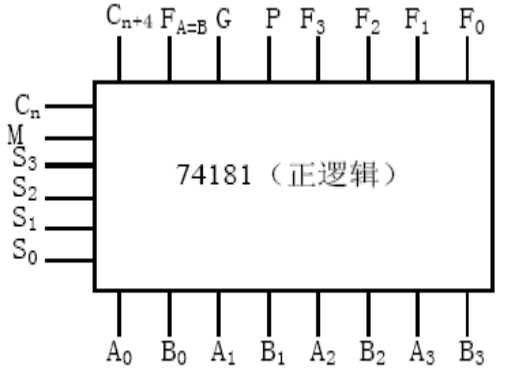
3.2 算术逻辑单元和定点运算器

3.2.1 单元电路

1. 寄存器
   - 8D 锁存器
   - 三态锁存器
2. 移位寄存器

3.2.2 算术逻辑单元ALU

1. 4位ALU —— 74181

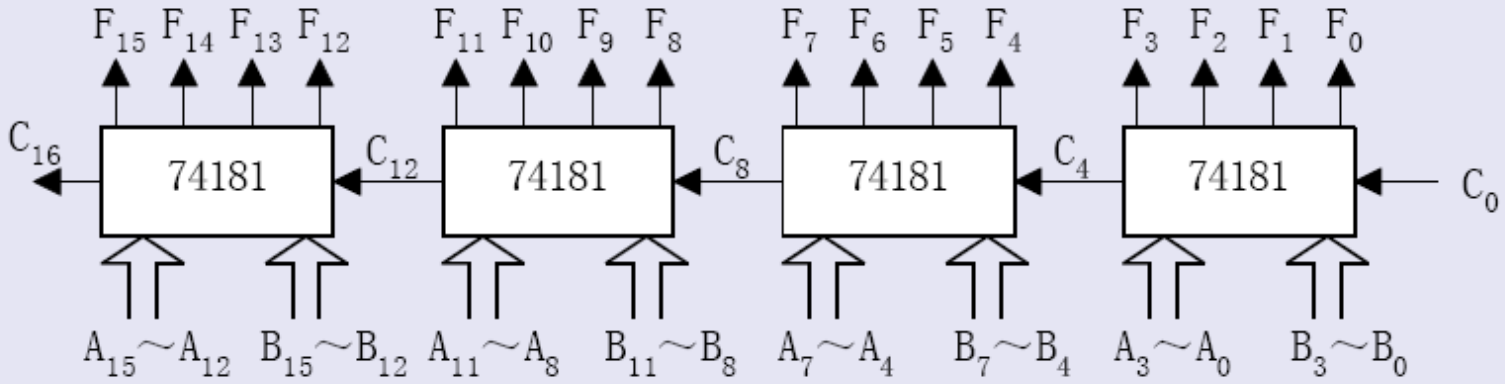


4位ALU74181引 线

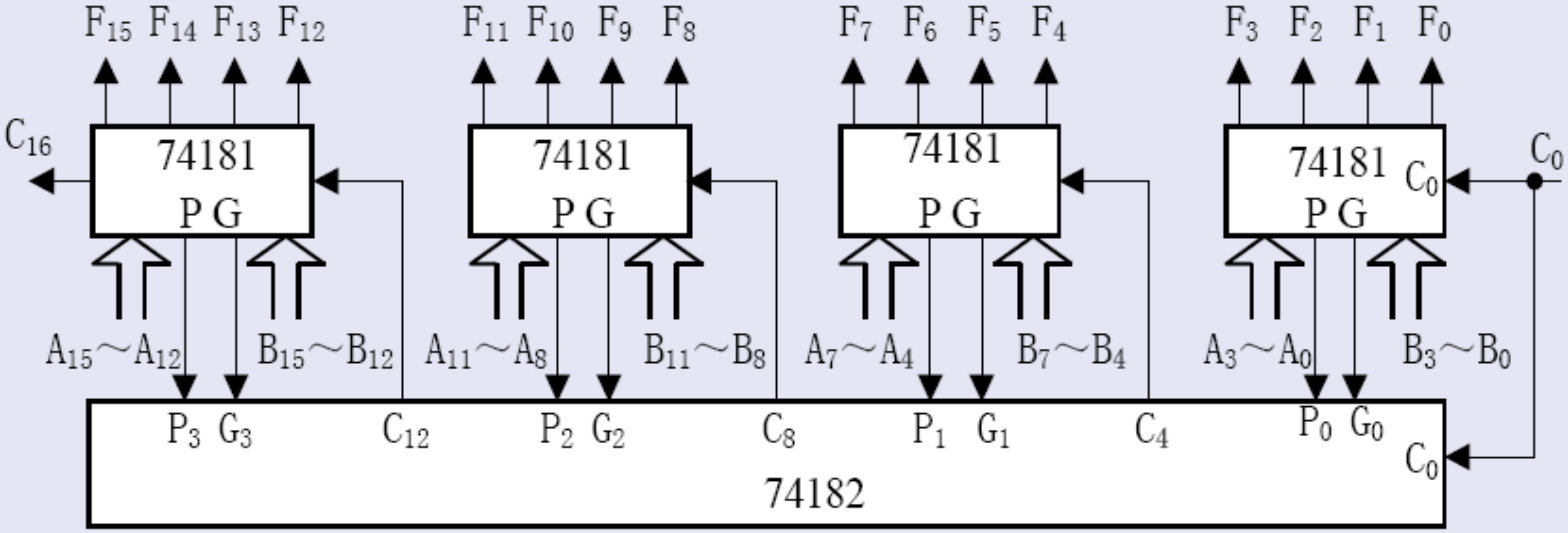
74181 的运算功能

操作选择	运	算	功	能
	M=1	M=0	算 术 运 算	
$S_3S_2S_1S_0$	逻辑运算	Cn=1 无进位	Cn=0 有进位	
0 0 0 0	$F=\overline{A}$	F=A	F=A 加 1	
0 0 0 1	$F=\overline{A+B}$	F=A+B	F=（A+B）加 1	
0 0 1 0	$F=\overline{A}B$	$F=A+\overline{B}$	F=（A+$\overline{B}$）加 1	
0 0 1 1	F=0	F=减 1	F=0	
0 1 0 0	$F=\overline{AB}$	F=A 加 $A\overline{B}$	F=A 加 $A\overline{B}$ 加 1	
0 1 0 1	$F=\overline{B}$	F=（A+B）加 $A\overline{B}$	F=（A+B）加 $A\overline{B}$ 加 1	
0 1 1 0	F=A ⊕ B	F=A 减 B 减 1	F=A 减 B	
0 1 1 1	$F=A\overline{B}$	$F=A\overline{B}$ 减 1	$F=A\overline{B}$	
1 0 0 0	$F=\overline{A}+B$	F=A 加 AB	F=A 加 AB 加 1	
1 0 0 1	$F=\overline{A \oplus B}$	F=A 加 B	F=A 加 B 加 1	
1 0 1 0	F=B	F=（A+$\overline{B}$）加 AB	F=（A+$\overline{B}$）加 AB 加 1	
1 0 1 1	F=AB	F=AB 减 1	F=AB	
1 1 0 0	F=1	F=2A	F=A 加 A 加 1	
1 1 0 1	$F=A+\overline{B}$	F=（A+B）加 A	F=（A+B）加 A 加 1	
1 1 1 0	F=A+B	F=（A+$\overline{B}$）加 A	F=（A+$\overline{B}$）加 A 加 1	
1 1 1 1	F=A	F=A 减 1	F=A	

