IT1110 Tin học đại cương

Phần I: Tin học căn bản

Chương 2: Biểu diễn dữ liệu trong máy tính

Nội dung chương này

2.1. Biểu diễn số trong các hệ đếm

- Hệ đếm là tập hợp các ký hiệu và quy tắc sử dụng các ký hiệu đó để biểu diễn và xác định giá trị số.
- Mỗi hệ đếm có một số chữ số/ký số hữu hạn.
- Số lượng chữ số của mỗi hệ đếm được gọi là **cơ số** (base hay radix), ký hiệu là b.

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
- Hệ bát phân (Octal System)

2.1.1. Hệ đếm cơ số b

- Hệ đếm cơ số b (b≥2 và nguyên dương) mang tính chất sau:
  - có b chữ số để thể hiện giá trị số. Chữ số nhỏ nhất là 0 và lớn nhất là b-1
  - giá trị (trọng số) vị trí thứ n trong một số của hệ đếm bằng cơ số b lũy thừa n: b^n
  - Số dương $N_{(b)}$ trong hệ đếm cơ số b được biểu diễn dưới dạng:

$N_{(b)} = a_n a_{n-1} ... a_0, a_{-1} a_{-2} ... a_{-m}$

2.1.1. Hệ đếm cơ số b

- trong đó, số $N_{(b)}$ có **n+1** chữ số biểu diễn cho phần nguyên và **m** chữ số biểu diễn cho phần sau dấu phẩy, và có thể chuyển đổi qua hệ cơ số 10 như sau:

$$N_{(b)} = M_{(10)} = \sum_{i=-m}^{n} a_i b^i$$

2.1.2. Hệ đếm thập phân (Decimal System, b=10)

- Hệ đếm thập phân hay hệ đếm cơ số 10 là một trong các phát minh của người Ả rập cổ, bao gồm 10 chữ số theo ký hiệu sau: **0, 1, 2, 3, 4, 5, 6, 7, 8, 9**
- Quy tắc tính giá trị của hệ đếm này là mỗi đơn vị ở một hàng bất kỳ có giá trị bằng 10 đơn vị của hàng kế cận bên phải. Ở đây b=10

2.1.2. Hệ đếm thập phân (Decimal System, b=10)

- Số nguyên dương bất kỳ trong hệ thập phân có thể biểu diễn như là một tổng các số hạng, mỗi số hạng là tích của một số với 10 lũy thừa, trong đó số mũ lũy thừa được tăng thêm 1 đơn vị kể từ số mũ lũy thừa phía bên phải nó. Số mũ lũy thừa của hàng đơn vị trong hệ thập phân là 0

2.1.2. Hệ đếm thập phân (Decimal System, b=10)

- Ví dụ: Số 5246 có thể được biểu diễn như sau:

  $5246 = 5x10^3 + 2x10^2 + 4x10^1 + 6x10^0$

  $= 5 \times 1000 + 2 \times 100 + 4 \times 10 + 6 \times 1$
- Thể hiện như trên gọi là ký hiệu mở rộng của số nguyên vì

  $5246 = 5000 + 200 + 40 + 6$

2.1.2. Hệ đếm thập phân (Decimal System, b=10)

- Như vậy, trong số 5246: chữ số 6 trong số nguyên đại diện cho giá trị 6 đơn vị, chữ số 4 đại diện cho giá trị 4 chục (hàng chục), chữ số 2 đại diện cho giá trị 2 trăm (hàng trăm) và chữ số 5 đại diện cho giá trị 5 nghìn (hàng nghìn)
- Số thực:
  - $254.68 = 2x10^2 + 5x10^1 + 4x10^0 + 6x10^{-1} + 8x10^{-2}$

$$= 200+50+4+\frac{6}{10}+\frac{8}{100}$$

2.1.3. Hệ đếm nhị phân (Binary System, b=2)

- Với cơ số b=2, chúng ta có hệ đếm nhị phân. Đây là hệ đếm đơn giản nhất với 2 chữ số là 0 và 1. Mỗi chữ số nhị phân gọi là BIT (viết tắt từ chữ BInary digiT). Ta có thể chuyển đổi số trong hệ nhị phân sang số trong hệ thập phân quen thuộc.

