Chương 4
SỐ HỌC MÁY TÍNH

Nguyễn Kim Khánh
Trường Đại học Bách khoa Hà Nội

Nội dung học phần

Chương 1. Giới thiệu chung

Chương 2. Cơ bản về logic số

Chương 3. Hệ thống máy tính

Chương 4. Số học máy tính

Chương 5. Kiến trúc tập lệnh

Chương 6. Bộ xử lý

Chương 7. Bộ nhớ máy tính

Chương 8. Hệ thống vào-ra

Chương 9. Các kiến trúc song song

Nội dung chương 4

4.1. Biểu diễn số nguyên

4.2. Phép cộng và phép trừ số nguyên

4.3. Phép nhân và phép chia số nguyên

4.4. Số dấu phẩy động

4.1. Biểu diễn số nguyên

- Số nguyên không dấu (Unsigned Integer)
- Số nguyên có dấu (Signed Integer)

1. Biểu diễn số nguyên không dấu

- Nguyên tắc tổng quát: Dùng n bit biểu diễn số nguyên không dấu A:

$$a_{n-1}a_{n-2}\ldots a_2a_1a_0$$

Giá trị của A được tính như sau:

$$A = \sum_{i=0}^{n-1} a_i 2^i$$

Dải biểu diễn của A: $[0, 2^n - 1]$

Ví dụ 1

- Biểu diễn các số nguyên không dấu sau đây bằng 8-bit:

A = 41 ; B = 150

Giải:

A = 41 = 32 + 8 + 1 = $2^5 + 2^3 + 2^0$

41 = 0010 1001

B = 150 = 128 + 16 + 4 + 2 = $2^7 + 2^4 + 2^2 + 2^1$

150 = 1001 0110

Ví dụ 2

- Cho các số nguyên không dấu M, N được biểu diễn bằng 8-bit như sau:
  - M = 0001 0010
  - N = 1011 1001

  Xác định giá trị của chúng ?

Giải:

- M = 0001 0010 = $2^4 + 2^1 = 16 + 2 = 18$
- N = 1011 1001 = $2^7 + 2^5 + 2^4 + 2^3 + 2^0$

  $= 128 + 32 + 16 + 8 + 1 = 185$

Với n = 8 bit

Biểu diễn được các giá trị từ 0 đến 255 ($2^8 - 1$)

Chú ý:

1111 1111

\+ 0000 0001

1 0000 0000

có *nhớ ra ngoài*

(Carry out)

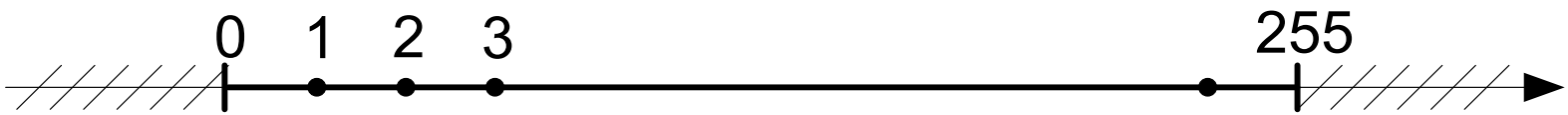
255 + 1 = 0 ???

do vượt ra khỏi dải biểu diễn

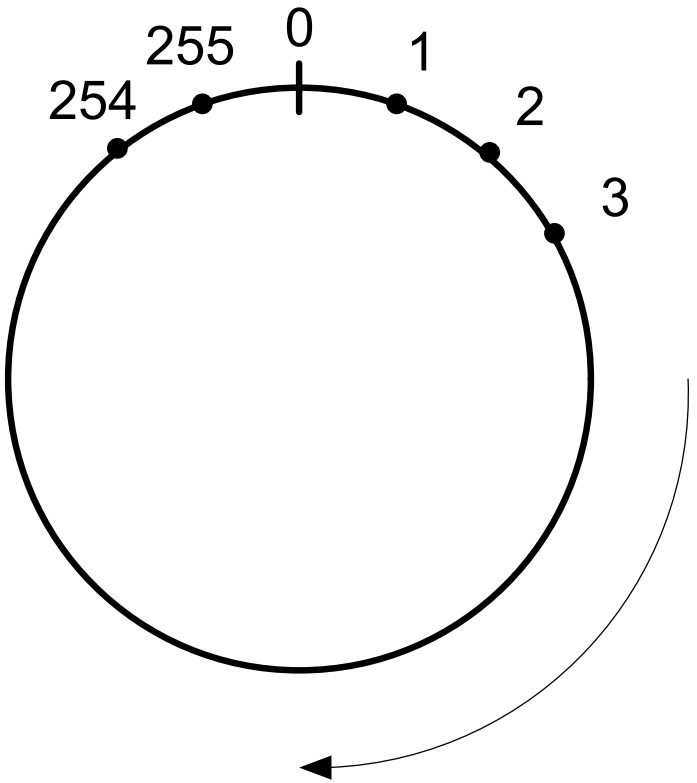
Biểu diễn nhị phân	Giá trị thập phân
0000 0000	0
0000 0001	1
0000 0010	2
0000 0011	3
0000 0100	4
...	
1111 1110	254
1111 1111	255

Trục số học với n = 8 bit

Trục số học:



Trục số học máy tính:



Với n = 16 bit, 32 bit, 64 bit

- n= 16 bit: dải biểu diễn từ 0 đến 65535 ($2^{16} - 1$)
  - 0000 0000 0000 0000 = 0
  - ...
  - 0000 0000 1111 1111 = 255
  - 0000 0001 0000 0000 = 256
  - ...
  - 1111 1111 1111 1111 = 65535
- n= 32 bit: dải biểu diễn từ 0 đến $2^{32} - 1$
- n= 64 bit: dải biểu diễn từ 0 đến $2^{64} - 1$

2. Biểu diễn số nguyên có dấu

Số bù một và Số bù hai

- Định nghĩa: Cho một số nhị phân A được biểu diễn bằng n bit, ta có:
  - Số bù một của A = $(2^n-1) - A$
  - Số bù hai của A = $2^n - A$
- Số bù hai của A = (Số bù một của A) +1

Ví dụ

Với n = 8 bit, cho A = 0010 0101

- Số bù một của A được tính như sau:

  1111 1111 $(2^8 - 1)$

  \- 0010 0101 (A)

  1101 1010

  → đảo các bit của A

- Số bù hai của A được tính như sau:

  1 0000 0000 (2^8)

  \- 0010 0101 (A)

  1101 1011

  → thực hiện khó khăn

Quy tắc tìm Số bù một và Số bù hai

- Số bù một của A = đảo giá trị các bit của A
- (Số bù hai của A) = (Số bù một của A) + 1
- Ví dụ:
  - Cho A = 0010 0101
  - Số bù một của A = 1101 1010
    + 1
  - Số bù hai của A = 1101 1011
- Nhận xét:

  A = 0010 0101

  Số bù hai của A = + 1101 1011

  1 0000 0000 = 0

  (bỏ qua bit nhớ ra ngoài)

  → Số bù hai của A = -A

Biểu diễn số nguyên có dấu theo mã bù hai

Nguyên tắc tổng quát: Dùng n bit biểu diễn số nguyên có dấu A:

$$a_{n-1}a_{n-2}\ldots a_2a_1a_0$$

- Với A là số dương: bit $a_{n-1} = 0$, các bit còn lại biểu diễn độ lớn như số không dấu
- Với A là số âm: được biểu diễn bởi số bù hai của số dương tương ứng, vì vậy bit $a_{n-1} = 1$

Ví dụ

- Biểu diễn các số nguyên có dấu sau đây bằng 8-bit:

A = + 58 ; B = - 80

Giải:

A = + 58 = 0011 1010

B = - 80

Ta có: + 80 = 0101 0000

Số bù một = 1010 1111

\+ 1

Số bù hai = 1011 0000

Vậy: B = - 80 = 1011 0000

Xác định giá trị của số dương

- Dạng tổng quát của số dương:

$$0a_{n-2} \ldots a_2 a_1 a_0$$

- Giá trị của số dương:

$$A = \sum_{i=0}^{n-2} a_i 2^i$$

- Dải biểu diễn cho số dương: [0, +(2^{n-1} - 1)]

Xác định giá trị của số âm

- Dạng tổng quát của số âm:

$$1a_{n-2}\ldots a_2 a_1 a_0$$

- Giá trị của số âm:

$$A = -2^{n-1} + \sum_{i=0}^{n-2} a_i 2^i$$

Chứng minh?

