中国科学院大学
University of Chinese Academy of Sciences

B0911006Y-01　2024-2025学年春季学期

计算机组成原理

Principles of Computer Organization

第 11 讲　计算机中数的运算V

各种除法、浮点四则运算

主讲教师：石　侃
shikan@ict.ac.cn

2025年4月7日

(1) 恢复余数法

例6.24 $x = -0.1011$ $y = -0.1101$ 求 $[\frac{x}{y}]_{原}$

解：$[x]_{原} = 1.1011$ $[y]_{原} = 1.1101$ $[y^*]_{补} = 0.1101$ $[-y^*]_{补} = 1.0011$

① $x_0 \oplus y_0 = 1 \oplus 1 = 0$

②

被除数（余数）	商	说　明
0.1011	0.0000	
+ 1.0011		$+[-y^*]_{补}$ (进行x^*-y^*作比较)
1.1110	0	余数为负，上商 0
+ 0.1101		恢复余数　$+[y^*]_{补}$
0.1011	0	恢复后的余数
逻辑左移 1.0110	0	←1 (余数和商同时左移)
+ 1.0011		$+[-y^*]_{补}$
0.1001	01	余数为正，上商 1
逻辑左移 1.0010	01	←1
+ 1.0011		$+[-y^*]_{补}$

例6.24(续)

ACC

回顾第一章除法操作中的寄存器(表1.3)

6.3

被除数（余数）	商 MQ	说　明
0.0101	011	余数为正，上商 1
0.1010	011	← 1　逻辑左移
+ 1.0011 X		$+[-y^*]_{补}$
1.1101	0110	余数为负，上商 0
+ 0.1101		恢复余数 $+[y^*]_{补}$
0.1010	0110	恢复后的余数
1.0100	0110	← 1　逻辑左移
+ 1.0011		$+[-y^*]_{补}$
0.0111	01101	余数为正，上商 1

这里都是移位后的余数的绝对值。因此需确保这些值要恢复为正数且最后为正。最后一步得到的是真正余数的绝对值R*左移4位，而R的实际符号位应与被除数一致，即从被除数直接获得，所以真正余数应为 -0.00000111

为什么是逻辑左移，而不是算术左移？

因为这里的移动是表示竖式运算过程中的移动，而不是数值变化

$$\frac{x^*}{y^*} = 0.1101$$

上商 5 次　移 4 次

$$\therefore \left[\frac{x}{y}\right]_{原} = 0.1101$$

对于小数除法，第一次上商判溢出

余数为正　上商 1

余数为负　上商 0，恢复余数

	ACC	MQ	X
加法	被加数 和		加数
减法	被减数 差		减数
乘法	乘积高位	乘数 乘积低位	被乘数
除法	被除数 余数	商	除数

(2) 不恢复余数法（原码加减交替除法）6.3

- 恢复余数法运算规则

对R_i左移一位后减去除数

余数 $R_i>0$ 上商“1”， $2R_i-y^*$

余数 $R_i<0$ 上商“0”， R_i+y^*

先恢复余数

$$2(R_i+y^*)-y^*=2R_i+y^*$$

再左移一位后减去除数

- 不恢复余数法运算规则

余数 $R_i>0$ 上商“1” $2R_i-y^*$ 正、1、移、减

余数 $R_i<0$ 上商“0” $2R_i+y^*$ 负、0、移、加

加减交替(注: 一减后面不一定是一加)

例6.25 $x=-0.1011$ $y=-0.1101$ 求 $[\frac{x}{y}]_{原}$ 6.3

解：

0.1011	0.0000	
+1.0011		$+[-y^*]_{补}$（减除数，判断溢出）
1.1110	0	余数为负，上商 0
1.1100	0	← 1
+0.1101		$+[y^*]_{补}$
0.1001	01	余数为正，上商 1
1.0010	01	← 1
+1.0011		$+[-y^*]_{补}$
0.0101	011	余数为正，上商 1
0.1010	011	← 1
+1.0011		$+[-y^*]_{补}$
1.1101	0110	余数为负，上商 0
1.1010	0110	← 1
+0.1101		$+[y^*]_{补}$
0.0111	01101	余数为正，上商 1