2. 级联工作



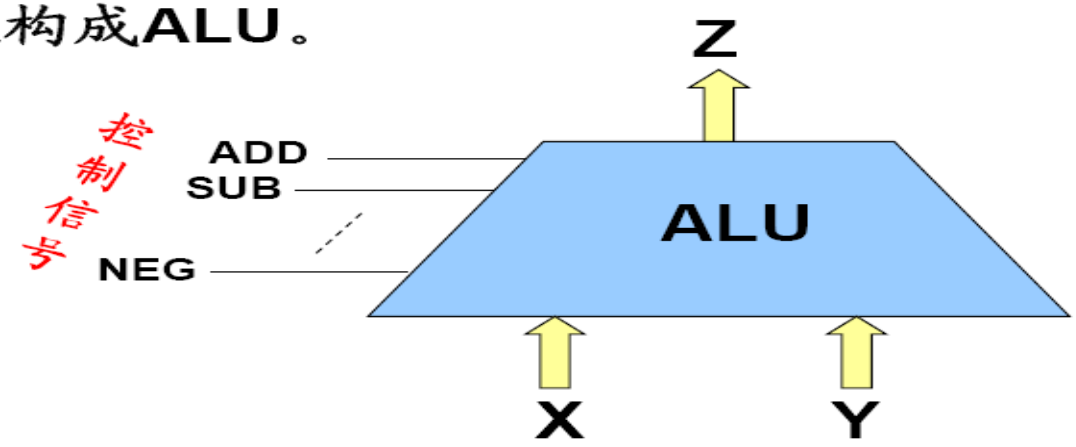
由74181构成组间串行进位的16位ALU



由74181和74182构成组内组间均并行进位的16位ALU

- **ALU**

将加减器、乘法器、除法器、移位器、与/或/非/异或逻辑部件、计数器、求补器等集合在一起构成ALU。



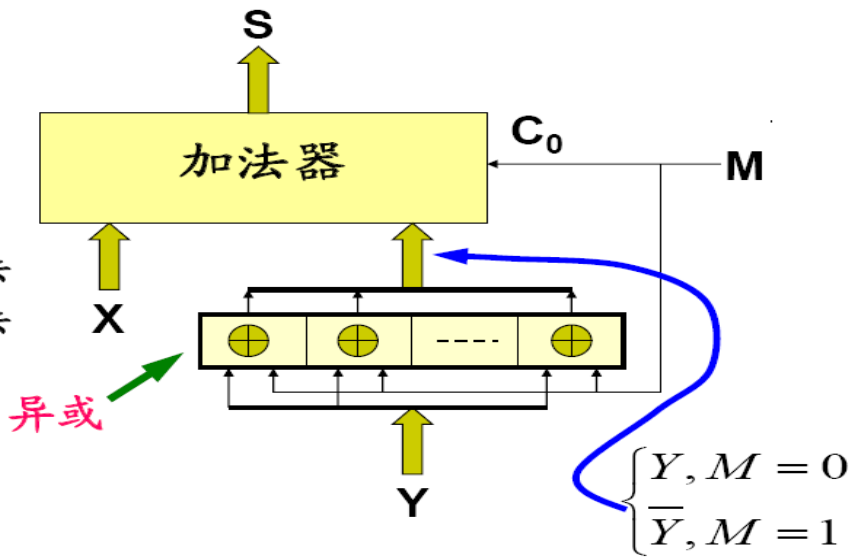
例如：

- 加减器

S=X ± Y

M=0，加法

M=1，减法



3.2.3 运算器的结构

1. 定点运算器的组成

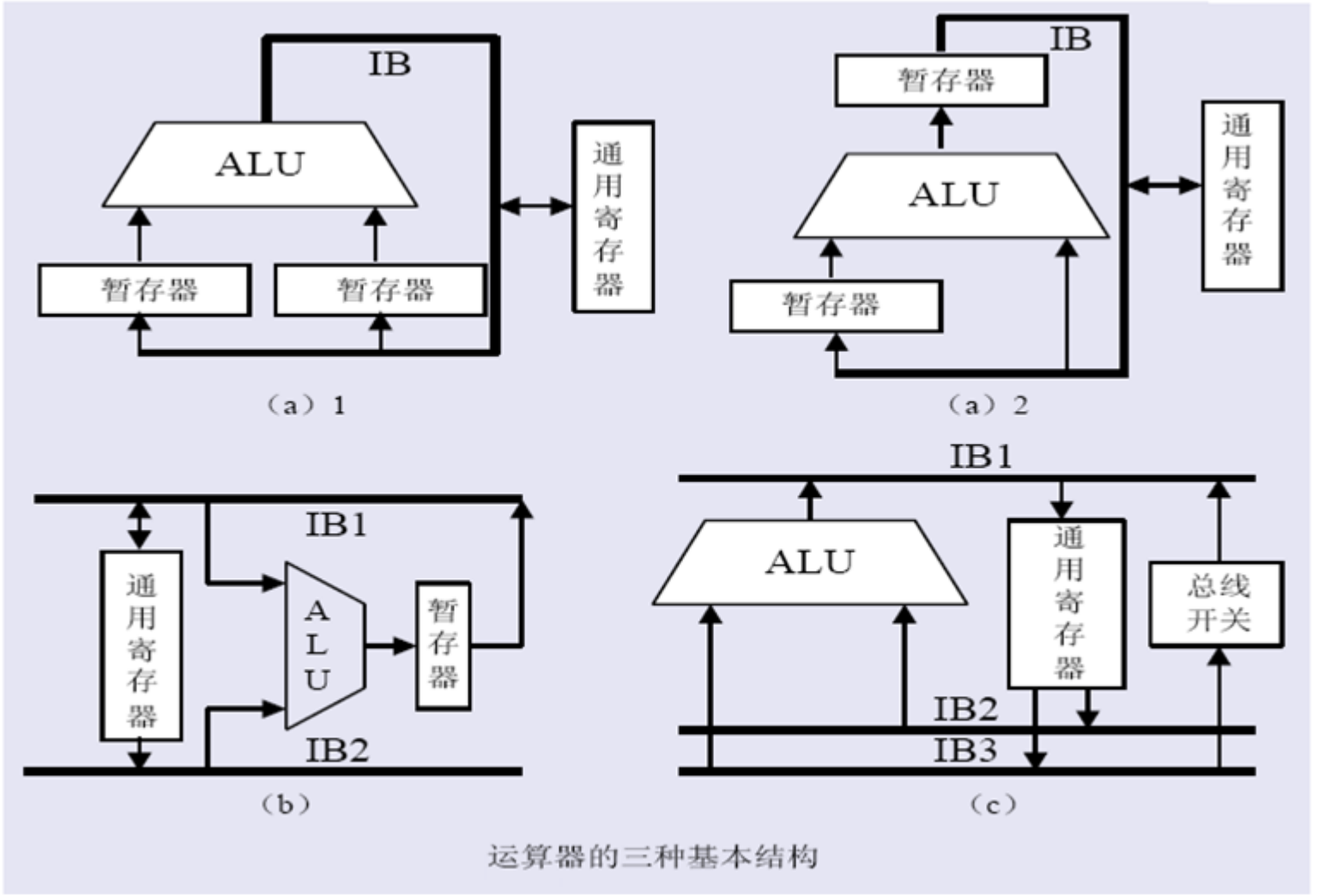
- 基本组成包括：
  - 算术逻辑运算单元ALU：核心部件，实现算术运算和逻辑运算
  - 暂存器：用来存放参与计算的数据及运算结果，它只对硬件设计者可见，即只被控制器硬件逻辑控制或微程序所访问
  - 通用寄存器堆：用于存放程序中用到的数据，它可以被软件设计者所访问。
  - 内部总线：用于连接各个部件的信息通道。
  - 其他可选电路

1. 定点运算器的组成

- 设计定点运算器，如何确定各部件的功能和组织方式是关键，这取决于以下几个方面：
  - 指令系统
  - 机器字长
  - 机器数及其运算原理
  - 体系结构

