# 第6章 计算机的运算方法

> 课件来源：《计算机组成原理》课件 cod 版 Slide 259–370（共 112 页），唯一主来源。本章是全书最重点的大章、计算最密集：机器数互求、补码加减与溢出、原码/补码乘除竖式、浮点加减几乎必考大题；6.5 进位链常考填空与简答。

## 0. 本章考点清单

| 考点 | 常见题型 | 重要度 |
| --- | --- | --- |
| 原码一位乘、补码一位乘（Booth）竖式计算 | 计算大题 | ★★★ |
| 原码除法（恢复余数/加减交替）、补码除法竖式计算 | 计算大题 | ★★★ |
| 补码加减运算与溢出判断（单符号位、双符号位） | 计算/填空 | ★★★ |
| 浮点加减五步：对阶→尾数求和→规格化→舍入→溢出判断 | 计算大题 | ★★★ |
| 原/反/补/移码定义、与真值互求、[−y]补 求法 | 填空/计算 | ★★★ |
| 浮点格式、表示范围、规格化判断、机器零 | 填空/简答 | ★★ |
| 算术移位添补规则与移位后真值 | 填空/计算 | ★★ |
| 原码两位乘（Cj 标志、判断表） | 计算 | ★★ |
| 串行/并行进位链、单重/双重分组跳跃进位及时间 | 填空/简答 | ★★ |
| IEEE 754 三种格式与隐含位 | 填空 | ★ |
| ALU（74181）与并行加法器结构 | 简答/填空 | ★ |

## 1. 无符号数、机器数与真值（6.1）

**概念精讲**：无符号数没有符号位，寄存器位数直接决定表示范围：8 位 0～255，16 位 0～65535（课件 p.260）。有符号数把符号数字化：0 表示正、1 表示负，与数值部分一起构成**机器数**；带 "+/−" 的实际值称为**真值**（课件 p.261）。

| 真值 | 机器数 | 说明 |
|---|---|---|
| +0.1011 | 0 1011 | 小数点约定在符号位后 |
| +1100 | 0 1100 | 小数点约定在末尾 |
| −1100 | 1 1100 | 符号数字化 |
| −0.1011 | 1 1011 | 小数点约定在符号位后 |

**要点**：小数点位置是"约定"的，不占存储位；定点机靠约定区分小数/整数（见 §6）。 **易错警示**：机器数 ≠ 真值。同一个代码 10000000 按无符号、原、补、反解释得到 128、−0、−128、−127 四个不同真值（课件 p.280）。

## 2. 原码表示法

**定义**（课件 p.262–263，n 为数值位数，符号位与数值用逗号/小数点隔开）：

$$
\begin{aligned}
&\text{整数：}[x]_{原} = 0,x && \text{当 } 2^n > x \geq 0\\
&\qquad\qquad 2^n - x && \text{当 } 0 \geq x > -2^n\\
&\text{小数：}[x]_{原} = x && \text{当 } 1 > x \geq 0\\
&\qquad\qquad 1 - x && \text{当 } 0 \geq x > -1
\end{aligned}
$$

即"符号位 + 绝对值"。例：$x=+1110 \rightarrow [x]_{原}=0,1110$；$x=-1110 \rightarrow [x]_{原}=2^4-(-1110)=10000+1110=1,1110$（符号 1、数值 1110）；$x=+0.1101 \rightarrow 0.1101$；$x=-0.1101 \rightarrow 1-(-0.1101)=1.1101$（课件 p.262–263）。

**例题结论**（详见习题册题 1–4）：$[x]_{原}=1.0011 \rightarrow x=-0.0011$；$[x]_{原}=1,1100 \rightarrow x=-1100$；$[x]_{原}=0.1101 \rightarrow x=+0.1101$（课件 p.264–265）。

**0 的原码不唯一**：$[+0]_{原}=0.0000$（整数 0,0000），$[-0]_{原}=1.0000$（整数 1,0000），二者不等（课件 p.265）。

**原码的缺点**（课件 p.266）：做加法时必须先比较符号——同号相加、异号相减，结果符号还要单独判断，无法"只做加法"。于是引出"找一个与负数等价的正数"——补码。 **易错警示**：原码范围对称（$\pm(2^n-1)$ / $\pm(1-2^{-n})$），且有两个 0；原码不能直接送加法器做加减。

## 3. 补码表示法

**补（模）的概念**（课件 p.267–269）：时钟以 12 为模，倒拨 3 小时等价于顺拨 9 小时：$-3 \equiv +9 \pmod{12}$；计数器以 16 为模：$-1011 \equiv +0101 \pmod{2^4}$；小数以 2 为模：$-0.1001 \equiv +1.0111 \pmod{2}$。结论： - 一个负数加上"模"即得该负数的补数； - 互为补数的正负两数绝对值之和 = 模； - 正数的补数是其本身；两互为补数的数同加模后仍互为补数。

**定义**（课件 p.270–271）：

```
整数：[x]补 = 0,x              当 2ⁿ > x ≥ 0
            2ⁿ⁺¹ + x          当 0 > x ≥ -2ⁿ   (mod 2ⁿ⁺¹)
小数：[x]补 = x                当 1 > x ≥ 0
            2 + x             当 0 > x ≥ -1    (mod 2)
```

例：x=−1011000 → [x]补 = $2^8$ + (−1011000) = 1,0101000；x=−0.1100000 → [x]补 = 2 + (−0.1100000) = 1.0100000。

**快捷求法**（课件 p.272、274）：真值为负时，**补码 = 原码符号位不变、数值位每位取反、末位加 1**；反之由补码求原码用同一规则。例 x=−1010：[x]原=1,1010 → 取反 1,0101 → +1 → [x]补=1,0110。

**重要结论**（课件 p.275 练习）：[+0]补 = [−0]补 = 0.0000（0 唯一）；[−1]补 = 2 + (−1) = 1.0000，而 −1 的原码**不能表示**——补码比原码多表示一个最小负数。 **易错警示**：补码的符号位是"算出来的"（模运算结果），不是人为贴上去的；[−1]补=1.0000、整数 [$-2^n$]补=1,00…0 是常考空。

## 4. 反码表示法与三种机器数小结

**定义**（课件 p.276–277）：

```
整数：[x]反 = 0,x                 当 2ⁿ > x ≥ 0
            (2ⁿ⁺¹ − 1) + x        当 0 ≥ x > -2ⁿ  (mod 2ⁿ⁺¹ − 1)
小数：[x]反 = x                   当 1 > x ≥ 0
            (2 − 2⁻ⁿ) + x         当 0 ≥ x > -1   (mod 2 − 2⁻ⁿ)
```

即负数反码 = 原码符号位不变、数值位每位取反（**不加 1**）。例：x=−1101 → [x]反=1,0010；x=−0.1010 → [x]反=1.0101。0 的反码也不唯一：[+0]反=0.0000，[−0]反=1.1111（课件 p.278）。

**三种机器数小结**（课件 p.279）： - 正数：原码 = 补码 = 反码； - 负数：符号位均为 1；原码数值位取反 → 反码；取反末位加 1 → 补码； - 最高位为符号位，书写用 ","（整数）或 "."（小数）隔开。

**例 6.11 四种解释对照**（课件 p.280，8 位含 1 位符号位）：

**例6.11** 设机器数字长为 8 位（其中 1 位为符号位）6.1
对于整数，当其分别代表无符号数、原码、补码和反码时，对应的真值范围各为多少？

| 二进制代码 | 无符号数对应的真值 | 原码对应的真值 | 补码对应的真值 | 反码对应的真值 |
|---|---|---|---|---|
| 00000000 | 0 | +0 | ±0 | +0 |
| 00000001 | 1 | +1 | +1 | +1 |
| 00000010 | 2 | +2 | +2 | +2 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| 01111111 | 127 | +127 | +127 | +127 |
| 10000000 | 128 | -0 | -128 | -127 |
| 10000001 | 129 | -1 | -127 | -126 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| 11111101 | 253 | -125 | -3 | -2 |
| 11111110 | 254 | -126 | -2 | -1 |
| 11111111 | 255 | -127 | -1 | -0 |

| 码制 | 真值范围 | 10000000 的真值 | 11111111 的真值 |
|---|---|---|---|
| 无符号数 | 0 ~ 255 | 128 | 255 |
| 原码 | −127 ~ +127 | −0 | −127 |
| 补码 | −128 ~ +127 | −128 | −1 |
| 反码 | −127 ~ +127 | −127 | −0 |

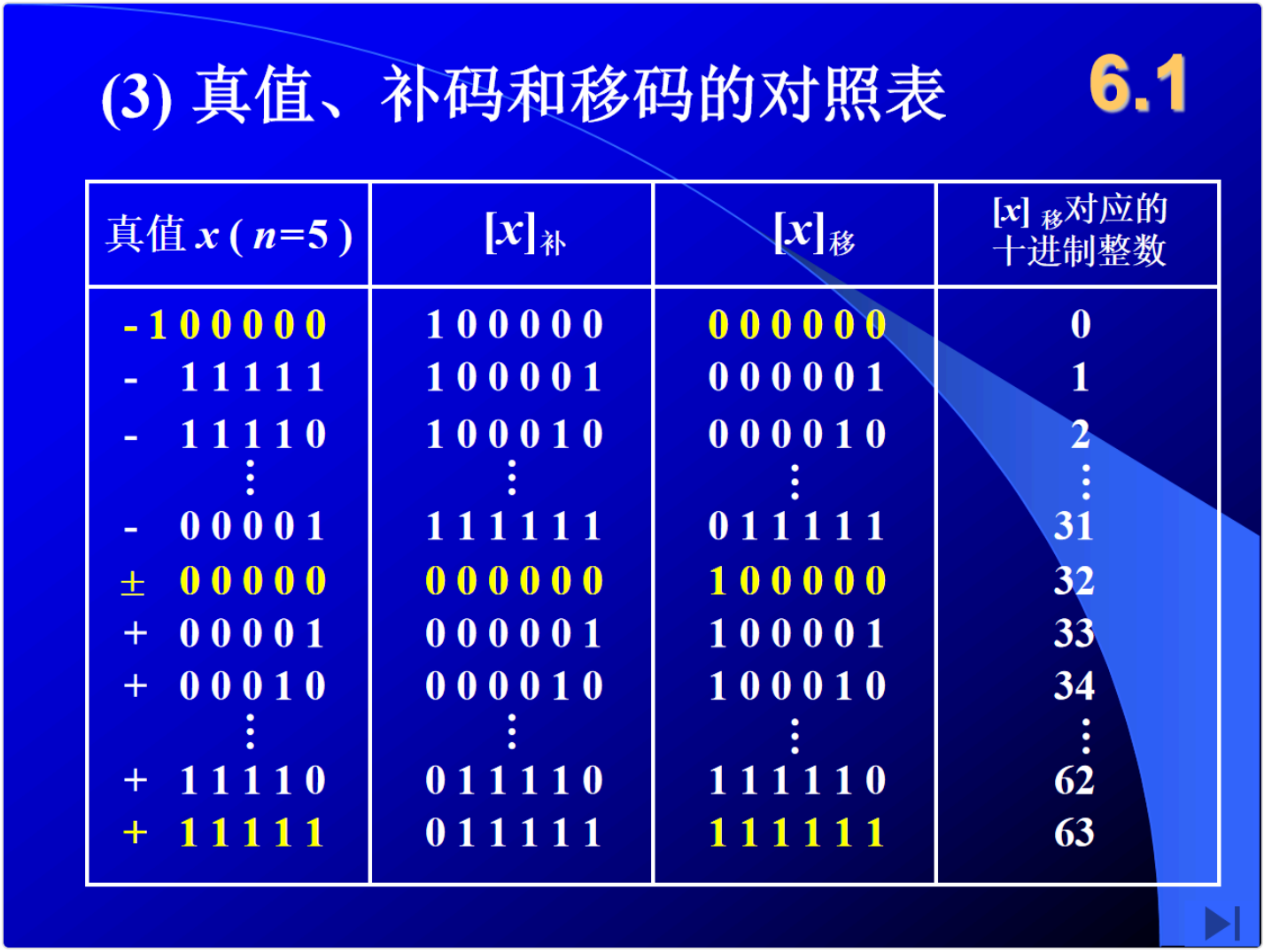
**例 6.12 求 [−y]补（"求补"操作）**（课件 p.281）：不论 [y]补 符号如何，**将 [y]补 连同符号位在内每位取反、末位加 1，即得 [−y]补**。这是补码减法硬件的核心操作。 **易错警示**：求 [−y]补 时符号位也要取反；而"原码↔补码"互求时符号位不变——两条规则别混。

