2.4 基本运算方法

补码（+、–、×、÷）➡ 定点数运算

原码（+、–、×、÷）
移码（+、–）
} 浮点数运算

2.4.1 定点数的运算

定点数一般用补码表示；

符号位参加运算。

重点：基于补码的加、减乘、除法

1. 补码的加减法

$$(X+Y)_{补} = X_{补} + Y_{补} \qquad (1)$$

$$(X-Y)_{补} = X_{补} + (-Y)_{补} \qquad (2)$$

数学依据:

$$(X+Y)_{补}=X+Y+2^n \;=X+Y+2^n+2^n \;=X_{补}+Y_{补}$$

$$(X-Y)_{补}=X-Y+2^n \;=X-Y+2^n+2^n \;=X_{补}+(-Y)_{补}$$

全过程以2^n为模，即除以2^n后取余数。

其中，$(-Y)_{补} = [Y_{补}]_{变补}$

Y的符号置反后再表示成补码

$Y_{补}$连同符号一起变反、末尾+1

【例】求$(X+Y)_补$

1) $X=3$ $X_补=\underline{0}\ 0011$

$Y=2$ $Y_补=\underline{0}\ 0010$ +

$\underline{0}\ 0101$（+5补码）

2) $X=-3$ $X_补=\underline{1}\ 1101$

$Y=-2$ $Y_补=\underline{1}\ 1110$ +

$\underline{1}\ 1011$（-5补码）

【例】求$(X-Y)_补$

1) $X=\ 4$ $X_补=\underline{0}\ 0100$

$Y=-5$ $Y_补=\underline{1}\ 1011$

$(-Y)_补=\underline{0}\ 0101$ +

$\underline{0}\ 1001$（+9补码）

2) $X=-4$ $X_补=\underline{1}\ 1100$

$Y=\ 5$ $Y_补=\underline{0}\ 0101$

$(-Y)_补=\underline{1}\ 1011$ +

$\underline{1}\ 0111$（–9补码）

※补码表示与变补运算的区别

【举例】

$1\,0101_{原} \xrightarrow{\text{补码表示}} 1\,1011$

$0\,0101_{原} \xrightarrow{\text{补码表示}} 0\,0101$

符号位不变

负数尾数改变，正数尾数不变。

$1\,0011_{补} \xrightarrow{\text{变补运算}} 0\,1101$

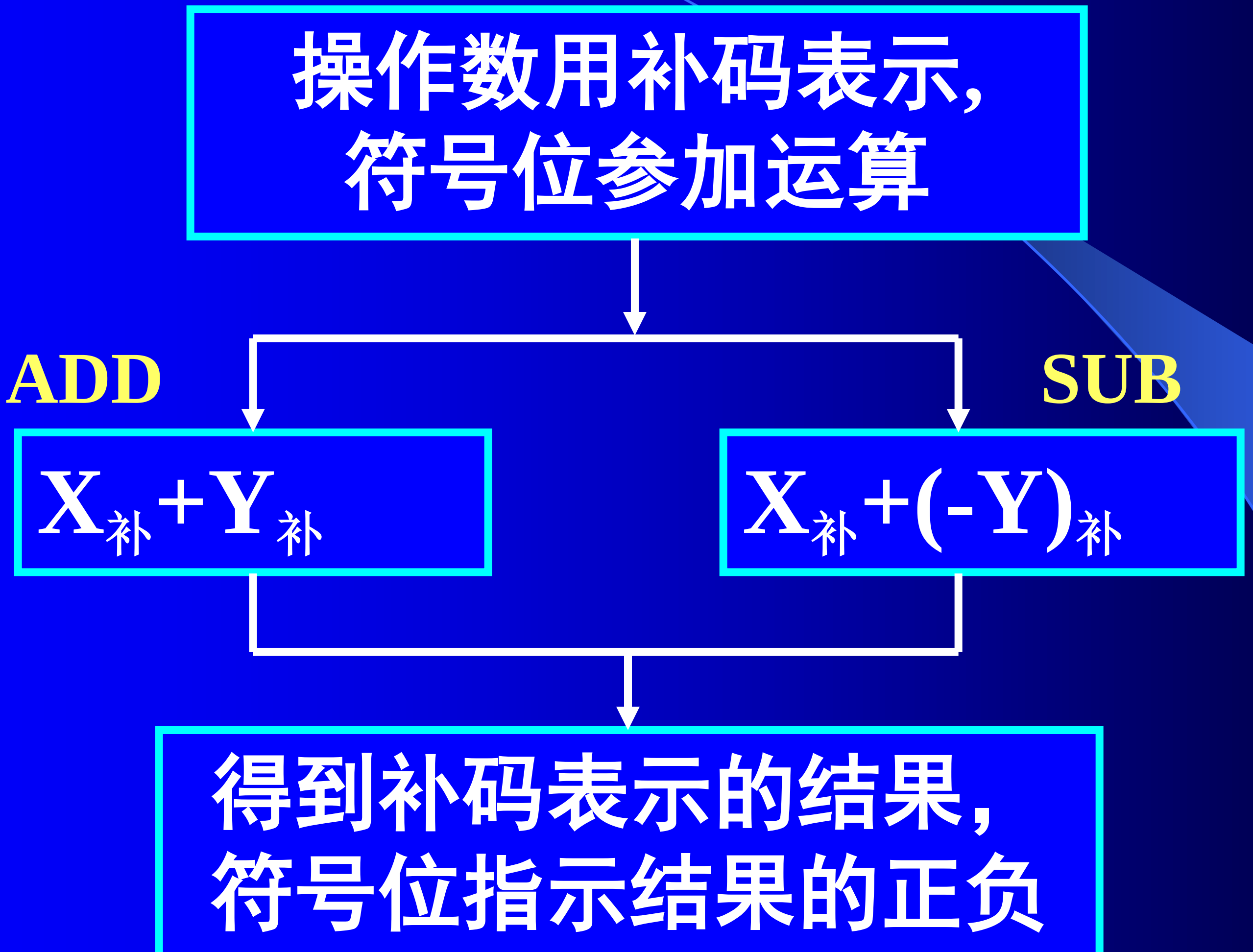
$0\,0011_{补} \xrightarrow{\text{变补运算}} 1\,1101$

