TRƯỜNG ĐẠI HỌC THỦY LỢI
KHOA CÔNG NGHỆ THÔNG TIN

KIẾN TRÚC MÁY TÍNH

Giảng viên: TS. Đoàn Thị Quế

Bộ môn: Mạng và an toàn thông tin

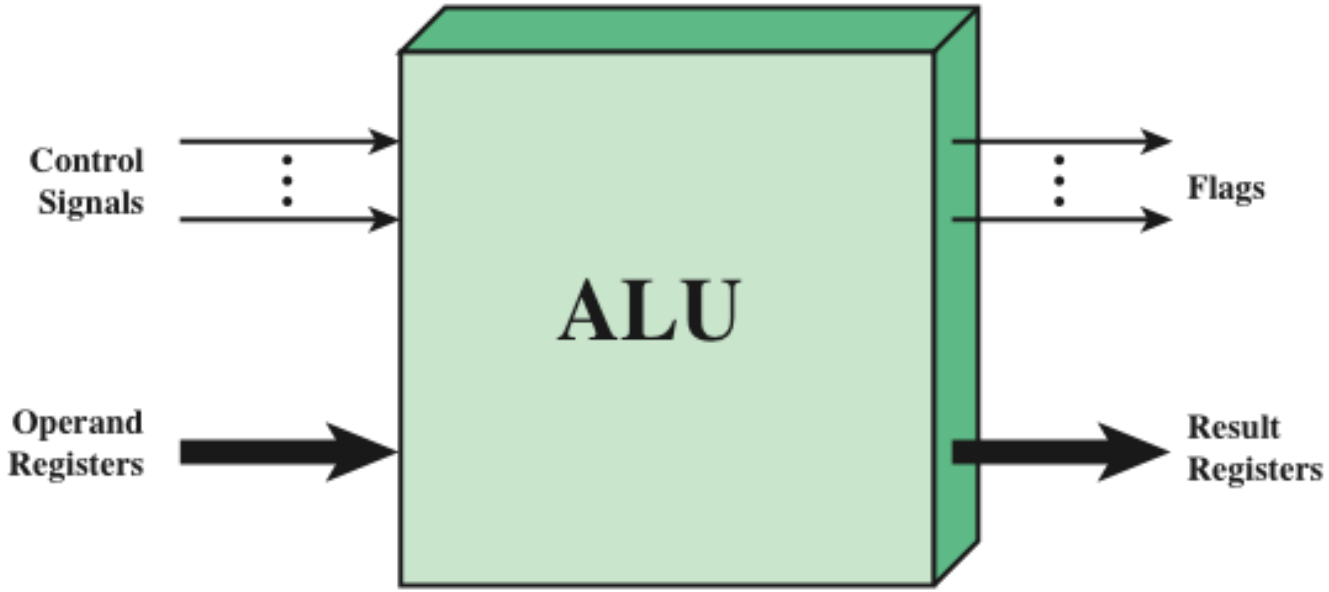
Chương 8. Bộ xử lý số học

1. Đơn vị số học và logic
2. Biểu diễn số nguyên
3. Phép toán số học với số nguyên
4. Biểu diễn dấu chấm động
5. Phép toán với số dấu chấm động

1. Đơn vị số học & logic (ALU)

- Phần của máy tính thực hiện phép toán số học và logic trên dữ liệu
- Tất cả các bộ phận khác trong hệ thống máy tính (CU, thanh ghi, bộ nhớ, I/O) đưa dữ liệu tới ALU để xử lý, sau đó gửi lại kết quả
- Khối ALU được thực hiện sử dụng các linh kiện logic số có thể lưu trữ các số nhị phân và thực hiện các phép toán logic Boolean đơn giản

Đầu vào và đầu ra ALU



- Control Signals: các tín hiệu điều khiển được gửi đến từ CU, điều khiển hoạt động của ALU
- Operand registers: các thanh ghi lưu trữ giá trị toán hạng của phép toán
- Result registers: các thanh ghi lưu trữ kết quả phép toán
- Flags: các cờ. Ví dụ: cờ tràn để đánh dấu kết quả tính toán vượt quá kích thước thanh ghi đang lưu trữ

2. Biểu diễn số nguyên

- Trong hệ thống số nhị phân, các số được biểu diễn dưới dạng các chữ số 0 hoặc 1 cùng với dấu âm (dành cho số âm) và dấu chấm (dành cho số thực có phần phân)

$$-1101.0101_2 = -13.3125_{10}$$

- Với mục đích lưu trữ và xử lý trong máy tính, chúng ta không có các ký hiệu đặc biệt sử dụng để biểu diễn dấu âm và dấu chấm.
- Chỉ có thể sử dụng chữ số nhị phân (0,1) để biểu thị số

Biểu diễn số nguyên không dấu

An 8-bit word can represent the numbers from 0 to 255, such as

$$00000000 = 0$$
$$00000001 = 1$$
$$00101001 = 41$$
$$10000000 = 128$$
$$11111111 = 255$$

In general, if an n-bit sequence of binary digits $a_{n-1}a_{n-2} \ldots a_1a_0$ is interpreted as an unsigned integer A, its value is

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

Biểu diễn số nguyên có dấu

- Có một số quy ước thay thế được sử dụng để biểu diễn số nguyên âm cũng như số nguyên dương
  - **Biểu diễn dấu – độ lớn**
  - **Biểu diễn bù 2**
- Đặc điểm chung: cả hai dạng biểu diễn đều sử dụng ***bit quan trọng nhất (MSB - most significant bit)*** làm bit dấu
  - Nếu bit dấu là 0 thì số đó là dương
  - Nếu bit dấu là 1 thì số đó là số âm

a. Biểu diễn dấu – độ lớn (Sign-magnitude representation)

- Đây là dạng biểu diễn đơn giản nhất
- Trong một từ n bit:
  - Bit ngoài cùng bên trái làm bit dấu: 0 (+), 1(-)
  - Sử dụng (n-1) bên phải biểu diễn độ lớn của số
- Ví dụ:

$$+18 = 00010010$$
$$-18 = 10010010 \quad \text{(sign magnitude)}$$

Bài tập áp dụng

- Biểu diễn các số sau sang dạng dấu-độ lớn 8b

a) + 58

b) - 80

c) - 54

d) 11

e) - 13

f) 145

a. Biểu diễn dấu – độ lớn (tiếp) (Sign-magnitude representation)

Miền giá trị của từ mã n bit: $-(2^{n-1}-1)$ đến $2^{n-1}-1$

Tính toán giá trị mã dấu-độ lớn:

- Một số nguyên A, biểu diễn dưới dạng dấu – độ lớn, n-bit: $\boldsymbol{a_{n-1}a_{n-2}\ldots a_0}$
- Nếu A là số dương
  - Bit dấu a_{n-1} có giá trị 0

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

- Nếu A là số âm
  - Bit dấu a_{n-1} có giá trị 1

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

❖ ***Trong đó, a_i là giá trị bit tại vị trí i***

Ví dụ

- Hãy xác định giá trị của các số nguyên có dấu được biểu diễn theo mã dấu-độ lớn:

  P = 0110 0010

  Q = 1101 1011

  Giải:

  $P = 2^6 + 2^5 + 2^1 = 64 + 32 + 2 = + 98$

  $Q = -(2^6 + 2^4 + 2^3 + 2^1 + 2^0) = - 91$

Bài tập áp dụng

Tính toán giá trị của các số biểu diễn dạng dấu- độ lớn 8b sau:

A = 0110 1011

B = 1101 1001

a. Biểu diễn dấu – độ lớn (tiếp) (Sign-magnitude representation)

- Nhược điểm:
  - Thực hiện phép toán cộng, trừ đòi hỏi phải xét cả dấu của các số và độ lớn của chúng
  - Có hai dạng biểu diễn của số 0: gây khó khăn khi thực hiện việc kiểm tra 0 trong một số phép toán

$$+0_{10} = 00000000$$
$$-0_{10} = 10000000 \quad \text{(sign magnitude)}$$

- Do những nhược điểm này, biểu diễn dấu – độ lớn hiếm khi được sử dụng trong việc mã hóa phần số nguyên trong ALU

b. Biểu diễn bù 2

- Trong một từ *n* bit:
  - **Số dương:** bit dấu là 0, *n-1* bit còn lại biểu diễn cho độ lớn giống quy tắc dấu – độ lớn

$$A = \sum_{i=0}^{n-2} 2^i a_i \qquad \text{for } A \geq 0$$

  - **Số 0** được xác định là dương và do đó có bit dấu 0 và độ lớn toàn là 0.
  - **Số âm:** Biểu diễn mã bù 2 của số âm bằng số bù hai của số dương tương ứng.

