1516.62 Q

十六进制数与二进制数的互换

■ 1 位十六进制数与 4 位二进制数的对应关系：

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

■ 转换方法：与八、二进制互换的方法类似

例 1 35A2.CFH → 11 0101 1010 0010.1100 1111B

例 2 1110 0100 1110.1100 11B → 34E.CCH

二进制数的算术运算

■ 1 位二进制数的加、减法运算规则：

被加数	加数	进位	和
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

(a) 加法规
则

被减数	减数	借位	差
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

(b) 减法规
则

■ 2 个多位二进制数的加、减法运算举例：

$$\begin{array}{r} 0101 \\ + 0100 \\ \hline 1001 \end{array}$$

$$\begin{array}{r} 1001 \\ - 0100 \\ \hline 0101 \end{array}$$

由低位到高位逐
位进行，低位向
高位进（借）位
！

1.2.3 信息在计算机中的表示

(1) 数值的表示

- 无符号整数的表示
- 带符号整数的表示
- 浮点数 (实数) 的表示

(2) (西文) 字符的表示

(3) (黑白) 图像的表示

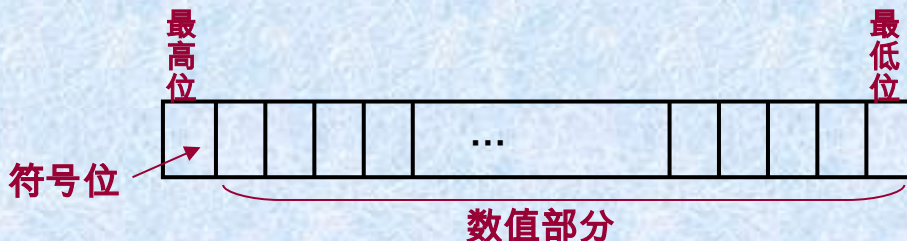
无符号整数的表示

- 采用“自然码”表示：
- 取值范围由位数决定：
 - 8 位：
可表示 $0 \sim 255 (2^8-1)$ 范围内的所有正整数
 - 16 位：
可表示 $0 \sim 65535 (2^{16}-1)$ 范围内的所有正整数
 - n 位：
可表示 $0 \sim 2^n-1$ 范围内的所有正整数。

十进制数	8 位无符号整
0	00000000
1	00000001
2	00000010
3	00000011
4	00000100
5	00000101
252	11111100
253	11111101
254	11111110
255	11111111

带符号整数的表示 (1)

- 表示方法：用一位表示符号，其余用来表示数值部分



- 符号用最高位表示：“0”表示正号 (+), “1”表示负号 (-)
- 数值部分有两种表示方法：

(1) 原码表示：

整数的绝对值以二进制自然码表示

(2) 补码表示：

正整数：绝对值以二进制自然码表示

负整数：绝对值使用补码表示

原码表示举例：

**[+43] 的 8 位原码
为： 00101011**

**[- 43] 的 8 位原码
为： 10101011**

选讲：

带符号整数的表示 (2)

■ 负数 (的绝对值) 如何用补码表示？

1. 先把绝对值表示为自然码
2. 将自然码的每一位取反码
3. 在最低位加“1”

■ 例 1: [- 43] 用 8 位补码表示

所以：

[- 43] 的 8 位补码为：11010101

(1) 43 => 0101011

(2) 取反：1010100

(3) 加 1：1010101

■ 例 2：[- 64] 用 8 位补码表示

所以：

[- 64] 的 8 位补码为：11000000

(1) 64 => 1000000

(2) 取反：0111111

(3) 加 1：1000000

选讲：

带符号整数的表示 (3)

■ 优缺点分析：

■ 原码表示法

- 优点：与日常使用的十进制表示方法一致，简单直观
- 缺点：加法与减法运算规则不统一，增加了成本；整数 0 有“00000000”和“10000000”两种表示形式，不方便

■ 补码表示法

- 优点：加法与减法运算规则统一，没有“-0”，可表示的数比原码多一个
- 缺点：不直观，人使用不方便

■ 结论：带符号整数在计算机内不采用“原码”而采用“补码”的形式表示！

选讲：

带符号整数的表示 (4)

