

Bài 03: Số chấm động

Nguyễn Đỗ Thái Nguyên
nguyenndt@hcmup.edu.vn

Vấn đề với biểu diễn số nguyên

- Số nguyên N bit biểu diễn được 2^N giá trị

- Biểu diễn không dấu (Unsigned Integer)

$$0 \quad \text{à} \quad 2^N - 1$$

$$(N=32, 2^N - 1 = 4,294,967,295)$$

- Biểu diễn bù 2

$$-2^{(N-1)} \quad \text{à} \quad 2^{(N-1)} - 1$$

$$(N=32, 2^{(N-1)} = 2,147,483,648)$$

- Biểu diễn số rất lớn ? Số giây / 1 nghìn năm

- 31,556,926,000 (3.1556926×10^{10})

- Biểu diễn số rất nhỏ ? Số giây / 1 nano giây

- 0.0000000001₁₀ ($1.0_{10} \times 10^{-9}$)

- Biểu diễn số thập phân 1.5 ?

Biểu diễn phần thập phân

- Biểu diễn số 5.375 thế nào ?
Cần bao nhiêu bit ?
- Giả sử dùng 8 bit để lưu trữ phần nguyên

$$5 = 4 + 1 = 00000101$$

- Tương tự có thể dùng 8 bit lưu trữ phần thập phân

$$0.375 = 0.25 + 0.125 = 01100000$$

- Vậy có thể biểu diễn

$$5.375 = 00000101.01100000$$

- Tổng quát ta có:

$$x_{n-1} \mathbf{K} x_1 x_0 . x_{-1} x_{-2} \mathbf{K} x_{-m} = \sum_{i=-m}^n x_i 2^i$$

=> Biểu diễn số chấm tĩnh (fixed point)

i	2 ⁻ⁱ
0	1.0 1
1	0.5 1/2
2	0.25 1/4
3	0.125 1/8
4	0.0625 1/16
5	0.03125 1/32
6	0.015625 ...
7	0.0078125
8	0.00390625
9	0.001953125
10	0.0009765625
11	0.00048828125
12	0.000244140625
13	0.0001220703125
14	0.00006103515625
15	0.000030517578125

Giới hạn biểu diễn số chấm tĩnh

- Với 8 bit

- Phần nguyên lớn nhất có thể biểu diễn là

$$2^8 - 1 = 255$$

- Phần thập phân nhỏ nhất có thể biểu diễn là

$$2^{-8} = 1/256 = 0.00390625 \sim 10^{-3}$$

- Nếu muốn tính toán với số nhỏ hơn như 0.0001_{10} hay 0.00001_{10} ?

- à Tăng số bit

- Với 16 bit phần thập phân

$$\text{min} = 1/65536 = 0.0000152587890625 \sim 10^{-5}$$

- Có cách nào tốt hơn ?

Số chấm động – Ý tưởng

- Hệ thập phân

- $123000000000 \sim 1.23 \times 10^{11}$ và $0.00000000000123 \sim 1.23 \times 10^{-11}$

- Tương tự với hệ nhị phân, ta có

- $x = 00000101.01100000 = 2^2 + 2^0 + 2^{-2} + 2^{-3}$

- Ta có thể viết lại

- $x = 1.01011 \times 2^2$

- Thay vì dùng 16 bit để lưu trữ, chỉ cần dùng 7 bit (5 bit phần trị + 2 bit phần mũ)

- $x = 1.\underline{01011} \ \underline{10}$

- Như vậy,

- Muốn tiết kiệm số bit lưu trữ, ta đã di chuyển vị trí của dấu chấm sang phải 14 vị trí

- Cần lưu: phần trị, phần mũ và ...phần dấu

=> Đây là ý tưởng cơ bản của số chấm động (floating point)