- Dải biểu diễn cho số âm: $[-2^{n-1}, -1]$

Công thức xác định giá trị số âm

$$A = 1\ a_{n-2}\ a_{n-3}\ \ldots\ a_2\ a_1\ a_0$$

$$-A = 0\ \overline{a_{n-2}}\ \overline{a_{n-3}}\ \ldots\ \overline{a_2}\ \overline{a_1}\ \overline{a_0} + 1$$

$$= 11\ldots111 -\ a_{n-2}\ a_{n-3}\ \ldots\ a_2\ a_1\ a_0 + 1$$

$$= (2^{n-1} - 1) - \left(\sum_{i=0}^{n-2} a_i 2^i\right) + 1$$

$$A = -2^{n-1} + \sum_{i=0}^{n-2} a_i 2^i$$

Công thức tổng quát cho số nguyên có dấu

- Dạng tổng quát của số nguyên có dấu A:

$$a_{n-1}a_{n-2}\ldots a_2a_1a_0$$

- Giá trị của A được xác định như sau:

$$A = -a_{n-1}2^{n-1} + \sum_{i=0}^{n-2} a_i 2^i$$

- Dải biểu diễn: $[-(2^{n-1}), +(2^{n-1}-1)]$

Ví dụ

- Hãy xác định giá trị của các số nguyên có dấu được biểu diễn theo mã bù hai với 8-bit như dưới đây:
  - P = 0110 0010
  - Q = 1101 1011

Giải:

- P = 0110 0010 = $2^6 + 2^5 + 2^1 = 64 + 32 + 2 = +98$
- Q = 1101 1011 = $-2^7 + 2^6 + 2^4 + 2^3 + 2^1 + 2^0$
  $= -128 + 64 + 16 + 8 + 2 + 1 = -37$

Với n = 8 bit

- Biểu diễn được các giá trị từ -2^7 đến $+2^7-1$
  - -128 đến +127
  - Chỉ có một giá trị 0
  - Không biểu diễn cho giá trị +128

Chú ý:

+127 + 1 = -128

(-128)+(-1) = +127

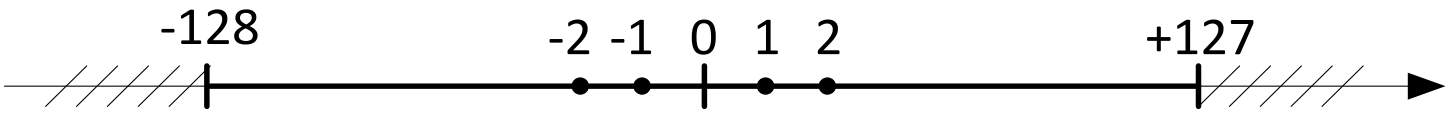
có tràn xảy ra (Overflow)

(do vượt ra khỏi dải biểu diễn)

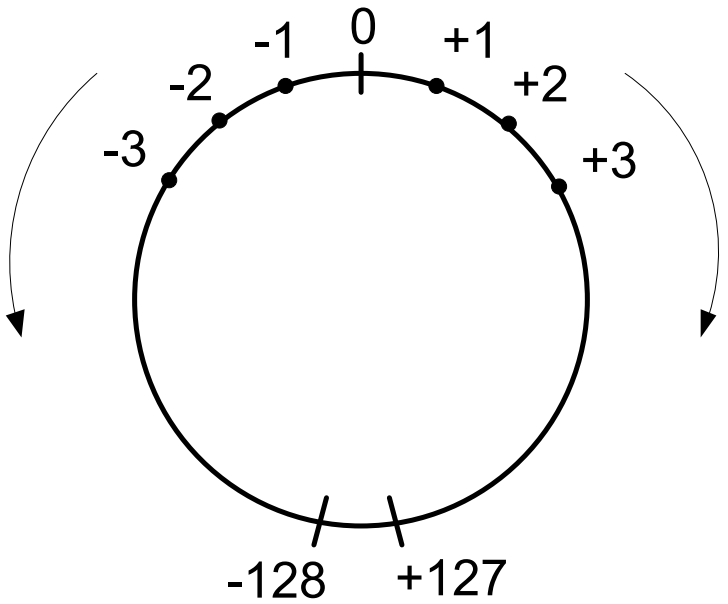
Giá trị thập phân	Biểu diễn bù hai
0	0000 0000
+1	0000 0001
+2	0000 0010
	...
+126	0111 1110
+127	0111 1111
-128	1000 0000
-127	1000 0001
	...
-2	1111 1110
-1	1111 1111

Trục số học số nguyên có dấu với n = 8 bit

- Trục số học:



- Trục số học máy tính:



Với n = 16 bit, 32 bit, 64 bit

- Với n = 16bit: biểu diễn từ -2^{15} đến $2^{15}-1$
  - 0000 0000 0000 0000 = 0
  - 0000 0000 0000 0001 = +1
  - ...
  - 0111 1111 1111 1111 = +32767 ($2^{15}-1$)
  - 1000 0000 0000 0000 = -32768 (-2^{15})
  - 1000 0000 0000 0001 = -32767
  - ...
  - 1111 1111 1111 1111 = -1
- Với n = 32bit: biểu diễn từ -2^{31} đến $2^{31}-1$
- Với n = 64bit: biểu diễn từ -2^{63} đến $2^{63}-1$

Mở rộng bit cho số nguyên

- Mở rộng theo số không dấu (Zero-extended): thêm các bit 0 vào bên trái
- Mở rộng theo số có dấu (Sign-extended):
  - Số dương:

    +19 = 0001 0011 (8bit)

    +19 = 0000 0000 0001 0011 (16bit)

    → thêm các bit 0 vào bên trái
  - Số âm:

    - 19 = 1110 1101 (8bit)

