BIỂU DIỄN DỮ LIỆU VÀ SỐ HỌC MÁY TÍNH

1. Các hệ đếm cơ bản

- 1.1 Hệ thập phân
- Hệ 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

Ví dụ hệ thập phân

$472.38 = 4x10^2 + 7x10^1 + 2x10^0 + 3x10^{-1} + 8x10^{-2}$

- 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

1.2 Hệ nhị phân

- Hệ cơ số 2 (gồm 2 chữ số nhị phân: 0 và 1)
- Chữ số nhị phân 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

Dạng tổng quát số nhị phân

- Một số nhị phân A như sau:

  $A=a_n a_{n-1}\ldots a_1 a_0 . a_{-1}\ldots a_{-m}$

- Giá trị của A được tính như sau:

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

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

Ví dụ số nhị phân

$$1101001.1011_{(2)}$$

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

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

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

$$= 105.6875_{(10)}$$

Chuyển đổi thập phân- nhị phân

- Phương pháp 1: chia dần cho 2 rồi lấy phần dư
- Phương pháp 2: phân tích thành tổng của các số lũy thừa của 2 (2^K)

Phương pháp chia dần cho 2

- 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

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

PP phân tích thành tổng

- Ví dụ 1: chuyển đổi $105_{(10)}$
  - $105 = 64 + 32 + 8 + 1 = 2^6 + 2^5 + 2^3 + 2^0$

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

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

$$17000_{(10)} = \underset{15\ 14\ 13\ 12}{0100}\ \underset{11\ 10\ 9\ 8}{0010}\ \underset{7\ 6\ 5\ 4}{0110}\ \underset{3\ 2\ 1\ 0}{1000}_{(2)}$$

Chuyển đổi phần lẻ thập phân (fractional part) sang nhị phân

- Ví dụ 1: chuyển 0.81_{10}

Kết quả

- 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

- $0.81_{(10)} \approx 0.1100111_{(2)}$

Ví dụ 2: Chuyển $0.6875_{(10)}$

Cách 1:

- 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

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

Ví dụ 2: Chuyển $0.6875_{(10)}$

Cách 2:

Ta có: $0.6875 \times 2^4 = 11_{(10)}$

$\Rightarrow 0.6875 = 11_{(10)} / 2^4$

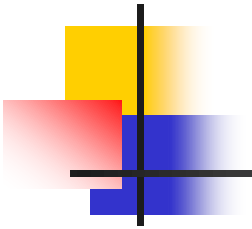
$\Rightarrow 0.6875 = 1011_{(2)} / 2^4$

$\Rightarrow 0.6875 = 0.1011_{(2)}$

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

Bài tập

- **Câu 1:** *Chuyển đổi các số sau sang hệ nhị phân (giữ 5 số lẻ nếu có)*
- 92; 156; 231; 143.
- 21.25; 32.04; 45.625; 69.32
- 34.75; 25.25; 27.1875



- **Câu 2:** *Chuyển các số sau sang hệ thập phân*
- 10101100.0111
- 01011110.01
- 01011100.11001

Bài giải

- $21.25 = 21 + 0.25 = 10101 + 0.01 = 10101.01_{(2)}$
  - Vì: $21 = 10101_{(2)}$
  - $0.25 = 2^{-2} = 0.01_{(2)}$
- $45.625 = 45 + 0.625 = 101101 + 0.101 = 101101.101_{(2)}$
  - $45 = 101101_{(2)}$
  - $0.625 = 0.5 + 0.125 = 2^{-1} + 2^{-3} = 0.101_{(2)}$

1.3 Hệ mười sáu

- Cơ số 16 (0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F)
- Dùng để viết gọn cho số nhị phân
- Cách chuyển đổi:
  - Từ số nhị phân sang số Hex: cứ một nhóm 4 bit (một Nibble) sẽ được thay thế bằng 1 chữ số Hex
  - Từ số Hex sang số nhị phân: cứ 1 chữ số Hex sẽ được thay thế bằng một nhóm 4 bit

1.3 Hệ mười sáu

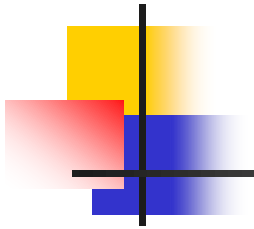
- Cơ số 16 (0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F)
- Cách chuyển đổi:
  - Từ số Hex sang số thập phân:
  - Ví dụ: $1C_{(16)} = 1x16^1 + 12x16^0 = 16 + 12 = 28_{(10)}$