Biểu diễn số chấm động

- Biểu diễn số chấm động



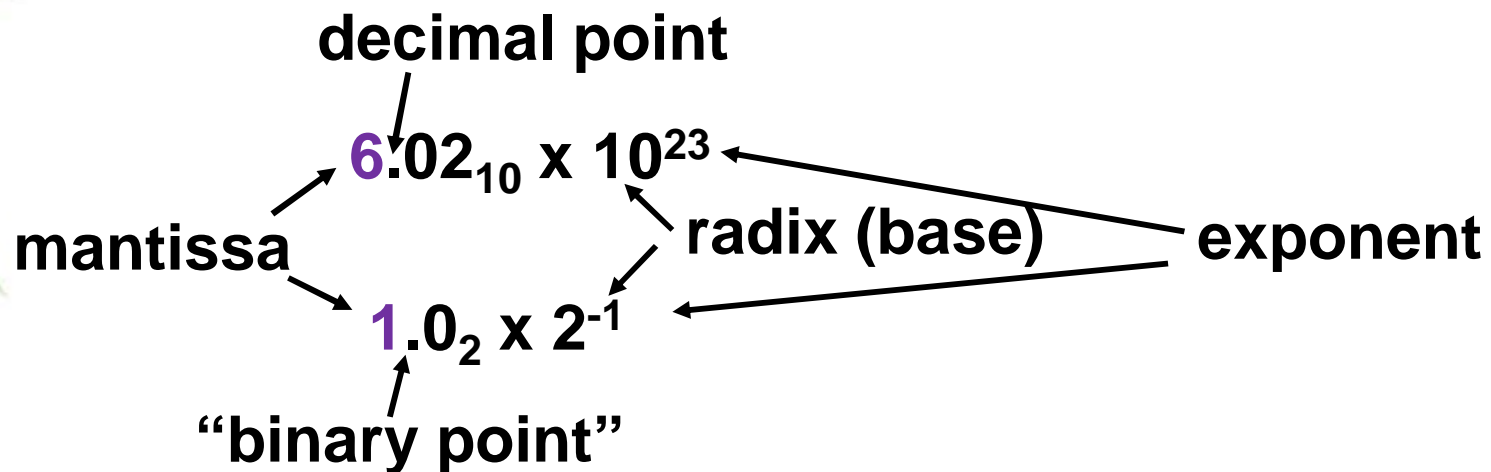
- Sign (S): phần dấu
- Exponent (E): phần số mũ
- Significand (S): phần định trị

- Giá trị

$$\pm S \times 2^E$$

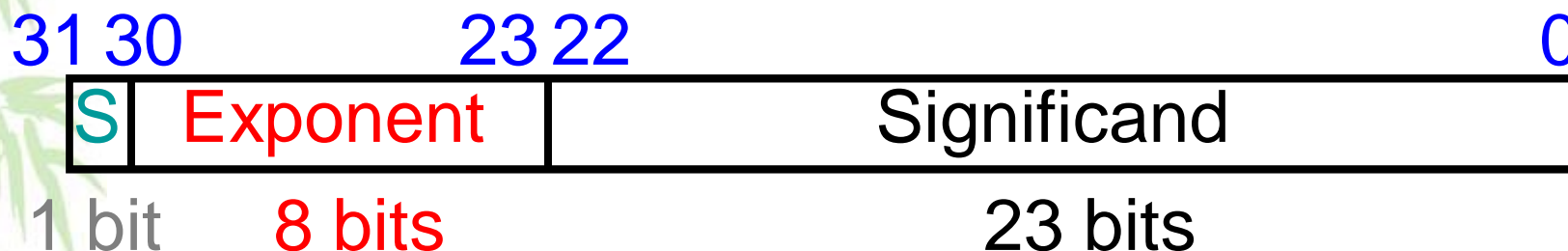
Biểu diễn khoa học

- Giá trị $1 / 1,000,000,000$ có thể biểu diễn như sau:
 - $1.0_{10} \times 10^{-9}$ à Dạng chuẩn (Normalized form)
 - $0.1_{10} \times 10^{-8}$, $10.0_{10} \times 10^{-10}$ à Dạng không chuẩn (Denormalized form)
- Dạng chuẩn: phần nguyên gồm 1 chữ số khác 0



Chuẩn số chấm động IEEE 754

- Biểu diễn số chấm động Single Precision (32 bit)



- S: dấu (Sign) – 0: dương, 1: âm
- Exponent: phần số mũ (lưu dưới dạng số biased)
- Significand: phần định trị
 - Ngầm định bắt đầu là 1 + phần trị ~ (1 + 23) bits
- Dạng chuẩn: $+/-1.xxx...x_2 \times 2^{yyy...y_2}$
- Ví dụ:

Biểu diễn: 0 10000001 010110000000000000000000

Có giá trị: $+1.0101100...00 \times 2^{10000001} \sim +(1+2^{-2} + 2^{-4} + 2^{-5}) \times 2^2 = 5.375$

Chuyển từ biểu diễn nhị phân sang thập phân

0	0110 1000	101 0101 0100 0011 0100 0010
---	-----------	------------------------------

- Dấu: 0 à dương

- Mũ:

– 0110 1000 có giá trị (dạng biased) là
 $104 - 127 = -23$

- Trị:

$$\begin{aligned} & 1 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} + 0 \times 2^{-4} + 1 \times 2^{-5} + \dots \\ & = 1 + 2^{-1} + 2^{-3} + 2^{-5} + 2^{-7} + 2^{-9} + 2^{-14} + 2^{-15} + 2^{-17} + 2^{-22} \\ & = 1.0 + 0.666115 \end{aligned}$$

