计算机组成原理与系统结构

第四章

运算方法与运算器

http://jpkc.hdu.edu.cn/computer/zcyl/dzkjdx/

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

第 4 章 运算方法与运算器

- 4.1 定点数的加减运算及实现
- 4. 定点数的乘法运算及实现
- 4.3 定点数除法运算及实现
- 4.4 定点运算器的组成与结构
- 4. 浮点运算及运算器
- 4. 浮点运算器举例
- 本章小结

BACK

4.5 浮点运算及运算器

一 浮点加减运算

二 浮点乘法运算

三 浮点除法运算

四 浮点运算器

一、浮点加减运算

- 假设计算 $9.97\times10^{1}+4.53\times10^{-1}$
- 小数点对齐

算法一：	算法二：
$9.97\times10^{1}+4.53\times10^{-1}$	$9.97\times10^{1}+4.53\times10^{-1}$
$=9.97\times10^{1}+0.0453\times10^{1}$	$=997\times10^{-1}+4.53\times10^{-1}$
$=(9.97+0.0453)\times10^{1}$	$=(997+4.53)\times10^{-1}$
$=10.0153\times10^{1}$	$=1001.53\times10^{-1}$
$=1.00153\times10^{2}$	$=1.00153\times10^{2}$

- 结果一致
- 实际上：数据存储和计算都要受到计算机物理部件的处理位数的限制，超出部分全部丢失

一、浮点加减运算

- 假设浮点数的尾数部分**有效数值只能保存 3 位**

算法一：

- $9.97\times10^{1}+4.53\times10^{-1}$
- $=9.97\times10^{1}+0.04\times10^{1}$
- $=(9.97+0.04)\times10^{1}$
- $=10.01\times10^{1}$
- $=1.001\times10^{2}$
- $=1.00\times10^{2}$

算法二：

- $9.97\times10^{1}+4.53\times10^{-1}$
- $=7.00\times10^{-1}+4.53\times10^{-1}$
- $=(7.00+4.53)\times10^{-1}$
- $=11.53\times10^{-1}$
- $=1.153\times10^{0}$
- $=1.15\times10^{0}$

- 结果不一致
- 原因：**有效数值**部分 9.97 进行了**左移操作**，使得有效数值的**高位数字全部舍弃**了，从而导致计算结果偏差较大。
- 处理：**选择误差小的那一种方法（算法一）**

一、浮点加减运算

❖ 假设两个浮点数 X 和 Y

$$X = M_X \times 2^{E_X} \qquad Y = M_Y \times 2^{E_Y}$$

❖ 对齐小数点时，要使得阶码小的那个数变得与阶码大的那个数的阶码相等；然后对尾数（有效数位）做加减运算。

$$Z = X \pm Y = (M_X \bullet 2^{(E_X - E_Y)} \pm M_Y) \times 2^{E_Y}$$

$$E_X \le E_Y$$

浮点加减运算步骤

（1） 0操作数检查：以尽可能的简化操作。

（2）对阶：原则是小阶对向大阶

- 求阶差 $\Delta E=E_X-E_Y$，若 $\Delta E\neq 0$，即 $E_X\neq E_Y$ 时需要对阶。
- 若 $\Delta E>0$，则 $E_X>E_Y$， M_Y 每右移一位， E_Y+1，直至 $E_Y=E_X$ 。
- 若 $\Delta E<0$，则 $E_X<E_Y$， M_X 每右移一位， E_X+1，直至 $E_X=E_Y$ 。

（3）尾数相加减

浮点加减运算步骤

（4）**结果规格化：**尾数运算的结果可能出现两种非规格化情况：

- **尾数溢出：**需要右规（1次），即尾数右移1位，阶码＋1
- |尾数|$<2^{-1}$：需要左规，即尾数左移1位，阶码－1，左规可能多次，直到尾数变为规格化形式。

