

## 2.1 数制与编码（下）

### 2.1.3 BCD码

8421码（最常用）	注意：如果两个8421码相加后的和小于9（10进制），需要加6（10进制）进行修正
余3码（无权码）	在8421码的基础上加（0011），因为每个数都多3，所以称为余3码 8--->1011
2421码（有权码）	权值由高到低为 2, 4, 2, 1 >=5的4位二进制数中最高位为1，<5的最高位为0 5--->1011

### 2.1.4 字符与字符串

字符串编码ASCII码	7位二进制编码
汉字的表示和编码	每个编码用两个字节表示
种类	输入编码 计算机输入 汉字内码 计算机内部处理 汉字字形码 计算机输出
字符串存放	小端模式 将数据的最高有效字节存放在高地址单元中 大端模式 将数据的最高有效字节存放在低地址单元中
从低地址到高地址逐字符存储，常采用'\0'作为结尾标志	

### 2.1.5 校验码

概念：能够发现或者自动纠错的数据编码	
原理：通过添加一些冗余码，实现检验或者纠错编码	
奇偶校验码	奇校验码：有效信息位和校验位中 1 的个数为奇数 偶校验码：有效信息位和校验位中 1 的个数为偶数 码距：2
但是如果编码中出现偶数位错误，无法检测	
海明码	在有效信息位中添加几个校验码形成海明码 不仅可以发现错位，还可以对错位进行纠错 编码最小码距L越大，检测位数越多，纠错能力越强（纠错能力恒小于等于检测能力） 先检错，才能纠错
分类	海明码有1位纠错，2位检错能力
补充	为了区分1位错和2位错，还需添加"全校验位"对整体进行偶校验 注意：有的题目位置编号可能是从小到大的，但处理方法雷同
CRC（循环冗余码）	常用于大量的数据传送时的校验 接收到循环冗余码后，对生成多项式做模2除法，余数为0则无错误 余数不为0，对相应位置取反
检错、纠错能力	可检测出所有奇数个错误 可检测出所有双比特的错误 可检测出所有小于等于校验位长度的连续错误 若选择合适的生成多项式，且 $2^R \geq K + R + 1$ ，则可纠正单比特错

## 2.2.1 定点数的表示

原码表示法：正零 00000 负零 10000

反码表示法：正零 00000 负零 11111

两种表示法

补码对真值零表示是唯一的

唯一表示

真值零表示方式

移码对真值零也是唯一的

### 有符号与无符号

无符号数：整个机器字长的全部二进制位均为数值位，没有符号位

有符号数：最高位的0/1分别表示正/负

### 机器数定点表示

定点小数

定义：定点小数是纯小数，约定小数点位置在符号位之后、有效数值部分最高位之前。

定点整数

定义：定点整数是纯整数，小数点位置在有效数值部分的最低位之后

### 机器数表示方法

原码表示法

最高位为符号，其余各位为表示数的绝对值

纯小数的原码

字长为  $n+1$ ，范围为： $-(1-2^{-(n)}) \leq X \leq 1-2^{-(n)}$

关于原点对称

纯整数的原码

字长为  $n+1$ ，范围为： $-(2^n-1) \leq X \leq (2^n-1)$

关于原点对称

由于原码表示中加减法计算复杂，补码表示法可以更好地去表示加减法

补码表示法

纯小数补码

表示范围  $-1 \leq X \leq 1-2^{-(n)}$

比原码多表示-1

纯整数补码

表示范围  $-2^n \leq X \leq 2^n-1$

比原码多  $-2^n$

补码的算数移位

实现除法功能：符号位与数值位一起右移，保持原符号位数值不变

变形补码（模4补码）

正：00

负：11

反码表示法

原码与补码相互转化的过渡

纯小数反码

表示范围： $-(1-2^{-(n)}) \leq X \leq 1-2^{-(n)}$

关于原点对称

纯整数反码

表示范围： $-(2^n-1) \leq X \leq 2^n-1$

关于原点对称

移码表示法

常用来表示浮点数的阶码，只能表示整数

最小值  $-2^n$ （全0） 最大值  $2^n-1$ （全1）

移码大 真值就大

### 不同表示方法间的相互转换

原码  $\longleftrightarrow$  补码

正数：补码与原码表示相同

负数：原码符号位不变 数值部分按位取反

不同机器数的转换关系图

## 2.2.2 定点数的运算（上）



## 2.2.2 定点数的运算（下）

### 定点数的乘法运算

原码一位乘法

符号位：由两个数的符号位异或

数值：两个数的绝对值相乘之积

符号不参与运算

部分积 2 位 乘数 0 位

累加次数  $n$

符号位参与运算

部分积 2 位 乘数 1 位

累加次数  $n+1$

移位 方向：向右 次数： $n$  每位次数：1

补码一位乘法

### 定点数的除法运算

原码除法运算（不恢复余数法）

商符 和 商值分开运算

商符由两个操作数异或

符号位不参与运算

加减次数 $n+1$ 或 $n+2$

若最终余数为负，需要恢复余数

符号位参与运算

补码除法运算（加减交替法）

加减次数 $n+1$

商末衡置为一

移位 方向：左 次数： $n$

### 强制类型转换

将所给数字写为二进制，然后按照转换成不同的码的规则进行读出

### 2.3.1浮点数的表示





## 2.3.2 浮点数的加减运算

### 运算步骤

对阶：小阶看齐大阶，阶码小的尾数右移一位，阶加一，直到阶码相等

尾数求和 尾数按照定点数加减规则运算

最高数值位与符号位不同即为规格化形式

规格化

左规：尾数左移1位，和的阶码减1 直到00.1XX或者11.0XXX

右规：尾数求和结果溢出（10.XXX或者01.XXX）尾数右移一位，和的阶码加1

舍入

0舍1入法

尾数右移时，移去的最高数值位为0，则舍去

尾数左移时，移去的最高数值位为1，则尾数末位加1

恒置1法

尾数右移，不论最高数值位丢掉的是1还是0，都将尾数末位恒置为1

只有右规后，仍然溢出，此时才是真正溢出

溢出判断

上溢出：进入中断处理

下溢出：按机器零处理

强制类型转换

char-->int 在前面补0

int <-----> unsigned 彼此都可能因为溢出丢失数据

float转换为int可能会出现精度损失和溢出

int<----->float int转换为float 可能会出现数据舍入

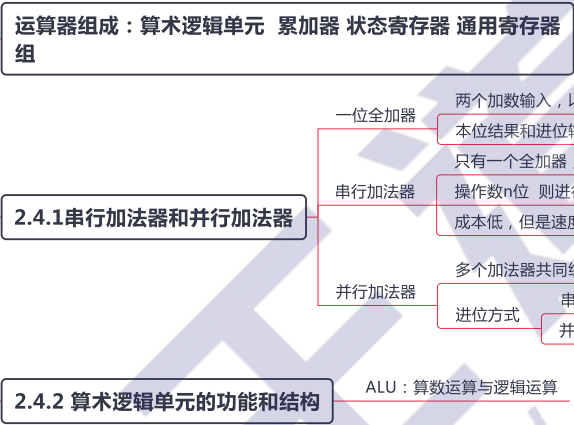
可能会导致溢出，此时需要再一次右规

### 边界对齐

现代计算机通常是按字节编址，即每个字节对应1个地址

通常也支持按字、按半字、按字节寻址

## 2.4 算术逻辑单元 (ALU)



提高并行加法器速度的关键在于加快进位产生和传递速度