



百年同济  
TONGJI UNIVERSITY

# 第三章 运算方法和运算部件

---

## 3.1 概述

## 3.2 定点加减运算

## 3.3 定点原码一位乘法

## 3.4 定点补码一位乘法

## 3.5 定点原码一位除法

## 3.6 定点补码一位除法

## 3.7 逻辑运算的实现



# 第三章 运算方法和运算部件

---

百年同济  
TONGJI UNIVERSITY

## 3.8 浮点数四则运算

## 3.9 BCD码运算

## 3.10 运算部件组织



百年同济  
TONGJI UNIVERSITY

# 主要知识点

---

- **掌握定点数加减运算方法和实现的原理及溢出的判断方法**
- **掌握定点数一位乘除运算方法和实现的原理**
- **掌握浮点数四则运算方法**
- **掌握运算部件组成原理**



百年同济  
TONGJI UNIVERSITY

## 3.1 概述

### 1. 运算器的分类

- (1) 串行运算器和并行运算器
- (2) 定点运算器和浮点运算器
- (3) 二进制运算器和十进制运算器

### 2. 运算器的主要指标

- (1) 机器字长



百年同济  
TONGJI UNIVERSITY

## 3.1 概述

**机器字长—运算器一次能处理的二进制位数称为机器字长。**

### **(2) 运算速度**

#### **① 普通法**

**是以完成一次加法或一次乘法运算所需的时间,或用每秒能完成算术运算的平均次数作为运算速度。**



百年同济  
TONGJI UNIVERSITY

## 3.1 概述

### ② 吉布森 (Gibson)法

$$T_m = \sum_{i=1}^n f_i t_i$$

**$f_i$ 为第*i*种指令的执行频度**  
 **$t_i$ 为第*i*种指令的执行时间**



百年同济  
TONGJI UNIVERSITY

## 3.1 概述

### ③基准法

用同一个程序，在不同的机器上运行所需的时间，为该机器的运算速度。

### ④MIPS

MIPS是Million Instructions Per Second的缩写，每秒处理的百万级的机器语言指令数。这是衡量CPU速度的一个指标。

## 3.1 概述

---

$$MIPS = \frac{CLOCK}{CPI \times 10^6}$$

**CPI—每条指令的平均时钟周期数**





百年同济  
TONGJI UNIVERSITY

## 3.2 定点加减运算

### 1. 定点数的加减运算

计算机中基本采用补码，补码加减运算基本关系：

$$(X+Y)_{\text{补}} = X_{\text{补}} + Y_{\text{补}}$$

$$(X-Y)_{\text{补}} = X_{\text{补}} + (-Y)_{\text{补}}$$

运算规则：

(1) 参与运算的操作数用补码表示，符号位作为数的一部分直接参与运算，运算结果为补码表示



百年同济  
TONGJI UNIVERSITY

## 3.2 定点加减运算

**(2) 如操作为加，则两数直接相加。**

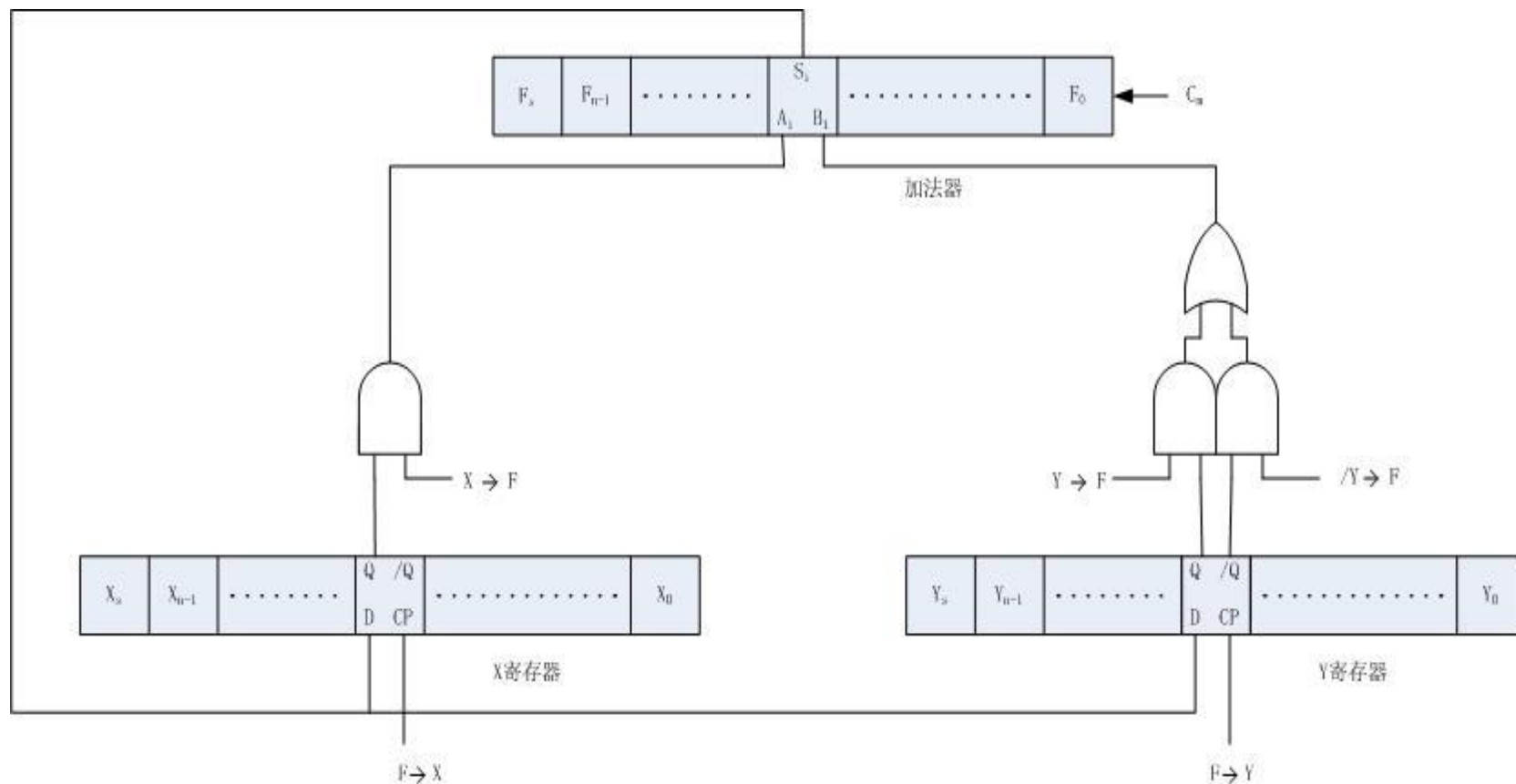
**(3) 如操作为减，则将减数变补后再与被减数相加。**

**实现补码加减运算逻辑电路**



百年同济  