- Kết quả: $1.666115 \times 2^{-23} \sim 1.986 \times 10^{-7}$
($\sim 2/10,000,000$)

Chuyển từ biểu diễn thập phân sang nhị phân

$$-2.340625 \times 10^1$$

1. Không chuẩn hóa: -23.40625

2. Chuyển phần nguyên:

$$23 = 16 + 4 + 2 + 1 = 10111$$

3. Chuyển phần thập phân:

$$.40625 = .25 + .125 + .03125 = .01101$$

4. Kết hợp và chuẩn hóa:

$$10111.01101 = 1.011101101 \times 2^4$$

5. Chuyển phần mũ: $127 + 4 = 10000011$

1	1000 0011	011 1011 0100 0000 0000 0000
---	-----------	------------------------------

Chuyển từ biểu diễn thập phân sang nhị phân (tt)

- 1/3

$$= 0.33333...$$

$$= 0.25 + 0.0625 + 0.015625 + 0.00390625 + ...$$

$$= 1/4 + 1/16 + 1/64 + 1/256 + ...$$

$$= 2^{-2} + 2^{-4} + 2^{-6} + 2^{-8} + ...$$

$$= 0.0101010101... * 2^0$$

$$= 1.0101010101... * 2^{-2}$$

– Dấu: 0

– Mũ = $-2 + 127 = 125 = 01111101$


– Trị = $0101010101...$

0	0111 1101	0101 0101 0101 0101 0101 010
---	-----------	------------------------------

Các số đặc biệt

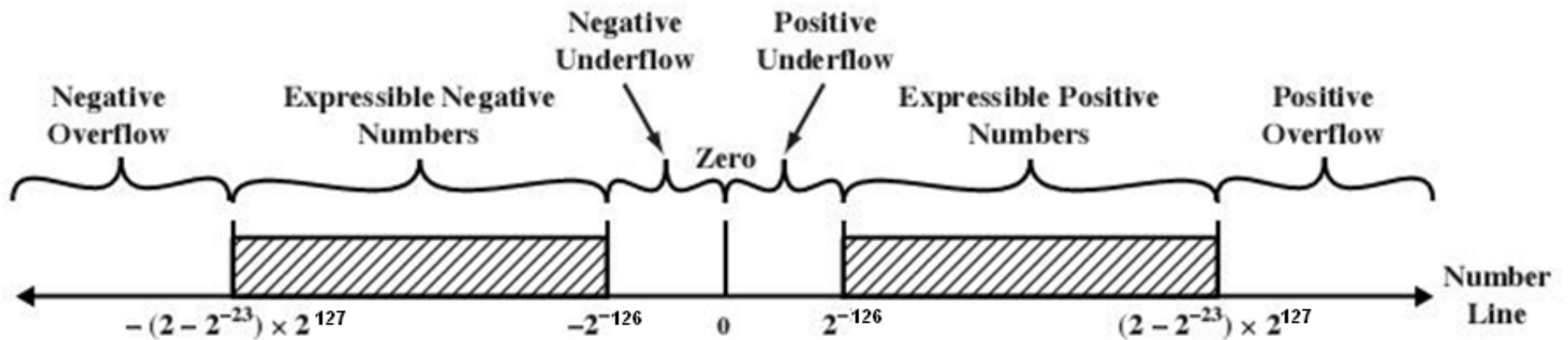
- Phần mũ = 0, phần trị = 0
 - Số zero
- Phần mũ = 0, phần trị $\neq 0$
 - Số dạng không chuẩn (denormalized)
- Phần mũ toàn bit 1, phần trị = 0
 - Số vô cùng (infinity)
- Phần mũ toàn bit 1, phần trị $\neq 0$
 - Số báo lỗi (NaN - Not a Number)
 - Signaling NaN
 - Quiet NaN

