

第4章 计算机组成原理

# 数值的机器运算

北京理工大学计算机学院

2025年10月

第4章 计算机组成原理

运算器是计算机进行算术运算和逻辑运算的主要部件，运算器的逻辑结构取决于机器的指令系统、数据表示方法和运算方法等。本章主要讨论数值数据在计算机中实现算术运算和逻辑运算的方法，以及运算部件的基本结构和工作原理。

北京理工大学计算机学院

2025年10月

第4章 计算机组成原理

### 4.1 基本算术运算的实现

- 4.2 定点加减运算
- 4.3 带符号数的移位和舍入操作
- 4.4 定点乘法运算
- 4.5 定点除法运算
- 4.6 规格化浮点运算
- 4.7 十进制整数的加减运算
- 4.8 逻辑运算与实现
- 4.9 运算器的基本组成与实例

北京理工大学计算机学院

2025年10月

第4章 计算机组成原理

### 4.1.1 加法器

加法器是由全加器再配以其他必要的逻辑电路组成的。

#### 1. 全加器

基本的加法单元称为全加器，它要求三个输入量：操作数  $A_i$  和  $B_i$ 、低位传来的进位  $C_{i-1}$ ，并产生两个输出量：本位和  $S_i$ 、向高位的进位  $C_i$ 。

北京理工大学计算机学院

2025年10月

第4章 计算机组成原理

### 4.1 基本算术运算的实现

全加器的逻辑表达式为

$$S_i = A_i \oplus B_i \oplus C_{i-1}$$

$$C_i = A_i B_i + (A_i \oplus B_i) C_{i-1}$$

#### 2. 串行加法器与并行加法器

在串行加法器中，只有一个全加器，数据逐位串行送入加法器进行运算。

如果操作数长  $n$  位，加法就要分  $n$  次进行，每次只能产生一位和。

北京理工大学计算机学院

2025年10月

第4章 计算机组成原理

并行加法器由多个全加器组成，其位数的多少取决于机器的字长，数据的各位同时运算。

并行加法器虽然操作数的各位是同时提供的，但低位运算所产生的进位有可能会影响高位的运算结果。例如：11...11和00...01相加，最低位产生的进位将逐位影响至最高位。因此，并行加法器的最长运算时间主要是由进位信号的传递时间决定的。**提高并行加法器速度的关键是尽量加快进位产生和传递的速度。**

北京理工大学计算机学院

2025年10月

**4.1 基本算术运算**

**4.1.2 进位**

进位产生用 $G_i$ 表示  
进位传递函数用 $P_i$ 表示

进位表达式  
 $C_i = A_i B_i + (A_i \oplus B_i) C_{i-1}$

$G_i$ 的含义是：若本位的两个输入均为1，必然要向高位产生进位。

$P_i$ 的含义是：当两个输入中有一个为1，低位传来的进位 $C_{i-1}$ 将超越本位向更高的位传送。

$\therefore C_i = G_i + P_i C_{i-1}$

北京理工大学计算机学院

2025年10月

**4.1 基本算术运算的实现**

**4.1.2 进位**

把n个全加器串接起来，就可进行两个n位数的相加。串行进位又称行波进位，每一级进位直接依赖于前一级的进位，即进位信号是逐级形成的。

$C_1 = G_1 + P_1 C_0$   
 $C_2 = G_2 + P_2 C_1$   
 $\vdots$   
 $C_n = G_n + P_n C_{n-1}$

北京理工大学计算机学院

2025年10月

**4.1 基本算术运算的实现**

**4.1.3 并行加法器的快速进位**

1. 并行进位方式

并行进位又叫先行进位、同时进位，其特点是各级进位信号同时形成。

$C_1 = G_1 + P_1 C_0$   
 $C_2 = G_2 + P_2 C_1 = G_2 + P_2 G_1 + P_2 P_1 C_0$   
 $C_3 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 C_0$   
 $C_4 = G_4 + P_4 G_3 + P_4 P_3 G_2 + P_4 P_3 P_2 G_1 + P_4 P_3 P_2 P_1 C_0$

北京理工大学计算机学院

2025年10月

**4.1 基本算术运算的实现**

**4.1.3 并行加法器的快速进位**

2. 分组并行进位方式

实际上，通常采用分组并行进位方式。这种进位方式是把n位字长分为若干小组，在组内各位之间实行并行快速进位，在组间既可以采用串行进位方式，也可以采用并行快速进位方式，因此有两种情况。

注意

北京理工大学计算机学院

2025年10月

**4.1 基本算术运算的实现**

**4.1.3 并行加法器的快速进位**

上述各式中所有各位的进位均不依赖于低位的进位，各位的进位可以同时产生。这种进位方式是快速的，若不考虑 $G_i$ 、 $P_i$ 的形成时间，从 $C_0 \rightarrow C_n$ 的最长延迟时间为 $2ty$ 。随着加法器位数的增加， $C_i$ 的逻辑表达式会变得越来越长，所以，完全采用并行进位是不现实的。

北京理工大学计算机学院

2025年10月

**4.1 基本算术运算的实现**

**4.1.3 并行加法器的快速进位**

2. 分组并行进位方式

实际上，通常采用分组并行进位方式。这种进位方式是把n位字长分为若干小组，在组内各位之间实行并行快速进位，在组间既可以采用串行进位方式，也可以采用并行快速进位方式，因此有两种情况。

注意

北京理工大学计算机学院

2025年10月

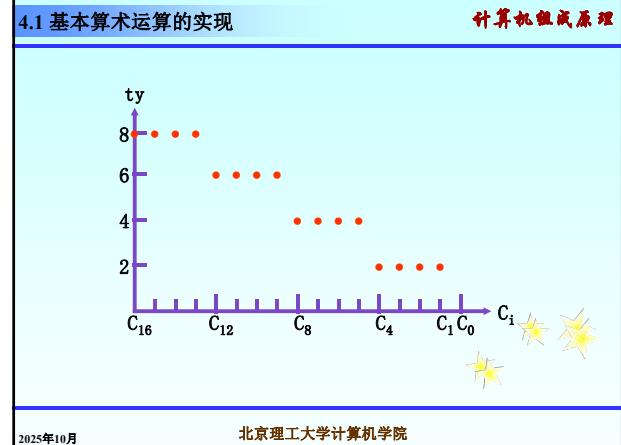
4.1 基本算术运算的实现 计算机组成原理

(1) 单级先行进位方式

这种进位方式又称为组内并行、组间串行方式。以16位加法器为例，可分为四组，每组四位。第1小组组内的进位逻辑函数 $C_1 \sim C_4$ 、 $C_2 \sim C_3$ 、 $C_3 \sim C_4$ 的表达式与前述相同， $C_1 \sim C_4$ 信号是同时产生的，从 $C_0$ 出现到产生 $C_1 \sim C_4$ 的延迟时间是 $2ty$ 。

北京理工大学计算机学院

2025年10月



4.1 基本算术运算的实现 计算机组成原理

(2) 多级先行进位方式

多级先行进位又称组内并行、组间并行

**组进位** 产生函数 $G_i^*$ ，6位的两级传递函数 $P_i^*$ ，去器，第一小组的最高位进位 $C_4$ ：

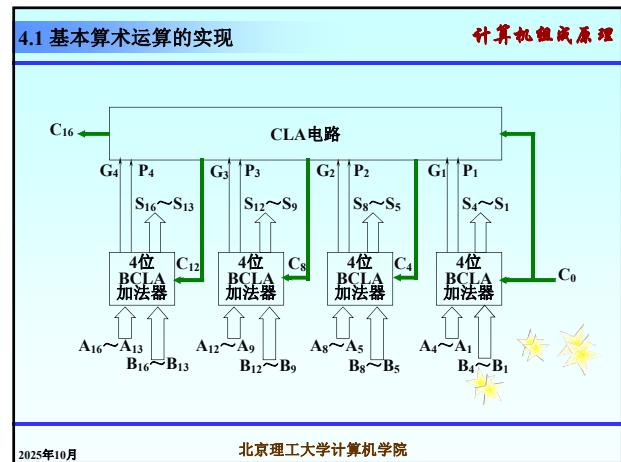
$C_4 = G_4^* + P_4 G_3^* + P_4 P_3 G_2^* + P_4 P_3 P_2 G_1^* + P_4 P_3 P_2 P_1 G_0^* = G_1^* + P_1^* C_0$

依次类推：

$C_8 = G_2^* + P_2^* G_1^* + P_2^* P_1^* C_0$   
 $C_{12} = G_3^* + P_3^* G_2^* + P_3^* P_2^* G_1^* + P_3^* P_2^* P_1^* C_0$   
 $C_{16} = G_4^* + P_4^* G_3^* + P_4^* P_3^* G_2^* + P_4^* P_3^* P_2^* G_1^* + P_4^* P_3^* P_2^* P_1^* C_0$

