计算机组成原理

数学与计算机科学学院
林嘉雯
ljw@fzu.edu.cn

第 2 章 运算方法和运算器

主要内容：

- 数据与文字的表示方法
- 定点的加、减法运算
- 定点的乘法运算
- 定点的除法运算
- 定点的运算器的组成
- 浮点运算方法和浮点运算器

2.1 数据与文字的表示方法

2.1.1 二进制数

- 数制的两大要素：
  - **基数R**：在某种进位制中允许使用的基本数码个数。**基数为R的数制称为R进制数。** **R**进制数的主要特点就是**逢R进1** 。
  - **权W_i**：权也称位权，指某一位i上的数码的权重值，即权与数码所处的位置i有关。$W_i = R^i$。

1、二进制数的定义

以2为基数的数制叫二进制数

二进制数有下列特征：

① 有 2 个符号表示数： 0 和 1 。

② R为2。当计数时，每一位计到2就往上进一位，即“逢二进一”。

③ 在一串数字中，上一个位的权是下一位的两倍。

- 对于整数，从右往左各位的权是1,2,4,8, ……；
- 对于小数，从左往右各位的权是1/2，1/4，1/8，1/16，1/32……。

同理，以16为基数的数制叫十六进制数。

2、不同数制的相互转换

（1）二进制数转换成十进制数

- 用十进制计数把二进制各位置的数按权展开后相加即可。

例1 $(1001.101)_2$

$=1*2^3+0*2^2+0*2^1+1*2^0+1*2^{-1}+0*2^{-2}+1*2^{-3}$

$=8+0+0+1+0.5+0+0.125$

$=(9.625)_{10}$

（2）十进制数转换成二进制数

a) 整数部分：

- 除基取余法：采用将十进制数连续除以 2 提取余数的方法，提取的余数依此为二进制的低位、次低位．．．高位。
- 减权定位法：依次与二进制权位比较，够减的为**1**，不够为**0**。