2.1.3. Hệ đếm nhị phân (Binary System, b=2)

- Ví dụ: Số $11101.11_{(2)}$ sẽ tương đương với giá trị thập phân là :

						← vị trí dấu chấm cách	
Số nhị phân :	1	1	1	0	1	. 1	1
Số vị trí :	4	3	2	1	0	-1	-2
Trị vị trí :	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}
Hệ 10 là :	16	8	4	2	1	0.5	0.25

như vậy:

$$\mathbf{11101.11}_{(2)} = 1x16 + 1x8 + 1x4 + 0x2 + 1x1 + 1x0.5 + 1x0.25 = \mathbf{29.75}_{(10)}$$

số 10101 (hệ 2) sang hệ thập phân sẽ là:

$$\mathbf{10101}_{(2)} = 1x2^4 + 0x2^3 + 1x2^2 + 0x2^1 + 1x2^0 = 16 + 0 + 4 + 0 + 1 = \mathbf{21}_{(10)}$$

2.1.4. Hệ đếm bát phân

- Nếu dùng 3 bit thì có thể biểu diễn 8 giá trị khác nhau : 000, 001, 010, 011, 100, 101, 110, 111. Các trị này tương đương với 8 giá trị trong hệ thập phân là 0, 1, 2, 3, 4, 5, 6, 7. Trong hệ bát phân, giá trị vị trí là lũy thừa của 8.
- Ví dụ:

$235.64_{(8)} = \mathbf{2}x8^2 + \mathbf{3}x8^1 + \mathbf{5}x8^0 + \mathbf{6}x8^{-1} + \mathbf{4}x8^{-2} = 157.8125_{(10)}$

2.1.5. Hệ đếm thập lục phân (Hexa-decimal System, b=16)

- Hệ đếm thập lục phân là hệ cơ số b=16, sử dụng 4 bit để biểu diễn 1 chữ số. Khi thể hiện ở dạng hexa-decimal, ta có 16 chữ số gồm 10 chữ số từ 0 đến 9, và 6 chữ in A, B, C, D, E, F để biểu diễn các giá trị số tương ứng là 10, 11, 12, 13, 14, 15. Với hệ thập lục phân, giá trị vị trí là lũy thừa của 16

2.1.5. Hệ đếm thập lục phân (Hexa-decimal System, b=16)

- Ví dụ:

$34F5C_{(16)} = 3x16^4 + 4x16^3 + 15x16^2 + 5x16^1 + 12x16^0 = 216294_{(10)}$

- *Ghi chú*: Một số ngôn ngữ lập trình quy định viết số hexa phải có chữ H ở cuối chữ số. Ví dụ: Số F viết là FH.

2.1.6. Chuyển đổi một số từ hệ thập phân sang hệ cơ số b

- Đổi phần nguyên từ hệ thập phân sang hệ cơ số b.
  - Lấy số nguyên thập phân $N_{(10)}$ lần lượt chia cho b cho đến khi thương số bằng 0. Kết quả số chuyển đổi $N_{(b)}$ là các số dư trong phép chia viết theo thứ tự ngược lại.
- Đổi phần thập phân từ hệ thập phân sang hệ cơ số b
  - Lấy phần thập phân $N_{(10)}$ lần lượt nhân với b cho đến khi phần thập phân của tích số bằng 0. Kết quả số chuyển đổi $N_{(b)}$ là các số phần nguyên trong phép nhân viết ra theo thứ tự tính toán.

Lưu ý 1: Đổi từ hệ 10 sang hệ 2

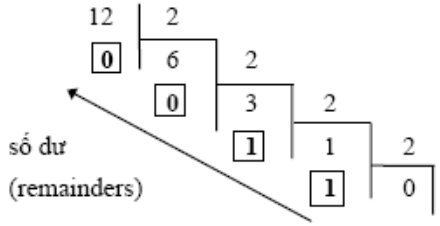
- Chuyển đổi phần nguyên và phần lẻ riêng
- Chuyển đổi phần nguyên: 2 cách
  - Phân tích thành tổng các số lũy thừa của 2
  - Chia cho 2 được thương và số dư, sau đó lại lấy thương chia tiếp cho 2 cho đến khi thương = 0, viết các số dư theo thứ tự ngược lại

Đổi từ hệ 10 sang hệ 2

- Ví dụ:

  $12 = 8 + 4 = 2^3 + 2^2$

  Kết quả: $12_{(10)} = 1100_{(2)}$



Kết quả: $\mathbf{12_{(10)} = 1100_{(2)}}$

Đổi từ hệ 10 sang hệ 2

- Chuyển đổi phần lẻ
  - Lấy phần lẻ nhân 2 rồi lấy phần nguyên,... →biểu diễn các phần nguyên theo chiều thuận
- Ví du:

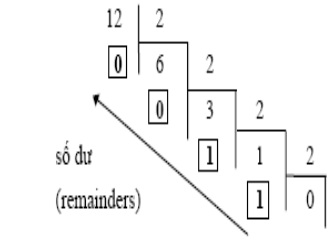
$\mathbf{0.6875_{(10)} = ?_{(2)}}$ phần nguyên của tích

0.6875	x 2 =	1	375	phần thập phân của tích
0.3750	x 2 =	0	.75	
0.75	x 2 =	1	.5	
0.5	x 2 =	1	.0	