逻辑左移

逻辑左移

逻辑左移

逻辑左移

真正余数为 -0.00000111

$[x]_{原}=1.1011$

$[y]_{原}=1.1101$

$[x^*]_{补}=0.1011$

$[y^*]_{补}=0.1101$

$[-y^*]_{补}=1.0011$

正、1、移、减

负、0、移、加

例6.25 结果

① $x_0 \oplus y_0 = 1 \oplus 1 = 0$

② $\frac{x^*}{y^*} = 0.1101$

$\therefore \ [\frac{x}{y}]_{原} = 0.1101$

特点 上商 $n+1$ 次

第一次上商判溢出

移 n 次，加（或负数补码的加法） $n+1$ 次

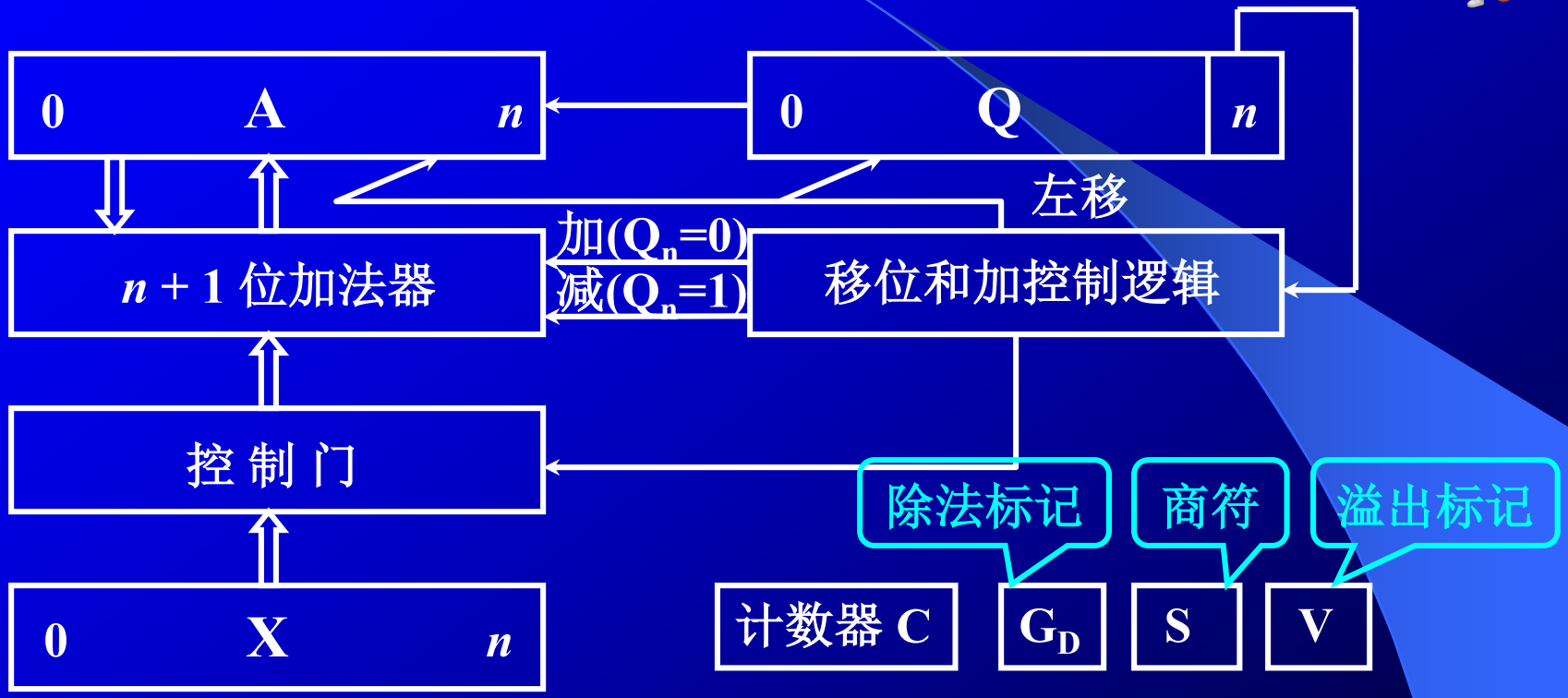
用移位的次数判断除法是否结束

少于恢复余数法

(3) 原码加减交替除法硬件配置

6.3

有问题吗

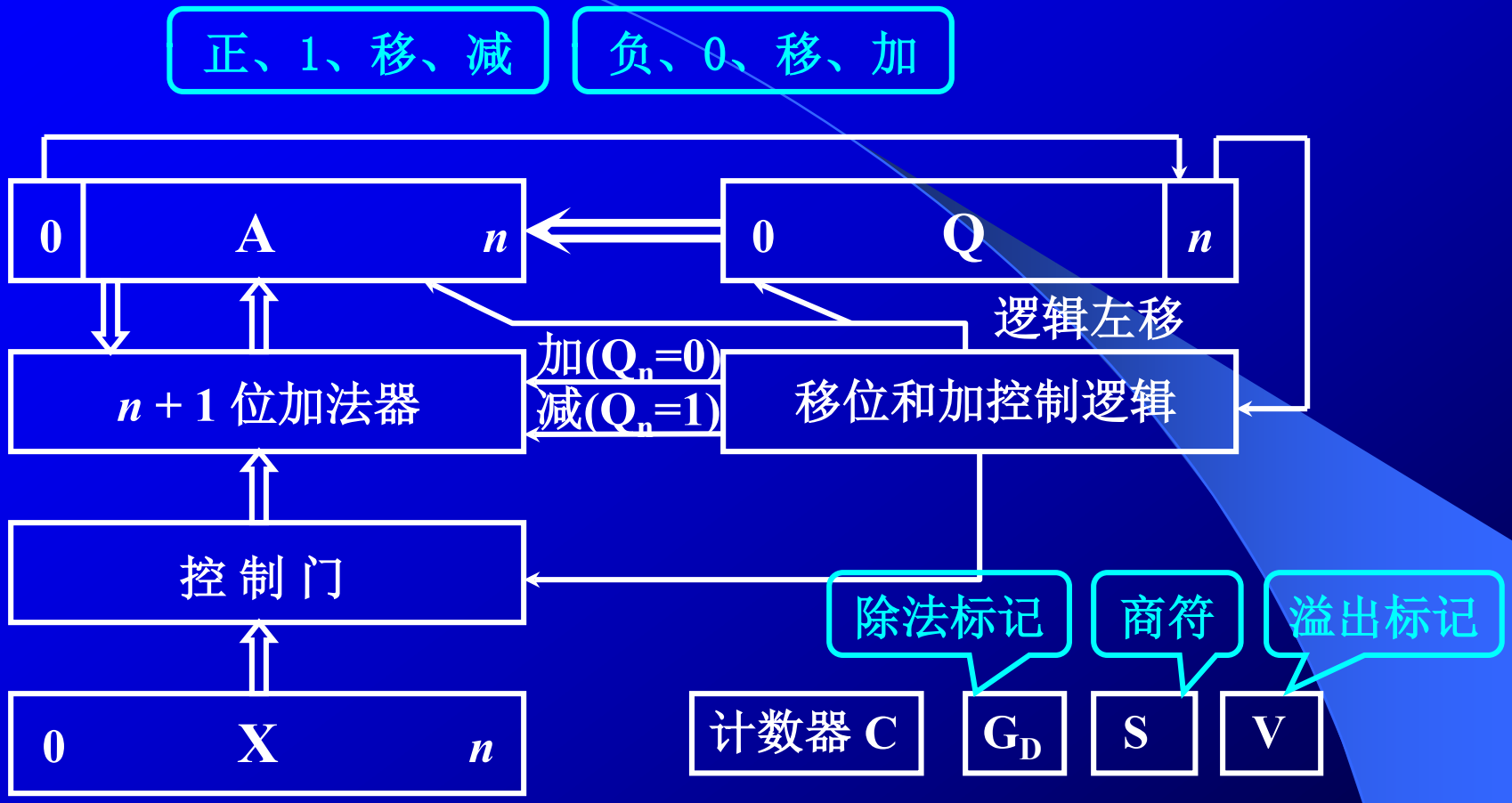


A(被除数原码)、X(除数原码)、Q(商) 均 n+1 位

用 Q_n 控制加或者减除数

(3) 原码加减交替除法及其硬件配置

6.3



A(被除数原码)、X(除数原码)、Q(商) 均 $n+1$ 位寄存器

用 Q_n 控制加或者减除数

4. 补码除法(加减交替法)

6.3

(1) 商值的确定 (先判断够不够减，再议上0还是上1)

① 比较被除数和除数绝对值的大小

x 与 y 同号

$x = 0.1011$　　$[x]_{补} = 0.1011$　　$[x]_{补} = 0.1011$　　$x^* > y^*$

$y = 0.0011$　　$[y]_{补} = 0.0011$　　$+[-y]_{补} = 1.1101$　　$[R_i]_{补}$与$[y]_{补}$同号

$[R_i]_{补} = 0.1000$　　“够减”

是指绝对值够减，而不是x与y的真值够减

$x = -0.0011$　　$[x]_{补} = 1.1101$　　$[x]_{补} = 1.1101$　　$x^* < y^*$

$y = -0.1011$　　$[y]_{补} = 1.0101$　　$+[-y]_{补} = 0.1011$　　$[R_i]_{补}$与$[y]_{补}$异号

$[R_i]_{补} = 0.1000$　　“不够减”

6.3

x 与 y 异号

$x = 0.1011$　　$[x]_{补} = 0.1011$

$y = -0.0011$　　$[y]_{补} = 1.1101$

$$\begin{array}{rl} & [x]_{补} = 0.1011 \\ + & [y]_{补} = 1.1101 \\ \hline & [R_i]_{补} = 0.1000 \end{array}$$