    B1: Lấy bù 1 của số dương tương ứng (đảo 0 → 1 và 1 → 0)

    B2: Cộng thêm 1

Ví dụ: Biểu diễn bù hai

+18	=	00010010	(twos complement, 8 bits)
−18	=	11101110	(twos complement, 8 bits)

Bài tập áp dụng

- Biểu diễn các số sau sang dạng mã bù 2-8b

a) + 58

b) - 80

c) - 54

d) 11

e) - 13

f) 145

b. Biểu diễn bù 2 (tiếp)

Miền giá trị của từ mã n bit: -2^{n-1} đến $2^{n-1}-1$

Tính toán giá trị mã bù 2:

- Một số nguyên A, biểu diễn dưới dạng bù 2, n-bit: $\boldsymbol{a_{n-1}a_{n-2}\ldots a_0}$
- Nếu A là số dương
  - Bit dấu a_{n-1} có giá trị 0

$$A = \sum_{i=0}^{n-2} 2^i a_i \qquad \text{for } A \geq 0$$

- Nếu A là số âm (A<0)
  - Bit dấu a_{n-1} có giá trị 1

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

❖ ***Trong đó, a_i là giá trị bit tại vị trí i***

Ví dụ

- Hãy xác định giá trị của các số nguyên được biểu diễn theo mã bù hai dưới đây:

  P = 0110 0010

  Q = 1101 1011

  Giải:

  $P = 2^6 + 2^5 + 2^1 = 64+32+2 = +98$

  $Q = -2^7 + 2^6 + 2^4 + 2^3 + 2^1 + 2^0$

  $= -128+64+16+8+2+1 = -37$

Bài tập áp dụng

Tính toán giá trị của các số biểu diễn dạng mã bù 2 8b sau:

A = 0110 1011

B = 1101 1001

Một số đặc điểm của biểu diễn bù 2

Miền giá trị (n bit)	-2^{n-1} đến $2^{n-1}-1$
Biểu diễn số 0	1 cách
Biểu diễn số âm	Lấy bù của số dương tương ứng sau đó cộng thêm 1
Mở rộng chiều dài chuỗi bit	Điền giá trị dấu vào bên trái
Luật tràn	Khi cộng hai số cùng dấu, nếu kết quả có dấu ngược lại → tràn
Luật trừ	Khi trừ A cho B, lấy bù 2 của B sau đó cộng với A

Bảng 10.2
Biểu diễn số nguyên 4-Bit

Biểu diễn thập phân	Biểu diễn dấu – độ lớn	Biểu diễn bù 2
+8	—	—
+7	0111	0111
+6	0110	0110
+5	0101	0101
+4	0100	0100
+3	0011	0011
+2	0010	0010
+1	0001	0001
+0	0000	0000
–0	1000	—
–1	1001	1111
–2	1010	1110
–3	1011	1101
–4	1100	1100
–5	1101	1011
–6	1110	1010
–7	1111	1001
–8	—	1000

Mở rộng phạm vi biểu diễn

- Trong một số trường hợp, ta muốn biểu diễn một số n-bit sang dạng biểu diễn m-bit ($m > n$): mở rộng phạm vi biểu diễn
- **Trong biểu diễn dấu – độ lớn:**
  - Di chuyển bit dấu tới vị trí mới ngoài cùng bên trái và điền (m-n-) bit 0 vào sau bit dấu

+18	=	00010010	(sign magnitude, 8 bits)
+18	=	0000000000010010	(sign magnitude, 16 bits)
−18	=	10010010	(sign magnitude, 8 bits)
−18	=	1000000000010010	(sign magnitude, 16 bits)

- **Biểu diễn mã bù 2:**
  - Quy tắc: di chuyển bit dấu sang vị trí ngoài cùng bên trái và điền vào bằng bản sao bit dấu
  - Đối với số dương, điền 0 vào, và số âm thì điền vào số 1
  - Đây được gọi là phần mở rộng dấu

Ví dụ: Mở rộng phạm vi với mã bù 2

+18	=	00010010	(twos complement, 8 bits)
+18	=	0000000000010010	(twos complement, 16 bits)

−18	=	11101110	(twos complement, 8 bits)
−18	=	1111111111101110	(twos complement, 16 bits)

3. Phép toán số học với số nguyên

a. Phép đảo dấu
b. Phép cộng
c. Phép trừ
d. Phép nhân
e. Phép chia

a. Phép đảo dấu

- **Biểu diễn dấu – độ lớn:** Phép đảo dấu được thực hiện bằng cách đảo bit dấu
- **Biểu diễn bù 2:** Phép đảo dấu được thực hiện bằng phép toán bù 2
  - Đảo từng bit (kể cả bit dấu)
  - Cộng 1

$$
\begin{array}{rcll}
+18 & = & 00010010 & \text{(twos complement)} \\
\text{bitwise complement} & = & 11101101 & \\
 & + & 1 & \\
 & & 11101110 = -18 &
\end{array}
$$

$$
\begin{array}{rcll}
-18 & = & 11101110 & \text{(twos complement)} \\
\text{bitwise complement} & = & 00010001 & \\
 & + & 1 & \\
 & & 00010010 = +18 &
\end{array}
$$

Nhận xét: Đảo của đảo một số là chính nó

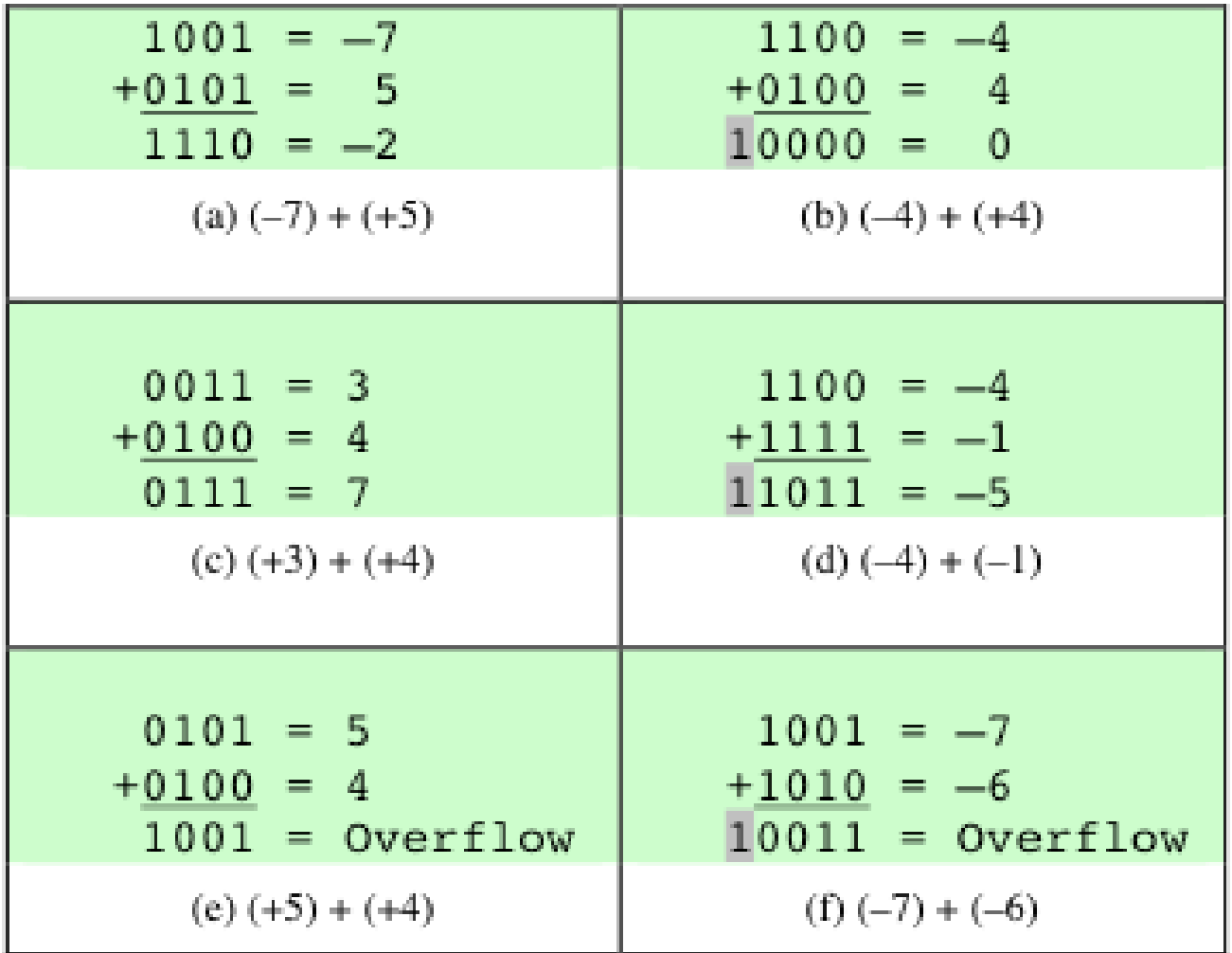
Số bù 2: xét hai trường hợp đặc biệt

$$
\begin{aligned}
0 &= 00000000 \text{ (bù hai)} \\
\text{Lấy bù} &= \begin{array}{r} 11111111 \\ +\quad\quad 1 \\ \hline 100000000 \end{array} = 0
\end{aligned}
$$

$$
\begin{aligned}
-128 &= 10000000 \text{ (bù hai)} \\
\text{Lấy bù} &= \begin{array}{r} 01111111 \\ +\quad\quad 1 \\ \hline 10000000 \end{array} = -128
\end{aligned}
$$

b. Phép cộng (bù 2-4b)

- Phép cộng được thực hiện bình thường như cộng hai số nhị phân
- Trong một số trường hợp, xuất hiện thêm 1b (bit bôi đen) → bỏ qua các bit này



