
TRƯỜNG ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

# KIẾN TRÚC MÁY TÍNH

## Computer Architecture

Course ID: IT3283

Nguyễn Kim Khánh

# Nội dung học phần

Chương 1. Giới thiệu chung

Chương 2. Hệ thống máy tính

Chương 3. Kiến trúc tập lệnh

Chương 4. Số học máy tính

Chương 5. Bộ xử lý

Chương 6. Bộ nhớ máy tính

Chương 7. Hệ thống vào-ra

Chương 8. Các kiến trúc song song

# Chương 4
# SỐ HỌC MÁY TÍNH

# Nội dung của chương 4

4.1. Phép cộng và phép trừ số nguyên

4.2. Phép nhân số nguyên

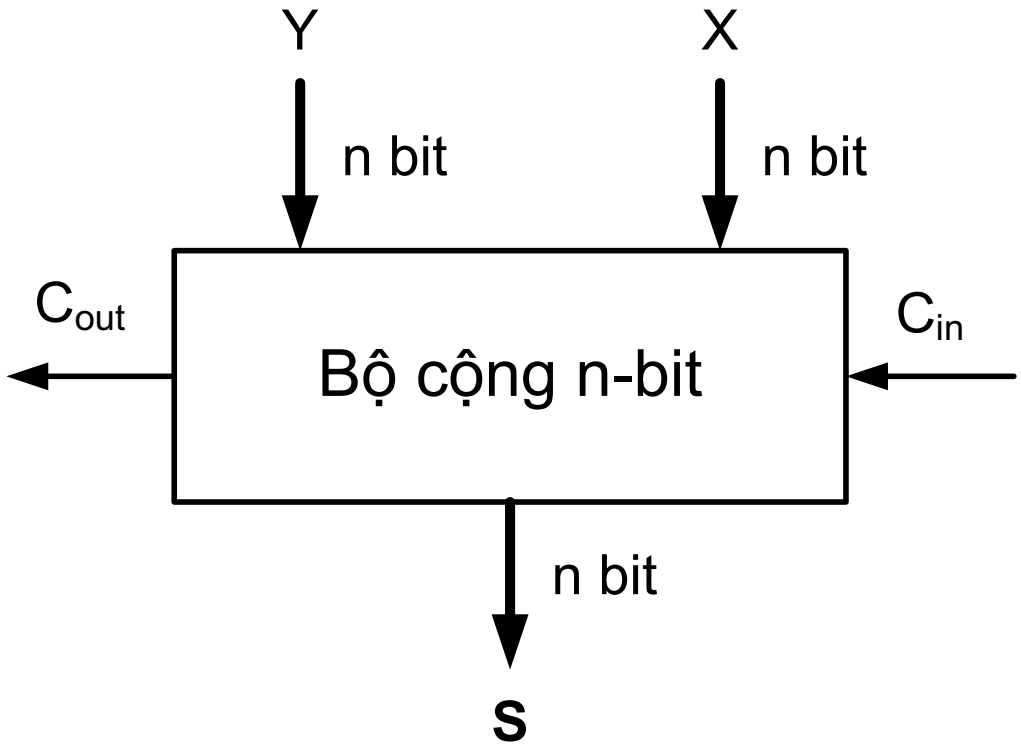
4.3. Phép chia số nguyên

4.4. Số dấu phẩy động

# 4.1. 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

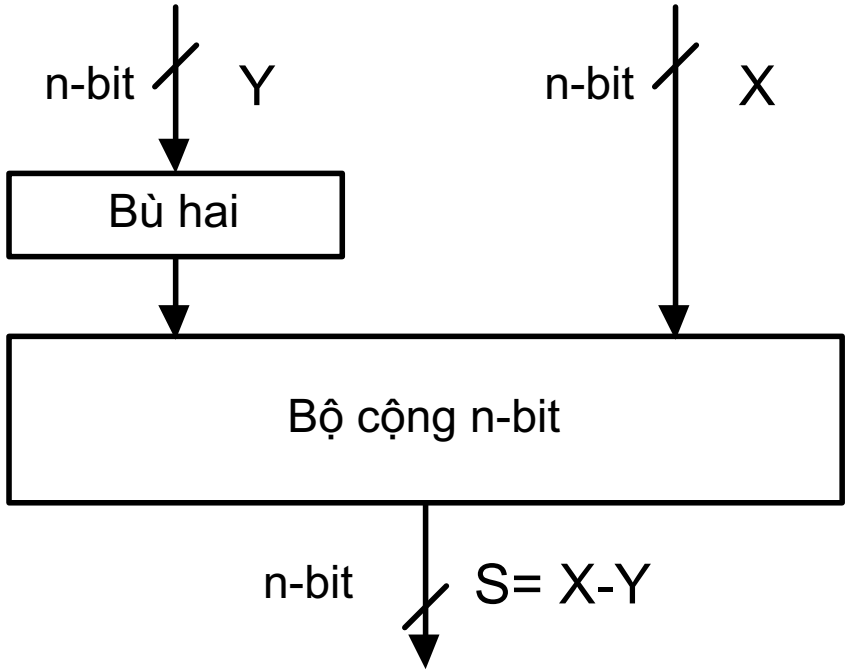
- ( + 75) = 0100 1011
  +( + 82) = 0101 0010
  +157 1001 1101
  = - 128+16+8+4+1= -99 → sai