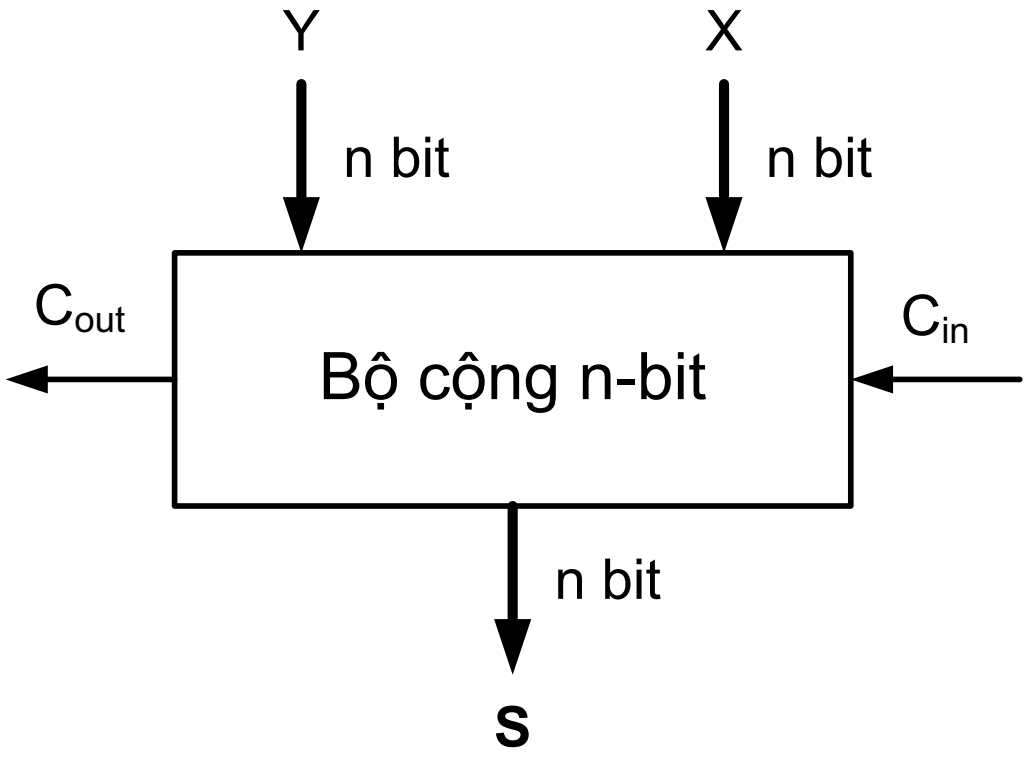
    - 19 = 1111 1111 1110 1101 (16bit)

    → thêm các bit 1 vào bên trái

4.2. Thực hiện phép cộng/trừ với số nguyên

1. Phép cộng số nguyên không dấu

Bộ cộng n-bit



Nguyên tắc cộng số nguyên không dấu

- Khi cộng hai số nguyên không dấu n-bit, kết quả nhận được là n-bit:
  - Nếu $C_{out} = 0$ → nhận được kết quả đúng
  - Nếu $C_{out} = 1$ → nhận được kết quả sai, do có *nhớ ra ngoài (Carry Out)*
- Hiện tượng *nhớ ra ngoài* xảy ra khi: tổng > $(2^n - 1)$

Ví dụ cộng số nguyên không dấu

- 57 = 0011 1001
  + 34 = + 0010 0010
  91 0101 1011 = 64+16+8+2+1=91 → đúng

- 209 = 1101 0001
  + 73 = + 0100 1001
  282 1 0001 1010
  kết quả = 0001 1010 =16+8+2=26 → sai
  do có *nhớ ra ngoài* (C_{out}=1)

Để có kết quả đúng, ta thực hiện cộng theo 16-bit:

209 = 0000 0000 1101 0001
\+ 73 = + 0000 0000 0100 1001
0000 0001 0001 1010 = 256+16+8+2 = 282

2. Phép đảo dấu

- Ta có:

+ 37	=	0010 0101		
bù một	=	1101 1010		
	+	1		
bù hai	=	1101 1011	=	-37

- Lấy bù hai của số âm:

- 37	=	1101 1011		
bù một	=	0010 0100		
	+	1		
bù hai	=	0010 0101	=	+37

- Kết luận: *Phép đảo dấu số nguyên trong máy tính thực chất là lấy bù hai*

3. Cộng số nguyên có dấu

- Khi cộng hai số nguyên có dấu n-bit, kết quả nhận được là n-bit và *không cần quan tâm đến bit C_{out}*
  - Khi cộng hai số khác dấu thì kết quả luôn luôn đúng
  - Khi cộng hai số cùng dấu, nếu dấu kết quả cùng dấu với các số hạng thì kết quả là đúng
  - Khi cộng hai số cùng dấu, nếu kết quả có dấu ngược lại, khi đó có ***tràn*** (Overflow) xảy ra và kết quả bị sai
- Hiện tượng *tràn* xảy ra khi tổng nằm ngoài dải biểu diễn: $[-(2^{n-1}),+(2^{n-1}-1)]$

Ví dụ cộng số nguyên có dấu không tràn

- (+ 70) = 0100 0110
  + (+ 42) = 0010 1010
  + 112 0111 0000 = +112

- (+ 97) = 0110 0001
  + (- 52) = 1100 1100 (+52=0011 0100)
  + 45 1 0010 1101 = +45

- (- 90) = 1010 0110 (+90=0101 1010)
  + (+36) = 0010 0100
  - 54 1100 1010 = - 54

- (- 74) = 1011 0110 (+74=0100 1010)
  +(- 30) = 1110 0010 (+30=0001 1110)
  -104 1 1001 1000 = -104

Ví dụ cộng số nguyên có dấu bị tràn

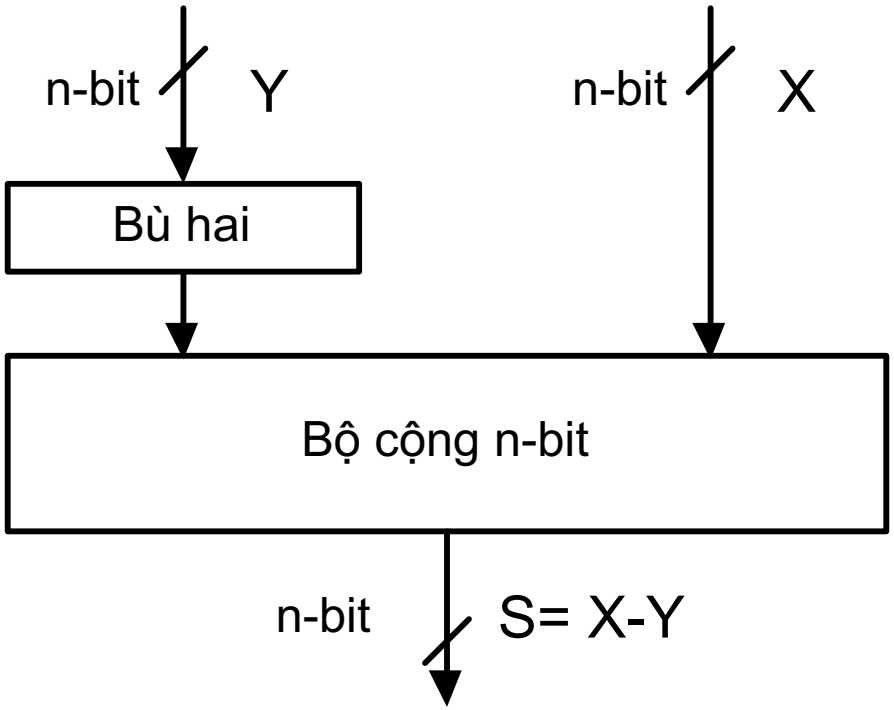
- (+ 75) = 0100 1011
  +(+ 82) = 0101 0010
  +157 1001 1101
  = - 128+16+8+4+1= -99 → sai

- (- 104) = 1001 1000 (+104=0110 1000)
  + (- 43) = 1101 0101 (+ 43 =0010 1011)
  - 147 1 0110 1101
  = 64+32+8+4+1= +109 → sai

- Cả hai ví dụ đều tràn vì tổng nằm ngoài dải biểu diễn [-128, +127]

4. Nguyên tắc thực hiện phép trừ

- Phép trừ hai số nguyên: X-Y = X+(-Y)
- Nguyên tắc: Lấy bù hai của Y để được –Y, rồi cộng với X



