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

KIẾN TRÚC MÁY TÍNH

Computer Architecture

Course ID: IT3030, IT3034, IT3283

Version: CA.RISCV.2024.1

RISC-V®

Nội dung học phần

Chương 4
SỐ HỌC MÁY TÍNH

Nội dung

4.0. Hệ đếm (ôn tập)

Phụ lục A. Cơ bản về logic số (dành cho sv KHMT)

4.1. Biểu diễn số nguyên

4.2. Phép cộng và phép trừ số nguyên

4.3. Phép nhân và phép chia số nguyên

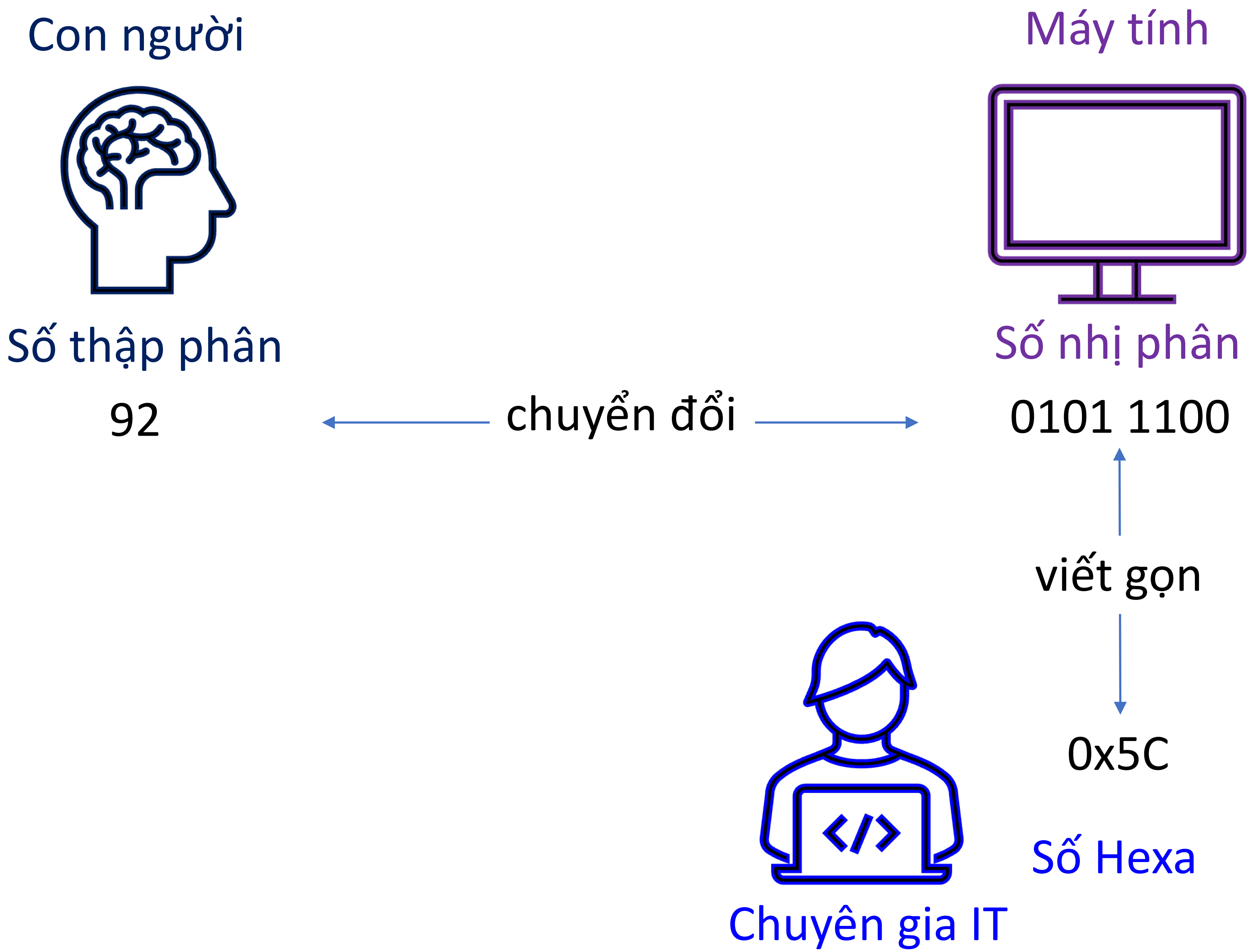
4.4. Số dấu phẩy động

4.0. Hệ đếm

Các hệ đếm cơ bản:

- Hệ thập phân (Decimal System)
  → con người sử dụng
- Hệ nhị phân (Binary System)
  → máy tính sử dụng
- Hệ mười sáu (Hexadecimal System)
  → dùng để viết gọn cho số nhị phân

Các hệ đếm cơ bản



1. Hệ thập phân

- Cơ số 10
- 10 chữ số: 0,1,2,3,4,5,6,7,8,9
- Dùng n chữ số thập phân có thể biểu diễn được 10^n giá trị khác nhau:
  - 00...000 = 0
  - 99...999 = $10^n - 1$

Ví dụ số thập phân

472.38

2 1 0 -1 -2

$$= 4x10^2 + 7x10^1 + 2x10^0 + 3x10^{-1} + 8x10^{-2}$$

- Các chữ số của phần nguyên:
  - 472 : 10 = 47 dư 2
  - 47 : 10 = 4 dư 7
  - 4 : 10 = 0 dư 4
- Các chữ số của phần lẻ:
  - 0.38 x 10 = 3.8 phần nguyên = 3
  - 0.8 x 10 = 8.0 phần nguyên = 8

Dạng tổng quát của số thập phân

$$A = a_n a_{n-1} \dots a_1\ a_0, a_{-1}\ a_{-2} \dots a_{-m}$$

Giá trị của A được hiểu như sau:

$$A = a_n 10^n + a_{n-1} 10^{n-1} + \dots + a_1\, 10^1 + a_0 10^0 + a_{-1} 10^{-1} + \dots + a_{-m} 10^{-m}$$

$$A = \sum_{i=-m}^{n} a_i 10^i$$

2. Hệ nhị phân

- Cơ số 2
- 2 chữ số nhị phân: 0 và 1
- Chữ số nhị phân được gọi là **bit** (binary digit)
  - bit là đơn vị thông tin nhỏ nhất
- Dùng n bit có thể biểu diễn được 2^n giá trị khác nhau:
  - 00...000 = 0
  - 11...111 = $2^n - 1$
- Các lệnh của chương trình và dữ liệu trong máy tính đều được mã hóa bằng số nhị phân

Biểu diễn số nhị phân

Số nhị phân				Số thập phân
1-bit	2-bit	3-bit	4-bit	
0	00	000	0000	0
1	01	001	0001	1
	10	010	0010	2
	11	011	0011	3
		100	0100	4
		101	0101	5
		110	0110	6
		111	0111	7
			1000	8
			1001	9
			1010	10
			1011	11
			1100	12
			1101	13
			1110	14
			1111	15

Đơn vị dữ liệu và thông tin trong máy tính

- bit – chữ số nhị phân (binary digit): là đơn vị thông tin nhỏ nhất, cho phép nhận một trong hai giá trị: 0 hoặc 1.
- byte là một tổ hợp 8 bit: có thể biểu diễn được 256 giá trị (2^8)
- Qui ước đơn vị dữ liệu trong Khoa học máy tính:
  - KB (Kilobyte) = 2^{10} bytes = 1024 bytes
  - MB (Megabyte) = 2^{10} KB = 2^{20} bytes (~10^6)
  - GB (Gigabyte) = 2^{10} MB = 2^{30} bytes (~10^9)
  - TB (Terabyte) = 2^{10} GB = 2^{40} bytes (~10^{12})
  - PB (Petabyte) = 2^{10} TB = 2^{50} bytes
  - EB (Exabyte) = 2^{10} PB = 2^{60} bytes

Qui ước mới về ký hiệu đơn vị dữ liệu

Theo thập phân			Theo nhị phân		
Đơn vị	Viết tắt	Giá trị	Đơn vị	Viết tắt	Giá trị
kilobyte	KB	10^3	kibibyte	KiB	$2^{10} = 1024$
megabyte	MB	10^6	mebibyte	MiB	2^{20}
gigabyte	GB	10^9	gibibyte	GiB	2^{30}
terabyte	TB	10^{12}	tebibyte	TiB	2^{40}
petabyte	PB	10^{15}	pebibyte	PiB	2^{50}
exabyte	EB	10^{18}	exbibyte	EiB	2^{60}

Ví dụ số nhị phân

$1101001.1011_{(2)}$ =

6 5 4 3 2 1 0 -1 -2 -3 -4

$= 2^6 + 2^5 + 2^3 + 2^0 + 2^{-1} + 2^{-3} + 2^{-4}$

$= 64 + 32 + 8 + 1 + 0.5 + 0.125 + 0.0625$

$= 105.6875_{(10)}$

Dạng tổng quát của số nhị phân

$$A = a_n a_{n-1} \dots a_1\ a_0, a_{-1}\ a_{-2} \dots\ a_{-m}$$

Giá trị của A được tính như sau:

$$A = a_n 2^n + a_{n-1} 2^{n-1} + \dots + a_1\ 2^1\ + a_0 2^0 + a_{-1} 2^{-1} + \dots + a_{-m} 2^{-m}$$

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

Chuyển đổi số nguyên thập phân sang nhị phân

- Phương pháp 1: chia dần cho 2 rồi lấy phần dư
- Phương pháp 2: Phân tích thành tổng của các số 2^i
  → nhanh hơn

Phương pháp chia dần cho 2

- Ví dụ: chuyển đổi $105_{(10)}$

105 : 2	=	52	dư	1
52 : 2	=	26	dư	0
26 : 2	=	13	dư	0
13 : 2	=	6	dư	1
6 : 2	=	3	dư	0
3 : 2	=	1	dư	1
1 : 2	=	0	dư	1

biểu diễn
số dư
theo chiều
mũi tên

- Kết quả: $105_{(10)} = 1101001_{(2)}$

Phương pháp phân tích thành tổng của các 2^i

- Ví dụ 1: chuyển đổi $105_{(10)}$
  - $105 = 64 + 41 = 64 + 32 + 9$

    $= 64 + 32 + 8 + 1 = 2^6 + 2^5 + 2^3 + 2^0$

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
0	1	1	0	1	0	0	1

  - Kết quả: $105_{(10)} = 0110\ 1001_{(2)}$
- Ví dụ 2:

$17000_{(10)} = 16384 + 512 + 64 + 32 + 8$

$= 2^{14} + 2^9 + 2^6 + 2^5 + 2^3$

$17000_{(10)} = 0100\ \ 0010\ \ 0110\ \ 1000_{(2)}$

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Chuyển đổi số lẻ thập phân sang nhị phân