Quan hệ giữa Hex và Nhị phân

4 bit	Chữ số Hex	4 bit	Chữ số Hex
0000	0	1000	8
0001	1	1001	9
0010	2	1010	A
0011	3	1011	B
0100	4	1100	C
0101	5	1101	D
0110	6	1110	E
0111	7	1111	F

Ví dụ 1

- Chuyển số nhị phân sau → số Hex
  - $0000\ 0000_2$ 00
  - $1011\ 0011_2$ B3
  - $0010\ 1101\ 1001\ 1010_2$
  - $1111\ 1111\ 1111\ 1111_2$



- Chuyển số nhị phân sau → số Hex
  - $0000\ 0000_2 = 00_{16}$ (= 0x00)
  - $1011\ 0011_2 = B3_{16}$ (= 0xB3)
  - $0010\ 1101\ 1001\ 1010_2 = 2D9A_{16}$
  - $1111\ 1111\ 1111\ 1111_2 = FFFF_{16}$
- Lưu ý: Ngôn ngữ lập trình C thêm tiền tố "0x" trước số Hex, ví dụ: $2D9A_{16}$ được ghi là 0x2D9A

Ví dụ 2

- Chuyển số Hex sau → số nhị phân
  1) 0xC1
  2) 0xAD03
  3) 0x80B1
  4) 0xC095F046
  5) 0x2E1F40AB
  6) 0x6DC0140C
  7) 0xAC0203CF

Bài tập

- Hãy chuyển các số nhị phân ra số Hex
  a) 100100110111
  b) 0111100001010110
  c) 0101100100110110
  d) 1100 0001 0001 0100 0000 0000 0000 0000
  e) 1100 0001 0011 1000 0000 0000 0000 0000

Bài tập

- Hãy chuyển các số nhị phân ra số Hex
  f) 1100 0001 0010 0001 1111 1001 1010 1100
  g) 0100 0100 0111 1111 1101 1100 0011 1101

Bài tập

- Hãy chuyển các số Hex ra số nhị phân
  1. 0x02488820
  2. 0x8D09FFF4
  3. 0xAD11FFF4
  4. 0xc47fdc3d
  5. 0x42f7207b
  6. 0x4cb300b2
  7. 0xc705ce3b

Các phép tính trong hệ nhị phân

- Được thực hiện tương tự như trong hệ thập phân, tuy nhiên cũng có điểm cần lưu ý
- Phép cộng
  - 0+0=0
  - 0+1=1
  - 1+0=1
  - 1 + 1 = 0 nhớ 1 (đem qua bit cao hơn)

Các phép tính trong hệ nhị phân

- Phép cộng
  - Ví dụ

```
    1 1    ← Số nhớ
    0 1 1
+   1 0 1
  ---------
  1 0 0 0
```

Phép cộng

- Ví dụ:
  1) 1000 1001 + 1001 1000
  2) 1001 1011 + 1101 1011
  3) 1011 1100 + 1001 1101
  4) 1100 1000 + 1001 1111
  5) 1100 1110 + 1011 1011
  6) 1101 1010 + 1111 1001

Các phép tính trong hệ nhị phân

- Phép trừ
- Lưu ý:
  - 0 – 0 = 0
  - 1 – 0 = 1
  - 1 – 1 = 0
  - 0 – 1 = 1 nhớ 1 (mượn bit cao hơn)

(Nhớ 1 (mượn 1): cộng 1 vào bit cao hơn trong số trừ)

Các phép tính trong hệ nhị phân

- Phép trừ
  - Ví dụ

```
      1 1 1        ← Số nhớ
      1 0 0 0
-     0 1 0 1
    -----------
      0 0 1 1
```

Lưu ý: bỏ số nhớ ra ngoài bit cao nhất (nếu có)

Phép trừ

- Ví dụ:
  1) 111000 - 110011
  2) 1100 1100 -10 1110
  3) 1001 1101 - 1011 1100
  4) 1100 1000 - 1001 1111
  5) 1011 1011 - 1100 1110
  6) 1111 0000 1111 – 1100 1111 0011
  7) 1101 1010 - 1111 1001

Các phép tính trong hệ nhị phân

- Phép nhân
- Lưu ý:
  - 0 x 0 = 0
  - 1 x 0 = 0
  - 0 x 1 = 0
  - 1 x 1 = 1

Các phép tính trong hệ nhị phân

- Phép nhân
  - Ví dụ

```
          1 1 0 1
    x       1 0 1
      ----------
          1 1 0 1
        0 0 0 0
      1 1 0 1
    --------------------
    1 0 0 0 0 0 1
```

Các phép tính trong hệ nhị phân

- Phép chia
  - Ví dụ: chia 1001 cho 11

```
 1 0 0 1  |  1 1
-  1 1    |  1 1
-------   |
   1 1
 - 1 1
 -----
   0 0
```