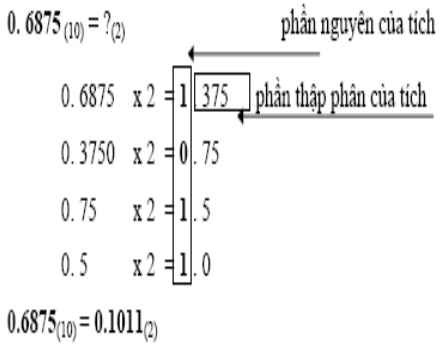
$\mathbf{0.6875_{(10)} = 0.1011_{(2)}}$

Đổi từ hệ 10 sang hệ 2

- $12.6875_{(10)} = 1100.1011_{(2)}$



Kết quả: $\mathbf{12_{(10)} = 1100_{(2)}}$



$\mathbf{0.6875_{(10)} = 0.1011_{(2)}}$

Đổi từ hệ 10 sang hệ 2

- Bài tập: đổi số $35.375_{(10)}$ sang hệ 2

Lưu ý 2: chuyển đổi nhị phân sang Hexa

- Duyệt từ phải sang trái, chia thành các nhóm 4 bit, sau đó thay từng nhóm 4 bit bằng một chữ số Hexa
- Ví dụ:

$10\ 0011_2 = 23_{16}$

2 3

Hệ thập phân	Hệ nhị phân	Hệ mười sáu
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Chuyển đổi thập phân sang Hexa

- Thập phân $\rightarrow$ Hexa: $14988 \rightarrow ?$

14988 : 16 = 936 dư 12 tức là C
936 : 16 = 58 dư 8
58 : 16 = 3 dư 10 tức là A
3 : 16 = 0 dư 3

Như vậy, ta có: $14988_{(10)} = 3A8C_{(16)}$

2.1.7. Mệnh đề logic

- Mệnh đề logic là mệnh đề chỉ nhận một trong 2 giá trị : Đúng (TRUE) hoặc Sai (FALSE), tương đương với TRUE = 1 và FALSE = 0.
- Qui tắc: TRUE = NOT FALSE và FALSE = NOT TRUE
- Phép toán logic áp dụng cho 2 giá trị TRUE và FALSE ứng với tổ hợp AND (và) và OR (hoặc) như sau:

Mệnh đề logic

x	y	X AND y	X OR y
TRUE	TRUE	TRUE	TRUE
TRUE	FALSE	FALSE	TRUE
FALSE	TRUE	FALSE	TRUE
FALSE	FALSE	FALSE	FALSE

Nội dung chương này

2.2. Biểu diễn dữ liệu trong máy tính và đơn vị đo

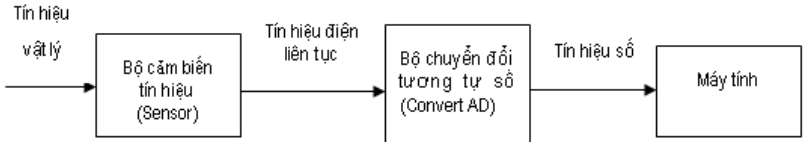
- 2.2.1. Nguyên tắc chung
  - Thông tin và dữ liệu mà con người hiểu được tồn tại dưới nhiều dạng khác nhau, ví dụ như các số liệu, các ký tự văn bản, âm thanh, hình ảnh,... nhưng trong máy tính mọi thông tin và dữ liệu đều được biểu diễn bằng số nhị phân (chuỗi bit).
  - Để đưa dữ liệu vào cho máy tính, cần phải mã hoá nó về dạng nhị phân. Với các kiểu dữ liệu khác nhau cần có cách mã hoá khác nhau. Cụ thể:

Nguyên tắc chung (tiếp)

- Các dữ liệu dạng số (số nguyên hay số thực) sẽ được chuyển đổi trực tiếp thành các chuỗi số nhị phân theo các chuẩn nhất định.
- Các ký tự được mã hoá theo một bộ mã cụ thể, có nghĩa là mỗi ký tự sẽ tương ứng với một chuỗi số nhị phân.
- Các dữ liệu phi số khác như âm thanh, hình ảnh và nhiều đại lượng vật lý khác muốn đưa vào máy phải **số hoá** (*digitalizing*). Có thể hiểu một cách đơn giản khái niệm số hoá như sau: các dữ liệu tự nhiên thường là quá trình biến đổi liên tục, vì vậy để đưa vào máy tính, nó cần được biến đổi sang một dãy hữu hạn các giá trị số (nguyên hay thực) và được biểu diễn dưới dạng nhị phân.