- **Tràn ô nhớ**: khi kết quả của một phép toán quá lớn vượt qua phạm vi biểu diễn của ô nhớ
- Cộng ở dạng bù 2: **Tràn** ô nhớ xảy ra nếu hai số cùng dấu cộng với nhau mà kết quả thu được lại là một số có dấu ngược với dấu của hai số đó
- Khi phát hiện tràn, ALU cần phải báo hiệu việc này để CPU không sử dụng kết quả

c. Phép trừ

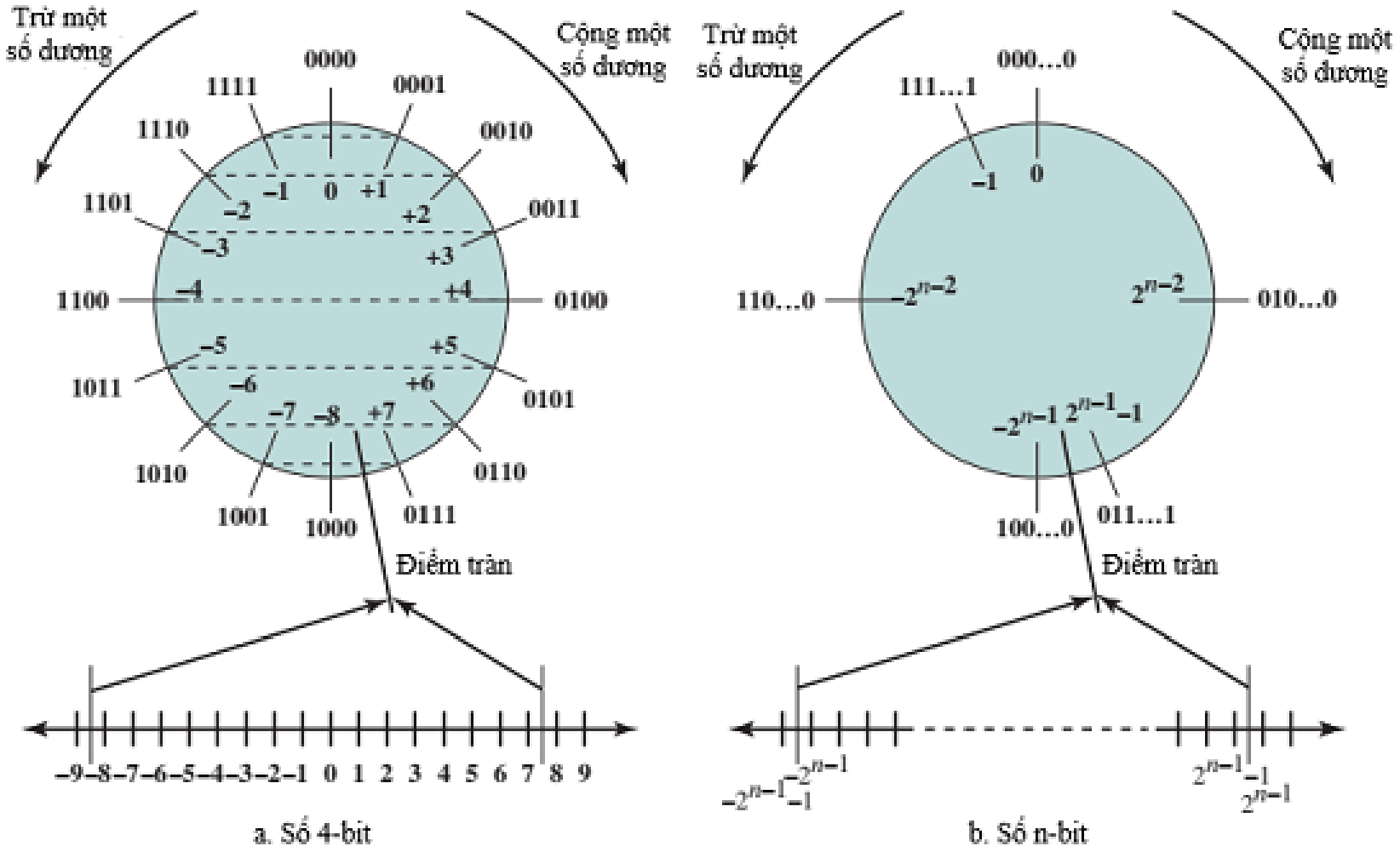
QUI TẮC TRỪ: Để thực hiện phép trừ giữa hai số, ta lấy bù hai (đảo dấu) của số trừ rồi cộng với số bị trừ.

c. Phép trừ

0010 = 2 +1001 = −7 1011 = −5 (a) M = 2 = 0010 S = 7 = 0111 −S = 1001	0101 = 5 +1110 = −2 10011 = 3 (b) M = 5 = 0101 S = 2 = 0010 −S = 1110
1011 = −5 +1110 = −2 11001 = −7 (c) M = −5 = 1011 S = 2 = 0010 −S = 1110	0101 = 5 +0010 = 2 0111 = 7 (d) M = 5 = 0101 S = −2 = 1110 −S = 0010
0111 = 7 +0111 = 7 1110 = **Tràn** (e) M = 7 = 0111 S = −7 = 1001 −S = 0111	1010 = −6 +1100 = −4 10110 = **Tràn** (f) M = −6 = 1010 S = 4 = 0100 −S = 1100

Hình 9.4 Phép trừ hai số biểu diễn Bù hai $(M - S)$

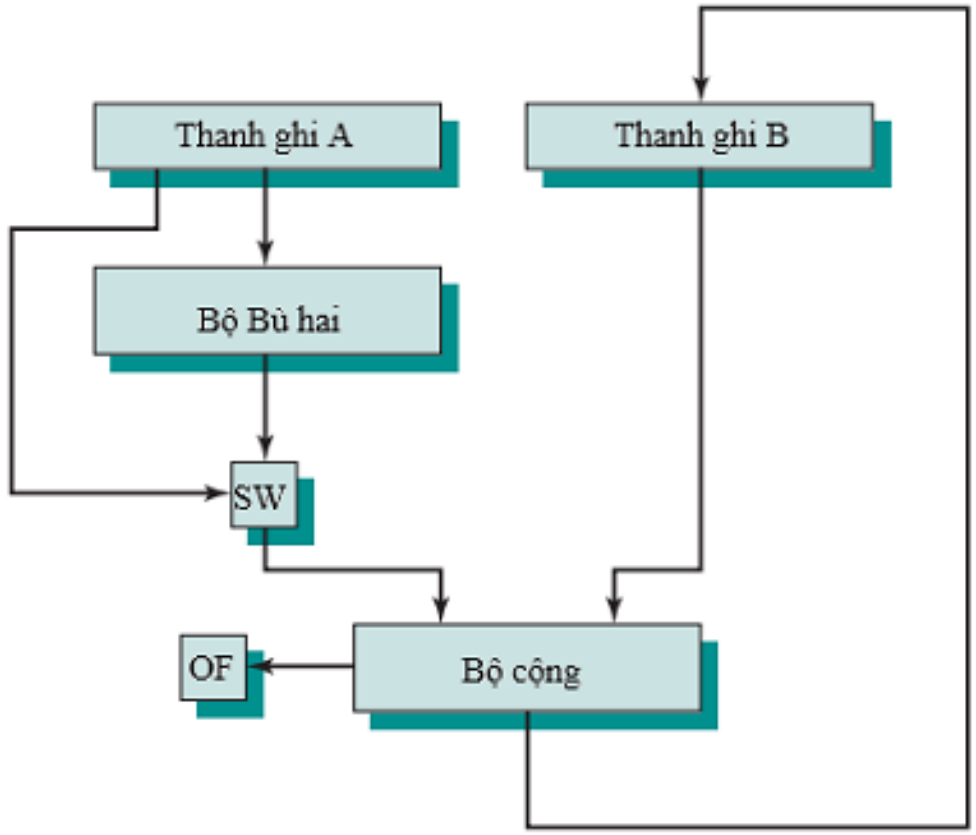
Mô tả hình học của số nguyên bù 2



Hình 9.5 Biểu diễn hình học các số nguyên Bù hai

Phần cứng thực hiện cộng và trừ

- Thanh ghi A, B (A, B Register): lưu trữ hai số
- Với phép trừ: thanh ghi B lưu trữ số trừ
- Complementer: bộ lấy bù 2
- SW (switch): lựa chọn cộng hoặc trừ
- Bộ cộng (Adder): thực hiện phép toán và đưa ra cờ tràn (nếu có)
- Cờ tràn (OF-overflow bit):
  - 0: không tràn
  - 1: tràn
- Kết quả có thể được lưu trữ ở thanh ghi thứ 3 hoặc một trong 2 thanh ghi A, B