$x^* > y^*$

$[R_i]_{补}$ 与 $[y]_{补}$ 异号

“够减”

$x = -0.0011$　　$[x]_{补} = 1.1101$

$y = 0.1011$　　$[y]_{补} = 0.1011$

$$\begin{array}{rl} & [x]_{补} = 1.1101 \\ + & [y]_{补} = 0.1011 \\ \hline & [R_i]_{补} = 0.1000 \end{array}$$

$x^* < y^*$

$[R_i]_{补}$ 与 $[y]_{补}$ 同号

“不够减”

小结

$[x]_{补}$ 和 $[y]_{补}$	求 $[R_i]_{补}$	$[R_i]_{补}$ 与 $[y]_{补}$
同号	$[x]_{补} - [y]_{补}$	同号，“够减”
异号	$[x]_{补} + [y]_{补}$	异号，“够减”

② 商值的确定　末位恒置“1”法

6.3

×.××××1　可能存在误差2^{-n}

$[x]_{补}$与 $[y]_{补}$同号　0.××××1　按原码上商　“够减”上“1”　“不够减”上“0”

正商　原码

$[x]_{补}$与 $[y]_{补}$异号　1.××××1　按反码上商　“够减”上“0”　“不够减”上“1”

负商　反码

小 结

$[x]_{补}$与 $[y]_{补}$	商	$[R_i]_{补}$与 $[y]_{补}$	商　值
同　号	正	够减　（同号） 不够减（异号）	1 0　原码上商
异　号	负	够减　（异号） 不够减（同号）	0 1　反码上商

回顾前一页

$[x]_{补}$和$[y]_{补}$	求 $[R_i]_{补}$	$[R_i]_{补}$与 $[y]_{补}$
同号	$[x]_{补}-[y]_{补}$	同号，“够减”
异号	$[x]_{补}+[y]_{补}$	异号，“够减”

合并简化为

$[R_i]_{补}$与$[y]_{补}$	商值
同　号	1
异　号	0

(2) 商符的形成

除法过程中自然形成

$[x]_{补}$和$[y]_{补}$同号 ⟶ $[x]_{补}-[y]_{补}$ ⟶ 比较$[R_i]_{补}$和$[y]_{补}$

同号(够)“1”

异号(不够)“0”

小数除法 第一次“不够”上“0”

原码上商

小数除法时表示溢出

即不能溢出

正商

$[x]_{补}$和$[y]_{补}$异号 ⟶ $[x]_{补}+[y]_{补}$ ⟶ 比较$[R_i]_{补}$和$[y]_{补}$

异号(够)“0”

同号(不够)“1”

小数除法 第一次“不够”上“1”

反码上商

负商

(3) 新余数的获得

加减交替

$[R_i]_{补}$和 $[y]_{补}$	商	新余数
同 号	1	$2[R_i]_{补} + [-y]_{补}$
异 号	0	$2[R_i]_{补} + [y]_{补}$