4.3. Phép nhân và phép chia số nguyên

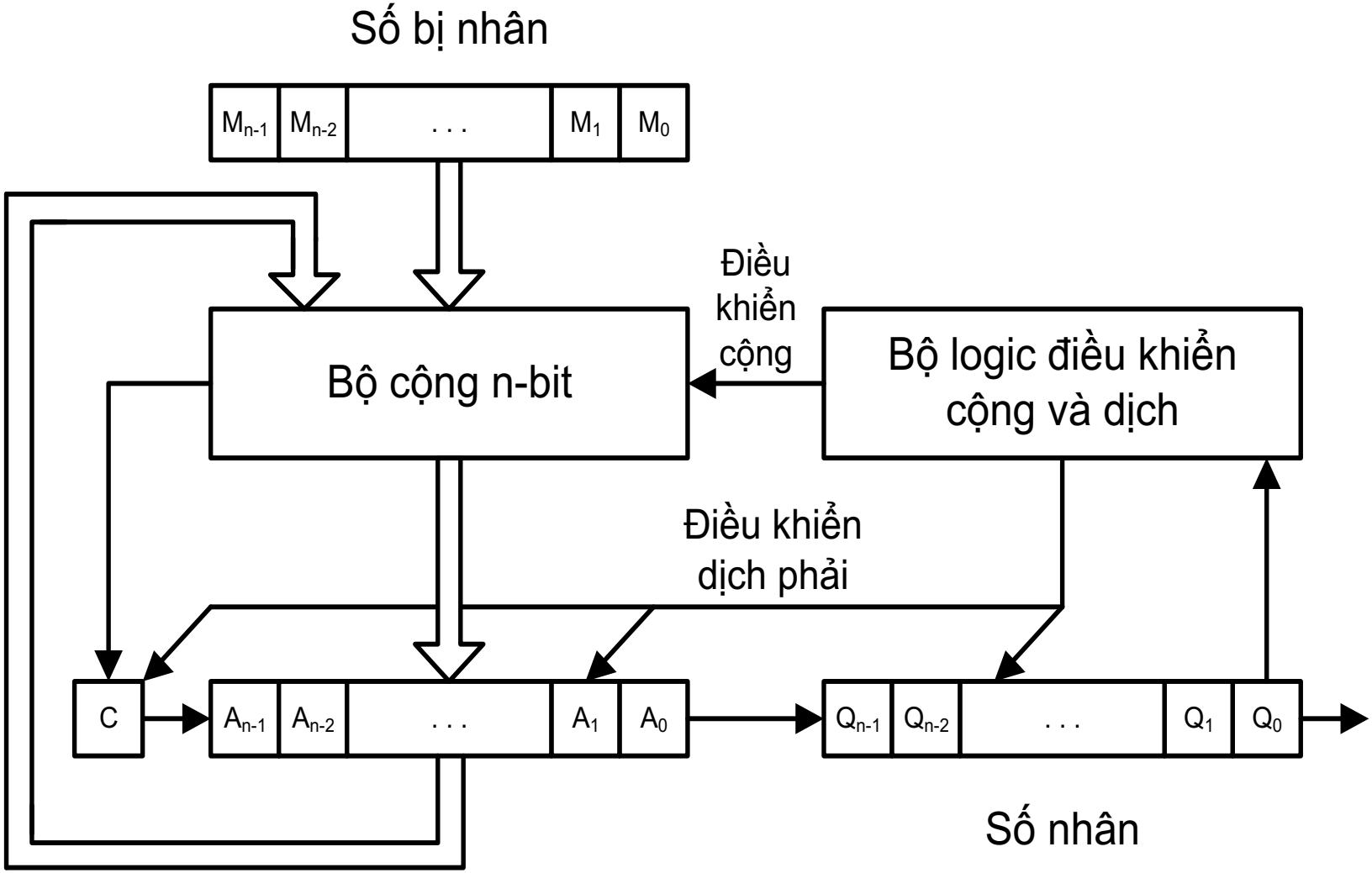
1. Nhân số nguyên không dấu

	1011	Số bị nhân (11)
x	1101	Số nhân (13)
	1011	Các tích riêng phần
	0000	
	1011	
	1011	
	10001111	Tích (143)

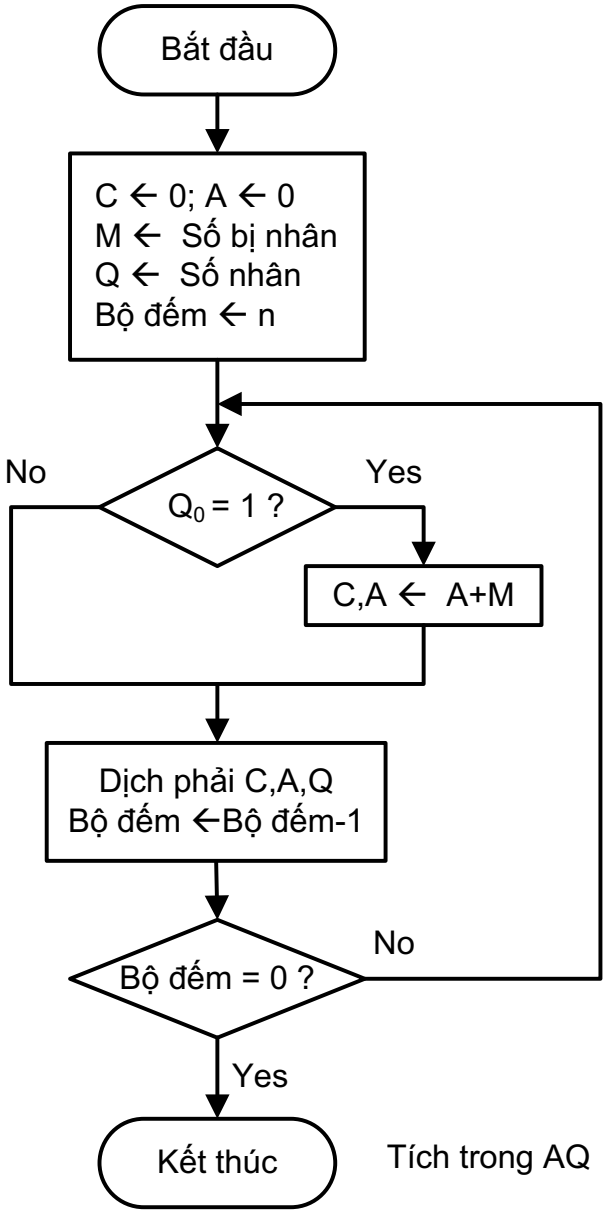
Nhân số nguyên không dấu (tiếp)

- Các tích riêng phần được xác định như sau:
  - Nếu bit của số nhân bằng 0 → tích riêng phần bằng 0
  - Nếu bit của số nhân bằng 1 → tích riêng phần bằng số bị nhân
  - Tích riêng phần tiếp theo được dịch trái một bit so với tích riêng phần trước đó
- Tích bằng tổng các tích riêng phần
- Nhân hai số nguyên n-bit, tích có độ dài 2n bit (không bao giờ tràn)

Bộ nhân số nguyên không dấu



Lưu đồ nhân số nguyên không dấu



Ví dụ nhân số nguyên không dấu

- Số bị nhân M = 1011 (11)
- Số nhân Q = 1101 (13)
- Tích = 1000 1111 (143)

	C	A	Q	
■	0	0000	1101	Các giá trị khởi đầu
		+ 1011		
	0	1011	1101	A ← A + M
■	0	0101	1110	Dịch phải
■	0	0010	1111	Dịch phải
		+ 1011		
	0	1101	1111	A ← A + M
■	0	0110	1111	Dịch phải
		+ 1011		
	1	0001	1111	A ← A + M
■	0	1000	1111	Dịch phải

Ví dụ nhân số nguyên không dấu (tiếp)

- Số bị nhân M = 0110 (6)
- Số nhân Q = 0101 (5)
- Tích = (30)

	C	A	Q	
■	0	0000	0101	Các giá trị khởi đầu
		+ 0110		
	0	0110	0101	A ← A + M
■	0	0011	0010	Dịch phải
■	0	0001	1001	Dịch phải
		+ 0110		
	0	0111	1001	A ← A + M
■	0	0011	1100	Dịch phải
■	0	0001	1110	Dịch phải