- ( - 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



# 4.2. Phép nhân số nguyên

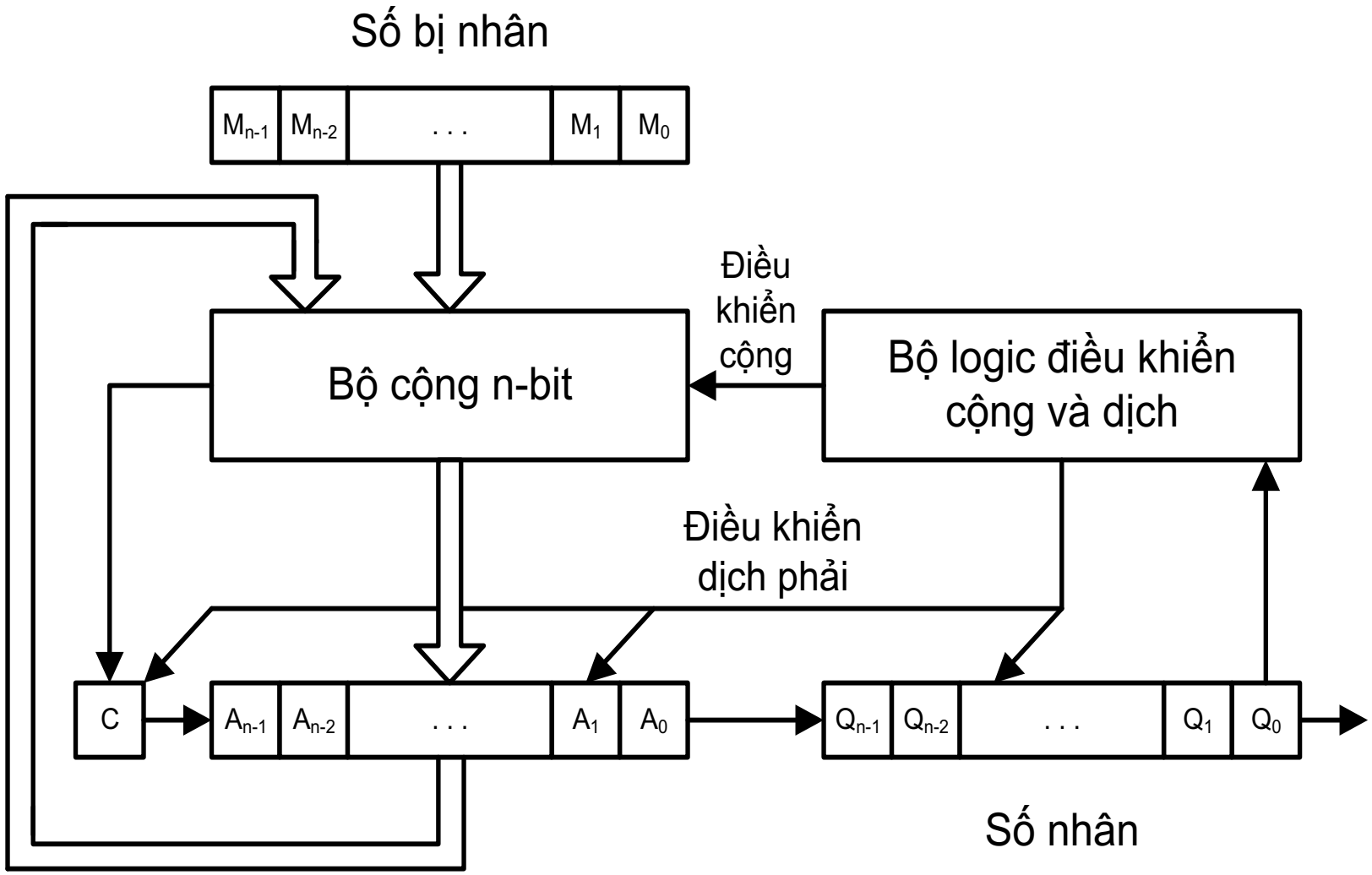
## Nhân số nguyên không dấu

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