例2　求**$(116)_{10}$**的二进制数值：

$$(116)_{10}=(1110100)_2$$

b) 小数部分：

- **乘基取整法：**采用将十进制小数部分连续乘以 2 提取乘积中整数的方法，提取的整数依此是小数部分的最高位、次高位．．．
- **减权定位法**

例3 求$(0.625)_{10}$二进制数值：

故（$0.625)_{10}=(0.101)_2$

有时会出现小数部分总不为零的情况，如（$0.6)_{10}=(0.100110011)_2$…这时转换过程的结束由所要求的转换精度确定。

（3）二进制数转换成十六进制数

- 从小数点往左或往右每 4 位一组地划分，不足 4 位整数部分在前面补 0，小数部分在后面补 0，然后将每 4 位写出其对应的十六进制数即可。

例4 $(11011011.01011)_2$

$=(\underline{1101}\ \underline{1011}.\underline{0101}\ \underline{1000})_2$

$=(DB.58)_{16}$

（4）十六进制数转换成二进制数

- 直接将每位十六进制数写成 4 位二进制数即可。

例5 $(3\underline{F}5.A\underline{8}C)_{16}$

$=(0011\ \underline{1111}\ 0101.1010\ \underline{1000}\ 1100)_2$

数据的表示

真值：根据书写习惯，用正负号加绝对值表示的数值。一般用X表示。

机器数：计算机内使用的，便于机器处理的数值，包括无符号数和有符号数。

- 无符号数：整个机器字的全部二进制位均为数值位，相当于数的绝对值
- 有符号数：将数的符号位一起数码化，符号位放在有效数字的前面

2.1.2 数据格式

根据小数点的位置是否固定，计算机中常用的数据表示格式有两种

- 定点格式
- 浮点格式

1、 定点数的表示

- 定义：约定机器中所有数据的**小数点位置是固定不变的**。通常将数据表示成**纯小数**或**纯整数**。

设n+1位定点数 $x = x_n x_{n-1} x_{n-2} \cdots x_0$， 则

X_n	$X_{n-1}X_{n-2}\cdots\cdots X_0$
符号位	数值位

. 小数点隐含位置

a）定点整数格式

X_n	$X_{n-1}X_{n-2}\cdots\cdots X_0$
符号位	数值位

小数点隐含位置

b）定点小数格式

- **定点整数：**用于表示纯整数，小数点位置隐含固定在最低位之后，**最高位为符号位**。

  机器字长为n+1位，纯整数表示范围为：

$$0 \le |x| \le 2^{n}-1$$

- **定点小数：**用于表示纯小数，小数点隐含固定在最高数据位的左边，**整数位则用于表示符号位**。

  机器字长为n+1位，纯小数的表示范围为：

$$0 \le |x| \le 1-2^{-n}$$

2、浮点数的表示

- 浮点表示法：数的计阶表示方法，把数的范围和精度分开表示的方法，小数点的位置随阶数的不同而浮动。

设任意一个进制数 N 用计阶法表示为：

$$N = R^{e}.M$$ 其中，

- M ：尾数，规定是一个纯小数，且计算机中一般约定为最高有效位为1 ，称为规格化。
- e ：指数，是一个整数，计算机中称为阶码。
- R ：比例因子的基数，计算机中一般为2，隐含表示。则计算机中浮点数可以表示为：