Nguyên tắc chung (tiếp)

- Với các tín hiệu như âm thanh, video, hay các tín hiệu vật lý khác, qui trình mã hoá được biểu diễn như sau:



Nguyên tắc chung (tiếp)

- Tuy rằng mọi dữ liệu trong máy tính đều ở dạng nhị phân, song do bản chất của dữ liệu, người ta thường phân dữ liệu thành 2 dạng:
  - **Dạng cơ bản**: gồm dạng số (nguyên hay thực) và dạng ký tự. Số nguyên không dấu được biểu diễn theo dạng nhị phân thông thường, số nguyên có dấu theo mã bù hai, còn số thực theo dạng dấu phảy động. Để biểu diễn một dữ liệu cơ bản, người ta sử dụng 1 số bit. Các bit này ghép lại với nhau để tạo thành cụm: cụm 8 bít, cụm 16 bít,...
  - **Dạng có cấu trúc**: Trên cơ sở dữ liệu cơ bản, trong máy tính, người ta xây dựng nên các dữ liệu có cấu trúc phục vụ cho các mục đích sử dụng khác nhau. Tuỳ theo cách "ghép" chúng ta có mảng, tập hợp, xâu, bản ghi,...

2.2.2. Đơn vị thông tin

- Đơn vị nhỏ nhất để biểu diễn thông tin gọi là **bit**. Một bit tương ứng với một sự kiện có 1 trong 2 trạng thái.
- Ví dụ: Một mạch đèn có 2 trạng thái là:
  - Tắt (Off) khi mạch điện qua công tắc là hở
  - Mở (On) khi mạch điện qua công tắc là đóng
- Số học nhị phân sử dụng hai ký số 0 và 1 để biểu diễn các số. Vì khả năng sử dụng hai số 0 và 1 là như nhau nên một chỉ thị chỉ gồm một chữ số nhị phân có thể xem như là đơn vị chứa thông tin nhỏ nhất.

Đơn vị dữ liệu (tiếp)

- Bit là chữ viết tắt của **BI**nary digi**T**. Trong tin học, người ta thường sử dụng các đơn vị đo lớn hơn như sau:

Tên gọi	Ký hiệu	Giá trị
Byte	B	8 bit
KiloByte	KB	2^{10} B = 1024 Byte
MegaByte	MB	2^{20} B
GigaByte	GB	2^{30} B
TeraByte	TB	2^{40} B

2.3. Biểu diễn số nguyên

- Số nguyên gồm số nguyên không dấu và số nguyên có dấu.
- Về nguyên tắc đều dùng 1 chuỗi bit để biểu diễn.
- Đối với số nguyên có dấu, người ta sử dụng bit đầu tiên để biểu diễn dấu và bit này gọi là bit dấu.

2.3.1. Biểu diễn số nguyên không dấu

- Dạng tổng quát: giả sử dùng n bit để biểu diễn cho một số nguyên không dấu A:

$a_{n-1}a_{n-2}...a_3a_2a_1a_0$

- Giá trị của A được tính như sau:

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

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

- Dải biểu diễn của A: từ 0 đến $2^n - 1$

Ví dụ:

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

A = 45 B = 156

Giải:

A = 45 = 32 + 8 + 4 + 1 = $2^5 + 2^3 + 2^2 + 2^0$

$\rightarrow$ A = 0010 1101

B = 156 = 128 + 16 + 8 + 4 = $2^7 + 2^4 + 2^3 + 2^2$

$\rightarrow$ B = 1001 1100

Ví dụ (tiếp)

- Cho các số nguyên không dấu X, Y được biểu diễn bằng 8 bit như sau:

X = 0010 1011

Y = 1001 0110

Xác định giá trị của X,Y

Giải:

X = 0010 1011 = $2^5 + 2^3 + 2^1 + 2^0$

= 32 + 8 + 2 + 1 = 43

Y = 1001 0110 = $2^7 + 2^4 + 2^2 + 2^1$

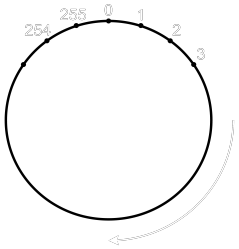
= 128 + 16 + 4 + 2 = 150

Với n = 8 bit

- Dải biểu diễn là [0, 255]

0000 0000 = 0
0000 0001 = 1
0000 0010 = 2
0000 0011 = 3
.....
1111 1111 = 255

- Trục số học máy tính:



Biểu diễn số nguyên không dấu

- Với n = 16 bit:
  - dải biểu diễn: [0, 65535]
- Với n = 32 bit:
  - dải biểu diễn: [0, 2^{32}-1]

2.3.2. Biểu diễn số nguyên có dấu

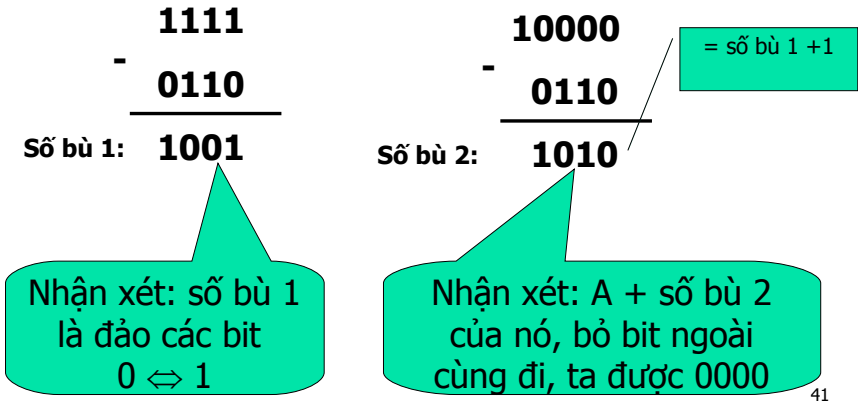
- Khái niệm về số bù
  - Số bù 9 và số bù 10 (hệ thập phân)
    - Giả sử có 1 số nguyên có dấu A được biểu diễn bởi n chữ số thập phân.
    - Số bù 9 của A: $(10^n - 1) - A$
    - Số bù 10 của A: $10^n - A$
    - Số bù 10 = số bù 9 + 1