符号位变反，

尾数变反、末尾加1

$(-Y)_{补}$也称为$Y_{补}$的机器负数。

（2）补码加减运算流程



（3）逻辑实现

#控制信号

加法器输入端：

Sub：控制MUX和Cin

A：输入$A_{补}$

B：输入$B_{补}$

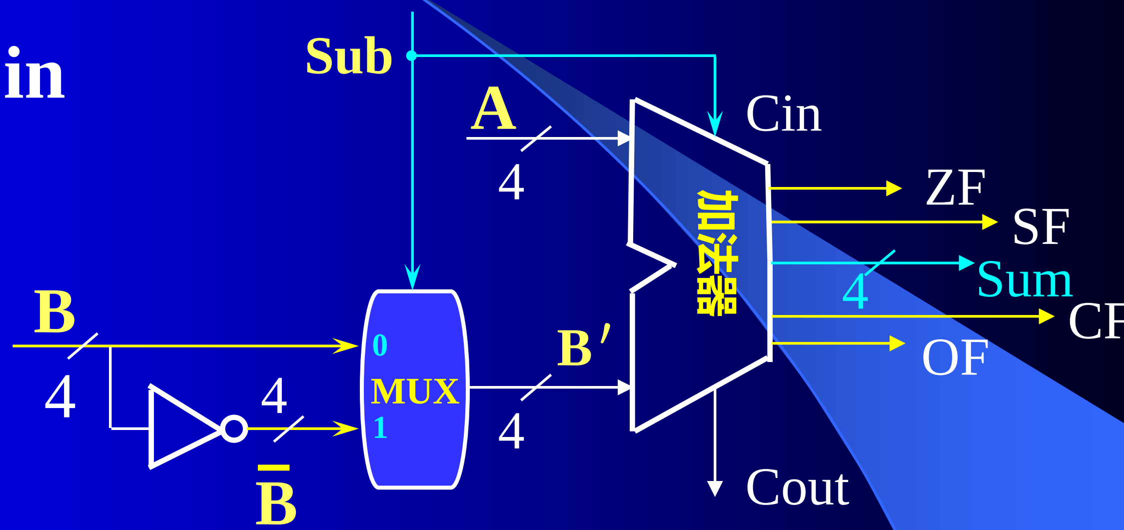
加法器输出端：

Sum: 加法结果

Cout: 进位信号

ZF: 0标志位；SF：符号标志；CF:进位/借位标志；OF:溢出标志。

#补码加减运算器框图



补码加/减运算部件逻辑

（4）溢出判断

*溢出的判断规则

思考：*在什么情况下可能产生溢出？*

［假设］补码表示的 A 、B两数做加减运算

A：4位尾数，1位符号(S_A)

B：4位尾数，1位符号(S_B)