有前导零

（5）**舍入：**可采用截断法、0舍1入法、末位恒置1。

一、浮点加减运算

❖ IEEE 754 标准规定了 4 种可选的舍入模式：

① 向上舍入（总是朝 +∞）：为正数时，只要多余位不全为 0，就向最低有效位进 1；为负数时，则采用简单的截断法。

② 向下舍入（总是朝 −∞）：为正数时，只要多余位不全为 0，就简单地截尾；为负数时，则向最低有效位进 1。

③ 向 0 舍入：即朝数轴的原点方向舍入，就是无论正数还是负数，都采用简单的截尾，从而使得绝对值总是变小。这种方法容易累积误差。

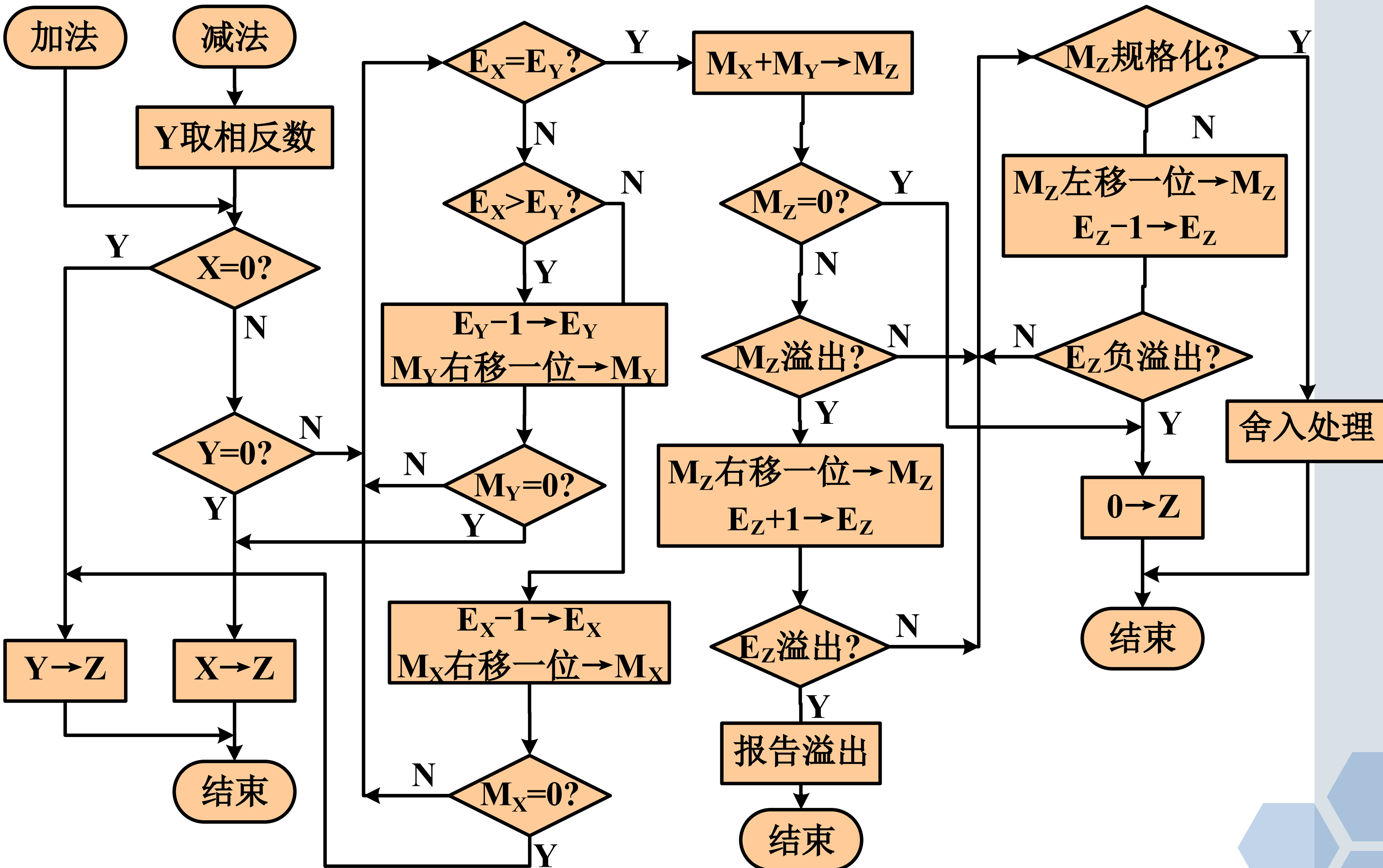
一、浮点加减运算

❖ IEEE 754 标准规定了 4 种可选的舍入模式：

④ 就近舍入：即舍入到最接近的数，就是通常的“四舍五入”。多余位 $=R_1R_2\cdots\cdots R_n$

- 当 $R_1=0$：截尾（即舍去）；
- 当 $R_1=1$，且 $R_1R_2\cdots R_n$ 不全为 0：向最低有效位进 1；
- 当 $R_1R_2\cdots\cdots R_n=10\cdots\cdots0$：即等于一半（中点），则若最低有效位为 0 就截尾，若最低有效位为 1 就进 1，以使得最低有效位总是为 0。

浮点加减运算流程



加法
减法
Y取相反数
X=0?
Y=0?
Y→Z
X→Z
结束
$E_X=E_Y$?
$E_X>E_Y$?
$E_Y-1→E_Y$
M_Y右移一位→M_Y
$M_Y=0$?
$E_X-1→E_X$