## 5. 移码表示法

**动机**（课件 p.282）：补码不便直接比较真值大小（如 +21 的补码 0,10101 与 −21 的补码 1,01011，按代码比大小会判错）；加上偏置 $2^n$ 后代码大小与真值大小一致。

**定义**（课件 p.283，n 为整数数值位数）：[x]移 = $2^n + x$（$2^n > x \geq -2^n$）。例 n=5：x=+10100 → 1,10100；x=−10100 → 0,01100。

**与补码的关系**（课件 p.284）：**移码与补码仅符号位相反**，数值位完全相同。例 x=+1100100：[x]补=0,1100100，[x]移=1,1100100；x=−1100100：[x]补=1,0011100，[x]移=0,0011100。




**特点**（课件 p.285–286）： - 最小真值 $-2^n$ 的移码为全 0；[+0]移 = [−0]移 = 1,00…0（0 唯一）； - 移码符号位 1 表示正、0 表示负（与补码相反）； - **移码只用于整数，典型用途是浮点数的阶码**：便于比较阶码大小、便于判 0。

**易错警示**：移码没有小数形式；"移码 = 补码符号位取反"只对同一真值、同一位数成立。

## 6. 数的定点表示（6.2）

小数点按约定固定（课件 p.287）：**小数定点机**小数点在符号位之后（Sf . S1S2…Sn）；**整数定点机**小数点在末尾。定点机中可用原码、补码、反码，范围如下（n 为数值位数）：

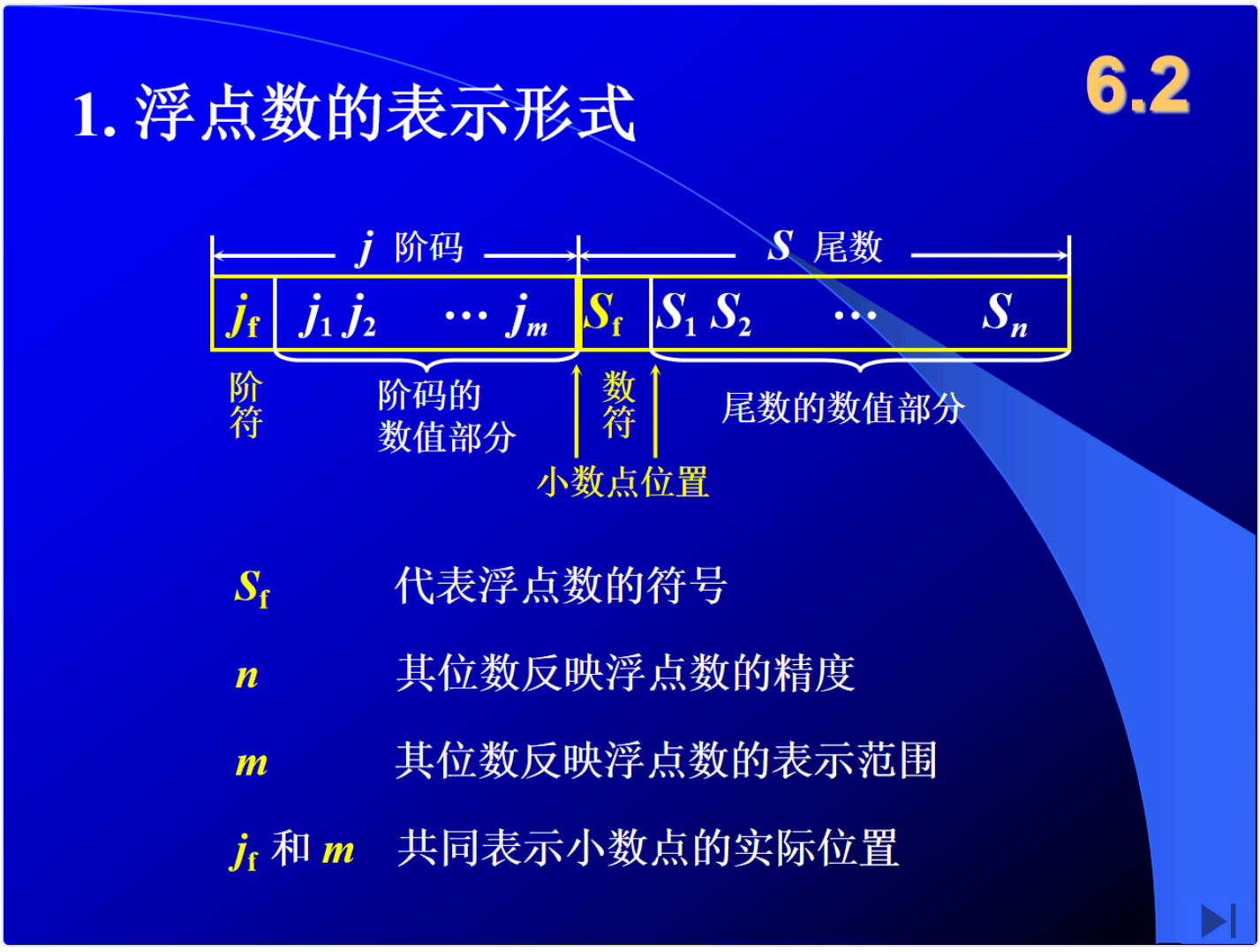
| 码制 | 小数定点机范围 | 整数定点机范围 |
| --- | --- | --- |
| 原码 | $-(1-2^{-n}) \sim +(1-2^{-n})$ | $-(2^n-1) \sim +(2^n-1)$ |
| 反码 | $-(1-2^{-n}) \sim +(1-2^{-n})$ | $-(2^n-1) \sim +(2^n-1)$ |
| 补码 | $-1 \sim +(1-2^{-n})$ | $-2^n \sim +(2^n-1)$ |

**易错警示**：补码小数能取到 −1 而原/反不能；做题先看清"含几位符号位"。

## 7. 数的浮点表示

**一般形式**（课件 p.288）：N = S × r^j。S 尾数（小数、可正可负），j 阶码（整数、可正可负），r 基数（计算机中取 2、4、8、16）。同一数可有多种浮点写法：$11.0101 = 0.110101\times2^{10} = 1.10101\times2^{1} = 1101.01\times2^{-10} = 0.00110101\times2^{100}$（指数为二进制）。其中**规格化数**要求尾数最高位为 1（r=2 时）。

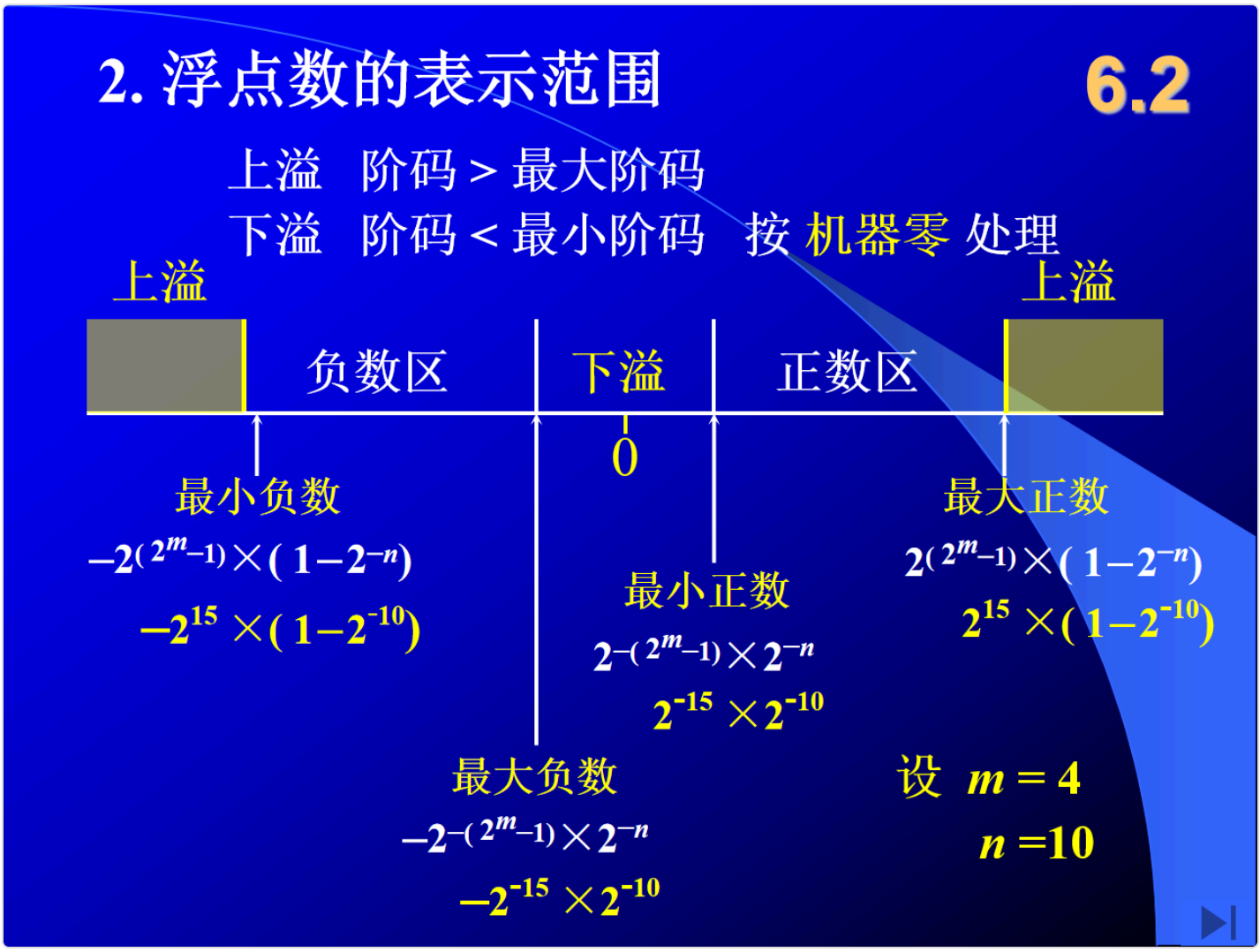
**格式**（课件 p.289）：




- 阶符 jf + 阶码数值 m 位：m 反映**表示范围**；jf 与 m 共同决定小数点实际位置；

- 数符 Sf + 尾数数值 n 位：n 反映**精度**；Sf 代表浮点数符号。

**表示范围**（课件 p.290，阶码、尾数均补码、未规格化）：




| 边界 | 公式 | m=4, n=10 时 |
|---|---|---|
| 最大正数 | 2^(2^m−1) × (1−2^−n) | $2^{15}\times(1-2^{-10})$ |
| 最小正数 | 2^−(2^m−1) × 2^−n | $2^{-15}\times 2^{-10}$ |
| 最大负数 | −2^−(2^m−1) × 2^−n | $-2^{-15}\times 2^{-10}$ |
| 最小负数 | −2^(2^m−1) × (1−2^−n) | $-2^{15}\times(1-2^{-10})$ |