补码，故符号位也参加运算

结果的符号，记为S_f

符号位的进位，记为C_f

尾数最高位的进位，记为C

1）A=3、B=2

3+2：

```
  0 0011
  0 0010  +
  --------
  0 0101  正确
```

2）A=10、B=7

10+7：

```
  0 1010
  0 0111  +
  --------
  1 0001  正溢
```

3）A= -3、B= -2

-3+(-2)：

```
  1 1101
  1 1110  +
  --------
  1 1011  正确
```

4）A= -10、B= -7

-10+(-7)：

```
  1 0110
  1 1001  +
  --------
  0 1111  负溢
```

5）A=6、B= -4

6+(-4)：

```
  0 0110
  1 1100  +
  --------
  0 0010  正确
```

6）A= -6、B=4

-6+4：

```
  1 1010
  0 0100  +
  --------
  1 1110  正确
```

①硬件判断逻辑一（根据S_A、S_B与S_f的关系）

（2）A=10 B=7

10+7： 0 1010

0 0111 ＋

正溢 1 0001

（4）A= -10 B= -7

-10+(-7)： 1 0110

1 1001＋

负溢 0 1111

溢出逻辑= $\overline{S_A}\,\overline{S_B}\,S_f + S_A S_B \overline{S_f}$

②硬件判断逻辑二（根据C_f与C的关系）

1）A=3　B=2

3+2:　0 0011

C_f=0　0 0010 +

C =0　0 0101　正确

2）A=10　B=7

10+7:　0 1010

C_f=0　0 $_1$0111 +

C =1　1 0001　正溢

3）A= -3　B= -2

-3+(-2):　1 1101

C_f=1　$_1$1 $_1$1110 +

C =1　1 1011　正确

4）A= -10　B= -7

-10+(-7):　1 0110

C_f=1　$_1$1 1001 +

C =0　0 1111　负溢

5）A=6　B= -4

6+(-4):　0 0110

C_f=1　$_1$1$_1$1100 +

C =1　0 0010　正确

6）A= -6　B=4

-6+4:　1 1010

C_f=0　0 0100 +

C =0　1 1110　正确

①硬件判断逻辑一（S_A、S_B与S_f的关系）

溢出逻辑= $\overline{S_A}\,\overline{S_B}S_f + S_A S_B \overline{S_f}$

②硬件判断逻辑二（C_f与C的关系）

溢出逻辑= $C_f \oplus C$

③硬件判断逻辑三（从双符号位）

1）3+2：

```
  00 0011
  00 0010  +
  ---------
  00 0101  正确
```

2）10+7：

```
  00 1010
  00 0111  +
  ---------
  01 0001  正溢
```

第１符号位 S_{f1}

第２符号位 S_{f2}

3）-3+(-2)：

```
  11 1101
  11 1110  +
  ---------
  11 1011  正确
```

4）-10+(-7)：

```
  11 0110
  11 1001  +
  ---------
  10 1111  负溢
```

5）6+(-4)：

```
  00 0110
  11 1100  +
  ---------
  00 0010  正确
```

6）-6+4：

```
  11 1010
  00 0100  +
  ---------
  11 1110  正确
```

①硬件判断逻辑一（S_A、S_B与S_f的关系）

溢出逻辑= $\overline{S_A}\,\overline{S_B}\,S_f + S_A S_B \overline{S_f}$

②硬件判断逻辑二（C_f与C的关系）

溢出逻辑= $C_f \oplus C$

③硬件判断逻辑三（双符号位S_{f1}、S_{f2}）

溢出逻辑= $S_{f1} \oplus S_{f2}$

00/11-正确；10-正溢；01-负溢；

2、原码加减运算

[符号位单独处理、数值位加减]

先比较两数符号：

①加法：同号数值位求和，异号求差；

②减法：异号数值位求和，同号求差；

求和	求差
$3_{原}+2_{原}$	$3_{原}+[-2]_{原}$
$3_{原}-[-2]_{原}$	$3_{原}-\ 2_{原}$

※求和时：数值位相加，和的符号取被加数（被减数）符号[若最高位产生进位，则结果有溢出。

※求差时：被加数（被减数）与加数（减数）求补后相加。

⊙最高数值位有进位，相加结果为正，数值位正确；符号取被加数（被减数）的符号。

⊙最高数值位无进位，相加结果为负，得到数值位的补码，需对结果求补还原为绝对值形式的数值位；符号位与被加数（被减数）的符号相反。

[例]已知 $[X]_{原} = 1.0011$，$[Y]_{原} = 1.1010$，计算$[X+Y]_{原}$

解：由原码加减运算规则知：同号相加，则求和，和的符号同被加数符号。

和的数值位为：0011 + 1010 = 1101

和的符号位为：1

$$[X+Y]_{原} = 1.1101$$

[例]已知 $[X]_{原} = 1.0011$，$[Y]_{原} = 1.1010$，要求计算$[X-Y]_{原}$

解：由原码加减运算规则知：同号相减，则数值位求差

差的数值位为：$0011+(1010)_{求补} = 0011 + 0110 = 1001$

最高数值位无进位，表明加法结果为负，需对1001求补，还原为绝对值形式的数值位。即：$(1001)_{求补} = 0111$

差的符号位为$[X]_{原}$的符号位取反，即

$$[X-Y]_{原} = 0.0111$$

3、标准移码的加减

[符号位和数值部分一起处理]