M_X右移一位→M_X
$M_X=0$?
$M_X+M_Y→M_Z$
$M_Z=0$?
M_Z溢出?
M_Z右移一位→M_Z
$E_Z+1→E_Z$
E_Z溢出?
报告溢出
结束
M_Z规格化?
M_Z左移一位→M_Z
$E_Z-1→E_Z$
E_Z负溢出?
0→Z
舍入处理
结束
Y
N

一、浮点加减运算

例： 12 位浮点数，阶码 4 位，包含 1 位阶符，尾数 8 位，包含 1 位数符，用补码表示，阶码在前，尾数（包括数符）在后，已知：

$X=(-0.1001011)\times 2^{001}$

$Y=0.1100101\times 2^{-010}$

求 $Z=X+Y$

解： $[X]_{浮}=0，\ 001\ \ 1.0110101$

$[Y]_{浮}=1，\ 110\ \ 0.1100101$

（1）对阶

- $\Delta E=E_X-E_Y=[E_X]_{补}+[-E_Y]_{补}=$ 00，001+00，010 = 00，011
- $\Delta E=3>0$，将 M_Y 右移 3 位，E_Y 加 3：
- $[Y]_{浮}=0，\ 001\ \ 0.0001100\ (101)$

一、浮点加减运算

（2）尾数相加： $[M_Z]_补$ = 1.1000001 （10 1）

$$
\begin{array}{rr}
[M_X]_补 & 11.0110101 \\
+\ [M_Y]_补 & 00.0001100 \\
\hline
[M_X+M_Y]_补 & 11.1000001
\end{array}
$$