E_s	E_{m-1} E_{m-2} … E_1 E_0	M_s	M_{n-1} M_{n-2} … M_1 M_0

实用浮点数格式：

以32位浮点数为例

- S：浮点数的符号位，1 位，0表示正数，1表示负数。
- M：尾数，23位，小数点放在尾数域的最前面。
- E：阶码，8 位，一般用移码表示，其真值记为e

则一个非规格化的32位浮点数 x 的真值可表示为：

$$x=(-1)^{s}\times(0.M)\times 2^{e}$$

规格化浮点数：

当尾数值不为0时，在用原码表示尾数的情况下，尾数域最高位必须为1，即 | M | >=0.5

若用补码表示尾数，

- 正数，尾数最高位为1，如0.1011
- 负数，尾数最高位为0，如1.0010

例如，0.001001 ⟶ $0.1001*2^{-2}$

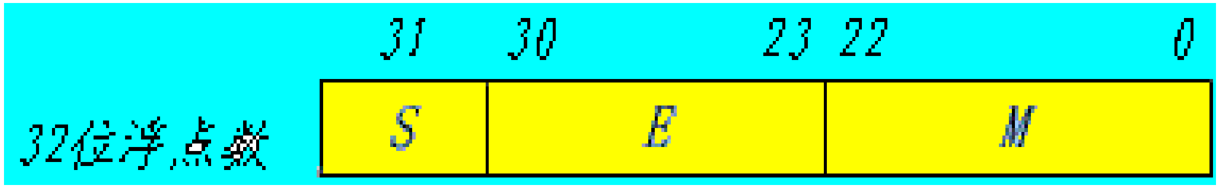
例如，0.001001 → $0.1001*2^{-2}$

阶码E一般使用**移码**表示。若E共占k位，则

$E=e+2^{k-1}$，e表示真值

注意，此时E为阶码的存储形式，e为阶码真值

例如，e=-2 → E=(-2)+128=126



0.001001 → 0 01111110 100100…0

Q：可以进一步提高精度吗？

实用浮点数格式：IEEE754标准

- 32位表示法：

	31	30　　23	22　　0
32位浮点数	S	E	M

- 64位表示法：

	63	62　　52	51　　0
64位浮点数	S	E	M

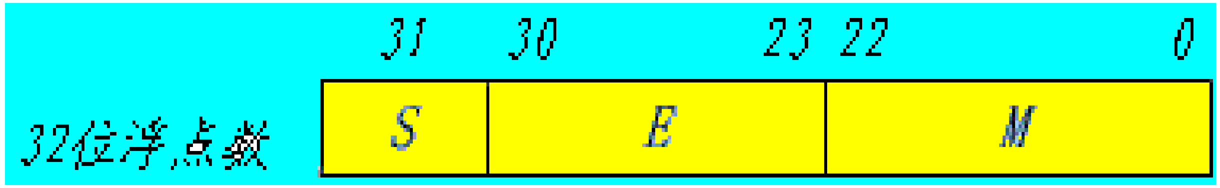
➢ S：浮点数的符号位，1 位，0表示正数，1表示负数。

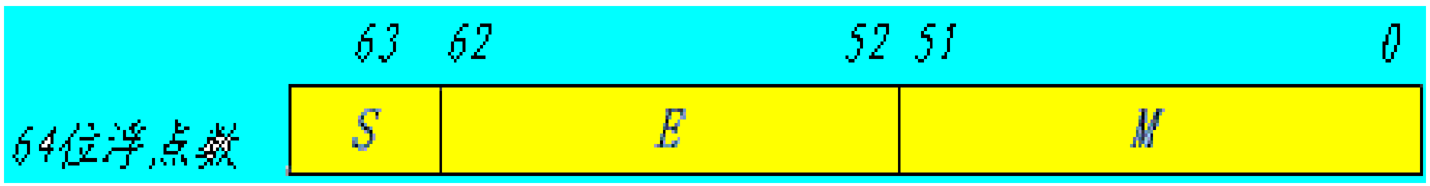
➢ M：尾数，23或52位，用规格化小数表示，小数点放在尾数域的最前面，小数点前第1位1隐含。

➢ E：阶码(8 或11位)，采用隐含移码方式来表示。

$$E=e+2^{k-1}-1$$

E=e（真值）+127/1023





则在IEEE754标准下一个规格化的32位浮点数 x 的真值可表示为：

$$x=(-1)^{s}\times(1.M)\times 2^{E-127}$$

一个规格化的64位浮点数 x 的真值为

$$x=(-1)^{s}\times(1.M)\times 2^{E-1023}$$

- 754标准浮点数数值范围的一些特殊情况：

- 当一个浮点数的尾数为0时，不论其阶码为何值，或当阶码的值遇到比机器能表示的最小值还小时，不管其尾数为何值，计算机都把该浮点数看成零，称为机器零。
- 当阶码E全为0且尾数M全为0时，表示的真值x为0；
- 当阶码E全为1且尾数M全为0时，表示的真值x为无穷大。
- 对32位的规格化浮点数，阶码E的范围为1到254，故32位浮点数的真正指数值e为-126—127

例6 若浮点数 x 的754标准存储格式为$(41360000)_{16}$，求其浮点数的十进制数值。 （P18例1）

解: 十六进制数展开后，可得二进制数格式为

0 100 0001 0011 0110 0000 0000 0000 0000

则，指数e＝阶码－127

＝10000010－01111111

＝00000011＝$(3)_{10}$

尾数为：1.M＝1.011 0110 0000 0000 0000 0000

＝1.011011

浮点数为：

$$x=(-1)^{s}\times 1.M\times 2^{e}$$

$$=+(1.011011)\times 2^{3}$$

$$=+1011.011=(11.375)_{10}$$

例7 将$(20.59375)_{10}$转换成754标准的32位浮点数的二进制存储格式。 （P18例2）

解:

$$(20.59375)_{10} = (10100.10011)_2$$
$$= 1.010010011 \times 2^4$$

则

$S = 0$,

$E = e + 127 = 4 + 127 = 131$,

$E = 10000011$

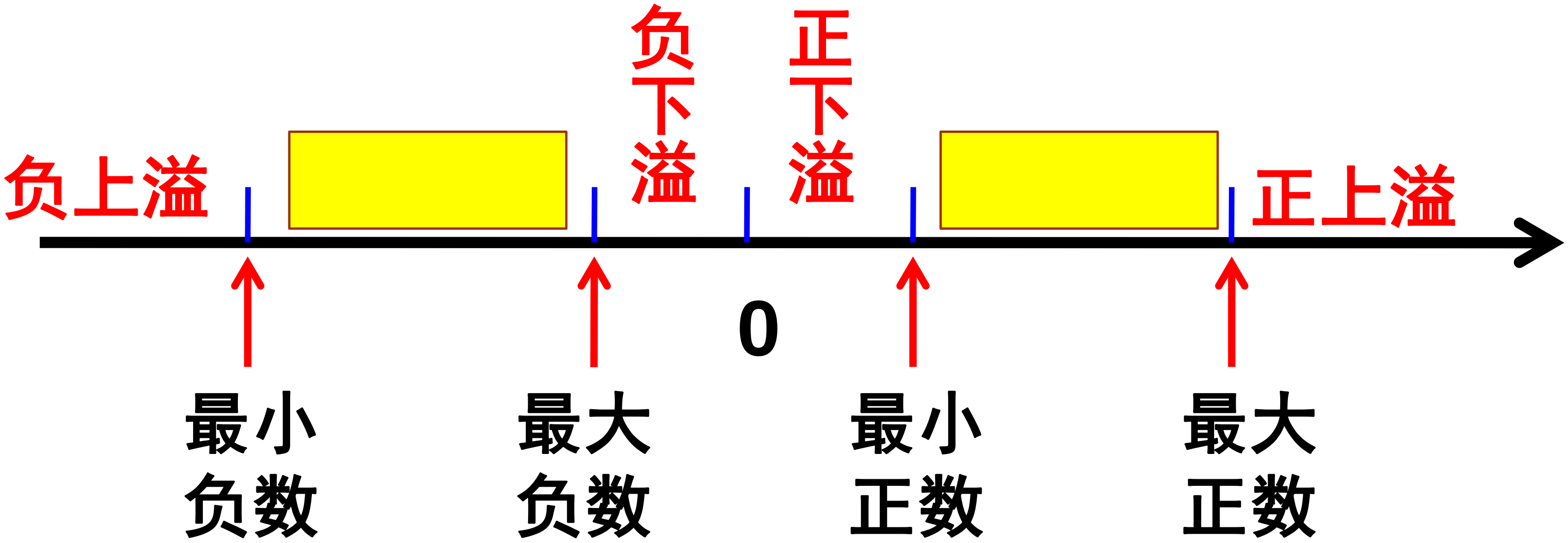
$M = 010010011$

32位浮点数存储格式为:

0 100 0001 1010 0100 1100 0000 0000 0000

$= (41A4C000)_{16}$

浮点数的表示范围



数据下溢时，浮点数值趋于0，计算机将其视为机器0处理；数据上溢时，计算机将其视为无穷大

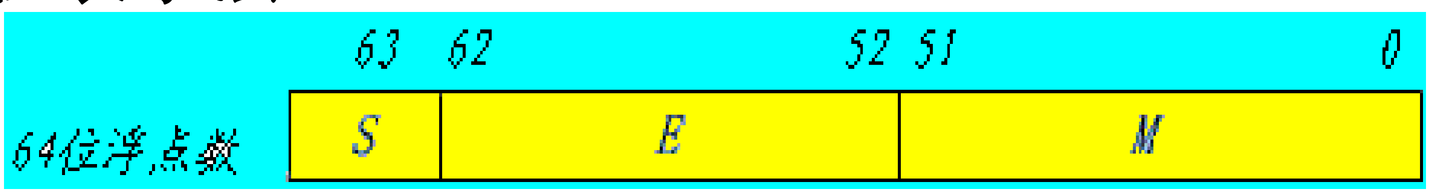
Q:浮点数的表示范围如何求得？

IEEE754标准 浮点数表示范围

- 32位表示法：（**float**）

$$2^{-126} \le |X| \le (2-2^{-23})*2^{127}$$

- 64位表示法（**double**）：



$$2^{-1022} \le |X| \le (2-2^{-52})*2^{1023}$$

浮点数格式小结（以32位为例）

	31	30 　　　　23	22 　　　　0
32位浮点数	S	E	M

非规格化：无特殊要求

规格化：尾数有要求

原码表示时	0/1	1XXXXXX
补码表示时	0	1XXXXXX
	1	0XXXXXX

计算对应真值：$x=(-1)^s\times(0.M)\times2^{E-128}$ 　　e ： -128~+127

IEEE754：尾数只有原码； 　尾数域前隐含1；

阶码为隐含移码，不用全0/全1

计算对应真值：$x=(-1)^s\times(1.M)\times2^{E-127}$ 　　e ： -126~+127

2.1.3 数的机器码表示

1、真值与机器数

- 真值：用正负号加绝对值表示的数值。
- 机器数：将数符一起数码化的数。常用机器数包括原码、补码、反码和移码表示法。

机器数形式的二进制位数因受机器字长的限制，其数的表示范围和精度相应受到限制，无法表示时，便产生溢出。

例8 真值： +1011 -1011

8位字长定点数：0 0001011 1 0001011

真值： +0.1011 -0.1011

8位字长定点数：0 .1011000 1 .1011000

2、原码表示法 $[X]_{原}$

定义：设原码形式为 $x = x_n x_{n-1} \cdots x_1 x_0$。

n+1位字长的原码为：

$$小数：[X]_{原} = \begin{cases} X & 1 > X \geqslant 0; \\ 1-X = 1+|X| & -1 < X \leqslant 0。\end{cases}$$