$$[E1]_{移}+[E2]_{移}=2^{n-1}+E1+2^{n-1}+E2=2^n+E1+E2$$

$$=[E1+E2]_{补}\quad(\mathrm{mod}\ 2^n)$$

$$[E1]_{移}-[E2]_{移}=2^{n-1}+E1+2^n-[E2]_{移}\quad(\mathrm{mod}\ 2^n)$$

$$=2^{n-1}+E1+2^n-2^{n-1}-E2$$

$$=2^n+E1-E2=[E1-E2]_{补}\quad(\mathrm{mod}\ 2^n)$$

[重要结论]

两数移码的加减等于两数加减后表示成的补码。

※补码和移码：符号位相反、数值位相同

因此，可得到移码的下列加减法则：

① 加法：直接将$[E1]_{移}$和$[E2]_{移}$进行模2^n加，结果的符号取反。

② 减法：先将减数$[E2]_{移}$求补，然后再与被减数 $[E1]_{移}$进行模2^n加，结果的符号取反。

[溢出判断]

进行模2^n相加时，如果两个加数与和数符号全相同，则发生了溢出。

[例]用4位移码计算“–7+（– 6）”和“–3 + 6”的值。

$[-7]_{移} = 0001$　　$[-6]_{移} = 0010$

$[-3]_{移} = 0101$　　$[6]_{移} = 1110$

$[-7]_{移} + [-6]_{移} = \underline{0}001 + \underline{0}010 = \underline{0}011_{补}$

（符号都为0，有溢出）

$[-3]_{移} + [6]_{移} = \underline{0}101 + \underline{1}110 = \underline{0}011_{补}$，

（符号取反后为 1011，其真值为+3）

思考：$[-7+(-6)]_{移} = ?$　　$[-3+(6)]_{移} = ?$

[例] 用四位移码计算“–7 –（– 6）”和“–3 – 5”

$[-7]_{移} = 0001 \qquad [-6]_{移} = 0010$

$[-3]_{移} = 0101 \qquad [5]_{移} = 1101$

$[-7]_{移} - [-6]_{移} = \underline{0}001 + \underline{1}110 = \underline{1}111_{补}$

符号取反后为 0111，其真值为–1。

$[-3]_{移} - [5]_{移} = \underline{0}101 + \underline{0}011 = \underline{1}000_{补}$

符号取反后为 0000，其真值为–8

2.3.2 定点数乘法

- √ 原码一位乘
- √ 补码一位乘
- √ 原码两位乘

1、原码一位乘

原码乘法 ⟶ 部分积累加、移位。

$X_{原}$ $Y_{原}$

[例] 0.1101 × 1.1011

乘积 $P = |X| \times |Y|$

符号 $S_P = S_X \oplus S_Y$

手工运算

$$
\begin{array}{r}
0.\underline{1101} \\
\times 0.1011 \\
\hline
1101 \\
1101 \\
0000 \\
+1101 \\
\hline
.10001111
\end{array}
$$

$0.1101 \leftarrow |X|$

$0.1011 \leftarrow |Y|$

部分积

①加数只为|X|或0

②个数为|Y|的位数

添加符号：$\underline{1}.10001111$

思考：1）加数的个数增多情况

2）加数的位数增多的情况

解决办法：可将1次总加改为分步移位累加

● 原码1位乘法

（1）算法原理[每次将1位乘数所对应的部分积与原部分积的累加和相加，并移位]