Biểu diễn số nguyên có dấu

- Số bù 1 và số bù 2 (hệ nhị phân)
  - Giả sử có 1 số nguyên nhị phân A được biểu diễn = n bit nhị phân
  - Số bù 1 của A: $(2^n - 1) - A$
  - Số bù 2 của A: $2^n - A$
  - Số bù 2 = số bù 1 + 1

Số bù 1 và bù 2 (tiếp)

- Ví dụ: n = 4 bit, A = 0110



Biểu diễn số nguyên có dấu bằng số bù 2

- Dùng n bit để biểu diễn số nguyên có dấu: $a_{n-1}a_{n-2}...a_2a_1a_0$
- Với số không âm:
  - bit $a_{n-1} = 0$
  - các bit còn lại biểu diễn độ lớn của số dương đó
  - Dạng tổng quát của số dương: $0a_{n-2}...a_2a_1a_0$
  - Giá trị của số dương:
  - Dải biểu diễn: $[0,2^{n-1}-1]$

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

Biểu diễn số nguyên có dấu bằng số bù 2

- Với số âm: được biểu diễn bằng số bù 2 của số dương tương ứng
  - bit $a_{n-1} = 1$
  - Dạng tổng quát của số âm: $1a_{n-2}...a_2a_1a_0$
  - Giá trị của số âm:

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

  - Dải biểu diễn: $[-2^{n-1}, -1]$

Biểu diễn số nguyên có dấu bằng số bù 2

- Kết hợp lại, ta có dải biểu diễn của số nguyên có dấu n bit là:
  - $[-2^{n-1}, 2^{n-1} - 1]$
- Công thức tổng quát:

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

Một số ví dụ về số nguyên có dấu

- Xác định giá trị của các số nguyên có dấu 8 bit sau đây:

  A = 0101 0110

  B = 1101 0010

Giải:

A = $2^6 + 2^4 + 2^2 + 2^1 = 64 + 16 + 4 + 2 = +86$

B = $-2^7 + 2^6 + 2^4 + 2^1 = -128 + 64 + 16 + 2 = -46$

Bài tập

- Biểu diễn các số nguyên sau với n = 8 bit:
  - X=+58
  - Y=-80
- Xác định giá trị của số nguyên có dấu 8 bit: Z = 1100 1001

Trường hợp cụ thể

- Trường hợp 8 bit: biểu diễn các giá trị từ -128 đến +127

0000 0000 = 0

0000 0001 = +1

.....

0111 1111 = +127

1000 0000 = -128

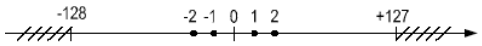
1000 0001 = -127

.....

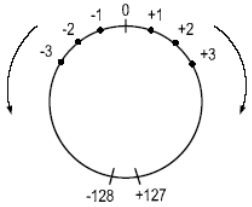
1111 1110 = -2

1111 1111 = -1

Trục số học:



Trục số học máy tính:



Trường hợp cụ thể

- Với n = 16 bit, dải biểu diễn:
  - [-32768, + 32767]
- Với n = 32 bit: -2^{31} đến $2^{31} - 1$
- Với n = 64 bit: -2^{63} đến $2^{63} - 1$
- Chuyển đổi từ byte thành word:
  - đối với số dương thêm 8 bit 0 bên trái

  +19 = 0001 0011 (8 bit)

  +19 = 0000 0000 0001 0011 (16 bit)
  - đối với số âm thêm 8 bit 1 bên trái

  -19 = 1110 1101 (8 bit)

  -19 = 1111 1111 1110 1101 (16 bit)

Binary Code Decimal Code

- Dùng 4 bit để mã hóa từng chữ số thập phân từ 0 đến 9

  0→0000 .....