**上溢**：阶码 > 最大阶码（报错/中断）；**下溢**：阶码 < 最小阶码，按**机器零**处理。 **位数分配练习**（课件 p.291）：24 位字长表示 ±3 万：$2^{14}=16384 < 30000 < 32768=2^{15}$，规格化后需 15 位二进制数值 → 阶码数值 m=4（$2^4-1=15$ 够用），尾数 n = 24−2−4 = **18**（保证最大精度）。

**规格化**（课件 p.292–293）： - r=2：尾数最高位为 1；r=4：最高 2 位不全 0；r=8：最高 3 位不全 0； - **左规**：尾数左移 1 位（r=2）、阶码减 1；**右规**：尾数右移 1 位、阶码加 1（r=4/8 时每次移 2/3 位）； - 基数 r 越大：

范围越大、精度越低； - 规格化后（m=4, n=10）：最小正数 = $2^{-15}\times2^{-1}=2^{-16}$，最大负数 = $-2^{-16}$（课件 p.293）。

**机器零**（课件 p.297）：① 尾数为 0，不论阶码为何值 → 机器零；② 阶码 ≤ 所能表示的最小阶码，不论尾数为何值 → 机器零。 - 阶码、尾数都用补码：机器零为 ×,×××; 0.00…0，或 1,0000; ×.××…×（阶码 = −16）； - **阶码移码、尾数补码：机器零为 0,0000; 0.00…0（全 0）**，最有利于"判 0"电路实现。

**例 6.13–6.15 结论**（完整步骤见习题册题 15–17）：+19/128 = 0.0010011 = $0.1001100000\times2^{-10}$；−58 = −111010 = $-0.1110100000\times2^{110}$，其"阶移尾补"形式为 1,0110; 1.0001100000（课件 p.294–295）。 **易错警示**：浮点数的符号由数符决定，与阶码符号无关；"阶码大"不代表"数大"还要看尾数，但规格化后阶码决定数量级。

## 8. IEEE 754 标准

课件 p.298 给出三种格式（尾数均为规格化表示，非 0 有效位最高位为 1 且**隐含**不存）：

| 格式 | 符号位 S | 阶码（含阶符） | 尾数 | 总位数 |
|---|---|---|---|---|
| 短实数 | 1 | 8 | 23 | 32 |
| 长实数 | 1 | 11 | 52 | 64 |
| 临时实数 | 1 | 15 | 64 | 80 |

（补充常识）短实数阶码偏置值为 127、长实数为 1023，即阶码字段 = 真值阶 + 偏置；课件未列偏置数值，若考转换按此常识处理。 **易错警示**：23 位尾数字段因隐含 1 实际提供 24 位精度；IEEE 754 的"规格化"是 1.xxx 形式，与本章 0.1xxx 形式差一次左规/右规。

## 9. 移位运算（6.3）

**意义**（课件 p.299）：左移绝对值扩大（×2）、右移绝对值缩小（÷2）；移位与加减配合即可实现乘除。

**算术移位添补规则**（课件 p.300，符号位不变）：

| 码制 | 真值 | 左移添 | 右移添 |
|---|---|---|---|
| 原/补/反 | 正数 | 0 | 0 |
| 原码 | 负数 | 0 | 0 |
| 补码 | 负数 | 0 | 1 |
| 反码 | 负数 | 1 | 1 |

**例 6.16/6.17**（课件 p.301–303，8 位含 1 符号位）：+26 = 0,0011010，左移 1/2 位得 +52/+104，右移 1/2 位得 +13/+6（丢 1 影响精度）。−26：原码 1,0011010、补码 1,1100110、反码 1,1100101；左移 1/2 位三码

均得 −52/−104；右移 1 位均得 −13；**右移 2 位：补码得 1,1111001 = −7，反码得 1,1111001 = −6**（同一代码不同码制真值不同；补码右移向 −∞ 舍入）。

**硬件实现**（课件 p.304）：左移丢 1 → **出错**（溢出性质）；右移丢 1 → **影响精度**。 **算术 vs 逻辑移位**（课件 p.305）：算术移位针对有符号数（按上表添补）；逻辑移位针对无符号数，左移低位添 0 高位移丢、右移高位添 0 低位移丢。课件例：01010011 逻辑左移 = 10100110，算术左移 = 00100110（符号位不变、数值最高位 1 移丢入 Cy，出错）；10110010 逻辑右移 = 01011001，算术右移 = 11011001（补码添 1）。 **易错警示**：负数补码右移添 1 不是添 0；"左移一位=×2"只在未丢 1 时成立。

## 10. 补码加减法与溢出判断

**公式**（课件 p.306）：

```
加法：[A+B]补 = [A]补 + [B]补      (mod 2ⁿ⁺¹ 整数 / mod 2 小数)
减法：[A−B]补 = [A]补 + [−B]补     (mod 2ⁿ⁺¹ / mod 2)
```

**连同符号位一起相加，符号位产生的进位自然丢掉**。[−B]补 用"连同符号位取反加 1"求得。例 6.18–6.20 见习题册题 20–22。

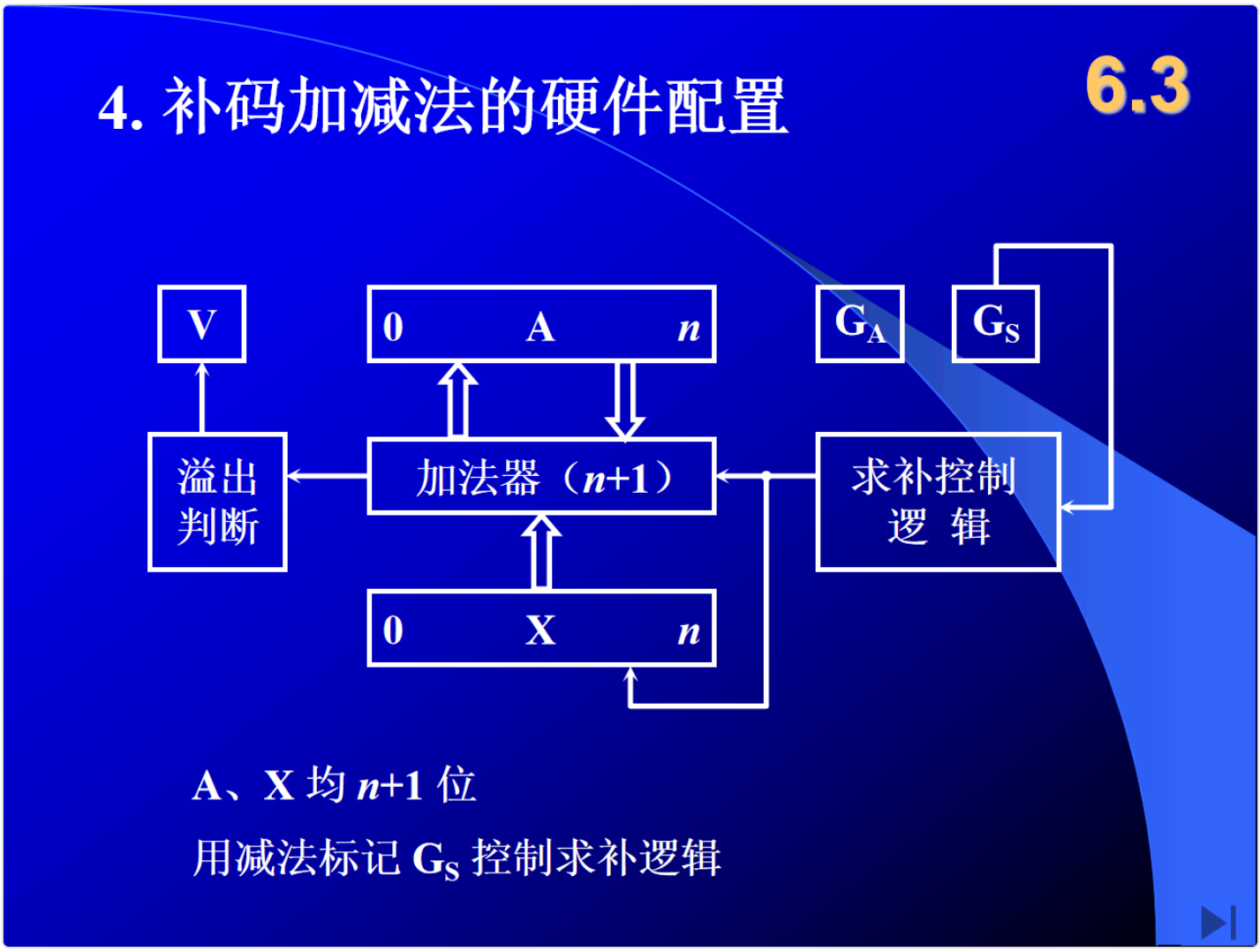
**溢出判断①：一位符号位**（课件 p.309）：两操作数符号相同（减法指被减数与求补后的减数），结果符号与之不同 → 溢出。硬件：**V = 符号位进位 ⊕ 最高数值位进位**（1⊕0、0⊕1 溢出；0⊕0、1⊕1 不溢出）。

**溢出判断②：两位符号位（变形补码，mod 4）**（课件 p.310）：

```
[x]补' = x        1 > x ≥ 0
         4 + x    0 > x ≥ −1   (mod 4)
```

结果双符号位 00（正）/11（负）→ 未溢出；**01 → 正溢（上溢），10 → 负溢（下溢）**；最高符号位代表真正符号。

**硬件配置**（课件 p.311）：




A、X 均 n+1 位；加法器 n+1 位；减法标记 GS 控制求补逻辑（对 X 求补后相加）；溢出判断输出 V。 **易错警示**：符号位进位丢弃 ≠ 溢出；溢出看"进位异或"或"双符号位是否一致"。正+正得负、负+负得正必溢出（课件 p.308 练习 1、2 的"错"即此）。

## 11. 原码乘法（一位乘、两位乘）

**笔算分析**（课件 p.312–315）：符号单独处理；乘数每一位决定是否加被乘数；n 个位积相加、积位数翻倍。改进：把"位积一起加"改为**加一次、右移一次**的递推：部分积只与被乘数的高位相加，乘数右移腾出高位存放部分积低位。硬件：3 个带移位功能的寄存器 + 1 个全加器；n=4 时加 4 次、移 4 次。