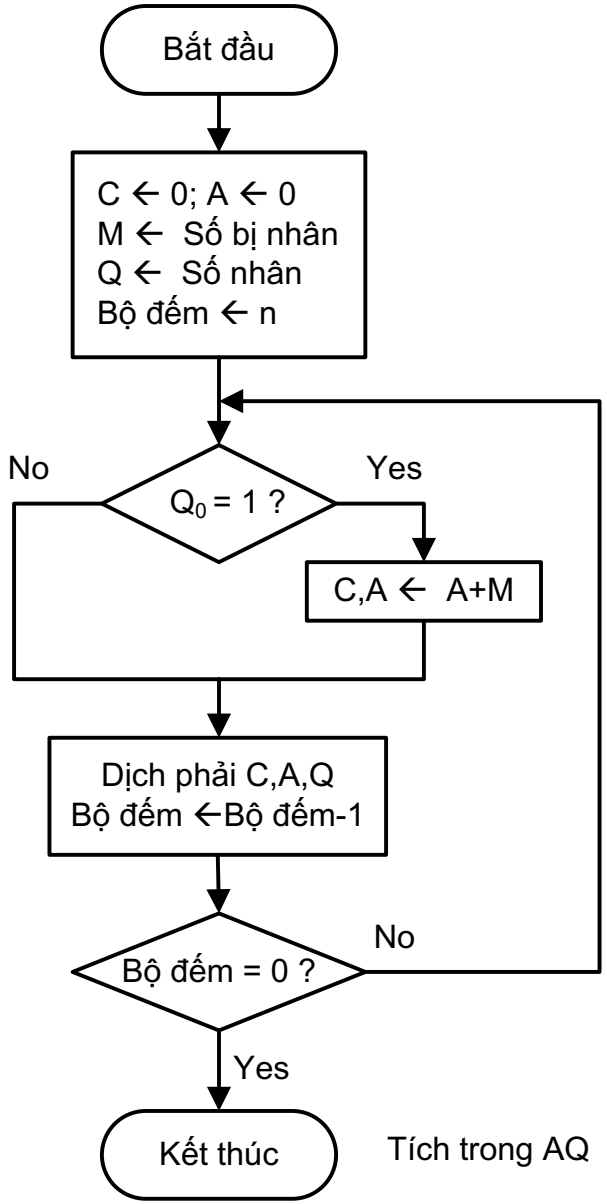
# Nhân số nguyên không dấu (tiếp)

- 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 |

# Nhân số nguyên có dấu

- Sử dụng thuật giải nhân không dấu
- Sử dụng thuật giải Booth (tham khảo sách COA)