（3）结果规格化：无溢出，左规一位：

- $[M_Z]_补$ = 1.0000011 （01）
- $[E_Z]_补$ = 00，001 + 11，111= 00，00 0

（4）舍入：按照 0 舍 1 入法，尾数多余位舍去

结果为： $[X+Y]_浮$ = 0，000 1.0000011

二、浮点乘法运算

❖假设两个浮点数 X 和 Y：

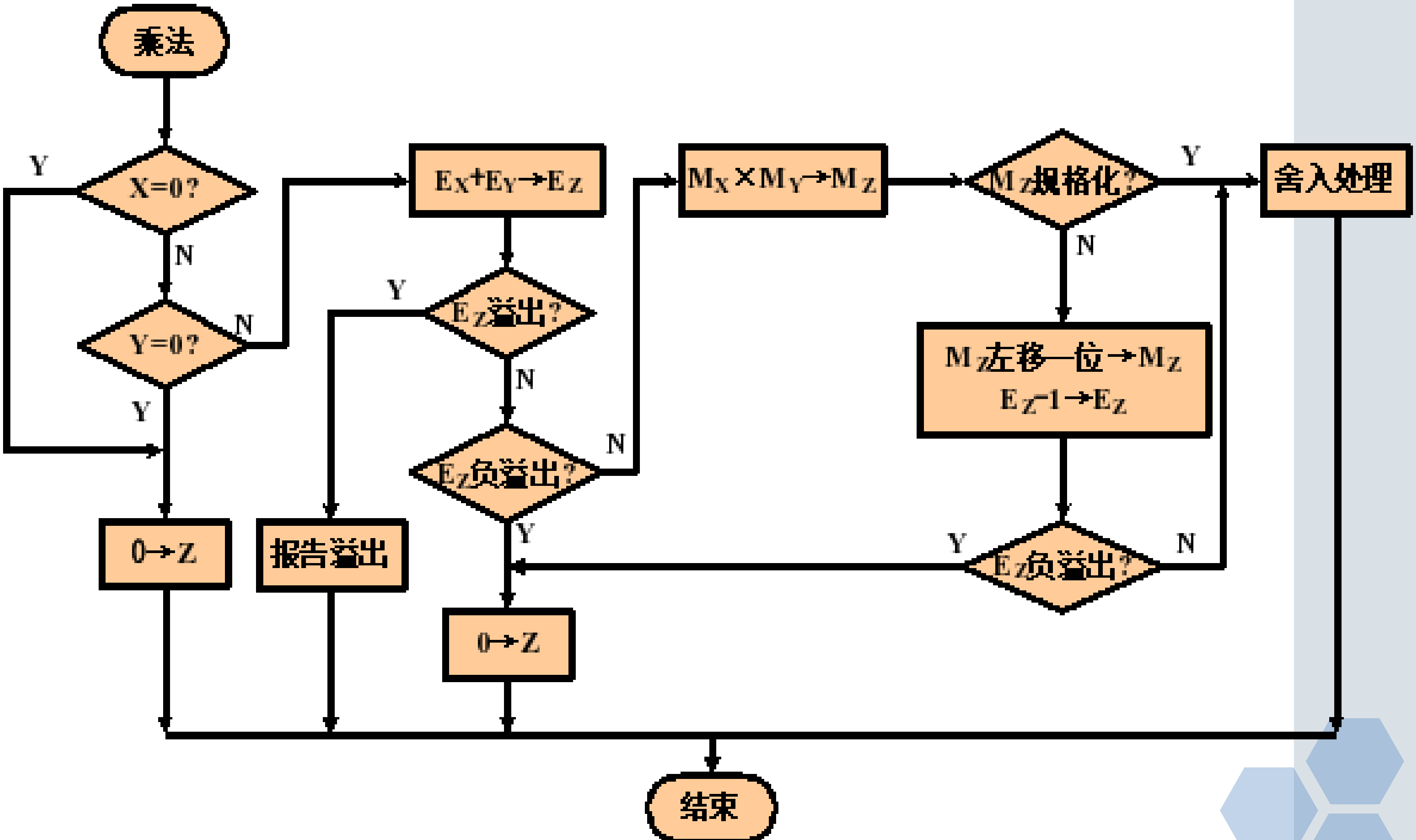
$$X = M_X \times 2^{E_X} \qquad Y = M_Y \times 2^{E_Y}$$

$$Z = X \times Y = (M_X \bullet M_Y) \times 2^{(E_X + E_Y)}$$

浮点乘法运算步骤

- （1） 0操作数检查
- （2）阶码相加：阶码相加可以采用补码或者移码的定点整数加法，同时对相加结果判溢，一旦发生正溢出，则需报告溢出，若发生负溢出，则将结果置为机器零。
- （3）尾数相乘
- （4）结果规格化：可能需要左规1位
- （5）舍入处理：尾数相乘的结果长度是尾数长度的两倍，必须对低位舍入。