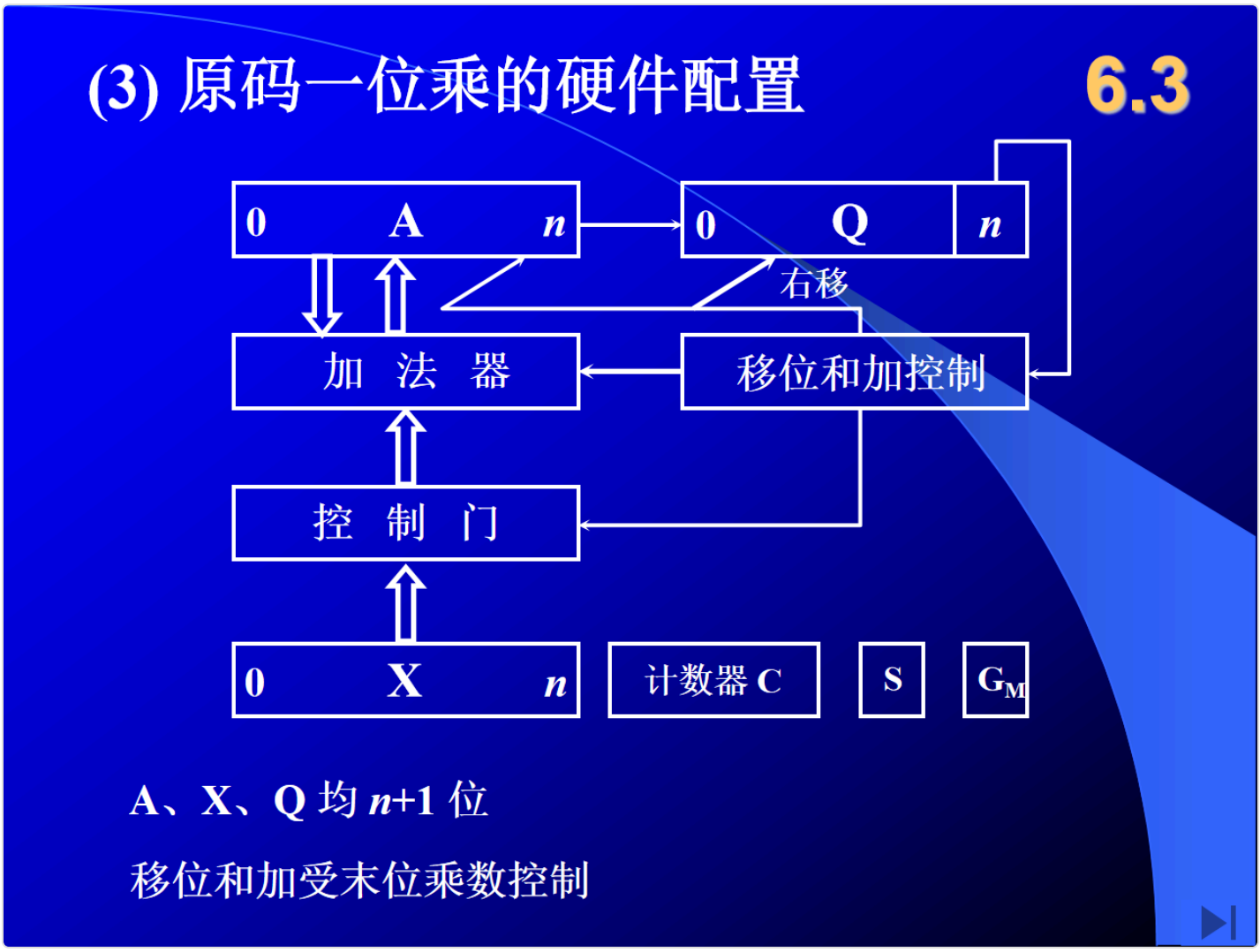
**一位乘规则**（课件 p.316–317）：$[x\cdot y]_{原} = (x_0 \oplus y_0).(x\cdot y)$，x、y 为绝对值。递推：$z_0=0$；$z_1=2^{-1}(y_n x+z_0)$；$z_2=2^{-1}(y_{n-1}x+z_1)$；…；$z_n=2^{-1}(y_1 x^*+z_{n-1})$。即**乘数末位为 1 加 x**为 0 加 0，然后逻辑右移 1 位**，共 n 次。

**例 6.21 竖式**（x=−0.1110, y=0.1101，课件 p.318–319）：




结果 x·*y* = 0.10110110，符号 1⊕0=1 → [x·y]原 = 1.10110110。特点：绝对值运算、**逻辑移位**、用移位次数判断结束。

**硬件**（课件 p.320）：




**两位乘**（课件 p.321–322）：每次看乘数 2 位 $y_{n-1}y_n$ 加标志 Cj。难点是 3x：*用 3x = 4x − x*** 化解——本步 −x、*置 Cj=1*，*下一步（高 2 位处）+x* 偿还 4x*。判断表：

| $y_{n-1}y_n$ | Cj | 操作 |
|---|---|---|
| 00 | 0 | +0，右移 2 位，Cj 保持 0 |
| 01 | 0 | +x*，右移 2 位，Cj 保持 0 |
| 10 | 0 | +2x*，右移 2 位，Cj 保持 0 |
| 11 | 0 | −x*，右移 2 位，Cj 置 1 |
| 00 | 1 | +x*，右移 2 位，Cj 置 0 |
| 01 | 1 | +2x*，右移 2 位，Cj 置 0 |
| 10 | 1 | −x*，右移 2 位，Cj 保持 1 |
| 11 | 1 | +0，右移 2 位，Cj 保持 1 |

实际操作为 +[x]补、+[2x]补、+[−x]补，**算术（补码）右移 2 位**；部分积取 3 位符号位防溢出。例 6.22（x=0.111111, y=−0.111001）：3 次右移后 Cj=1，最后再 +x **不移位**校正，得 [x·y]原 = 1.111000000111（课件 p.323–324，竖式见习题册题 26）。

**一位乘 vs 两位乘**（课件 p.325）：符号均 $x_0 \oplus y_0$；操作数分别为绝对值 / 绝对值的补码；逻辑右移 / 算术右移；移位 n 次 / n/2 次（n 偶）；最多加 n 次 / n/2+1 次（n 偶）。**思考：n 为奇数时移 (n+1)/2 次、最多加 (n+1)/2+1 次。** **易错警示**：两位乘的"−x"是加 [−x]补；最后 Cj=1 的校正加**不移位**；两位乘移位次数按"组"数算。

## 12. 补码乘法（Booth 算法）

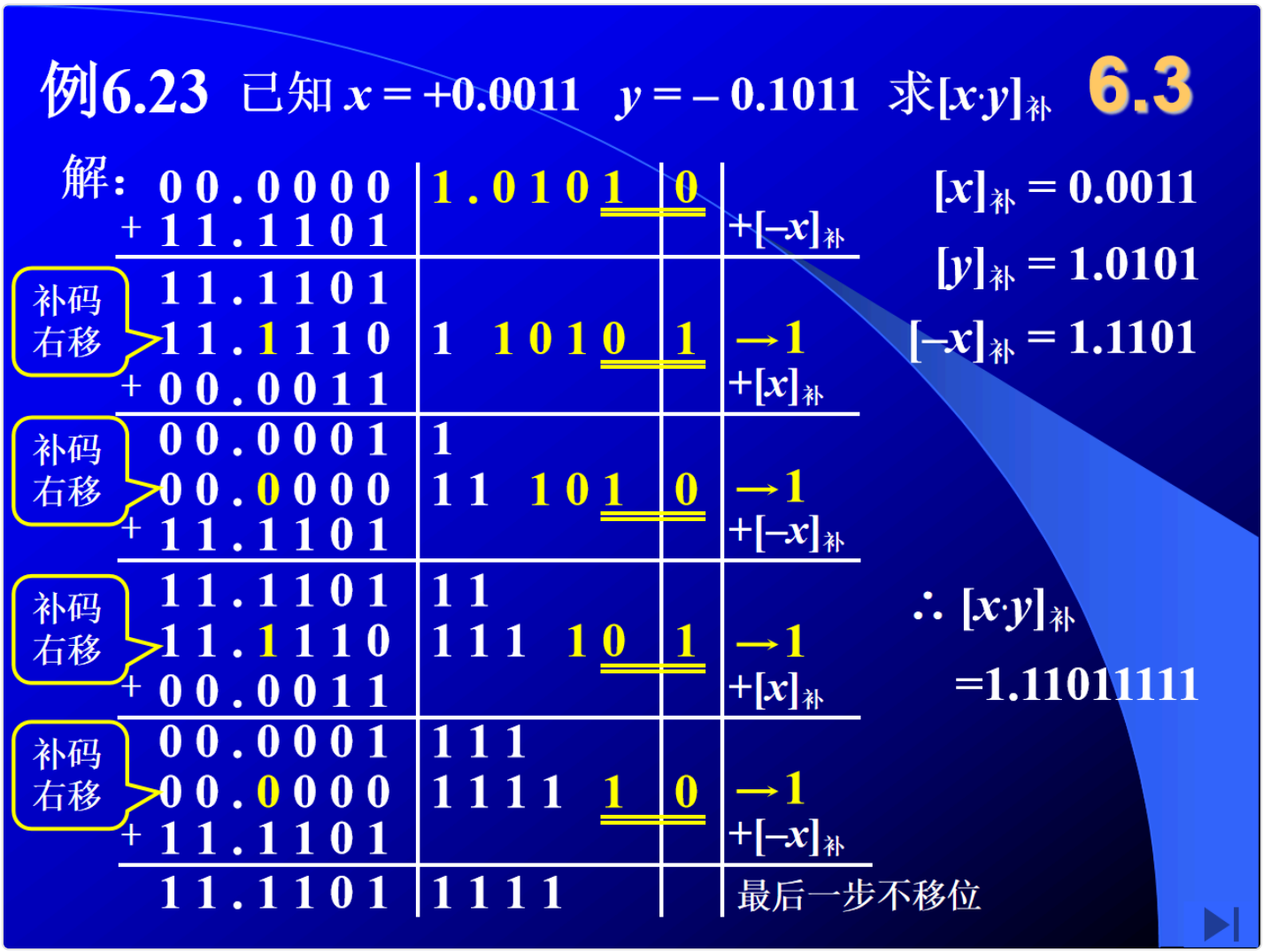
**规则铺垫**（课件 p.326）：乘数为正时同原码乘但按补码规则加/移，符号自然形成；乘数为负时去掉符号位照做、最后 +[−x]补 校正。Booth 把两者统一。

**Booth 推导结论**（课件 p.327–328）：$[x \cdot y]_{补} = [x]_{补} \times [(y_1 - y_0) + (y_2 - y_1)2^{-1} + \dots + (y_{n+1} - y_n)2^{-n}]$，附加位 $y_{n+1}=0$。递推：$[z_0]_{补}=0$；每步按 $y_i y_{i+1}$ 判断：

| $y_i\ y_{i+1}$ | $y_{i+1} - y_i$ | 操作 |
|---|---|---|
| 00 | 0 | +0，右移 1 位 |
| 01 | +1 | +[x]补，右移 1 位 |
| 10 | −1 | +[−x]补，右移 1 位 |
| 11 | 0 | +0，右移 1 位 |