TONGJI UNIVERSITY

## 3.2 定点加减运算





百年同济  
TONGJI UNIVERSITY

## 3.2 定点加减运算

### 2. 溢出判别

**当运算的结果超出允许的表示范围，则产生溢出，正超出称为正溢出，负超出称为负溢出。**



百年同济  
TONGJI UNIVERSITY

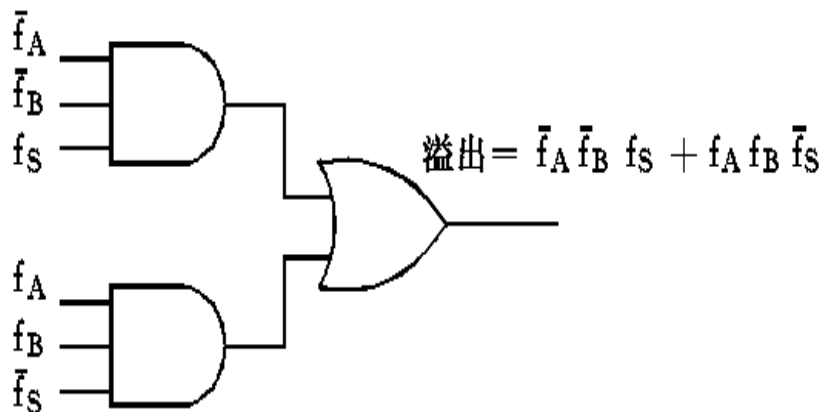
## 3.2 定点加减运算

### (1) 溢出判断逻辑一

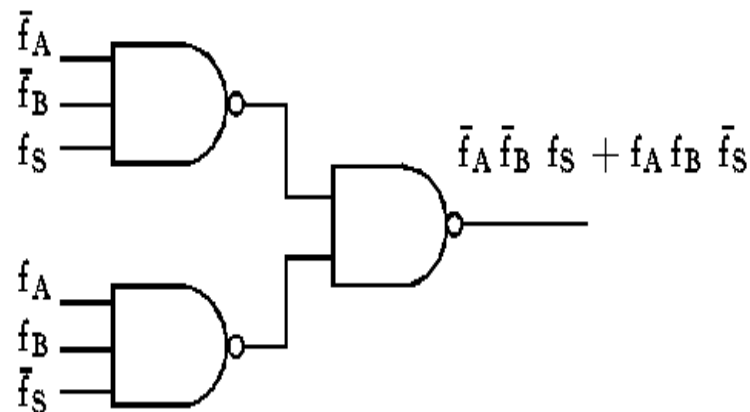
$$\text{溢出} = \bar{F}_a \bar{F}_b F_s + F_a F_b \bar{F}_s = 1$$

**F<sub>a</sub>, F<sub>b</sub>**—加数, 被加数符号位

**F<sub>s</sub>**—运算和符号位



(a)



(b)



百年同济  
TONGJI UNIVERSITY

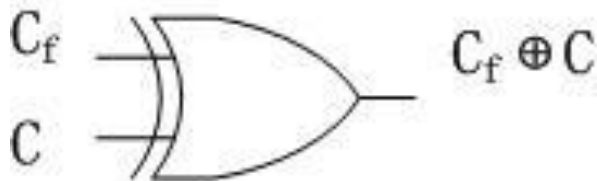
## 3.2 定点加减运算

### (2) 溢出判断逻辑二

$$\text{溢出} = C_f \oplus C = 1$$

$C_f$  — 符号位运算产生的进位

$C$  — 最高有效位产生的进位





百年同济  
TONGJI UNIVERSITY

## 3.2 定点加减运算

### (3) 溢出判断逻辑三

单符号位的信息量只能表示两种可能，数为正或为负，如产生溢出，就会使符号位含义产生混乱。将符号位扩充到二位，采用变形补码（或称模4补码），就能通过符号位直接判断是否溢出。变形补码定义：

$$[X]_{\text{变形补}} = \begin{cases} X & 0 \leq X \leq 2^{n-1}-1 \\ 2^{n+1}-|X| & -2^{n-1} \leq X < 0 \end{cases}$$



百年同济  
TONGJI UNIVERSITY

## 3.2 定点加减运算

变形补码用00表示正，11表示负。

计算结果符号位为：

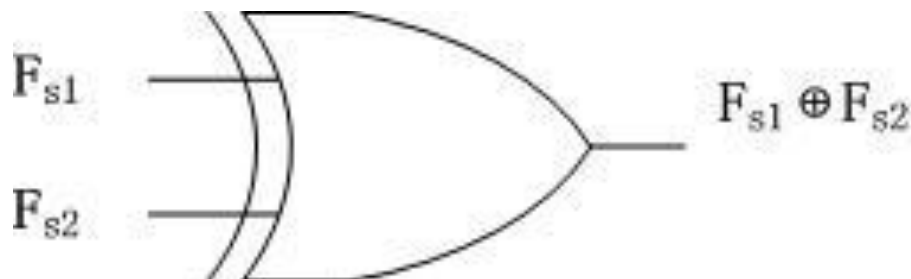
00—结果为正，无溢出；

11—结果为负，无溢出；

01—结果为正溢出；

10—结果为负溢出；

溢出 =  $F_{s1} \oplus F_{s2}$







百年同济  
TONGJI UNIVERSITY

## 3.2 定点加减运算

**注意：数据在主存中仍为单符号，运算时传送到运算器时扩充成双符号，运算结束后紧缩成单符号位存入主存中。**



百年同济  
TONGJI UNIVERSITY

## 3.3 定点原码一位乘法

### 两个原码数相乘

- 数值则为两数绝对值之积  $P = |X| \times |Y|$
- 其乘积的符号为相乘两数符号的异或值

$$S_p = S_x \oplus S_y$$

算法:

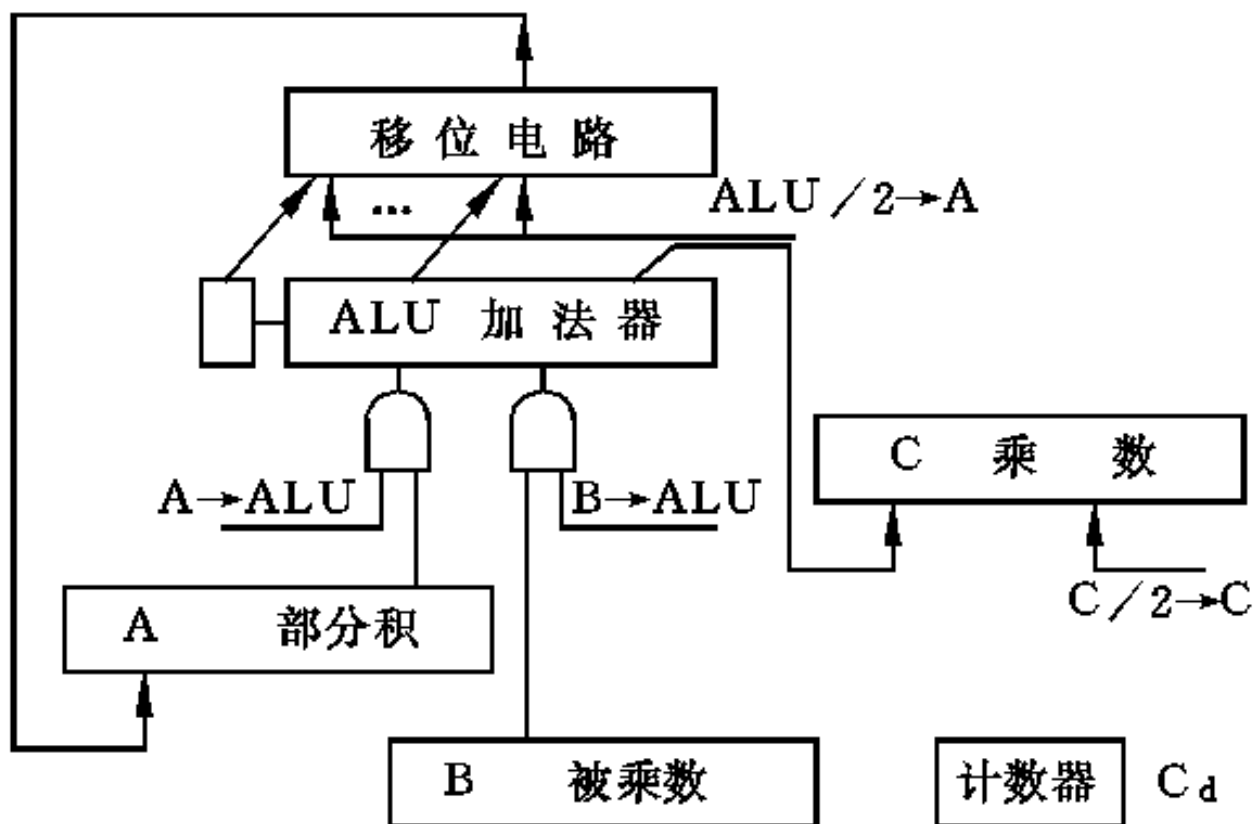
- ① 根据乘数最末一位判断  
最末位是1, 加被乘数, 右移一位  
最末位是0, 右移一位
- ② 如乘数为n位, 则需要需要进行n次累加移位