$$整数：[X]_{原} = \begin{cases} X & 2^n > X \geqslant 0; \\ 2^n-X = 2^n+|X| & -2^n < X \leqslant 0。\end{cases}$$

- 表示方法：最高位表示数的符号，其他位表示数值位。
  - 符号位：0—正数，1—负数。
  - 数值位：与绝对值相同。

X=1011， Y＝－1011， 则：

$[X]_{原}$＝0,1011； $[Y]_{原}$＝1,1011；

X=0.1101， Y＝- 0.1101， 则：

$[X]_{原}$＝0.1101； $[Y]_{原}$＝1.1101；

X=1011， Y＝- 0.1101， 求X和Y的8位原码机器数。

$[X]_{原}$＝0,0001011； $[Y]_{原}$＝1. 1101000；

$[0]_{原}$＝？

- **0 的表示：** 0 的原码表示有两种形式，即分别按照正数和负数表示。

$[+0]_{原} = 00\cdots0$　　　　$[-0]_{原} = 10\cdots0$

- **表示范围：** 对于n＋1位原码机器数X，它所能表示的数据范围为：

包括1位符号位，n位数值位

整数：　$-(2^n-1) \leq X \leq (2^n-1)$

小数：　$-(1-2^{-n}) \leq X \leq (1-2^{-n})$

2、反码表示法$[X]_{反}$

- 定义：正数反码与原码相同，负数的反码将数除符号位外按位求反。
- 反码的特点：
  - 0的反码也有两个，

$$[+0]_{反}=00000000，$$

$$[-0]_{反}=11111111。$$

  - 反码的数值范围与原码相同。

3、补码表示法 $[X]_{补}$

- 定义：把某数X加上模数M，称为以M为模的X的补码。

$$[X]_{补}=（M+X）_{模M}。$$

机器内，字长为n+1的整数模为2^{n+1}，小数的模为2。

小数：$[X]_{补}=\begin{cases} X & 1>X\geq 0; \\ 2+X=2-|X| & -1\leq X<0。\end{cases}$

整数：$[X]_{补}=\begin{cases} X & 2^n>X\geq 0; \\ 2^{n+1}+X=2^{n+1}-|X| & -2^n<X<0。\end{cases}$

- 原码转化成补码方法：
  - 正数的补码与原码相同；
  - 负数时：
    ① 求得该数的反码，然后在末位加1。
    ② 符号位不变，尾数部分自右向左，第一个1及以前的各位0保持不变，以后的各位按位取反。
- 补码转换成原码、真值方法：
  - 正数的原码与补码相同；
  - 负数时，符号位不变，尾数先按位取反，然后在末位加1。
  - 将原码的符号位用“+”、“-”号表示即为真值
- 上述转换不包括-0及补码的负数最小值。