同、1、移、减

异、0、移、加

例：

设 $x = -0.1011$　$y = 0.1101$

求 $[\frac{x}{y}]_{补}$ 并还原成真值

6.3

解：$[x]_{补} = 1.0101$　$[y]_{补} = 0.1101$　$[-y]_{补} = 1.0011$

1.0101	0.0000	
+0.1101		$[x]_{补}$和 $[y]_{补}$异号做加法
0.0010	1	$[R_i]_{补}$和 $[y]_{补}$同号上“1”
0.0100	1	←1
+1.0011		$+[-y]_{补}$
1.0111	10	异号上“0”
0.1110	10	←1
+0.1101		$+[y]_{补}$
1.1011	100	异号上“0”
1.0110	100	←1
+0.1101		$+[y]_{补}$
0.0011	1001	同号上“1”
0110	10011	←1 末位恒置“1”

逻辑左移

逻辑左移

逻辑左移

逻辑左移

$[R_i]_{补}$和 $[y]_{补}$	商	新余数
同 号	1	$2[R_i]_{补} + [-y]_{补}$
异 号	0	$2[R_i]_{补} + [y]_{补}$

$\therefore [\frac{x}{y}]_{补} = 1.0011$

则 $\frac{x}{y} = -0.1101$

$[R_i]_{补}$和 $[y]_{补}$	商	新余数
同 号	1	$2[R_i]_{补}+[-y]_{补}$
异 号	0	$2[R_i]_{补}+[y]_{补}$

例 6.26 已知 $x=0.1001$，$y=0.1101$，求 $\left[\frac{x}{y}\right]_{补}$。

解：由 $x=0.1001$，$y=0.1101$

得 $[x]_{补}=0.1001$，$[y]_{补}=0.1101$，$[-y]_{补}=1.0011$

其运算过程如表 6.26 所示。

表 6.26 例 6.26 的运算过程

被除数(余数)	商	说 明
0.1001	0.0000	
+1.0011		$[x]_{补}$与$[y]_{补}$同号，+$[-y]_{补}$
1.1100	0	$[R]_{补}$与$[y]_{补}$异号，上商“0”
1.1000	0	←1 位
+0.1101		+$[y]_{补}$
0.0101	01	$[R]_{补}$与$[y]_{补}$同号，上商“1”
0.1010	01	←1 位
+1.0011		+$[-y]_{补}$
1.1101	010	$[R]_{补}$与$[y]_{补}$异号，上商“0”
1.1010	010	←1 位
+0.1101		+$[y]_{补}$
0.0111	0101	$[R]_{补}$与$[y]_{补}$同号，上商“1”
0.1110	01011	←1 位，末位商恒置“1”

所以 $\left[\frac{x}{y}\right]_{补}=0.1011$

(4) 小结

- 补码除法共上商 $n+1$ 次（末位恒置 1）
  第一次为商符
- 第一次商可判溢出
- 加 n 次　移 n 次
- 用移位的次数判断除法是否结束
- 精度误差最大为 2^{-n}

(5) 补码除和原码除（加减交替法）比较 6.3

	原码除	补码除
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	余数的正负	比较余数和除数的符号
上商次数	$n+1$	$n+1$
加法次数	$n+1$	n 因为末位商恒置为1
移位	逻辑左移	逻辑左移
移位次数	n	n
第一步操作	$[x^*]_{补} - [y^*]_{补}$ 或$[x^*]_{补} + [-y^*]_{补}$	同号$[x]_{补} - [y]_{补}$ 或$[x]_{补} + [-y]_{补}$ 异号$[x]_{补} + [y]_{补}$