百年同济  
TONGJI UNIVERSITY

## 3.3 定点原码一位乘法

### 原码一位乘法逻辑图





百年同济  
TONGJI UNIVERSITY

## 3.4 定点补码一位乘法

**运算规则：**

- ① 符号位参加运算,被乘数是两个符号位,乘数是一个符号位。
- ② 在乘数的最后一位加个0
- ③ 判断乘数最末两位

00	部分积右移一位
11	部分积右移一位
01	部分积加被乘数右移一位
10	部分积减被乘数右移一位

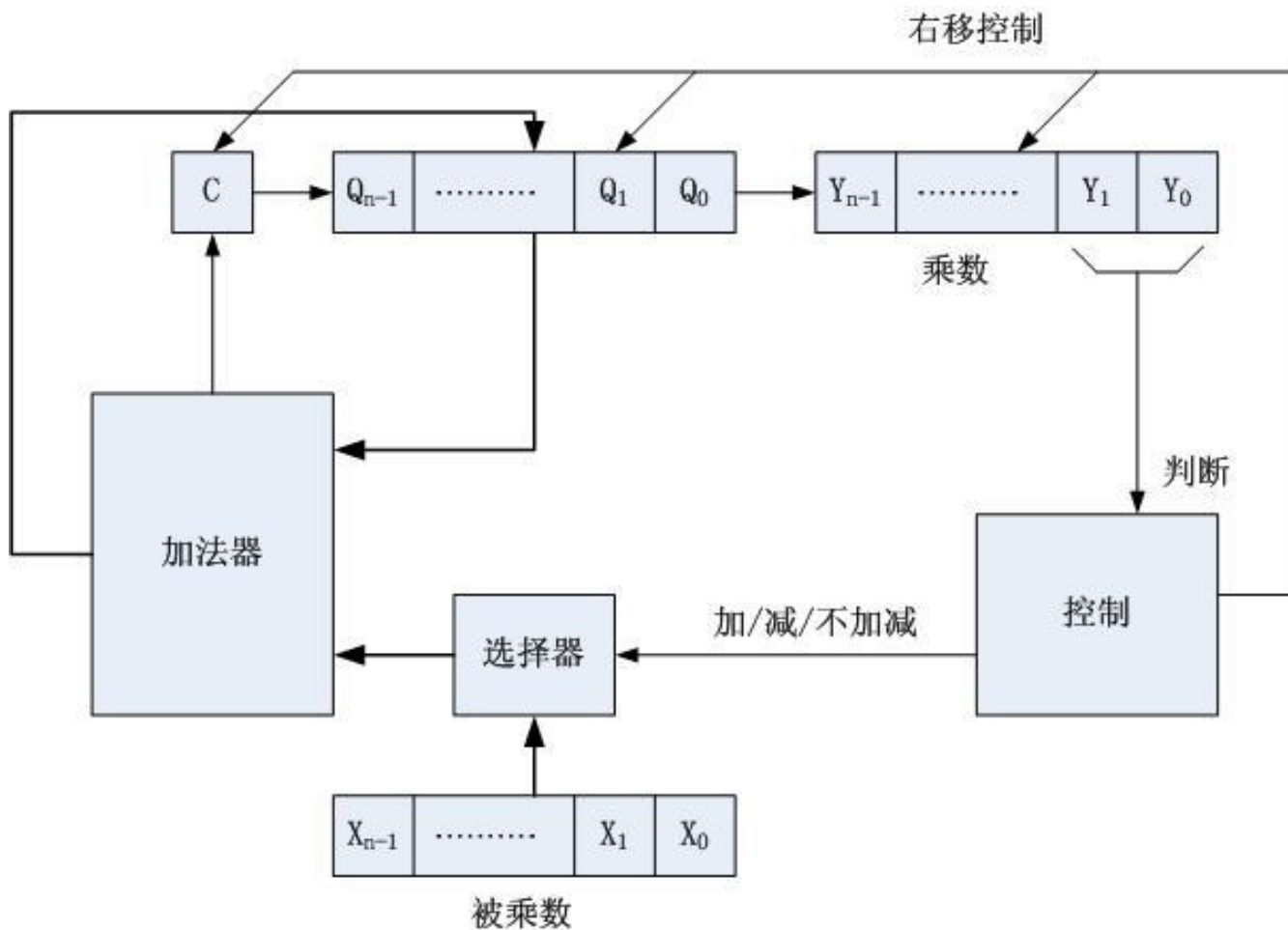
**最后一步只加减不移位**

- ④ 移位按补码方式移位，即当部分积最高位为0时，移入0，为1时，移入1。



百年同济  
TONGJI UNIVERSITY

## 3.4 定点补码一位乘法





百年同济  
TONGJI UNIVERSITY

## 3.5 定点原码一位除法

**运算规则：**

- ① 商的符号位同定点数原码乘法的处理方法： $S_p = S_x \oplus S_y$
- ② 两数的绝对值部分进行相除



百年同济  
TONGJI UNIVERSITY

## 3.5 定点原码一位除法

**算法：**

- ① 被除数先减除数，结果（余数）为正商上1；  
结果（余数）为负商上0；
- ② 结果（余数）左移一位
- ③ 如上次商为1 时，则被除数减除数；  
如上次商为0 时，则被除数加除数；  
结果（余数）为正商上1；  
结果（余数）为负商上0；