设置寄存器：

A：存放部分积累加和、乘积高位

B：存放被乘数

C：存放乘数、乘积低位

设置初值：

A = 00.0000

B = |X| = 00.1101

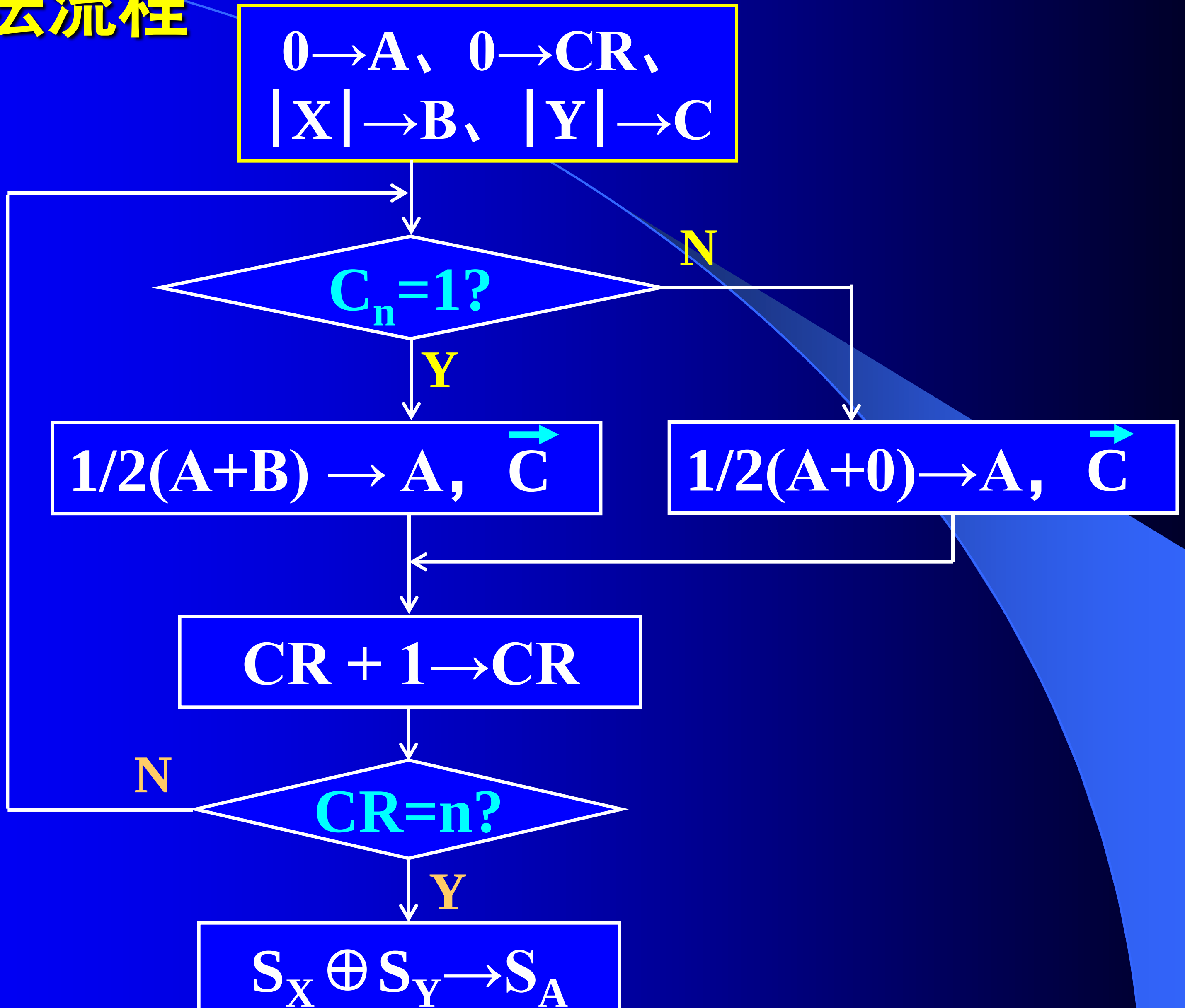
C = |Y| = .1011

步数CR	条件	操作	A	C C_n
			00.0000	.1011
0)	$C_n=1$	+B	+ 00.1101	
			00.1101	1011
		→	00.0110	1.101
1)	$C_n=1$	+B	+ 00.1101	
			01.0011	1101
		→	00.1001	11.10
2)	$C_n=0$	+0	+ 00.0000	
			00.1001	1110
		→	00.0100	111.1
3)	$C_n=1$	+B	+ 00.1101	
			01.0001	1111
		→	00.1000	1111