浮点数乘法运算流程


乘法
X=0?
Y
N
Y=0?
N
Y
0→Z
$E_X+E_Y \rightarrow E_Z$
E_Z溢出?
Y
N
报告溢出
E_Z负溢出?
N
Y
0→Z
$M_X \times M_Y \rightarrow M_Z$
M_Z规格化?
Y
N
舍入处理
M_Z左移一位→M_Z
$E_Z-1 \rightarrow E_Z$
E_Z负溢出?
Y
N
结束

二、浮点乘法运算（举例）

- 一浮点数表示格式为：10 位浮点数，阶码 4 位，包含 1 位阶符，用移码表示，尾数 6 位，包含 1 位数符，用补码表示，阶码在前，尾数（包括数符）在后，已知：

$X=(-0.11001)\times 2^{011}$　　$Y=0.10011\times 2^{-001}$，求 Z=X·Y 要求阶码用移码计算，尾数用补码 Booth 算法计算。

- 解：按照浮点数的格式分别写出它们的表示形式，为计算方便，阶码采用双符号位移码，尾数采用双符号位补码：
- $[X]_{浮}$ = 01，011　11.00111
- $[Y]_{浮}$ = 00，111　00.10011

二、浮点乘法运算（举例）

1. 阶码相加

$[E_Z]_{移} = [E_X]_{移} + [E_Y]_{补}$

$= 01,011 + 11,111$

$= 01,010$ 无溢出

$[E_Z]_{移} = 1，010$

2. 尾数相乘

采用补码Booth算法计算 $[M_X \cdot M_Y]_{补}$：

11.00111

$[M_Z]_{补} = 1.10001\ 00101$

部分积	乘数Y ($Y_n Y_{n+1}$)	操作说明
00.00000	0.1 0 0 1 1 0	$Y_5Y_6=10$，$+[-X]_{补}$
+ 00.11001		
00.11001		
00.01100	1 0.1 0 0 1 1	右移一位
+ 00.00000		$Y_4Y_5=11$，$+0$
00.01100		
00.00110	0 1 0. 1 0 0 1	右移一位
+ 11.00111		$Y_3Y_4=01$，$+[X]_{补}$
11.01101		
11.10110	1 0 1 0. 1 0 0	右移一位
+ 00.00000		$Y_2Y_3=00$，$+0$
11.10110		
11.11011	0 1 0 1 0. 1 0	右移一位
+ 00.11001		$Y_1Y_2=10$，$+[-X]_{补}$
00.10100		
00.01010	0 0 1 0 1 0. 1	右移一位
+ 11.00111		$Y_0Y_1=01$，$+[X]_{补}$
11.10001	0 0 1 0 1	