Những trường hợp tạo số đặc biệt

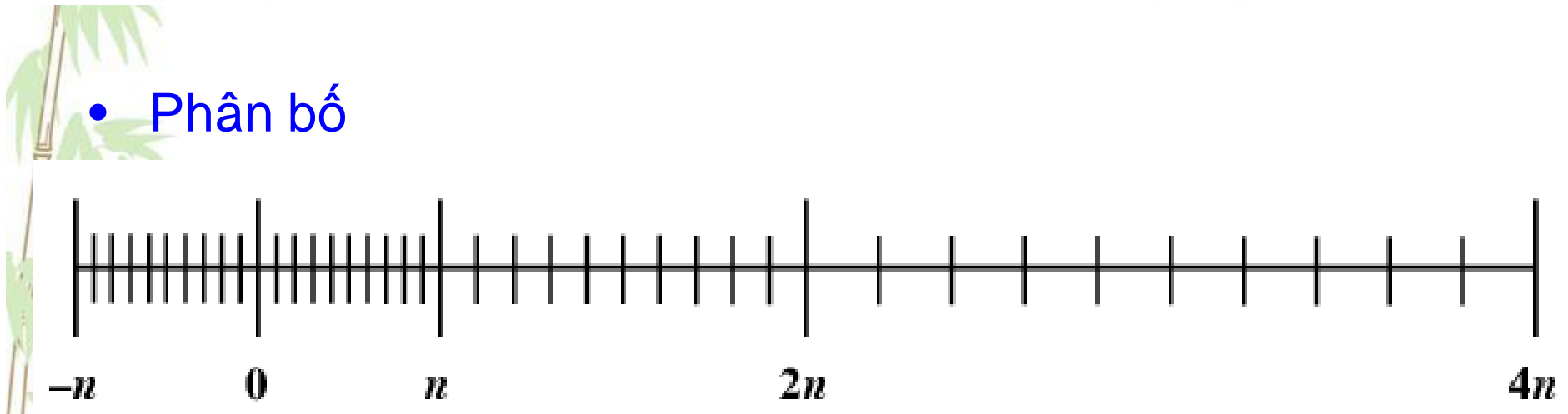
- 
- | | | |
|-----------------------------|-----------------------------|----------|
| 1. $X + (+\infty)$ | 11. $(+\infty) + (-\infty)$ | 21. |
| 2. $X - (+\infty)$ | 12. $(-\infty) + (+\infty)$ | |
| 3. $X + (-\infty)$ | 13. $(+\infty) - (+\infty)$ | |
| 4. $X - (-\infty)$ | 14. $(-\infty) - (-\infty)$ | |
| 5. $X \times (+\infty)$ | 15. $\infty \times 0$ | |
| 6. $X / (-\infty)$ | 16. $\infty / 0$ | |
| 7. $(+\infty) + (+\infty)$ | 17. $X / 0$ | |
| 8. $(-\infty) + (-\infty)$ | 18. $0 / 0$ | |
| 9. $(-\infty) - (+\infty)$ | 19. ∞ / ∞ | |
| 10. $(+\infty) - (-\infty)$ | 20. $\text{sqrt}(X), X < 0$ | |

Phân bố, phạm vi biểu diễn

- Phạm vi biểu diễn. Chứng minh ?



- Phân bố



Phân bố

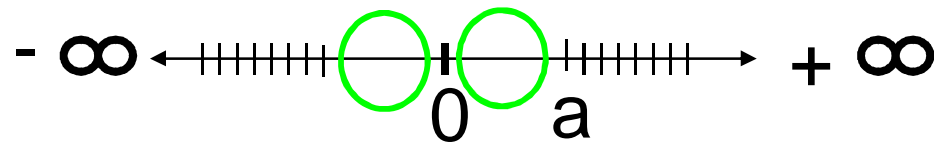
- Đặt $f(1, 2) =$ số lượng số chấm động trong khoảng 1 và 2
- Đặt $f(2, 3) =$ số lượng số chấm động trong khoảng 2 và 3
- Hỏi
 1. $f(1,2) < f(2,3)$
 2. $f(1,2) = f(2,3)$
 3. $f(1,2) > f(2,3)$

Số dạng không chuẩn

- Số dương nhỏ nhất có thể biểu diễn

$$a = 1.0..._2 \times 2^{-126} = 2^{-126}$$

Gaps!

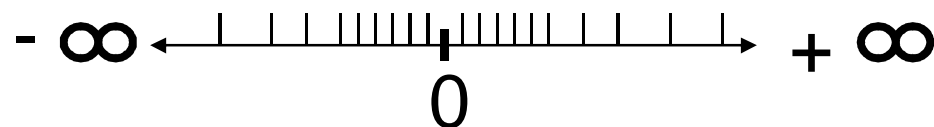


Lý do: ngầm định 1 + phần trị

- Giải pháp:

- Qui ước nếu số mũ = 0 (phần trị $\neq 0$), không ngầm định bắt đầu là 1 à **Số dạng không chuẩn (denormalized)**
- Số dương nhỏ nhất có thể biểu diễn