$X_{原} \times Y_{原} = 1.10001111$

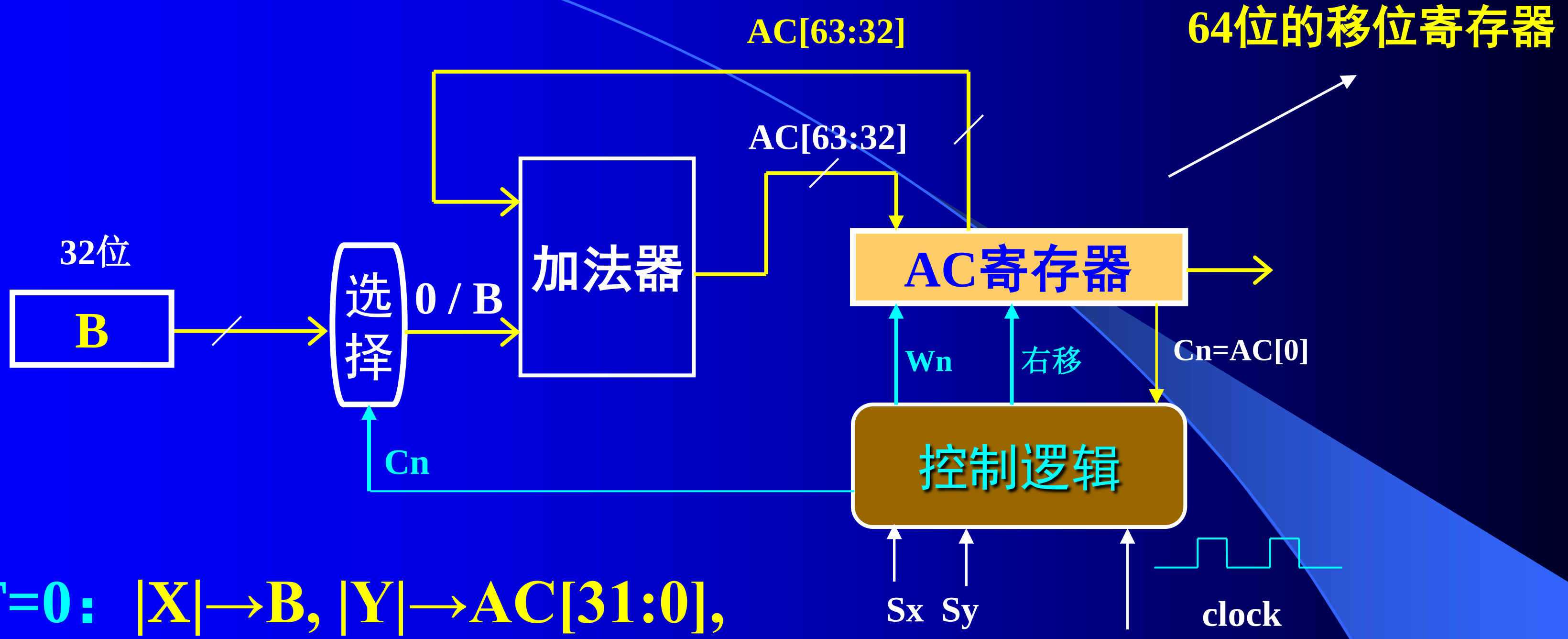
```
   0.1101  —B
 × 0.1011  —C
 ---------
     1101
    1101
   0000
 +1101
 ---------
 0.10001111
```

（2）算法流程



0→A、0→CR、
|X|→B、|Y|→C
C_n=1?
N
Y
1/2(A+B) → A，$\vec{C}$
1/2(A+0)→A，$\vec{C}$
CR + 1→CR
CR=n?
N
Y
$S_X \oplus S_Y \rightarrow S_A$

（3）32位硬件逻辑方案



T=0：|X|→B, |Y|→AC[31:0],
0→AC[63:32]

T=1：AC[63:32]+0/B→AC[63:32], AC右移

T=2：

⋮

T=n：同上，然后置符号。

2. 补码一位乘法

※Booth（比较法）

$$[XY]_{补}=[An]_{补}+(Y_1-Y_0)\times[X]_{补}$$

Y_n (高位)	Y_{n+1} (低位)	运算操作
0	0	$\frac{1}{2}\times A_{补}$
0	1	$\frac{1}{2}\times(A_{补}+X_{补})$
1	0	$\frac{1}{2}\times(A_{补}-X_{补})$
1	1	$\frac{1}{2}\times A_{补}$

*乘数尾添加Y_{n+1}，循环判别Y_nY_{n+1},累加如表情况的的校正值，再整体右移1位（即$\frac{1}{2}\times$）。

[例] $X=-0.1101$，$Y=-0.1011$，计算$[XY]_{补}$

初始化$A=00.0000$，$B=X_{补}=11.0011$，$-B=-X_{补}=001101$

$C=Y_{补}=1.0101$

步数	$C_n C_{n+1}$条件	操作	A	C C_{n+1}	
			000000	10101 0	
1	1 0	-B	+ 001101		
			001101	10101 0	右移1位→
			000110	11010 1	
2	0 1	+B	+ 110011		
			111001	11010 1	右移1位→
			111100	11101 0	

步数	C_nC_{n+1}条件	操作	111100 11101 0
3	1 0	-B	+ 001101
			1 001001 11101 0 右移1位→
			000100 11110 1
4	0 1	+B	+ 110011
			110111 11110 1 右移1位→
			111011 11111 0
校正	1 0	-B	+ 001101
			1 001000 11111 0 保持不变!!