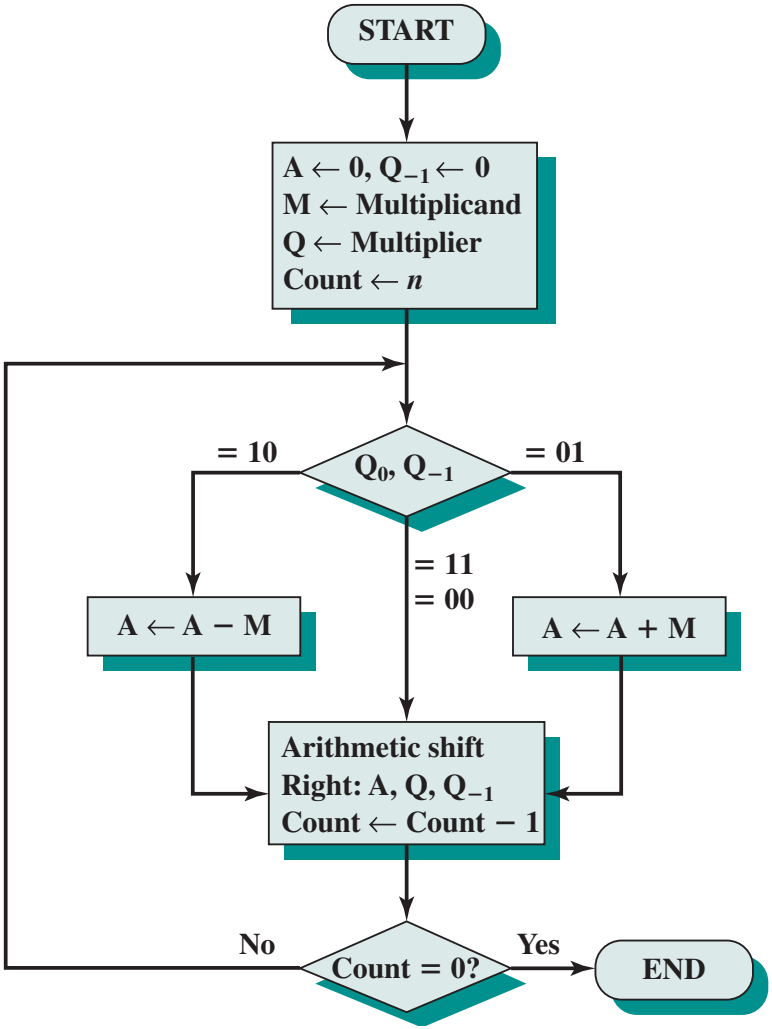
2. Nhân số nguyên có dấu

- Sử dụng thuật giải nhân không dấu
- Sử dụng thuật giải Booth

Sử dụng thuật giải nhân không dấu

- Bước 1. Chuyển đổi số bị nhân và số nhân thành số dương tương ứng
- Bước 2. Nhân hai số dương bằng thuật giải nhân số nguyên không dấu, được tích của hai số dương.
- Bước 3. Hiệu chỉnh dấu của tích:
  - Nếu hai thừa số ban đầu cùng dấu thì giữ nguyên kết quả ở bước 2
  - Nếu hai thừa số ban đầu là khác dấu thì đảo dấu kết quả của bước 2 (lấy bù hai)

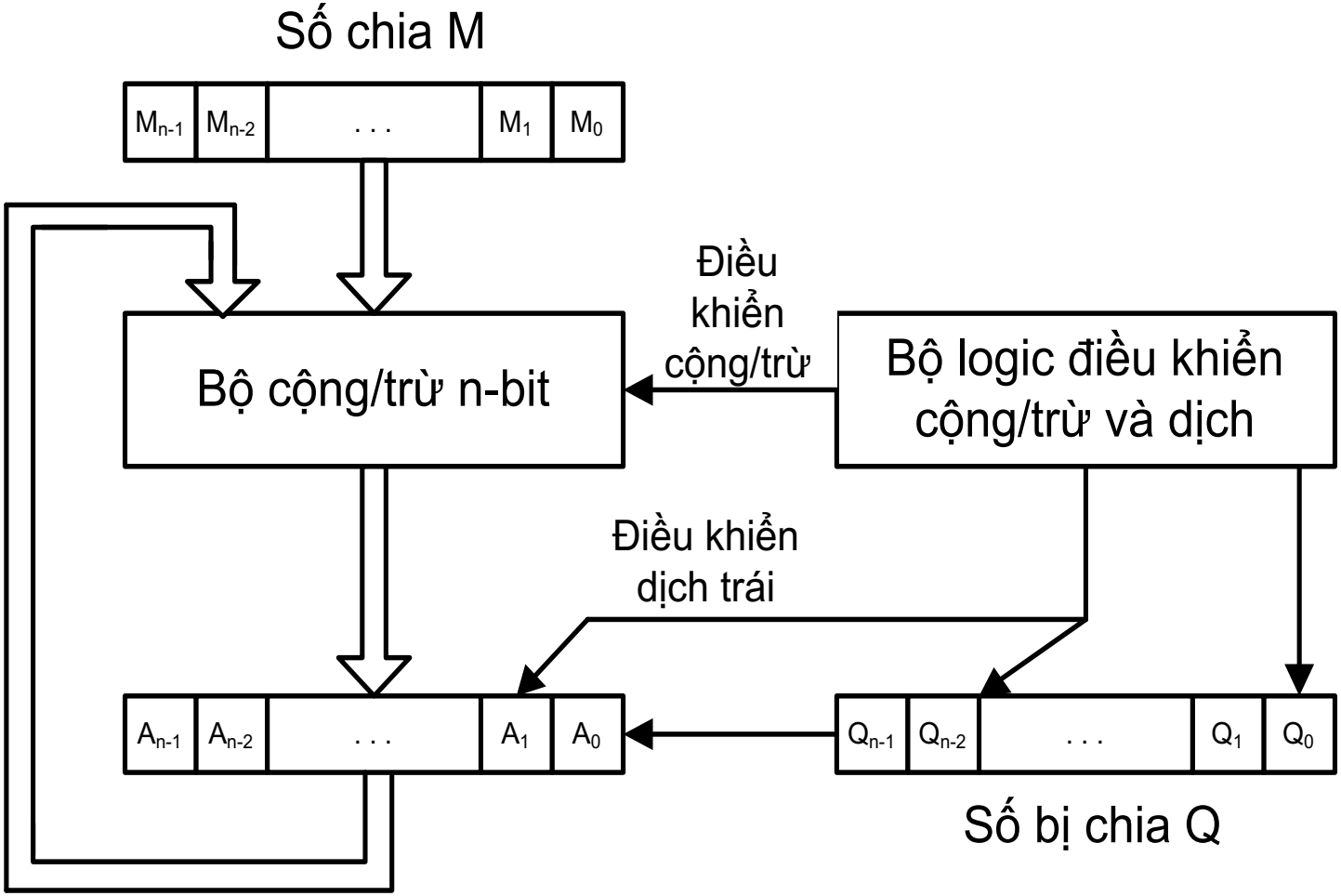
Thuật giải Booth (tham khảo sách COA)