SW: Bộ chuyển mạch (cho phép thực hiện phép cộng hoặc phép trừ)
OF: Cờ tràn

Hình 9.6 Sơ đồ khối phần cứng Bộ cộng và bộ trừ

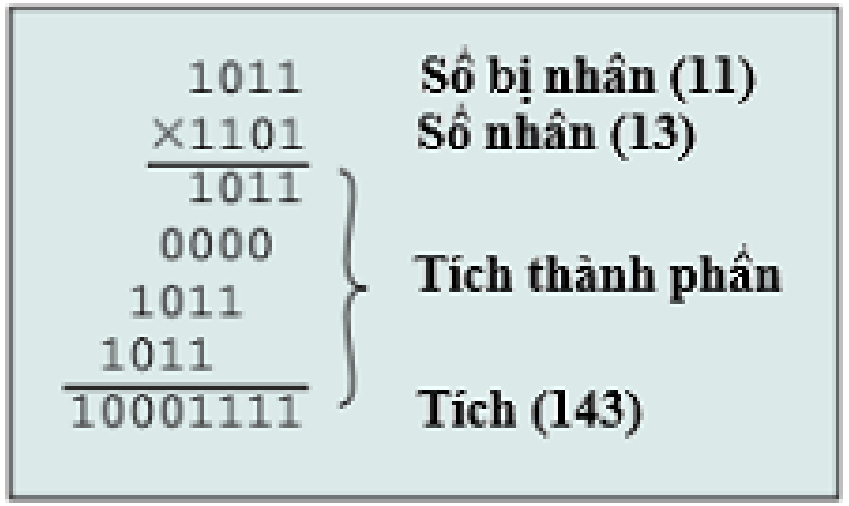
d. Phép nhân

- So với phép cộng và phép trừ, phép nhân phức tạp hơn
- Nhiều thuật toán tính toán phép nhân khác nhau đã được sử dụng trong các máy tính khác nhau
- Trong phần này:
  i. Phép nhân giữa hai số nguyên không dấu
  ii. Phép nhân hai số biểu diễn bù 2

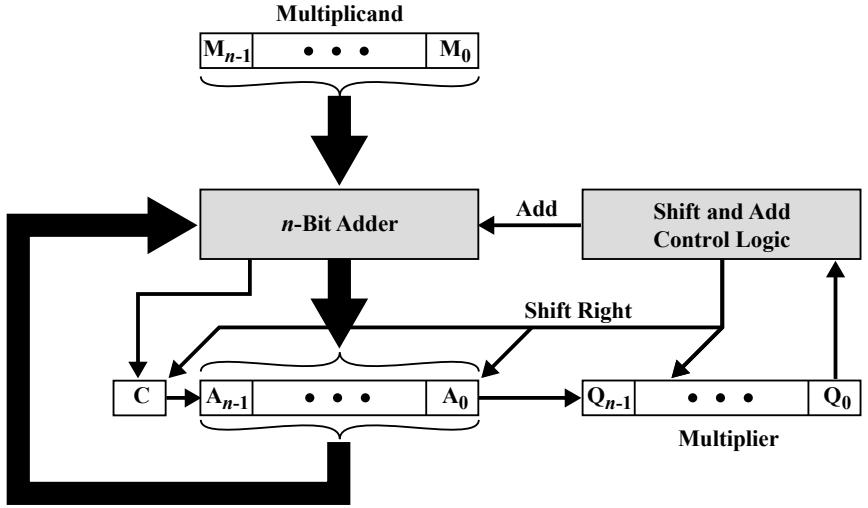
i. Phép nhân giữa hai số nguyên không dấu

Các bước bằng tay:

- Tính các tích thành phần
- Nếu số nhân là bit 0, tích thành phần bằng 0. Nếu số nhân là bit 1, tích thành phần là *số bị nhân (multiplicand)*



- Tính tổng các tích thành phần (mỗi tích dịch trái đơn vị so với tích trước đó)
- Tích của hai số *n*-bit là một số có kích thước *2n*-bit



(a) Block Diagram

C	A	Q	M		
0	0000	1101	1011	Initial Values	
0	1011	1101	1011	Add	First Cycle
0	0101	1110	1011	Shift	
0	0010	1111	1011	Shift	Second Cycle
0	1101	1111	1011	Add	Third Cycle
0	0110	1111	1011	Shift	
1	0001	1111	1011	Add	Fourth Cycle
0	1000	1111	1011	Shift	

1011	Multiplicand (11)
×1101	Multiplier (13)
1011	Partial products
0000	
1011	
1011	
10001111	Product (143)

(b) Example from Figure 9.7 (product in A, Q)

Figure 10.8 Hardware Implementation of Unsigned Binary Multiplication

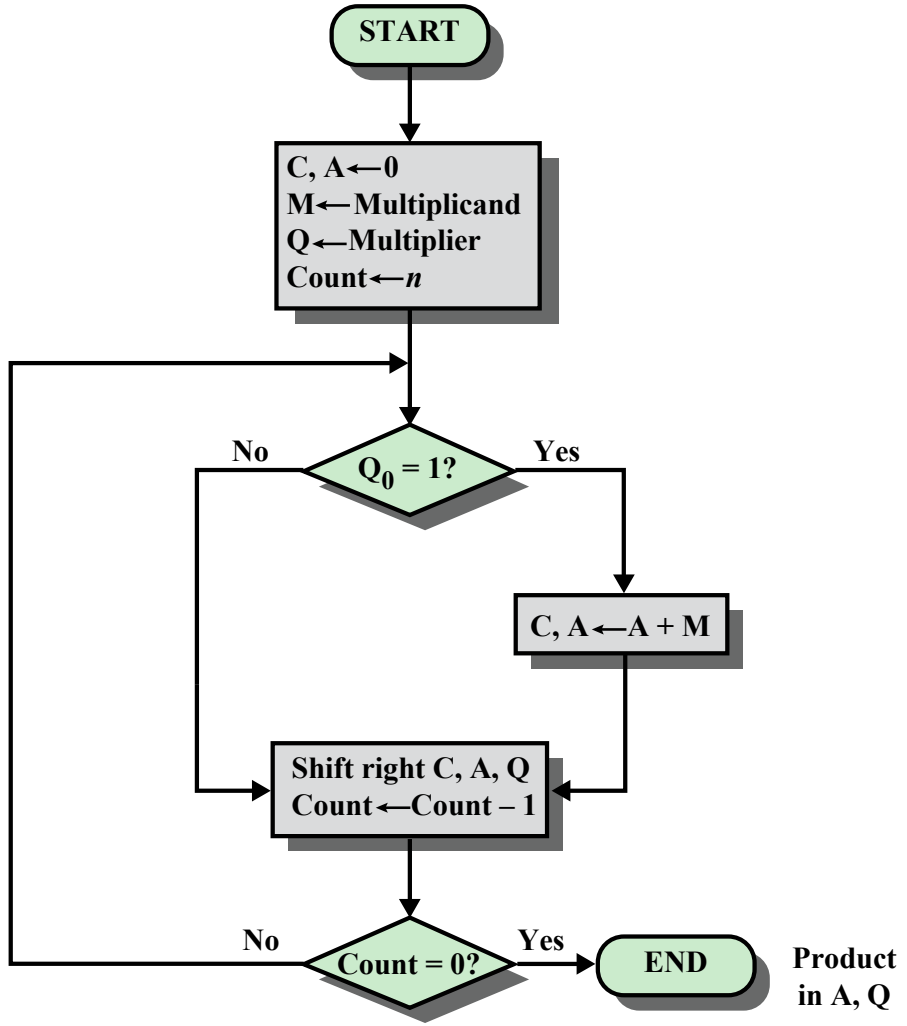


Figure 10.9 Flowchart for Unsigned Binary Multiplication

ii. Phép nhân bù 2

- Nếu hai số là số nguyên dương: nguyên tắc nhân giống số nguyên không dấu
- Nếu xuất hiện thừa số là số nguyên âm: do có sự xuất hiện của bit dấu nên nguyên tắc trên không còn đúng nữa

1001 (9) ´0011 (3) 00001001 1001 ´ 2^0 00010010 1001 ´ 2^1 00011011 (27)	1001 (−7) ´0011 (3) 11111001 (−7) ´ 2^0 = (−7) 11110010 (−7) ´ 2^1 = (−14) 11101011 (−21)
(a) Unsigned integers	(b) Twos complement integers

Figure 10.11 Comparison of Multiplication of Unsigned and Twos Complement Integers

Giải pháp 1

Sử dụng thuật toán nhân hai số không dấu:

- Bước 1: Chuyển đổi số nhân và số bị nhân thành số dương tương ứng.
- Bước 2: Nhân 2 số bằng thuật giải nhân số nguyên không dấu → được tích 2 số dương.
- Bước 3: Hiệu chỉnh dấu của tích:
  - Nếu 2 thừa số ban đầu cùng dấu thì tích nhận được ở bước 2 là kết quả cần tính.
  - Nếu 2 thừa số ban đầu khác dấu nhau thì kết quả là số bù 2 của tích nhận được ở bước 2.