- 单总线结构运算器
- 双总线结构运算器
- 三总线结构运算器

单总线、双总线、三总线结构运算器



运算器的三种基本结构

单总线、双总线、三总线结构运算器

除了图中所表示的结构外，还有其他类似的连接形式。

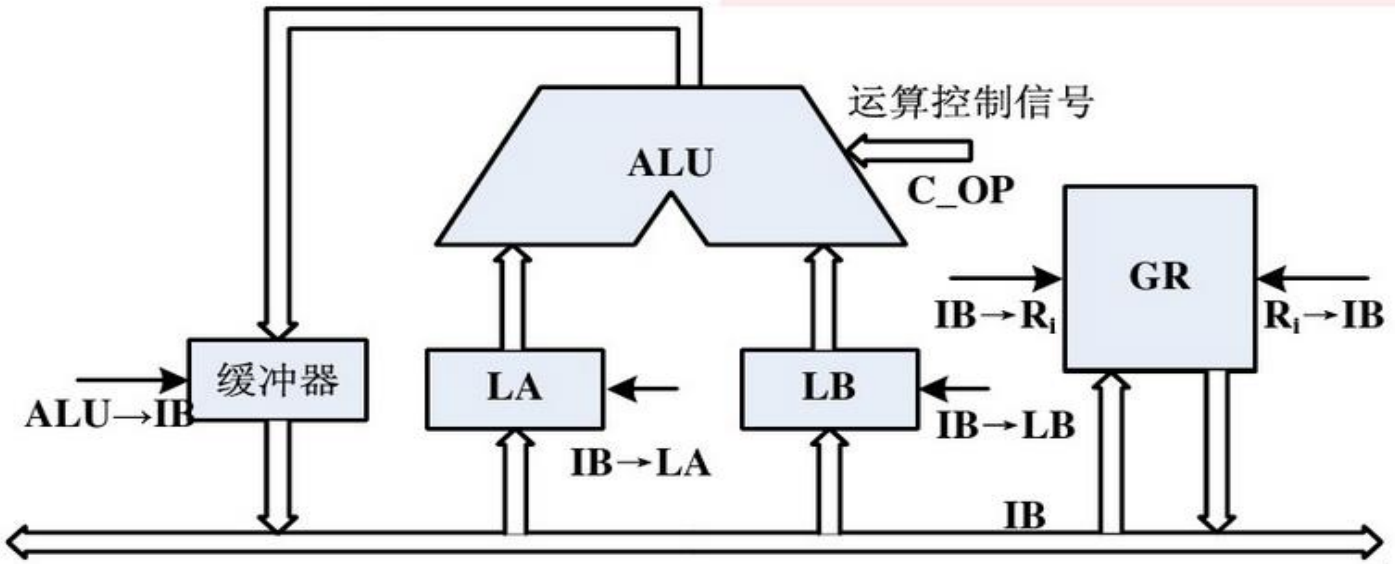
需要强调的是：

- 总线总是分时工作的，即在任何时刻只允许传送一个部件的信号。也就是说任何时侯只允许一个器件将其信号输出加到总线上。多于一个器件必然引起总线竞争。
- 同一个功能部件一次只能做一件事。如ALU可以完成加、减、与、或等多种功能。但某一时刻只能完成一种功能，做加法时不可能同时做与运算。
- 在双总线及三总线结构的运算器中需要多端口器件。

单总线结构运算器(1)

- 单总线运算器的结构形式1：ALU+2个暂存器

❖ （Ri） θ （Rj）→Rk：需要3步
- （Ri）→LA；
- （Rj）→LB；
- ALU运算，结果→Rk

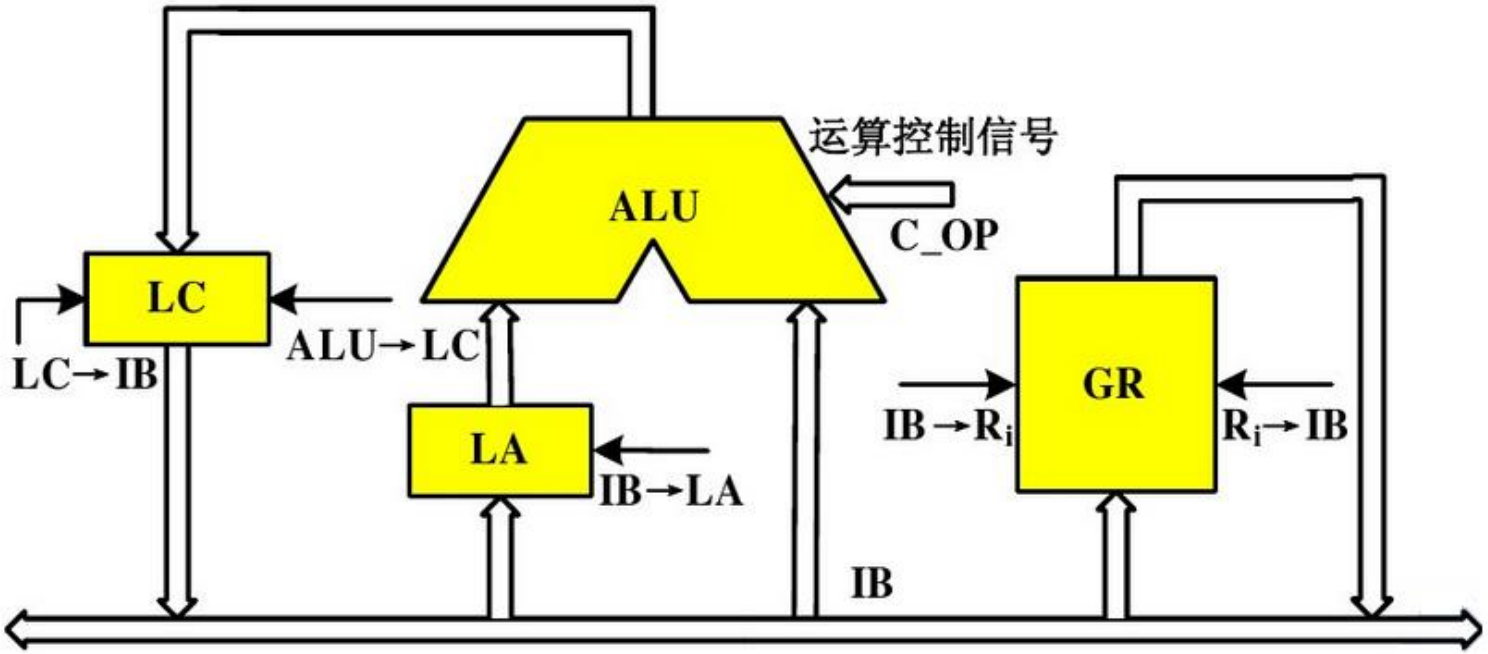


必须指出的是，在分析某一种运算器的运算过程和通路时，一个基本的原则就是在一个CPU周期（一步）内，某条总线上的数据必须是唯一的，且不能保留（至下一个CPU周期）。

单总线结构运算器(2)

- 单总线运算器的结构形式2 ： ALU+2个暂存器

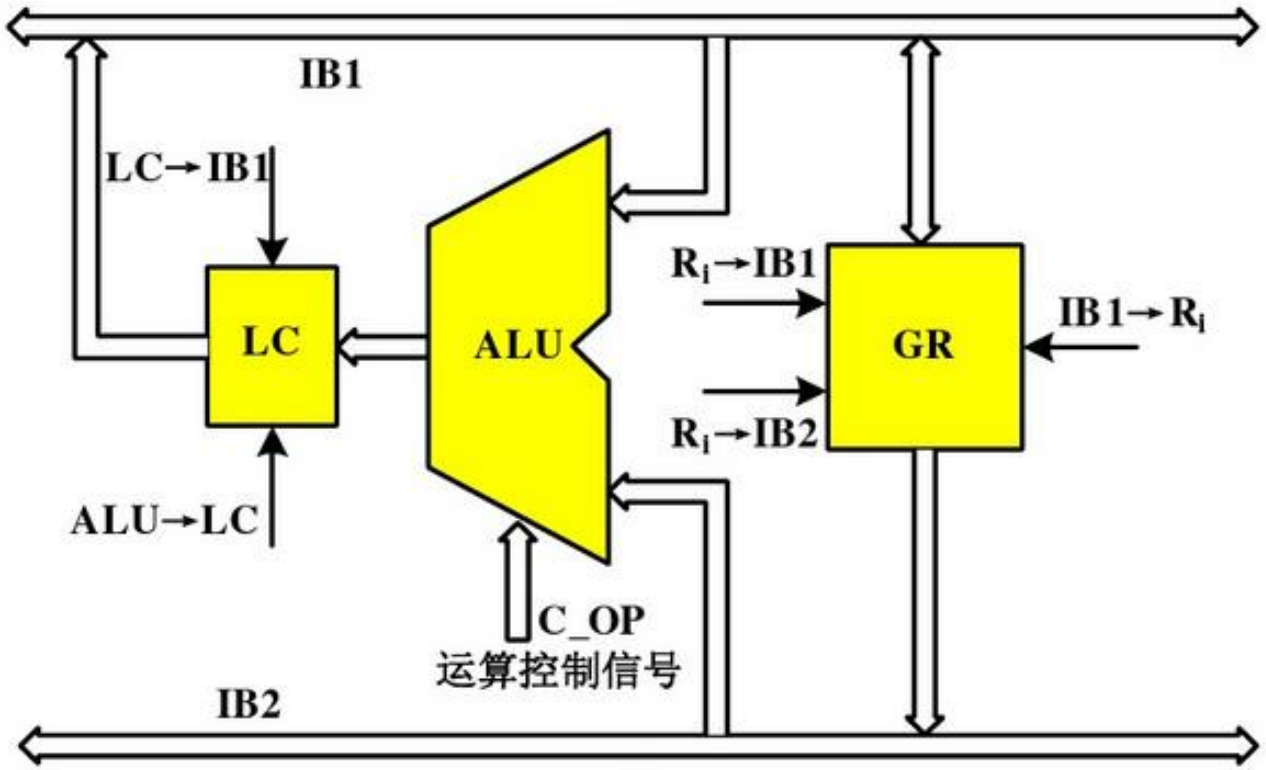
- （Ri） θ （Rj）→Rk：需要3步
  - （Ri）→LA；
  - （Rj）→IB，ALU运算，结果→LC；
  - （LC）→Rk；