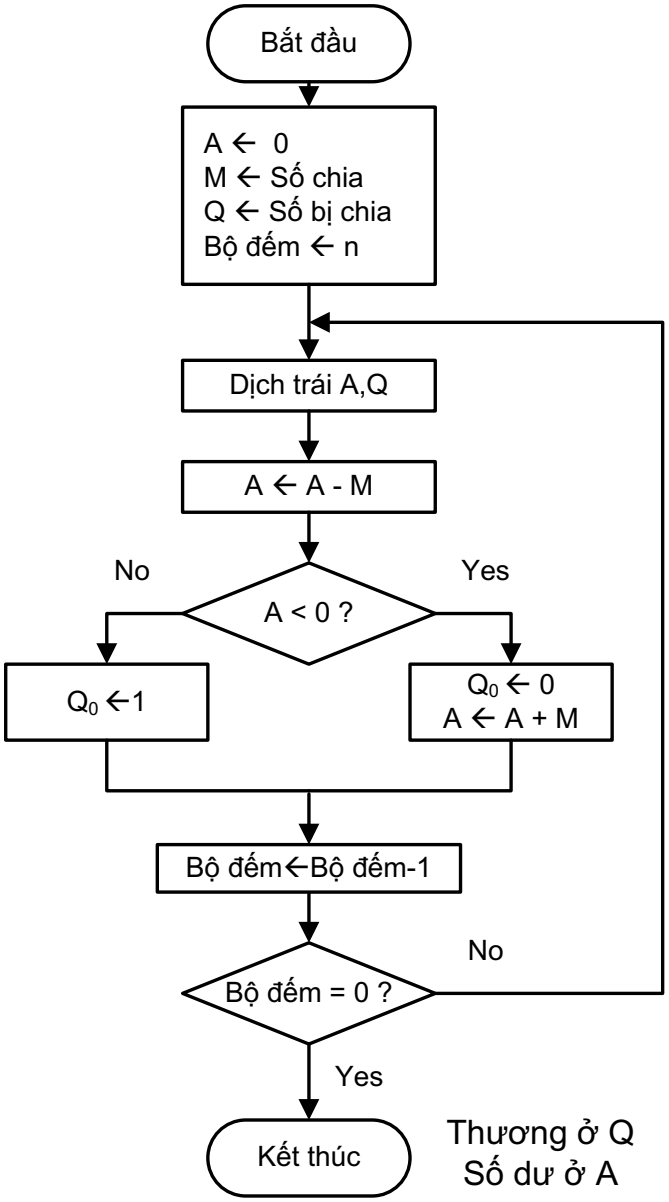
3. Chia số nguyên không dấu

Số bị chia	10010011	\1011	Số chia
	- 1011	00001101	Thương
	001110		
	- 1011		
	001111		
	- 1011		
	100		Phần dư

Bộ chia số nguyên không dấu



Lưu đồ chia số nguyên không dấu



Ví dụ: Q = 1011 (11) M = 0011 (3) → -M = 1101

A	Q		
0000	1011		BĐ = 4
0001	0110	dịch trái	
1101			
1110		A = A – M < 0	
0011			
0001	0110	A = A + M	BĐ = 3
0010	1100	dịch trái	
1101			
1111		A = A – M < 0	
0011			
0010	1100	A = A + M	BĐ = 2
0101	1000	dịch trái	
1101			
0010		A = A – M > 0	
0010	1001		BĐ = 1
0101	0010	dịch trái	
1101			
0010		A = A – M > 0	
0010	0011		BĐ = 0
2	3		

4. Chia số nguyên có dấu

- Bước 1. Chuyển đổi số bị chia và số chia về thành số dương tương ứng.
- Bước 2. Sử dụng thuật giải chia số nguyên không dấu để chia hai số dương, kết quả nhận được là thương Q và phần dư R đều là dương
- Bước 3. Hiệu chỉnh dấu của kết quả như sau:

  (Lưu ý: phép đảo dấu thực chất là thực hiện phép lấy bù hai)

Số bị chia	Số chia	Thương	Số dư
dương	dương	giữ nguyên	giữ nguyên
dương	âm	đảo dấu	giữ nguyên
âm	dương	đảo dấu	đảo dấu
âm	âm	giữ nguyên	đảo dấu

4.4. Số dấu phẩy động

1. Nguyên tắc chung

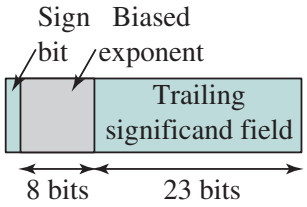
- Floating Point Number → biểu diễn cho số thực
- Tổng quát: một số thực X được biểu diễn theo kiểu số dấu phẩy động như sau:

$$X = \pm M * R^E$$

  - M là phần định trị (Mantissa),
  - R là cơ số (Radix),
  - E là phần mũ (Exponent).

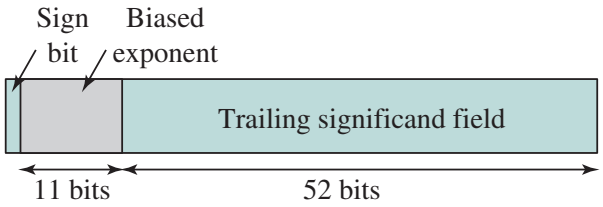
2. Chuẩn IEEE754-2008

- Cơ số R = 2
- Các dạng:
  - Dạng 32-bit



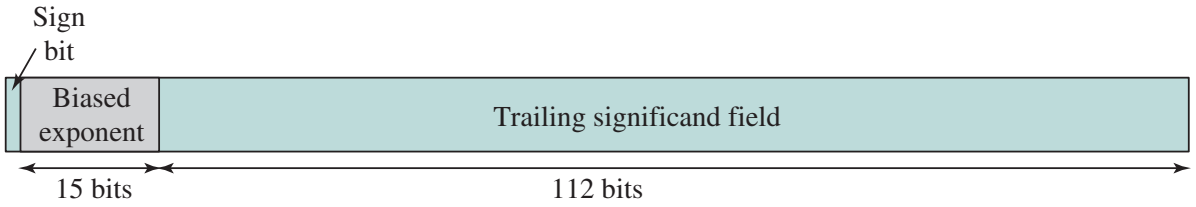
(a) Binary32 format

  - Dạng 64-bit



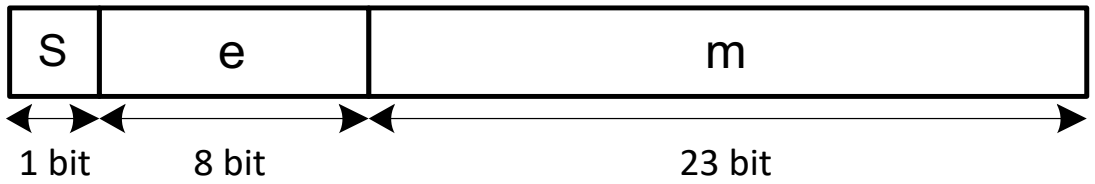
(b) Binary64 format

  - Dạng 128-bit



(c) Binary128 format

Dạng 32-bit



- S là bit dấu:
  - S = 0 → số dương
  - S = 1 → số âm
- e (8 bit) là giá trị dịch chuyển của phần mũ E:
  - e = E + 127 → phần mũ E = e - 127
- m (23 bit) là phần lẻ của phần định trị M:
  - M = 1.m
- Công thức xác định giá trị của số thực:

$$X = (-1)^S * 1.m * 2^{e-127}$$

Ví dụ 1

Xác định giá trị của các số thực được biểu diễn bằng 32-bit sau đây:

- 1100 0001 0101 0110 0000 0000 0000 0000
  - S = 1 → số âm
  - e = $1000\ 0010_{(2)} = 130_{(10)}$ → E = 130 - 127 = 3

  Vậy

  $X = -1.10101100_{(2)} * 2^3 = -1101.011_{(2)} = -13.375_{(10)}$

- 0011 1111 1000 0000 0000 0000 0000 0000 = ?