百年同济  
TONGJI UNIVERSITY

## 3.6 定点补码一位除法

### 介绍加减交替法

运算规则：

如被除数与除数同号，先减除数

如被除数与除数异号，先加除数

然后：

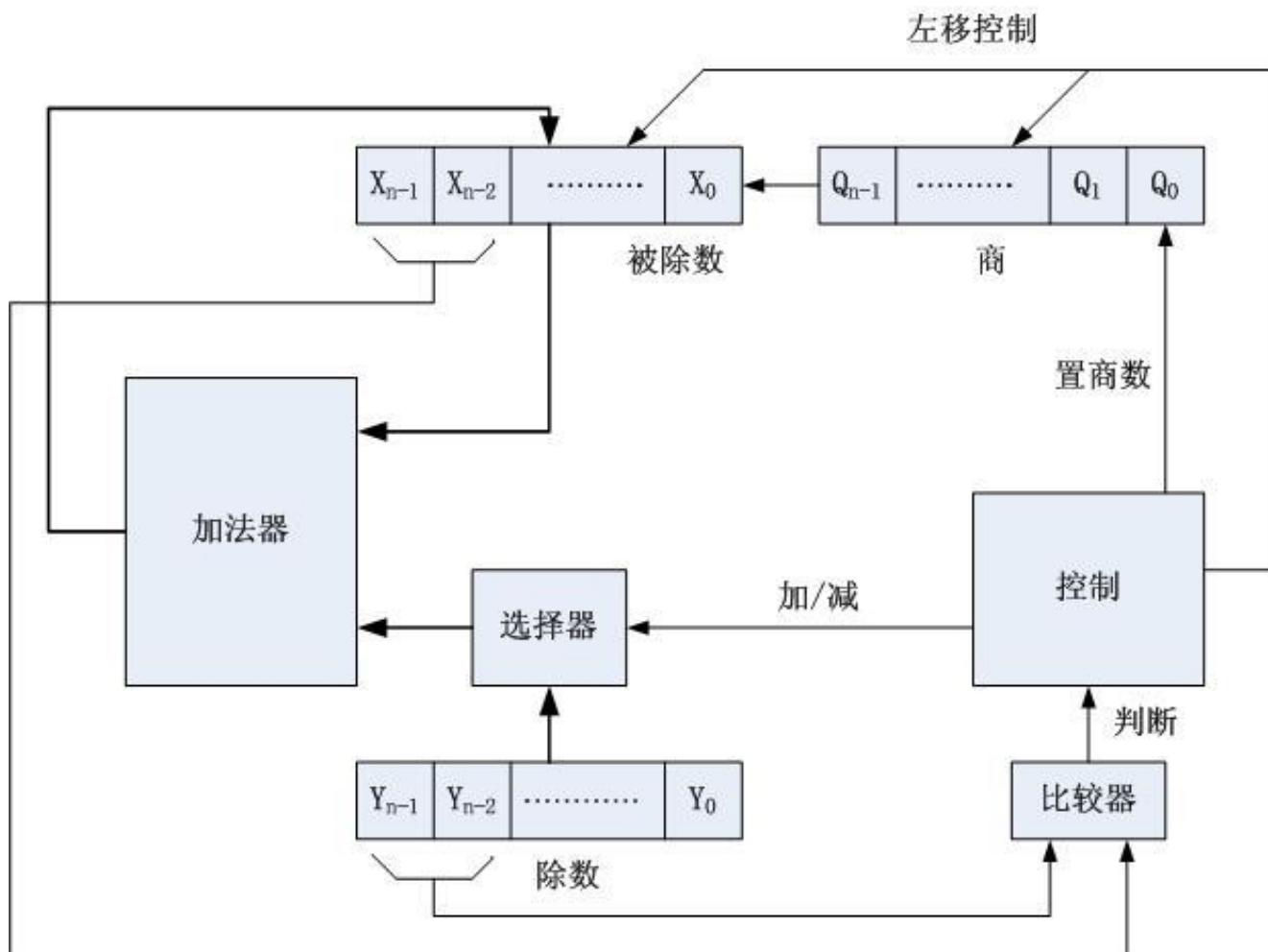
- ① 余数和除数同号，商上1，余数左移一位，下次减除数
- ② 余数和除数异号，商上0，余数左移一位，下次加除数





百年同济  
TONGJI UNIVERSITY

## 3.6 定点补码一位除法





百年同济  
TONGJI UNIVERSITY

## 3.7 逻辑运算的实现

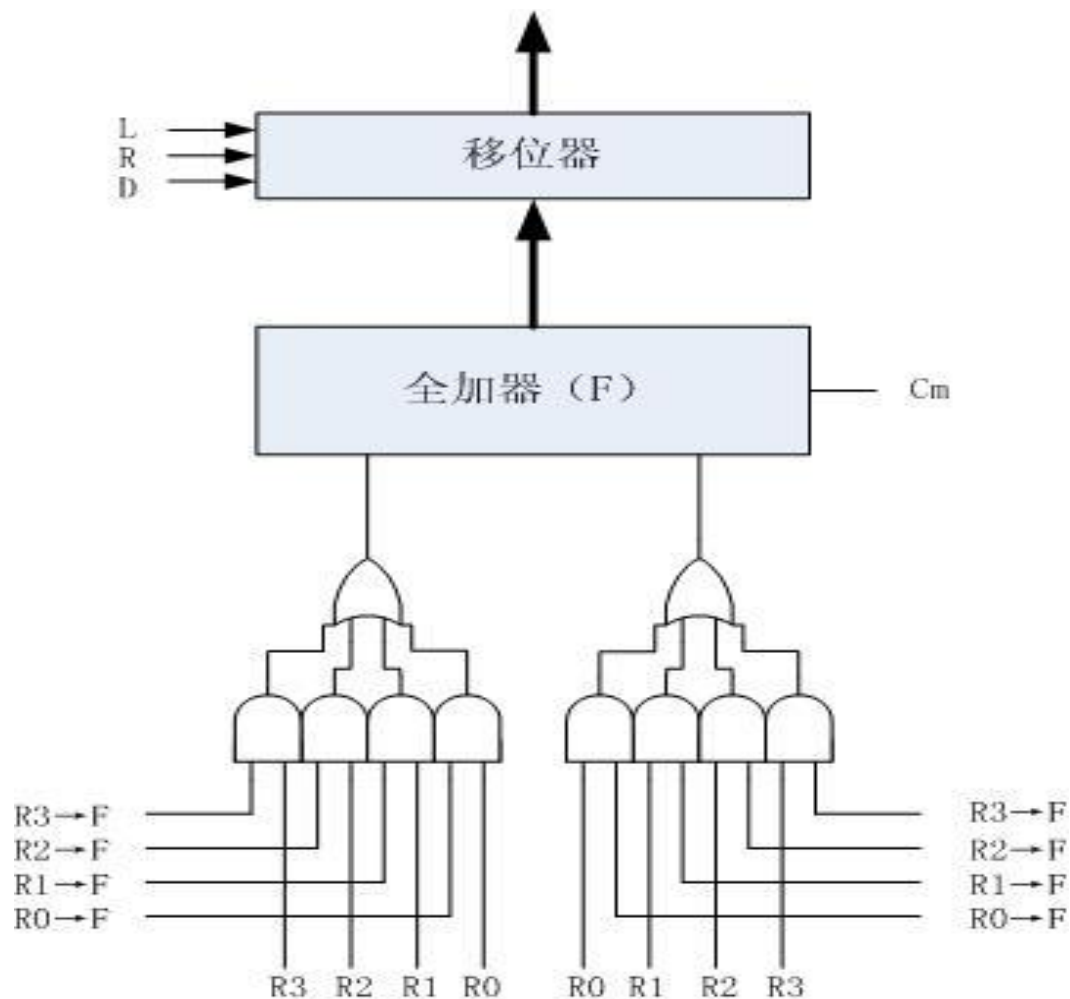
### 1. 逻辑运算

**基本的逻辑运算是求反、或、与、异或等，这些逻辑运算是通过在原有加法器上再附加部分线路实现的。**

### 2. 移位操作

**一般由移位器实现，移位器放在全加器输出端。**

## 3.7 逻辑运算的实现





百年同济  
TONGJI UNIVERSITY

## 3.8 浮点运算

### 3.8.1 浮点数的加减法运算

设有两浮点数 $X$ 、 $Y$ ,实现 $X \pm Y$ 运算, 其中:  
 $X = M_x \times 2^{E_x}$ ;  $Y = M_y \times 2^{E_y}$ 。均为规格化数。  
执行以下五步完成运算。