作业

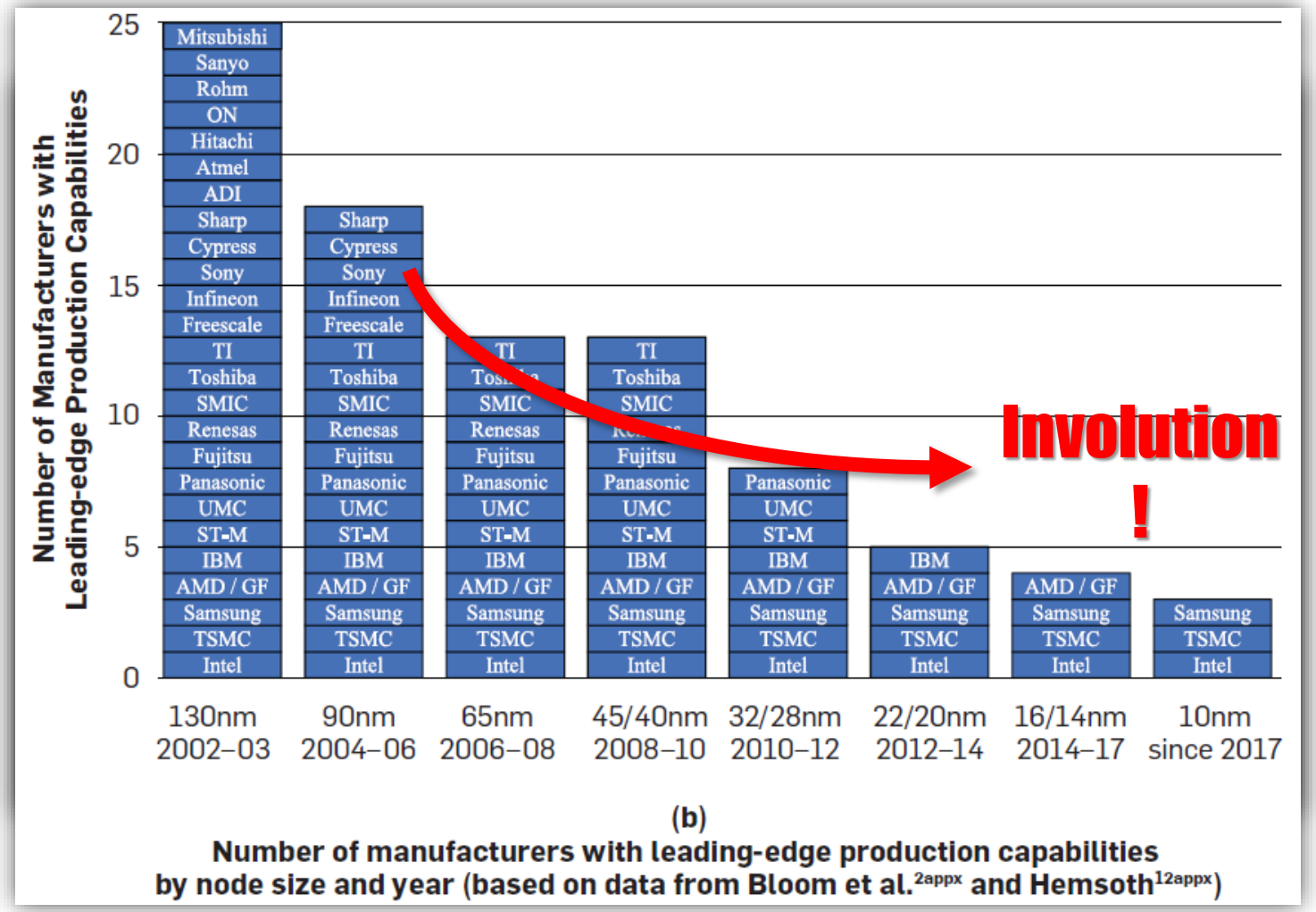
- 【hw5】
- 习题：
  - 6.20（原码一位乘、原码两位乘、补码一位乘Booth算法必做，补码两位乘选做）
  - 6.23
  - 6.21（原码加减交替除法，补码加减交替除法）
- 本次作业提交截止时间：
  - 4月14日上课前提交

芯片晶圆厂

Moore's Second Law

Moore's Law:
2X Transistors / chip
every 2 years

The cost of a chip fab doubles every four years



(b)
Number of manufacturers with leading-edge production capabilities by node size and year (based on data from Bloom et al.[2appx] and Hemsoth[12appx])

Neil C. Thompson and Svenja Spanuth, "The Decline of Computers as a General Purpose Technology", Communications of the ACM, March 2021

6.4 浮点四则运算

一、浮点加减运算

$$x = S_x \cdot 2^{j_x} \qquad y = S_y \cdot 2^{j_y}$$

S：一般为绝对值小于1的规格化数

1. 对阶(使得两数的小数点对齐)

(1) 求阶差

$$\Delta j = j_x - j_y = \begin{cases} = 0 & j_x = j_y \quad \text{已对齐} \\ > 0 & j_x > j_y \begin{cases} x \text{ 向 } y \text{ 看齐} & S_x \leftarrow 1,\ j_x - 1 \\ y \text{ 向 } x \text{ 看齐} & \checkmark\ S_y \rightarrow 1,\ j_y + 1 \end{cases} \\ < 0 & j_x < j_y \begin{cases} x \text{ 向 } y \text{ 看齐} & \checkmark\ S_x \rightarrow 1,\ j_x + 1 \\ y \text{ 向 } x \text{ 看齐} & S_y \leftarrow 1,\ j_y - 1 \end{cases} \end{cases}$$

(2) 对阶原则　小阶向大阶看齐

(即阶小的尾数向右移，阶码++，直至两个阶码相等)

6.4

例如 $x = 0.1101 \times 2^{01}$　　$y = (-0.1010) \times 2^{11}$

求 $x+y$

解：$[x]_{补} = 00, 01; 00.1101$　　$[y]_{补} = 00, 11; 11.0110$

1. 对阶

① 求阶差　$[\Delta j]_{补} = [j_x]_{补} - [j_y]_{补}$

$$\begin{array}{rr} & = 00, 01 \\ + & 11, 01 \\ \hline & 11, 10 \end{array}$$

阶差为负（-2）　$\therefore S_x \longrightarrow 2$　$j_x + 2$

② 对阶　$[x]_{补'} = 00, 11; 00.0011$

2. 尾数求和

$$\begin{array}{rll} & [S_x]_{补'} & = 00.0011 \\ + & [S_y]_{补} & = 11.0110 \\ \hline & & 11.1001 \end{array}$$

对阶后的$[S_x]_{补'}$

$\therefore [x+y]_{补} = 00, 11; 11.1001$

3. 规格化

(1) 规格化数的定义

$$r = 2 \qquad \frac{1}{2} \le |S| < 1$$

(2) 规格化数的判断

$S>0$	规格化形式	$S<0$	规格化形式
真值	0.1××…×	真值	–0.1××…×
原码	0.1××…×	原码	1.1××…×
补码	0.1××…×	补码	1.0××…×
反码	0.1××…×	反码	1.0××…×