Ví dụ 2

Biểu diễn số thực $X = 83.75_{(10)}$ về dạng số dấu phẩy động IEEE754 32-bit

Giải:

- $X = 83.75_{(10)} = 1010011.11_{(2)} = 1.01001111 \times 2^6$
- Ta có:
  - $S = 0$ vì đây là số dương
  - $E = e - 127 = 6 \rightarrow e = 127 + 6 = 133_{(10)} = 1000\ 0101_{(2)}$
- Vậy:

$X = 0100\ 0010\ 1010\ 0111\ 1000\ 0000\ 0000\ 0000$

Các qui ước đặc biệt

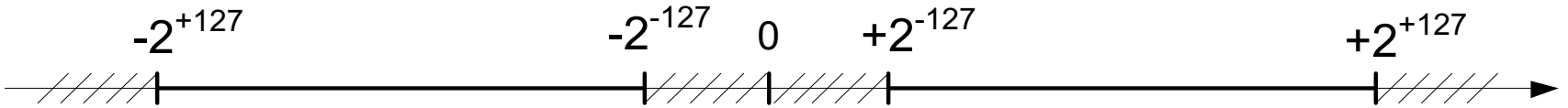
- Các bit của e bằng 0, các bit của m bằng 0, thì X = ± 0
  x000 0000 0000 0000 0000 0000 0000 0000 → X = ± 0

- Các bit của e bằng 1, các bit của m bằng 0, thì X = ± ∞
  x111 1111 1000 0000 0000 0000 0000 0000 → X =± ∞

- Các bit của e bằng 1, còn m có ít nhất một bit bằng 1, thì nó không biểu diễn cho số nào cả (NaN - not a number)

Dải giá trị biểu diễn

- 2^{-127} đến 2^{+127}
- 10^{-38} đến 10^{+38}



Dạng 64-bit

- S là bit dấu
- e (11 bit) là giá trị dịch chuyển của phần mũ E:
  - e = E + 1023 → phần mũ E = e - 1023
- m (52 bit): phần lẻ của phần định trị M
- Giá trị số thực:

$$X = (-1)^S * 1.m * 2^{e-1023}$$

- Dải giá trị biểu diễn: 10^{-308} đến 10^{+308}

Dạng 128-bit

- S là bit dấu
- e (15 bit) là giá trị dịch chuyển của phần mũ E:
  - e = E + 16383 → phần mũ E = e - 16383
- m (112 bit): phần lẻ của phần định trị M
- Giá trị số thực:

$$X = (-1)^S * 1.m * 2^{e-16383}$$

- Dải giá trị biểu diễn: 10^{-4932} đến 10^{+4932}

3. Thực hiện phép toán số dấu phẩy động

- $X1 = M1 * R^{E1}$
- $X2 = M2 * R^{E2}$
- Ta có
  - $X1 * X2 = (M1 * M2) * R^{E1+E2}$
  - $X1 / X2 = (M1 / M2) * R^{E1-E2}$
  - $X1 \pm X2 = (M1 * R^{E1-E2} \pm M2) * R^{E2}$, với $E2 \geq E1$

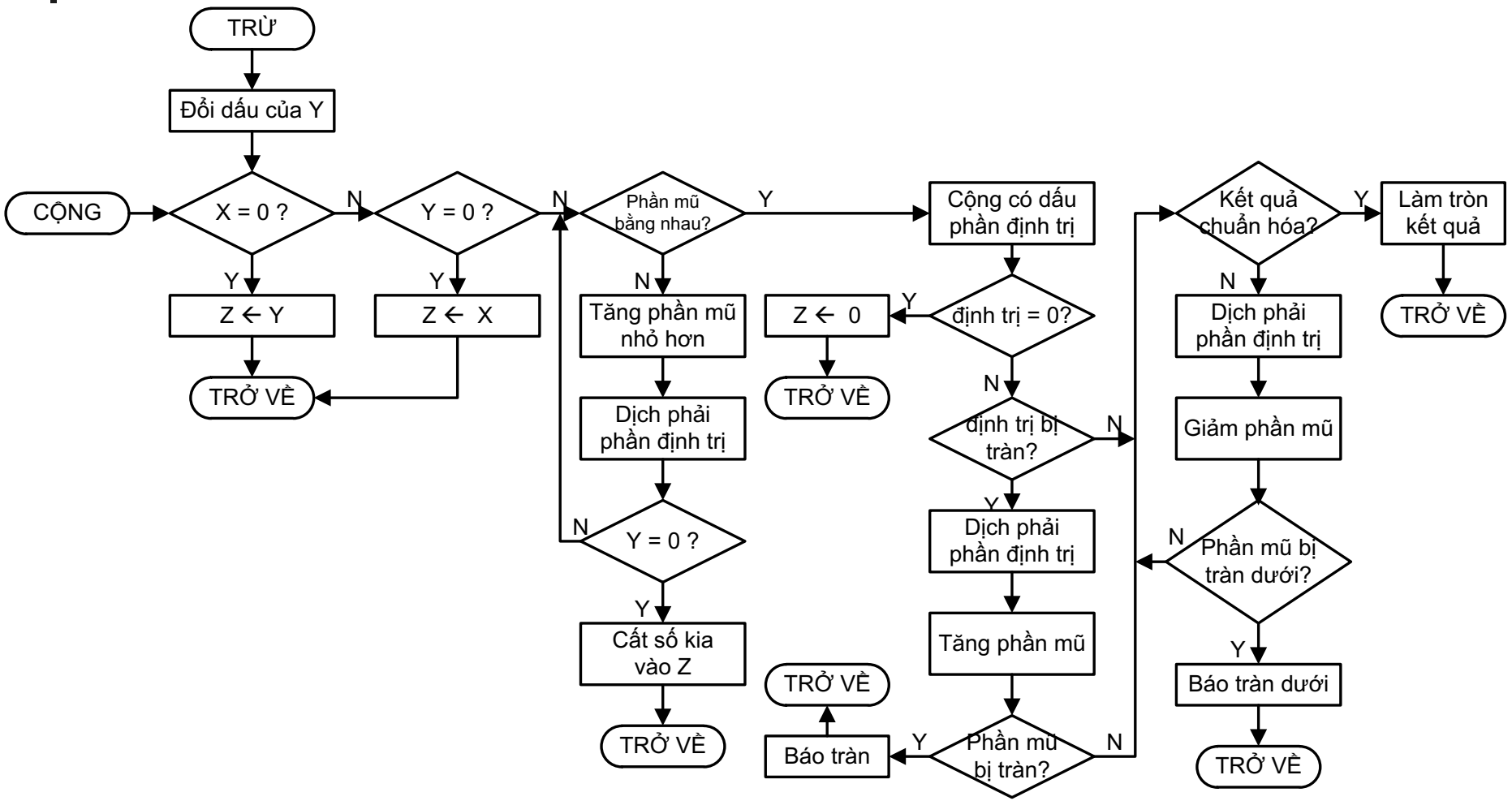
Các khả năng tràn số

- Tràn trên số mũ (Exponent Overflow): mũ dương vượt ra khỏi giá trị cực đại của số mũ dương có thể (→ ∞)
- Tràn dưới số mũ (Exponent Underflow): mũ âm vượt ra khỏi giá trị cực đại của số mũ âm có thể (→ 0)
- Tràn trên phần định trị (Mantissa Overflow): cộng hai phần định trị có cùng dấu, kết quả bị nhớ ra ngoài bit cao nhất
- Tràn dưới phần định trị (Mantissa Underflow): Khi hiệu chỉnh phần định trị, các số bị mất ở bên phải phần định trị