Giải pháp 2: thuật toán Booth

- Tốc độ tính toán nhanh hơn do số lượng phép toán ít hơn
- Thuật toán chung cho cả số nguyên dương và âm

Thuật toán Booth

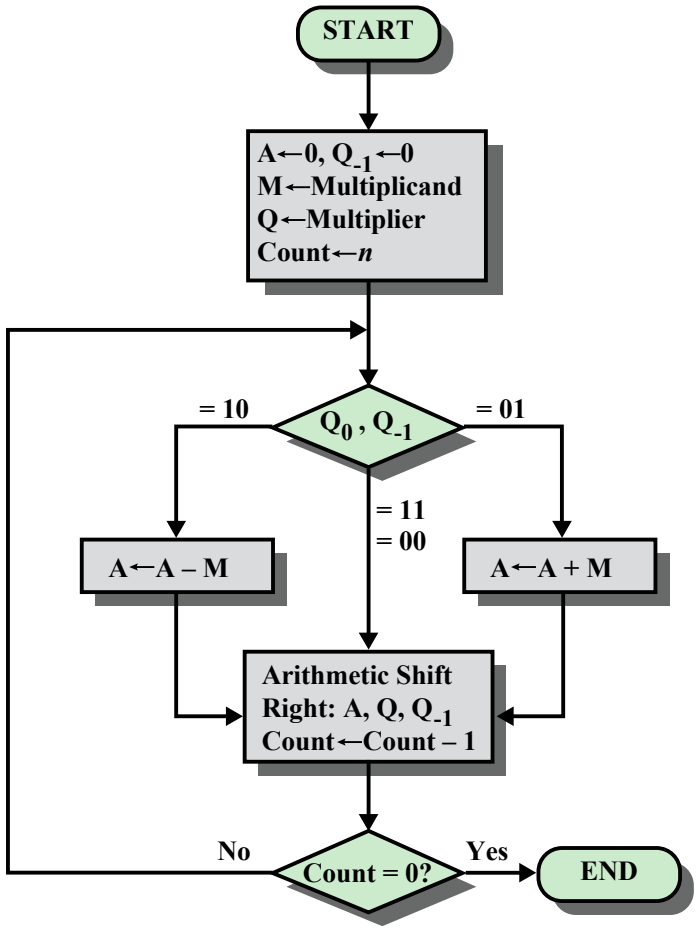


Figure 10.12 Booth's Algorithm for Twos Complement Multiplication

A	Q	Q_{-1}	M		
0000	0011	0	0111	Initial Values	
1001	0011	0	0111	A ← A - M	First Cycle
1100	1001	1	0111	Shift	First Cycle
1110	0100	1	0111	Shift	Second Cycle
0101	0100	1	0111	A ← A + M	Third Cycle
0010	1010	0	0111	Shift	Third Cycle
0001	0101	0	0111	Shift	Fourth Cycle

Figure 10.13 Example of Booth's Algorithm (7× 3)

e. Phép chia

Phức tạp hơn phép nhân nhưng cũng sử dụng chung nguyên lý

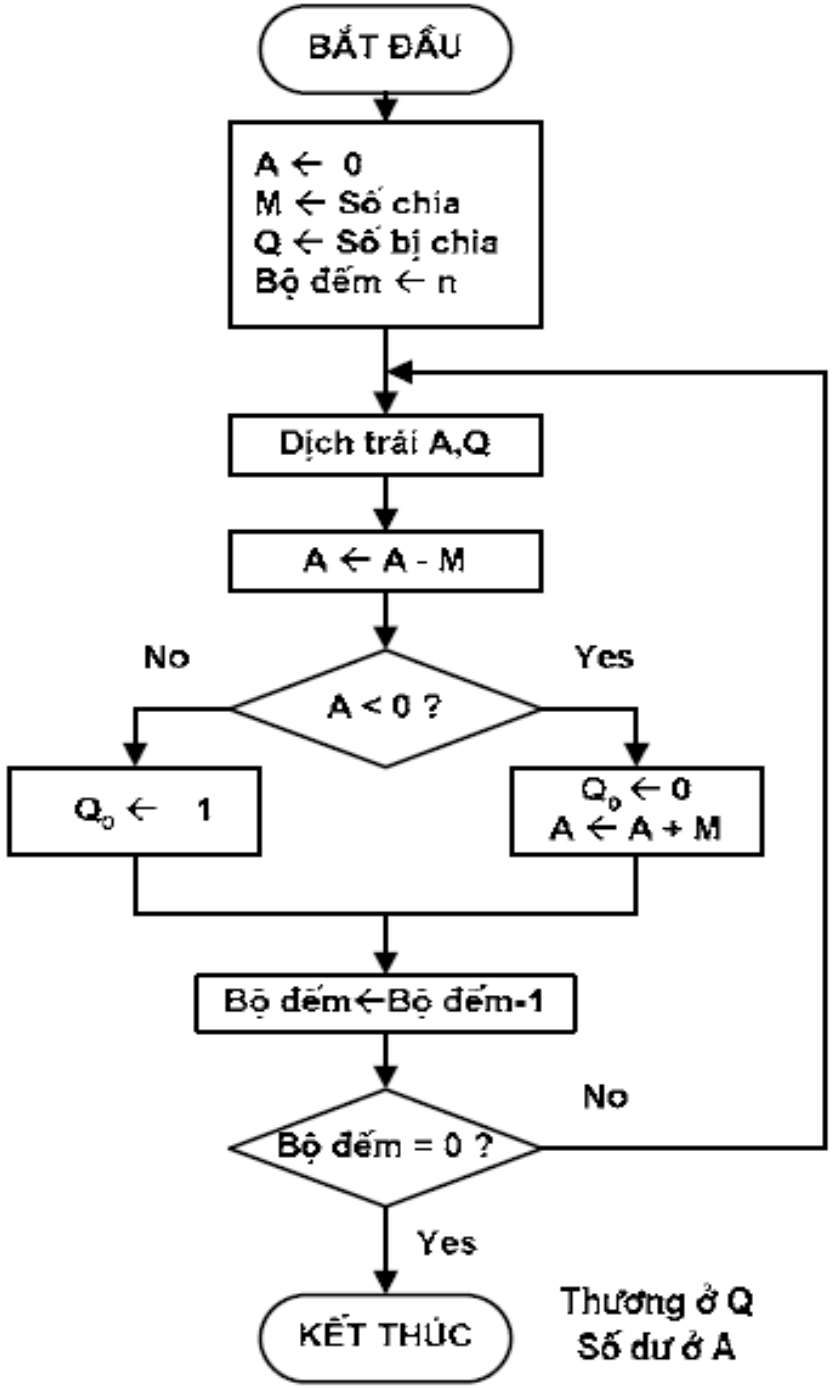
Bằng tay

```
10010011 |1011        Số chia
 1011    |00001101    Thương
 ----
  1110
  1011
  ----
    1111
    1011
    ----
     100             Số dư
```

Thuật toán thực hiện phép chia hai số nguyên không dấu trong máy tính

- Số chia đặt trong thanh ghi M, số bị chia trong thanh ghi Q, A=0, Bộ đếm = n
- Tại mỗi bước:
  - A và Q dịch trái 1 đơn vị
  - Thực hiện $A - M$, nếu
    - $A - M \geq 0$ thì $Q_0 = 1$
    - $A - M < 0$ thì $Q_0 = 0, \mathrm{A} = A + M$
  - Giảm bộ đếm và quay trở lại thực hiện vòng lặp đến khi Bộ đếm = 0

Lưu đồ thuật toán phép chia hai số nguyên không dấu



Chia số nguyên có dấu

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

Số bị chia	Số chia	Thương	Số dư
+	+	giữ nguyên	giữ nguyên
+	-	đảo dấu	giữ nguyên
-	+	đảo dấu	đảo dấu
-	-	giữ nguyên	đảo dấu

Phép chia số bù 2

- Tương tự như phép nhân, do có bit dấu nên phải có thuật toán khác:
- Giả sử số chia V và số bị chia D dương và $|V| < |D|$
  - Nếu $|V| = |D|$: thương = 1, dư = 0.
  - Nếu $|V| > |D|$: thương = 0, dư = D
- Thuật toán như sau:
  - B1: Ghi số bù 2 của V vào thanh ghi M (M chứa số âm của V), ghi D vào thanh ghi A, Q, bộ đếm = n
  - B2: Dịch A,Q sang trái 1 đơn vị
  - B3: Tính A+M →A
  - B4: Kiểm tra:
    - Nếu A dương (bit dấu = 0), $Q_0 = 1$
    - Nếu A âm (bit dấu =1), $Q_0 = 0$, khôi phục A lại giá trị trước đó
  - B5: Giảm bộ đếm đi 1 đơn vị
  - Lặp lại các bước từ 2 đến 5 cho đến khi bộ đếm = 0
- Với các trường hợp V, D không dương, hiệu chỉnh kết quả dựa theo bảng ở trên

Ví dụ phép chia bù 2

A	Q	
0000	0111	Initial value
0000	1110	Shift
1101		Use twos complement of 0011 for subtraction
1101		Subtract
0000	1110	Restore, set Q_0 = 0
0001	1100	Shift
1101		
1110		Subtract
0001	1100	Restore, set Q_0 = 0
0011	1000	Shift
1101		
0000	1001	Subtract, set Q_0 = 1
0001	0010	Shift
1101		
1110		Subtract
0001	0010	Restore, set Q_0 = 0

Figure 10.17 Example of Restoring Twos Complement Division (7/3)

4. Biểu diễn số dấu chấm động

a. Nguyên lý

- Quy ước: "dấu chấm" (point) được hiểu là kí hiệu ngăn cách giữa phần nguyên và phần lẻ của một số thực.
- Có 2 cách biểu diễn số thực trong máy tính:
  - Số dấu chấm tĩnh (fixed-point number):
    - Dấu chấm là cố định (số bit dành cho phần nguyên và phần lẻ là cố định)
    - Hạn chế: không thể biểu diễn số rất lớn hoặc số thập phân rất nhỏ. Phần thập phân trong thương của một phép chia hai số lớn có thể bị mất
    - Dùng trong các bộ vi xử lý hay vi điều khiển thế hệ cũ.
  - Số dấu chấm động (floating-point number):
    - Dấu chấm không cố định
    - Dùng trong các bộ vi xử lý hiện nay, có độ chính xác cao hơn.