■ 原码可表示的整数范围

8 位原码： $-2^7+1 \sim 2^7-1$ ($-127 \sim 127$)

16 位原码： $-2^{15}+1 \sim 2^{15}-1$ ($-32767 \sim 32767$)

n 位原码： $-2^{n-1}+1 \sim 2^{n-1}-1$

■ 补码可表示的整数范围

8 位补码： $-2^7 \sim 2^7-1$ ($-128 \sim 127$)

-128 表示为 10000000

+127 表示为 01111111

n 位补码： $-2^{n-1} \sim 2^{n-1}-1$ -2^{n-1} 表示为 10000...000

$2^{n-1}-1$ 表示为 01111...111

PC 中 3 种不同长度的带符号整数

整数类型	格式	可表示的数值范围
16 位整数	16 个二进位， 补码表示	$- 2^{15} \sim 2^{15} - 1$ ($- 32768 \sim 32767$)
短整数	32 个二进位， 补码表示	$- 2^{31} \sim 2^{31} - 1$ ($- 2147483648 \sim 2147483647$)
长整数	64 个二进位， 补码表示	$- 2^{63} \sim 2^{63} - 1$ ($- 9223372036854775808$ ~ 9223372036854775807)

小结：3 种整数的比较

- 计算机中整数有多种，同一个二进制代码表示不同类型的整数时，其含义（数值）可能不同
- 一个代码它到底代表哪种整数（或其它东西），是由指令决定的

8 位二进制码	表示无符号整数时的数值	表示带符号整数（原码）时的值	表示带符号整数（补码）时的值
0000 0000	0	0	0
0000 0001	1	1	1
.....
0111 1111	127	127	127
1000 0000	128	- 0	- 128
1000 0001	129	- 1	- 127
.....
1111 1111	255	- 127	- 1

实数的特点与表示方法

■ 特点：

- 既有整数部分又有小数部分，小数点位置不固定
- 整数和纯小数是实数的特例
- 任何一个实数总可以表达成一个乘幂和一个纯小数之积

- 例如： $56.725 = 0.56725 \times 10^2$
 $-0.0034756 = -0.34756 \times 10^{-2}$

■ 实数的表示方法（浮点表示法）：用3个部分表示

1. 乘幂中的**指数**：表示实数中小数点的位置
2. 纯小数部分（**尾数**）：表示实数中的有效数字部分
3. 数的正负（**符号**）

选讲：

二进制实数的浮点表示

■ 与十进制实数一样，二进制实数也可用浮点表示

■ 例如： $+1001.011B = +0.1001011B \times 2^{100}$

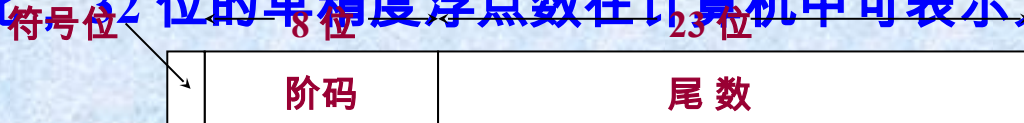
$-0.0010101B = -0.10101B \times 2^{-10}$

■ 可见，任一个二进制实数 N 均可表示为：

$$N = \pm S \times 2^P$$

(其中， \pm 是该数的**符号**； S 是 N 的**尾数**； P 是 N 的**阶码**)