#### (1) “对阶” 操作

**对阶的规则:**

**阶码小的数向阶码大的数对齐, 即: 阶码小的数的尾数右移并相应增大阶码。**



百年同济  
TONGJI UNIVERSITY

## 3.8 浮点运算

**比较两浮点数阶码的大小，求出其差 $\Delta E$ ，并保留其大值 $E$ ， $E = \max(E_x, E_y)$ 。**

**当 $\Delta E \neq 0$ 时，将阶码值小的数的尾数右移 $\Delta E$ 位，并将其阶码值加上 $\Delta E$ ，使两数的阶码值相等，这一操作称之为“对阶”。尾数右移时：**

**对原码表示的尾数，符号位不参加移位，尾数数值部分的高位补0；**

**对补码表示的尾数，符号位参加右移，并保持原符号位不变。**



百年同济  
TONGJI UNIVERSITY

## 3.8 浮点运算

### (2) 尾数的加/减运算

执行对阶后，两尾数进行加/减运算，得到两数之和/差。

### (3) 规格化操作

规格化的目的是使尾数部分的绝对值尽可能以最大值的形式出现。设尾数M的数值部分有n位，规格化数的范围为： $\frac{1}{2} \leq | [M]_{\text{原}} | \leq 1 - 2^{-n}$ ， $\frac{1}{2} \leq [M]_{\text{补}} \leq 1 - 2^{-n}$  (当M为正)， $\frac{1}{2} \leq | [M]_{\text{补}} | \leq 1$  (当M为负)。



百年同济  
TONGJI UNIVERSITY

## 3.8 浮点运算

**当运算的结果(和/差)不是规格化数时，需将它转变成规格化数。**

**规格化操作的规则是：**

**① 如果结果的两个符号位的值不同  
( $01.XX \cdots X$  ,  $10.XX \cdots X$ ) , 表示加/减运算尾数结果溢出，此时将尾数结果右移1位，阶码 $E+1$ ，称为“向右规格化”，简称“右规”。**



百年同济  
TONGJI UNIVERSITY

## 3.8 浮点运算

② 如果结果的两个符号位的值相同，表示加/减运算尾数结果不溢出。但若最高数值位与符号位相同，此时尾数连续左移，直到最高数值位与符号位的值不同为止；同时从E中减去移位的位数，这称之为“向左规格化”，简称“左规”。

### (4) 舍入

在执行右规或对阶时，尾数低位上的数值会移掉，使数值的精度受到影响，常用“0”舍“1”入法。





百年同济  
TONGJI UNIVERSITY

## 3.8 浮点运算

### (5) 检查阶码是否溢出

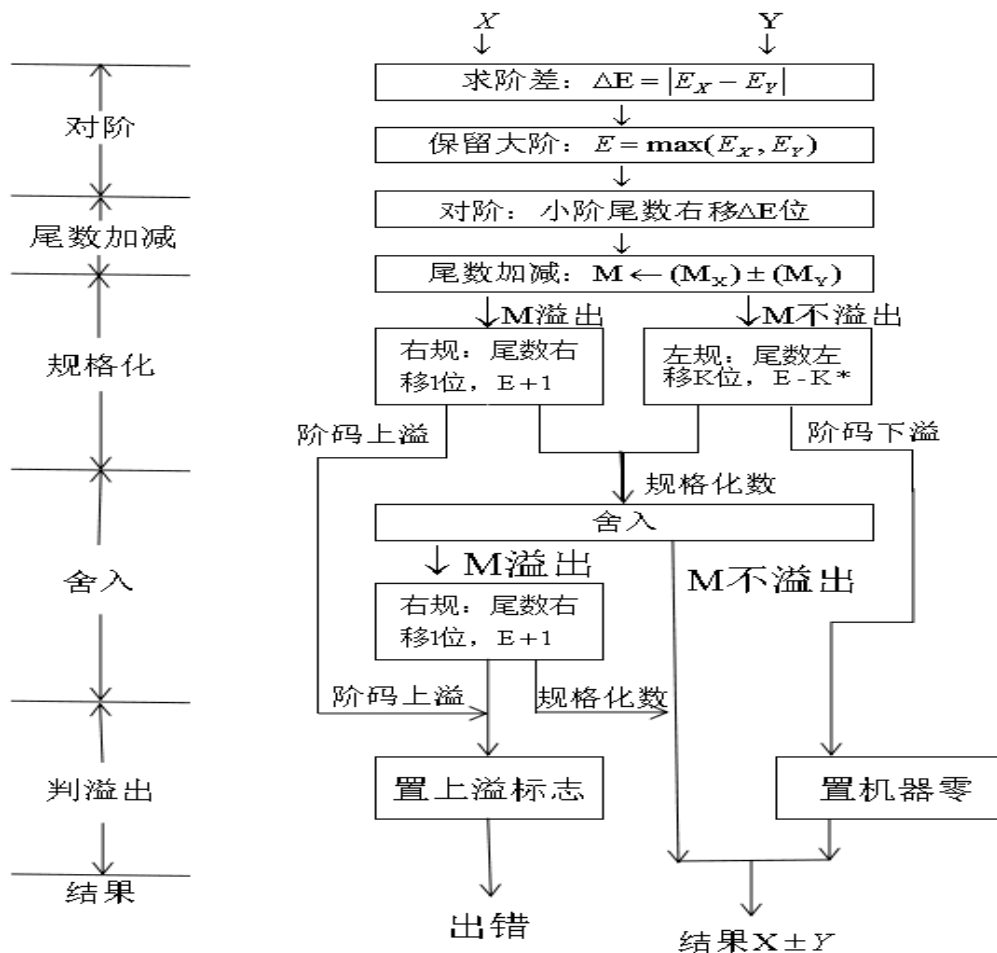
阶码溢出表示浮点数溢出。在规格化和舍入时都可能发生溢出，若阶码正常，加/减运算正常结束。若阶码下溢，则置运算结果为机器零，若上溢，则置溢出标志。

下图为规格化浮点数加减运算流程。



百年同济  
TONGJI UNIVERSITY

## 3.8 浮点运算





百年同济  
TONGJI UNIVERSITY

## 3.8 浮点运算

**例：两浮点数相加，求 $X+Y$ 。**

**已知： $X=0.11011011 \times 2^{010}$**

**$Y=-0.10101100 \times 2^{100}$**

**解：计算过程：**

**① 对阶操作**

**阶差 $\Delta E = [E_X]_{\text{补}} + [-E_Y]_{\text{补}}$**   
 **$= 00010 + 11100 = 11110$**

**$X$ 阶码小， $M_X$ 右移2位，保留阶码 $E=00100$ 。**

**$[M_X]_{\text{补}} = 00 \ 00 \ 110 \ 110 \ \underline{11}$**

**下划线上的数是右移出去而保留的附加位。**



百年同济  
TONGJI UNIVERSITY

## 3.8 浮点运算

### ② 尾数相加

$$\begin{aligned} & [M_x]_{\text{补}} + [M_y]_{\text{补}} \\ & = 0000110110\underline{11} + 1101010100 = 1110001010 \\ & \quad \underline{11} \end{aligned}$$

### ③ 规格化操作

左规，移1位，结果=1100010101 10；阶码减1，  
E=00011。

### ④ 舍入

附加位最高位为1，在所得结果的最低位+1，得新  
结果：  $[M]_{\text{补}} = 1100010110, M = -0.11101010$ 。



百年同濟  
TONGJI UNIVERSITY

## 3.8 浮点运算

### ⑤ 判溢出

阶码符号位为00，故不溢出，最终结果为：

$$X+Y=-0.11101010 \times 2^{011}$$



百年同济  
TONGJI UNIVERSITY

## 3.8 浮点运算

### 3.8.2 浮点数的乘除法运算

两浮点数相乘，其乘积的阶码为相乘两数阶码之和，其尾数应为相乘两数的尾数之积。两个浮点数相除，商的阶码为被除数的阶码减去除数的阶码得到的差，尾数为被除数的尾数除以除数的尾数所得的商。参加运算的两个数都为规格化浮点数。乘除运算都可能出现结果不满足规格化要求的问题，因此也必须进行规格化、舍入和判溢出等操作。规格化时要修改阶码。