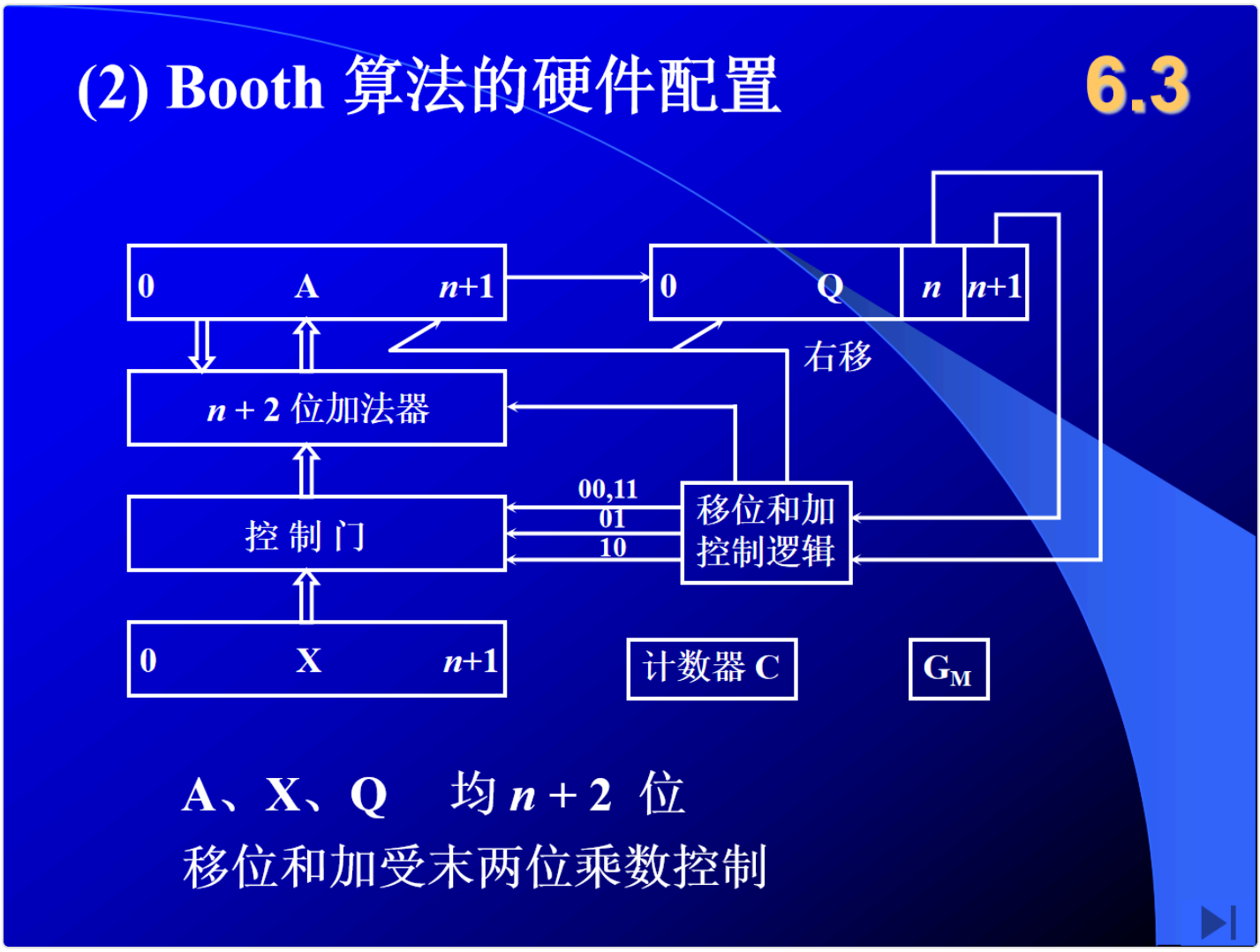
共 n+1 步，**最后一步（$y_0 y_1$）只加不移位**；移位为补码（算术）右移。

**例 6.23 竖式**（x=+0.0011, y=−0.1011，课件 p.329）：




[x]补=0.0011、[y]补=1.0101、[−x]补=1.1101；五步操作依次为 +[−x]补、+[x]补、+[−x]补、+[x]补、+[−x]补（末步不移位），得 [x·y]补 = 1.11011111（真值 −33/256）。

**硬件**（课件 p.330）：




**乘法小结**（课件 p.331）：原码乘符号单独处理、补码乘符号自然形成；原码乘去掉符号位即无符号数乘法；整数乘与小数乘完全相同（逗号代小数点）。 **易错警示**：Booth 判断位是"当前末位 + 附加位"，附加位初值 0 且随移位更新；步数 = n+1 而非 n。

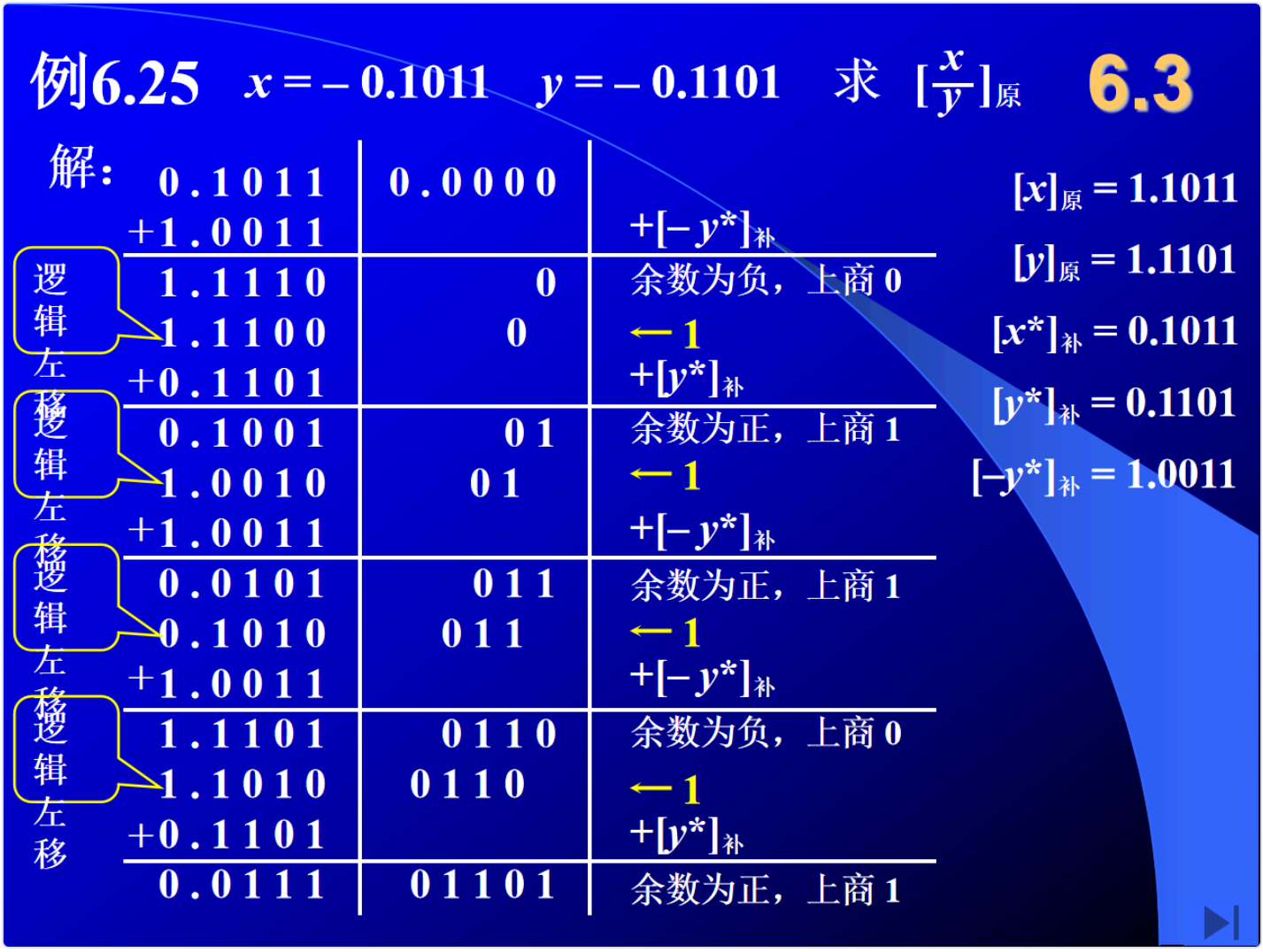
## 13. 原码除法（恢复余数法、加减交替法）

**笔算 vs 机器**（课件 p.332–333）：机器除法改为**余数左移 1 位、减除数**（1 倍字长加法器），在寄存器最末位上商；商符 $x_0 \oplus y_0$ 单独形成。 **约定**（课件 p.334）：除数 ≠ 0；小数定点除要求 $|x| < |y|$（整数除要求 $|x| > |y|$），否则商溢出；数值部分按绝对值相除。

**恢复余数法**（课件 p.335–336，例 6.24）：余数为正上商 1；余数为负上商 0 并*加 $y$ 恢复余数，**再左移。上商 n+1 次（**第一次上商判溢出**：若为 1 说明 $|x| \geq |y|$ 溢出），左移 n 次。