二、浮点乘法运算（举例）

3. 结果规格化

M_Z 左规一次得：$[M_Z]_补$ = 1.00010 01010

E_Z 减 1 得：

$[E_Z]_移$ = 01，010 + 11，111 =01，001

4. 舍入

对尾数 M_Z 进行 0 舍 1 入，最后得

$[Z]_浮$ = 1，001   1.00010

三、浮点除法运算

❖假设两个浮点数 X 和 Y：

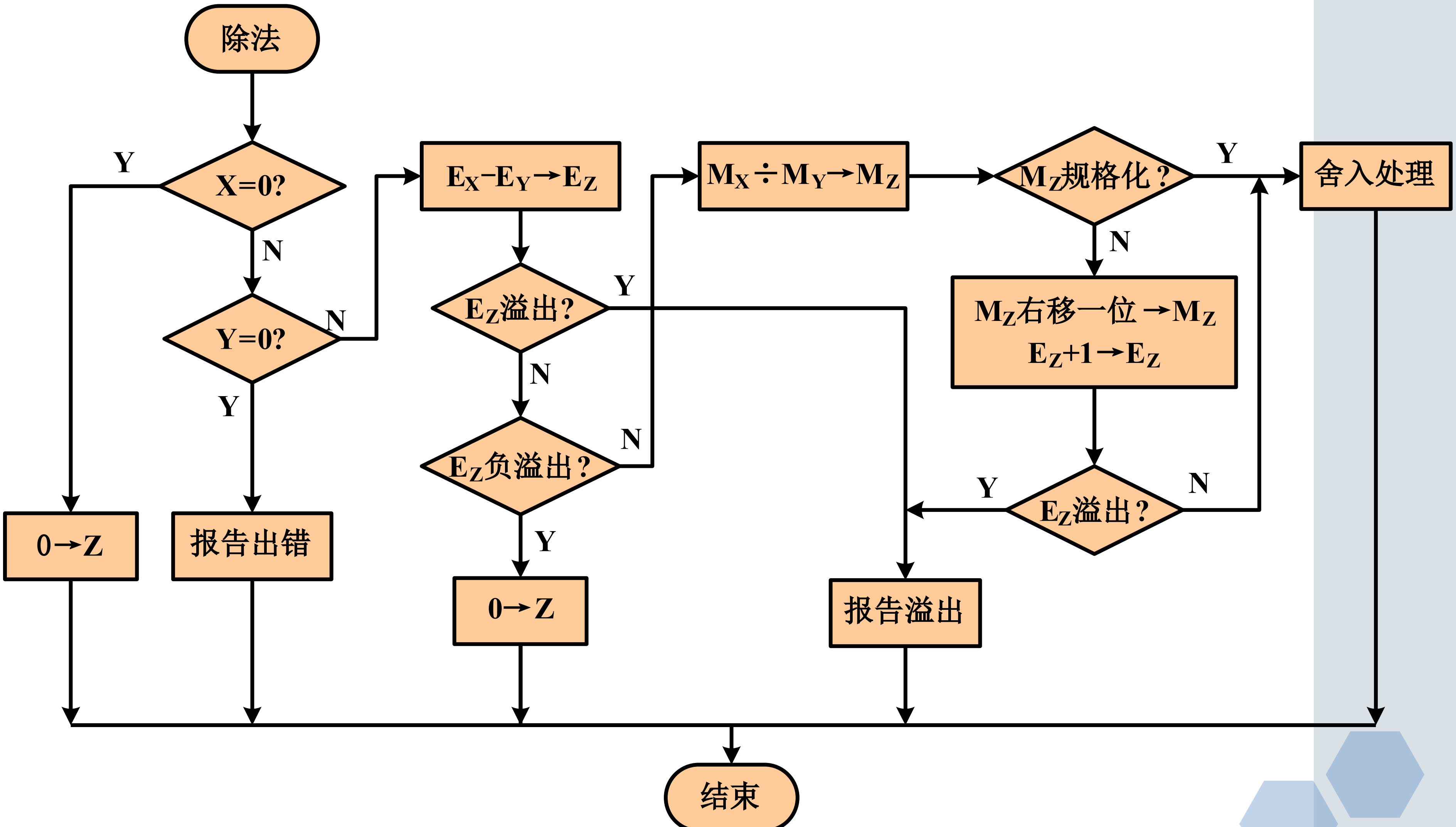
$$X = M_X \times 2^{E_X} \qquad Y = M_Y \times 2^{E_Y}$$

$$Z = X \div Y = (M_X \div M_Y) \times 2^{(E_X - E_Y)}$$

浮点数除法运算步骤

- （1） 0操作数检查
  - 当除数为0，则报告除法出错，或者结果（商）无穷大；当被除数为0，则商为0。
- （2）阶码相减
  - 阶码相减的结果也可能溢出，若发生正溢出，则需报告浮点数溢出，若发生负溢出，则将结果置为机器零。
- （3）尾数相除
- （4）结果规格化
- （5）舍入处理

浮点数除法运算流程



除法
X=0?
Y
N
Y=0?
N
Y
0→Z
报告出错
$E_X - E_Y \to E_Z$
E_Z溢出?
Y
N
E_Z负溢出?
N
Y
0→Z
$M_X \div M_Y \to M_Z$
M_Z规格化?
Y
N
舍入处理
M_Z右移一位→M_Z
$E_Z+1 \to E_Z$
E_Z溢出?
Y
N
报告溢出
结束