  1→0001 8→1000

  ..... 9→1001

- Có 6 tổ hợp không dùng: 1010, 1011, 1100, 1101, 1110, 1111

Binary Code Decimal Code

- 35→0011 0101_{BCD}
- 61→0110 0001_{BCD}
- 1087→0001 0000 1000 0111_{BCD}
- Cứ 1 chữ số thập phân đơn lẻ được mã hóa bằng 4 bit

Binary Code Decimal Code

- Phép cộng số BCD:

35 → $0011\ 0101_{BCD}$

+ 61 → $+0110\ 0001_{BCD}$

96 ← $1001\ 0110_{BCD}$

Kết quả đúng, không phải hiệu chỉnh

87 → $1000\ 0111_{BCD}$

+ 96 → $+1001\ 0110_{BCD}$

183 $1\ 0001\ 1101_{BCD}$

Kết quả sai, phải hiệu chỉnh

Binary Code Decimal Code

- Hiệu chỉnh:
  - Nhận xét: 7 + 6 hay 8 + 9 đều vượt 9 nên có nhớ.
  - Hiệu chỉnh bằng cách cộng thêm 6 ở những vị trí có nhớ (>9)

1 0001 1101

+ 0110 0110 ← hiệu chỉnh

$0001\ 1000\ 0011_{BCD}$ → kết quả đúng

Các kiểu lưu trữ số BCD

- BCD không gói (Unpacked BCD): mỗi số BCD 4 bit được lưu trữ trong 4 bit thấp của mỗi byte. Ví dụ: Số 35 được lưu trữ:

	0011		0101

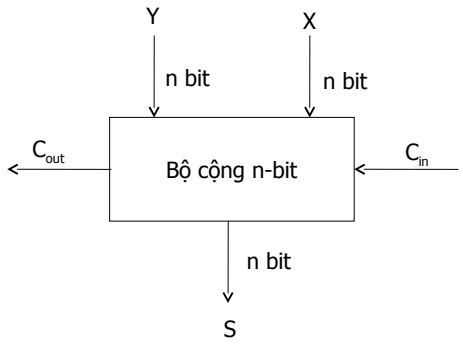
- BCD gói (packed BCD): hai số BCD được lưu trữ trong một byte. Ví dụ: Số 35 được lưu trữ:

0011	0101

Nội dung chương này

2.4. Các phép toán số học với số nguyên

- Phép cộng số nguyên không dấu



2.4. Các phép toán số học với số nguyên

- Phép cộng số nguyên không dấu
  - Tiến hành cộng lần lượt từng bít từ phải qua trái.
  - Khi cộng hai số nguyên không dấu n bits ta thu được một số nguyên không dấu cũng n bits.
  - Nếu tổng của hai số đó lớn hơn 2^{n-1} thì khi đó sẽ tràn số (C_{out} = 1) và kết quả sẽ là sai.
  - Để tránh hiện tượng này, ta dùng nhiều bit hơn

Ví dụ phép cộng số nguyên không dấu

- Với trường hợp 8 bit, nếu tổng nhỏ hơn 255 thì kết quả đúng

```
  57 = 00111001
+ 34 = 00100010
-----------------
  91 = 01011011

 209 = 11010001
+ 73 = 01001001
-----------------
 282 = 100011010
       ↓
Bit tràn ra ngoài      => kết quả = 26 là sai.
```


Phép đảo dấu

- Phép đảo dấu thực chất là lấy bù 2

+37 = 0010 0101
bù 1: 1101 1010
+1
bù 2: 1101 1011 = -37

-37 = 1101 1011
bù 1: 0010 0100
+1
bù 2: 0010 0101 = +37

Cộng hai số nguyên có dấu

Khi cộng 2 số nguyên có dấu n bit, không quan tâm đến bit Cout, và kết quả nhận được là n bit:

- Cộng 2 số khác dấu kết quả luôn đúng
- Cộng 2 số cùng dấu:
  - nếu dấu kết quả cùng dấu với các số hạng thì kết quả là đúng.
  - nếu kết quả có dấu ngược lại, khi đó có tràn xảy ra (Overflow) và kết quả bị sai
- Tràn xảy ra khi tổng nằm ngoài dải biểu diễn

$$[-(2^{n-1}),+(2^{n-1} - 1)]$$

Cộng hai số nguyên có dấu- ví dụ:

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

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

Cộng hai số nguyên có dấu- ví dụ:

(+75) = 0100 1011

+(+82)= 0101 0010

+157 = 1001 1101 = -99

tổng vượt +127 → chuyển sang bên âm

(-104) = 1001 1000 (vì +104 = 0110 1000)

+ (-43) = 1101 0101 (vì +43 = 0010 1011)

-147 = 1 0110 1101 = +109 → sai

không quan tâm

âm + âm → dương

Nguyên tắc thực hiện phép trừ

- Phép trừ hai số nguyên: X-Y = X + (-Y)
- Nguyên tắc: lấy bù 2 của số trừ Y để được –Y, sau đó cộng với số bị trừ X