**加减交替法（不恢复余数）**（课件 p.337）：余数 $R_i > 0$ 上商 1，下一步 $2R_i - y$；*$R_i < 0$ 上商 0，下一步 $2R_i + y$*（把"恢复+左移+减"合并为"左移+加"）。

**例 6.25 竖式**（$x = -0.1011$, $y = -0.1101$，课件 p.338–339）：

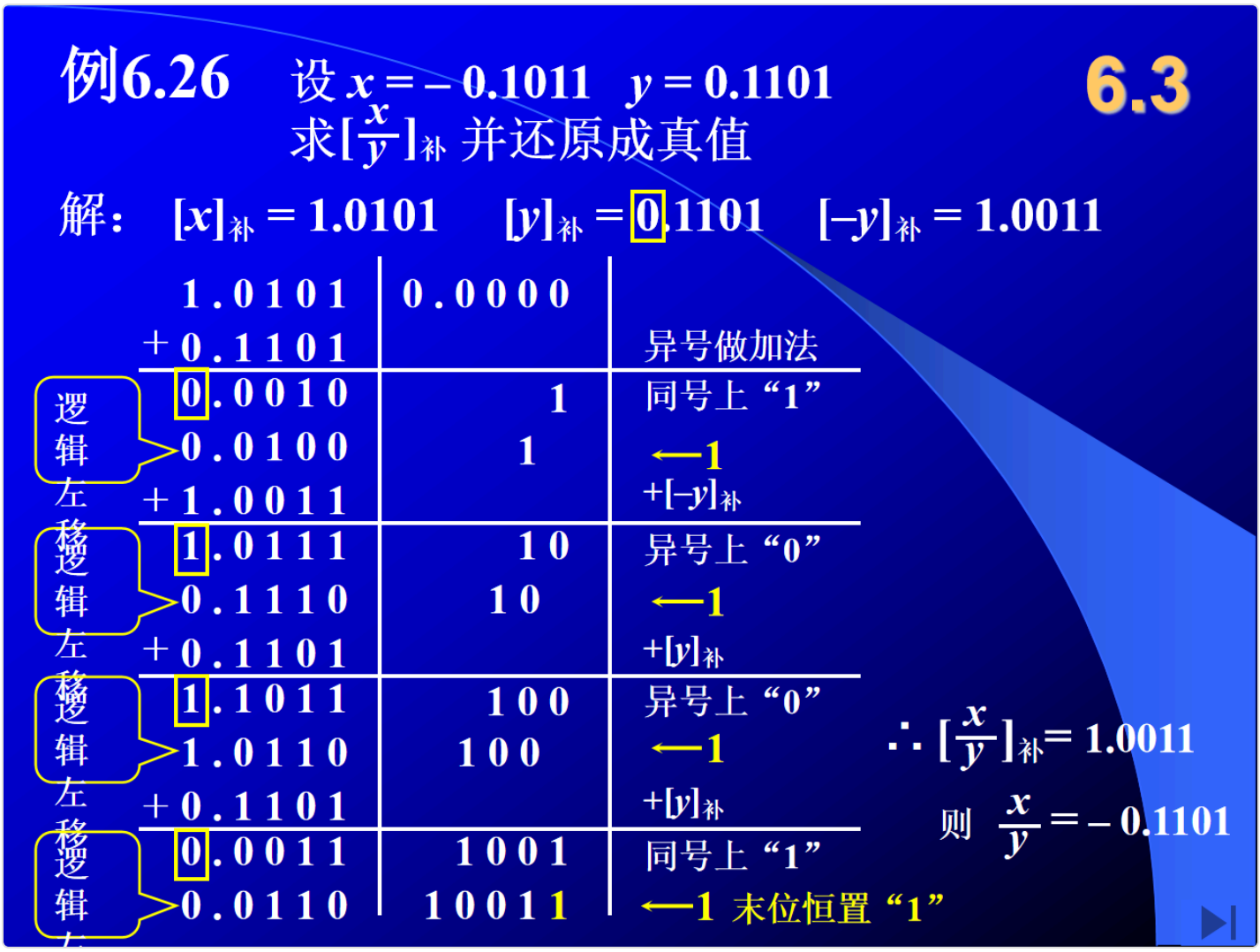



商 0.1101、余数 $0.0111\times2^{-4}$；商符 $1\oplus1=0$ → [x/y]原 = 0.1101。特点：移 n 次、加 n+1 次、上商 n+1 次，用移位次数判结束。 **硬件**（课件 p.340）：A、X、Q 均 n+1 位，n+1 位加法器，**$Q_n$（商末位）控制加减交替**，左移。 **易错警示**：加减交替法最后一步若余数为负，仍需恢复余数才是正确余数（课件例题末余数为正故未体现）；上商顺序是"先判旧余数符号上商，再算新余数"。

## 14. 补码除法（加减交替法）

**商值确定**（课件 p.341–343）：操作数直接用补码。第一步：[x]补与[y]补**同号做减法、异号做加法**得 $R_0$。之后统一规则：**$[R_i]$补与[y]补同号上商 1，异号上商 0**（正商按原码上商、负商按反码上商，二者化简后一致）。**末位恒置 1。** **商符**（课件 p.344）：第一次上商即商符，自然形成；小数除法第一次"不够减"：正商上 0、负商上 1；**第一次商也可判溢出**（与约定相反即溢出）。 **新余数**（课件 p.345）：上商 1 → $2[R_i]$补 + [–y]补；上商 0 → $2[R_i]$补 + [y]补；余数逻辑左移。

**例 6.26 竖式**（x=–0.1011, y=+0.1101，课件 p.346）：

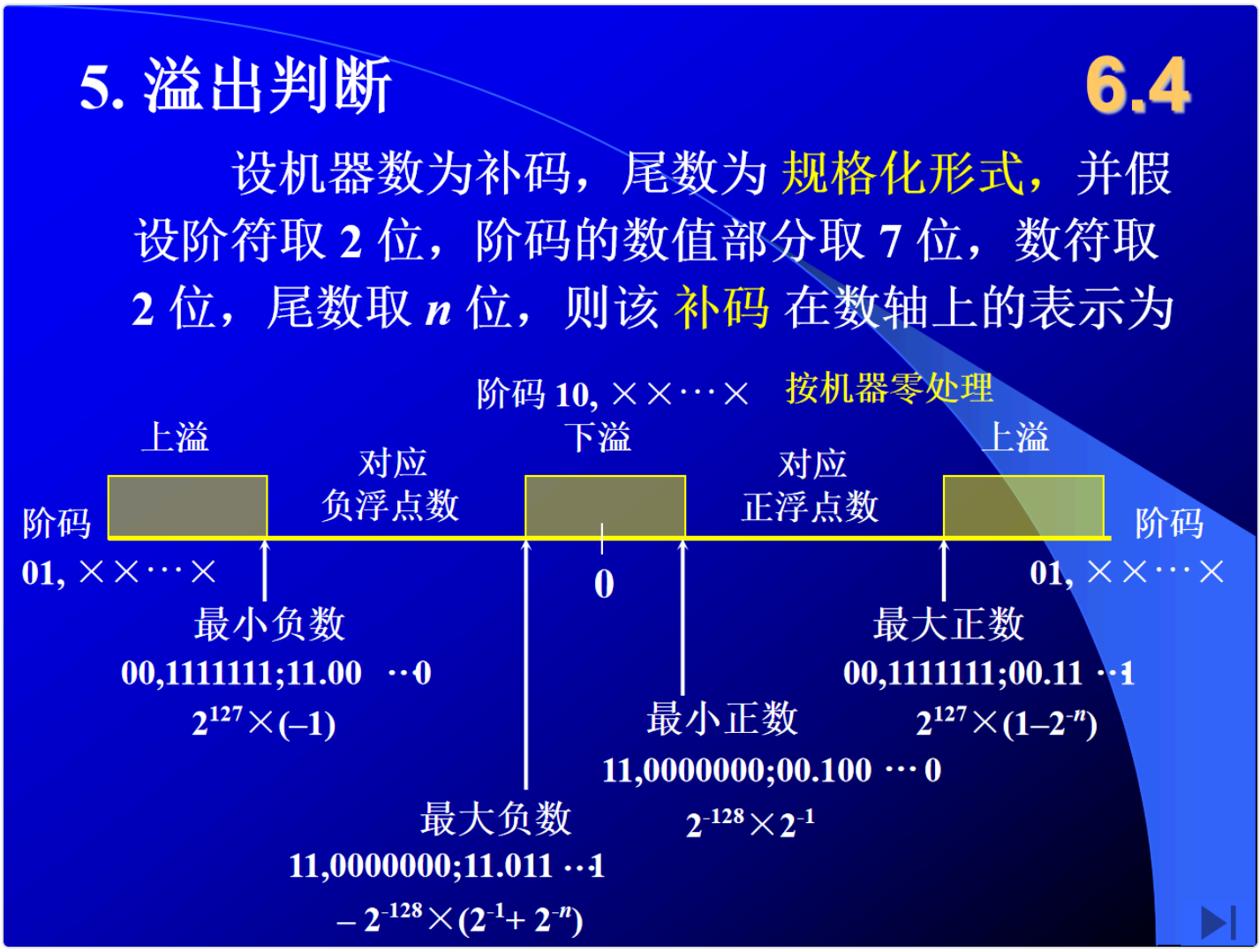



[x]补=1.0101、[y]补=0.1101、[−y]补=1.0011；商序列 1,0,0,1,1（末位恒置 1）→ [x/y]补 = 1.0011 → 真值 −0.1101。 **小结**（课件 p.347）：上商 n+1 次（末位恒置 1）、加 n 次、移 n 次；精度误差最大 $2^{-n}$。 **与原码除比较**（课件 p.348）：商符 $x_0 \oplus y_0$ / 自然形成；操作数 绝对值补码 / 补码；上商原则 余数正负 / 余数与除数符号比较；上商均 n+1 次；加法 n+1 / n 次；移位均逻辑左移 n 次；第一步 [x]*补*−[y]补 / 同号减异号加。 **易错警示**：补码除"够减/不够减"看余数与除数**符号关系**而非余数正负；末位恒置 1 是约定不是算出来的。

## 15. 浮点加减运算（6.4）

五步：**对阶 → 尾数求和 → 规格化 → 舍入 → 溢出判断**。

**① 对阶**（课件 p.349）：Δj = jx − jy；**小阶向大阶看齐**：小阶的尾数右移 |Δj| 位、阶码加 |Δj|（右移只丢低位，误差小且可舍入补救；左移丢高位会严重出错）。 **② 尾数求和**：按定点补码加减（双符号位）。 **③ 规格化**（课件 p.351–353）：r=2 规格化定义 1/2 ≤ |S| < 1。判断：**原码不论正负第一数位为 1；补码符号位与第一数位不同。** 特例：[−1/2]补 = 1.100…0 **不是**规格化数；[−1]补 = 1.000…0 **是**规格化数。**左规**：尾数左移 1 位、阶码减 1，直到数符与第一数位不同；**右规**：尾数出现 01.××… 或 10.××…（尾数溢出）时右移 1 位、阶码加 1。 **④ 舍入**（课件 p.355）：对阶/右移丢低位时用 **0 舍 1 入法**或**恒置 1 法**。 **⑤ 溢出判断**（课件 p.358）：




**浮点溢出由阶码决定**：阶码双符号位 01 → 上溢（报错）；10 → 下溢按机器零。尾数溢出只需右规，不算真正溢出。设阶符 2 位、阶码数值 7 位、数符 2 位、尾数 n 位：最大正数 $2^{127}\times(1-2^{-n})$（00,1111111;00.11…1），最小正数 $2^{-128}\times2^{-1}$（11,0000000;00.100…0），最大负数 $-2^{-128}\times(2^{-1}+2^{-n})$（11,0000000;11.011…1），最小负数 $2^{127}\times(-1)$（00,1111111;11.00…0）。

**例题结论**（完整步骤见习题册题 32–34）：p.350 例 x+y 对阶后 11.1001，左规得 00,10;11.0010 = $-0.1110\times2^{10}$；例 6.27 尾数和 01.001010 右规得 00,011;00.100101 = $0.100101\times2^{11}$；例 6.28 x−y 右规得 11,101;11.011010 = $-0.100110\times2^{-11}$ = $(-19/32)\times2^{-3}$。 **易错警示**：对阶永远"小阶向大阶"；左规可能多次、右规最多一次；规格化判断对补码看"符号位⊕第一数位=1"。

## 16. 浮点乘除运算

课件 p.359：x·y = (Sx·Sy) × 2^(jx+jy)；x/y = (Sx/Sy) × 2^(jx−jy)。 - 阶码用**补码定点加（乘）/减（除）**； - 尾数乘除同定点乘除； - 结果**规格化**； - 浮点运算部件 = **阶码运算部件 + 尾数运算部件**。 **易错警示**：浮点乘除不需要对阶；阶码相加后可能溢出，需按阶码溢出处理。