北京理工大学计算机学院

2025年10月

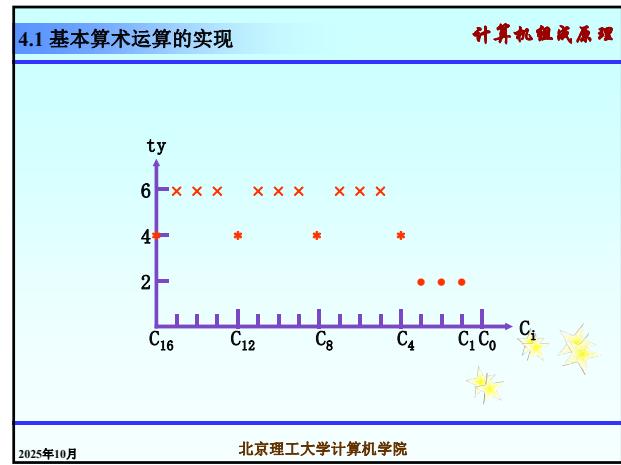


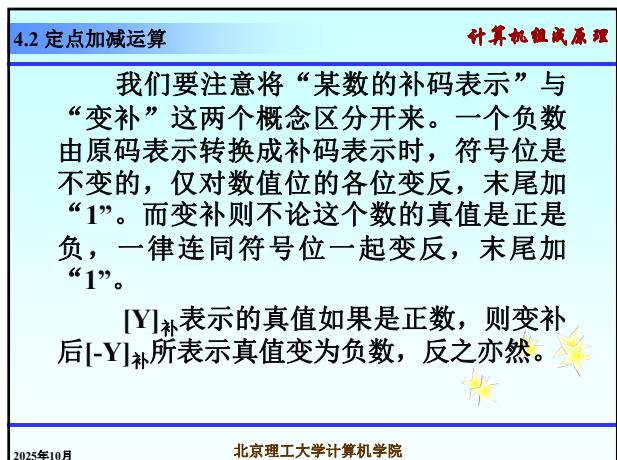
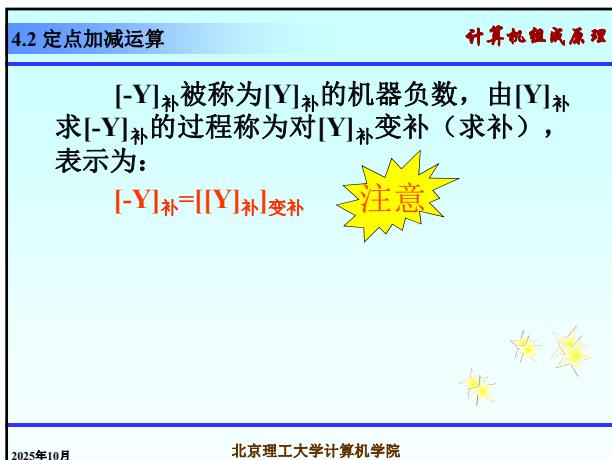
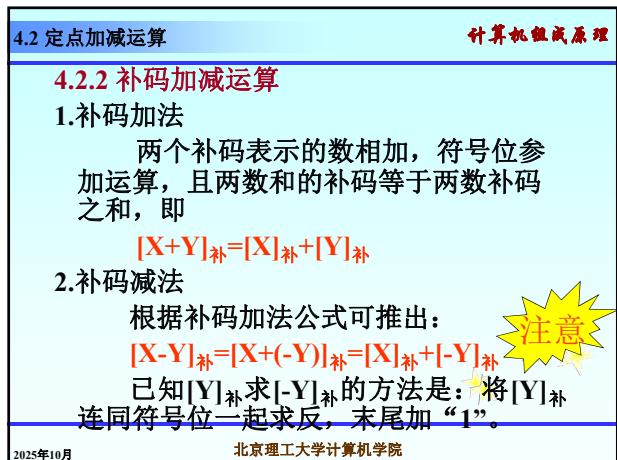
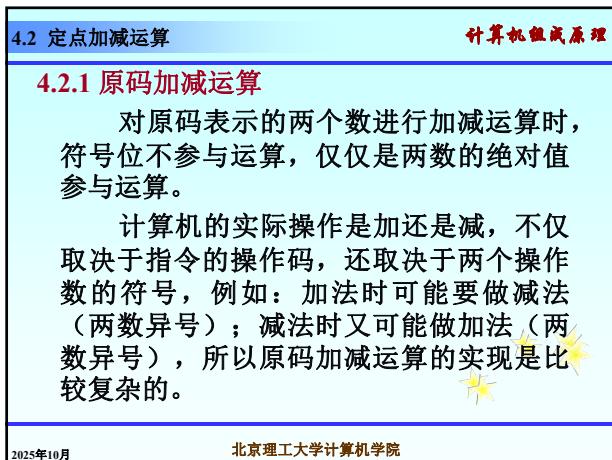
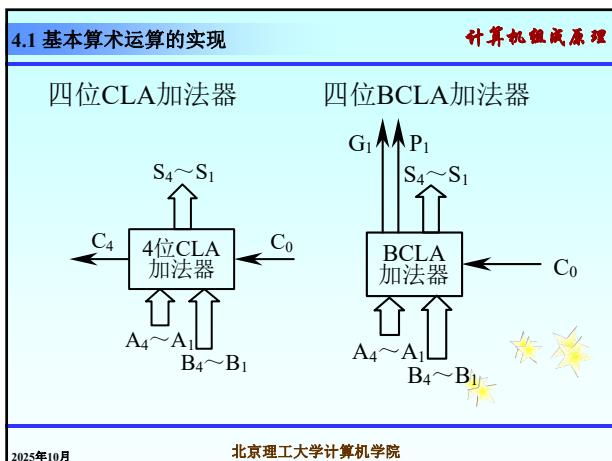
4.1 基本算术运算的实现 计算机组成原理

若不考虑 $G_i$ 、 $P_i$ 的形成时间， $C_0$ 经过 $2ty$ 产生第1小组的 $C_1$ 、 $C_2$ 、 $C_3$ 及所有组进位产生函数 $G_i^*$ 和组进位传递函数 $P_i^*$ ；再经过 $2ty$ ，产生 $C_4$ 、 $C_8$ 、 $C_{12}$ 、 $C_{16}$ ；最后经过 $2ty$ 后，才能产生第2、3、4小组内的 $C_5 \sim C_7$ 、 $C_9 \sim C_{11}$ 、 $C_{13} \sim C_{15}$ 。

北京理工大学计算机学院

2025年10月





4.2 定点加减运算 计算机组成原理

**例1:**  $Y = -0.0110$   
 $[Y]_{\text{补}} = 1.1010, [-Y]_{\text{补}} = 0.0110$

**例2:**  $Y = 0.0110$   
 $[Y]_{\text{补}} = 0.0110, [-Y]_{\text{补}} = 1.1010$



2025年10月 北京理工大学计算机学院

4.2 定点加减运算 计算机组成原理

**3. 补码加减运算规则**

补码加减运算规则如下：

- (1) 参加运算的两个操作数均用补码表示；
- (2) 符号位作为数的一部分参加运算；
- (3) 若做加法，则两数直接相加；若做减法，则将被减数与减数的机器负数相加；
- (4) 运算结果用补码表示。



2025年10月 北京理工大学计算机学院

4.2 定点加减运算 计算机组成原理

**例1:**  $A = 0.1011, B = -0.1110$ , 求:  $A + B$   
 $\because [A]_{\text{补}} = 0.1011, [B]_{\text{补}} = 1.0010$

|          |
|----------|
| 0.1011   |
| + 1.0010 |
| <hr/>    |
| 1.1101   |

$\therefore [A+B]_{\text{补}} = 1.1101, A + B = \textcolor{red}{-0.0011}$



2025年10月 北京理工大学计算机学院

4.2 定点加减运算 计算机组成原理

**例2:**  $A = 0.1011, B = -0.0010$ , 求:  $A - B$   
 $\because [A]_{\text{补}} = 0.1011, [B]_{\text{补}} = 1.1110, [-B]_{\text{补}} = 0.0010$

|          |
|----------|
| 0.1011   |
| + 0.0010 |
| <hr/>    |
| 0.1101   |

$\therefore [A-B]_{\text{补}} = 0.1101, A - B = \textcolor{red}{0.1101}$



2025年10月 北京理工大学计算机学院

4.2 定点加减运算 计算机组成原理

**4. 符号扩展**

在计算机算术运算中，有时必须将采用给定位数表示的数转换成具有更多位数的某种表示形式。例如某个程序需要将一个8位数与另外一个32位数相加。要想得到正确的结果，在将8位数与32位数相加之前，必须将8位数转换成32位数形式，这被称为“符号扩展”。

对于补码，符号扩展方法是：原有符号位保持不变，若为正数则所有附加位都用0进行填充，若为负数则所有附加位都用1进行填充。也可以理解为是用符号位来填充附加的高位。

2025年10月 北京理工大学计算机学院

4.2 定点加减运算 计算机组成原理

**例：**将用8位二进制补码表示的十进制数-121，扩展成16位二进制补码，结果用十六进制表示。

解：十进制数-121的8位二进制补码表示为10000111，扩展成16位二进制补码，符号扩展，表示为111111110000111= FF87H。



2025年10月 北京理工大学计算机学院

4.2 定点加减运算 计算机组成原理

### 4.2.3 补码的溢出判断与检测方法

#### 1. 溢出的产生

在补码加减运算中，有时会遇到这样的情况：两个正数相加，而结果的符号位却为1（结果为负）；两个负数相加，而结果的符号位却为0（结果为正）。

例1：X=1011B=11D, Y=111B=7D  
 $[X]_{\text{补}}=0,1011, [Y]_{\text{补}}=0,0111$   

$$\begin{array}{r} 0,1011 \\ + 0,0111 \\ \hline 1,0010 \end{array}$$

北京理工大学计算机学院

2025年10月

4.2 定点加减运算 计算机组成原理

### [X+Y]<sub>补</sub>=1,0010, X+Y=-1110B=-14D

两正数相加结果为-14D，显然是错误的。

例2：X=-1011B=-11D, Y=-111B=-7D  
 $[X]_{\text{补}}=1,0101, [Y]_{\text{补}}=1,1001$   

$$\begin{array}{r} 1,0101 \\ + 1,1001 \\ \hline 0,1110 \end{array}$$

两负数相加结果为14D，显然也是错误的。

北京理工大学计算机学院

2025年10月

4.2 定点加减运算 计算机组成原理

为什么会发生这种错误呢？原因在于两数相加之和的数值已超过了机器允许的表示范围。

字长为n+1位的定点整数（其中一位为符号位），采用补码表示，当运算结果大于 $2^n-1$ 或小于 $-2^n$ 时，就产生溢出。

北京理工大学计算机学院

2025年10月

4.2 定点加减运算 计算机组成原理

设参加运算的两数为X、Y，做加法运算。

若X、Y异号，不会溢出。

若X、Y同号，运算结果为正且大于所能表示的最大正数或运算结果为负且小于所能表示的最小负数（绝对值最大的负数）时，产生溢出。将两正数相加产生的溢出称为正溢；反之，两负数相加产生的溢出称为负溢。

北京理工大学计算机学院

2025年10月

4.2 定点加减运算 计算机组成原理

#### 2. 溢出检测方法

设：被操作数为： $[X]_{\text{补}}=X_s, X_1, X_2, \dots, X_n$   
 操作数为： $[Y]_{\text{补}}=Y_s, Y_1, Y_2, \dots, Y_n$   
 其和（差）为： $[S]_{\text{补}}=S_s, S_1, S_2, \dots, S_n$

(1) 采用一个符号位

两正数相加，结果为负表明产生正溢；  
 两负数相加，结果为正表明产生负溢。因此可得出采用一个符号位检测溢出的方法：

当 $X_s=Y_s=0, S_s=1$ 时，产生正溢。  
 当 $X_s=Y_s=1, S_s=0$ 时，产生负溢。  
 溢出 =  $\bar{X}_s \bar{Y}_s S_s + X_s Y_s \bar{S}_s$

北京理工大学计算机学院

2025年10月

4.2 定点加减运算 计算机组成原理

#### (2) 采用进位位

两数运算时，产生的进位为 $C_s, C_1, C_2, \dots, C_n$ ，其中： $C_s$ 为符号位产生的进位， $C_1$ 为最高数值位产生的进位。