Nhân số nguyên không dấu

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

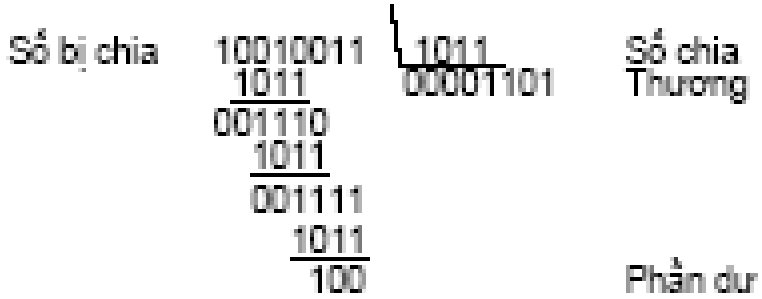

Nhân số nguyên không dấu

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

Nhân hai số nguyên có dấu

- Sử dụng thuật giải nhân hai số nguyên không dấu
- Bước 1: chuyển đổi số bị nhân và số nhân thành số dương tương ứng
- Bước 2: nhân 2 số dương bằng thuật giải đã học, được tích của 2 số dương
- Bước 3: hiệu chỉnh dấu của tích như sau:
  - nếu 2 thừa số ban đầu cùng dấu thì không cần hiệu chỉnh
  - nếu 2 thừa số ban đầu là khác dấu thì ta lấy bù 2 của tích ở kết quả bước 2

Chia số nguyên không dấu



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 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

Nội dung chương này

2.5. Tính toán logic với số nhị phân

		AND	OR	XOR
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

2.5. Tính toán logic với số nhị phân

	NOT
0	1
1	0

2.5. Tính toán logic với số nhị phân

- Thực hiện các phép toán logic với 2 số nhị phân:
  - Kết quả là 1 số nhị phân khi thực hiện các phép toán logic với từng cặp bit của 2 số nhị phân đó
  - Các phép toán này chỉ tác động lên từng cặp bit mà không ảnh hưởng đến bit khác.

2.5. Tính toán logic với số nhị phân

VD: A = 1010 1010 và B = 0000 1111

	AND	OR	XOR	NOT
1010 1010				01010101
0000 1111				11110000
	00001010	10101111	10100101	

Nhận xét: +Phép AND dùng để xoá một số bit và giữ nguyên 1 số bit còn lại.
+Phép OR dùng để thiết lập 1 số bit và giữ nguyên 1 số bit khác.

Nội dung chương này

2.6. Biểu diễn ký tự

- **Nguyên tắc chung:**
  - Các ký tự cũng cần được chuyển đổi thành chuỗi bit nhị phân gọi là **mã ký tự**.
  - Số bit dùng cho mỗi ký tự theo các mã khác nhau là khác nhau.

Vd : Bộ mã ASCII dùng 8 bit cho 1 ký tự.
Bộ mã Unicode dùng 16 bit.

Bộ mã ASCII (American Standard Code for Information Interchange)

- Do ANSI (American National Standard Institute) thiết kế
- http://www.asciitable.com/
- ASCII là bộ mã được dùng để *trao đổi thông tin chuẩn của Mỹ*. Lúc đầu chỉ dùng 7 bit (128 ký tự) sau đó mở rộng cho 8 bit và có thể biểu diễn 256 ký tự khác nhau trong máy tính
- Bộ mã 8 bit → mã hóa được cho $2^8 = 256$ kí tự, có mã từ $00_{16} \div FF_{16}$, bao gồm:
  - 128 kí tự chuẩn có mã từ $00_{16} \div 7F_{16}$
  - 128 kí tự mở rộng có mã từ $80_{16} \div FF_{16}$

ASCII Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2	SPC	!	"	#	$	%	&	'	(	)	*	+	,	–	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Physical Device Controls: Format Effectors

Bộ mã ASCII (tiếp)

- 95 kí tự hiển thị được:có mã từ 20_{16} ÷ $7E_{16}$
  - 26 chữ cái hoa Latin 'A' ÷ 'Z' có mã từ 41_{16} ÷ $5A_{16}$
  - 26 chữ cái thường Latin 'a' ÷ 'z' có mã từ 61_{16} ÷ $7A_{16}$
  - 10 chữ số thập phân '0' ÷ '9' có mã từ 30_{16} ÷ 39_{16}

Bộ mã ASCII (tiếp)

- 95 ký tự hiển thị được:
  - Các dấu câu: . , ? ! : ; …
  - Các dấu phép toán: + - * / …
  - Một số kí tự thông dụng: #, $, &, @, …
  - Dấu cách (mã là 20_{16})
- 33 mã điều khiển: mã từ 00_{16} ÷ $1F_{16}$ và $7F_{16}$ dùng để mã hóa cho các chức năng điều khiển

Điều khiển định dạng

BS	**Backspace – Lùi lại một vị trí. Ký tự điều khiển con trỏ lùi lại một vị trí.**
HT	**Horizontal Tab – Ký tự điều khiển con trỏ dịch đi một khoảng định trước**
LF	**Line Feed – Ký tự điều khiển con trỏ xuống dòng**
VT	**Vertical Tab – Ký tự điều khiển con trỏ dịch đi một số dòng**
FF	**Form Feed – Ký tự điều khiển con trỏ chuyển xuống đầu trang tiếp theo.**
CR	**Carriage Return – Ký tự điều khiển con trỏ về đầu dòng hiện hành.**