百年同济  
TONGJI UNIVERSITY

## 3.8 浮点运算

### 1. 浮点数的阶码运算

阶码有+1, -1, 两阶码求和以及两阶码求差四种运算, 还要检查结果是否溢出。在计算机中, 阶码通常用补码或移码形式表示。补码运算规则和判定溢出的方法, 已在前面说明。这里讨论移码的运算规则和判定溢出的方法。

当阶码由1位符号位和n位数据组成时, 其移码的定义为:  $[X]_{\text{移}} = 2^n + X$   $-2^n \leq X < 2^n$

按此定义, 则有  $[X]_{\text{移}} + [Y]_{\text{移}}$   
 $= 2^n + X + 2^n + Y = 2^n + (2^n + (X + Y)) = 2^n + [X + Y]_{\text{移}}$



百年同济  
TONGJI UNIVERSITY

## 3.8 浮点运算

**即直接用移码实现求阶码之和时，结果的最高位多加了个1，要得到移码形式的结果，需对结果的符号取反。**

根据补码定义： $[Y]_{\text{补}} = 2^{n+1} + Y \bmod 2^{n+1}$

对同一个数值，移码和补码的数值位完全相同，而符号位正好相反。因此求阶码和(移码表示)可用如下方式完成：

$$\begin{aligned} [X]_{\text{移}} + [Y]_{\text{补}} &= 2^n + X + 2^{n+1} + Y \\ &= 2^{n+1} + (2^n + (X + Y)) = [X + Y]_{\text{移}} \bmod 2^{n+1} \end{aligned}$$

同理有  $[X]_{\text{移}} + [-Y]_{\text{补}} = [X - Y]_{\text{移}}$ 。





百年同济  
TONGJI UNIVERSITY

## 3.8 浮点运算

以上表明执行移码加或减时，取加数或减数符号位的反码进行运算。

如果阶码运算的结果溢出，上述条件则不成立。此时，使用双符号位的阶码加法器，并规定移码的第二个符号位，即最高符号位恒用0参加加减运算，则溢出条件是结果的最高符号位为1。此时，**当低位符号位为0时，表明结果上溢，为1时，表明结果下溢。**当最高符号位为0时，表明没有溢出，低位符号位为1，表明结果为正，为0时，表明结果为负。



百年同济  
TONGJI UNIVERSITY

## 3.8 浮点运算

### 2. 浮点数的舍入处理

在计算机中，浮点数的尾数是用确定的位数来表示的，但浮点数的运算结果却常常超过给定的位数。如加减运算过程中的对阶和右规处理，会使尾数低位部分的一位或多位的值丢失；乘除运算(无论是定点数或浮点数)也可有更多位数的结果，在这里讨论如何处理这些多出来的位上的值。处理的原则是**使本次处理所造成的误差以及按此原则产生的累计误差都比较小。**



百年同济  
TONGJI UNIVERSITY

## 3.8 浮点运算

- ① 无条件地丢掉正常尾数最低位之后的全部数值。这种办法被称为截断处理，其好处是处理简单，缺点是影响结果的精度。
- ② 运算过程中保留右移中移出的若干高位的值，然后再按某种规则用这些位上的值修正尾数。这种处理方法被称为舍入处理。较简单的舍入方法是：只要尾数最低位为1，或移出去的几位中有1，就把尾数的最低位置1，否则仍保持原有的0值。或者采用更简便的方法，即最低位恒置1的方法。



百年同济  
TONGJI UNIVERSITY

## 3.8 浮点运算

③ 0舍1入法(相当于十进制中的四舍五入法), 即当丢失的最高位的值为1时, 把这个1**加到**最低数值位上进行修正, 否则舍去丢失的各位的值, 其缺点是要多进行一次加法运算。下面举例说明0舍1入情况。



百年同济  
TONGJI UNIVERSITY

## 3.8 浮点运算

例：设有5位数(其中有二附加位)，用原码或补码表示，舍入后保留4位结果。

设：  $[X]_{\text{原}} = 0.1101\mathbf{10}$  舍入后  $[X]_{\text{原}} = 0.1110$

$[X]_{\text{原}} = 0.1110\mathbf{01}$  舍入后  $[X]_{\text{原}} = 0.1110$

$[X]_{\text{补}} = 1.0010\mathbf{10}$  舍入后  $[X]_{\text{补}} = 1.0011$

$[X]_{\text{补}} = 1.0010\mathbf{01}$  舍入后  $[X]_{\text{补}} = 1.0010$

舍入后产生了误差，但误差值小于末位的权值。



百年同济  
TONGJI UNIVERSITY

## 3.8 浮点运算

### 3. 浮点乘法运算步骤

- ① 检测操作数是否为0;
- ② 两数阶码相加, 求积的阶码;
- ③ 两数尾数相乘, 求积的尾数;
- ④ 尾数乘积规格化
- ⑤ 判断阶码有无溢出。



百年同济  
TONGJI UNIVERSITY

## 3.8 浮点运算

**例：阶码4位(移码)，尾数8位(补码，含1符号位)，阶码以2为底。运算结果仍取8位尾数。**

**设： $X=2^{-5} \times 0.1110011$ ,  $Y=2^3 \times (-0.1110010)$   
 $X$ ,  $Y$ 为真值，此处阶码用十进制表示，尾数用二进制表示。运算过程中阶码取双符号位。**



百年同济  
TONGJI UNIVERSITY

## 3.8 浮点运算

(1) 求乘积的阶码。乘积的阶码为两数阶码之和。

$$\begin{aligned} [E_X + E_Y]_{\text{移}} &= [E_X]_{\text{移}} + [E_Y]_{\text{补}} \\ &= 00011 + 00011 = 00110 \end{aligned}$$

(2) 尾数相乘。用定点数相乘的办法，

$$[X \times Y]_{\text{补}} = 1.0011001 \text{ } 1001010 \text{ (尾数部分)}$$

(3) 规格化处理。本例尾数已规格化，不需要再处理。如未规格化，需左规。





百年同济  
TONGJI UNIVERSITY

## 3.8 浮点运算

**(4) 舍入。尾数(乘积)低位部分的最高为1，需要舍入，在乘积高位部分的最低位加1，因此：**

$$[X \times Y]_{\text{补}} = 1.0011010 \text{ (尾数部分)}$$

**(5) 判溢出。阶码未溢出，故结果为正确。**

$$X \times Y = 2^{-2} \times (-0.1100110)$$

**在求乘积的阶码(即两阶码相加)时，有可能产生上溢或下溢的情况；在进行规格化处理时，有可能产生下溢。**



百年同济  
TONGJI UNIVERSITY

## 3.8 浮点运算

### 4. 浮点数乘法运算(阶码的底为8或16)

为了用相同位数的阶码表示更大范围的浮点数，在一些计算机中也有选用阶码的底为8或16的。此时浮点数N被表示成：

$$N = 8^E \cdot M \quad \text{或} \quad N = 16^E \cdot M$$

阶码E和尾数M还都是用二进制表示的，其运算规则，与阶码以2为底基本相同，但关于对阶和规格化操作有新的相应规定。



百年同济  
TONGJI UNIVERSITY

## 3.8 浮点运算

当阶码以8为底时，只要尾数满足 $1/8 \leq M < 1$ 或 $-1 \leq M < -1/8$ 就是规格化数。执行对阶和规格化操作时，**每当阶码的值增或减1，尾数要相应右移或左移三位。**