两正数相加，当最高有效位产生进位（ $C_1=1$ ）而符号位不产生进位（ $C_s=0$ ）时，发生正溢。

两负数相加，当最高有效位没有进位（ $C_1=0$ ）而符号位产生进位（ $C_s=1$ ）时，发生负溢。

北京理工大学计算机学院

2025年10月

4.2 定点加减运算      计算机组成原理

$$\begin{array}{r}
 & 0,1\ 0\ 1\ 1 \\
 + & 0,0\ 1\ 1\ 1 \\
 \hline
 & 1,0\ 0\ 1\ 0
 \end{array}$$
  

$$\begin{array}{r}
 & 1,0\ 1\ 0\ 1 \\
 + & 1,1\ 0\ 0\ 1 \\
 \hline
 & 0,1\ 1\ 1\ 0
 \end{array}$$

溢出 =  $\bar{C}_s C_1 + C_s \bar{C}_1 = C_s \oplus C_1$

北京理工大学计算机学院

4.2 定点加减运算      计算机组成原理

当两位符号位的值不一致时，表明产生溢出。

溢出 =  $S_{s1} \oplus S_{s2}$

**注意**

北京理工大学计算机学院

4.2 定点加减运算      计算机组成原理

(3)采用变形补码(双符号位补码)

在双符号位的情况下，把左边的符号位  $S_{s1}$  叫做真符，因为它代表了该数真正的符号，两个符号位都作为数的一部分参加运算。这种编码又称为变形补码。

双符号位的含义如下：

- $S_{s1} S_{s2} = 00$  结果为正数，无溢出
- $S_{s1} S_{s2} = 01$  结果正溢
- $S_{s1} S_{s2} = 10$  结果负溢
- $S_{s1} S_{s2} = 11$  结果为负数，无溢出

北京理工大学计算机学院

4.2 定点加减运算      计算机组成原理

前例中字长为5位，数的表示范围为-16~15，采用变形补码(双符号位)运算，则有：

$$\begin{array}{r}
 11+7=18 \text{ (正溢)} \\
 0,0,1\ 0\ 1\ 1 \\
 + 0,0,0\ 1\ 1\ 1 \\
 \hline
 0,1,0\ 0\ 1\ 0
 \end{array}$$

$$\begin{array}{r}
 -11+(-7)=-18 \text{ (负溢)} \\
 1,1,0\ 1\ 0\ 1 \\
 + 1,1,1\ 0\ 0\ 1 \\
 \hline
 1,0,1\ 1\ 1\ 0
 \end{array}$$

北京理工大学计算机学院

4.2 定点加减运算      计算机组成原理

#### 4.2.4 补码定点加减运算的实现

补码加法： $X \rightarrow F$ 、 $Y \rightarrow F$ 、 $F \rightarrow X$ 、 $CP_X$

北京理工大学计算机学院

4.2 定点加减运算      计算机组成原理

补码减法： $X \rightarrow F$ 、 $\bar{Y} \rightarrow F$ 、 $1 \rightarrow F$ 、 $F \rightarrow X$ 、 $CP_X$

北京理工大学计算机学院

**第4章** 计算机组成原理

**4.1 基本算术运算的实现**  
**4.2 定点加减运算**  
**4.3 带符号数的移位和舍入操作**  
**4.4 定点乘法运算**  
**4.5 定点除法运算**  
**4.6 规格化浮点运算**  
**4.7 十进制整数的加减运算**  
**4.8 逻辑运算与实现**  
**4.9 运算器的基本组成与实例**

北京理工大学计算机学院

2025年10月

**4.3 带符号数的移位和舍入操作** 计算机组成原理

**4.3.1 带符号数的移位操作**

算术移位应保持数的符号不变，而数值的大小则要发生变化。左移一位使数值增大一倍，相当于该数乘以2，而右移一位则使数值缩小一倍，相当于该数除以2。

1. 原码的移位规则

负数的原码移位后的空出位补0

|    |   |       |       |     |       |   |
|----|---|-------|-------|-----|-------|---|
| 左移 | 1 | $X_2$ | $X_3$ | ... | $X_n$ | 0 |
|----|---|-------|-------|-----|-------|---|

右移 

|   |   |       |     |           |           |
|---|---|-------|-----|-----------|-----------|
| 1 | 0 | $X_1$ | ... | $X_{n-2}$ | $X_{n-1}$ |
|---|---|-------|-----|-----------|-----------|

2025年10月 北京理工大学计算机学院

**4.3 带符号数的移位和舍入操作** 计算机组成原理

**2. 补码的移位规则**

负数的补码左移后的空出位补0，右移后的空出位补1。

|    |   |       |       |     |       |   |
|----|---|-------|-------|-----|-------|---|
| 左移 | 1 | $X_2$ | $X_3$ | ... | $X_n$ | 0 |
|----|---|-------|-------|-----|-------|---|

|    |   |   |       |     |           |           |
|----|---|---|-------|-----|-----------|-----------|
| 右移 | 1 | 1 | $X_1$ | ... | $X_{n-2}$ | $X_{n-1}$ |
|----|---|---|-------|-----|-----------|-----------|

北京理工大学计算机学院

2025年10月

**4.3 带符号数的移位和舍入操作** 计算机组成原理

**3. 移位功能的实现**

通常移位操作由移位寄存器来实现。但也有一些计算机不设置专门的移位寄存器，而在加法器的输出端加一个实现直传、左移一位和右移一位的控制逻辑电路（称为移位器）。

分别用 $2F \rightarrow L$ 、 $F \rightarrow L$ 和 $F/2 \rightarrow L$ 这三个不同控制信号选择左移、直传和右移操作。

2025年10月 北京理工大学计算机学院

**4.3 带符号数的移位和舍入操作** 计算机组成原理

**左移**

$F/2 \rightarrow L$   $F \rightarrow L$   $2F \rightarrow L$

北京理工大学计算机学院

2025年10月

**4.3 带符号数的移位和舍入操作** 计算机组成原理

**4.3.2 带符号数的舍入操作**

在算术右移中，由于受硬件的限制，运算结果有可能需要舍去一定的尾数，会造成一些误差。为了缩小误差，就要进行舍入处理。

1. 恒舍（切断）

这是一种最容易实现的舍入方法，无论多余部分q位为何代码，一律舍去，保留部分p位不作任何改变。

保留部分p位  
多余部分q位

2025年10月 北京理工大学计算机学院

**4.3 带符号数的移位和舍入操作**      **计算机组成原理**

### 2. 冯·诺依曼舍入法

这种舍入法又称为恒置1法，即不论多余部分q位为何代码，都把p位的最低位置1。

保留部分p位多余部分q位  
保留部分最低位为1

保留部分p位多余部分q位  
保留部分最低位为0

2025年10月      北京理工大学计算机学院

**4.3 带符号数的移位和舍入操作**      **计算机组成原理**

### 3. 下舍上入法

下舍上入就是0舍1入。用将要舍去的q位部分的最高位作为判断标志，如该位为0，则舍去整个q位部分，如该位为1，则在前面的p位部分的最低位上加1。

保留部分p位多余部分q位  
多余部分最高位为0

保留部分p位多余部分q位  
多余部分最高位为1

2025年10月      北京理工大学计算机学院

**4.3 带符号数的移位和舍入操作**      **计算机组成原理**

### 4.查表舍入法

用ROM存放下溢处理表，每次经查表来读得相应的处理结果。ROM表的容量为 $2^K$ 个单元，每个单元字长为K-1位。下溢处理表的内容设置一般采用的方法是：当K位数据的高K-1位为全“1”时，让那些单元按截断法填入K-1位全“1”，其余单元都按最低位（即附加位）0舍1入的结果来填其内容。

| 地址  | 内容 |
|-----|----|
| 000 | 00 |
| 001 | 01 |
| 010 | 01 |
| 011 | 10 |
| 100 | 10 |
| 101 | 11 |
| 110 | 11 |
| 111 | 11 |

2025年10月      北京理工大学计算机学院

**第4章**      **计算机组成原理**

- 4.1 基本算术运算的实现
- 4.2 定点加减运算
- 4.3 带符号数的移位和舍入操作
- 4.4 定点乘法运算**
- 4.5 定点除法运算
- 4.6 规格化浮点运算
- 4.7 十进制整数的加减运算
- 4.8 逻辑运算与实现
- 4.9 运算器的基本组成与实例

2025年10月      北京理工大学计算机学院

**4.4 定点乘法运算**      **计算机组成原理**

### 4.4.1 原码一位乘法

用原码实现乘法运算是十分方便的。原码一位乘法是从手算演变而来的，即用两个操作数的绝对值相乘，乘积的符号为两操作数符号的异或值（同号为正，异号为负）。

乘积  $P = |X| \times |Y|$   
符号  $P_s = X_s \oplus Y_s$   
例如： $X=0.1101$ ,  $Y=-0.1011$ , 列出手算乘法算式为

2025年10月      北京理工大学计算机学院

**4.4 定点乘法运算**      **计算机组成原理**

$$\begin{array}{r}
 0.1\ 1\ 0\ 1 \\
 \times 0.1\ 0\ 1\ 1 \\
 \hline
 & 1\ 1\ 0\ 1 \\
 & 1\ 1\ 0\ 1 \\
 & 0\ 0\ 0\ 0 \\
 + & 1\ 1\ 0\ 1 \\
 \hline
 0.1\ 0\ 0\ 0\ 1\ 1\ 1\ 1
 \end{array}$$

因为  $P_s = X_s \oplus Y_s = 0 \oplus 1 = 1$   
所以  $X \times Y = -0.10001111$

2025年10月      北京理工大学计算机学院

4.4 定点乘法运算

计算机组成原理

原码一位乘法的规则为：

- ① 参加运算的操作数取其绝对值；
- ② 令乘数的最低位为判断位，若为“1”，加被乘数，若为“0”，不加被乘数（加0）；
- ③ 累加后的部分积右移一位；
- ④ 重复n次②和③；
- ⑤ 符号位单独处理，同号为正，异号为负。



4.4 定点乘法运算

计算机组成原理

已知:  $X=0.1101$ ,  $Y=-0.1011$ , 求:  $X \times Y$ 。

$|X|=0.1101 \rightarrow B$ ,  $|Y|=0.1011 \rightarrow C$ ,  $0 \rightarrow A$

符号位不参加运算

A、C寄存器级联, 右移。

Three yellow star icons arranged in a triangular pattern in the bottom right corner of the slide.