- $a = 0.00...1_2 \times 2^{-126} = 2^{-23} \times 2^{-126} = 2^{-149}$



Một số loại chấm động

- Single Precision (32 bit)
 - 1/8/23 (kiểu *float* trong C), 10^{-38} à 10^{38}
- Double Precision (64 bit)
 - 1/11/52 (kiểu *double* trong C), 10^{-308} à 10^{308}
- Half Precision (16 bit)
 - 1/5/10
- Quad Precision (8 bit)
 - 1/4/3
- IEEE 754-2008 “binary128” (128 bit)
 - 1/15/112

en.wikipedia.org/wiki/Floating_point

Biểu diễn số chấm động 8 bit

	s	exp	frac	E	Value
Denormalized numbers	0	0000	000	-6	0
	0	0000	001	-6	$1/8 * 1/64 = 1/512$ ← closest to zero
	0	0000	010	-6	$2/8 * 1/64 = 2/512$
	...				
	0	0000	110	-6	$6/8 * 1/64 = 6/512$
	0	0000	111	-6	$7/8 * 1/64 = 7/512$ ← largest denorm
	0	0001	000	-6	$8/8 * 1/64 = 8/512$ ← smallest norm
Normalized numbers	0	0001	001	-6	$9/8 * 1/64 = 9/512$
	...				
	0	0110	110	-1	$14/8 * 1/2 = 14/16$
	0	0110	111	-1	$15/8 * 1/2 = 15/16$ ← closest to 1 below
	0	0111	000	0	$8/8 * 1 = 1$
	0	0111	001	0	$9/8 * 1 = 9/8$ ← closest to 1 above
	0	0111	010	0	$10/8 * 1 = 10/8$
	...				
	0	1110	110	7	$14/8 * 128 = 224$
	0	1110	111	7	$15/8 * 128 = 240$ ← largest norm
	0	1111	000	n/a	inf

Bảng tóm tắt số chấm động

	Single Precision (32 bit)				Double Precision (64 bit)			
	Dấu	Mũ	Trị	Giá trị	Dấu	Mũ	Trị	Giá trị
+0	0	0	0	0	0	0	0	0
-0	1	0	0	-0	1	0	0	-0
$+\infty$	0	255 (toàn bit 1)	0	∞	0	2047 (toàn bit 1)	0	∞
$-\infty$	1	255 (toàn bit 1)	0	$-\infty$	1	2047 (toàn bit 1)	0	$-\infty$
Quiet NaN	0/ 1	255 (toàn bit 1)	$\neq 0$	NaN	0/ 1	2047 (toàn bit 1)	$\neq 0$	NaN
Signaling NaN	0/ 1	255 (toàn bit 1)	$\neq 0$	NaN	0/ 1	2047 (toàn bit 1)	$\neq 0$	NaN
Số dương (dạng chuẩn)	0	$0 < e < 255$	f	$2^{e-127} (1.f)$	0	$0 < e < 2047$	f	$2^{e-1023} (1.f)$
Số âm (dạng chuẩn)	1	$0 < e < 255$	f	$-2^{e-127} (1.f)$	1	$0 < e < 2047$	f	$-2^{e-1023} (1.f)$
Số dương (dạng không chuẩn)	0	0	$f \neq 0$	$2^{-126} (0.f)$	0	0	$f \neq 0$	$2^{-1022} (0.f)$
Số âm (dạng không chuẩn)	1	0	$f \neq 0$	$-2^{-126} (0.f)$	1	0	$f \neq 0$	$-2^{-1022} (0.f)$

Khái niệm Precision và Accuracy

- Precision: số bit được sử dụng trong máy tính để biểu diễn 1 giá trị.
- Accuracy: độ chính xác mà một kiểu biểu diễn trong máy tính có thể biểu diễn được một giá trị.
- Thường thì precision cao sẽ dẫn tới accuracy cao.
- Ví dụ: float $\pi = 3.14$;
 - π được biểu diễn bởi 24 bit phần trị (precise cao), nhưng chỉ có thể biểu diễn được gần đúng π (không accuracy).

Làm tròn (Rounding)

- Khi thực hiện các phép toán trên số chấm động, kết quả nhận được có thể vượt ra ngoài khả năng biểu diễn của phần định trị.
- Phần cứng phục vụ các phép toán trên số chấm động thường có thêm 2 bit nhớ hỗ trợ cho phần định trị giúp thực hiện việc làm tròn để có được kết quả chính xác nhất có thể.
- Ví dụ: thực hiện $(1.00...00 \times 2^1) - (1.11...11 \times 2^0)$

$\begin{array}{r} 1.00...00 \times 2^1 \\ - 0.111...1 \times 2^1 (= 1.11...11 \times 2^0) \\ \hline 0.000..01 \times 2^1 \\ = 1.00...00 \times 2^{-22} \end{array}$	$\begin{array}{r} 1.00...00 \underline{00} \times 2^1 \\ - 0.111...1 \underline{10} \times 2^1 (= 1.11...11 \times 2^0) \\ \hline 0.000..00 \underline{10} \times 2^1 \\ = 1.00...00 \underline{00} \times 2^{-23} \end{array}$
---	---