双总线结构运算器(1)

双总线运算器的结构形式1：ALU+1个暂存器

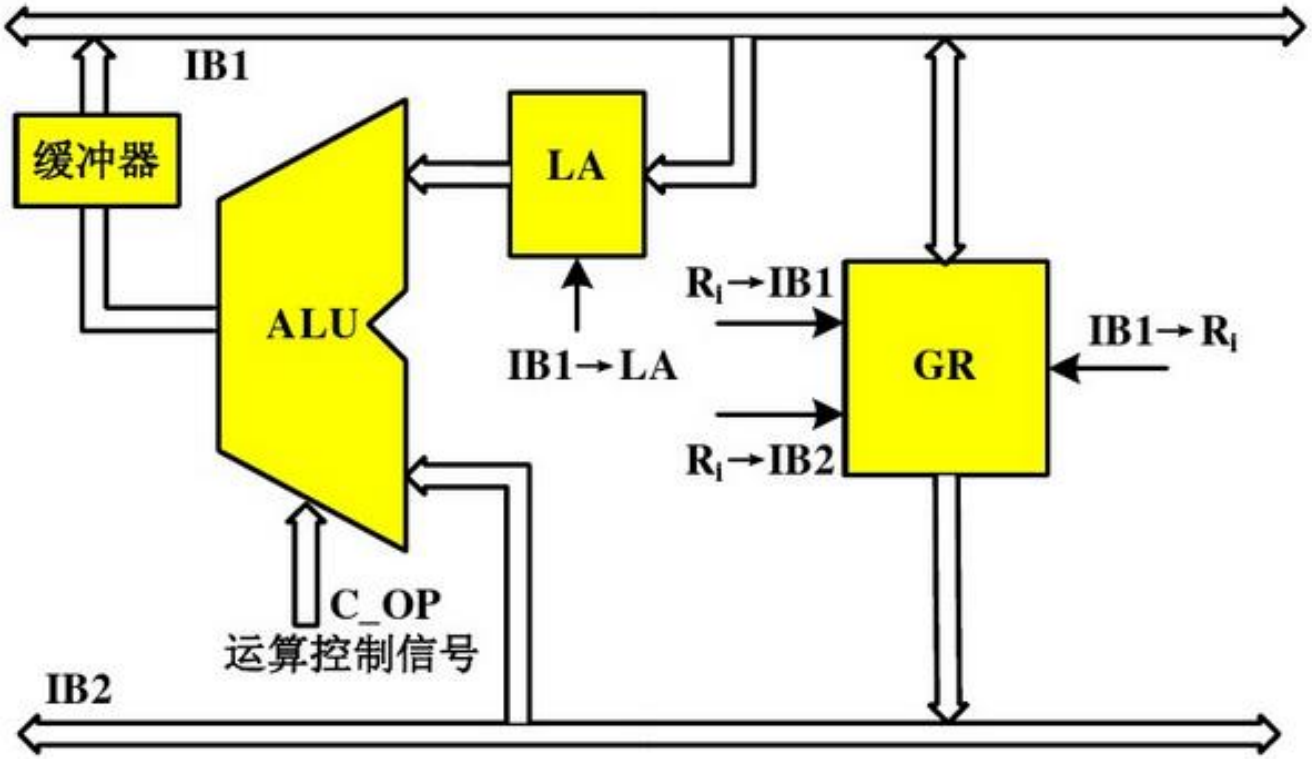
- （Ri）θ（Rj）→Rk：需要2步
  - （Ri）→IB1，（Rj）→ IB2，ALU运算，结果→LC；
  - （LC）→Rk；



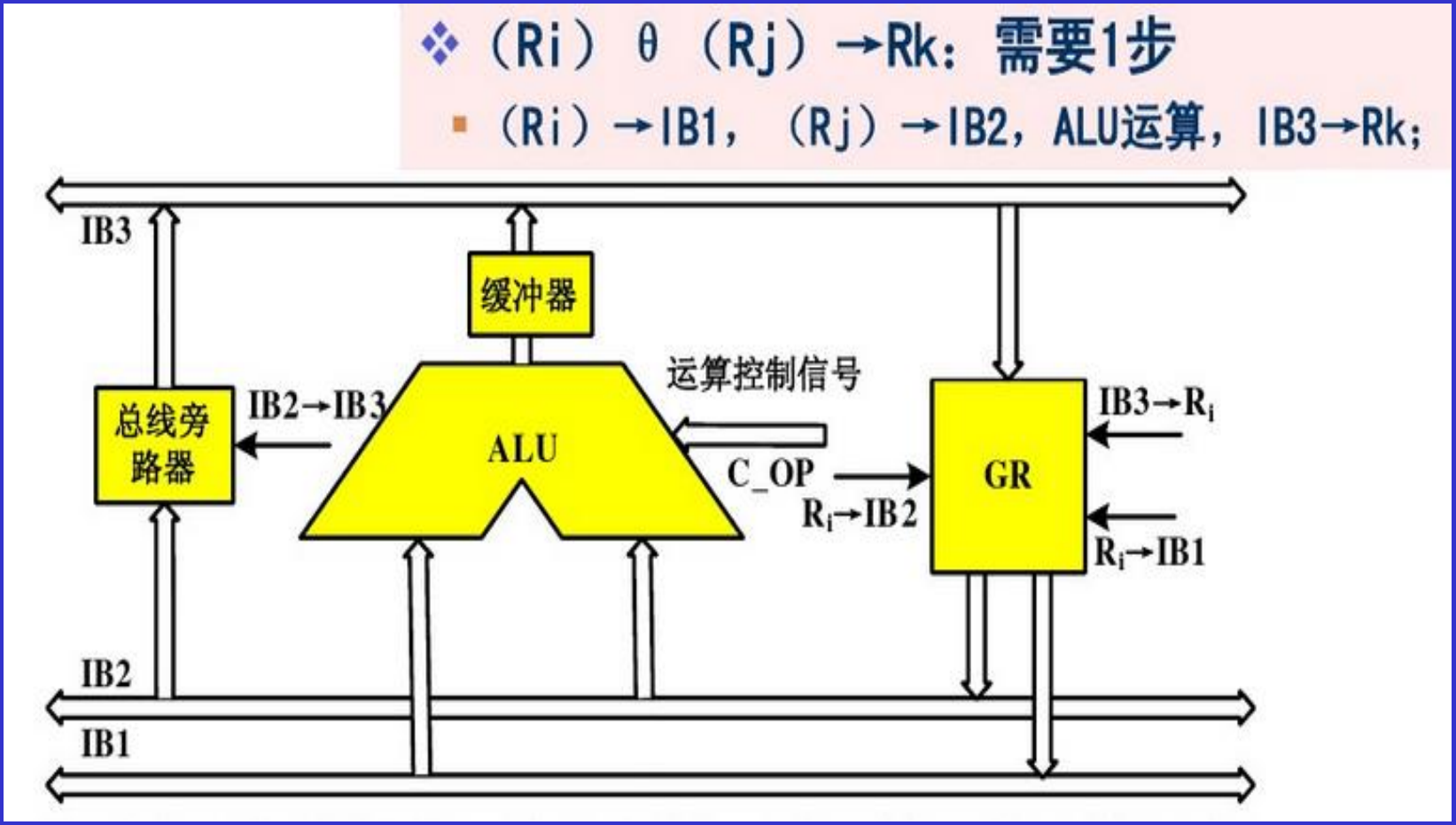
双总线结构运算器(2)