- Ví dụ 1: chuyển đổi $0.6875_{(10)}$
  - 0.6875 x 2 = 1.375 phần nguyên = 1
  - 0.375 x 2 = 0.75 phần nguyên = 0
  - 0.75 x 2 = 1.5 phần nguyên = 1
  - 0.5 x 2 = 1.0 phần nguyên = 1

biểu diễn theo chiều mũi tên

- Kết quả : $0.6875_{(10)} = 0.1011_{(2)}$

Chuyển đổi số lẻ thập phân sang nhị phân (tiếp)

- Ví dụ 2: chuyển đổi $0.81_{(10)}$
  - 0.81 x 2 = 1.62 phần nguyên = 1
  - 0.62 x 2 = 1.24 phần nguyên = 1
  - 0.24 x 2 = 0.48 phần nguyên = 0
  - 0.48 x 2 = 0.96 phần nguyên = 0
  - 0.96 x 2 = 1.92 phần nguyên = 1
  - 0.92 x 2 = 1.84 phần nguyên = 1
  - 0.84 x 2 = 1.68 phần nguyên = 1
- $0.81_{(10)} \approx 0.1100111_{(2)}$

3. Hệ mười sáu (Hexa)

- Cơ số 16
- 16 chữ số: 0,1,2,3,4,5,6,7,8,9, A,B,C,D,E,F
- Dùng để viết gọn cho số nhị phân: cứ một nhóm 4-bit sẽ được thay bằng một chữ số Hexa

Quan hệ giữa số nhị phân và số Hexa

Ví dụ:

- $0000\ 1000_{(2)} = 08_{(16)}$
- $1011\ 0011_{(2)} = B3_{(16)}$

- $0010\ 1101\ 1001\ 1010_{(2)} = 2D9A_{(16)}$
- $1111\ 1111\ 1111\ 1111_{(2)} = FFFF_{(16)}$

4-bit	Số Hexa	Thập phân
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

Phụ lục A. Cơ bản về logic số

- Đại số Boolean
- Các cổng logic
- Mạch tổ hợp
- Mạch dãy

1. Đại số Boolean

- Đại số Boolean sử dụng các biến logic và phép toán logic
- Biến logic có thể nhận giá trị 1 (TRUE) hoặc 0 (FALSE)
- Các phép toán logic cơ bản: AND, OR và NOT
  - $\text{A AND B} = \text{A} \cdot \text{B hay AB}$
  - $\text{A OR B} = \text{A} + \text{B}$
  - $\text{NOT A} = \overline{\text{A}}$
  - Thứ tự ưu tiên: NOT > AND > OR
- Thêm các phép toán logic: NAND, NOR, XOR
  - $\text{A NAND B} = \text{NOT (A AND B)} = \overline{\text{A} \cdot \text{B}}$
  - $\text{A NOR B} = \text{NOT (A OR B)} = \overline{\text{A} + \text{B}}$
  - $\text{A XOR B} = \text{A} \oplus \text{B} = \text{A} \cdot \overline{\text{B}} + \overline{\text{A}} \cdot \text{B}$

Phép toán đại số Boolean

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	**1**	**1**

A	B	A OR B
0	**0**	**0**
0	1	1
1	0	1
1	1	1

A	NOT A
0	1
1	0

NOT là phép toán 1 biến

A	B	A NAND B
0	0	1
0	1	1
1	0	1
1	**1**	**0**

A	B	A NOR B
0	**0**	**1**
0	1	0
1	0	0
1	1	0

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	**1**	**0**

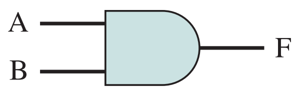
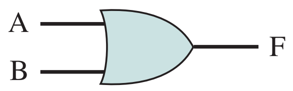
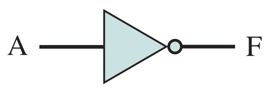
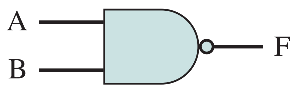
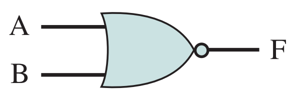
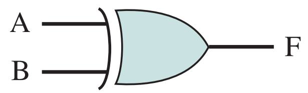
Các đồng nhất thức của đại số Boolean

$A \cdot B = B \cdot A$	$A + B = B + A$
$A \cdot (B + C) = (A \cdot B) + (A \cdot C)$	$A + (B \cdot C) = (A + B) \cdot (A + C)$
$1 \cdot A = A$	$0 + A = A$
$A \cdot \overline{A} = 0$	$A + \overline{A} = 1$
$0 \cdot A = 0$	$1 + A = 1$
$A \cdot A = A$	$A + A = A$
$A \cdot (B \cdot C) = (A \cdot B) \cdot C$	$A + (B + C) = (A + B) + C$
$\overline{A \cdot B} = \overline{A} + \overline{B}$ (Định lý De Morgan)	$\overline{A + B} = \overline{A} \cdot \overline{B}$ (Định lý De Morgan)

2. Các cổng logic (Logic Gates)

- Thực hiện các phép toán logic:
  - NOT, AND, OR, NAND, NOR, XOR
- Cổng logic một đầu vào:
  - Cổng NOT
- Cổng hai đầu vào:
  - AND, OR, XOR, NAND, NOR
- Cổng nhiều đầu vào:
  - AND, OR, XOR, NAND, NOR

Ký hiệu các cổng logic

Name	Graphical Symbol	Algebraic Function	Truth Table
AND		$F = A \bullet B$ or $F = AB$	A B | F 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1
OR		$F = A + B$	A B | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
NOT		$F = \overline{A}$ or $F = A'$	A | F 0 | 1 1 | 0
NAND		$F = \overline{AB}$	A B | F 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
NOR		$F = \overline{A + B}$	A B | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0
XOR		$F = A \oplus B$	A B | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0

3. Mạch tổ hợp

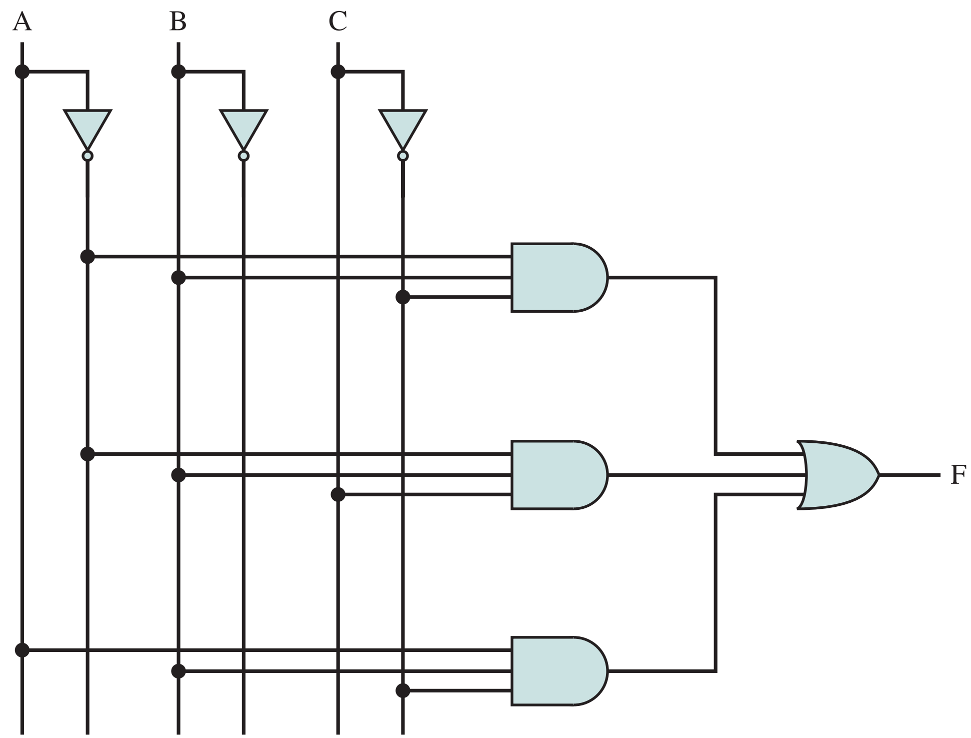
- Mạch logic là mạch bao gồm:
  - Các đầu vào (Inputs)
  - Các đầu ra (Outputs)
  - Đặc tả chức năng (Functional specification)
  - Đặc tả thời gian (Timing specification)
- Các kiểu mạch logic:
  - Mạch tổ hợp (Combinational Circuits)
    - Mạch không nhớ
    - Đầu ra được xác định bởi các giá trị hiện tại của đầu vào
  - Mạch dãy (Sequential Circuits)
    - Mạch có nhớ
    - Đầu ra được xác định bởi các giá trị trước đó và giá trị hiện tại của đầu vào

Mạch tổ hợp

- Mạch tổ hợp là mạch logic trong đó đầu ra chỉ phụ thuộc đầu vào ở thời điểm hiện tại
- Là mạch không nhớ và được thực hiện bằng các cổng logic
- Mạch tổ hợp có thể được định nghĩa theo ba cách:
  - Bảng chân lý (Truth Table)
  - Dạng sơ đồ
  - Phương trình Boolean

Ví dụ

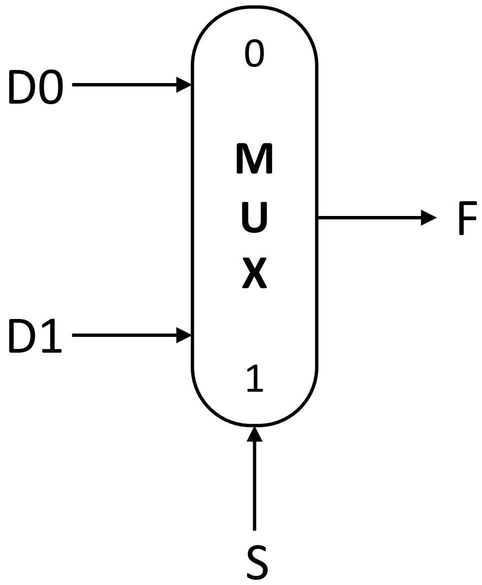
Đầu vào			Đầu ra
A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

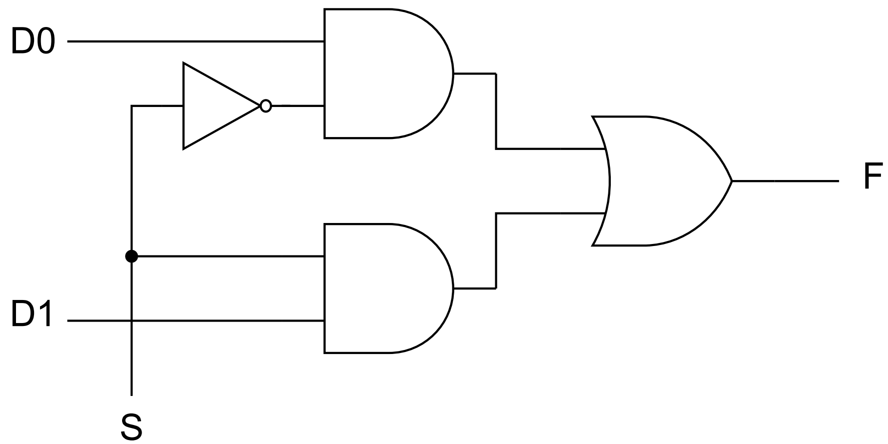


$$F = \overline{A}B\overline{C} + \overline{A}BC + AB\overline{C}$$

Ví dụ: Bộ ghép kênh (Multiplexer - MUX)