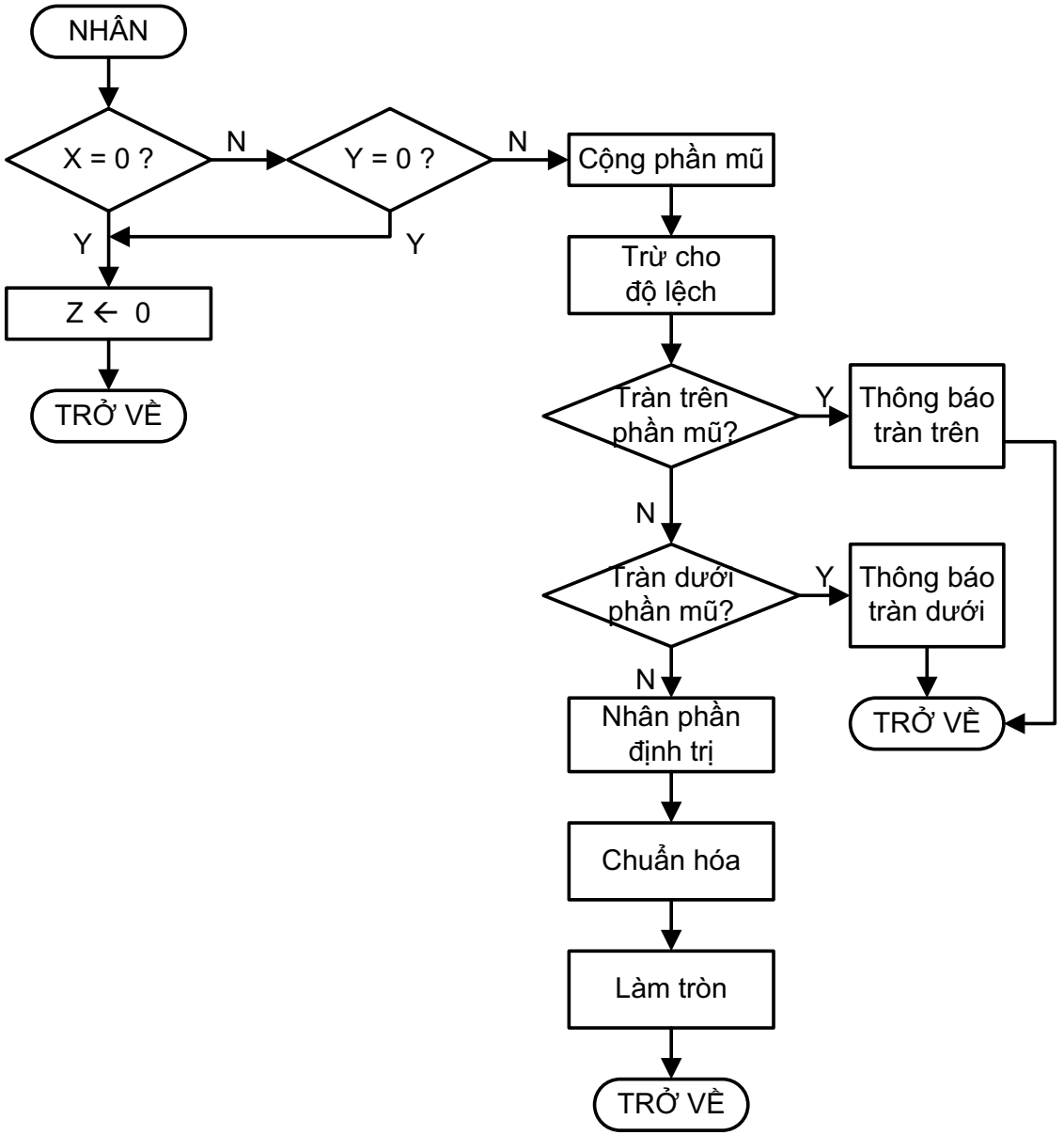
Phép cộng và phép trừ

- Kiểm tra các số hạng có bằng 0 hay không
- Hiệu chỉnh phần định trị
- Cộng hoặc trừ phần định trị
- Chuẩn hoá kết quả

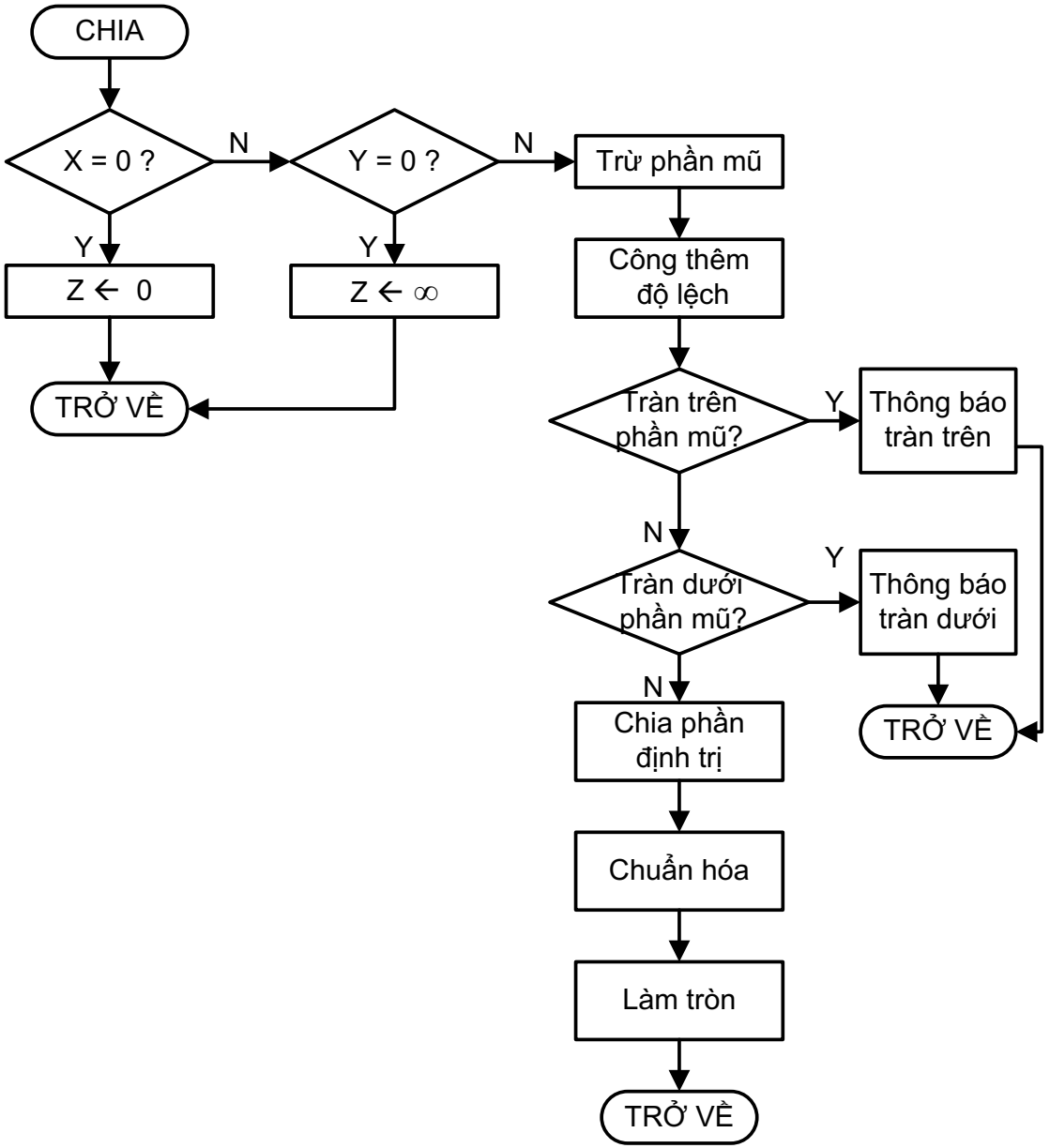
Thuật toán cộng/trừ số dấu phẩy động



Thuật toán nhân số dấu phẩy động



Thuật toán chia số dấu phẩy động



Hết chương 4