Chuẩn IEEE làm tròn số chấm động

- Làm tròn lên (Round up / Round towards $+\infty$)
 $1.01 \underline{10} \rightarrow 1.10$, $-1.01 \underline{10} \rightarrow -1.01$
- Làm tròn xuống (Round down / Round towards $-\infty$)
 $1.01 \underline{10} \rightarrow 1.01$, $-1.01 \underline{10} \rightarrow -1.10$
- Làm tròn về 0 (Truncate / Round towards 0)
 - Bỏ giá trị 2 bit nhớ
- Làm tròn về giá trị gần nhất (Round to nearest):
 - $1.01 \underline{01} \rightarrow 1.01$, $-1.01 \underline{11} \rightarrow -1.10$
 - Trường hợp 2 bit nhớ là 10 (halfway) ?
 - Làm tròn về số chẵn gần nhất (mặc định), nghĩa là LSB của phần định trị luôn bằng 0
 $1.01 \underline{10} \rightarrow 1.10$, $-1.10 \underline{11} \rightarrow -1.10$

Các trường hợp làm tròn khác

- Làm tròn cũng được thực hiện khi thực hiện chuyển đổi:
 - Chuyển đổi từ kiểu double precision thành single precision
 - Chuyển đổi từ số chấm động thành số nguyên và ngược lại
 - Ép kiểu từ số chấm động thành số nguyên và ngược lại
- Hãy khảo sát các trường hợp sau:

1.Chuyển đổi float -> int -> float .Kết quả như ban đầu ?

2.Chuyển đổi int -> float -> int. Kết quả như ban đầu ?

3.Phép cộng số chấm động có tính kết hợp ?

$$(x+y)+z = x+(y+z)$$

4.i = (int) (3.14159 * f);

5.f = f + (float) i;

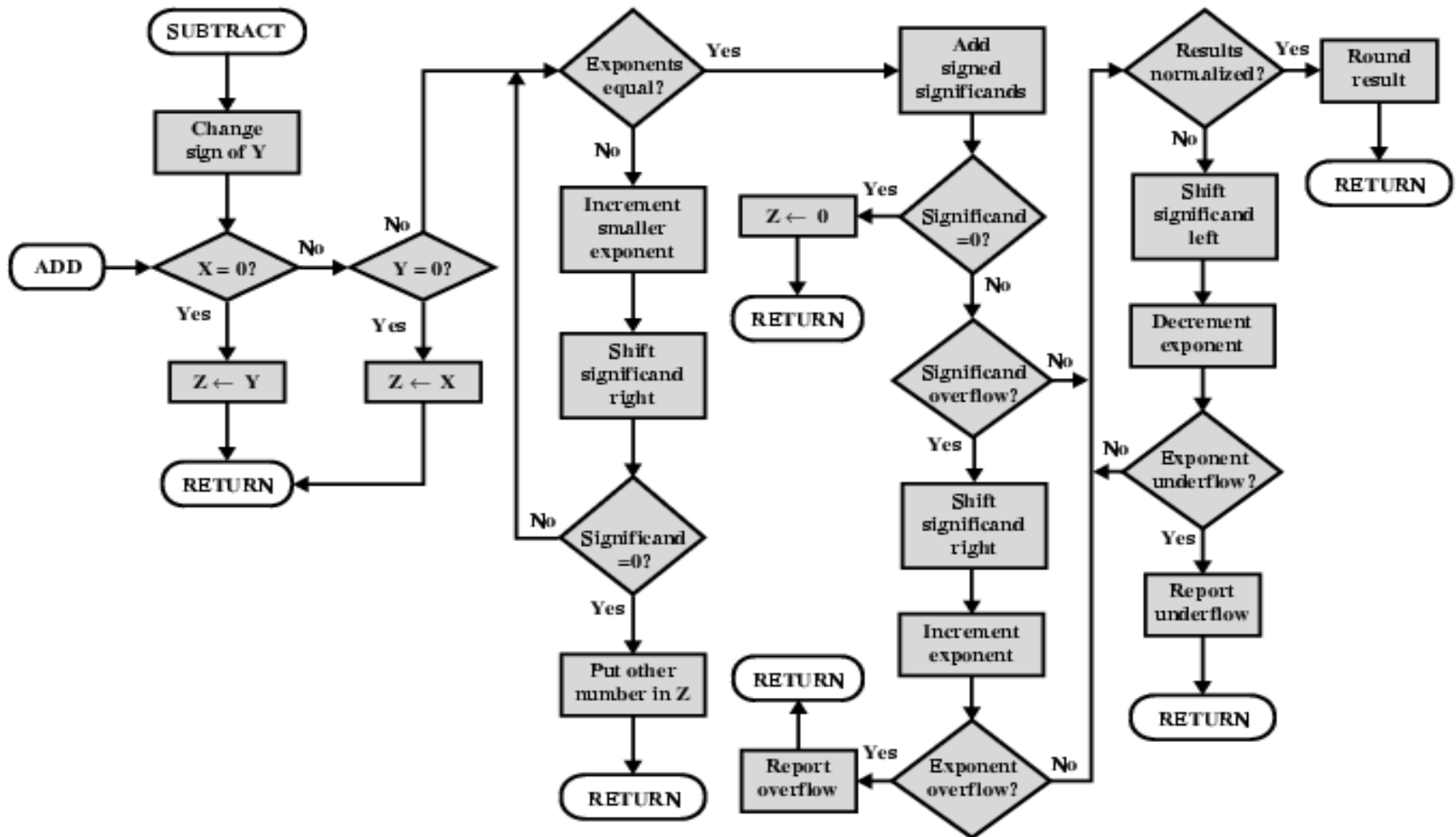
6.if (i == (int)((float) i)) { printf("true"); }