Bộ MUX 2 đầu vào

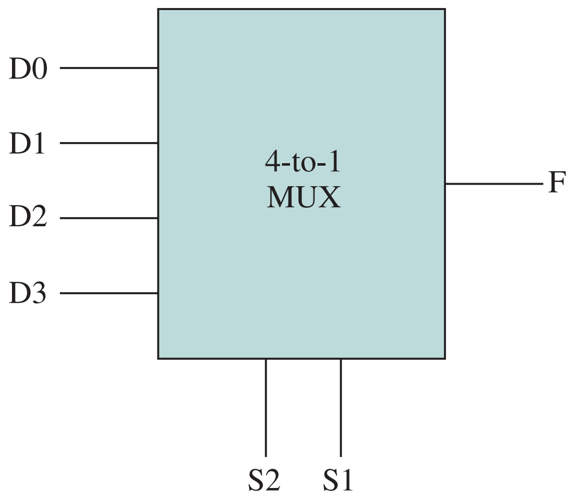


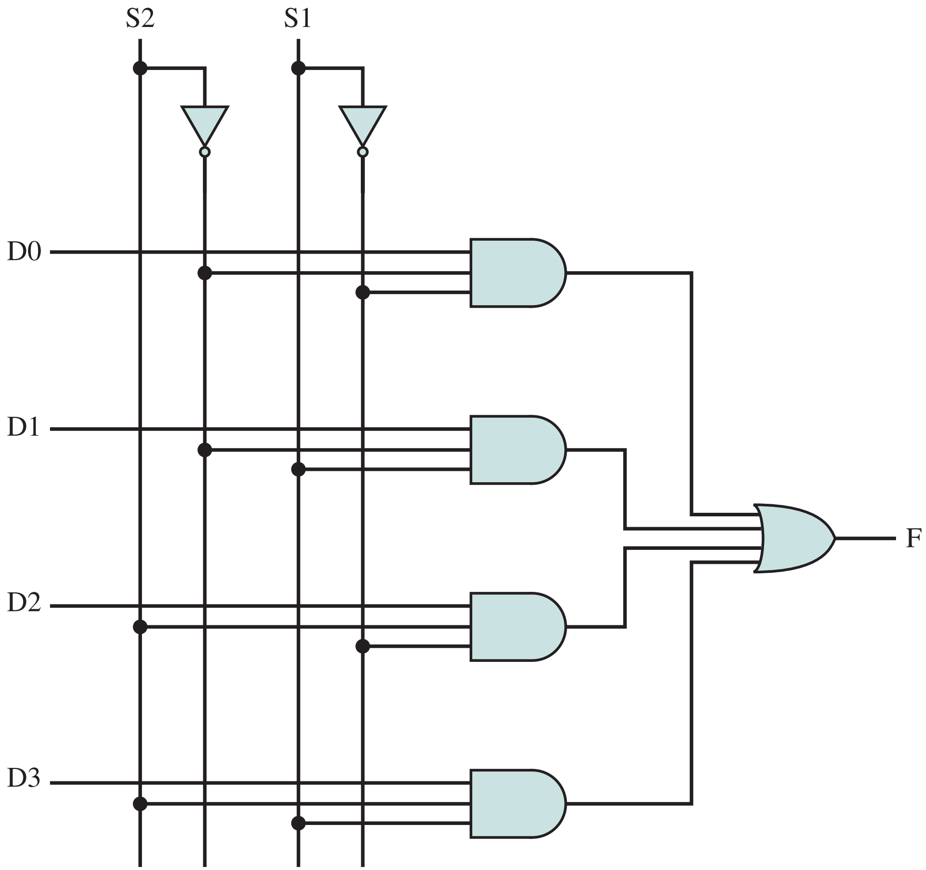


Đầu vào chọn S	Đầu ra F
0	D0
1	D1

$$F = D0.\bar{S} + D1.S$$

Bộ MUX 4 đầu vào



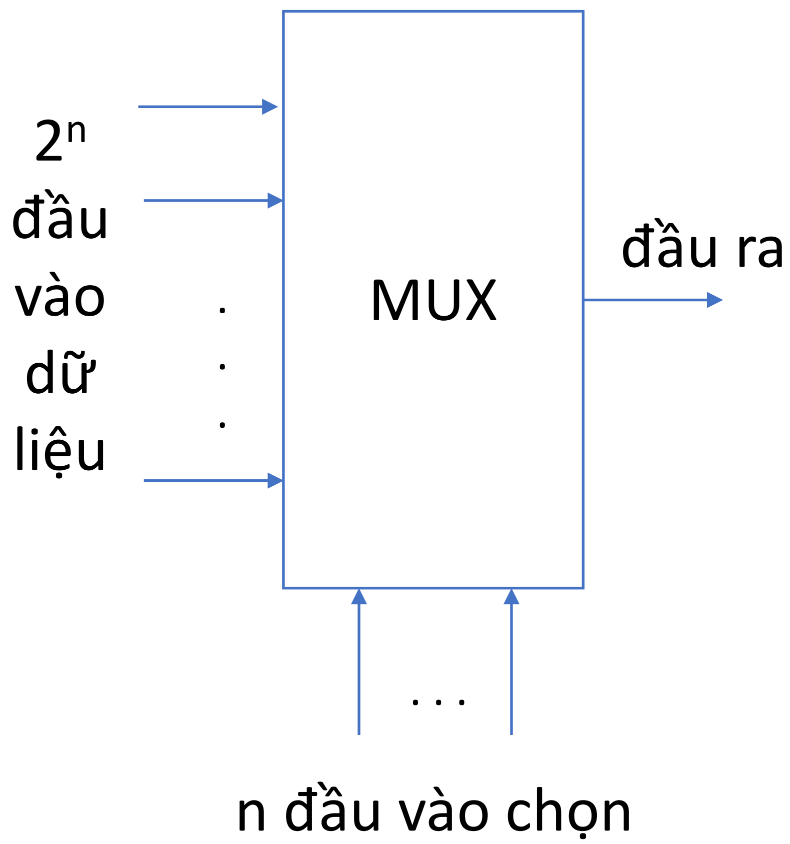


Đầu vào chọn		Đầu ra
S2	S1	F
0	0	D0
0	1	D1
1	0	D2
1	1	D3

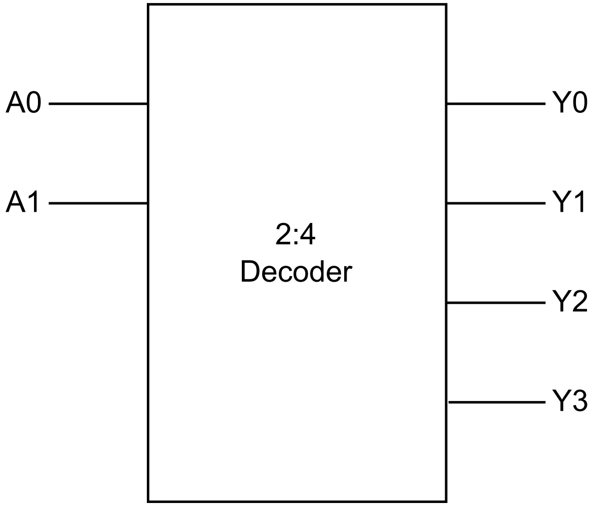
$$F = D0.\overline{S2}.\overline{S1} + D1.\overline{S2}.S1 + D2.S2.\overline{S1} + D3.S2.S1$$

Bộ ghép kênh tổng quát

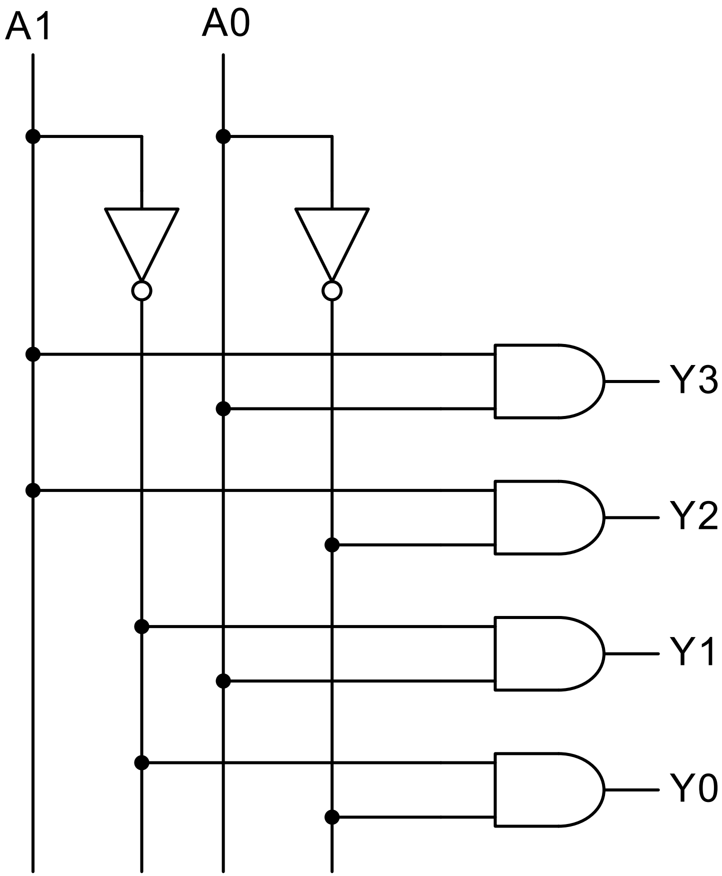
- 2^n đầu vào dữ liệu (D)
- 1 đầu ra dữ liệu (F)
- n đầu vào chọn (S)
- Mỗi tổ hợp đầu vào chọn (S) xác định đầu vào dữ liệu nào (D) sẽ được nối với đầu ra (F)



Bộ giải mã 2 ra 4 (Decoder)



Đầu vào		Đầu ra			
A1	A0	Y3	Y2	Y1	Y0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0