■ 因此，32 位的单精度浮点数在计算机中可表示为：



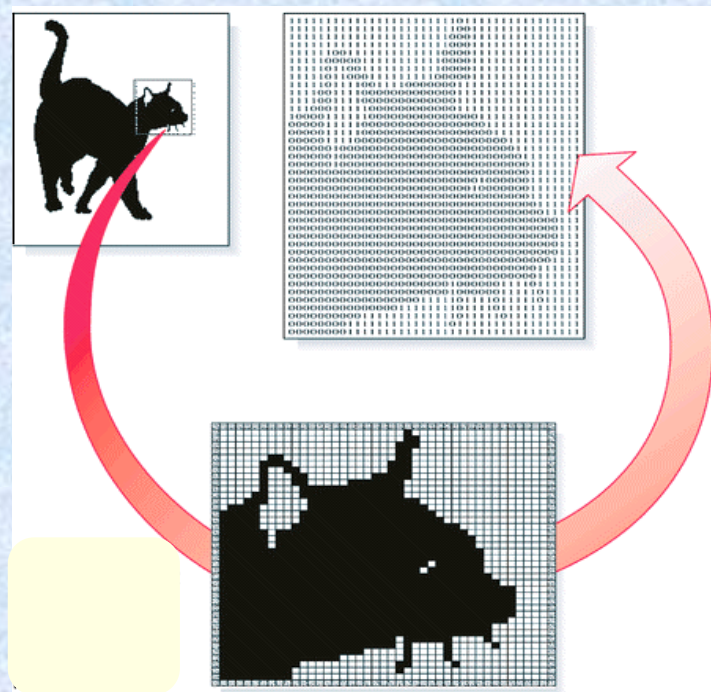
文字符号在计算机中的表示

- 日常使用的书面文字由一系列称为“字符” (character) 的书写符号所构成
- 计算机中常用字符的集合叫做“字符集”
 - 西文字符集
 - 中文 (汉字) 字符集 (参见第 5 章)
- 最常用的西文字符集是 ASCII (American Standard Code for Information Interchange) 字符集
 - 包含 96 个可打印字符和 32 个控制字符
 - 每个字符采用 7 个二进位进行编码
 - 计算机中使用 1 个字节存储 1 个 ASCII 字符

标准 ASCII 字符集及其码表

图像在计算机中如何表示？

- 把图像离散成为 M 列、 N 行，这个过程称为图像的取样
- 经过取样之后，图像就分解成为 $M \times N$ 个取样点，每个取样点称为图像的一个“像素”
- 如果是黑白图像，每个像素只有 2 个值：黑 (0) / 白 (1)，所以每个像素用一个二进位表示
- 因此，一幅黑白图像可使用一个矩阵表示
- 灰度图像和彩色图像表示比较复杂些 (参见第 5 章)



举例：黑白图像的表示

- 每个像素使用 1 个比特表示：0= 黑；1= 白



```
01010101010101010101010110101101001001000111110000
011010101010101010101001011010010110010100000110
100101010101010101010110110001010000101001010100
101101101011011010110101100110010110100010001001
011010010110100101101010001001100100101101010010
100101101100101010101110110011001001010101100
011010010011010110010010001001100110101010010001
010101101100101100100101110110011001010100100101
010101010101010011011010001001100010100001010100
101010101010101100010010110010001101001110100001
010101010101010001000101000101101000010000001101
1101101010100100100110100011010010011100101101000
101001010100100010100101100101101100001010000010
101011010001001001001001011110101011010100101100
101010000100010010010111110101111100101001001001
010100101001000100101010111010101010010010000
10100100001001100110111101011101010101000100101
010010010100100011011000011110111011010110101000
000100000001001100100111111111110110111000000010
101000101010010011011000010101011101000010101000
000010000100101101010011111111111111011101000101
0010001010011010101010010001110111110100010010000
010010010110001001001001111011110101101100100101
100100100000111010010010010111111111011001001000
```


关于信息表示的小结

- 计算机（包括其它数字设备）中所有信息都使用比特（二进位）表示
 - 例如数值、文字符号、图像、声音、动画、温度、压力、运动等，包括指挥计算机工作的软件（程序），也是用二进位表示的
- 只有使用比特表示的信息计算机才能进行处理、存储和传输！

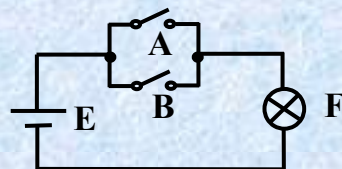
1.2.4 比特的基本运算

比特的 3 种基本逻辑运算

逻辑加： $F = A \vee B$

(“或”运算)

A	B	$A \vee B$
0	0	0
0	1	1
1	0	1
1	1	1

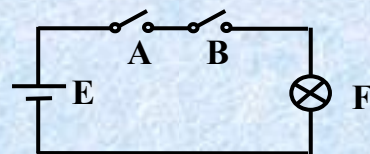


开关闭合 = 1，断开 = 0

逻辑乘： $F = A \wedge B$

(“与”运算)