# 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)

# 4.3. Chia số nguyên

Số bị chia 10010011 \1011 Số chia

- 1011 00001101 Thương

001110

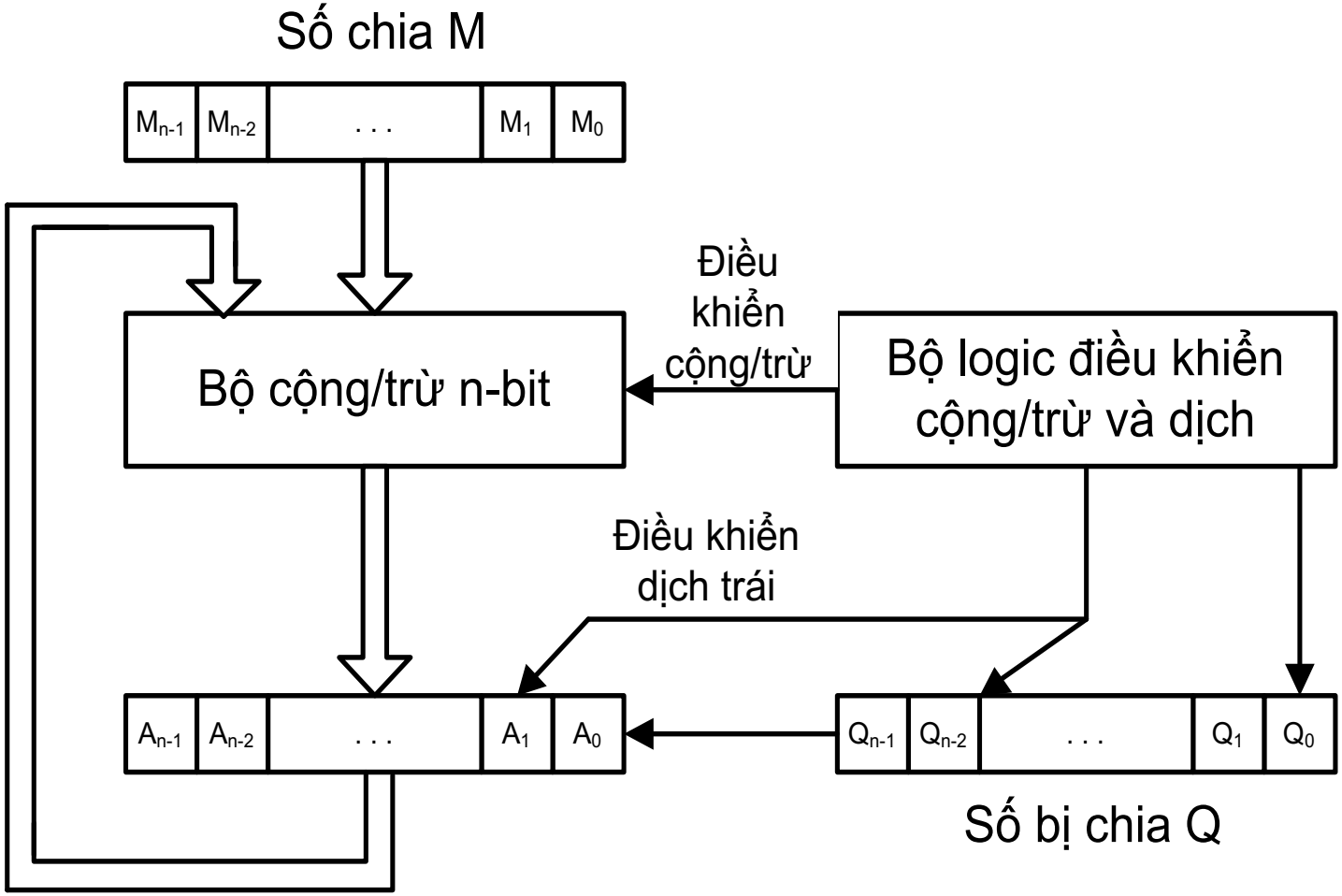
- 1011

001111

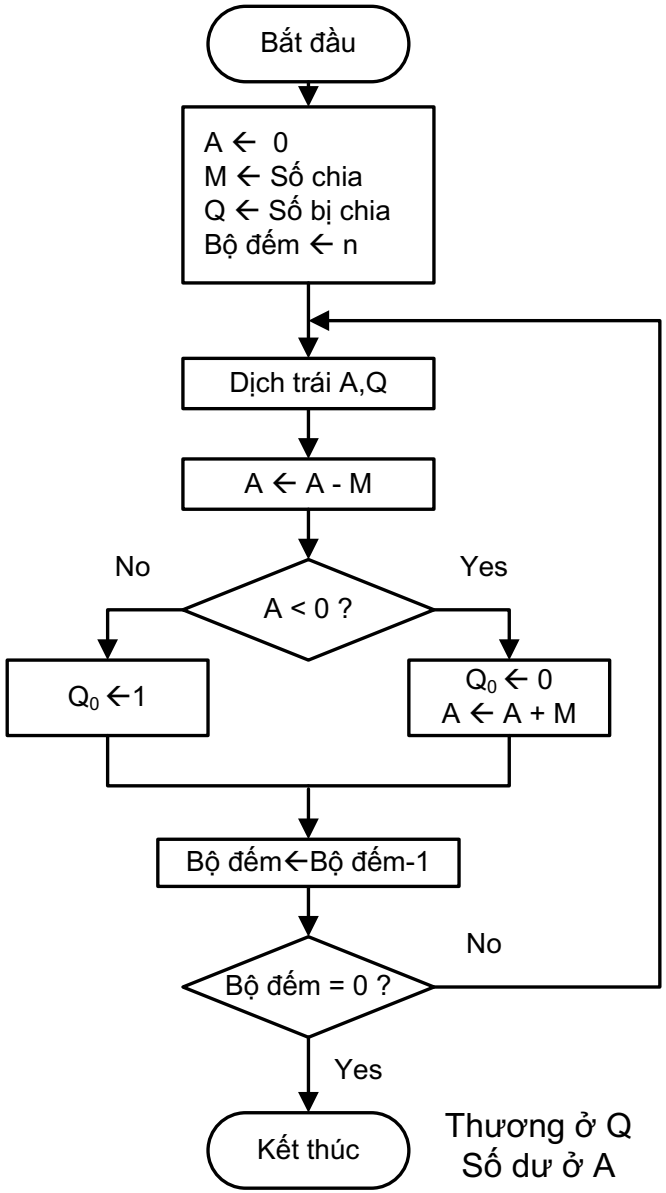
- 1011

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 | | |

# 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 |

# Các lệnh nhân và chia số nguyên của MIPS

- MIPS có hai thanh ghi 32-bit: HI (high) và LO (low)
- Các lệnh liên quan:
  - `mult rs, rt  # nhân số nguyên có dấu`
  - `multu rs, rt  # nhân số nguyên không dấu`
    - Tích 64-bit nằm trong cặp thanh ghi HI/LO
  - `div rs, rt  # chia số nguyên có dấu`
  - `divu rs, rt  # chia số nguyên không dấu`
    - HI: chứa phần dư, LO: chứa thương
  - `mfhi rd  # Move from Hi to rd`
  - `mflo rd  # Move from LO to rd`

# 4.4. Số dấu phẩy động

## 1. Nguyên tắc chung

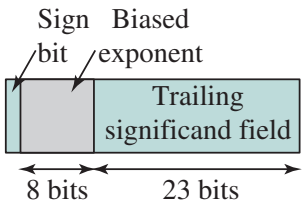
- Floating Point Number → biểu diễn cho số thực