2025年10月

北京理工大学计算机学院

| 4.4 定点乘法运算   |                     | 计算机组成原理       |
|--|---------------------|---------------|
| A<br><u>0 0 0 0 0 0</u>                            | C<br><u>1 0 1 1</u> | 说明            |
| + X <br>0 0 0 1 1 0 1                              |                     | $C_4=1, + X $ |
| —————<br>0 0 1 1 0 0 1                             |                     | 部分积右移一位       |
| → 0 0 0 1 1 0 0                                    | 1 1 0 1             | $C_4=1, + X $ |
| + X <br>0 0 1 1 0 1                                |                     | 部分积右移一位       |
| —————<br>0 1 0 0 1 1                               |                     | $C_4=0, +0$   |
| → 0 0 1 0 0 1                                      | 1 1 1 0             | 部分积右移一位       |
| +0<br>0 0 0 0 0 0                                  |                     | $C_4=1, + X $ |
| —————<br>0 0 1 0 0 1                               |                     | 部分积右移一位       |
| → 0 0 0 1 0 0                                      | 1 1 1 1             | $C_4=1, + X $ |
| + X <br>0 0 1 1 0 1                                |                     | 部分积右移一位       |
| —————<br>0 1 0 0 0 1                               |                     | $C_4=0, +0$   |
| → 0 0 1 0 0 0                                      | 1 1 1 1             | 部分积右移一位       |
| $\therefore P_s = X_s \oplus Y_s = 0 \oplus 1 = 1$ |                     | *             |
| $\therefore X \times Y = -0.10001111$              |                     | *             |

4.4 定点乘法运算

计算机组成原理

```

graph TD
    Start["|X|→B, |Y|→C  
0→A, 0→CR"] --> Cond{Cn=1}
    Cond -- N --> A1["1/2A→A  
1/2C→C"]
    Cond -- Y --> A2["1/4(A+B)→A  
1/4C→C"]
    A1 --> CRplus1["CR+1→CR"]
    A2 --> CRplus1
    CRplus1 --> Cond2{CR=n?}
    Cond2 -- N --> End
    Cond2 -- Y --> PS["Xs ⊕ Ys → Ps"]
    PS --> End
  
```

2025年10月

北京理工大学计算机学院

4.4 定点乘法运算

计算机组成原理

## 4.4.2 补码一位乘法

### 比较法—Booth乘法

设：被乘数 $[X]_{\text{补}} = X_s X_1 X_2 \dots X_n$ ，乘数 $[Y]_{\text{补}} = Y_s Y_1 Y_2 \dots Y_n$ 。

在乘数的最低位之后增加一位附加位 $Y_{n+1}$ ，它的初值为0，增加附加位不会影响运算结果。

每次运算取决于乘数相邻两位 $Y_i$ 、 $Y_{i+1}$ 的值，把它们称为乘法的判断位。根据校正法的统一表达式推出：由乘数相邻两位的比较结果 $(Y_{i+1} - Y_i)$ 来确定运算操作。

2025年10月

北京理工大学计算机学院

**4.4 定点乘法运算**      **计算机组成原理**

**Booth乘法规则如下：**

- ① 参加运算的数用补码表示；
- ② 符号位参加运算；
- ③ 乘数最低位后面增加一位附加位  $Y_{n+1}$ ，其初值为0；
- ④ 由于每求一次部分积要右移一位，所以乘数的最低两位  $Y_n$ 、  $Y_{n+1}$  的值决定了每次应执行的操作；

北京理工大学计算机学院

2025年10月

**4.4 定点乘法运算**      **计算机组成原理**

| <b>判断位 <math>Y_n Y_{n+1}</math></b> | <b>操作</b>            |
|-------------------------------------|----------------------|
| 0 0                                 | 原部分积右移一位             |
| 0 1                                 | 原部分积加 $[X]_b$ 后右移一位  |
| 1 0                                 | 原部分积加 $[-X]_b$ 后右移一位 |
| 1 1                                 | 原部分积右移一位             |

北京理工大学计算机学院

2025年10月

**4.4 定点乘法运算**      **计算机组成原理**

**⑤ 移位按补码右移规则进行；**  
**⑥ 共需做  $n+1$  次累加，  $n$  次移位，第  $n+1$  次不移位。**

例：已知  $X=-0.1101$ ,  $Y=0.1011$ ; 求  $X \times Y$ 。  
 $[X]_b = 1.0011 \rightarrow B$ ,  $[Y]_b = 0.1011 \rightarrow C$ ,  
 $0 \rightarrow A$   
 $[-X]_b = 0.1101$

北京理工大学计算机学院

2025年10月

**4.4 定点乘法运算**      **计算机组成原理**

| A   | C 附加位   | 说明                                 |
|---|---|------------------------------------|
| $\begin{array}{r} 0 \\ +[-X]_b \\ \hline \end{array}$   | $0.1 \ 0 \ 1 \ \underline{\underline{1}} \ 0$ | $C_4C_5=10$ , $+[-X]_b$            |
| $\begin{array}{r} 0 \\ 0.0 \ 1 \ 1 \ 0 \ 1 \\ \rightarrow \\ 0 \\ \hline 0 \ 0 \ 0 \ 0 \ 0 \ 0 \end{array}$   | $1 \ 0 \ 1 \ 0 \ \underline{1} \ 1$           | 部分积右移一位<br>$C_4C_5=11$ , $+0$      |
| $\begin{array}{r} 0 \\ 0.0 \ 0 \ 1 \ 1 \ 0 \\ \rightarrow \\ 0 \\ \hline 0 \ 0 \ 0 \ 0 \ 1 \ 1 \end{array}$   | $0 \ 1 \ 0 \ 1 \ \underline{0} \ 1$           | 部分积右移一位<br>$C_4C_5=01$ , $+[X]_b$  |
| $\begin{array}{r} 0 \\ +[X]_b \\ \hline 1 \ 1 \ 0 \ 0 \ 1 \ 1 \end{array}$                                    | $1 \ 1 \ 0 \ 1 \ \underline{1} \ 0$           | 部分积右移一位<br>$C_4C_5=10$ , $+[-X]_b$ |
| $\begin{array}{r} 0 \\ 1 \ 1 \ 0 \ 0 \ 1 \ 1 \\ \rightarrow \\ 0 \\ \hline 0 \ 0 \ 1 \ 1 \ 0 \ 1 \end{array}$ | $0 \ 0 \ 1 \ 0 \ \underline{1} \ 0$           | 部分积右移一位<br>$C_4C_5=01$ , $+[X]_b$  |
| $\begin{array}{r} 0 \\ +[-X]_b \\ \hline 1 \ 1 \ 0 \ 0 \ 1 \ 1 \end{array}$                                   | $0 \ 0 \ 0 \ 1 \ \underline{0} \ 1$           | 部分积右移一位<br>$C_4C_5=10$ , $+[-X]_b$ |
| $\begin{array}{r} 0 \\ 1 \ 1 \ 0 \ 0 \ 1 \ 1 \\ \rightarrow \\ 0 \\ \hline 1 \ 1 \ 0 \ 1 \ 1 \ 1 \end{array}$ | $0 \ 0 \ 0 \ 1 \ 0 \ \underline{0} \ 1$       | 部分积右移一位<br>$C_4C_5=01$ , $+[-X]_b$ |
| $\therefore [X \times Y]_b = 1.01110001$  |   |                                    |
| $\therefore X \times Y = -0.10001111$   |   |                                    |

北京理工大学计算机学院

2025年10月

**4.4 定点乘法运算**      **计算机组成原理**

Diagram illustrating the Booth multiplication algorithm flowchart, showing the steps from input to output.

北京理工大学计算机学院

2025年10月

**4.4.3 补码两位乘法**

为了提高乘法的执行速度，可以选用两位乘法的方案。所谓两位乘法，就是每次处理乘数中的两位，从而使乘法的速度提高了一倍。两位乘法又可分为原码两位乘法和补码两位乘法，在此只讨论补码两位乘法。

根据前面介绍的Booth乘法方便地推导出补码两位乘法，即把补码两位乘理解为将Booth乘法的两次合并为一次来做。

北京理工大学计算机学院

2025年10月

4.4 定点乘法运算 计算机组成原理

补码两位乘法可以通过 $Y_{i-1}Y_iY_{i+1}$ 三位的不同组合来判断原部分积与 $[X]_b$ 的运算情况，然后右移两位得到新的部分积。



北京理工大学计算机学院

2025年10月

4.4 定点乘法运算 计算机组成原理

补码两位乘法规则如下：

- ① 参加运算的数用补码表示；
- ② 符号位参加运算；
- ③ 乘数最低位后增加一位附加位 $Y_{n+1}$ ，初值为0；
- ④ 根据乘数的最低三位 $Y_{n-1}Y_nY_{n+1}$ 的值决定每次应执行的操作；
- ⑤ 移位按补码右移规则进行。

比较结果 ( $Y_{i+1} + Y_i - 2Y_{i-1}$ ) 

北京理工大学计算机学院

2025年10月

4.4 定点乘法运算 计算机组成原理

| $Y_{n-1}Y_nY_{n+1}$ |                    |
|---------------------|--------------------|
| 0 0 0               | +0, 右移2位           |
| 0 0 1               | + $[X]_b$ , 右移2位   |
| 0 1 0               | + $[X]_b$ , 右移2位   |
| 0 1 1               | +2 $[X]_b$ , 右移2位  |
| 1 0 0               | +2 $[-X]_b$ , 右移2位 |
| 1 0 1               | + $[-X]_b$ , 右移2位  |
| 1 1 0               | + $[-X]_b$ , 右移2位  |
| 1 1 1               | +0, 右移2位           |



北京理工大学计算机学院

2025年10月

4.4 定点乘法运算 计算机组成原理

被乘数和部分积取三符号位，当乘数的数值位n为偶数时，乘数取两符号位，共需作 $(n/2)+1$ 次累加， $n/2$ 次移位（最后一次不移位）；当n为奇数时，乘数只需一个符号位，共需 $(n+1)/2$ 次累加和移位，但最后一次仅移一位。

已知： $X=0.0110011$ ,  $Y=-0.0110010$ , 求： $X \times Y$ 。

$[X]_b = 000.0110011 \rightarrow B$ , 

$[Y]_b = 1.1001110 \rightarrow C$ ,  $0 \rightarrow A$

$2[X]_b = 000.1100110$ ,  $[-X]_b = 111.1001101$ , 

$2[-X]_b = 111.0011010$  北京理工大学计算机学院

2025年10月

4.4 定点乘法运算 计算机组成原理

| A  | C                       | 附加位   |
|--|-------------------------|---|
| 0 0 0 0 0 0 0 0 0 0 0 0                        | 1.1 0 0 1 1 1 0 0       |   |
| +2 $[-X]_b$                                    | 1 1 1 0 0 1 1 0 1 0     |   |
|  | 1 1 1 0 0 1 1 0 1 0     |   |
| 2 →  | 1 1 1 1 1 0 0 1 1 0     |   |
| +0   | 0 0 0 0 0 0 0 0 0 0 0 0 |   |
|  | 0 0 0 0 0 0 0 0 0 0 0 0 |   |
| + [X]_b  | 1 1 1.1 1 0 0 1 1 0     |   |
|  | 1 1 1.1 1 1 1 1 0 0 1   |   |
| 2 →  | 0 0 0.0 1 1 0 0 1 1     |   |
| + [-X]_b                                       | 0 0 0.0 1 0 1 1 0 0     |   |
|  | 0 0 0.0 0 0 1 0 1 1     |   |
| 2 →  | 0 0 1 0 1 0 1 1 0       |   |
| + [-X]_b                                       | 1 1 1.1 0 0 1 1 0 1     |   |
|  | 1 1 1.1 1 0 1 1 0 0     |   |
| 1 →  | 1 1 1.1 1 0 1 1 0 0     |   |
| $\therefore [X \times Y]_b = 1.11011000001010$ | 0 0 0 1 0 1 0 1 1       |  |
| $X \times Y = 0.0010011110110$                 |                         |   |