Các phép tính trong hệ nhị phân

- Phép chia
  - Ví dụ 2: chia 1001 cho 10

```
1 0 0 1  |  1 0
-1 0     |  1 0 0 . 1
  0 0
    0 1
      10
    - 10
      --
       0
```

2. Mã hóa dữ liệu trong MT

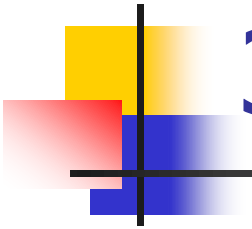
- 2.1 Nguyên tắc chung về mã hóa dữ liệu
- Mọi dữ liệu đưa vào MT đều được mã hóa thành số nhị phân
- Các loại dữ liệu
  - Dữ liệu nhân tạo: do con người quy ước
  - Dữ liệu tự nhiên: tồn tại khách quan với con người

Mã hóa dữ liệu nhân tạo

- Dữ liệu số nguyên: mã hóa theo một số chuẩn quy ước
- Dữ liệu số thực: mã hóa bằng số dấu chấm động
- Dữ liệu ký tự: mã hóa theo bộ mã ký tự

Độ dài từ dữ liệu (word)

- Độ dài từ dữ liệu là số bit được sử dụng để mã hóa loại dữ liệu tương ứng
- Thường bội của 8 bit
- VD: 8, 16, 32, 64



3. Biểu diễn số nguyên

- Có 2 loại số nguyên:
  - Số nguyên không dấu (Unsigned Integer)
  - Số nguyên có dấu (Signed Integer)

3.1 Biểu diễn số nguyên không dấu

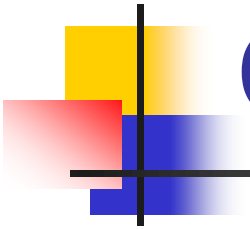
- Nguyên tắc tổng quát: Dùng n bit biểu diễn số nguyên không dấu A:

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

Giá trị của A được tính như sau:

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

Dải biểu diễn của A: từ 0 đến 2^n-1



Các ví dụ

- Ví dụ 1: Biểu diễn các số nguyên không dấu sau đây bằng 8 bit

  A=41; B=150

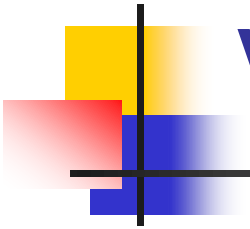
Kết quả ví dụ 1

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

  $41 = 0010\ 1001$

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

  $150 = 1001\ 0110$



Ví dụ 2

- Cho các số nguyên không dấu M, N được biểu diễn bằng 8 bit sau:
- M = 0001 0010
- N = 1011 1001
- Xác định giá trị của chúng

Kết quả

- 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

0000 0000 = 0

0000 0001 = 1

0000 0010 = 2

0000 0011 = 3

...

1111 1111 = 255

Chú ý:

1111 1111

\+ 0000 0001

1 0000 0000

Vậy: 255 + 1 = 0?

→ do tràn nhớ ra n

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

3.2 Biểu diễn số nguyên có dấu

N (binary)	N (unsigned)	N (signed magnitude)	N (2's complement)
000	0	+0	0
001	1	+1	1
010	2	+2	2
011	3	+3	3
100	4	-0	-4
101	5	-1	-3
110	6	-2	-2
111	7	-3	-1

3.2 Biểu diễn số nguyên có dấu

N (binary)	N (unsigned)	N (excess 3) [bias = 3]
000	0	-3 (0 – 3 = -3)
001	1	-2
010	2	-1
011	3	0
100	4	1
101	5	2
110	6	3
111	7	4

3.2 Biểu diễn số nguyên có dấu

- a. Số bù một và số bù hai
- Cách tính: giả sử A là một số nhị phân, ta có:
  - Số bù một của A nhận được bằng cách đảo từng bit của A (0 thành 1 và 1 thành 0).
  - Số bù hai của A = Số bù một A + 1
- Lưu ý: Nếu lấy bù hai của A ta sẽ được số đối của A *[$CP_2(A) = -A$ và $CP_2(CP_2(A)) = A$]*

Ví dụ

- Giả sử có: A = 0010 0101
- Số bù một của A = 1101 1010
- \+ 1
- Số bù hai của A = 1101 1011
- Vì A + $CP_2(A)$ = 0 → $CP_2(A)$ = -A → dùng số bù hai để biểu diễn số nguyên có dấu (đặc biệt là số âm).