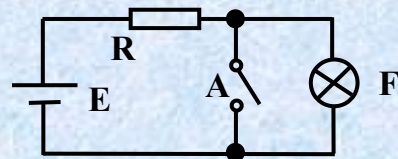
A	B	$A \wedge B$
0	0	0
0	1	0
1	0	0
1	1	1



取反： $F = \text{NOT } A$

(“否定”运算)

A	NOT A
0	1
1	0



逻辑运算可以用开关电路实现！

两个多位数的逻辑运算

- 按位独立进行逻辑运算，每一位都不受其它位的影响

例 1

$$\begin{array}{rcl} \text{A:} & & 0110 \\ \text{B:} & \vee & \underline{1010} \\ \text{F:} & & 1110 \end{array}$$

例 2

$$\begin{array}{rcl} \text{A:} & & 0110 \\ \text{B:} & \wedge & \underline{1010} \\ \text{F:} & & 0010 \end{array}$$

例 3

$$\begin{array}{rcl} \text{A:} & & \underline{0110} \\ \text{NOT A:} & & 1001 \end{array}$$

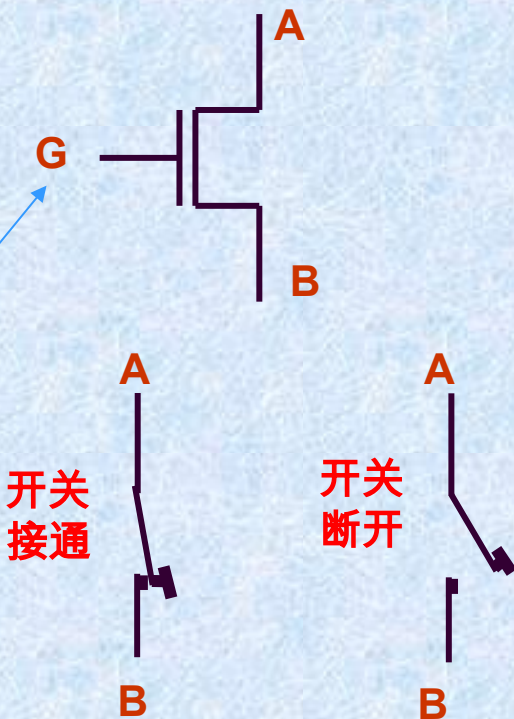
例 4

$$\begin{array}{rcl} \text{B:} & & \underline{1010} \\ \text{NOT B:} & & 0101 \end{array}$$

晶体管是一种电子开关

- 使用**机械**开关实现逻辑操作**速度太慢**，**工作也不可靠**！
- 晶体管好像是一个电子开关，它工作在两种状态：导通状态 / 绝缘状态，效果相当于 A 和 B 之间的接通或断开

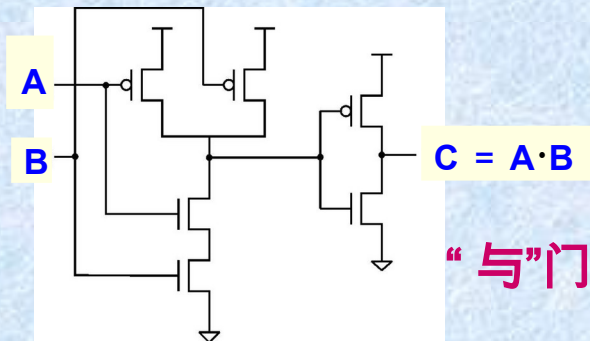
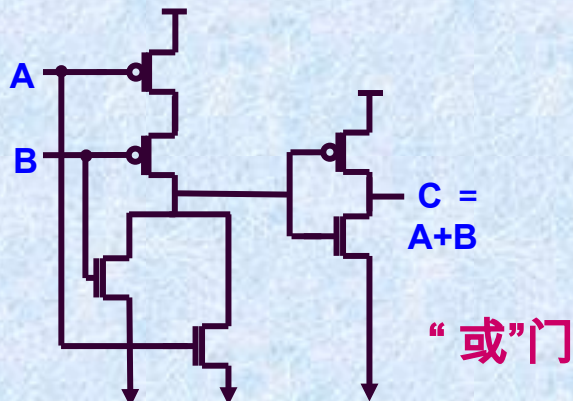
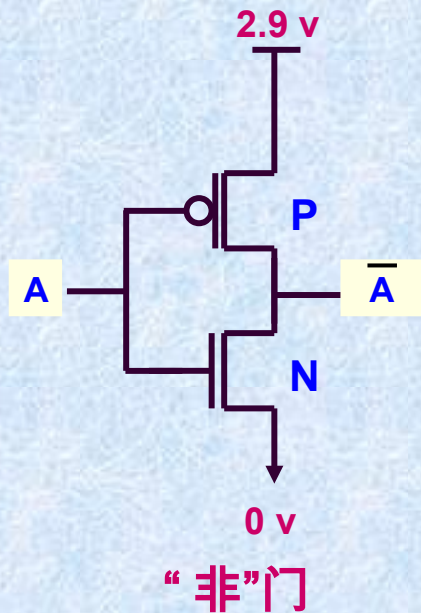
晶体管的两种状态
(通 / 断) 由**控制端 G** 的电压决定