北京理工大学计算机学院

2025年10月

第4章 计算机组成原理

- 4.1 基本算术运算的实现
- 4.2 定点加减运算
- 4.3 带符号数的移位和舍入操作
- 4.4 定点乘法运算
- 4.5 定点除法运算
- 4.6 规格化浮点运算
- 4.7 十进制整数的加减运算
- 4.8 逻辑运算与实现
- 4.9 运算器的基本组成与实例



北京理工大学计算机学院

2025年10月

4.5 定点除法运算 计算机组成原理

### 4.5.1 原码除法运算

$X=0.1011, Y=0.1101$

$$\begin{array}{r} 0.1101 \\ \underline{\quad\quad\quad\quad} \\ 0.1101 \\ \underline{0.01101} \\ 0.010010 \\ \underline{0.001101} \\ 0.00010100 \\ \underline{0.00001101} \\ 0.00000111 \end{array} \dots \text{商}$$

$X \div Y = \text{商} + \frac{\text{余数}}{\text{除数}} = 0.1101 + 0.0111 \times 2^{-4} / 0.1101$

北京理工大学计算机学院

2025年10月

4.5 定点除法运算 计算机组成原理

### 1. 原码比较法和恢复余数法

#### (1) 比较法

比较法类似于手工运算，只是为了便于机器操作，将除数右移改为余数左移。

比较法要对两个操作数进行比较( $A > B?$ )，这就需要设置比较线路，从而增加了硬件的代价。

北京理工大学计算机学院

2025年10月

4.5 定点除法运算 计算机组成原理

#### (2) 恢复余数法

恢复余数法是直接作减法试探方法，不管被除数（或余数）减除数是否够减，都一律先做减法。若余数为正，表示够减，该位商上“1”；若余数为负，表示不够减，该位商上“0”，并要恢复原来的被除数（或余数）。

北京理工大学计算机学院

2025年10月

4.5 定点除法运算 计算机组成原理

由于余数的正、负是根据不同的操作数组合随机出现的，这就使得除法运算的实际操作次数不固定，从而导致控制电路比较复杂。而且在恢复余数时，要多作一次加法，降低了执行速度。因此，原码恢复余数法在计算机中一般很少采用。

北京理工大学计算机学院

2025年10月

4.5 定点除法运算 计算机组成原理

### 2. 原码不恢复余数法（原码加减交替法）

原码不恢复余数法是对恢复余数法的一种改进。在恢复余数法中，若第*i*-1次求商的余数为 $r_{i-1}$ ，则第*i*次求商操作为：  
 $r_i = 2r_{i-1} - Y$

若够减， $r_i = 2r_{i-1} - Y > 0$ ，商1。若不够减， $r_i = 2r_{i-1} - Y < 0$ ，商0，恢复余数后， $r'_i = r_i + Y = 2r_{i-1}$ ，然后再左移一位，进行第*i*+1次操作：

$$r_{i+1} = 2r'_i - Y = 2(r_i + Y) - Y = 2r_i + 2Y - Y = 2r_i + Y$$

北京理工大学计算机学院

2025年10月

4.5 定点除法运算 计算机组成原理

上式表明，当出现不够减（负余数）的情况下并不需要恢复余数，可以直接做下一次操作，但操作是 $2r_i + Y$ ，其结果与恢复余数后左移一位再减 $Y$ 是等效的。

$$r_{i+1} = 2r_i + (1-2Q_i) \times Y$$

北京理工大学计算机学院

2025年10月

**4.5 定点除法运算**      **计算机组成原理**

**原码加减交替除法规则：**

- 1) 第一步做被除数减去除数。
- 2) 当余数为正时，商“1”，余数左移一位，减去除数；当余数为负时，商“0”，余数左移一位，加去除数。
- 3) 重复步骤2)，共做 $n+1/n+2$ 步加减运算( $n$ 步移位)。

北京理工大学计算机学院

2025年10月

**4.5 定点除法运算**      **计算机组成原理**

**除法运算需要3个寄存器：**

- A寄存器：存放被除数X，最后A寄存器中剩下的是扩大了若干倍的余数。运算过程中A寄存器的内容将不断地发生变化。
- B寄存器：存放除数Y。
- C寄存器：存放商Q，它的初值为0。

北京理工大学计算机学院

2025年10月

**4.5 定点除法运算**      **计算机组成原理**

**例：**已知 $X=0.10101$ ,  $Y=-0.11110$ , 求：  
 $X \div Y$ 。

$|X|=0.10101 \rightarrow A$ ,  $|Y|=0.11110 \rightarrow B$ ,  
 $0 \rightarrow C$

$||Y||_{\text{变补}} = 1.00010$

北京理工大学计算机学院

2025年10月

**4.5 定点除法运算**      **计算机组成原理**

| A               | C             | 说明                                |
|-----------------|---------------|-----------------------------------|
| 0 0 1 0 1 0 1 0 | 0 0 0 0 0 0 0 | $- Y $                            |
| 1 1 1 0 1 1 1 0 | 0 0 0 0 0 0 0 | 余数为负，商0<br>左移一位                   |
| 0 0 1 1 1 1 1 0 | 0 0 0 0 0 0 1 | $+ Y $<br>余数为正，商1<br>左移一位         |
| 0 0 0 1 1 0 0 0 | 0 0 0 0 0 1 0 | $- Y $<br>余数为负，商0<br>左移一位         |
| 0 0 1 1 0 0 0 0 | 0 0 0 0 1 0 0 | $+ Y $<br>余数为正，商1<br>左移一位         |
| 1 1 1 0 0 0 1 0 | 0 0 0 1 0 0 1 | $- Y $<br>余数为负，商0<br>左移一位         |
| 0 0 1 1 1 1 1 0 | 0 0 0 1 0 1 0 | $+ Y $<br>余数为正，商1<br>左移一位         |
| 0 0 1 0 0 0 1 0 | 0 0 1 0 1 0 1 | $- Y $<br>余数为负，商0<br>左移一位         |
| 1 1 0 0 0 0 1 0 | 0 0 1 0 1 1 0 | $+ Y $<br>余数为正，商1<br>左移一位         |
| 0 0 0 0 0 1 1 0 | 0 1 0 1 1 1 0 | $- Y $<br>余数为负，商0<br>恢复余数， $+ Y $ |
| 0 0 0 1 1 0 0 0 |               |                                   |

北京理工大学计算机学院

2025年10月

**4.5 定点除法运算**      **计算机组成原理**

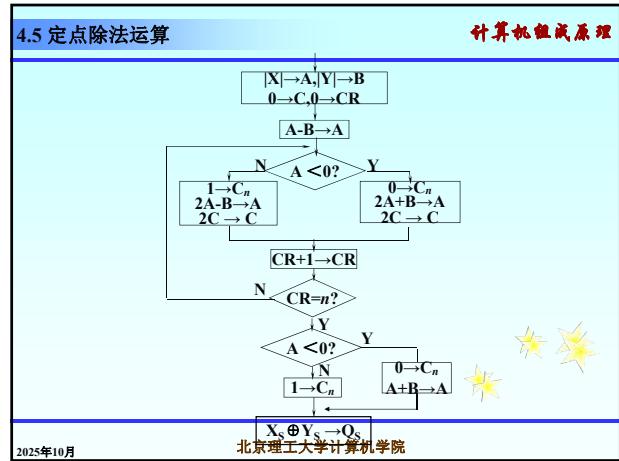
经过原码加减交替除法，有：  
 $\therefore \text{商} = 0.10110$

$\text{余数} = 0.01100 \times 2^{-5}$

$X \div Y = -(0.10110 + \frac{0.01100 \times 2^{-5}}{0.11110})$

北京理工大学计算机学院

2025年10月



**4.5 定点除法运算**      **计算机组成原理**

需要指出的是，在定点小数除法运算时，为了防止溢出，要求被除数的绝对值小于除数的绝对值，即 $|X| < |Y|$ ，且除数不能为0。另外，在原码加减交替法中，当最终余数为负数时，必须恢复一次余数，使之变为真余数，注意此时不需要再左移了。

北京理工大学计算机学院

2025年10月

**4.5 定点除法运算**      **计算机组成原理**

**4.5.2 补码除法运算**

被除数和除数都用补码表示，符号位参加运算。

**1.够减的判断**

参加运算的两个数符号情况如下：

(1) 同号  $X > 0, Y > 0, X - Y > 0$   
 $X < 0, Y < 0, -X - (-Y) > 0$

(2) 异号  $X > 0, Y < 0, X - (-Y) = (X + Y) > 0$   
 $X < 0, Y > 0, (-X) - Y > 0 \Rightarrow X + Y < 0$

北京理工大学计算机学院

2025年10月

**4.5 定点除法运算**      **计算机组成原理**

**2.上商规则**

如果 $[X]_b$ 和 $[Y]_b$ 同号，则商为正数，上商规则与原码除法相同，即够减时上商“1”，不够减时上商“0”；如果 $[X]_b$ 和 $[Y]_b$ 异号，则商为负数，上商规则与同号时相反，即够减时上商“0”，不够减时上商“1”。