7.if (i == (int)((double) i)) { printf("true"); }

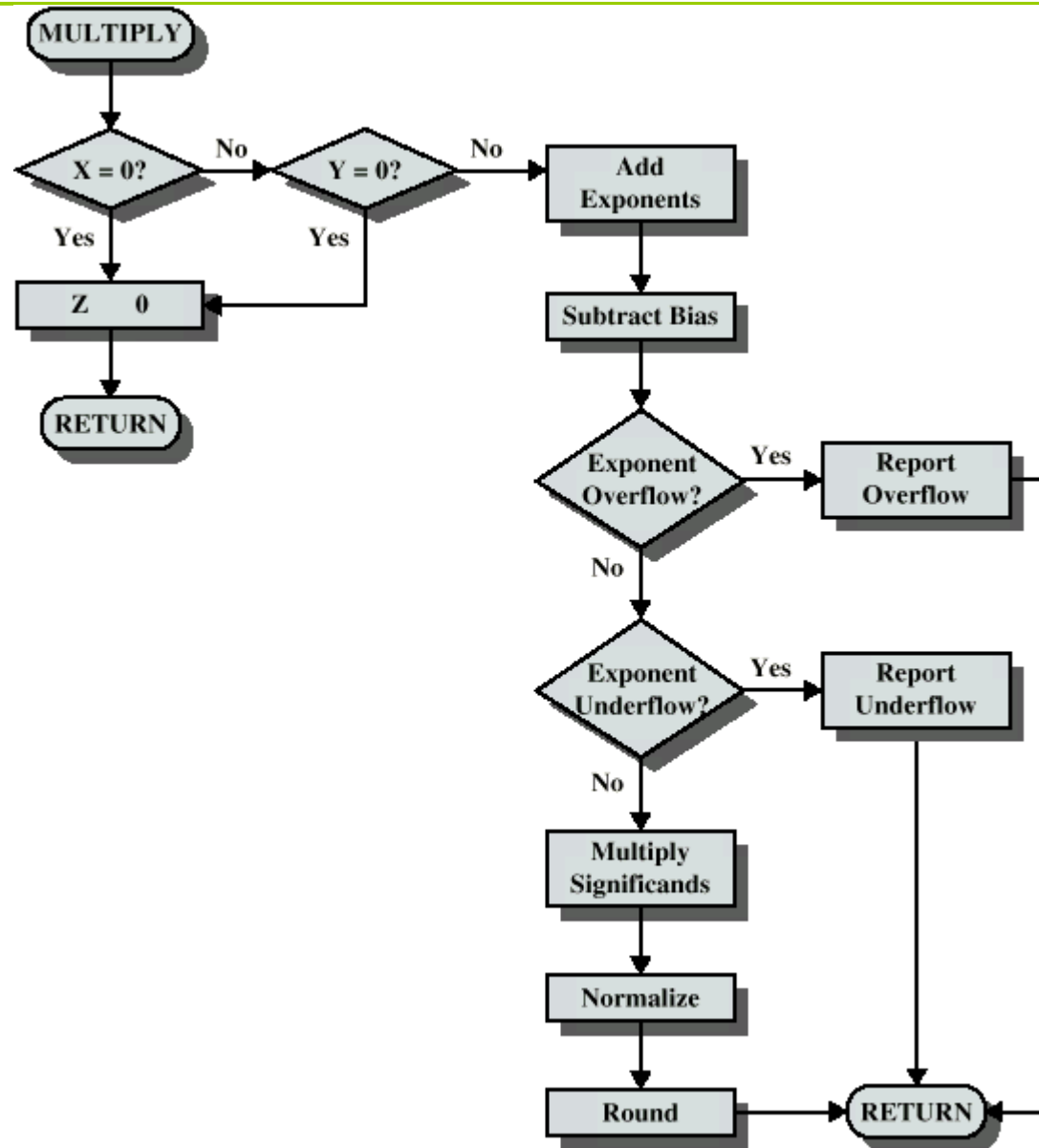
8.if (f == (float)((int) f)) { printf("true"); }