- 双总线运算器的结构形式2 ： ALU+1个暂存器

- （Ri） θ （Rj）→Rk：需要2步
  - （Ri）→LA
  - （Rj）→ IB2，ALU运算，IB1→Rk；



三总线结构运算器


(Ri) θ (Rj) →Rk：需要1步
(Ri) →IB1，(Rj) →IB2，ALU运算，IB3→Rk；
IB3
缓冲器
运算控制信号
总线旁路器
IB2→IB3
ALU
C_OP
GR
IB3→Ri
Ri→IB2
Ri→IB1
IB2
IB1

3.2.3 运算器的结构

2. 标志寄存器、标志位

- 标志寄存器：又称为状态寄存器。
  - 用来保存ALU操作结果的某些状态。
  - 不同CPU，标志寄存器中包含的标志也不尽相同。
- 最基本的5种运算结果标志：
  - ZF：结果为零标志，

$$ZF = \overline{F_n + F_{n-1} + \cdots\cdots + F_0}$$

    - 运算结果为全0，ZF置1
    - 运算结果不全为0， ZF置0。
  - CF：进位/借位标志位，CF标志只对无符号数运算有意义
    - 加法运算时：C=1则CF置1（表示有进位），否则置0；
    - 减法运算时：C=0则CF置1（表示不够减，有借位），否则置0。

$$CF = ADD \bullet C + SUB \bullet \overline{C}$$

2. 标志寄存器、标志位

- **OF**：溢出标志，反映有符号数加减运算所得结果是否溢出；OF标志只对带符号数运算有意义。
  - 运算溢出：OF=1
  - 运算没有溢出：OF=0

$$OF = C \oplus C'$$

- **SF**：符号标志，记录运算结果的符号，=运算结果的最高位。
  - 在现代微机中，有符号数采用补码表示法
  - 运算结果为正数时，SF=0，为负数SF=1。

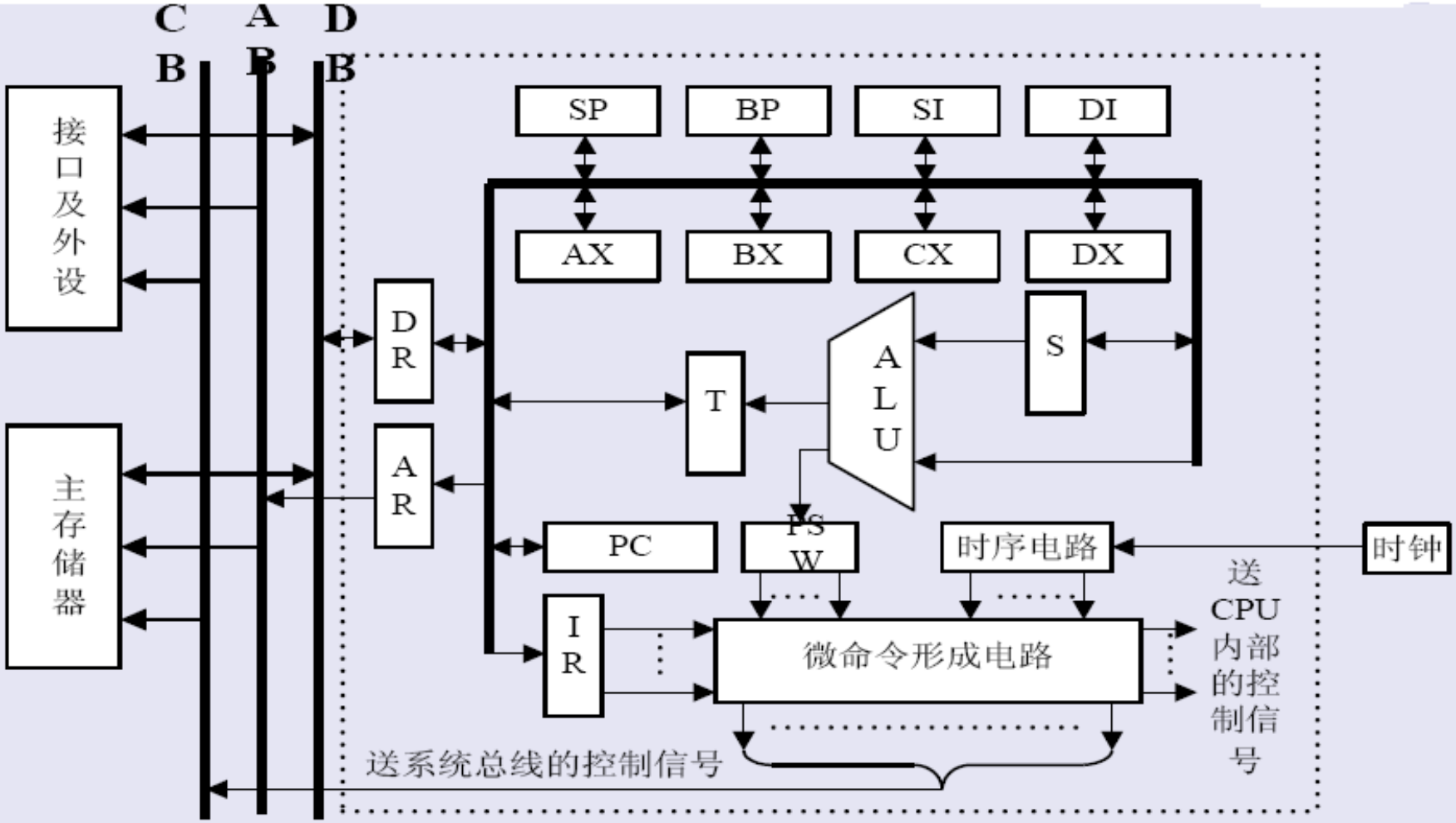
$$SF = F_n$$

- **PF**奇偶标志：反映运算结果中“1”的个数的奇偶性
  - 当结果操作数中“1”的个数为偶数：PF=1
  - 当结果操作数中“1”的个数为奇数：PF=0。

$$PF = \overline{F_n \oplus F_{n-1} \oplus \cdots\cdots \oplus F_0}$$

一种简化的计算机结构图

- 计算机简单框图



某计算机简化框图

本章作业3-3

第13、29题

注：本次作业与下次作业3-4一起交

3.3 浮点运算

- 两个浮点数：$X=Mx\cdot 2^{Ex}$，$Y=My\cdot 2^{Ey}$
- X、Y为规格化浮点数

3.3.1 浮点加减运算

步骤：1. 对阶

2. 尾数加（减）运算

3. 规格化

4. 舍入处理

1. 对阶

- 就是小数点对齐，只有当两者的阶码相同时才能进行加减运算。
- 对阶的原则是小阶对大阶，即小阶码每增加1，其尾数右移一位，直到增大到与大阶码相同。

2. 尾数加（减）运算

- 对阶之后，尾数进行加（减）运算。
- 减法可以用加法实现。

3. 规格化

- 如果结果是非规格化数(尾数和（差）的绝对值可能小于1/2，也可能大于1)，则需要规格化。
- 有两种情况:

（1）左规:

- 如果运算结果尾数为双符号补码11.1XX...X或者是00.0XX...X（尾数未溢出）时，规格化需将尾数左移。每左移一位，阶码减1，直到使尾数成为规格化数为止。
- 阶码减1时，需判断是否下溢。若发生下溢出，认为结果为0。

（2）右规

- 若尾数加/减时，结果（尾数）发生溢出，即出现10.XX…X或者01.XX…X时，表明尾数有溢出，整个浮点数结果未必溢出。
- 可将尾数右移一位，阶码加1，即右规。右规最多1次。
- 阶码加1时，需判断是否上溢。若发生上溢出，认为结果为∞。