Bộ giải mã tổng quát

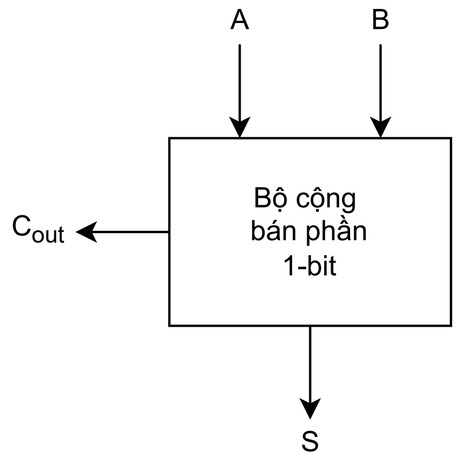
- N đầu vào, 2^N đầu ra
- Với một tổ hợp của N đầu vào, chỉ có một đầu ra tích cực (khác với các đầu ra còn lại)

Bộ cộng

- Bộ cộng bán phần 1-bit (Half-adder)
  - Cộng hai bit tạo ra bit tổng và bit nhớ ra
- Bộ cộng toàn phần 1-bit (Full-adder)
  - Cộng 3 bit
  - Cho phép xây dựng bộ cộng N-bit

Bộ cộng bán phần 1-bit

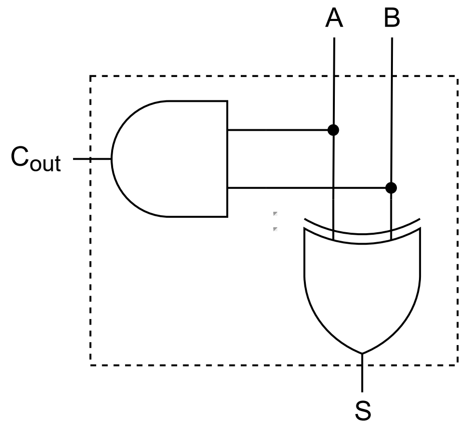
0	0	1	1
+ 0	+ 1	+ 0	+ 1
0	1	1	10



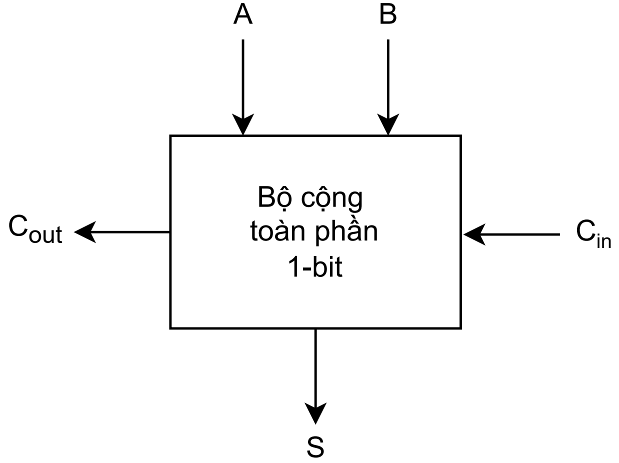
Đầu vào		Đầu ra	
A	B	S	C_{out}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$S = A \oplus B$$

$$C_{out} = AB$$



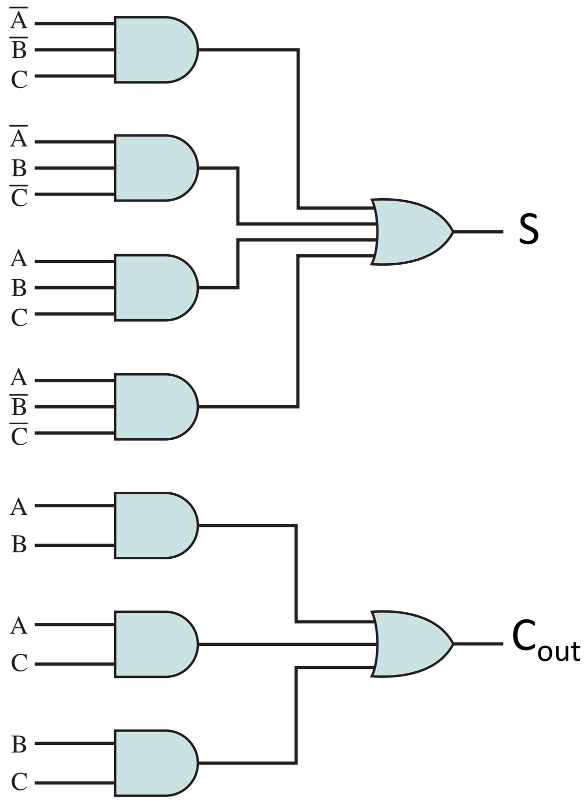
Bộ cộng toàn phần 1-bit



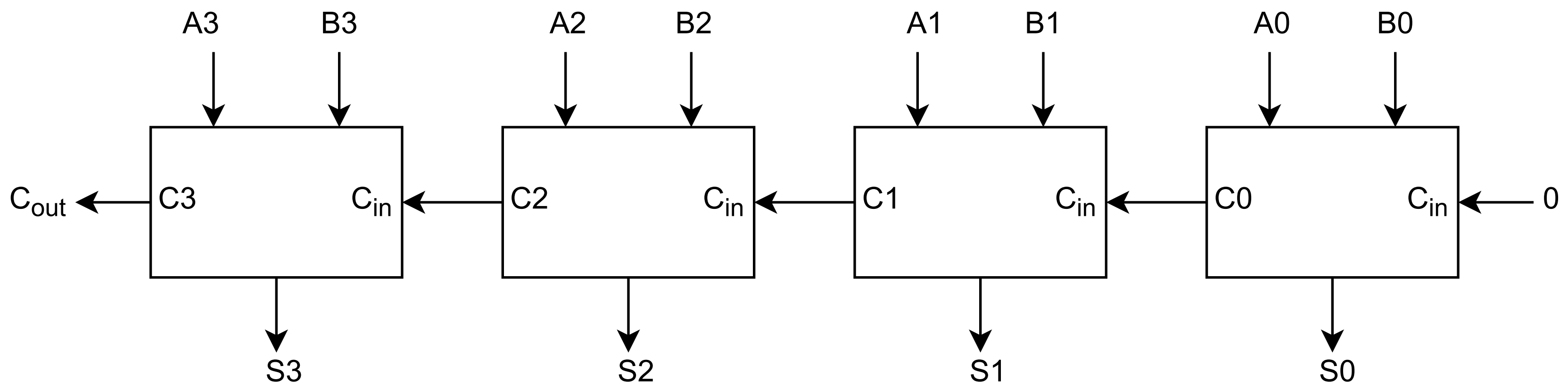
Đầu vào			Đầu ra	
C_{in}	A	B	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

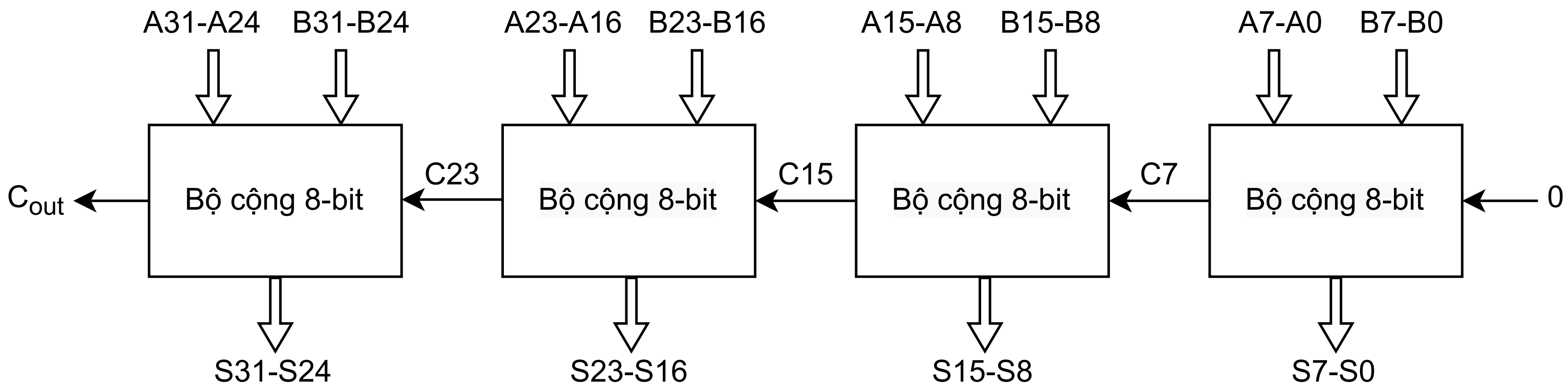
$$S = \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + ABC + A\overline{B}\,\overline{C}$$

$$C_{out} = AB + AC + BC$$



Bộ cộng 4-bit và bộ cộng 32-bit

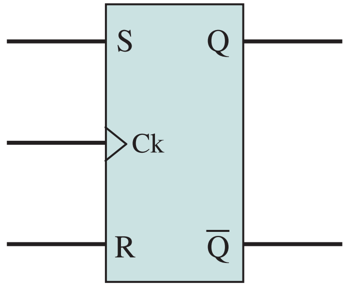
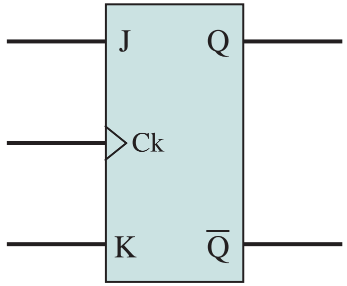
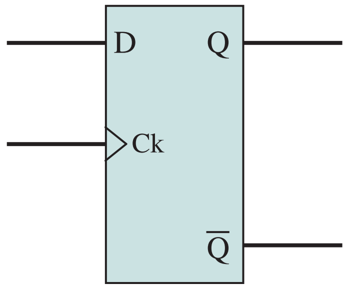