三、浮点除法运算（举例）

- 一浮点数表示格式为：10位浮点数，阶码4位，包含1位阶符，用移码表示，尾数6位，包含1位数符，用补码表示，阶码在前，尾数（包括数符）在后，已知：

$X=(-0.11001)\times 2^{011}$　　$Y=0.10011\times 2^{-001}$，求 $Z=X\div Y$。要求阶码用移码计算，尾数用原码加减交替除法计算。

- 按照浮点数的格式分别写出它们的表示形式为：

$[X]_{浮}$ = 1，011　1.00111

$[Y]_{浮}$ = 0，111　0.10011

1. 阶码相减

$[EZ]_{移} = [EX]_{移} + [-EY]_{补}$

= 01，011 + 00，001

= 01，100

三、浮点除法运算（举例）

2. 尾数相除

采用原码加减交替法计算 $|M_X| \div |M_Y|$：

$|M_X| = 00.11001$

$|M_Y| = 00.10011$

被除数 余数	商Q	操作说明
00.11001	0 0 0 0 0 0	
+ 11.01101		$+[-\|M_Y\|]_补$
00.00110	0 0 0 0 0 1	$R_0>0$，上商1
00.01100	0 0 0 0 1.0	左移一位
+ 11.01101		$+[-\|M_Y\|]_补$
11.11001	0 0 0 0 1 0	$R_1<0$，上商0
11.10010	0 0 0 1.0 0	左移一位
+ 00.10011		$+\|M_Y\|$
00.00101	0 0 0 1.0 1	$R_2>0$，上商1
00.01010	0 0 1.0 1 0	左移一位
+ 11.01101		$+[-\|M_Y\|]_补$
11.10111	0 0 1.0 1 0	$R_3<0$，上商0
11.01110	0 1.0 1 0 0	左移一位
+ 00.10011		$+\|M_Y\|$
00.00001	0 1.0 1 0 1	$R_4>0$，上商1
00.00010	1.0 1 0 1 0	左移一位
+ 11.01101		$+[-\|M_Y\|]_补$
11.01111	1.0 1 0 1 0	$R_5<0$，上商0
+ 00.10011		$+\|M_Y\|$ 恢复余数
00.00010		