将上商规则与够减的判断结合起来，可得到本次余数 $[r_i]_b$ 和除数 $[Y]_b$ 同号，商上“1”，反之，商上“0”。

北京理工大学计算机学院

2025年10月

**4.5 定点除法运算**      **计算机组成原理**

**3.商符的确定**

商符是在求商的过程中自动形成的，按补码上商规则，第一次得出的商，就是实际应得的商符。

**4.求新部分余数**

求新余数 $[r_{i+1}]_b$ 的通式如下：  

$$[r_{i+1}]_b = 2[r_i]_b + (1-2Q_i) \times [Y]_b$$

$Q_i$ 表示第*i*步的商。若商上“1”，下一次操作为余数左移一位，减去除数；若商上“0”，下一次操作为余数左移一位，加上除数。

**5.末位恒置1**

北京理工大学计算机学院

2025年10月

**4.5 定点除法运算**      **计算机组成原理**

**补码加减交替除法规则：**

| $[X]_b$ 与 $[Y]_b$ | 第一次操作           | $[r_i]_b$ 与 $[Y]_b$ | 上商 | 求新余数 $[r_{i+1}]_b$ 的操作           |
|-------------------|-----------------|---------------------|----|----------------------------------|
| 同号                | $[X]_b - [Y]_b$ | ①同号<br>(够减)         | 1  | $[r_{i+1}]_b = 2[r_i]_b - [Y]_b$ |
|                   |                 | ②异号<br>(不够减)        | 0  | $[r_{i+1}]_b = 2[r_i]_b + [Y]_b$ |
| 异号                | $[X]_b + [Y]_b$ | ①同号<br>(不够减)        | 1  | $[r_{i+1}]_b = 2[r_i]_b - [Y]_b$ |
|                   |                 | ②异号<br>(够减)         | 0  | $[r_{i+1}]_b = 2[r_i]_b + [Y]_b$ |

北京理工大学计算机学院

2025年10月

**4.5 定点除法运算**      **计算机组成原理**

**补码加减交替除法规则：**

- 1) 被除数与除数同号，被除数减去除数；被除数与除数异号，被除数加上除数。
- 2) 余数和除数同号，商上1，余数左移一位，下次减除数；余数和除数异号，商上0，余数左移一位，下次加除数。
- 3) 重复步骤2)，包括符号位在内，共做n+1步加减运算(n步移位)。
- 4) 商的末位恒置1。

北京理工大学计算机学院

2025年10月

| 4.5 定点除法运算  |          | 计算机组成原理   |
|---|----------|---|
| 已知: $X=0.1000, Y=-0.1010$ ; 求 $X \div Y$  |          |   |
| $[X]_{\text{补}}=0.1000 \rightarrow A, [Y]_{\text{补}}=1.0110 \rightarrow B, 0 \rightarrow C, [-Y]_{\text{补}}=0.1010$ |          |   |
| $+ [Y]_{\text{补}}$  | $0.0000$ | $[X]_{\text{补}}, [Y]_{\text{补}} \text{ 异号}, + [Y]_{\text{补}}$   |
| $\begin{array}{r} 0.0.1000 \\ 1.1.0110 \\ \hline 1.1.1110 \end{array}$  | $0.0001$ | $[r_i]_{\text{补}}, [Y]_{\text{补}} \text{ 同号}, \text{商}1$        |
| $\overleftarrow{-} [Y]_{\text{补}}$  | $0.0010$ | $\begin{array}{l} \text{左移一位} \\ + [-Y]_{\text{补}} \end{array}$ |
| $+ [Y]_{\text{补}}$  | $0.0010$ | $[r_i]_{\text{补}}, [Y]_{\text{补}} \text{ 异号}, \text{商}0$        |
| $\overleftarrow{-} [Y]_{\text{补}}$  | $0.0100$ | $\begin{array}{l} \text{左移一位} \\ + [Y]_{\text{补}} \end{array}$  |
| $+ [Y]_{\text{补}}$  | $0.0100$ | $[r_i]_{\text{补}}, [Y]_{\text{补}} \text{ 同号}, \text{商}0$        |
| $\overleftarrow{-} [Y]_{\text{补}}$  | $0.1001$ | $\begin{array}{l} \text{左移一位} \\ + [Y]_{\text{补}} \end{array}$  |
| $+ [Y]_{\text{补}}$  | $0.1001$ | $[r_i]_{\text{补}}, [Y]_{\text{补}} \text{ 同号}, \text{商}1$        |
| $\overleftarrow{-} [Y]_{\text{补}}$  | $0.0101$ | $\begin{array}{l} \text{左移一位} \\ + [Y]_{\text{补}} \end{array}$  |
| $+ [Y]_{\text{补}}$  | $1.0011$ | 末位恒置1   |

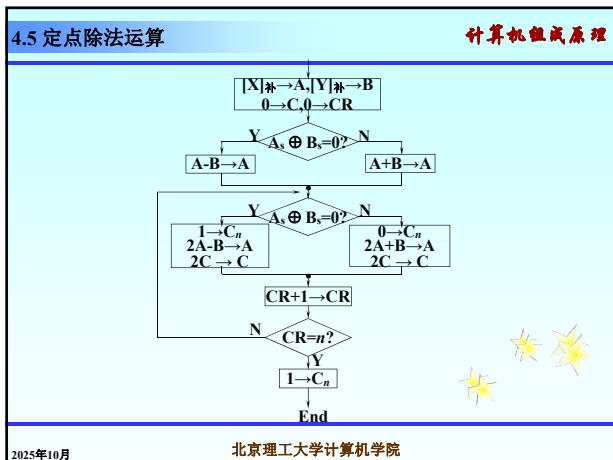
2025年10月

北京理工大学计算机学院

| 4.5 定点除法运算   |  | 计算机组成原理   |
|--|--|---|
| $[商]_{\text{补}}=1.0011$  |  |   |
| $[余数]_{\text{补}}=1.1110 \times 2^{-4}$                               |  |   |
| $[X \div Y]_{\text{补}}=1.0011 + \frac{1.1110 \times 2^{-4}}{1.0110}$ |  |   |
| $\therefore \text{商}=-0.1101$  |  |   |
| $\text{余数}=-0.0010 \times 2^{-4}$                                    |  | $X \div Y = -0.1101 + \frac{-0.0010 \times 2^{-4}}{-0.1010}$                        |
|  |  |  |

2025年10月

北京理工大学计算机学院



2025年10月

北京理工大学计算机学院

| 第4章              |  | 计算机组成原理   |
|------------------|--|---|
| 4.1 基本算术运算的实现    |  |   |
| 4.2 定点加减运算       |  |   |
| 4.3 带符号数的移位和舍入操作 |  |   |
| 4.4 定点乘法运算       |  |   |
| 4.5 定点除法运算       |  |   |
| 4.6 规格化浮点运算      |  |   |
| 4.7 十进制整数的加减运算   |  |  |
| 4.8 逻辑运算与实现      |  |   |
| 4.9 运算器的基本组成与实例  |  |   |

2025年10月

北京理工大学计算机学院

| 4.6 规格化浮点运算  |  | 计算机组成原理   |
|--|--|---|
| 4.6.1 浮点加减运算   |  |   |
| 设两个非0的规格化浮点数分别为  |  |   |
| $A=M_A \times 2^{E_A}$                                       |  |   |
| $B=M_B \times 2^{E_B}$                                       |  |   |
| $A \pm B = (M_A, E_A) \pm (M_B, E_B) =$                      |  |   |
| $M_A \pm M_B \times 2^{-(E_A - E_B)}, E_A \quad E_A > E_B$   |  |   |
| $(M_A \times 2^{-(E_B - E_A)} \pm M_B, E_B) \quad E_A < E_B$ |  |  |

2025年10月

北京理工大学计算机学院

| 4.6 规格化浮点运算  |  | 计算机组成原理   |
|--|--|---|
| 1. 浮点数加减运算步骤   |  |   |
| (1) 对阶   |  |   |
| 两个浮点数相加或相减，首先要把小数点的位置对齐，而浮点数的小数点的实际位置取决于阶码的大小。因此，对齐两数的小数点，就是使两数的阶码相等，这个过程称为对阶。 |  |   |
| 要对阶，首先应求出两数阶码 $E_A$ 和 $E_B$ 之差，即： $\Delta E = E_A - E_B$                       |  |  |

2025年10月

北京理工大学计算机学院

**4.6 规格化浮点运算** **计算机组成原理**

若  $\Delta E = 0$ , 表示两数阶码相等, 即  $E_A = E_B$ ; 若  $\Delta E > 0$ , 表示  $E_A > E_B$ ; 若  $\Delta E < 0$ , 表示  $E_A < E_B$ 。

当  $E_A \neq E_B$  时, 要通过尾数的移位来改变  $E_A$  或  $E_B$ , 使  $E_A = E_B$  相等。

对阶的规则是: 小阶向大阶看齐。采用这一规则的原因是当阶码小的数的尾数右移并相应增加阶码时, 舍去的仅是尾数低位部分, 误差比较小。要使小阶的阶码增大, 则相应的尾数右移, 直到两数的阶码相等为止。对于  $r=2$ , 每右移一位, 阶码加1。

北京理工大学计算机学院

2025年10月

**4.6 规格化浮点运算** **计算机组成原理**

$E_A = E_B$ , 不须对阶。

$E_A > E_B$ , 则  $M_B$  右移。每右移一位,  $E_B + 1 \rightarrow E_B$ , 直至  $E_A = E_B$  为止。

$E_A < E_B$ , 则  $M_A$  右移。每右移一位,  $E_A + 1 \rightarrow E_A$ , 直至  $E_A = E_B$  为止。

尾数右移后, 应对尾数进行舍入。

(2) 尾数加/减

对阶之后, 就可以进行尾数加/减, 即  $M_A \pm M_B \rightarrow M_C$

北京理工大学计算机学院

2025年10月

**4.6 规格化浮点运算** **计算机组成原理**

(3) 尾数结果规格化

尾数加/减运算之后得到的数可能不是规格化数, 为了增加有效数字的位数, 提高运算精度, 必须进行结果规格化操作。

规格化的尾数  $M$  应满足:

$$\frac{1}{2} \leq |M| < 1$$

设尾数用双符号位补码表示, 经过加/减运算之后, 可能出现以下六种情况:

北京理工大学计算机学院

2025年10月

**4.6 规格化浮点运算** **计算机组成原理**

- ① 00.1 x x ... x
- ② 11.0 x x ... x
- ③ 00.0 x x ... x
- ④ 11.1 x x ... x
- ⑤ 01.x x x ... x
- ⑥ 10.x x x ... x