Biểu diễn số nhị phân dấu chấm động

- Một số nhị phân có thể được biểu diễn dưới dạng như sau:

$$\pm S \times B^{\pm E}$$

- Trong đó:
  - S: phần định trị
  - B: cơ số (hệ nhị phân: B = 2)
  - E: số mũ
- Nếu B là một số định sẵn, ta chỉ cần lưu trữ S và E
- Vậy một số nhị phân có thể được lưu trữ trong máy tính với 3 trường sau:
  - Dấu
  - Định trị S
  - Số mũ E

b. Chuẩn IEEE 754

- Hiệp hội IEEE đã chuẩn hóa cho việc biểu diễn số dấu phẩy động nhị phân trong máy tính
- Mục đích:
  - Hỗ trợ tính di động của chương trình từ bộ xử lý này sang bộ xử lý khác
  - Khuyến khích phát triển các chương trình định hướng số học tinh vi hơn
- Chuẩn được công nhận rộng rãi và được sử dụng trên hầu hết các bộ VXL và bộ tính toán số học hiện đại
- IEEE 754-2008 quy định các định dạng biểu diễn dấu phẩy động nhị phân và thập phân
- Trong phần này chỉ đề cập đến dạng biểu diễn dấu phẩy động nhị phân

IEEE 754-2008

- IEEE 754-2008 định nghĩa 3 định dạng dấu chấm động sau:
- **Định dạng số học**
  - Được sử dụng để biểu diễn các toán hạng hoặc kết quả phép toán dưới dạng dấu chấm động.
- **Định dạng cơ bản:** quy định 5 dạng biểu diễn dấu chấm động:
  - **Ba cho số nhị phân: chiều dài 32b, 64b và 128b**
  - Hai cho thập phân: chiều dài 64b và 128b
- **Định dạng chuyển đổi**
  - Đưa ra dạng mã hoá nhị phân độ dài cố định cho phép trao đổi dữ liệu giữa các nền tảng khác nhau và có thể được sử dụng để lưu trữ.

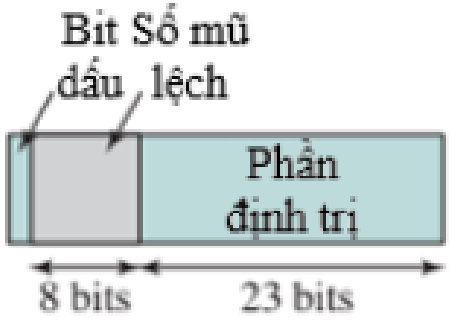
Biểu diễn dấu chấm động cho số nhị phân theo định dạng IEEE 754

- Biểu diễn số X:

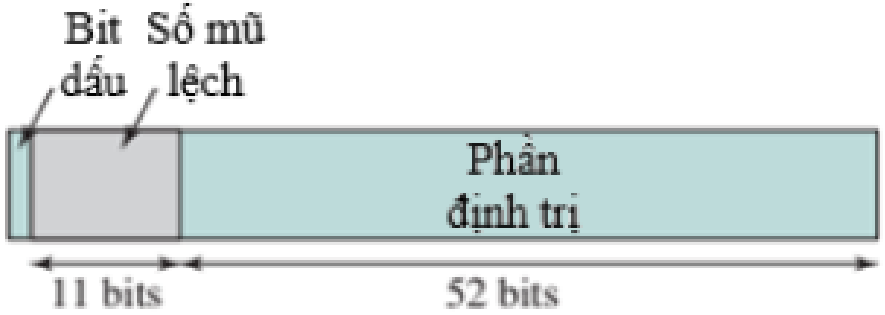
$$X = (-1)^s \times 1.m \times 2^E = (-1)^s \times 1.m \times 2^{e-B}$$

- s là bit dấu:
  - Số dương: s = 0
  - Số âm: s = 1
- e là số mũ được dịch chuyển đi B
  - B (độ dịch) = $2^{k-1} - 1$
  - k là độ dài bit của trường số mũ e
  - ví dụ: độ dài của trường e là 8 bit $\rightarrow$ B = $2^{8-1} - 1 = 127$

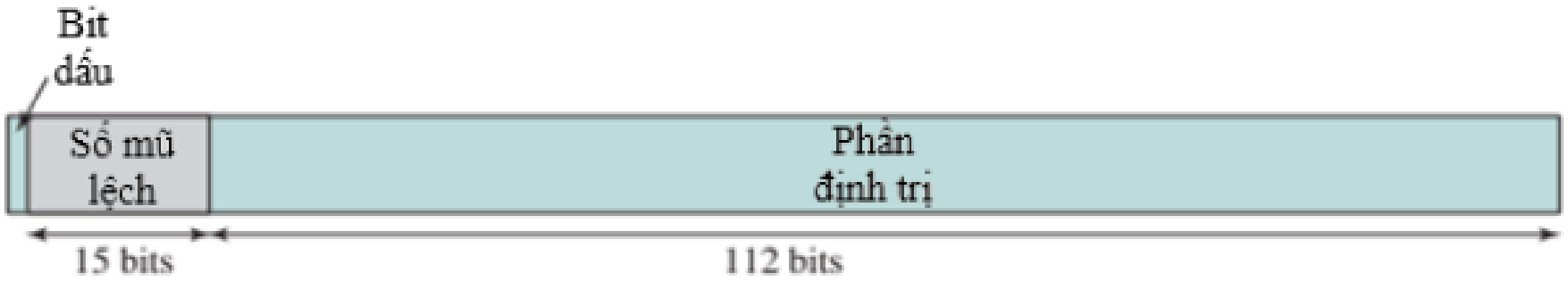
Định dạng IEEE 754



a. Định dạng nhị phân 32



b. Định dạng nhị phân 64



c. Định dạng nhị phân 128

Hình 9.21 Các định dạng IEEE 754

- Cơ số R = 2
- Có các dạng cơ bản:
  - *Số có độ chính xác đơn* 32-bit: 1b dấu, 8b mũ, 23b định trị
  - *Số có độ chính xác kép* 64-bit: 1b dấu, 11b mũ, 52b định trị
  - *Số có độ chính xác mở rộng* 128-bit: 1b dấu, 15b mũ, 112b định trị

Thông số	Định dạng		
	Nhị phân 32	Nhị phân 64	Nhị phân 128
Kích thước (bit)	32	64	128
Số bit trường mũ	8	11	15
Độ lệch	127	1023	16383
Số mũ tối đa	127	1023	16383
Số mũ tối thiểu	−126	−1022	−16382
Miền giá trị xấp xỉ (hệ 10)	$10^{-38}, 10^{+38}$	$10^{-308}, 10^{+308}$	$10^{-4932}, 10^{+4932}$
Số bit trường định trị theo sau	23	52	112
Số lượng số mũ	254	2046	32766
Số lượng các số phân số	2^{23}	2^{52}	2^{112}
Số lượng các giá trị	1.98×3^{31}	1.99×2^{63}	1.99×2^{127}
Số dương nhỏ nhất	2^{-126}	2^{-1022}	2^{-16362}
Số dương lớn nhất	$2^{128} - 2^{104}$	$2^{1024} - 2^{971}$	$2^{16384} - 2^{16271}$

Bảng 10.3 Thông số định dạng chính trong chuẩn IEEE 754

*không bao gồm bit ngầm định và bit dấu

IEEE 754: Một số quy ước đặc biệt

- Nếu tất cả các bit của phần số mũ đều bằng 0, các bit của phần định trị đều bằng 0, thì X = ± 0
  - Ví dụ: định dạng 32b:

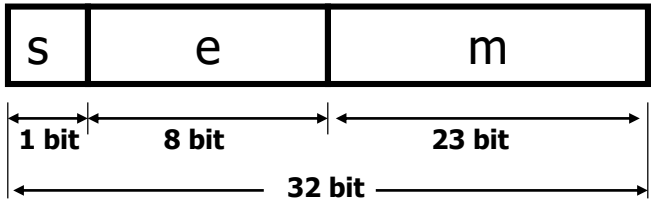
$$+0_{10} = 0\ 00000000\ 00000000000000000000000$$
$$-0_{10} = 1\ 00000000\ 00000000000000000000000$$

- Nếu tất cả các bit của phần mũ đều bằng 1, các bit của phần định trị đều bằng 0, thì X = ± ∞
  - Ví dụ: định dạng 32b:

$$+\infty = 0\ 11111111\ 00000000000000000000000$$
$$-\infty = 1\ 11111111\ 00000000000000000000000$$

- Nếu tất cả các bit của phần mũ đều bằng 1, phần định trị có ít nhất một bit bằng 1, thì X biểu diễn một giá trị quy ước NaN (not a number, không biểu diễn cho số nào)