4. 舍入处理

在对阶及规格化时需要将尾数右移，右移将丢掉尾数的最低位，这就出现舍入的问题。在进行舍入时，通常可采用下面的方法。

- 截（尾）断法

  此法最简单，就是将需丢弃的尾数低位丢弃。

- 末位恒置1法

  无论尾数右移丢弃的是0还是1，此法将保证要保留的尾数的最低位永远为1。

- 0舍1入法

  当尾数右移丢弃的是1时，要保留的最末位加1；当尾数右移丢弃的是0时，要保留的最末位不变。

  这种方法误差较小。但当遇到 01.111…11 这种需在规的尾数时，采用此法会再次使尾数溢出。遇到这种情况可采用截尾法。

例　两浮点数为：

$X=0.110101\times 2^{-010}$，$Y=-0.101010\times 2^{-001}$

求两数之和及差。

解：设两浮点数阶码为4位，用补码表示。尾数用8位，均用双符号位补码表示。则两数可表示为：

$[X]_{浮}=1110;\ 00.110101$

$[Y]_{浮}=1111;\ 11.010110$

① 对阶

求阶差：$[\triangle E]_{补}=[Ex]_{补}+[-Ey]_{补}=1110+0001=1111$。即X的阶码比Y的阶码小。

因此，X尾数右移一位，使两者阶码相同。这时的X为：

$[X]'_{浮}=1111;\ 00.011011$（0舍1入法）

① 对阶

求阶差：$[\Delta E]_{补}=[Ex]_{补}+[-Ey]_{补}=1110+0001=1111$。即X的阶码比Y的阶码小。

因此，X尾数右移一位，使两者阶码相同。这时的X为：

$[X]'_{浮}=1111$；00.011011（0舍1入法）

②尾数求和：

$$\begin{array}{r} 00.011011 \\ +\ 11.010110 \\ \hline 11.110001 \end{array}$$

尾数求差：

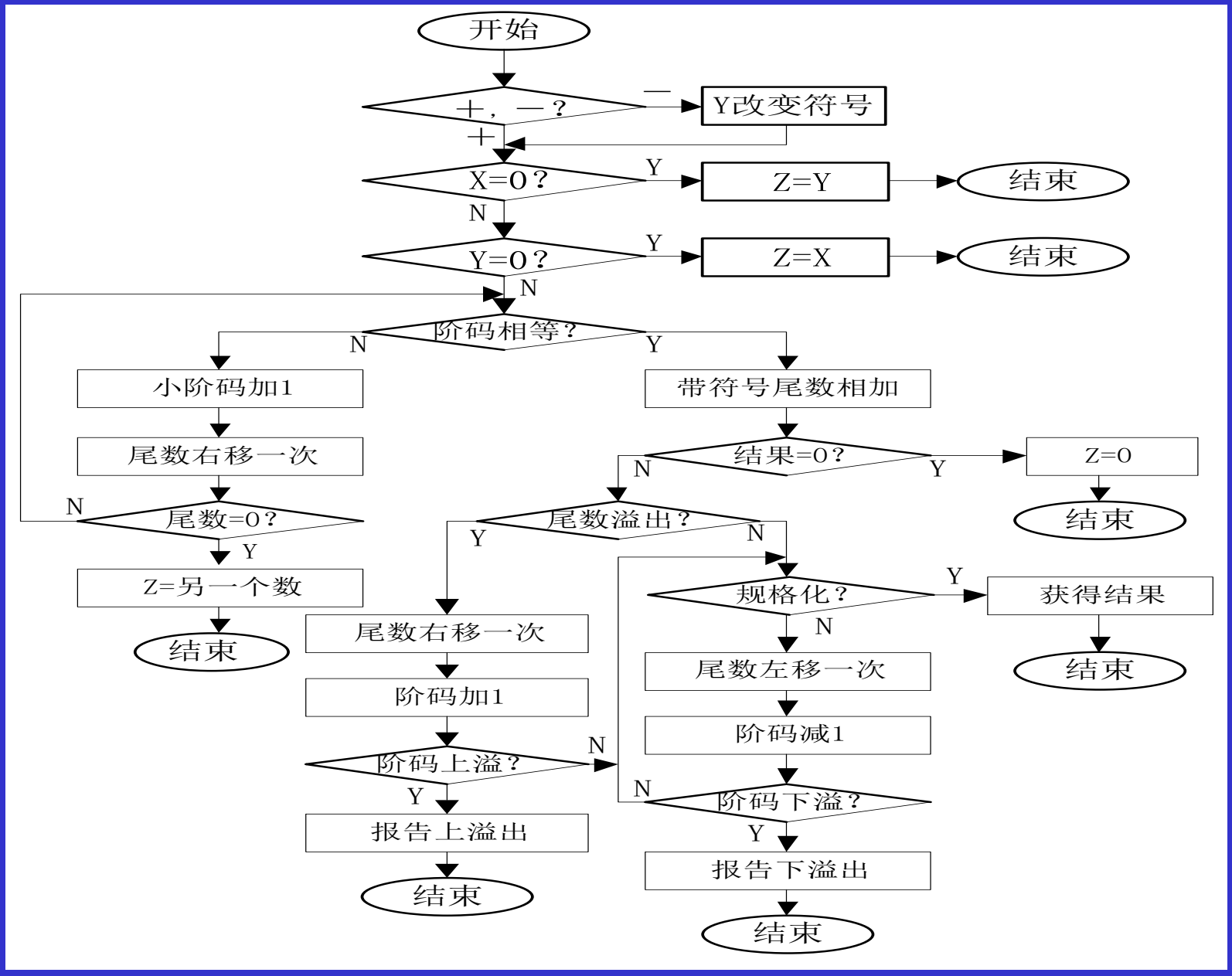
$$\begin{array}{r} 00.011011 \\ +\ 00.101010 \\ \hline 01.000101 \end{array}$$

③ 规格化

相加结果为非规格化尾数，需左规。将尾数左移2位，变为11.000100。同时，阶码减2，则阶码变为1101。两数相加结果为：$[X+Y]_{浮}=1101$；11.000100。

相减结果也为非规格化尾数，需右规。将尾数右移1位，变为00.100011，采用0舍1入法。阶码加1，则阶码为0000。两数相减结果为：$[X-Y]_{浮}=0000$；00.100011。

浮点加减运算流程


开始
+，－？
－
Y改变符号
+
X=0？
Y
Z=Y
结束
N
Y=0？
Y
Z=X
结束
N
阶码相等？
N
Y
小阶码加1
尾数右移一次
N
尾数=0？
Y
Z=另一个数
结束
带符号尾数相加
结果=0？
N
Y
Z=0
结束
尾数溢出？
Y
N
尾数右移一次
阶码加1
阶码上溢？
N
Y
报告上溢出
结束
规格化？
Y
获得结果
结束
N
尾数左移一次
阶码减1
阶码下溢？
N
Y
报告下溢出
结束

3.3.2 浮点乘除运算

1. 浮点乘法运算

- 两个规格化浮点数：$X=Mx\cdot 2^{Ex}$，$Y=My\cdot 2^{Ey}$
- 两浮点数相乘为：$Z=(Mx\cdot My)\,2^{(Ex+Ey)}$。