- Tổng quát: một số thực X được biểu diễn theo kiểu số dấu phẩy động như sau:

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

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

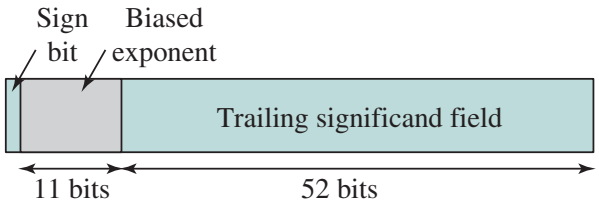
# 2. Chuẩn IEEE754-2008

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



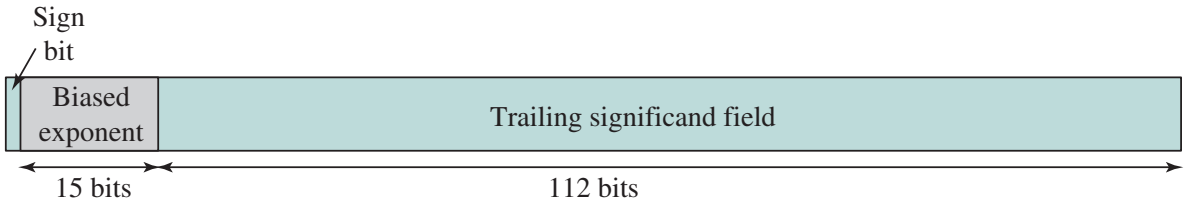
(a) Binary32 format

  - Dạng 64-bit



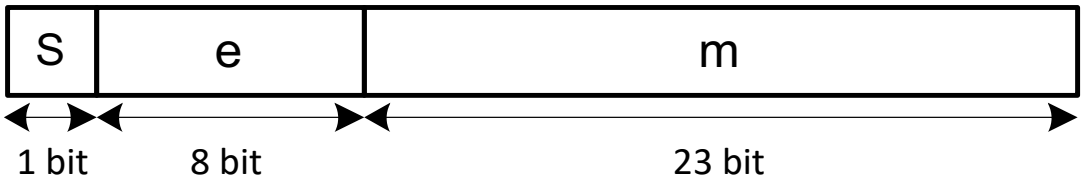
(b) Binary64 format

  - 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 → e = 127 + 6 = $133_{(10)}$ = $1000\ 0101_{(2)}$
- Vậy:

X = 0100 0010 1010 0111 1000 0000 0000 0000

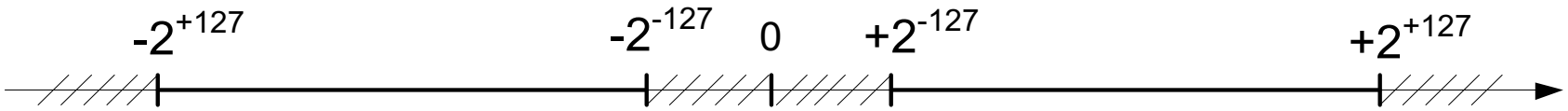
# Các qui ước đặc biệt

- Các bit của e bằng 0, các bit của m bằng 0, thì X = $\pm$ 0
  x000 0000 0000 0000 0000 0000 0000 0000 → X = $\pm$ 0
- Các bit của e bằng 1, các bit của m bằng 0, thì X = $\pm \infty$
  x111 1111 1000 0000 0000 0000 0000 0000 → X =$\pm \infty$
- 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 $E2 \geq E1$