$[XY]_补 = 00.1000\ 1111$

$= +0.1000\ 1111$

※补码1位乘法除了**比较法**，还有**校正法**。

①乘数$Y_{补}$为正，乘以$X_{补}$累加，结果不校正；

②乘数$Y_{补}$为负，乘以$X_{补}$累加，结果$-X_{补}$校正；

（请参考教材）

2.4.3 定点数的除法

√补码不恢复余数除法

√补码恢复余数的除法

√原码恢复/不恢复余数除法

1. 补码不恢复余数除法

[算法思想]（$|X|<|Y|$）

被除数$X_{补}$、除数$Y_{补}$、余数r_i, $i=0$、1…

初始化：令$r_0=X_{补}$, 比较r_0与$Y_{补}$符号，同号上商1，异号上商0

循环：$i=1\ldots n$, 按下表条件决定每步操作

r_i、$Y_{补}$数符	商	对应操作
同号	1	$r_{i+1}=2\times[r_i]_{补}-Y_{补}$
异号	0	$r_{i+1}=2\times[r_i]_{补}+Y_{补}$

$[-Y_{补}] \leftrightarrow +[Y_{补}]_{变补}$

$\times 2 \leftrightarrow$ 左移1位

商修正：符号位+1，末尾恒置1

余数修正：左移了n次，则余数$=2^{-n}\times r_n$

[例] $X \div Y = +0.1000 \div (-0.1010) = ?$

$R = X_补 = \underline{00}1000$，$B = Y_补 = \underline{11}0110$，$-B = 001010$，$Q = 00000$

步数	条件	操作	被除数/余数		商Q
1	$r_0 Y_补$异号	上商0	001000	r_0	00000
		$2r_0$ /←	010000		0000$\underline{0}$
		+B	+ 110110		
			1 000110	r_1	
2	$r_1 Y_补$异号	上商0	000110		00000
		$2r_1$ /←	001100		0000$\underline{0}$
		+B	+ 110110		
			1 000010	r_2	

$R=X_{补}=\underline{00}1000$，$B=Y_{补}=\underline{11}0110$，$-B=001010$

			000010	r_2	$00\underline{000}$
3	$r_2Y_{补}$异号	上商0	000010		00000
		$2r_2$ /←	000100		$0000\underline{0}$
		+B	+ 110110		
			111010	r_3	
4	$r_3Y_{补}$同号	上商1	111010		00001
		$2r_3$ /←	110100		$0001\underline{0}$
		-B	+ 001010		+1　置1
			111110	r_4	

商校正：　左移了4位　$1001\underline{1}_{补}$

余数：　$2^{-4} \times r_4$　　数符+1、末尾恒置1

2.4.4 浮点数四则运算

浮点运算的实现：

低档微机，通过子程序

中档微机，通过浮点处理器（协处理器）

高档微机，通过专门的浮点运算部件

[以IEEE754浮点数为例]

浮点数运算 → 尾数运算、阶码运算

尾数运算	阶码运算
↓	↓
原码	移码
↓	↓
加减乘除	加减

0、IEEE754短浮点数阶码的加减

阶码用非标准移码表示（仅偏移$2^{n-1}-1=127$）

①Ex和Ey分别是两数阶码，Eb是结果阶码，则：

$Eb = Ex+Ey+129 \pmod{2^8}$

原理推导：

$$
\begin{aligned}
[Ex+Ey]_{移} &= 127 + (Ex+ Ey) = (127+Ex)+(127+Ey) -127 \\
&= [Ex]_{移}+[Ey]_{移}-127 \\
&= [Ex]_{移}+[Ey]_{移}+127_{求补} =[Ex]_{移}+[Ey]_{移}+(-127)_{补} \\
&= [Ex]_{移} + [Ey]_{移} +10000001 \\
&= [Ex]_{移} + [Ey]_{移} +129 \pmod{2^8}
\end{aligned}
$$

②Ex和Ey分别是两数阶码，Eb是结果阶码，则：

$Eb = Ex - Ey = Ex + [-Ey]_{补} + 127 \quad (\bmod 2^8)$

原理推导：

$[Ex - Ey]_{移} = 127 + (Ex - Ey)$

$= 127 + Ex - Ey - 127 + 127$

$= [Ex]_{移} - [Ey]_{移} + 127$

$= [Ex]_{移} + ([Ey]_{移})_{求补} + 01111111 \quad (\bmod 2^8)$

或者 $= [Ex]_{移} + (-[Ey]_{移})_{补} + 01111111 \quad (\bmod 2^8)$

$\bmod 2^8$，等价于忽略最高位的进位。

[例]若两个阶码分别为10和-5，求10+(-5)和10-(-5)