当阶码以16为底时，只要尾数满足 $1/16 \leq M < 1$ 或 $-1 \leq M < -1/16$ 就是规格化数。执行对阶和规格化操作时，**阶码的值增或减1，尾数必须移四位。**

判别为规格化数或实现规格化操作，均应使数值的最高三位(以8为底)或四位(以16为底)中至少有一位与符号位不同。



百年同济  
TONGJI UNIVERSITY

## 3.8 浮点运算

### 5. 浮点数除法运算步骤:

- ① 检测操作数是否为0;
- ② 尾数调整, 检测被除数的绝对值是否小于除数的尾数的绝对值, 如果不是, 将被除数的尾数右移一位, 并相应调整阶码;
- ③ 被除数阶码减除数阶码;
- ④ 被除数尾数除以除数尾数;
- ⑤ 尾数规格化。



百年同济  
TONGJI UNIVERSITY

## 3.9 BCD码运算

### 1. BCD码加法运算

用BCD码进行相加时，按二进制方法进行，但要对“和”进行加6修正。

修正条件：

- ① 某位和大于9时，加6修正。
- ② 低位向高位产生进位时，加6修正。



百年同济  
TONGJI UNIVERSITY

## 3.9 BCD码运算

### 2. BCD码减法运算

用BCD码进行相减时，按二进制方法进行，但要对“差”进行减6修正。

修正条件：

低位向高位产生借位时，减6修正。



百年同濟  
TONGJI UNIVERSITY





百年同济  
TONGJI UNIVERSITY

## 3.10 运算部件组织

### 1. 定点运算部件

定点运算部件由算术逻辑运算部件ALU、若干个寄存器、移位电路、计数器、门电路等组成。ALU部件主要完成加减法算术运算及逻辑运算，其中还应包含有快速进位电路。下图为定点运算部件的框图，图中仅有三个寄存器(A, B, C)，而目前一般的运算部件都设置有数量较多的寄存器，可任意放置操作数和运算结果等，称之为通用寄存器。







百年同济  
TONGJI UNIVERSITY

## 3.10 运算部件组织

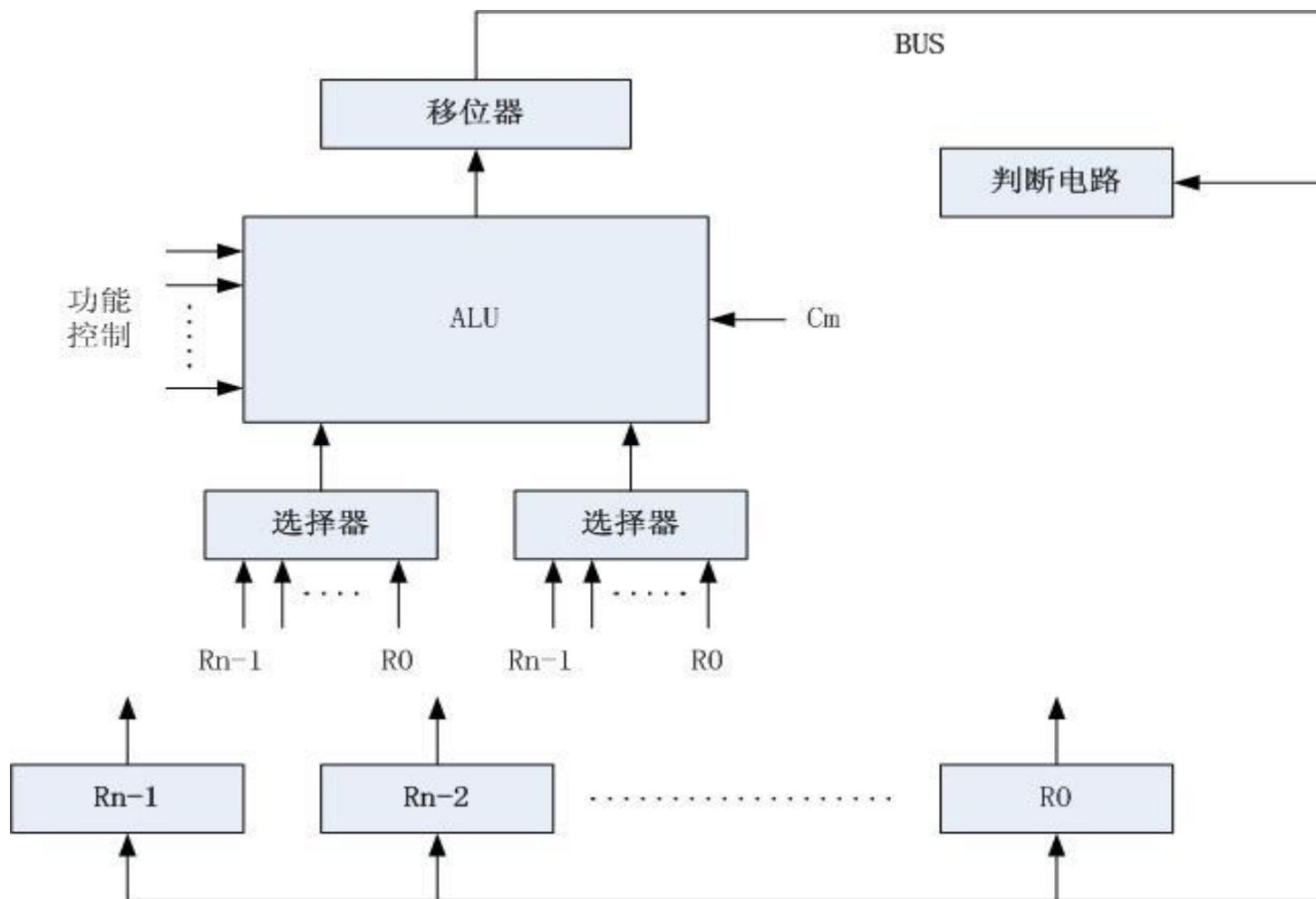
### A, B, C寄存器的作用

运算	A	B	C
加法	被加数 运算结果	加数	无用
减法	被减数 运算结果	减数	无用
乘法	部分积 乘积高位	被乘数	乘数, 乘积低位
除法	被除数 余数	除数	商