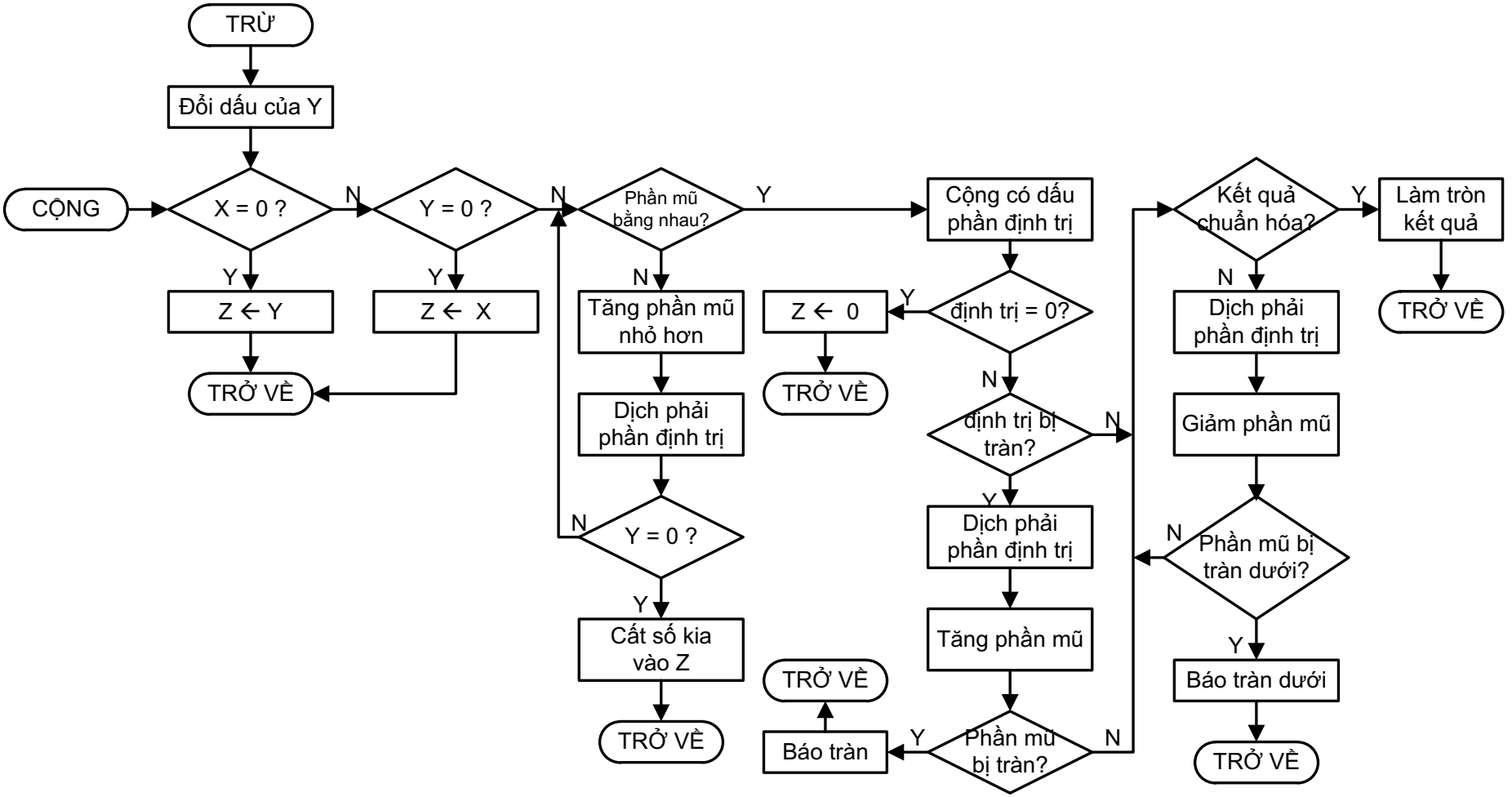
# Các khả năng tràn số

- 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

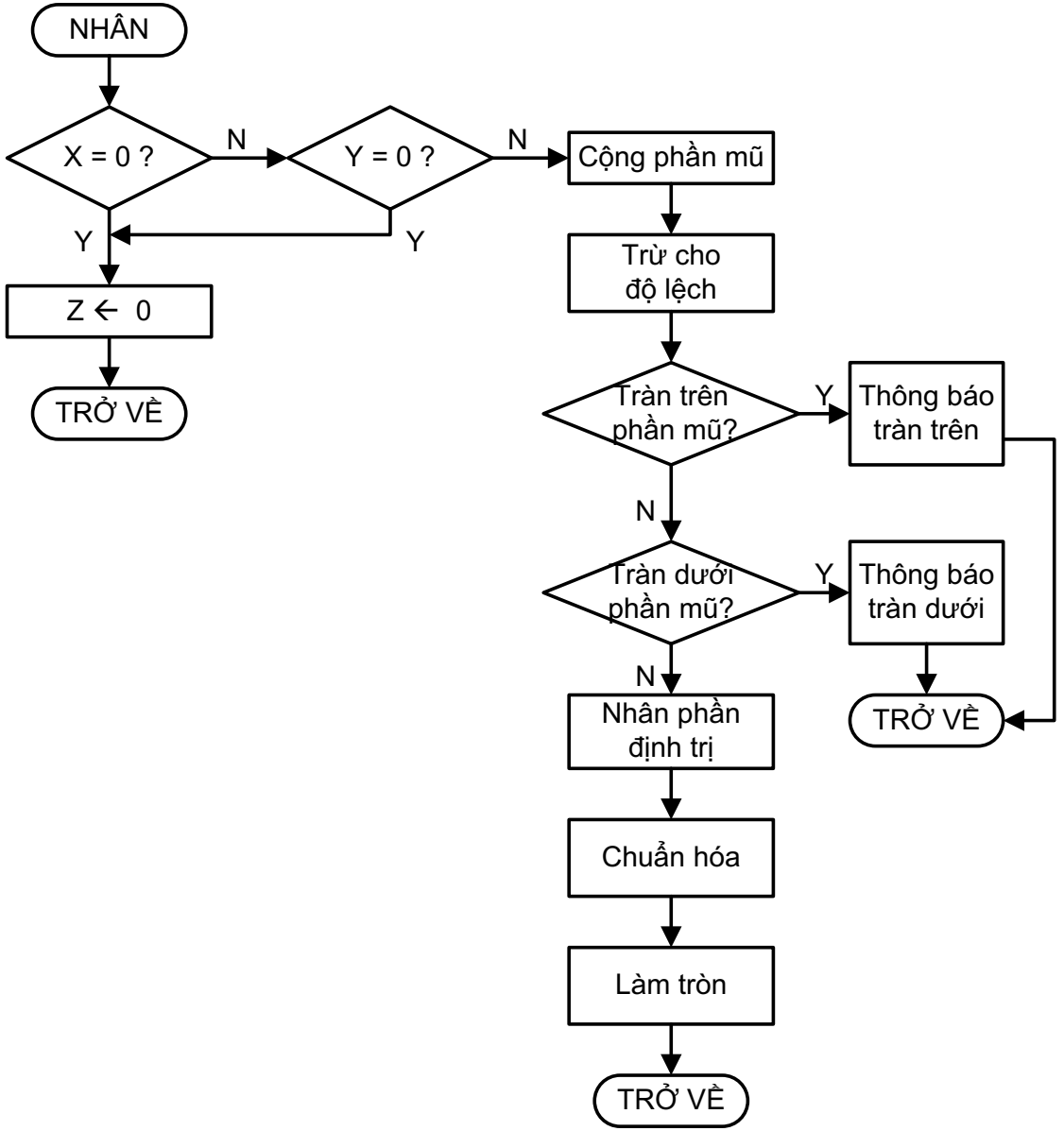
# Phép cộng và phép trừ

- 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

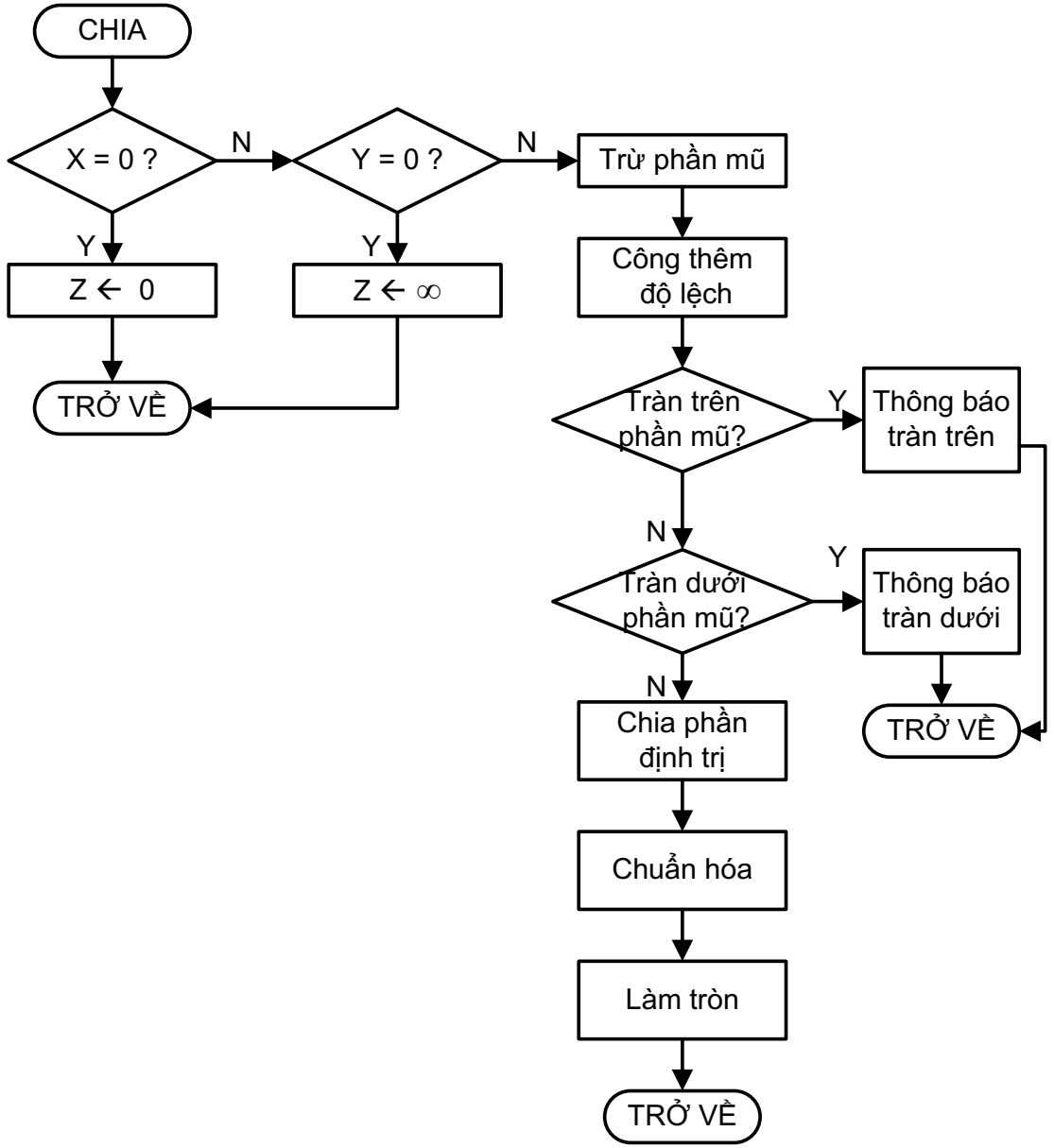
# Thuật toán cộng/trừ số dấu phẩy động



# Thuật toán nhân số dấu phẩy động



# Thuật toán chia số dấu phẩy động



# 4. Các lệnh với số dấu phẩy động của MIPS

- Các thanh ghi số dấu phẩy động