Dạng chính xác đơn 32 bit



- S là bit dấu:
  - S = 0 $\rightarrow$ số dương
  - S = 1 $\rightarrow$ số âm
- m (23 bit) là phần lẻ của phần định trị M
- e (8 bit) là số mũ được dịch chuyển đi B
  - $B = 2^{8-1} - 1 = 127$
- Công thức xác định giá trị của số thực X:

$$X = (-1)^s \times 1.m \times 2^E = (-1)^s \times 1.m \times 2^{e-127} \quad (E = e-127)$$

Ví dụ 1

Biểu diễn số thực X= 83.75 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

Ví dụ 2

Biểu diễn số thực X= -0,2 về dạng số dấu phẩy động IEEE754 32-bit

Giải:

- $X = -0,2_{(10)} = -\ 0.00110011...0011..._{(2)} =$
  $= -\ 1.100110011..0011... \times 2^{-3}$
- Ta có:
  - S = 1 vì đây là số âm
  - $E = e-127 = -3 \rightarrow e = 127 - 3 = 124_{(10)} = 0111\ 1100_{(2)}$
- Vậy:

  X = 1011 1110 0100 1100 1100 1100 1100 1100

Ví dụ 3

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

1100 0001 0101 0110 0000 0000 0000 0000

Giải:

- S = 1 → số âm
- e = $1000\ 0010_2$ = 130 → E = 130-127=3

Vậy

X = -1.10101100 * 2^3 = -1101.011 = -13.375

Bài tập

1) Biểu diễn số thực X = 0.375 về dạng số dấu chấm động theo chuẩn IEEE 754 dạng 32 bit.
2) Một số thực X có dạng biểu diễn nhị phân theo chuẩn IEEE 754 dạng 32 bit như sau: 1100 0001 0101 0110 0000 0000 0000 0000. Xác định giá trị thập phân của số thực đó.
3) Xác định giá trị thập phân của số thực X có dạng biểu diễn theo chuẩn IEEE 754 dạng 32 bit như sau:

   0011 1111 1000 0000 0000 0000 0000 0000

5 Các phép toán với số dấu chấm động

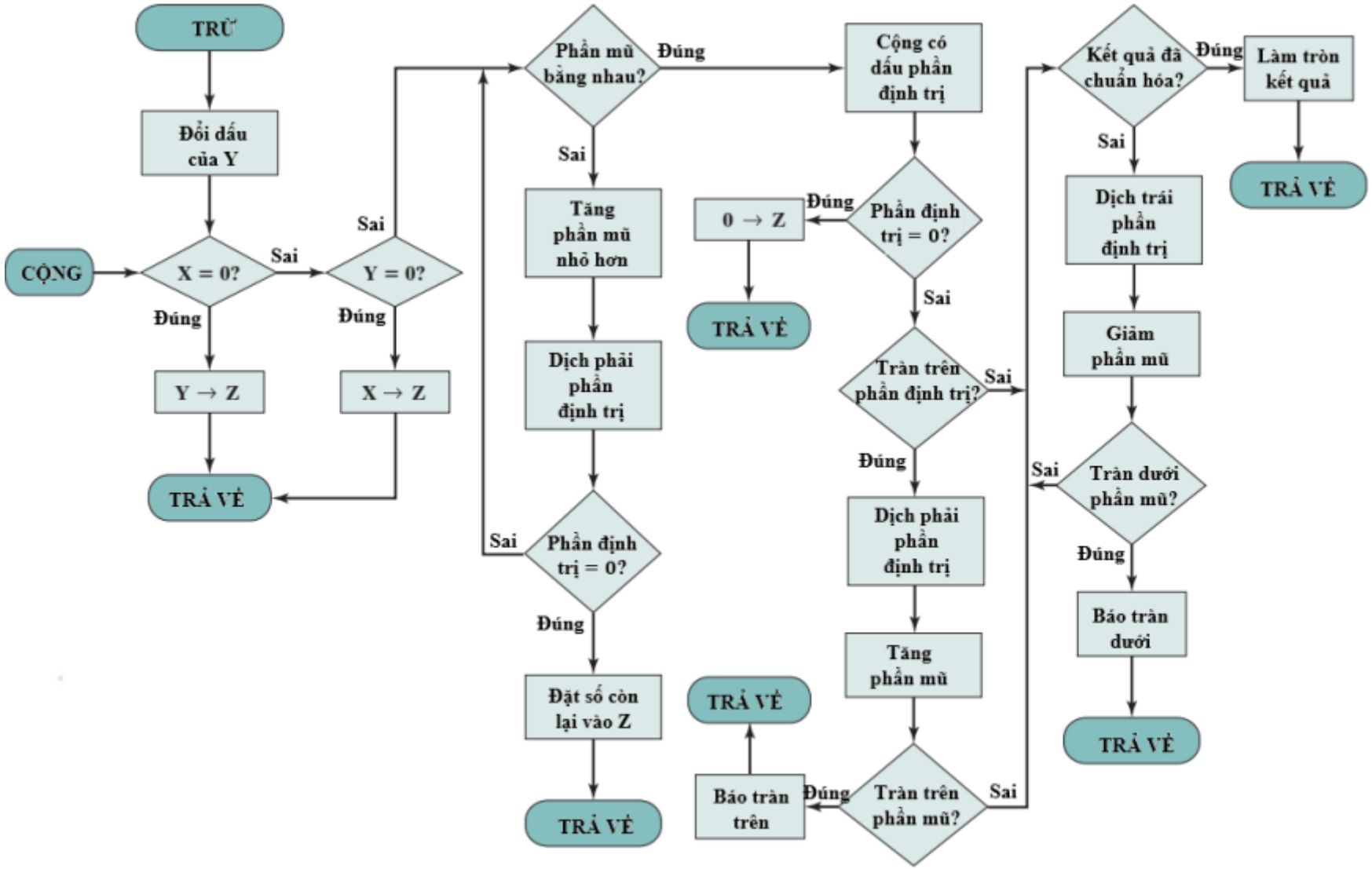
a. Phép toán cộng và trừ

Floating Point Numbers	Arithmetic Operations
$X = X_S \times B^{X_E}$ $Y = Y_S \times B^{Y_E}$	$X + Y = \left(X_S \times B^{X_E - Y_E} + Y_S\right) \times B^{Y_E}$ $X - Y = \left(X_S \times B^{X_E - Y_E} - Y_S\right) \times B^{Y_E}$ } $X_E \leq Y_E$ $X \times Y = \left(X_S \times Y_S\right) \times B^{X_E + Y_E}$ $\frac{X}{Y} = \left(\frac{X_S}{Y_S}\right) \times B^{X_E - Y_E}$

$X = 0.3 \times 10^2 = 30$
$Y = 0.2 \times 10^3 = 200$

$X + Y = (0.3 \times 10^{2-3} + 0.2) \times 10^3 = 0.23 \times 10^3 = 230$
$X - Y = (0.3 \times 10^{2-3} - 0.2) \times 10^3 = (-0.17) \times 10^3 = -170$
$X \times Y = (0.3 \times 0.2) \times 10^{2+3} = 0.06 \times 10^5 = 6000$
$X \div Y = (0.3 \div 0.2) \times 10^{2-3} = 1.5 \times 10^{-1} = 0.15$

Cộng và trừ số dấu chấm động



Hình 9.22 Sơ đồ thuật toán phép cộng và phép trừ ($X \pm Y \rightarrow Z$)

Bốn bước cơ bản của thuật toán cộng và trừ

- Kiểm tra các số hạng có bằng 0 hay không
  - Nếu có thì gán kết quả dựa trên số còn lại.
- Hiệu chỉnh phần định trị
  - Sao cho 2 số có phần mũ giống nhau: tăng số mũ nhỏ và dịch phải phần định trị tương ứng (dịch phải để hạn chế sai số nếu có).
  - VD: $1.01 * 2^3 + 1.11 = 1.01 * 2^3 + 0.00111 * 2^3$
- Cộng hoặc trừ phần định trị
  - Nếu tràn thì dịch phải và tăng số mũ, nếu bị tràn số mũ thì báo lỗi tràn số.
- Chuẩn hóa kết quả
  - Dịch trái phần định trị để bit trái nhất (bit MSB) khác 0.
  - Tương ứng với việc giảm số mũ nên có thể dẫn đến hiện tượng tràn dưới số mũ.