选讲：



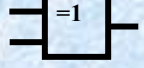
几个晶体管组合可完成逻辑运算

门电路：



选讲：

逻辑运算是用“门”电路实现的

名称	运算符号	定义			门电路符号 (国标)	门电路符号 (国外)
		A	B	F		
与	AB , $A \cdot B$ $A \wedge B$	0 0 1 1	0 1 0 1	0 0 0 1		
或	$A+B$, $A \vee B$	0 0 1 1	0 1 0 1	0 1 1 1		
非	\overline{A}	0 1		1 0		
与非	$\overline{A \cdot B}$	0 0 1 1	0 1 0 1	1 1 1 0		
或非	$\overline{A+B}$	0 0 1 1	0 1 0 1	1 0 0 0		
异或	$A \cdot \overline{B} + \overline{A} \cdot B$	0 0 1 1	0 1 0 1	0 1 1 0		

选讲：

两个 1 位二进制数加法的实现

- 设被加数 A ，加数 B ，用半加器完成加法，产生和数 S ，进位 C



- 则半加器的规则是：

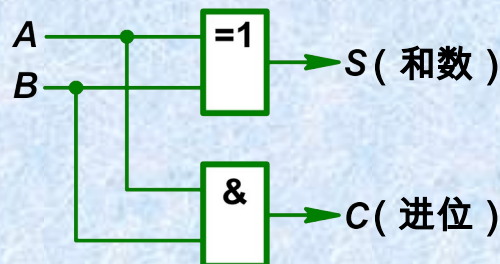
A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

- 半加器的逻辑公式为：

$$S = \bar{A}B + A\bar{B} = A \oplus B$$

$$C = AB$$

- 半加器的逻辑结构为：



1.2.5 小结

小结：二进制的运算

- 二进制数的运算有 2 类：
 - 逻辑运算： \vee ， \wedge ，NOT. 按位进行，不考虑进位
 - 算术运算： $+$ ， $-$ ， \times ， $/$. 从低位到高位逐位进行，需考虑低位的进位（借位）
- 逻辑运算可以用门电路（与门、或门、非门等）实现
- 算术运算可以表达为逻辑运算，因此二进制数的四则运算同样也可以使用门电路来实现
- 成千上万个门电路可以制作在集成电路上，工作速度极快，因而能高速度地完成二进制数的各种运算

小结：用比特表示信息的优点

1. 比特只有 0 和 1 两个符号，具有 2 个状态的器件和装置就能表示和存储比特，而制造两个稳定状态的电路又很容易
2. 比特的运算规则很简单，使用门电路就能高速度地实现二进制数的算术和逻辑运算
3. 比特不仅能表示“数”，而且能表示文字、符号、图像、声音，可以毫不费力地相互组合，开发“多媒体”应用
4. 信息使用比特表示以后，可以通过多种方法进行“数据压缩”，从而大大降低信息传输和存储的成本。
5. 使用比特表示信息后，只要再附加一些额外的比特，就能发现甚至纠正信息传输和存储过程中的错误，大大提高了信息系统的可靠性