$|M_Z| = |M_X| \div |M_Y|$
$= 1.01010$

三、浮点除法运算（举例）

3. 结果规格化

由于 $|M_X| > |M_Y|$，所以 $|M_Z|>1$，必须右规一位，得 $|M_Z| = 0.10101\ \ 0$

E_Z 加 1 得：$[E_Z]_移 = 01，100 + 00，001$

$= 01，101$

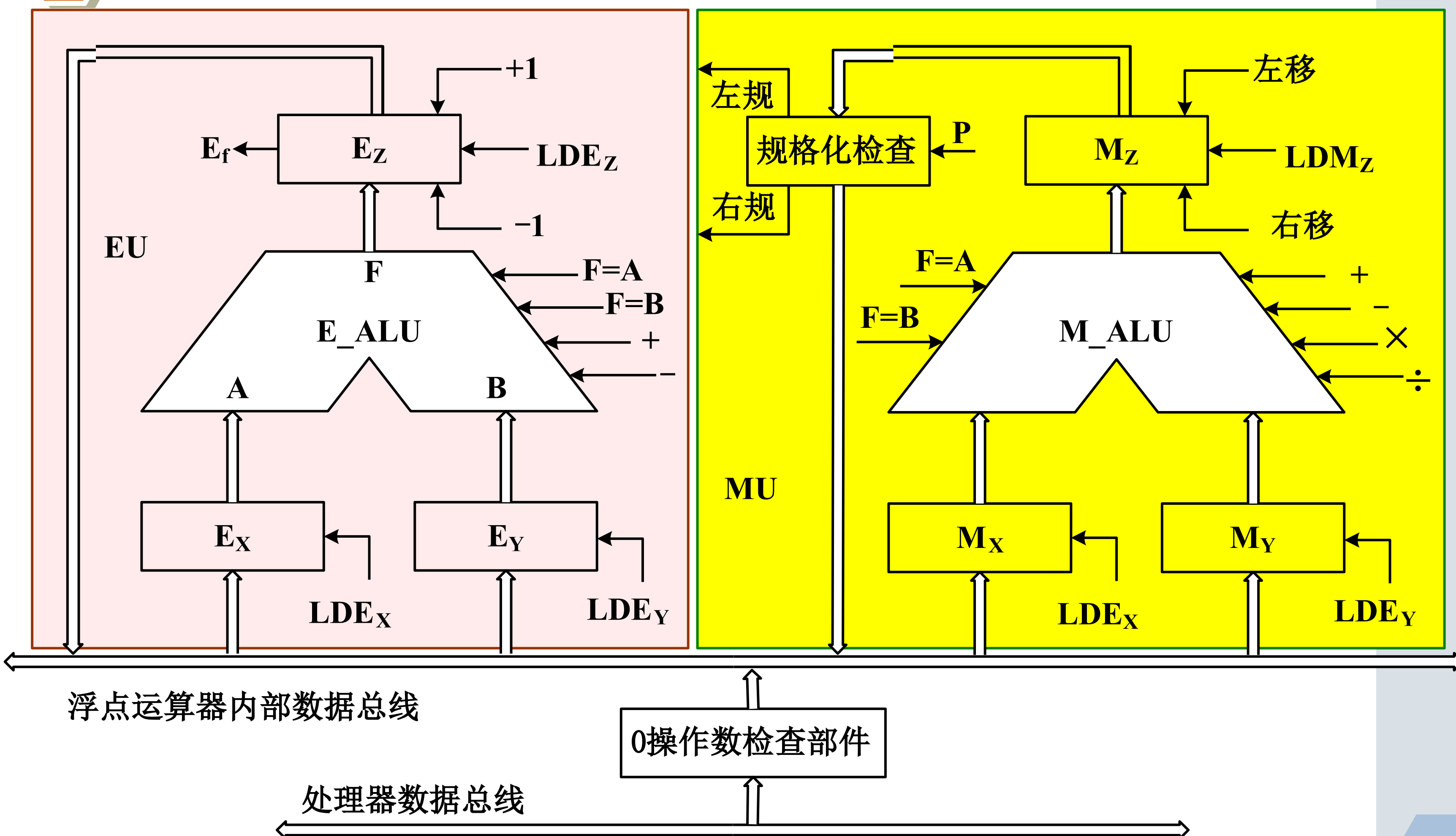
4. 舍入

对 $|M_Z|$ 进行 0 舍 1 入，得 $|M_Z| = 0.10101$

$[M_Z]_原 = 1.10101 \quad [M_Z]_补 = 1.01011$

最后：$[Z]_浮 = 1，101 \quad 1.01011$

四、浮点运算器



+1
Ef
EZ
LDEZ
EU
−1
F
F=A
F=B
E_ALU
+
−
A
B
EX
EY
LDEX
LDEY
左规
右规
规格化检查
P
MZ
左移
LDMZ
右移
F=A
F=B
M_ALU
+
−
×
÷
MU
MX
MY
LDEX
LDEY
浮点运算器内部数据总线
0操作数检查部件
处理器数据总线

四、浮点运算器

- 浮点运算复杂程度远远大于定点运算。
- 根据性能要求和需要来确定是否设置浮点运算器。
- 在**没有浮点运算器**的机器中，基于一定的定点运算部件，可以按照上述浮点运算的算法**用软件来实现**。这种方法速度较慢。
- 浮点运算器**由两个松散连接的定点运算部件组成：阶码运算部件和尾数运算部件**。
  - 阶码运算部件要求具有加减运算和 +1 、 −1 的功能
  - 尾数运算部件要求具有加减乘除四则运算和移位的功能。

The End !