第①、②种情况, 已是规格化数。

第③、④种情况需要使尾数左移以实现规格化, 这个过程称为左规。

北京理工大学计算机学院

2025年10月

**4.6 规格化浮点运算** **计算机组成原理**

尾数每左移一位, 阶码相应减1 ( $E_C - 1 \rightarrow E_C$ ), 直至成为规格化数为止。(左规可能需进行多次) 只要满足下列条件:

$$\text{左规} = \bar{C}_{s1} \bar{C}_{s2} \bar{C}_1 + C_{s1} C_{s2} C_1$$

第⑤、⑥种情况在定点加减运算中称为溢出; 但在浮点加减运算中, 只表明此时尾数的绝对值大于1, 而并非真正的溢出。这种情况应将尾数右移以实现规格化。这个过程称为右规。尾数每右移一位, 阶码相应加1 ( $E_C + 1 \rightarrow E_C$ )。(右规最多进行一次) 右规的条件如下:

$$\text{右规} = \bar{C}_{s1} \oplus C_{s2}$$

北京理工大学计算机学院

2025年10月

**4.6 规格化浮点运算** **计算机组成原理**

(4) 舍入

由于受到硬件的限制, 在对阶和右规处理之后有可能将尾数的低位丢失, 这会引起一些误差。舍入方法已在前面做过介绍, 这里不再赘述。

北京理工大学计算机学院

2025年10月

**4.6 规格化浮点运算**      **计算机组成原理**

**(5)溢出判断**

当尾数之和（差）出现 $10.x\ x\ x\dots x$ 或 $01.x\ x\ x\dots x$ 时，并不表示溢出，只有将此数右规后，再根据阶码来判断浮点运算结果是否溢出。

浮点数的溢出情况由阶码的符号决定，若阶码也用双符号位补码表示，当：

北京理工大学计算机学院

2025年10月

**4.6 规格化浮点运算**      **计算机组成原理**

$[E_C]_{\text{补}}=01, x\ x\ x\dots x$ ，表示上溢。此时，浮点数真正溢出，机器需停止运算，做溢出中断处理。

$[E_C]_{\text{补}}=10, x\ x\ x\dots x$ ，表示下溢。浮点数值趋于零，机器不做溢出处理，而是按机器零处理。

北京理工大学计算机学院

2025年10月

**4.6 规格化浮点运算**      **计算机组成原理**

**2. 浮点数加减运算举例**

有两浮点数为  
 $A=0.101110 \times 2^{+1}$   
 $B=-(0.101011) \times 2^{-10}$

假设这两数的格式：阶码4位，用移码表示（偏置值为 $2^3$ ）；尾数8位，用补码表示，包含一位符号位，即

|   |       |   |
|---|-------|---|
| E | $m_s$ | m |
|---|-------|---|

$[A]_{\text{浮}}=0111; 0.1011100$   
 $[B]_{\text{浮}}=0110; 1.0101010$

北京理工大学计算机学院

2025年10月

**4.6 规格化浮点运算**      **计算机组成原理**

**(1)对阶**

求阶差： $\Delta E=E_A-E_B=-1-(-2)=1$   
 $\Delta E=1$ ，表示 $E_A > E_B$ 。按对阶规则，将 $M_B$ 右移一位，其阶码加1，得：

$[B]_{\text{浮}}'=0111; 1.1010101$

**(2)尾数求和**

|            |   |            |       |            |
|------------|---|------------|-------|------------|
| 00.1011100 | + | 11.1010101 | ————— | 00.0110001 |
|------------|---|------------|-------|------------|

北京理工大学计算机学院

2025年10月

**4.6 规格化浮点运算**      **计算机组成原理**

**(3)尾数结果规格化**

由于结果的尾数是非规格化的数，故应左规。尾数每左移一位，阶码减1，直至尾数成为规格化数为止。

$[A+B]_{\text{尾补}}=00.0110001$   
 $[A+B]_{\text{尾补}}'=00.1100010 \times 2^{-10}$

最后结果为

$[A+B]_{\text{浮}}'=0110; 0.1100010$   
 $\therefore A+B=(0.110001) \times 2^{-10}$

北京理工大学计算机学院

2025年10月

**4.6.2 浮点乘除运算**

设两个非0的规格化浮点数分别为  
 $A=M_A \times 2^{E_A}$   
 $B=M_B \times 2^{E_B}$

则浮点乘法和除法为  
 $A \times B = (M_A \times M_B) \times 2^{(E_A+E_B)}$   
 $A \div B = (M_A \div M_B) \times 2^{(E_A-E_B)}$

北京理工大学计算机学院

2025年10月

**4.6 规格化浮点运算** **计算机组成原理**

### 1. 乘法步骤

#### (1) 阶码相加

两个浮点数的阶码相加，当阶码用移码表示的时候，应注意要减去一个偏置值 $2^n$ 。

因为 $[E_A]_{\text{移}}=2^n+E_A$ ,  $[E_B]_{\text{移}}=2^n+E_B$   
 $[E_A+E_B]_{\text{移}}=2^n+(E_A+E_B)$   
而 $[E_A]_{\text{移}}+[E_B]_{\text{移}}=2^n+E_A+2^n+E_B$

显然，此时阶码和中多余了一个偏置值 $2^n$ ，应将它减去。另外，阶码相加后有可能产生溢出，此时应另作处理。

2025年10月 北京理工大学计算机学院

**4.6 规格化浮点运算** **计算机组成原理**

#### (2) 尾数相乘

与定点小数乘法算法相同。

#### (3) 尾数结果规格化

因为 $1/2 < |M_A| < 1$ ,  $1/2 < |M_B| < 1$ , 所以 $1/4 < |M_A \times M_B| < 1$ 。

当 $1/2 < |M_A \times M_B| < 1$ 时，乘积已是规格化数，不须再进行规格化操作；当 $1/4 < |M_A \times M_B| < 1/2$ 时，则需要左规一次。

2025年10月 北京理工大学计算机学院

**4.6 规格化浮点运算** **计算机组成原理**

### 2. 除法步骤

#### (1) 尾数调整

首先须要检测 $|M_A| < |M_B|$ 。如果不小于，则 $M_A$ 右移一位， $E_A+1 \rightarrow E_A$ ，称为尾数调整。因为A、B都是规格化数，所以最多调整一次。

#### (2) 阶码相减

两浮点数的阶码相减，当阶码用移码表示时，应注意要加上一个偏置值 $2^n$ 。

#### (3) 尾数相除

与定点小数除法算法相同。

2025年10月 北京理工大学计算机学院

**第4章** **计算机组成原理**

- 4.1 基本算术运算的实现
- 4.2 定点加减运算
- 4.3 带符号数的移位和舍入操作
- 4.4 定点乘法运算
- 4.5 定点除法运算
- 4.6 规格化浮点运算
- 4.7 十进制整数的加减运算
- 4.8 逻辑运算与实现
- 4.9 运算器的基本组成与实例

2025年10月 北京理工大学计算机学院

**4.7 十进制整数的加减运算** **计算机组成原理**

### 4.7.1 一位十进制加法运算

#### 1.8421码加法运算

因为一位8421码用四位二进制数表示，所以8421码十位数的“1”是个位数的进位。按四位二进制数而言，这个进位的值是16，而不是8421码的10。因此，必须+6校正，才能使该进位正确。8421码的加法规则：

- ①两个8421码相加时，“逢二进一”；
- ②当和 $\leq 9$ ，无需校正；
- ③当和 $> 9$ ，则+6校正；
- ④在做+6校正的同时，将产生向上一位的进位。

2025年10月 北京理工大学计算机学院

**4.7 十进制整数的加减运算** **计算机组成原理**

| 十进制数 | 8421码<br>C4S4S3S2S1 | 校正前的二进制数<br>C4'S4'S3'S2'S1' | 校正与否 |
|------|---------------------|-----------------------------|------|
| 0    | 0 0 0 0             | 0 0 0 0 0                   | 不校正  |
| 1    | 1                   | 1                           |      |
| 9    | 0 1 0 0 1           | 0 1 0 0 1                   |      |
| 10   | 1 0 0 0 0           | 0 1 0 1 0                   | +6校正 |
| 11   | 1 0 0 0 1           | 0 1 0 1 1                   |      |
| 12   | 1 0 0 1 0           | 0 1 1 0 0                   |      |
| 13   | 1 0 0 1 1           | 0 1 1 0 1                   |      |
| 14   | 1 0 1 0 0           | 0 1 1 1 0                   |      |
| 15   | 1 0 1 0 1           | 0 1 1 1 1                   |      |
| 16   | 1 0 1 1 0           | 1 0 0 0 0                   |      |
| 17   | 1 0 1 1 1           | 1 0 0 0 1                   |      |
| 18   | 1 1 0 0 0           | 1 0 0 1 0                   |      |
| 19   | 1 1 0 0 1           | 1 0 0 1 1                   |      |

+6校正函数=C4'+S4'S3'+S4'S2'

2025年10月 北京理工大学计算机学院

4.7 十进制整数的加减运算 计算机组成原理

### 校正举例

$$\begin{array}{r} 0101 \\ + 1000 \\ \hline 1101 \\ + 0110 \\ \hline 10011 \end{array} \quad \begin{array}{r} 5 \\ 8 \\ 6 \\ 13 \end{array}$$
  

$$\begin{array}{r} 1001 \\ + 1000 \\ \hline 10001 \\ + 0110 \\ \hline 10111 \end{array} \quad \begin{array}{r} 9 \\ 8 \\ 6 \\ 17 \end{array}$$

北京理工大学计算机学院

2025年10月

4.7 十进制整数的加减运算 计算机组成原理

### 2.余3码加法运算

十进制余3码加法规则：

- ①两个余3码相加，“逢二进一”；
- ②若其和没有进位，则减3（即+1101）校正；
- ③若其和有进位，则加3（即+0011）校正。

北京理工大学计算机学院

2025年10月

4.7 十进制整数的加减运算 计算机组成原理

| 十进制数 | 余3码 C4S4S3S2S1 | 校正前的二进制数 C4'S4'S3'S2'S1' | 校正与否 |
|------|----------------|--------------------------|------|
| 0    | 0 0 0 1 1      | 0 0 1 1 0                | -3校正 |
| 1    | 0 0 1 0 0      | 0 0 1 1 1                |      |
|      |                |                          |      |
| 8    | 0 1 0 1 1      | 0 1 1 1 0                |      |
| 9    | 0 1 1 0 0      | 0 1 1 1 1                |      |
| 10   | 1 0 0 1 1      | 1 0 0 0 0                | +3校正 |
| 11   | 1 0 1 0 0      | 1 0 0 0 1                |      |
|      |                |                          |      |
| 18   | 1 1 0 1 1      | 1 1 0 0 0                |      |
| 19   | 1 1 1 0 0      | 1 1 0 0 1                |      |

-3校正函数=C4'

+3校正函数=C4'

北京理工大学计算机学院

2025年10月

4.7 十进制整数的加减运算 计算机组成原理

### 校正举例

$$\begin{array}{r} 0110 \\ + 1000 \\ \hline 1110 \\ + 1101 \\ \hline 1011 \end{array} \quad \begin{array}{r} 3 \\ 5 \\ 无进位, -3, +1101 \\ 8 \end{array}$$
  

$$\begin{array}{r} 1001 \\ + 1000 \\ \hline 10001 \\ + 0011 \\ \hline 10100 \end{array} \quad \begin{array}{r} 6 \\ 5 \\ 有进位, +3 \\ 11 \end{array}$$