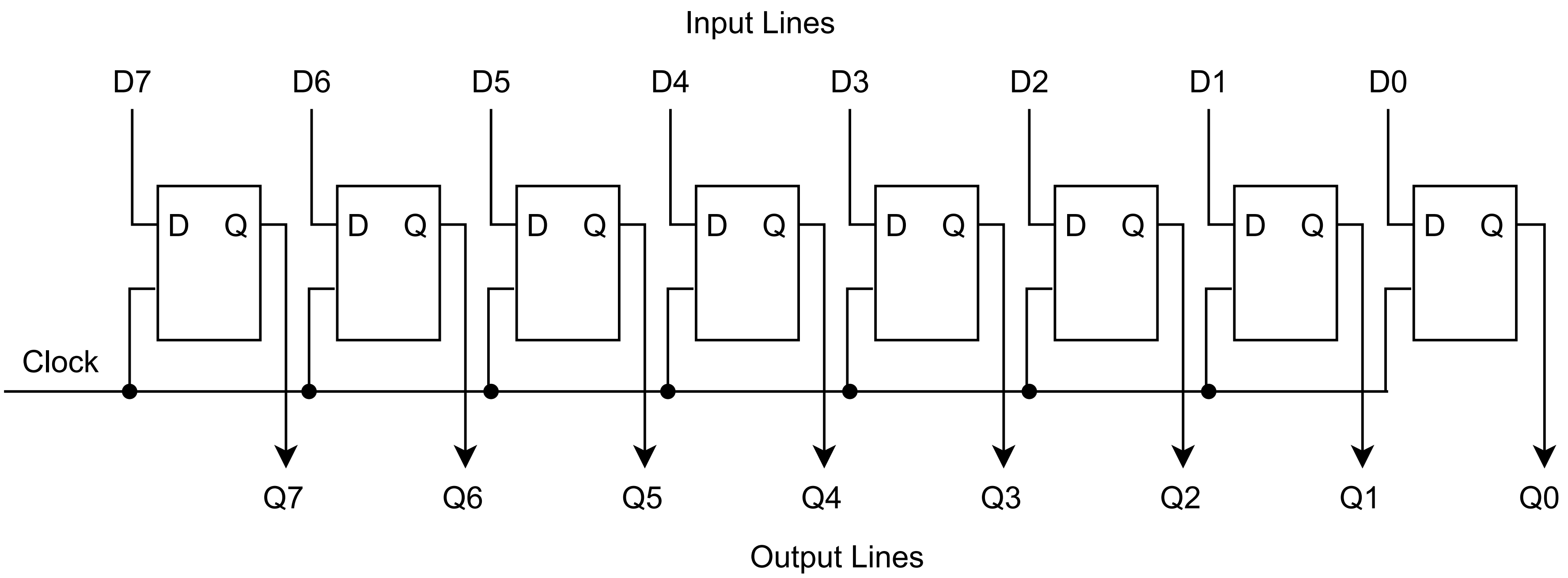
4. Mạch dãy

- Mạch dãy là mạch logic trong đó đầu ra phụ thuộc giá trị đầu vào ở thời điểm hiện tại và đầu vào ở thời điểm quá khứ
- Là mạch có nhớ, được thực hiện bằng phần tử nhớ (Flip-Flop) và có thể kết hợp với các cổng logic

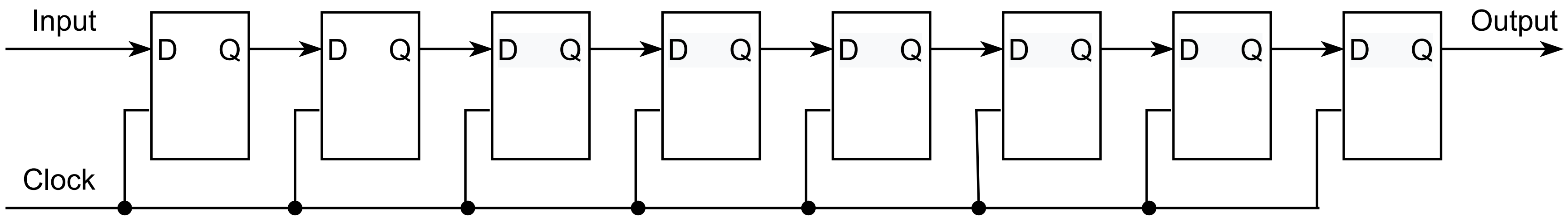
Các Flip-Flop cơ bản

Name	Graphical Symbol	Truth Table
S–R		S R Q_{n+1} 0 0 Q_n 0 1 0 1 0 1 1 1 –
J–K		J K Q_{n+1} 0 0 Q_n 0 1 0 1 0 1 1 1 $\overline{Q_n}$
D		D Q_{n+1} 0 0 1 1

Thanh ghi 8-bit song song



Thanh ghi dịch



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:

$$A = a_{n-1}a_{n-2}...a_2a_1a_0$$

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:

$$A = 41 \ ; \quad B = 150$$

Giải:

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

$A = 0010\ 1001$

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

$B = 1001\ 0110$

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:

$$\begin{array}{r} 1111\ 1111 \\ +\ \underline{0000\ 0001} \\ 1\ \ 0000\ 0000 \end{array}$$

có tràn nhớ ra ngoài (Carry out)

255 + 1 = 0 ???

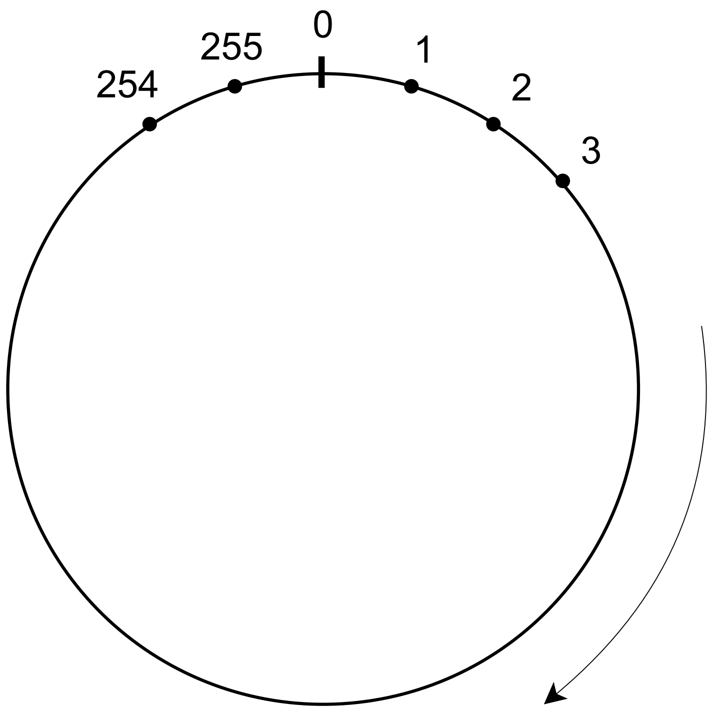
do vượt ra khỏi dải biểu diễn

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:

$$
\begin{array}{rll}
 & 1111\ 1111 & (2^8 - 1) \\
- & \underline{0010\ 0101} & (A) \\
 & 1101\ 1010 &
\end{array}
$$

→ đảo giá trị các bit của A

- Số bù hai của A được tính như sau:

$$
\begin{array}{rll}
 & 1\ 0000\ 0000 & (2^8) \\
- & \underline{0010\ 0101} & (A) \\
 & 1101\ 1011 &
\end{array}
$$

→ 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:

$$A = a_{n-1}a_{n-2}...a_2a_1a_0$$

- 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:

$$A = 0a_{n-2}...a_2a_1a_0$$

Giá trị của số dương:

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

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

Xác định giá trị của số âm

Dạng tổng quát của số âm:

$$A = 1a_{n-2}...a_2a_1a_0$$

Giá trị của số âm:

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

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

Công thức xác định giá trị số âm

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

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

$$= \underbrace{11\,...\,111}_{n-1} - a_{n-2}a_{n-3}\,...\,a_2a_1a_0 + 1$$

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

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

Công thức tổng quát cho số nguyên có dấu

Dạng tổng quát của số nguyên có dấu A:

$$A = a_{n-1}a_{n-2}\dots a_2a_1a_0$$

Giá trị của A được tính 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

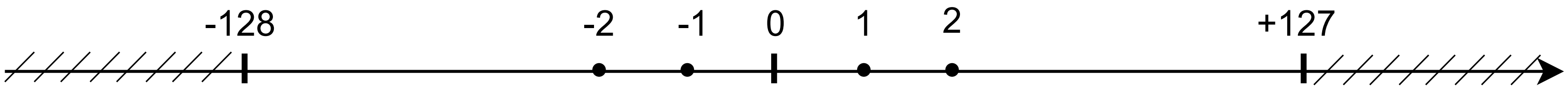
có tràn xảy ra (Overflow)

(do vượt ra khỏi dải biểu diễn)

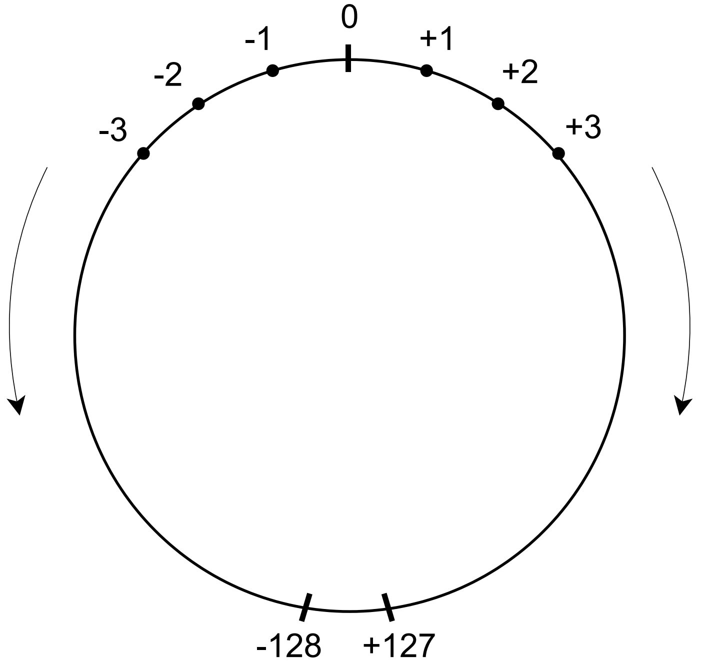
Giá trị thập phân	Biểu diễn bù hai
0	0000 0000
+1	0000 0001
+2	0000 0010
	...
+126	0111 1110
+127	0111 1111
-128	1000 0000
-127	1000 0001
	...
-2	1111 1110
-1	1111 1111

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

    +19 = 0001 0011 (8-bit)

    +19 = 0000 0000 0001 0011 (16-bit)

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

    - 19 = 1110 1101 (8-bit)

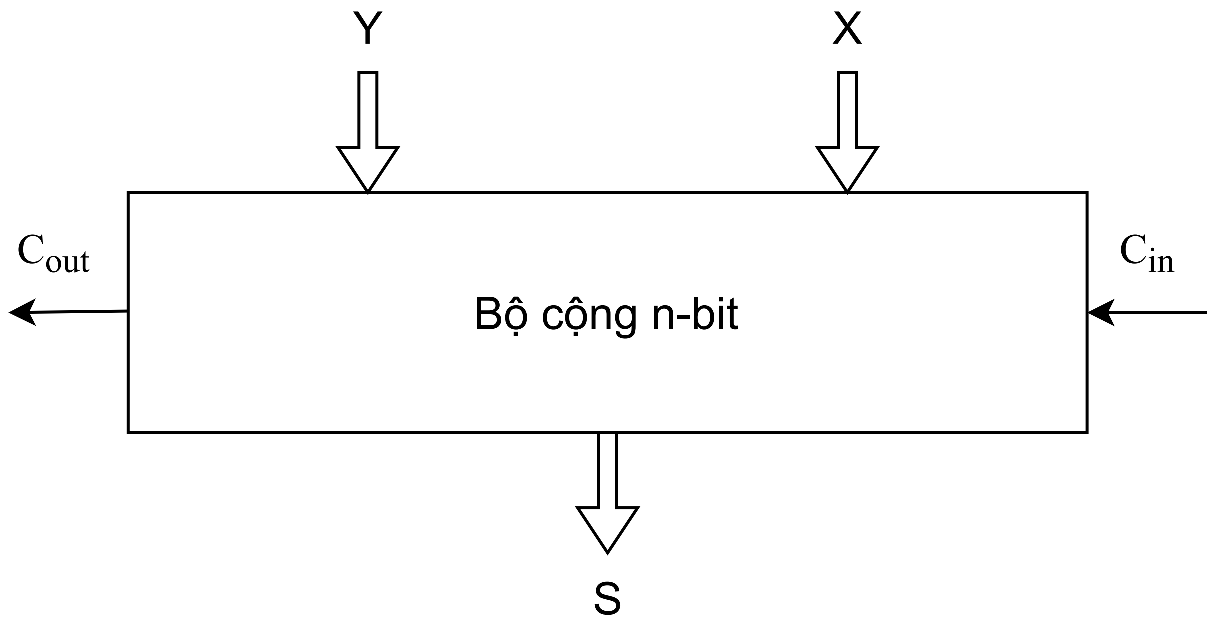
    - 19 = 1111 1111 1110 1101 (16-bit)