例9 设机器位长为8位，用补码表示±0.1001和±59 。

$[+0.1001]_{补}= [+0.1001000]=0.1001000$

$[-0.1001]_{补}= 2 -0.1001000=1.0111000$

$[+59]_{补}= [00111011]=00111011$

$[-59]_{补}= 2^8 -00111011=11000101$

- 补码特点：
  - 可实现变减为加运算，且补码的符号位由计算获得。
  - 0的补码只有一个，就是n位的零。

    $[\pm 0]_{补}= 2^8 \pm 00000000=00000000$
  - N+1位字长补码的数值范围

    整数： $-2^n \leqslant X \leqslant (2^n-1)$

    小数： $-1 \leqslant X \leqslant (1-2^{-n})$

- 补码转换十进制真值的方法：
  - 设补码形式为 $x = x_n\, x_{n-1} \dots x_1\, x_0$。

整数：

$$x = -2^n x_n + \sum_{i=0}^{n-1} 2^i x_i$$

小数：

$$x = (-1) x_n + \sum_{i=n-1}^{0} 2^{(i-n)} x_i$$

  - 上述符号直接由运算得到

4、移码表示法 $[X]_{移}$

- 移码是在真值X上加上一个常数（偏移量），相当于X在数轴上向正方向偏移了若干单位。通常，当字长为n+1时，这个偏移量取2^n。
- 定义：计算机中对k+1位字长的带符号数，其真值X所对应的移码为:

$$[X]_{移}= 2^k+X，\quad -2^k \leqslant X \leqslant 2^k-1。$$

例10 求8位长±59的移码

$[+59]_{移}=10000000+111011=10111011$

$[-59]_{移}=10000000-111011=01000101$

移码特点：

- 移码的符号位与原码、补码相反，1为正，0为负。
- x的移码和补码符号位相反其余位相同，故移码可以先求数的补码，再将符号取反即得。
- 0的移码只有一个，$[0]_{移}=10000000$。
- 移码一般用于浮点数的阶码表示。
- 移码一般以整数形式出现，其数值范围与补码相同。

例11 假设由S,E,M三个域组成的一个32位二进制字所表示的非零规格化浮点数 x,真值表示为：

$$x=(-1)^s\times(1.M)\times 2^{E-128}$$

问：它所表示的规格化的最大正数、最小正数、最大负数、最小负数是多少？ （P23例9）

解:

阶码用移码表示，8位；尾数用原码，23位。

(1)最大正数

阶码为最大正数， 尾数为最大正数

存储码： 0 11 111 111 111 111 111 111 111 111 111 11

真值: $x=[1+(1-2^{-23})]\times 2^{127}$

(2)最小正数

阶码为最小负数（绝对值最大），尾数为最小正数

0 00 000 000 000 000 000 000 000 000 000 00

$$x = 1.0 \times 2^{-128}$$

(3)最小负数（绝对值最大）

阶码为最大正数，尾数为最小负数（绝对值最大）

1 11 111 111 111 111 111 111 111 111 111 11

$$x = -[1 + (1 - 2^{-23})] \times 2^{127}$$

(4)最大负数（绝对值最小）

阶码为最小负数（绝对值最大），尾数为最大负数（绝对值最小）

1 00 000 000 000 000 000 000 000 000 000 00

$$x = -1.0 \times 2^{-128}$$

2.1.4 字符与字符串的表示方法

美国国家信息交换标准代码，简称**ASCII码**

- 7位二进制编码,能表示2^7=128种国际上最通用的西文文字。
- ASCII码包括 4 类最常用的字符。
  - 数字：包括0~9 10个数字字符。
  - 通用字符：“＋”、“－”、“＝”、“*”、“/”等共32个。
  - 字母：包括26个大写字母和26个小写字母。
  - 控制字符：包括空格ＳＰ、回车ＣＲ、换行ＬＦ等共34个。
- ASCII编码有一定的规律。

高3位 低4位	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	、	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	“	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	‘	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	-	o	DEL