原码　不论正数、负数，第一数位为1，即为规格化形式

补码　符号位和第一数位不同，即为规格化形式

补码　符号位和第一数位不同，即为规格化形式

特例

$$S = -\frac{1}{2} = -0.100\cdots0$$

$$[S]_{原} = 1.100\cdots0$$

$$[S]_{补} = \boxed{1.1}00\cdots0$$

$\therefore$ $[-\frac{1}{2}]_{补}$ 不是规格化的数（规定）

$$S = -1$$

$$[S]_{补} = \boxed{1.0}00\cdots0$$

$\therefore$ $[-1]_{补}$ 是规格化的数（规定）

(3) 左规

即尾数出现 00. 0 ××… ×或 11.1 ××… ×时

尾数左移一位，阶码减 1，直到数符和第一数位不同为止

上例　$[x+y]_{补} = 00, 11;\ 11.\ 1001$

左规后　$[x+y]_{补} = 00, 10;\ 11.\ 0010$

负数的补码左移补0还是1？

$\therefore\ x + y = (-\ 0.1110) \times 2^{10}$

(4) 右规

尾数溢出不是浮点运算溢出

当 尾数溢出（ >1）时，需 右规

即尾数出现 01. ×× …×或 10. ×× …×时

尾数右移一位，阶码加 1

例6.27 $x = 0.1101 \times 2^{10}$ $y = 0.1011 \times 2^{01}$

求 $x+y$（除阶符、数符外，阶码取 3 位，尾数取 6 位）

解：$[x]_{补} = 00, 010; 00.\ 110100$

$[y]_{补} = 00, 001; 00.\ 101100$

① 对阶

$[\Delta j]_{补} = [j_x]_{补} - [j_y]_{补}$ = 00, 010

\+ 11, 111

100, 001

阶差为 +1 $\therefore S_y \longrightarrow 1,\ j_y+1$

$\therefore [y]_{补'} = 00, 010; 00.\ 010110$

② 尾数求和

$[S_x]_{补}$ = 00. 110100

\+ $[S_y]_{补'}$ = 00. 010110 对阶后的$[S_y]_{补'}$

01. 001010 尾数溢出需右规

③ 右规

$[x+y]_{补} = 00,\ 010;\ 01.\ 001010$

右规后

$[x+y]_{补} = 00,\ 011;\ 00.\ 100101$

$\therefore\ x+y = 0.\ 100101 \times 2^{11}$

移动时认为此数为正数（最高符号位为0），所以正数补码移动补0

4. 舍入

在 对阶 和 右规 过程中，可能出现 尾数末位丢失 引起误差，需考虑舍入

(1) 0 舍 1 入法

(2) 恒置 “1” 法

例 6.28 $x=(-\frac{5}{8})\times 2^{-5}$ $y=(\frac{7}{8})\times 2^{-4}$

求 $x-y$（除阶符、数符外，阶码取 3 位，尾数取 6 位）

解：$x=(-0.101000)\times 2^{-101}$ $y=(0.111000)\times 2^{-100}$

$[x]_{补}=11,011;\ 11.011000$ $[y]_{补}=11,100;\ 00.111000$

① 对阶

$$[\Delta j]_{补}=[j_x]_{补}-[j_y]_{补}$$

$$\begin{array}{r} 11,011 \\ +\ 00,100 \\ \hline 11,111 \end{array}$$

阶差为 –1 ∴ $S_x \longrightarrow 1$, j_x+1

∴ $[x]_{补'}=11,100;\ 11.101100$

② 尾数求和

$$[S_x]_{补'} = 11.\ 101100$$
$$+\ [-S_y]_{补} = 11.\ 001000$$
$$\overline{\qquad\qquad 110.\ 110100}$$