    → thêm các bit 1 vào bên trái

4.2. 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 tràn nhớ ra ngoài xảy ra khi:

$$\text{tổng} > (2^n - 1)$$

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 tràn 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]

Giải thích kết quả của chương trình (1)

```
#include <stdio.h>
int main(){
  signed char i, j, k;
  i = 120;
  j = 29;
  k = i + j;
  printf("k = %d\n", k);
  return 0;
}
```

Sau đó đổi kiểu biến i, j, k thành short

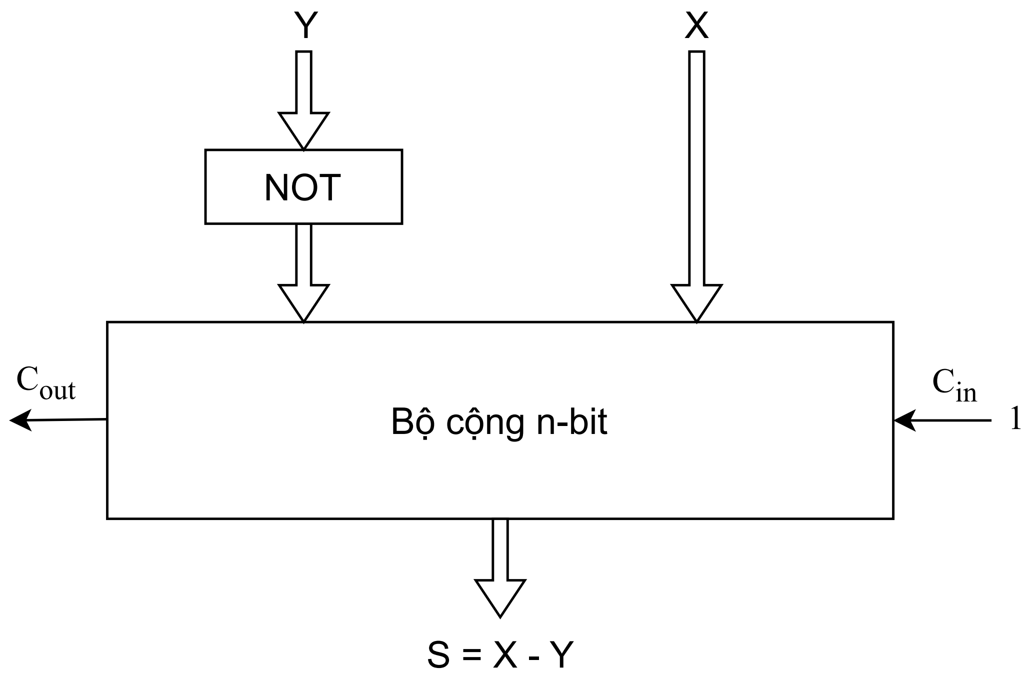
signed char: 8-bit [-128; +127], short: 16-bit [-32768; +32767]

Giải thích kết quả của chương trình (2)

```
#include <stdio.h>
int main(){
  signed char i, j, k;
  i = -86;
  j = -51;
  k = i + j;
  printf("k = %d\n", k);
  return 0;
}
```

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

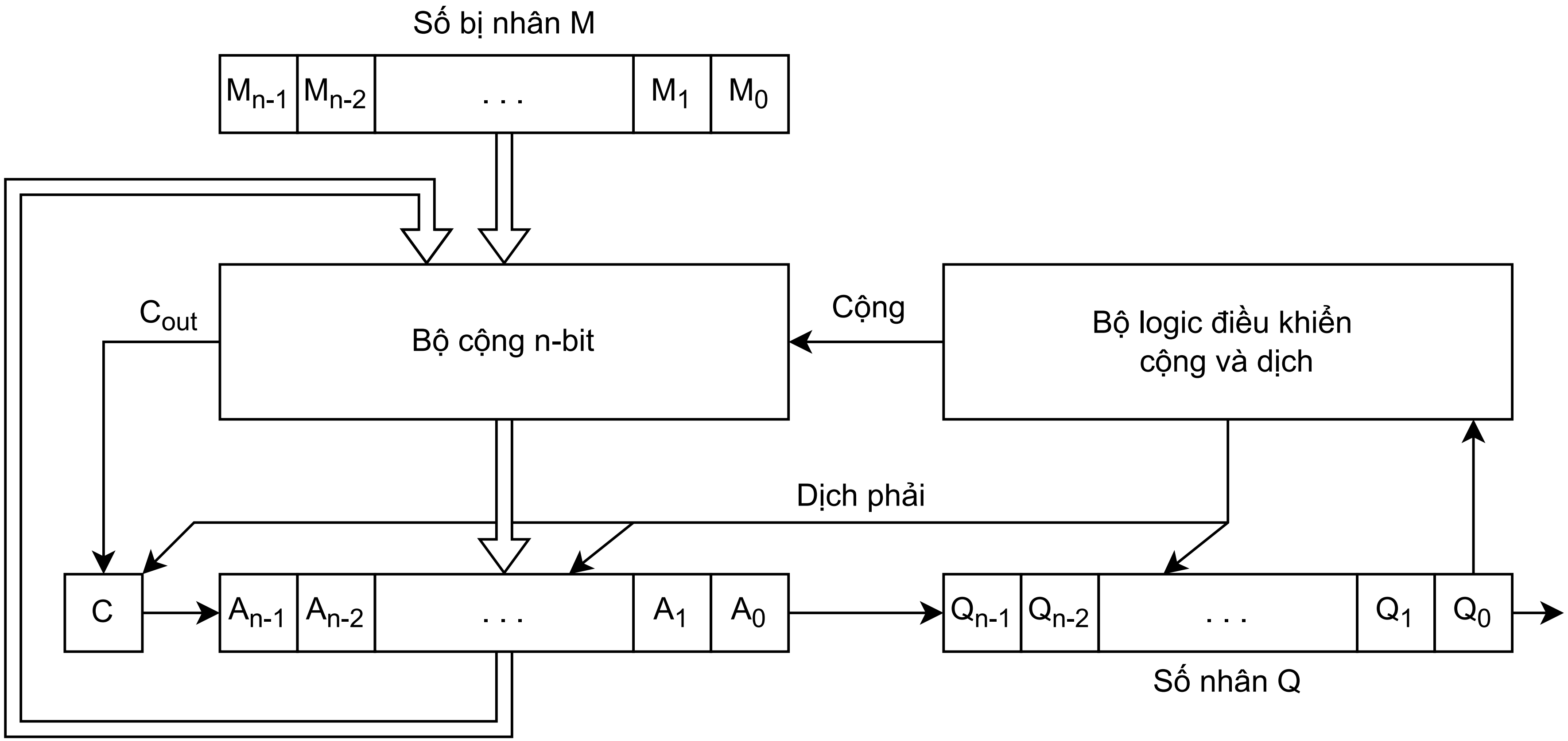
	1011	Số bị nhân (11)
x	1101	Số nhân (13)
	1011	Các tích riêng phần
	0000	
	1011	
	1011	
	10001111	Tích (143)

Phép nhân được thực hiện bằng phép dịch bit và phép cộng

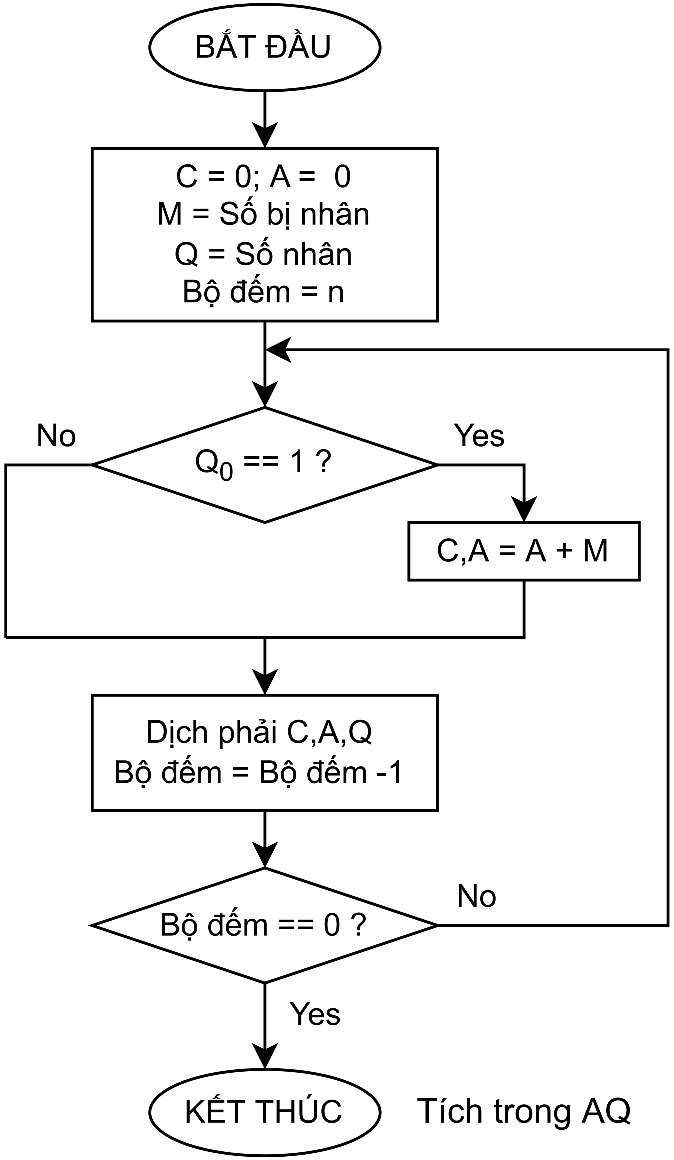
Nhân số nguyên không dấu (tiếp)