浮点乘法运算过程

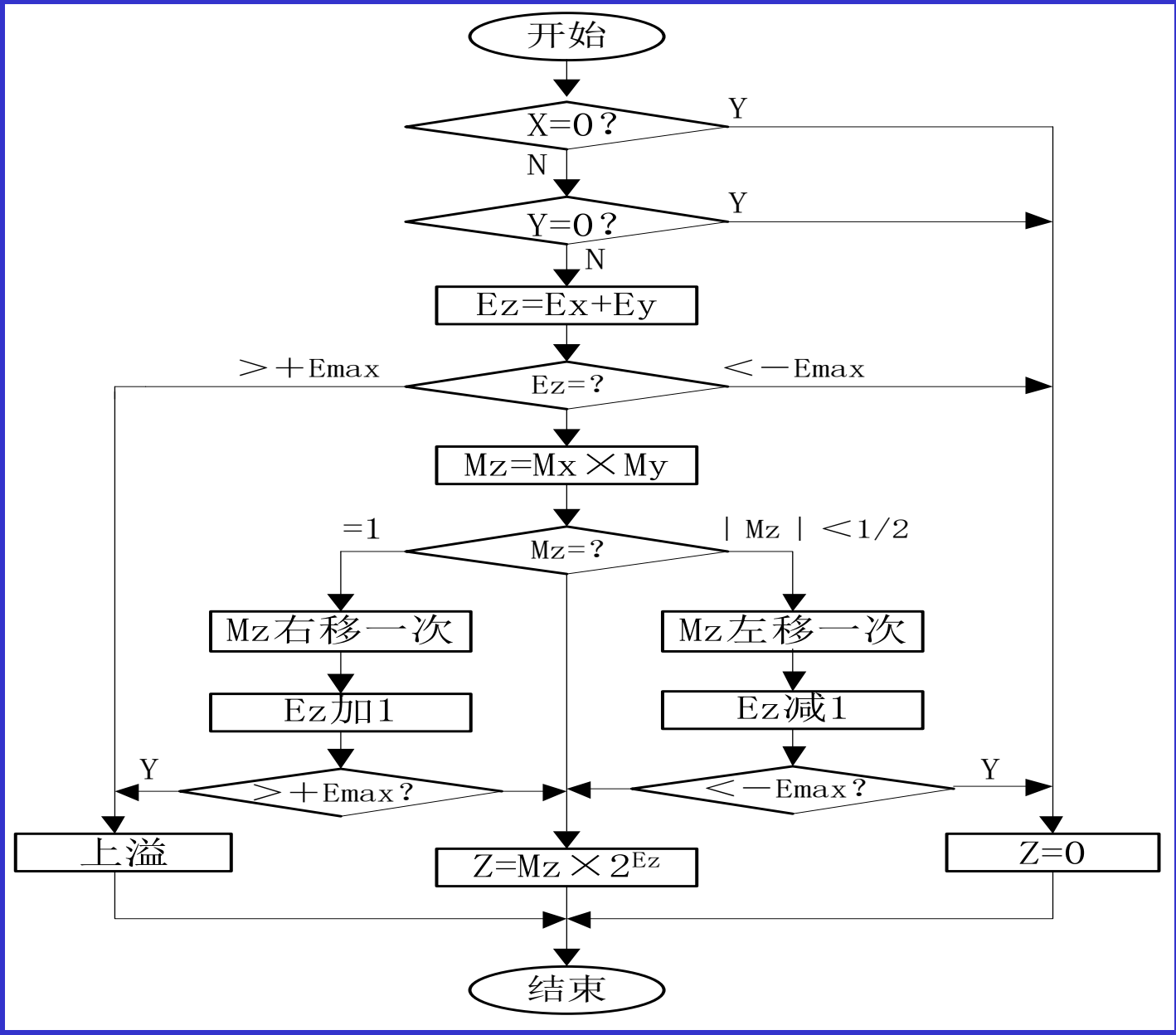
①参与运算的两浮点数一定是规格化数，且不为0。若有一个乘数为0，则乘积必为0。

②求乘积的阶码，即Ez=Ex+Ey，判断积的阶码是否溢出。

③两乘数的尾数相乘。

④规格化乘积的尾数。

浮点乘法运算流程


开始
X=0？
Y
N
Y=0？
Y
N
Ez=Ex+Ey
＞＋Emax
Ez=？
＜－Emax
Mz=Mx×My
=1
Mz=？
| Mz | ＜1/2
Mz右移一次
Mz左移一次
Ez加1
Ez减1
Y
＞＋Emax？
＜－Emax？
Y
上溢
Z=Mz×2Ez
Z=0
结束

2. 浮点除法运算

- 两个规格化浮点数：$X=Mx\cdot 2^{Ex}$，$Y=My\cdot 2^{Ey}$
- 两浮点数相除为：$Z=(Mx \div My)\,2^{(Ex-Ey)}$