2.1.5 汉字的表示方法

- 输入码
  - 数字码
  - 拼音码
  - 字形码
- 机内码
- 输出码
  - 点阵式
  - 矢量式

输入码、机内码和输出码是计算机中用于输入、内部处理和输出三种不同用途的编码。

2.1.6 校验技术

校验的方法是让写入的信息符合某种规律，在读出时检验信息是否符合这一规律，如符合可判定读出信息正确，否则有误。

目前使用的校验方法常采用冗余校验思想，即：

有效信息位+校验位 → 校验码 → 译码纠错

- 奇偶校验码

有效信息位+1位校验位 ⟶ 校验码

编码规则：约定校验码中1的个数为奇数/偶数。

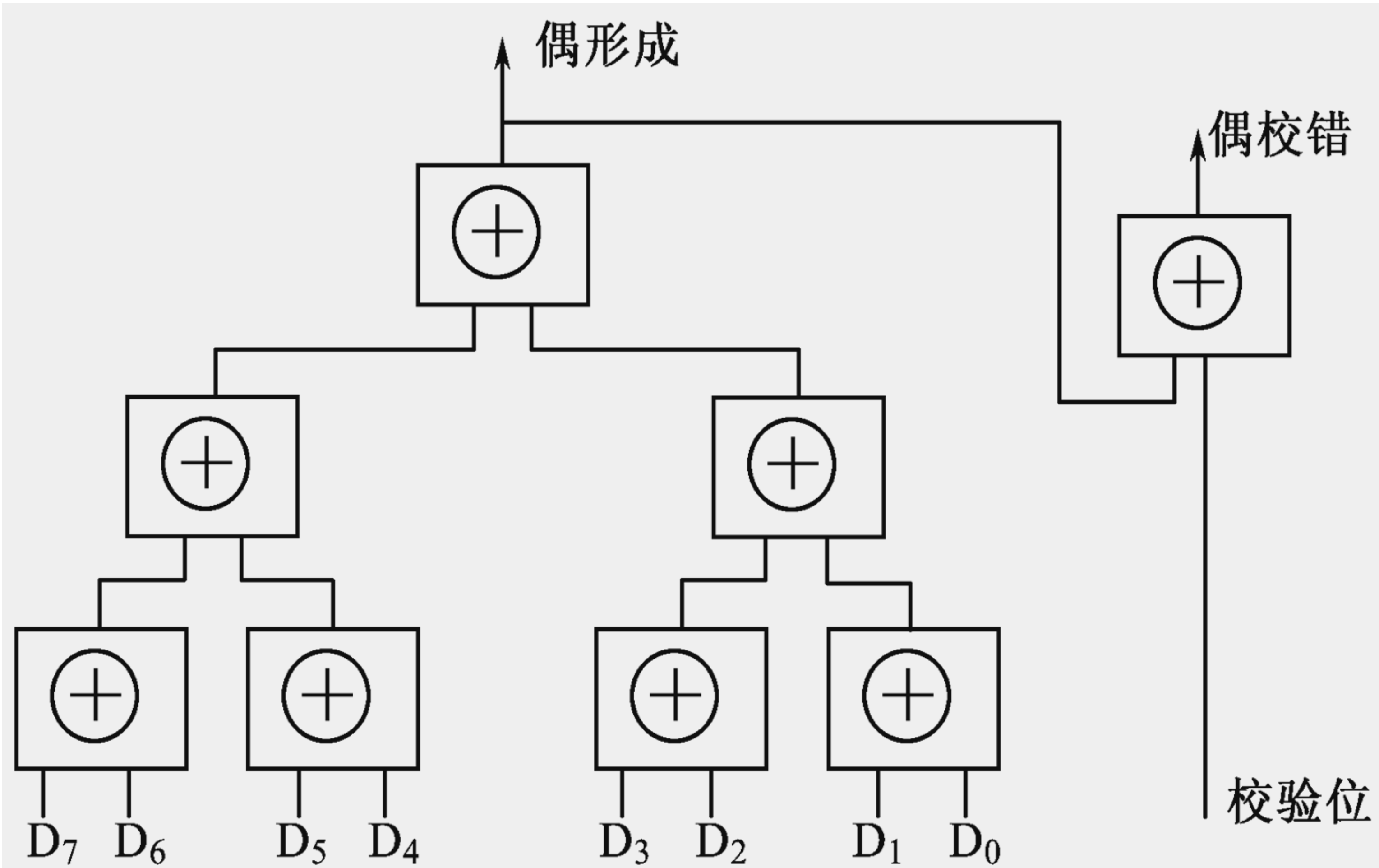
校验位公式： $C = x_0 \oplus x_1 \oplus \cdots \oplus x_{n-1}$。（偶）

$\overline{C} = x_0 \oplus x_1 \oplus \cdots \oplus x_{n-1}$。（奇）

例如：待编有效信息　10110001

奇校验码　101100011

偶校验码　101100010



偶校验判错实现电路

奇偶校验实现简单，但缺点是不能纠错、无法检测出偶数个错误。

其他校验方式如循环冗余校验码(**CRC**)可以实现纠错。