9.if (f == (double)((int) f)) { printf("true"); }

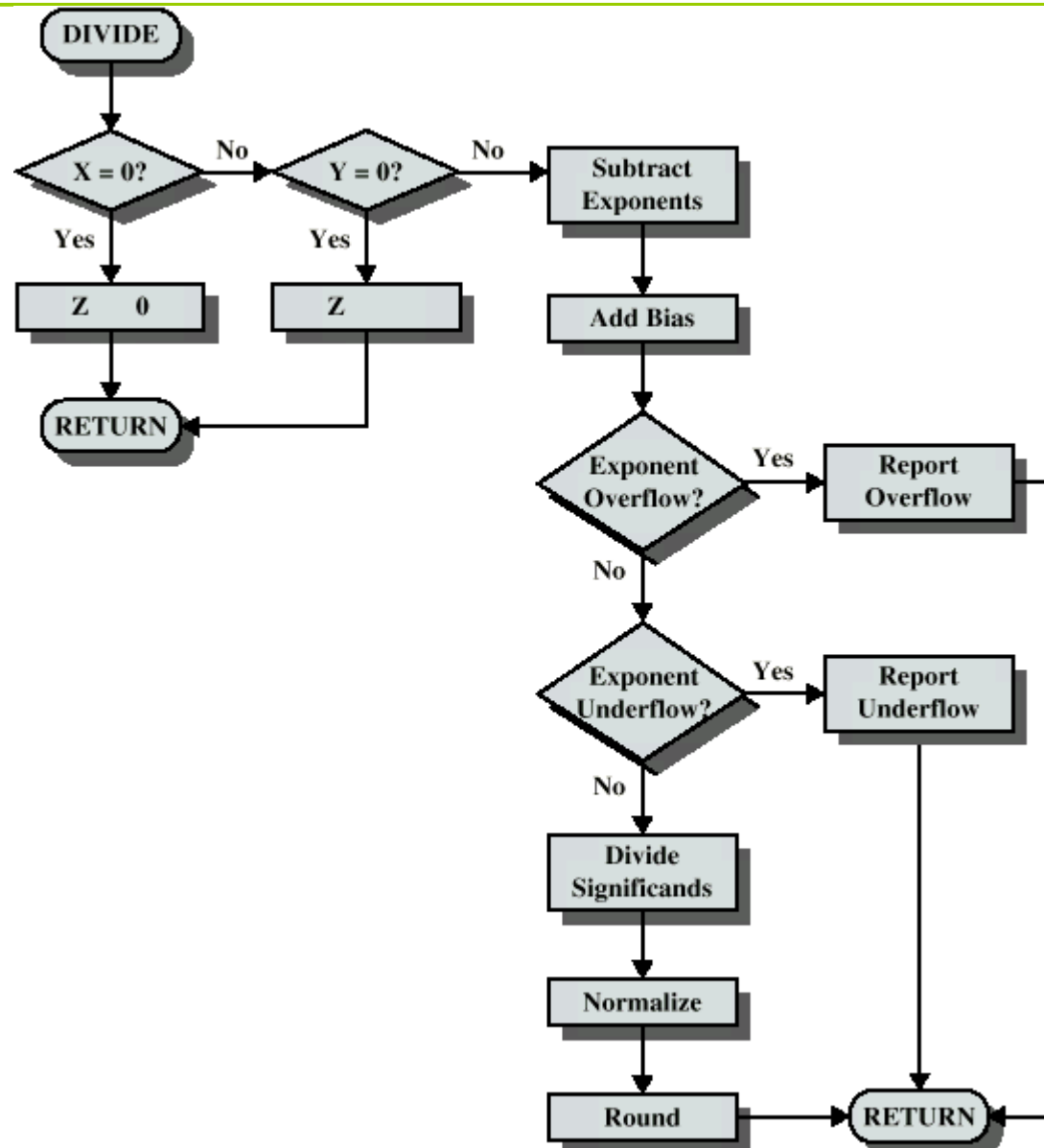
Phép cộng, trừ số chấm động



Phép nhân số chấm động



Phép chia số chấm động



Tham khảo

- Chương 3, P&H
- Chương 9, William Stallings