Lưu ý: nếu A dương thì bù 2 của A là số âm, nếu A âm thì bù 2 của A là số dương.

b. Biểu diễn số nguyên có dấu bằng số bù hai

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

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

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

Biểu diễn số dương

- Dạng tổng quát của số dương A

$$0a_{n-2}\ldots a_2a_1a_0$$

- Giá trị của số dương A

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

- Dải biểu diễn cho số dương: 0 đến $2^{n-1}-1$

Biểu diễn số âm

- Dạng tổng quát của số âm A:

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

- Giá trị của số âm A

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

- Dải biểu diễn cho số âm -1 đến -2^{n-1}

Biểu diễn tổng quát cho số nguyên có dấu

- Dạng tổng quát của số nguyên A:

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

- Giá trị của A được xác định như sau:

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

- Dải biểu diễn: từ $-(2^{n-1})$ đến $+(2^{n-1}-1)$

Các ví dụ (số nguyên có dấu)

Ví dụ 1: Chuyển sang số nhị phân 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 = CP_2(+80) = CP_2(0101 0000) = 1011 0000

Các ví dụ

Ví dụ 2: Xác định giá trị của các số nguyên có dấu dạng nhị phân sau

- P = 0110 0010
- Q = 1101 1011

Giải:

- P = 64 + 32 + 2 = +98
- Q = 1101 1011 = -128 + 64 + 16 + 8 + 2 + 1 = -37

Các ví dụ

- Q = 1101 1011

Giải:

Cách 2:

- $CP_2(Q)$ = 0010 0100 + 1 = 0010 0101 = 32+5 = 37

=> Q = -37

Phép trừ dùng số bù hai

- Phép trừ 2 số nguyên: X-Y=X+(-Y)
- Nguyên tắc: Lấy bù hai của Y để được –Y, rồi cộng với X (bỏ qua số nhớ ra ngoài bit cao nhất)
- Ví dụ: 9 - 3 = 9 + (-3)
- Ta có:
  - $9 = 01001_{(2)}$
  - $-3 = CP_2(00011) = 11101_{(2)}$

Phép trừ dùng số bù hai

- Ví dụ: 9 - 3 = 9 + (-3)
- 01001 (9)
- \+ 11101 (-3)
- 1 00110 (6)

Bỏ qua

Với n = 8 bit

Biểu diễn được các giá trị từ -128 đến +127

0000 0000	=	0
0000 0001	=	+1
0000 0010	=	+2
0000 0011	=	+3
...		
0111 1111	=	+127
1000 0000	=	- 128
1000 0001	=	- 127
...		
1111 1110	=	-2
1111 1111	=	-1

Chú ý:

+127 + 1 = -128

-128 - 1 = +127

→ do tràn xảy ra

Với n=16 bit, 32 bit, 64 bit

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

Chuyển đổi từ byte thành word

(Sign Extend)

- Đối với số dương:

  +19 = 0001 0011 (8 bit)

  +19 = 0000 0000 0001 0011 (16 bit)

  → thêm 8 bit 0 bên trái

- Đối với số âm:

  - 19 = 1110 1101 (8 bit)

  - 19 = 1111 1111 1110 1101 (16 bit)

  → thêm 8 bit 1 bên trái

Nguyên tắc cộng số nguyên không dấu

Khi cộng hai số nguyên không dấu n-bit, kết quả nhận được là n-bit:

- Nếu không có nhớ ra khỏi bit cao nhất thì kết quả nhận được luôn luôn đúng ($C_{out} = 0$).
- Nếu có nhớ ra khỏi bit cao nhất thì kết quả nhận được là sai, $\Leftrightarrow$ có tràn nhớ ra ngoài ($C_{out} = 1$).
- *Tràn nhớ ra ngoài* (Carry Out) xảy ra khi tổng $>2^n-1$

Ví dụ cộng số nguyên không dấu

- 57 = 0011 1001
  + 34 = + 0010 0010
  91

- 209 = 1101 0001
  + 73 = 0100 1001
  282

Phép đảo dấu (negation)

- 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

Cộng số nguyên có dấu

Khi cộng 2 số nguyên có dấu n-bit không quan tâm đến bit C_{out} và kết quả nhận được là n-bit:

- Cộng 2 số khác dấu: kết quả luôn 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 là sai.
- 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) = $
  $+ (+42) = $
  $+112$
- $(+97) = $
  $+ (-52) = $
  $+45$
- $(-90) = $
  $+ (+36) = $
  -54
- $(-74) = $
  $+ (-30) = $
  -104