Ex = 127+10 =137=1000 1001

Ey = 127+ (–5) = 122 = 0111 1010

$[–Ey]_{补}$= 1000 0110

Eb = Ex+Ey +129

= 1000 1001 + 0111 1010 +1000 0001

= 1000 0100 = 132 （mod 2^8）

= +5 正确

```
  1000 1001
  0111 1010
+ 1000 0001
-----------
1 1000 0100
```

Eb = Ex+$[Ey]_{求补}$ +127

= 1000 1001 + 1000 0110 +0111 1111

= 1000 1110 = 142（mod 2^8）

= +15 正确

```
  1000 1001
  1000 0110
+ 0111 1111
-----------
1 1000 1110
```

1、浮点的加减运算

假设：$A=2^{AE}\times A_M$，$B=2^{BE}\times B_M$

调整阶码和尾数：$AE\rightarrow E\leftarrow BE$、$A_M$、$B_M$

$A+B=(A_M+B_M)\times 2^E$

浮点加减法的思路：

浮点数的加减

→移位操作

→尾数原码加减

(1) 检测能否简化操作

判操作数是否为0 — 尾数为0 / 阶码下溢(归0)

(2) 计算阶差

(3) 对阶

$2^2 \times 0.1001 \rightarrow \rightarrow 2^3 \times 0.0101$

$2^3 \times 0.1101 \rightarrow \rightarrow 2^3 \times 0.1101$

[目的]使两数阶码相等(小数点实际位置对齐，尾数对应权值相同)。

[规则] 小阶向大阶对齐。

※对阶操作：小阶的阶码增大，尾数右移。

[例]AE>BE，则BE+1→BE，$\overrightarrow{BM}$，直到BE=AE

※阶码比较：比较线路或减法。

（4）尾数加减.

AM±BM→AM

（5）结果规格化

尾数M左右移动，使：

$1 \le |M| < 2$

$|M|<1$ 应左移规格化 $|M| \geq 2$ 应右移规格化

溢出判断

以下情况下，可能会导致阶码溢出

- 左规（阶码 - 1）时
  - 左规时：先判断阶码是否为全0，若是，则直接置阶码下溢；否则，阶码减1后判断阶码是否为全0，若是，则阶码下溢。
- 右规（阶码 +1）时
  - 右规（+ 1）时，先判断阶码是否为全1，若是，则直接置阶码上溢；否则，阶码加1后判断阶码是否为全1，若是，则阶码上溢。

[例] 已知x=0.5, y=-0.4375, 求x+y=?

$x=0.5=1/2=(0.100...0)_2=(1.00...0)_2 \times 2^{-1}$

$y=-0.4325=(-0.01110...0)_2=(-1.110..0)_2 \times 2^{-2}$

[x]浮=0 01111110,000…00

[y]浮=1 01111101,110…00

求补

```
  0111 1110
  1000 0011
+ 0111 1111
-----------
1 1000 0000
```

求阶差、对阶: $\Delta E_{移}$=0111 1110 - 0111 1101+127 =1000 0000=128

e=128-127=+1

故对y进行对阶：[y]浮=1 0111 1110 1110…00

尾数真值相加：01.0000...00 + (10.1110...00) = 00.00100…00 < 1

(原码加法：异号求差，加数变补后求和)

求补

```
  1.0000 …00
+1.0010 …00
------------
1 0.0010 …00
```

左规： $+0.00100…00 \times 2^{-1} = +1.00…0 \times 2^{-4}$

[x+y]浮=0 0111 1011 00…00

= + 0.0625

2、浮点数的乘法

设：$A=2^{AE}\times A_M$， $B=2^{BE}\times B_M$

则：$A\times B=(A_M\times B_M)\times 2^{AE+BE}$

浮点乘法➔分解成：移码加法、原码乘法；

※运算步骤：

①求阶和AE+BE

②尾数相乘，

③结果规格化。不需左规！最多右规1次！

④其它处理：

如舍入、置0、阶码溢出判断

3、浮点数的除法

设：$A=2^{AE}\times A_M$，$B=2^{BE}\times B_M$

则：$A\div B=(A_M/B_M)\times 2^{AE-BE}$

浮点除法→分解成：移码减法、原码除法

※运算步骤：

①求阶差AE-BE

②尾数相除，

③结果规格化。需右规？？

④其它处理：

如舍入、置0、阶码溢出判断