③ 右规

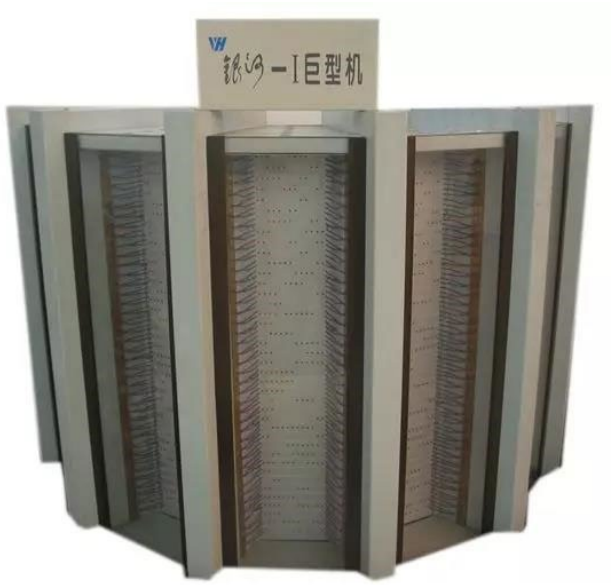
$$[x-y]_{补} = 11,\ 100;\ 10.\ 110100$$

右规后

$$[x-y]_{补} = 11,\ 101;\ 11.\ 011010$$

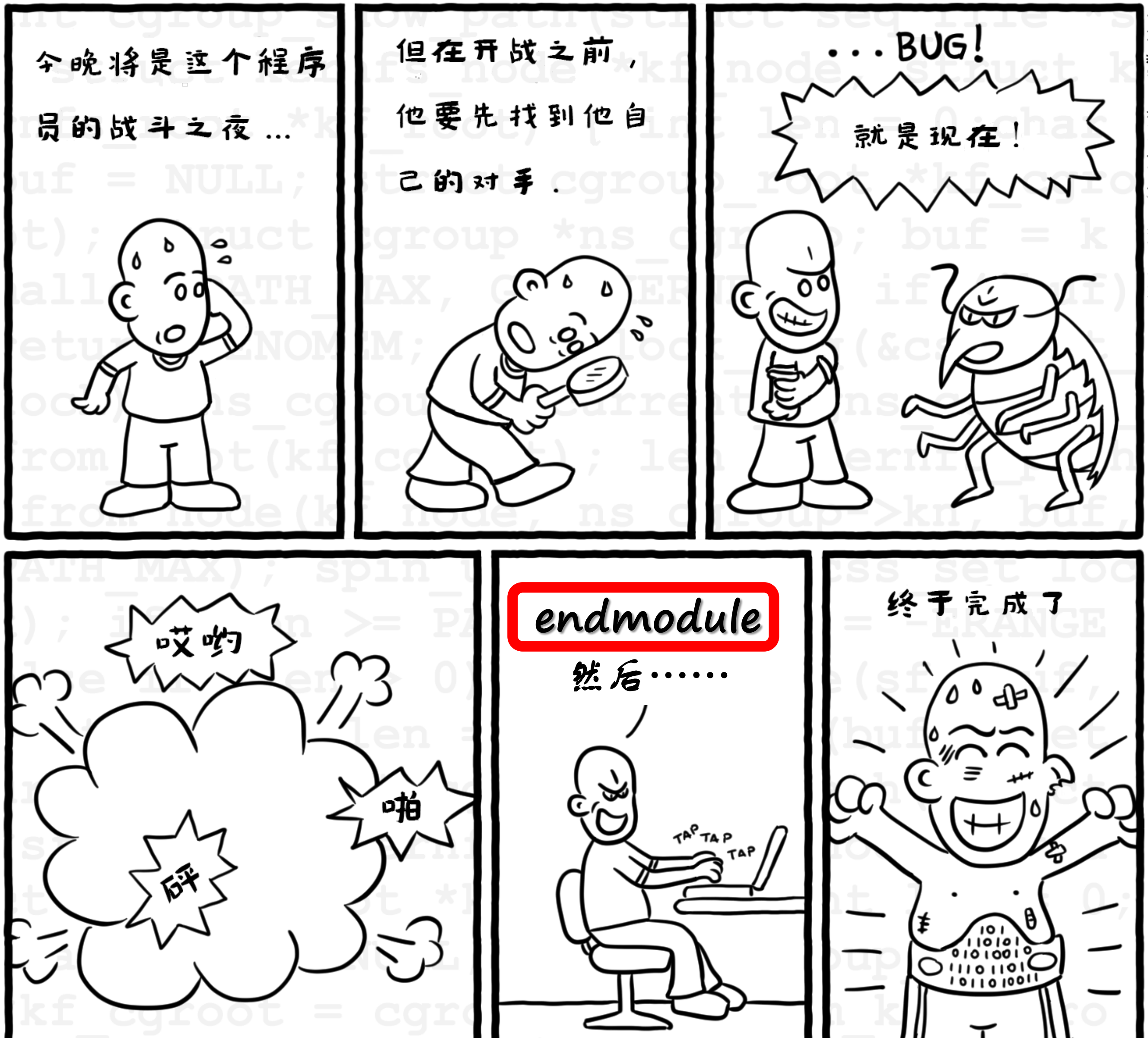
$$\therefore\ x-y = (-0.100110)\times 2^{-11}$$

$$= (-\frac{19}{32})\times 2^{-3}$$



周兴铭院士科研成就——“一生就是做了那么几台机器”

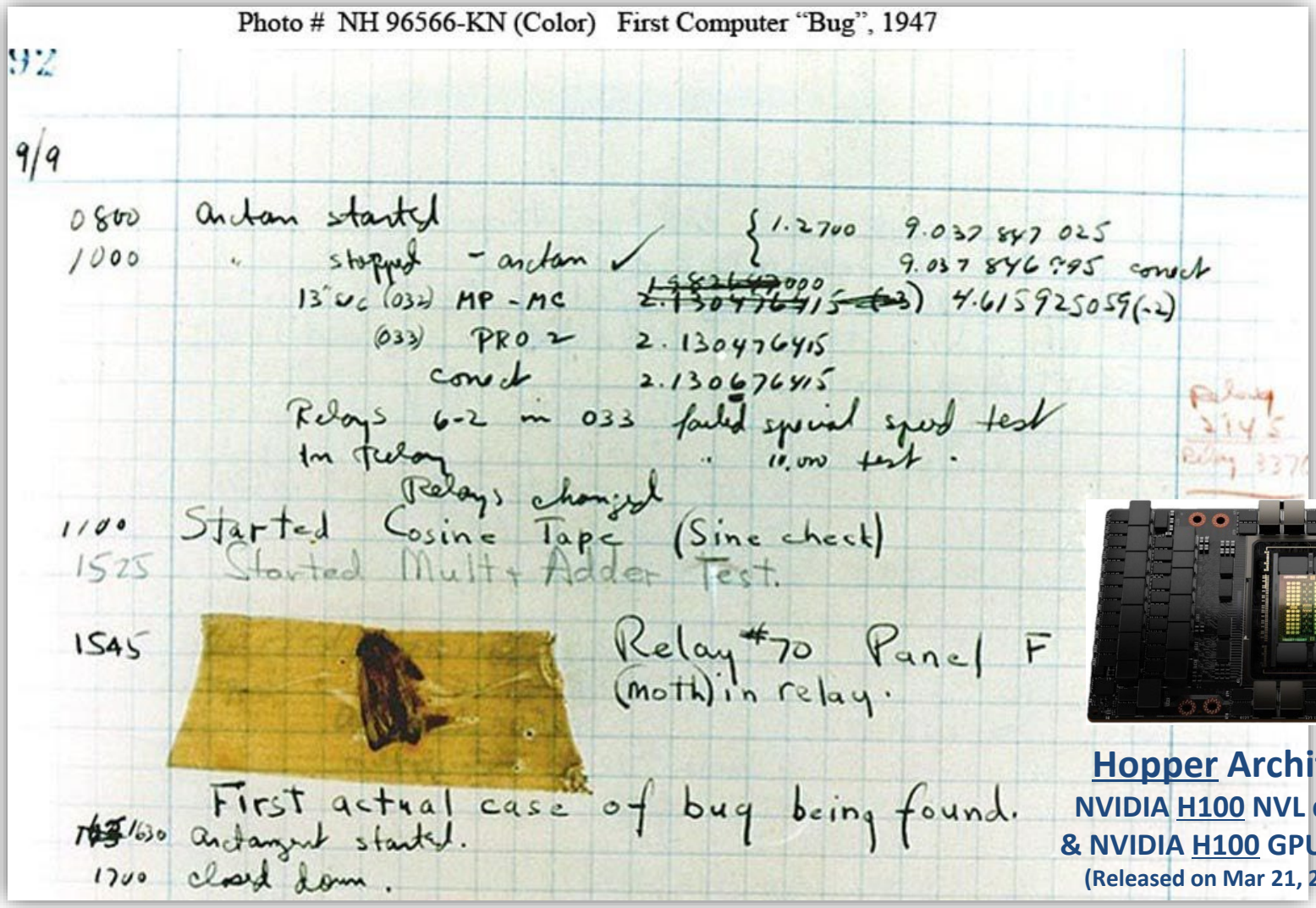
- 先后参加晶体管计算机、集成电路计算机、百万次级大型计算机研制，在锗晶体管电路抗高温稳定性、TTL信号传输抗干扰以及快速除法算法等方面做出创新性工作
- 中国第一台巨型计算机银河I主机系统负责人，中国第一台全数字实时仿真计算机银河仿I总负责人，中国第一台面向科学/工程计算的并行巨型计算机银河II总设计师，在总体方案、CPU结构、可靠性及RAS技术方案、系统接口协议等方面做出创新性工作，攻克若干技术难关
- 研究领域还包括：高性能计算、移动计算和微处理器体系结构