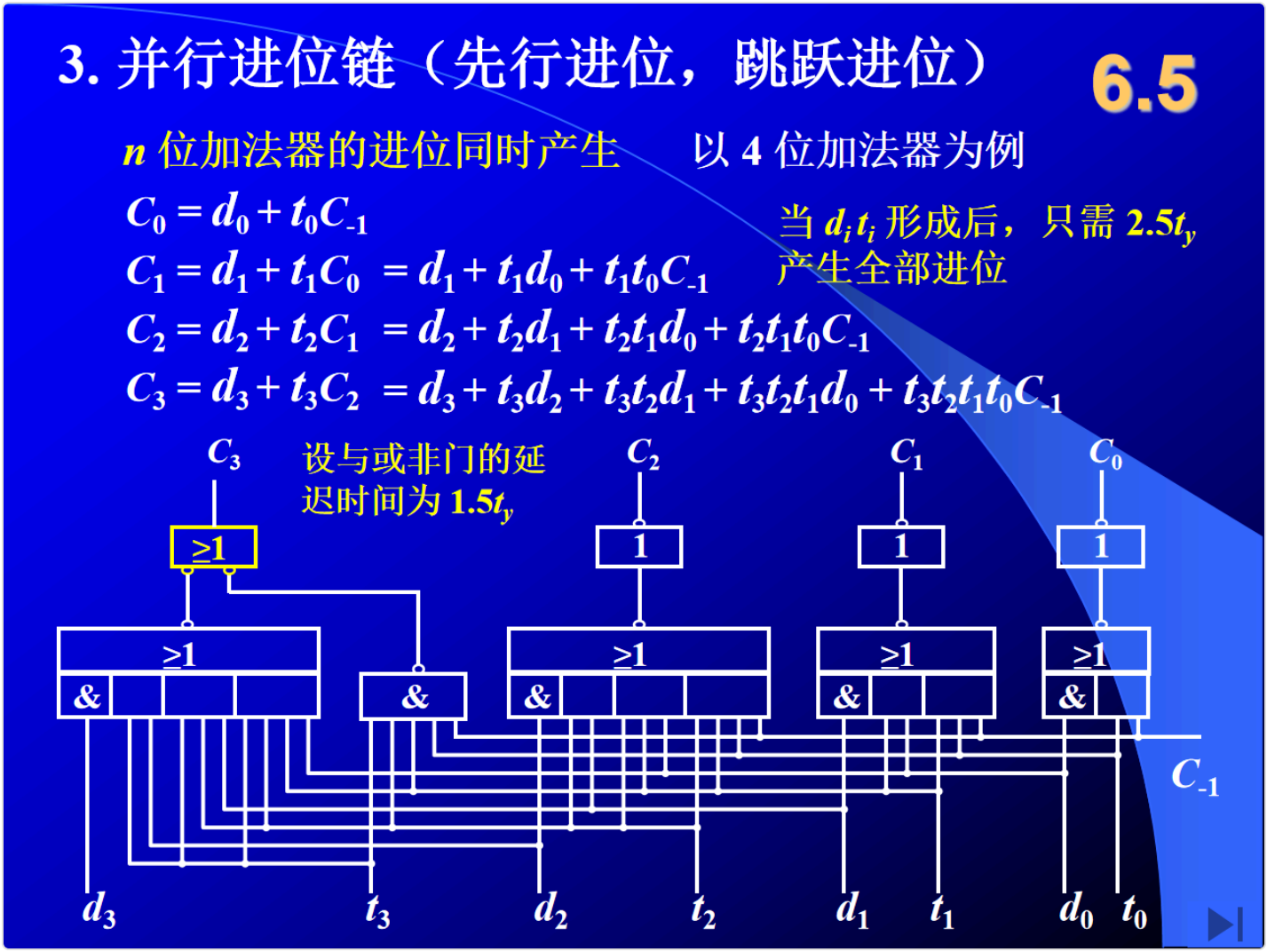
## 17. 算术逻辑单元 ALU 与快速进位链（6.5）

**ALU 电路**（课件 p.360）：组合逻辑电路；核心是多位全加器加控制门 Ki；典型芯片**四位 ALU 74181**：**M=0 算术运算、M=1 逻辑运算**，S3～S0 不同取值选择不同运算。

**并行加法器**（课件 p.361）：各位全加器同时相加，但进位仍可能串行。定义 **$d_i = A_iB_i$（本地进位/生成）**、**$t_i = A_i+B_i$（传送条件）**，则 $C_i = d_i + t_iC_{i-1}$；$S_i = A_i \oplus B_i \oplus C_{i-1}$。

**串行进位链**（课件 p.362）：$C_0=d_0+t_0C_{-1}$，$C_1=d_1+t_1C_0$，$C_2=d_2+t_2C_1$，$C_3=d_3+t_3C_2$，逐级传递。设与非门级延迟 ty：每位进位 2ty，**4 位全加器产生全部进位 8ty，n 位 2n•ty**。

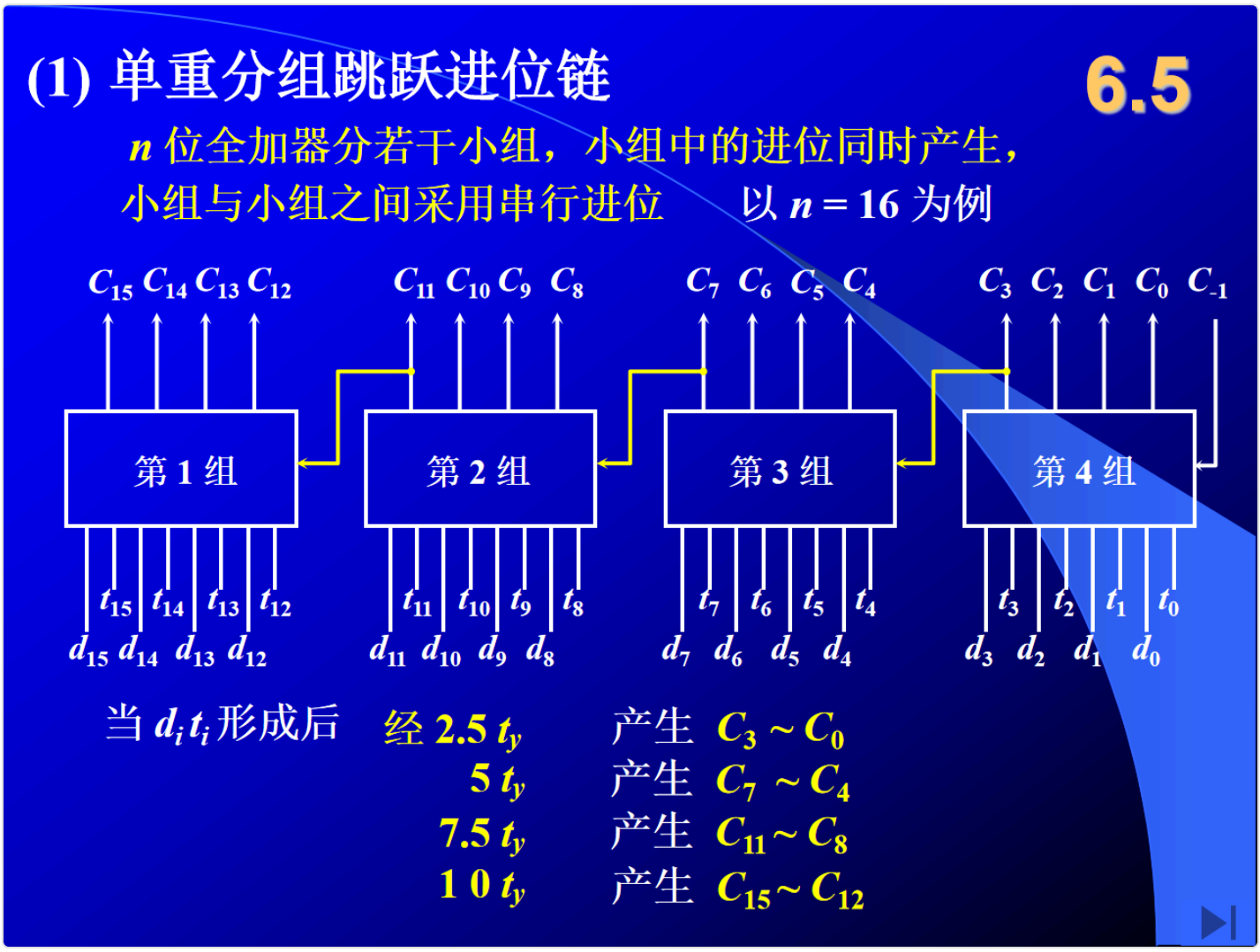
**并行进位链（先行进位、跳跃进位）**（课件 p.363）：




```
C0 = d0 + t0C-1
C1 = d1 + t1d0 + t1t0C-1
C2 = d2 + t2d1 + t2t1d0 + t2t1t0C-1
C3 = d3 + t3d2 + t3t2d1 + t3t2t1d0 + t3t2t1t0C-1
```

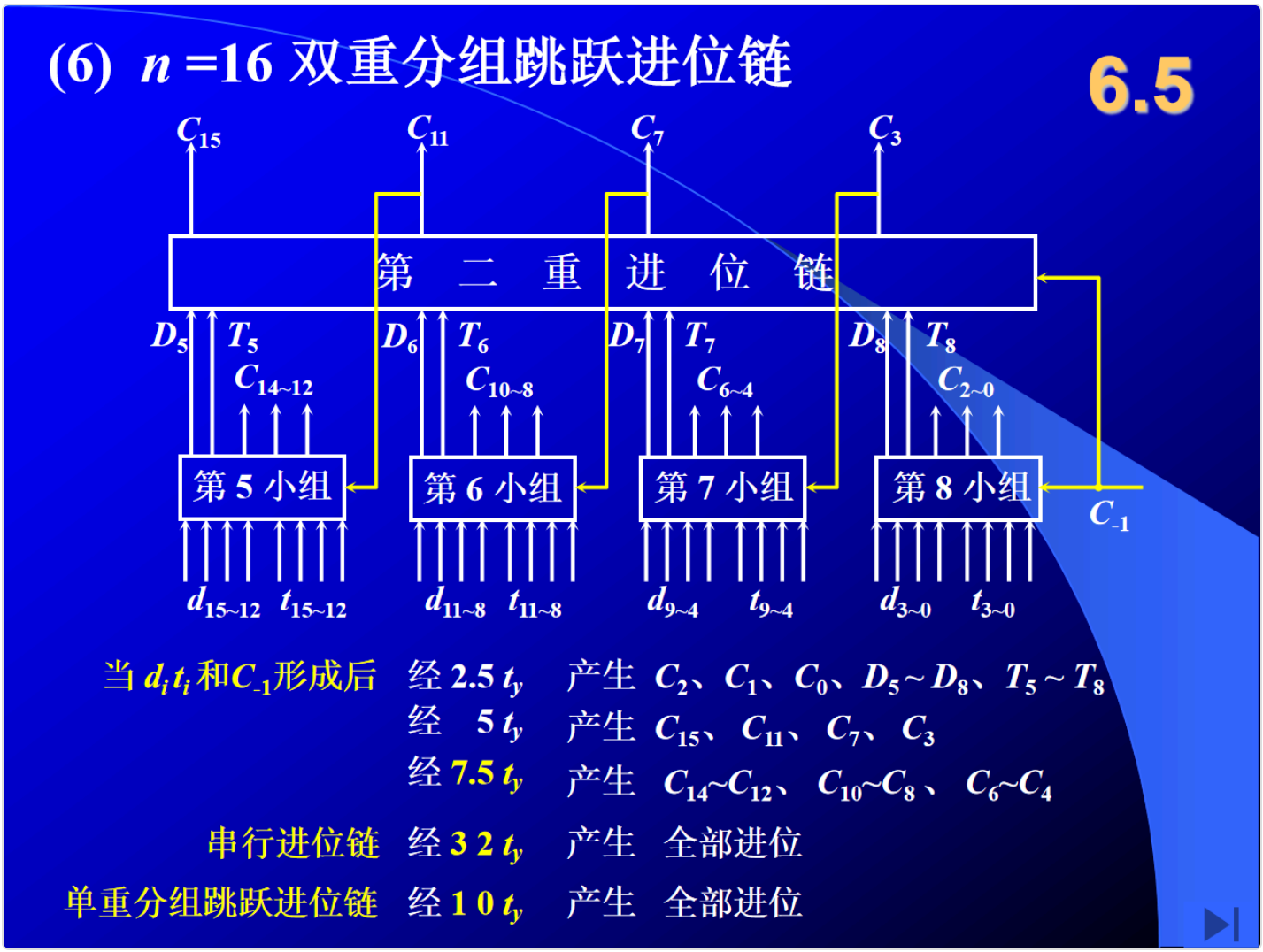
各级进位同时产生；设与或非门延迟 1.5ty，**$d_i$、$t_i$ 形成后只需 2.5ty 产生全部进位。**