浮点除法的运算步骤为：

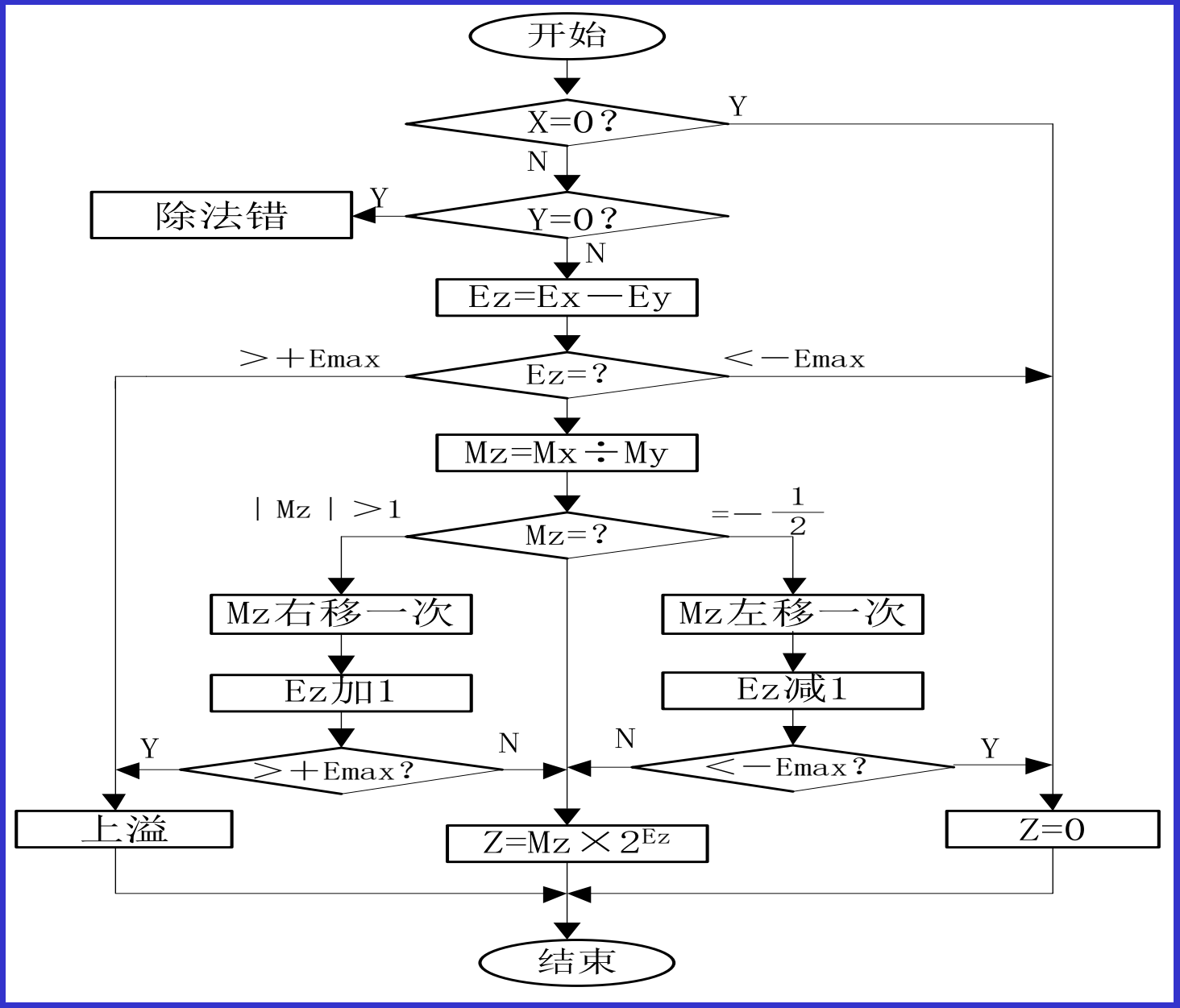
①若被除数为0，商为0。若除数为0，出错处理。

②求商的阶码，即Ez=Ex-Ey，判断商的阶码是否溢出。

③被除数尾数除以除数尾数。

④结果规格化及舍入处理。

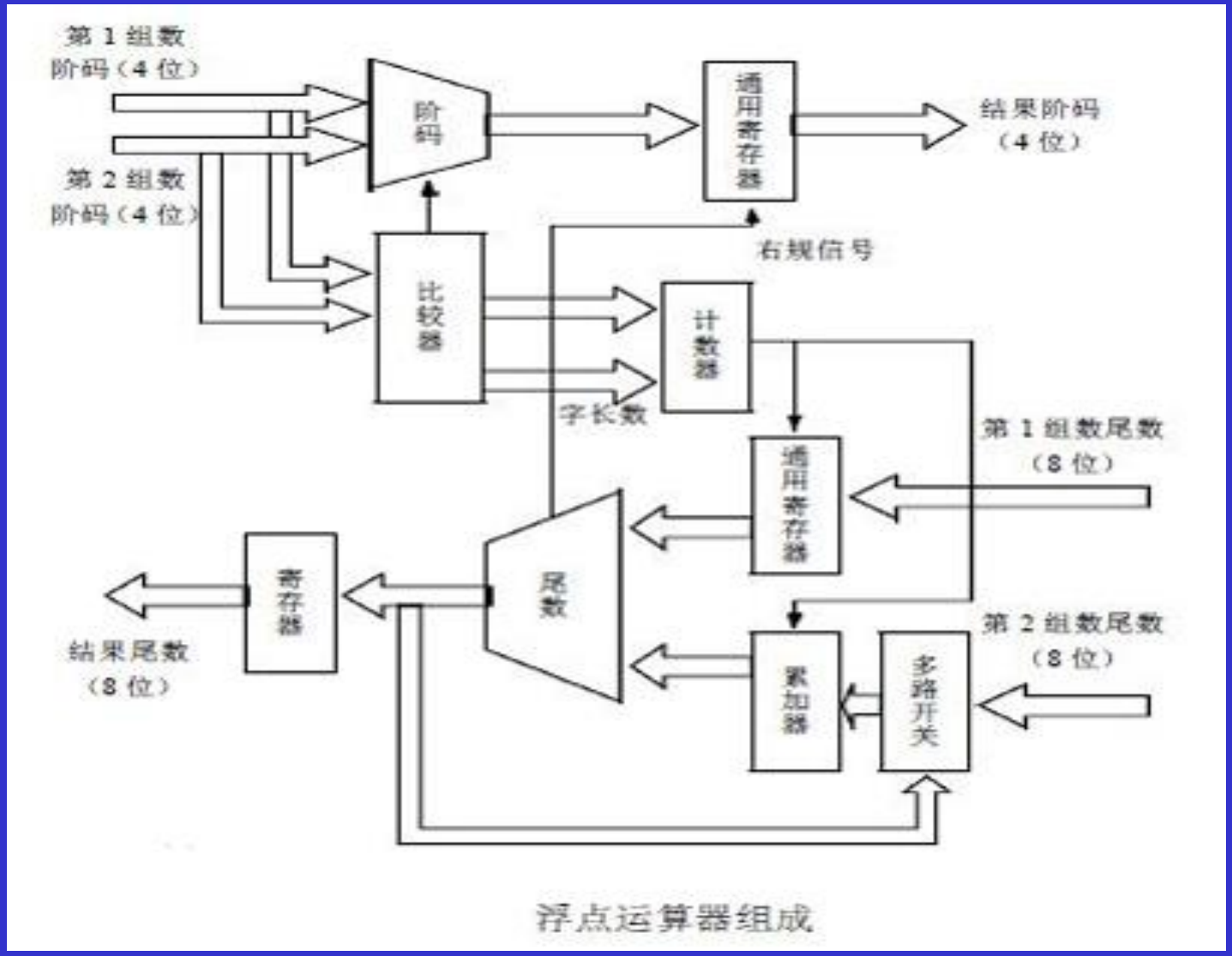
浮点除法运算流程


开始
X=0？
Y
N
除法错
Y
Y=0？
N
Ez=Ex—Ey
＞＋Emax
Ez=？
＜—Emax
Mz=Mx÷My
| Mz | ＞1
Mz=？
=— 1/2
Mz右移一次
Mz左移一次
Ez加1
Ez减1
Y
＞＋Emax？
N
N
＜—Emax？
Y
上溢
Z=Mz×2^Ez
Z=0
结束

3.3.3 浮点数运算的实现

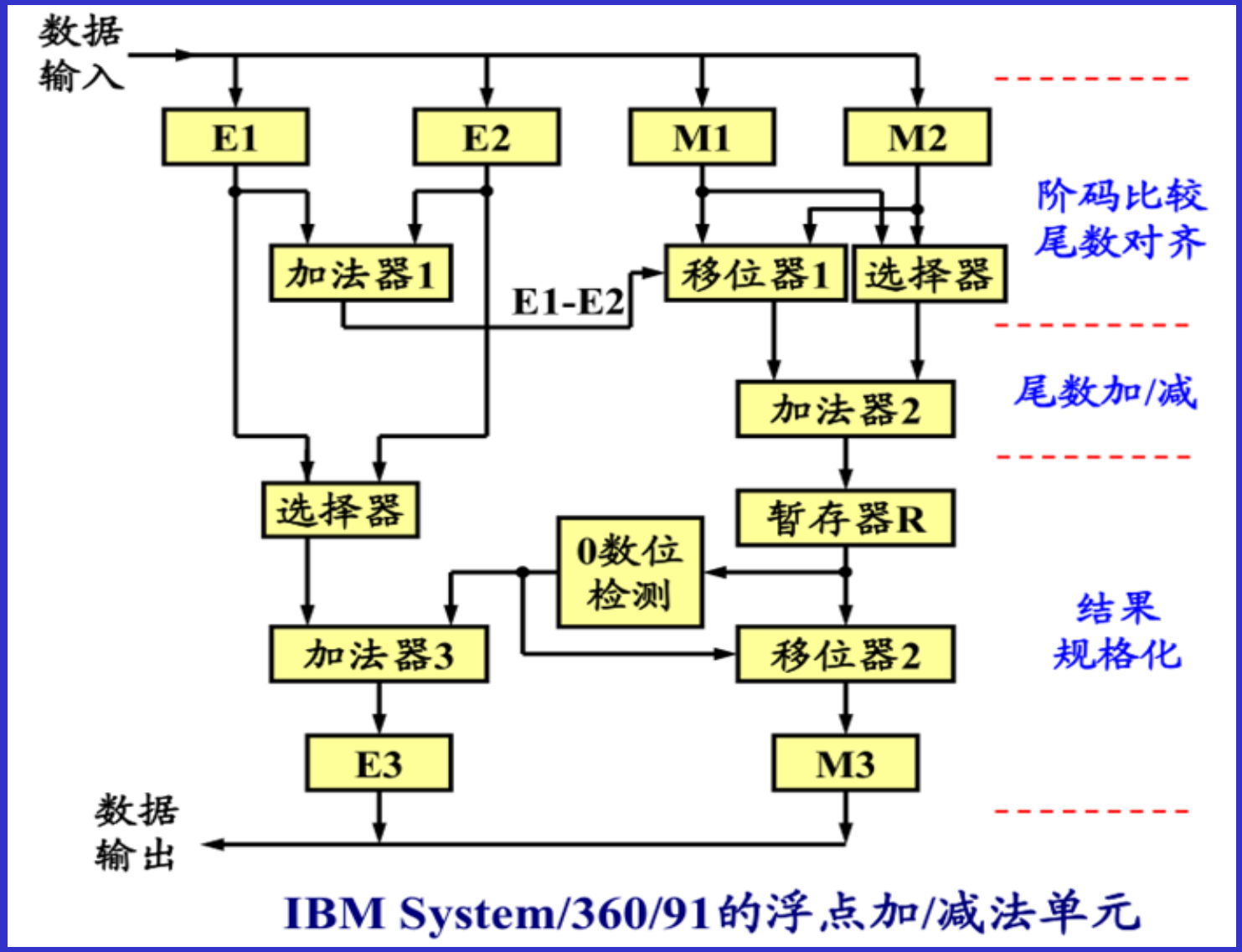
- 软件方法
- 配专用浮点处理器
- 在处理器中设置浮点运算部件
- 浮点运算的流水线处理

例：


第 1 组数
阶码（4 位）
第 2 组数
阶码（4 位）
阶码
通用寄存器
结果阶码
（4 位）
右规信号
比较器
计数器
字长数
第 1 组数尾数
（8 位）
通用寄存器
寄存器
尾数
结果尾数
（8 位）
第 2 组数尾数
（8 位）
累加器
多路开关

浮点运算器组成

例：



IBM System/360/91的浮点加/减法单元

本章作业3-4（第5次作业）

第 10 题、第 26（1）题

注：与上次作业3-3一起交

本章作业3-3

第 13、29题

注：本次作业与下次作业3-4一起交