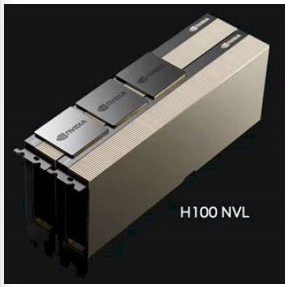
字体：黎云汉禾真佳佳书法体，Comic Sans MS

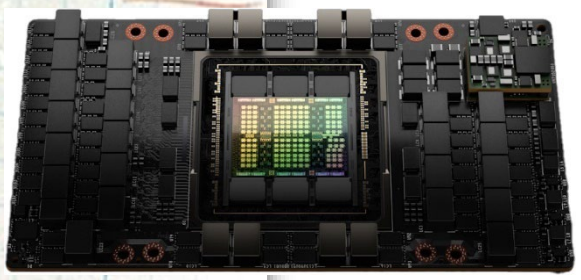
原创：Daniel Stori {turnoff.us} 文字翻译：LCTT@GHLandy 图文合成：LCTT@GHLandy {GHLandy.com}

World's First Computer "Bug" (Sep 9, 1947)



Grace Hopper





Hopper Architecture-based
NVIDIA H100 NVL dual-GPU on PCIe
& NVIDIA H100 GPU on SXM5 module
(Released on Mar 21, 2023 & Mar 22, 2022)

https://www.nationalgeographic.org/thisday/sep9/worlds-first-computer-bug/
https://developer.nvidia.com/blog/nvidia-hopper-architecture-in-depth/

复习：

1. 除法运算：“加(减)”和“左移”实现
2. 原码除法：符号位和数值位分开运算。数值部分用无符号数除法实现。可用于浮点数尾数除法运算。
   ① 恢复余数法：核心操作是“试商”
   如果试商失败(余数为负，不够减)，则上商0，并恢复余数，然后再执行下一轮操作
   如果试商成功，则上商1，直接下一轮操作
   ② 不恢复余数法（加减交替法）：核心是将“本轮恢复余数”操作合并到“下一轮试商”操作中。　　正、1、移、减；　负、0、移、加
   余数 $R_i>0$ 上商“1” $2R_i-y^*$ → $+[-y^*]_{补}$
   余数 $R_i<0$ 上商“0” $2R_i+y^*$ → $+[y^*]_{补}$
3. 补码除法：符号位和数值位一起运算。可用于整数除法运算。
   ① 不恢复余数法（加减交替法）：同、1、移、减；　异、0、移、加

$[R_i]_{补}$和 $[y]_{补}$	商	新余数
同 号	1	$2[R_i]_{补}+[-y]_{补}$
异 号	0	$2[R_i]_{补}+[y]_{补}$

4. 注意区别$[-y^*]_{补}$和$[-y]_{补}$
   ① 原码除法中使用$[-y^*]_{补}$
   ② 补码除法中使用$[-y]_{补}$
5. 在除法运算之前先做检查：被除数不等于 0、除数不能为 0

复习：

6. 关于余数的三点说明
   - ① 在原码除法中，若最后一次上商为0，不论采用恢复余数还是不恢复余数法，都需要进行一次显式的恢复余数操作
   - ② 教材中原码除法竖式最后一行的“余数”如何变为“真实余数”？仅右移*n*位？例6.24和6.25的结果是否正确？
   - ③ 补码除法中的商末位恒置1法，不仅可致商的精度损失，也会得到错误的余数。请自行核对教材P266例6.26和例6.27的结果
7. 浮点数运算：由多个ALU + 移位器实现
8. 浮点加减法
   - ① 阶码对阶操作：求阶差，小阶向大阶看齐
   - ② 尾数求和操作：对阶后两个尾数按定点加减法规则计算
   - ③ 尾数规格化操作：左规提高有效位数、右规避免尾数溢出
   - ④ 尾数舍入操作：对阶和右规中可能产生舍入

   还需要进行其他操作吗？