**单重分组跳跃进位链**（课件 p.364）：n 位分若干小组，组内并行、组间串行。n=16 每 4 位一组：




$d_i$、$t_i$ 形成后经 2.5ty 产生 C3～C0，5ty 产生 C7～C4，7.5ty 产生 C11～C8，**10ty 产生 C15～C12（全部）**。

**双重分组跳跃进位链**（课件 p.365–370）：n 位分大组、大组含小组；**每个大组内各小组的最高位进位同时产生**，大组间串行。n=32 分 2 大组 × 4 小组 × 4 位。小组进位输出：$D_8$ = d3+t3d2+t3t2d1+t3t2t1d0（小组本地进位，与外来进位无关）、$T_8$ = t3t2t1t0（小组传送条件）；大组链：C3 = D8+T8$C_{-1}$，C7 = D7+T7C3，C11 = D6+T6C7，C15 = D5+T5C11（课件 p.366）。小组线路只产生低 3 位进位 C2、C1、C0 和本组 D、T（课件 p.368）。




n=16 时序（$d_i$、$t_i$ 与 $C_{-1}$ 形成后）：2.5ty 产生 C2、C1、C0 及 D5～D8、T5～T8；5ty 产生 C15、C11、C7、C3；7.5ty 产生 C14～12、C10～8、C6～4；**10ty 全部进位**（对比：串行 32ty、单重 10ty）。n=32 双重分组同样四级：2.5ty → C2,C1,C0,D1～D8,T1～T8；5ty → C15,C11,C7,C3；7.5ty → C18～16,C14～12,C10～8,C6～4 及 C31,C27,C23,C19；10ty → C30～28,C26～24,C22～20（课件 p.370）。 **易错警示**：2.5ty/10ty 均指"$d_i$、$t_i$ 形成之后"；串行 n 位是 2n·ty 不是 n·ty；双重分组大组间仍串行，故 n=32 与 n=16 同为 10ty（大组数=2）。

## 18. 易错点与高频陷阱汇总

1. 同一代码四种解释真值不同：10000000 = 无符号 128 / 原 −0 / 补 −128 / 反 −127（p.280）。
2. 原码、反码的 0 有两种表示；补码、移码的 0 唯一（p.265、278、286）。
3. 补码多表示一个最小负数：小数 [−1]补=1.0000、整数 [$-2^n$]补=1,00…0，原码表示不了（p.275）。
4. 原↔补互求"符号位不变、数值取反加 1"；求 [−y]补"连同符号位取反加 1"——两者别混（p.272、281）。
5. 移码 = 补码符号位取反；移码全 0 对应最小真值而非 0；移码只用于整数阶码（p.284–286）。
6. 负数补码算术右移添 1、反码添 1、原码添 0；左移一律添 0（p.300）。
7. 补码右移向 −∞ 舍入：−26 右移 2 位补码得 −7、反码得 −6（p.303）。

8. 符号位进位丢弃是 mod 运算的正常行为，不等于溢出；溢出用 V=C符⊕C数 或双符号位 01/10 判（p.309–310）。
9. 正+正得负、负+负得正必溢出：9/16+11/16 机器得 −12/16、−97−41 机器得 +118 都是错的（p.308）。
10. 原码一位乘用逻辑右移、两位乘与 Booth 用算术右移；Booth 共 n+1 步且末步不移位（p.319、324、328）。
11. 两位乘 Cj=1 的末尾校正加不移位；n 奇数时移 (n+1)/2 次、最多加 (n+1)/2+1 次（p.323–325）。
12. 除法第一次上商判溢出：小数除 |x|≥|y| 即溢出；补码除上商看余数与除数符号同异，不是余数正负（p.336、343）。
13. 补码除末位恒置 1，误差 $\le 2^{-n}$；商符由第一次上商自然形成（p.346–347）。
14. 对阶只能小阶向大阶（尾数右移）；尾数溢出右规不是真溢出，阶码双符号位 01 才是上溢、10 下溢按机器零（p.349、358）。
15. 补码规格化看"符号位与第一数位不同"；[−1/2]补=1.100…0 不规格化、[−1]补=1.000…0 规格化（p.351–352）。
16. 进位时间基准是 $d_i$、$t_i$ 形成后：串行 2n·ty、4 位并行 2.5ty、16 位单重/双重均 10ty（p.362–369）。

## 19. 典型例题精讲

**精讲 1（Booth 算法，课件 p.329 例 6.23）**：x=+0.0011，y=−0.1011，求 [x·y]补。 ① 准备：[x]补=00.0011（双符号），[−x]补=11.1101，[y]补=1.0101，附加位 $y_5=0$。 ② 逐步（判断位→操作→算术右移）： - $y_4y_5=10$ → +[−x]补：00.0000+11.1101=11.1101 → 右移 11.1110，乘数串 1 1010，附加 1； - $y_4y_5=01$ → +[x]补：11.1110+00.0011=00.0001 → 右移 00.0000，乘数串 11 101，附加 0； - $y_4y_5=10$ → +[−x]补：00.0000+11.1101=11.1101 → 右移 11.1110，乘数串 111 10，附加 1； - $y_4y_5=01$ → +[x]补：11.1110+00.0011=00.0001 → 右移 00.0000，乘数串 1111 1，附加 0； - $y_0y_1=10$ → +[−x]补：00.0000+11.1101=11.1101，**不移位**。 ③ 拼接：[x·y]补 = 11.1101 1111 → 单符号 1.11011111。验证：x·y = (3/16)(−11/16) = −33/256，而 1.11011111 真值 = −(0.00100001) = −33/256 ✓。 教学点：判断位永远取"乘数串末位+附加位"；加法用双符号位防中间溢出；最后一步对应 $(y_1-y_0)$ 项故不移位。

**精讲 2（补码除法，课件 p.346 例 6.26）**：x=−0.1011，y=+0.1101。 ① [x]补=1.0101，[y]补=0.1101，[−y]补=1.0011；异号 → 第一步加：1.0101+0.1101=0.0010。 ② $R_0$=0.0010 与 y 同号 → 商 1（商符，负商的反码形式首位）；左移 0.0100，商 1 故 +[−y]补：0.0100+1.0011=1.0111。 ③ 1.0111 与 y 异号 → 商 0；左移 0.1110，+[y]补：0.1110+0.1101=1.1011。 ④ 异号 → 商 0；左移 1.0110，+[y]补：1.0110+0.1101=0.0011。 ⑤ 同号 → 商 1；左移 0.0110，+[−y]补：0.0110+1.0011=1.1001；末位恒置 1。 ⑥ 商 = 1.0011 → [x/y]补=1.0011，真值 −0.1101。验证：−11/13 ≈ −0.846，−0.1101 = −0.8125，误差 0.034 < $2^{-4}$ ✓。 教学点：第一步"同号减、异号加"；上商统一"余除同号 1、异号 0"；左移是逻辑左移（符号位移丢）。

**精讲 3（浮点减法全流程，课件 p.356–357 例 6.28）**：x=(−5/8)×$2^{-5}$，y=(7/8)×$2^{-4}$，阶码 3 位、尾数 6 位（均另加双符号）。 ① 写补码：x=−0.101000×$2^{-101}$ → [x]补=11,011;11.011000；y=0.111000×$2^{-100}$ → [y]补=11,100;00.111000。 ② 对阶：[Δj]补=11,011+00,100=11,111（−1）→ x 阶小，Sx 右移 1 位、阶 +1：[x]补'=11,100;11.101100。 ③ 尾数求差：[Sx]补'+[−Sy]补 = 11.101100+11.001000 = 10.110100（丢最高进位）

→ 双符号 10，尾数溢出。 ④ 右规：右移 1 位得 11.011010，阶码 11,100+1=11,101 → [x−y]补 =11,101;11.011010。 ⑤ 还原：尾数真值 −0.100110，阶 −3 → x−y = $-0.100110\times2^{-11} = (-19/32)\times2^{-3}$。验证：−5/256−14/256 = −19/256 ✓。阶码双符号 11 未溢出。 教学点：五步缺一不可；右规后必须重新检查规格化（11.011010 符号 1 首位 0，已规格化）。

**精讲 4（溢出判断，课件 p.308 练习 2）**：8 位补码 A=−97、B=+41，求 A−B。 [A]补=1,0011111，[−B]补 =1,1010111；相加 = 10,1110110 → 丢进位 0,1110110 = +118。两负数相加得正 → 溢出（V：符号位进位 1 ⊕ 数值最高位进位 0 = 1）。真值 −138 < −128 超出范围，故课件标"错"。双符号位验证：11,0011111+11,1010111 = 110,1110110 → 10,1110110，双符号 10 = 负溢 ✓。

## 附：本章速查表

**四码定义（负数情形，n 为数值位数）**

| 码制 | 整数 | 小数 | 负数快捷求法 | 0 的表示 |
|---|---|---|---|---|
| 原码 | $2^n-x$ | $1-x$ | 符号 1+绝对值 | 两种 |
| 反码 | $(2^{n+1}-1)+x$ | $(2-2^{-n})+x$ | 原码数值取反 | 两种 |
| 补码 | $2^{n+1}+x \pmod{2^{n+1}}$ | $2+x \pmod 2$ | 原码数值取反+1 | 唯一 |
| 移码 | $2^n+x$ | 无 | 补码符号取反 | 唯一 |

**8 位（含 1 符号位）范围**：无符号 0～255；原/反 −127～+127；补 −128～+127。 **求补操作**：[−y]补 = [y]补全位取反 +1。 **溢出**：V = C符 ⊕ C最高数值位；双符号 01 正溢、10 负溢。 **移位添补**：正数全 0；负原 0/0；负补 左 0 右 1；负反 1/1。 **Booth 判断**：00/11→+0 移；01→+[x]补 移；10→+[−x]补 移；n+1 步末步不移。 **原码两位乘判断（$y_{n-1}y_n$,Cj）**：000→+0；001/010→+x；*011/100→+2x*；101/110→−x* 置 Cj=1；111→+0 保持 Cj=1；均右移 2 位（末校正不移）。 **除法上商**：原码除 余正 1 余负 0；补码除 余除同号 1 异号 0，末位恒置 1；均上商 n+1 次、左移 n 次；原码加 n+1 次、补码加 n 次。 **规格化**：原码第一数位 1；补码符号⊕第一数位=1；左规左移阶减、右规右移阶加。 **浮点边界（阶 m 位数值、尾 n 位数值，补码）**：最大正 $2^{(2^m-1)}(1-2^{-n})$；最小正 $2^{-(2^m-1)}\cdot2^{-n}$；规格化后最小正 $2^{-(2^m-1)}\cdot2^{-1}$。 **机器零**：阶移尾补 = 全 0；阶补尾补 = 1,00…0;×.×…× 或 ×,…;0.0…0。 **IEEE 754**：32=1+8+23；64=1+11+52；80=1+15+64；隐含最高 1。 **进位时间（$d_i$,$t_i$ 形成后）**：串行 n 位 2n·ty；4 位并行 2.5ty；16 位单重 10ty；16/32 位双重 10ty。 **74181**：M=0 算术、M=1 逻辑；S3～S0 选运算。