Các ký tự mở rộng của bảng mã ASCII

- Được định nghĩa bởi:
  - Nhà chế tạo máy tính
  - Người phát triển phần mềm
- Ví dụ:
  - Bộ mã ký tự mở rộng của IBM: được dùng trên máy tính IBM-PC.
  - Bộ mã ký tự mở rộng của Apple: được dùng trên máy tính Macintosh.
  - Các nhà phát triển phần mềm tiếng Việt cũng đã thay đổi phần này để mã hoá cho các ký tự riêng của chữ Việt, ví dụ như bộ mã TCVN 5712.

Bộ mã Unicode

- Do các hãng máy tính hàng đầu thiết kế
- Là bộ mã 16-bit, Vậy số ký tự có thể biểu diễn (mã hoá) là 2^{16}
- Được thiết kế cho đa ngôn ngữ, trong đó có tiếng Việt

Nội dung chương này

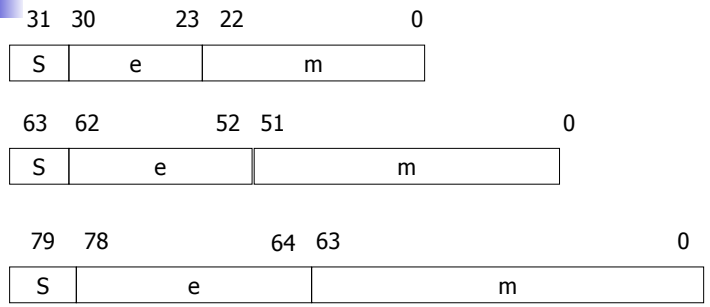
2.7. Biểu diễn số thực

- 2.7.1. Nguyên tắc chung
  - Để biểu diễn số thực, trong máy tính người ta thường dùng ký pháp dấu phẩy động (Floating Point Number).
  - Tổng quát: một số thực X được biểu diễn theo kiểu số dấu phẩy động như sau:
    - $X = M * R^E$
    - M là phần định trị (Mantissa)
    - R là cơ số (Radix)
    - E là phần mũ (Exponent)

2.7.2. Chuẩn IEEE754/85

- Cơ số R = 2
- Các dạng:
  - 32 – bit (4 byte float trong C)
  - 48 – bit (real trong Pascal)
  - 64 – bit (8 byte)
  - 80 – bit (10 byte)

Các dạng biểu diễn chính



- trường S nằm bên trái nhất biểu diễn dấu
- e: mũ
- m: định trị

Dạng 32 – bit

- S là bit dấu
  - S = 0: số dương
  - S = 1: số âm
- e (8 bit) là mã excess – 127 của phần mũ E:
  - E = e – 127
  - khi e = 0 thì phần mũ = -127, khi e = 127 thì phần mũ = 0
  - e_{max} = 255 (8 bit)
  - giá trị 127 gọi là độ lệch (bias)
- m (23 bit) là phần lẻ của phần định trị M: M=1.m
- Công thức xác định giá trị của số thực:

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

Dạng 32 – bit

- Các quy ước đặc biệt
  - Các bit của e = 0, các bit của m = 0 thì X = ±0

  x000 0000 0000 0000 0000 0000 0000 0000 → X = ±0
  - Các bit của e = 1, các bit của m = 0 thì X = ±∞

  x111 1111 1000 0000 0000 0000 0000 0000 → X = ±∞
  - Các bit của e = 1, còn m có ít nhất 1 bit = 1 thì nó không biểu diễn cho số nào cả (NaN – Not A Number)

Dạng 32 – bit

- Dải biểu diễn giá trị
  - 2^{-127} đến 2^{+127}
  - 10^{-38} đến 10^{+38}

Dạng 32 – bit. Ví dụ:

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

  1100 0001 0101 0110 0000 0000 0000 0000
- S = 1 → số âm
- e = 1000 0010_2 = 130 → E = 130 – 127 = 3
- Vậy, X= -1.10101100*2^3 = -1101.011 = -13.375

Dạng 32 – bit. Ví dụ (tiếp):

- 0011 1111 1000 0000 0000 0000 0000 0000
- Kết quả = +1.0

Dạng 64 – bit

- S là bit dấu
- e (11 bit): mã excess-1023 của phần mũ E → 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 80 – bit

- S là bit dấu
- e (15 bit): mã excess-16383 của phần mũ E → E = e – 16383
- m (64 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}

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 ($\rightarrow\infty$)
- Tràn dưới số mũ (Exponent Underflow): mũ âm vượt ra khỏi giá trị cực đại của số mũ âm có thể ($\rightarrow 0$)
- 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 hóa kết quả.

Hỏi - đáp