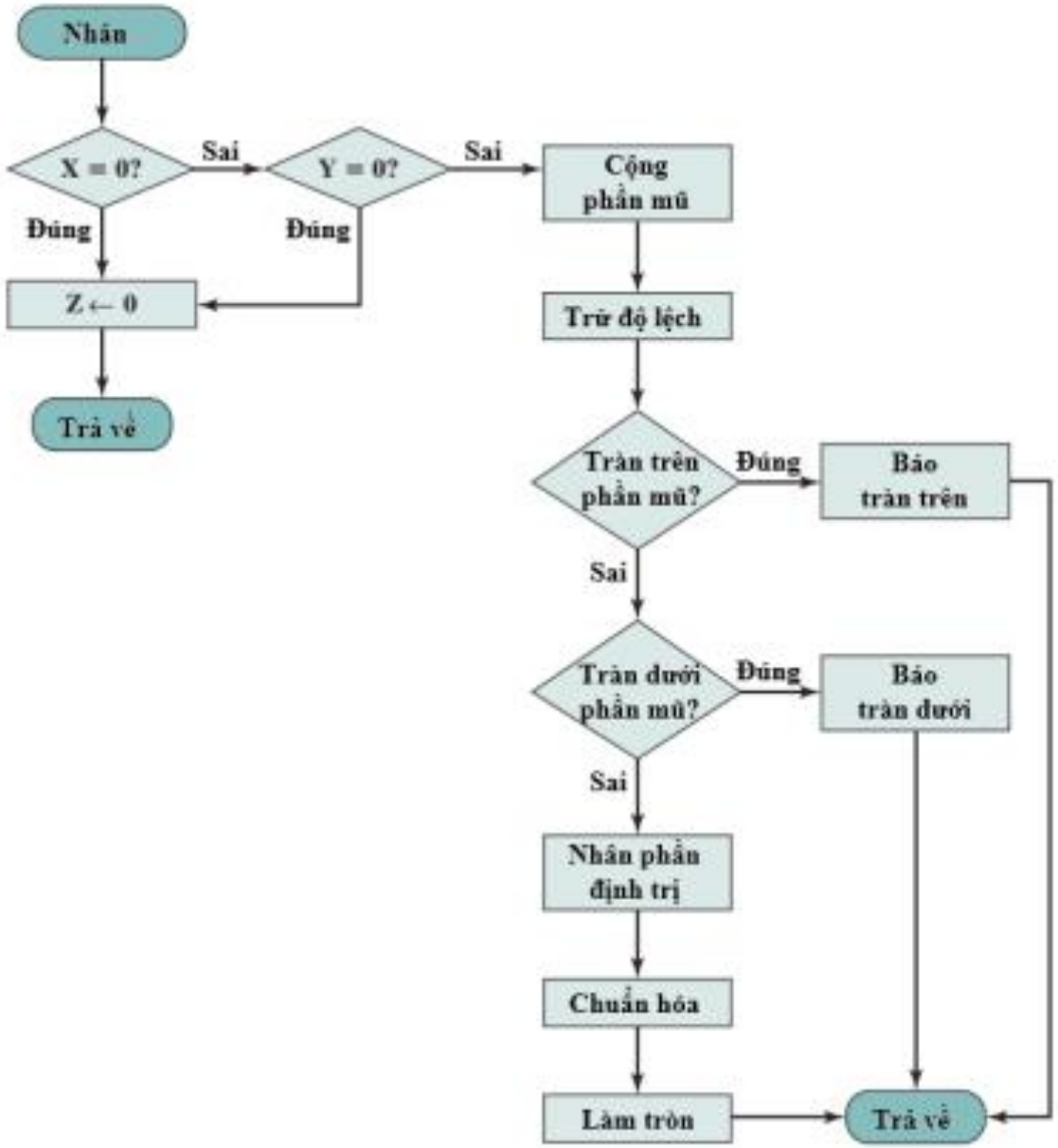
Các khả năng tràn số

Phép toán cộng hoặc trừ có thể gây ra một số khả năng tràn như sau:

- 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.
- 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.

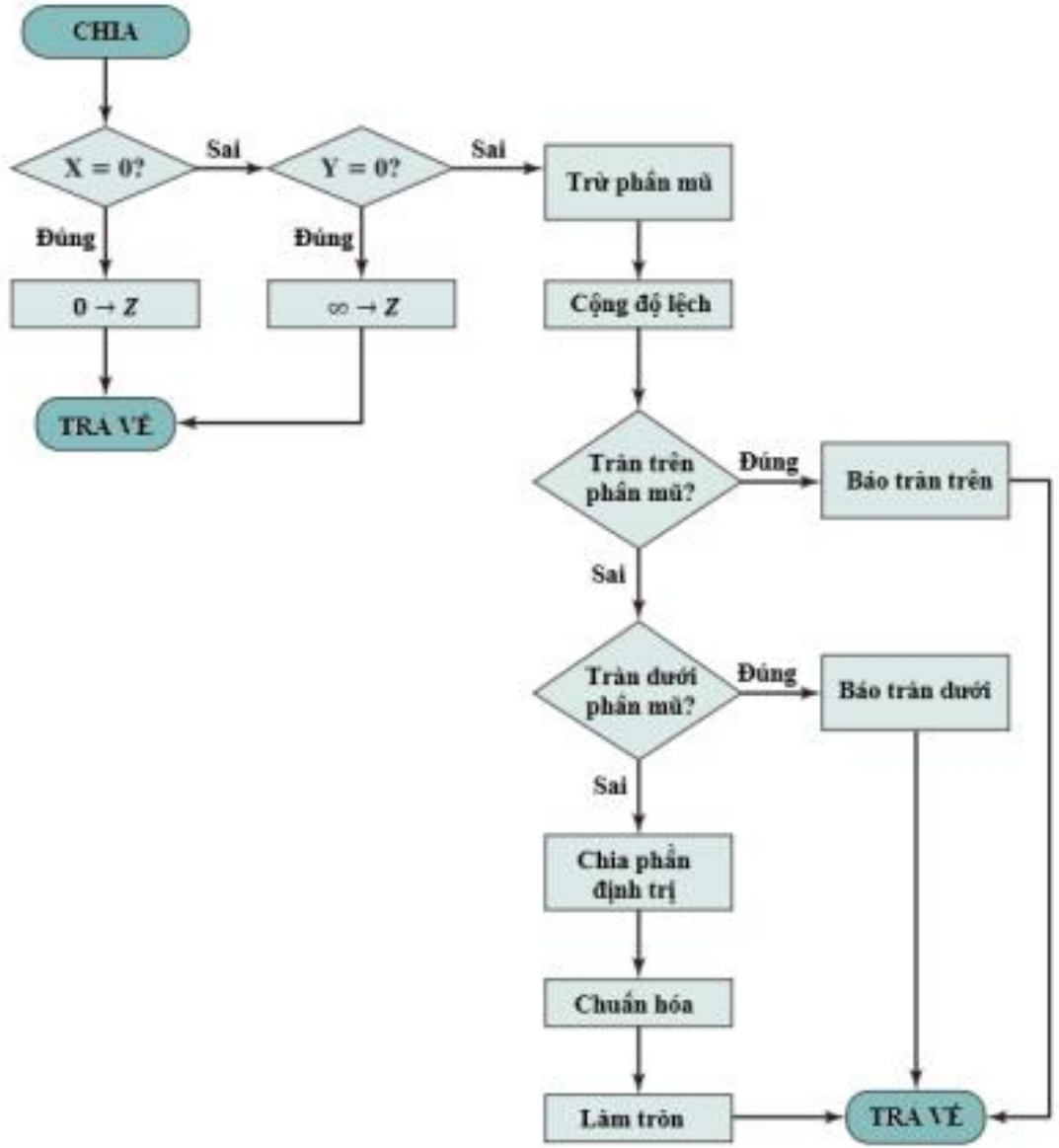
b. Phép nhân số dấu chấm động

- Bộ nhân phần định trị được thực hiện giống như thuật toán nhân hai số nhị phân thông thường.
- Kết quả sẽ có kích thước lớn gấp đôi, tuy nhiên giá trị sẽ được chuẩn hóa theo dạng biểu diễn



Hình 9.23 Phép nhân dấu chấm động (X × Y → Z)

c. Phép chia số dấu chấm động



Hình 9.24 Phép chia dấu chấm động (X/Y → Z)

Tổng kết Chương 8

- ALU
- Biểu diễn số nguyên
  - Biểu diễn dấu-độ lớn
  - Biểu diễn mã bù hai
  - Mở rộng phạm vi
- Thực hiện các phép toán với số nguyên
  - Phép đảo
  - Cộng và trừ
  - Nhân
  - Chia
- Biểu diễn dấu chấm động
  - Nguyên tắc
  - Chuẩn IEEE cho biểu diễn dấu chấm động nhị phân
- Thực hiện các phép toán với số dấu chấm động
  - Cộng, trừ, nhân và chia

Câu hỏi ôn tập

1. Giải thích cách làm thế nào để xác định xem một số có phải là số âm không trong biểu diễn dấu-độ lớn, bù hai.
2. Sự khác biệt giữa biểu diễn bù hai một số và lấy bù hai của một số là gì?
3. Nguyên tắc mở rộng phạm vi của số biểu diễn dạng dấu-độ lớn là gì?
4. Nguyên tắc mở rộng phạm vi của số biểu diễn dạng bù hai là gì?
5. Phép đảo dấu một số biểu diễn dạng dấu-độ lớn được thực hiện như thế nào?
6. Phép đảo dấu một số biểu diễn dạng bù hai được thực hiện như thế nào?
7. Các phép toán số học đối với số nguyên: Phép đảo dấu, Phép cộng, Phép trừ, Phép nhân, Phép chia
8. Nêu qui tắc xét tràn khi thực hiện cộng theo dạng biểu diễn bù hai.
9. Giải thích tóm tắt phương pháp biểu diễn số dấu chấm động theo chuẩn IEEE 754.

Hình ảnh và nội dung trong bài giảng này tham khảo từ cuốn sách và slide bài giảng “Computer Organization and Architecture”, 10th Edition, của tác giả William Stallings.