  - 32 thanh ghi 32-bit (single-precision): $f0, $f1, … $f31
  - Cặp đôi để chứa dữ liệu dạng 64-bit (double-precision): $f0/$f1, $f2/$f3, …
- Các lệnh số dấu phẩy động chỉ thực hiện trên các thanh ghi số dấu phẩy động
- Lệnh load và store với FP
  - `lwc1, ldc1, swc1, sdc1`
    - `Ví dụ:  ldc1 $f8, 32($s2)`

# Các lệnh với số dấu phẩy động

- Các lệnh số học với số FP 32-bit (single-precision)
  - `add.s, sub.s, mul.s, div.s`
  - `VD:  add.s $f0, $f1, $f6`
- Các lệnh số học với số FP 64-bit (double-precision)
  - `add.d, sub.d, mul.d, div.d`
  - `VD:  mul.d $f4, $f4, $f6`
- Các lệnh so sánh
  - `c.xx.s, c.xx.d` (trong đó xx là eq, lt, le, …)
  - Thiết lập hoặc xóa các bit mã điều kiện
  - `VD:  c.lt.s $f3, $f4`
- Các lệnh rẽ nhánh dựa trên mã điều kiện
  - `bc1t, bc1f`
  - `VD:  bc1t TargetLabel`

# Hết chương 4