百年同济  
TONGJI UNIVERSITY

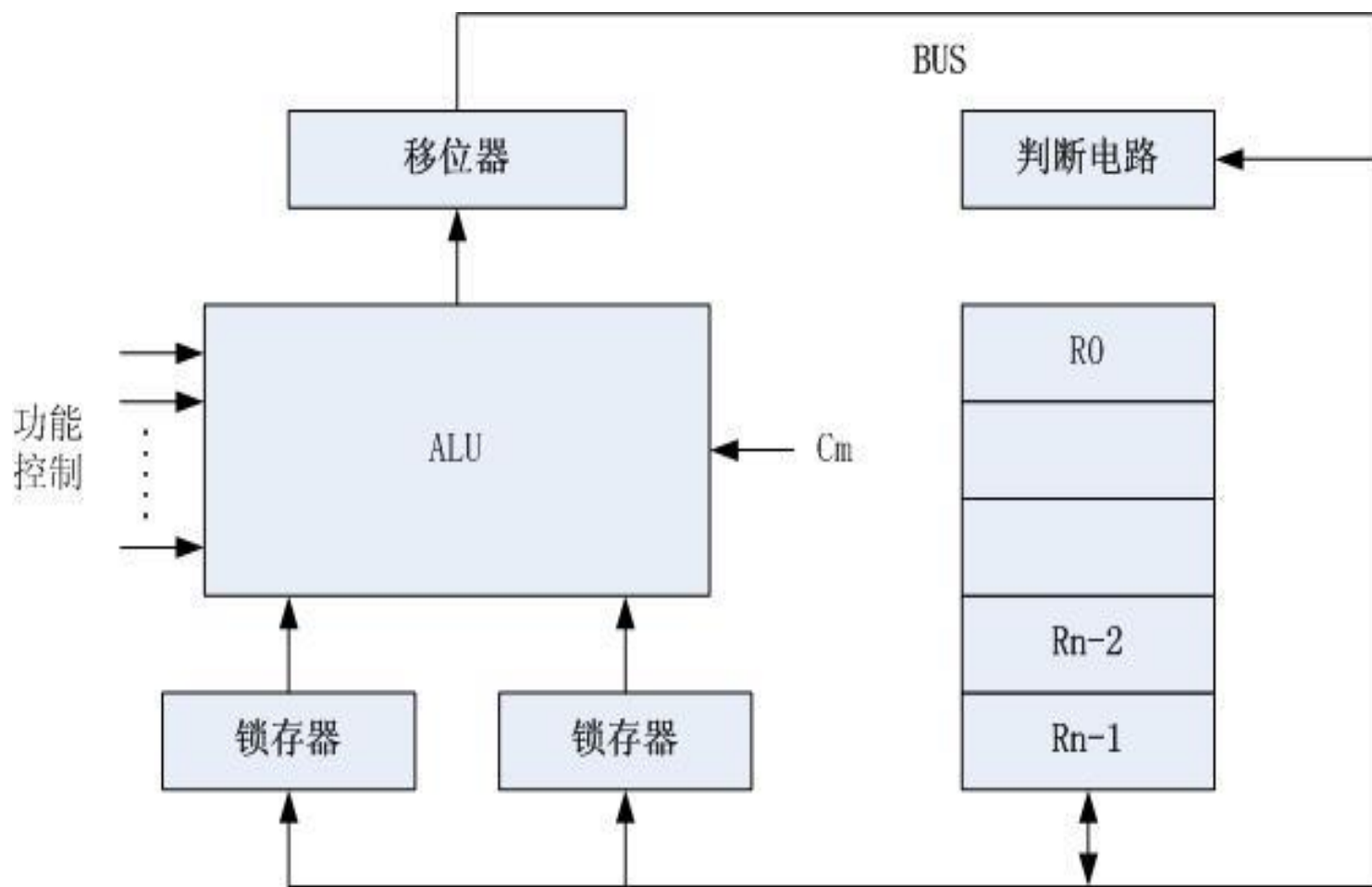
## 3.10 运算部件组织





百年同济  
TONGJI UNIVERSITY

## 3.10 运算部件组织





百年同济  
TONGJI UNIVERSITY

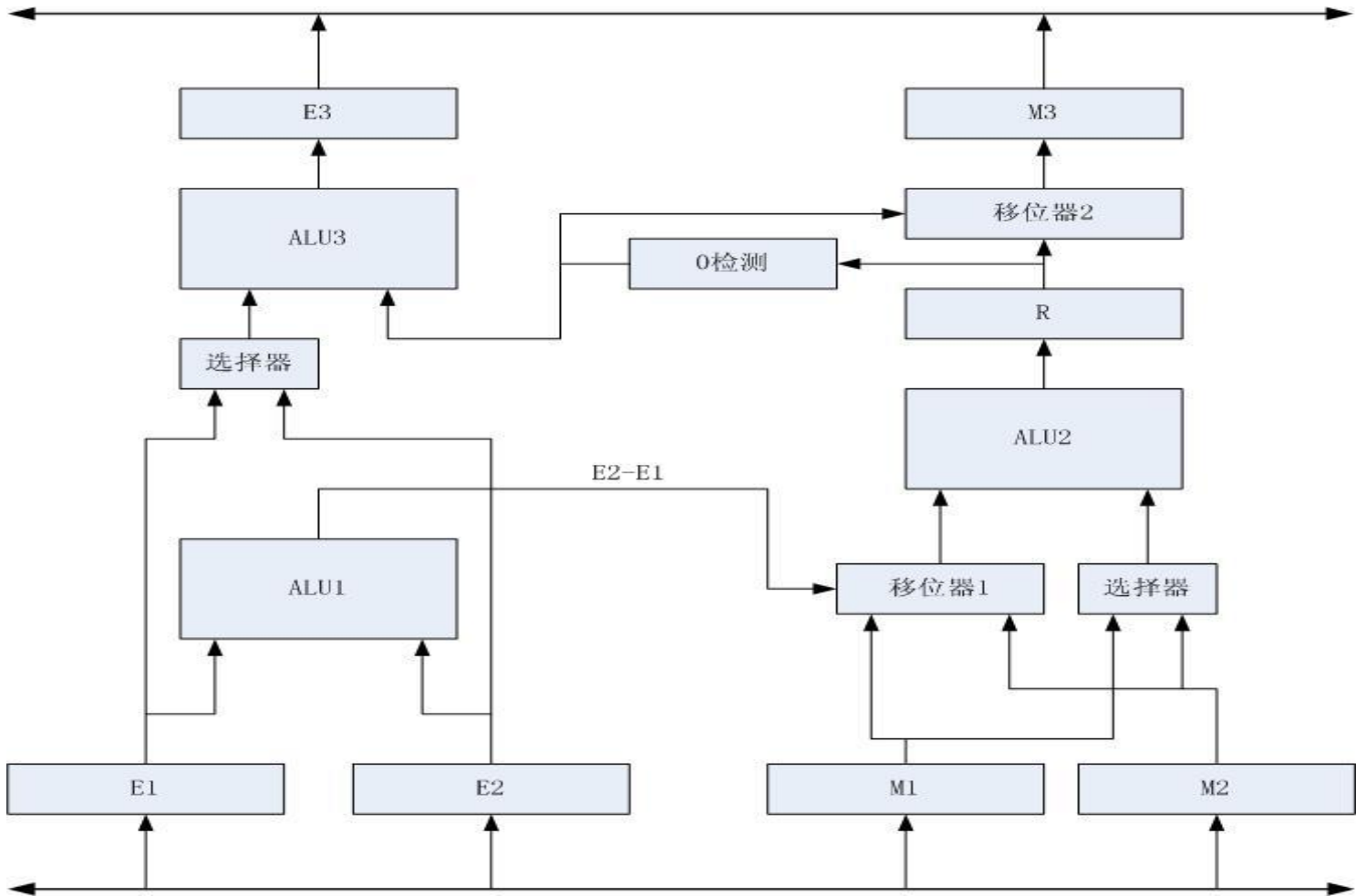
## 3.10 运算部件组织

### 2. 浮点运算部件

通常由阶码运算部件和尾数运算部件组成，其各自的结构与定点运算部件相似。但阶码部分仅执行加减法运算。其尾数部分则执行加减乘除运算，左规时有时需要左移多位。为加速移位过程，有的机器设置了可移动多位的电路（桶形移位器）。



### 3.8 浮点运算





百年同濟  
TONGJI UNIVERSITY

# 习 题

---

**P67**

**习题: 16, 17, 18, 21, 22, 23, 24**