- 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

2. 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

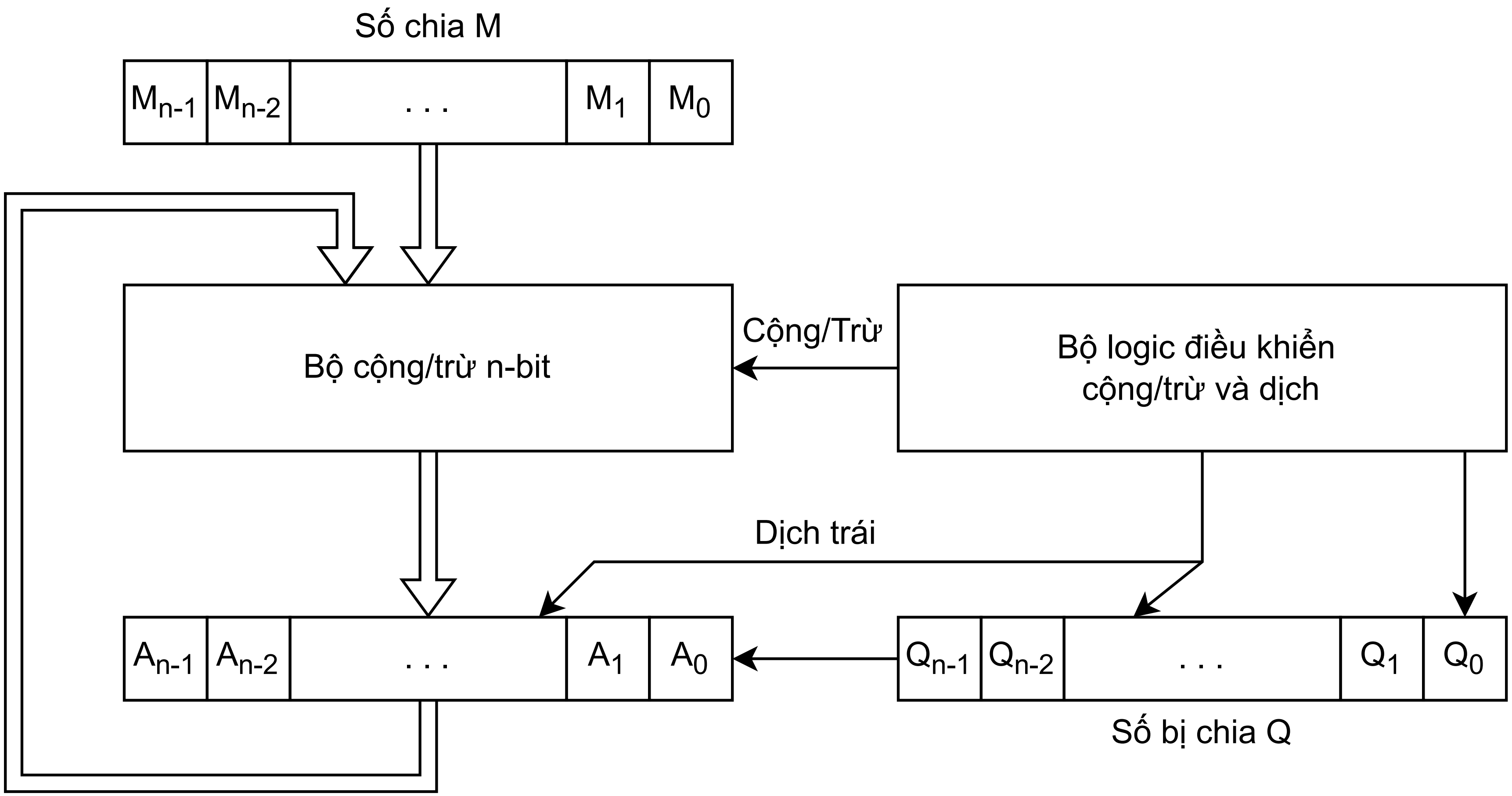
- Bước 1. Chuyển đổi số bị nhân và số nhân thành số dương tương ứng
- Bước 2. Nhân hai số dương bằng thuật giải nhân số nguyên không dấu, được tích của hai số dương.
- Bước 3. Hiệu chỉnh dấu của tích:
  - Nếu hai thừa số ban đầu cùng dấu thì giữ nguyên kết quả ở bước 2
  - Nếu hai thừa số ban đầu là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

3. Chia số nguyên không dấu

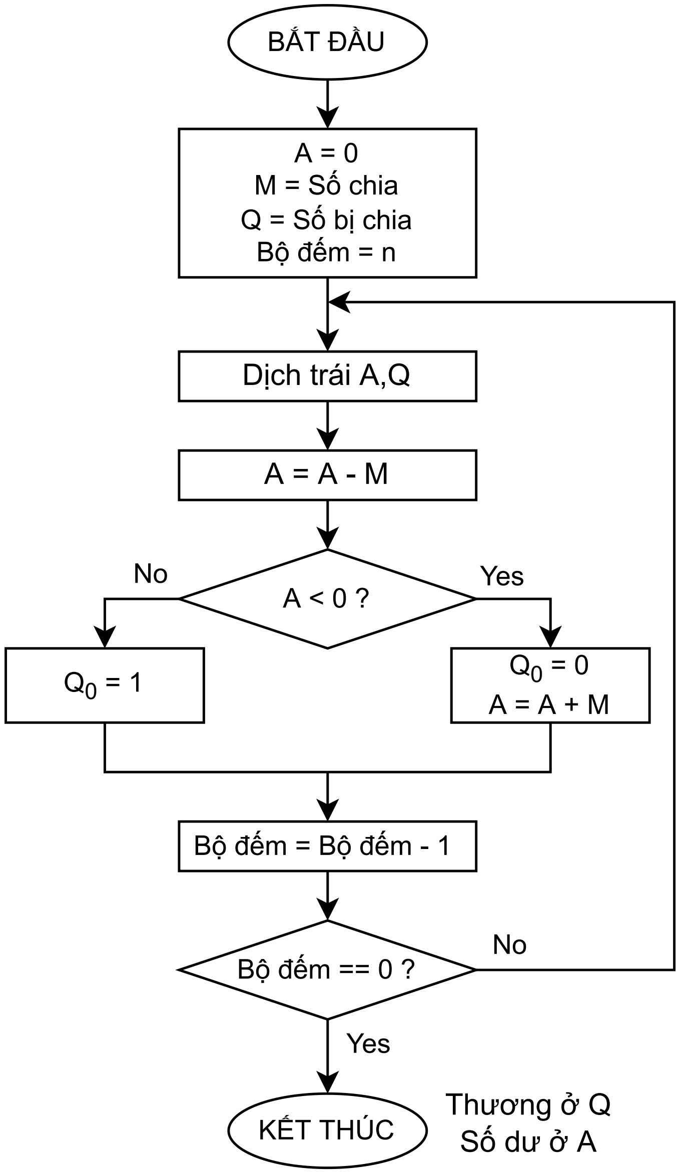
Số bị chia	10010011	1011	Số chia
	- 1011	00001101	Thương
	001110		
	- 1011		
	001111		
	- 1011		
	100		Phần dư

Phép chia được thực hiện bằng phép dịch bit và phép trừ (phép cộng)

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

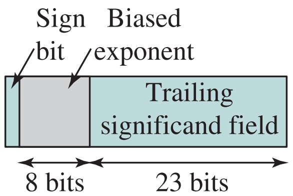
- Floating Point Number → biểu diễn cho số thực
- Tổng quát: một số thực X được biểu diễn theo kiểu số dấu phẩy động như sau:

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

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

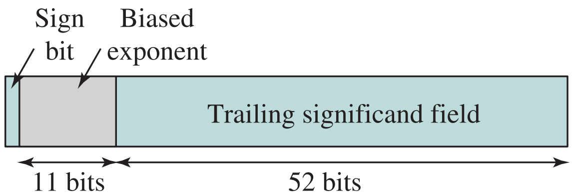
Chuẩn IEEE754-2008

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



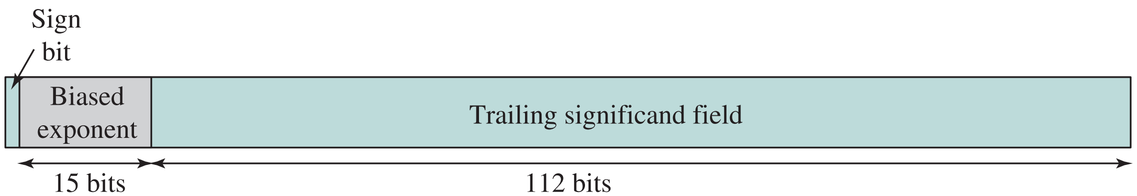
(a) Binary32 format

  - Dạng 64-bit



(b) Binary64 format

  - Dạng 128-bit



(c) Binary128 format

Dạng 32-bit

S	e	m
1 bit	8 bit	23 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+0.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:

1 100 0001 0 101 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 = $83.75_{(10)}$ = $1010011.11_{(2)}$ = 1.01001111×2^6
- Ta có:
  - S = 0 vì đây là số dương
  - e = E + 127 = 6 + 127 = $133_{(10)}$ = $1000\ 0101_{(2)}$
- Vậy:

X = 0100 0010 1010 0111 1000 0000 0000 0000

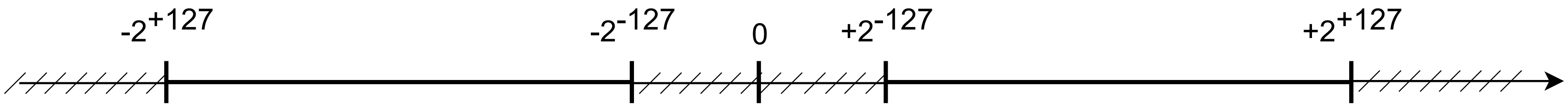
Các qui ước đặc biệt

- 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}

Dạng byte (8-bit) và halfword (16-bit)