北京理工大学计算机学院

2025年10月

4.7 十进制整数的加减运算 计算机组成原理

### 4.7.2 十进制加法器

#### 1.一位8421码加法器

北京理工大学计算机学院

2025年10月

4.7 十进制整数的加减运算 计算机组成原理

### 2.一位余3码加法器

北京理工大学计算机学院

2025年10月

**第4章** 计算机组成原理

4.1 基本算术运算的实现  
 4.2 定点加减运算  
 4.3 带符号数的移位和舍入操作  
 4.4 定点乘法运算  
 4.5 定点除法运算  
 4.6 规格化浮点运算  
 4.7 十进制整数的加减运算  
**4.8 逻辑运算与实现**  
**4.9 运算器的基本组成与实例**

☆☆

2025年10月 北京理工大学计算机学院

**4.8 逻辑运算与实现** 计算机组成原理

逻辑运算比算术运算要简单得多，这是因为**逻辑运算是按位进行的，位与位之间没有进位/借位的关系。**

1.逻辑非  
 逻辑非又称求反操作，它对某个寄存器或主存单元中各位代码按位取反。  
 2.逻辑乘  
 逻辑乘就是将两个寄存器或主存单元中的每一相应位的代码进行“与”操作。

☆☆

2025年10月 北京理工大学计算机学院

**4.8 逻辑运算与实现** 计算机组成原理

3.逻辑加  
 逻辑加就是将两个寄存器或主存单元中的每一相应位的代码进行“或”操作。  
 4.按位异或  
 按位异或是计算机中一个特定的逻辑操作，它对寄存器或主存单元中各位的代码求模2和，又称模2加或半加，也叫异或。

☆☆

2025年10月 北京理工大学计算机学院

**第4章** 计算机组成原理

4.1 基本算术运算的实现  
 4.2 定点加减运算  
 4.3 带符号数的移位和舍入操作  
 4.4 定点乘法运算  
 4.5 定点除法运算  
 4.6 规格化浮点运算  
 4.7 十进制整数的加减运算  
**4.8 逻辑运算与实现**  
**4.9 运算器的基本组成与实例**

☆☆

2025年10月 北京理工大学计算机学院

**4.9 运算器的基本组成与实例** 计算机组成原理

运算器是在控制器的控制下实现其功能的，运算器不仅可以完成数据信息的算逻运算，还可以作为数据信息的传送通路。

☆☆

2025年10月 北京理工大学计算机学院

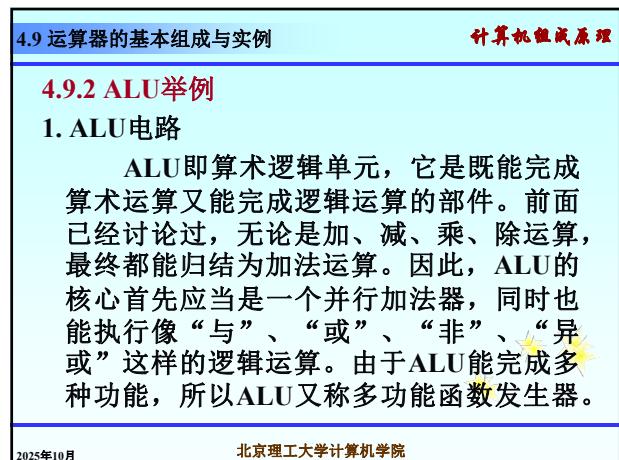
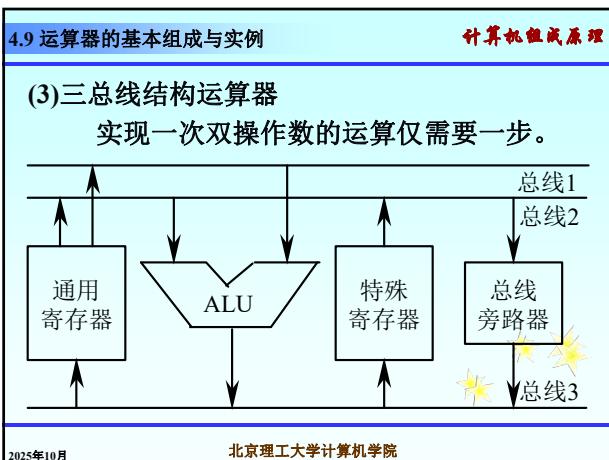
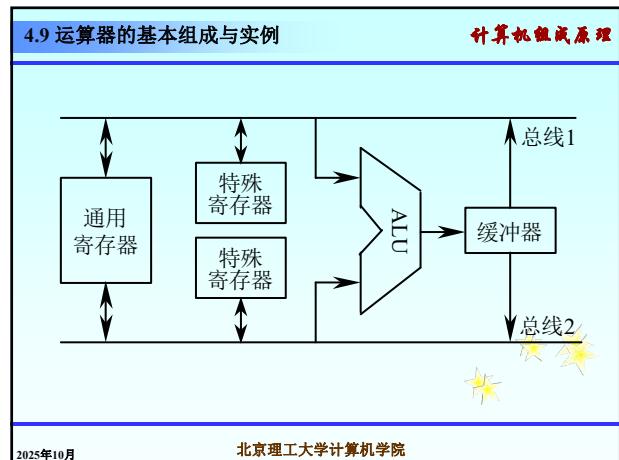
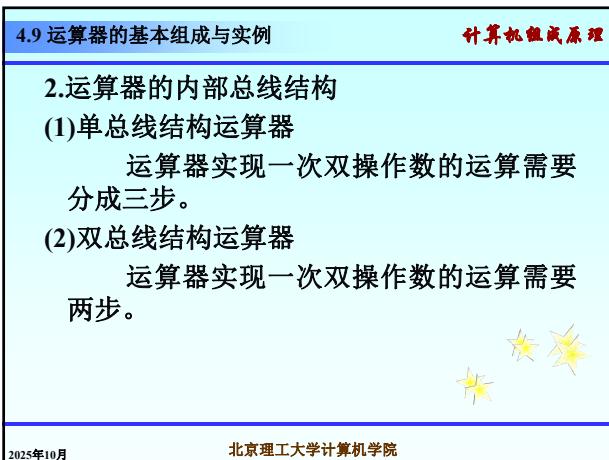
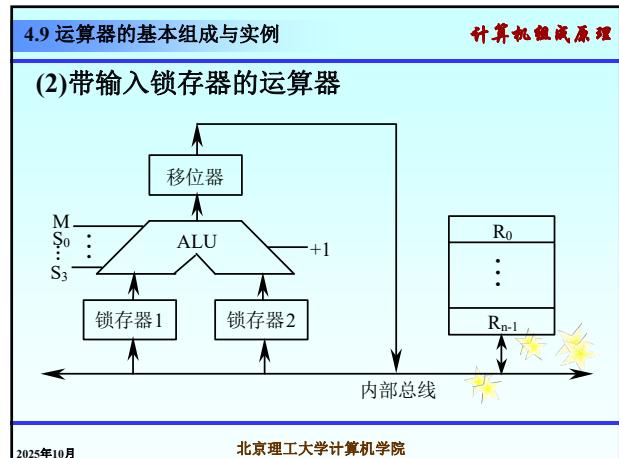
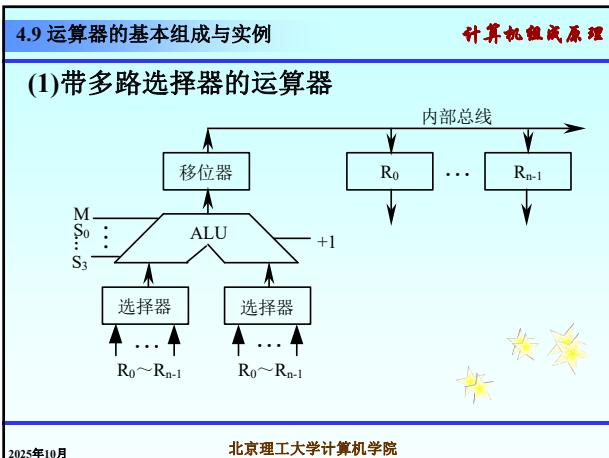
**4.9 运算器的基本组成与实例** 计算机组成原理

**5.9.1 运算器结构**

1.运算器的基本组成  
 基本的运算器包含以下几个部分：  
 实现基本算术、逻辑运算功能的ALU，  
 提供操作数与暂存结果的寄存器组，  
 有关的判别逻辑和控制电路等。

☆☆

2025年10月 北京理工大学计算机学院





**4.9 运算器的基本组成与实例** **计算机组成原理**

74181和74182的结合可组成各种位数的ALU部件。  
8片74181和2片74182构成的32位两级行波ALU。各片74181输出的组进位产生函数和组进位传递函数作为74182的输入，而74182输出的进位信号 $C_{n+x}$ 、 $C_{n+y}$ 、 $C_{n+z}$ 作为74181的输入，74182输出的大组进位产生函数和大组进位传递函数可作为更高一级74182的输入。

北京理工大学计算机学院

2025年10月

**第4章 小结** **计算机组成原理**

**4.1 基本运算的实现**

- ✓ 加法器
- 串行加法器与并行加法器
- 进位的产生和传递
- 并行加法器快速进位

北京理工大学计算机学院

2025年10月

**第4章 小结** **计算机组成原理**

**4.2 定点加减运算**

- ✓ 补码加法运算
- ✓ 补码减法运算
- 已知 $|Y|_补$ 求 $|-Y|_补$ 的方法
- ✓ 补码的溢出判断与检测方法
- 一位符号位，进位位，双符号位补码

北京理工大学计算机学院

2025年10月

**第4章 小结** **计算机组成原理**

**4.3 带符号数的移位运算和舍入操作**

- ✓ 补码的移位运算
- ✓ 舍入操作

**4.4 定点乘法运算**

- ✓ 原码一位乘法
- ✓ 补码一位乘法

北京理工大学计算机学院

2025年10月

**第4章 小结** **计算机组成原理**

**4.5 定点除法运算**

- ✓ 原码加减交替除法
- ✓ 补码加减交替除法

**4.6 规格化浮点运算算法**

- ✓ 浮点加减运算
- ✓ 浮点乘除运算

北京理工大学计算机学院

2025年10月

**第4章 小结** **计算机组成原理**

**4.7 十进制整数的加减运算**

- ✓ 十进制加法运算的校正

**4.8 逻辑运算与实现**

**4.9 运算器的基本组成与实例**

- ✓ 运算器的基本结构
- ✓ ALU举例

北京理工大学计算机学院

2025年10月