S	e	m
1-bit	5-bit	10-bit

S	e	m
1-bit	4-bit	3-bit

- Dạng 8-bit dùng trong các hệ nhúng

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 $E2 \geq E1$

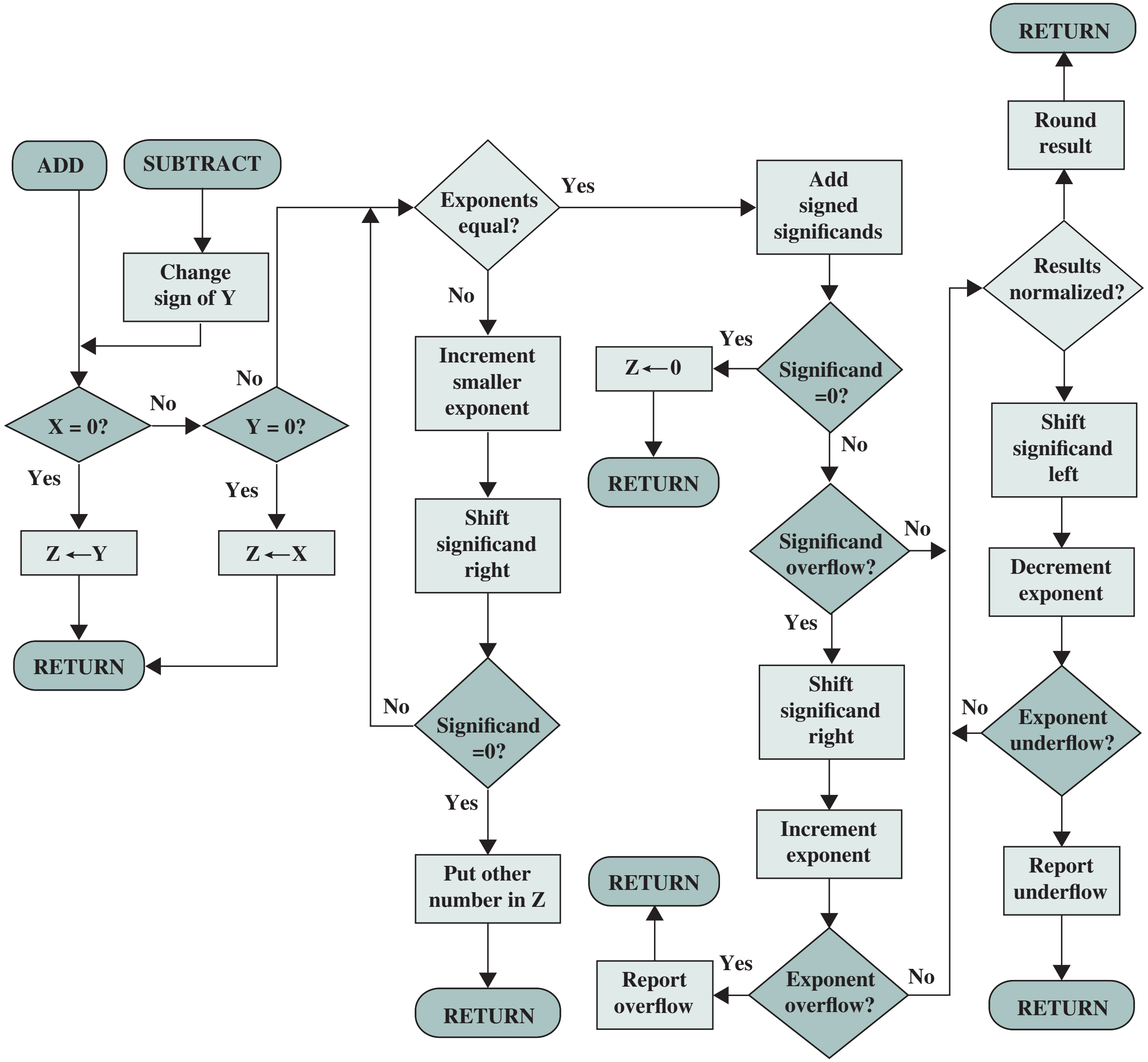
Các khả năng tràn số

- 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

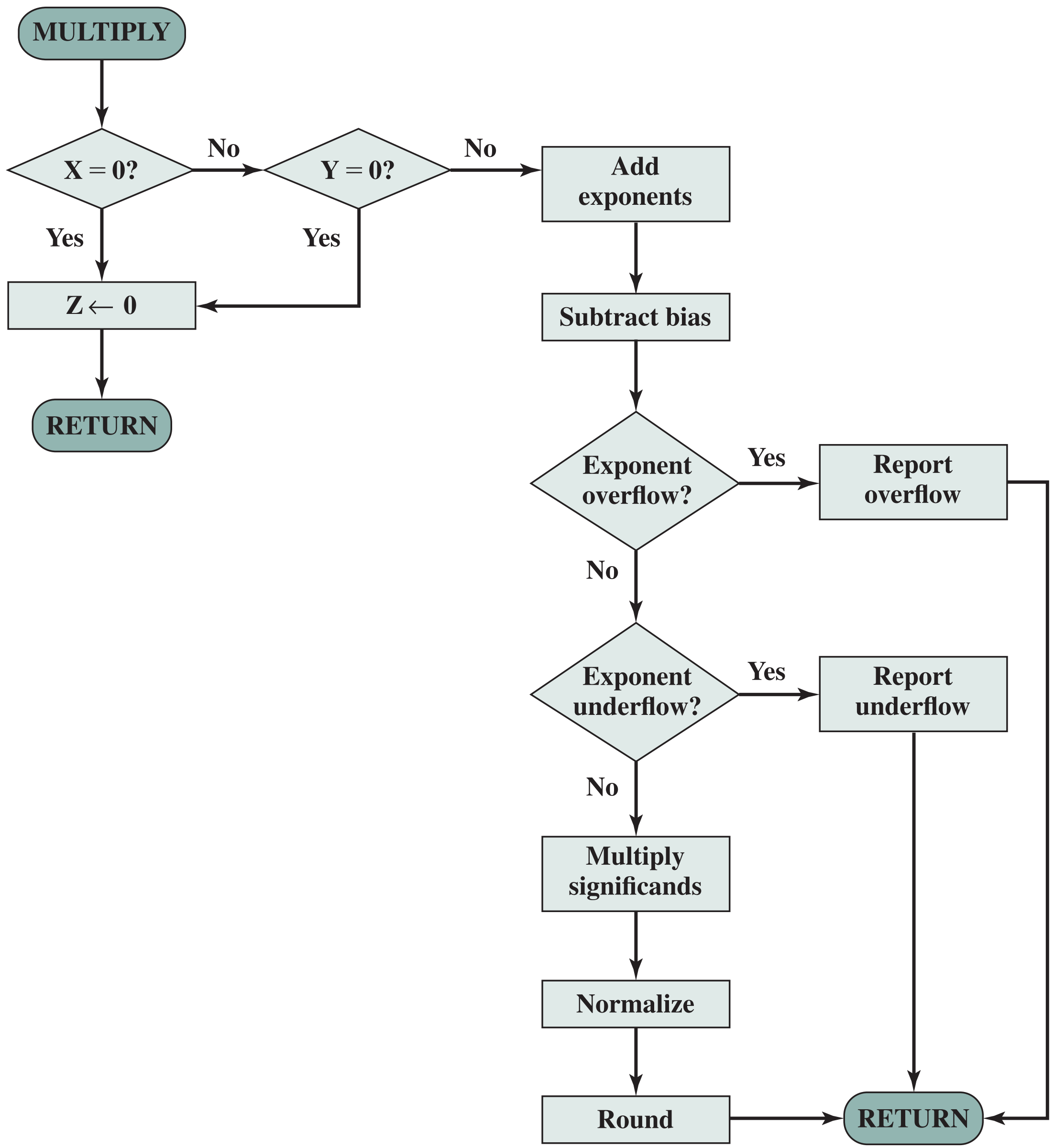
Phép cộng và phép trừ

- 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

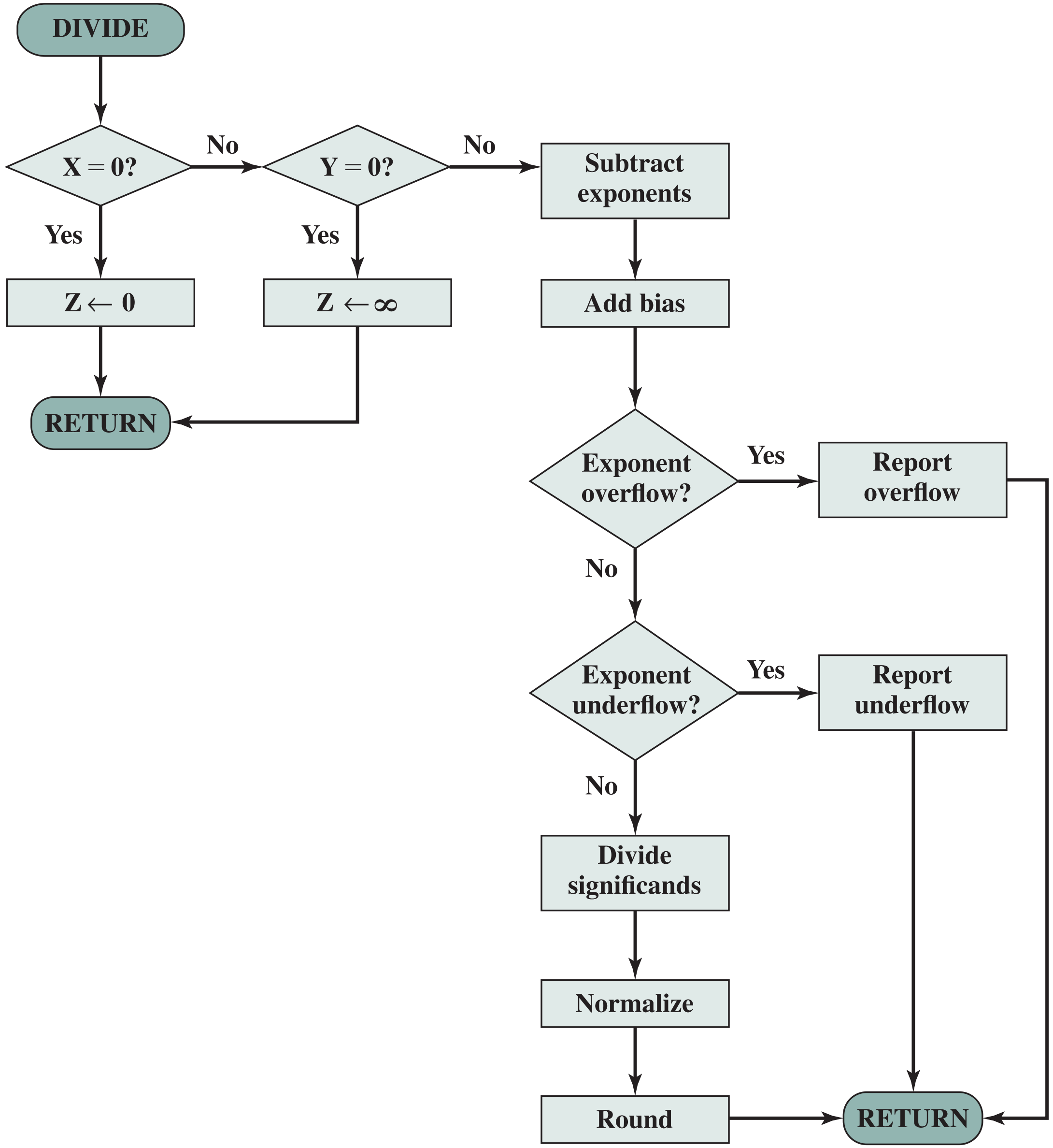
Thuật toán cộng/trừ số dấu phẩy động



Thuật toán nhân số dấu phẩy động



Thuật toán chia số dấu phẩy động



Hết chương 3