Ví dụ cộng số nguyên có dấu bị tràn

- $(+75) =$
  $+\ (+82) =$
  $+157$

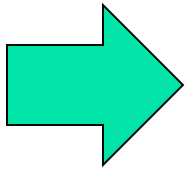
- $(-104) =$
  $+\ (-43) =$
  -147

Bài tập

- Cộng số nguyên không dấu 8 bit và xét tràn nhớ
  - A) 140 + 55
  - B) 170 + 90
- Cộng số nguyên có dấu 8 bit và xét tràn nhớ
  - 1) 100 + 20
  - 2) 80 + (-120)
  - 3) -83 + (-64)
  - 4) 103 + 26

Số dấu chấm động

- (Xem chương 4)
- Bài tập 1: biểu diễn dạng nhị phân với chuẩn IEEE 754 với độ chính xác đơn (32 bit) cho các số thực sau
  - A) 10.25
  - B) -14.125
  - C) 16.625
  - D) -18.0625



Số dấu chấm động

- Bài tập 2: biểu diễn dạng thập phân cho các biểu diễn dạng nhị phân (chuẩn IEEE 754 với độ chính xác đơn 32 bit) sau:
  1) 0 10000011 00111011000000000000000 (0x419d8000)
  2) 1 10000010 11111100000000000000000 (0xc17e0000)

Số dấu chấm động

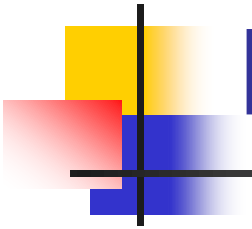
- Bài tập 3: biểu diễn dạng thập phân cho các biểu diễn dạng số Hex (chuẩn IEEE 754 với độ chính xác đơn 32 bit) sau:
  1) 0xc1180000
  2) 0x41a10000
  3) 0xc14b0000
  4) 0x41b08000
  5) 0xc1850000

Số dấu chấm động

- Bài tập 4: biểu diễn dạng thập phân cho các biểu diễn dạng số Hex (chuẩn IEEE 754 với độ chính xác kép 64 bit) sau:
  1) 0x4031B00000000000
  2) 0xC037C00000000000
  3) 0xC03BB00000000000
  4) 0xC059200000000000

Số dấu chấm động

- Bài tập 5: biểu diễn dạng nhị phân với chuẩn IEEE 754 với độ chính xác kép (64 bit) cho các số thực sau
  - A) -27.25
  - B) 106.5
  - C) -56.6875
  - D) 60.625

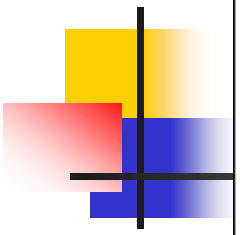


Biểu diễn ký tự

- Bộ mã ASCII (American Standard Code for Information Interchange)
- Bộ mã Unicode

1. Bộ mã ASCII

- Do ANSI (American National Standard Institute) thiết kế
- Bộ mã 8 bit → có thể mã hóa được 2^8 =256 ký tự, có mã từ: $00_{16} \div FF_{16}$ trong đó
- 128 ký tự chuẩn, có mã từ $00_{16} \div 7F_{16}$
- 128 ký tự mở rộng, có mã từ $80_{16} \div FF_{16}$



00	NUL	10	DLE	20	SP	30	0	40	@	50	P	60	`	70	p
01	SOH	11	DC1	21	!	31	1	41	A	51	Q	61	a	71	q
02	STX	12	DC2	22	"	32	2	42	B	52	R	62	b	72	r
03	ETX	13	DC3	23	#	33	3	43	C	53	S	63	c	73	s
04	EOT	14	DC4	24	$	34	4	44	D	54	T	64	d	74	t
05	ENQ	15	NAK	25	%	35	5	45	E	55	U	65	e	75	u
06	ACK	16	SYN	26	&	36	6	46	F	56	V	66	f	76	v
07	BEL	17	ETB	27	'	37	7	47	G	57	W	67	g	77	w
08	BS	18	CAN	28	(	38	8	48	H	58	X	68	h	78	x
09	HT	19	EM	29	)	39	9	49	I	59	Y	69	i	79	y
0A	LF	1A	SUB	2A	*	3A	:	4A	J	5A	Z	6A	j	7A	z
0B	VT	1B	ESC	2B	+	3B	;	4B	K	5B	[	6B	k	7B	{
0C	FF	1C	FS	2C	´	3C	<	4C	L	5C	\	6C	l	7C	|
0D	CR	1D	GS	2D	-	3D	=	4D	M	5D	]	6D	m	7D	}
0E	SO	1E	RS	2E	.	3E	>	4E	N	5E	^	6E	n	7E	~
0F	SI	1F	US	2F	/	3F	?	4F	O	5F	_	6F	o	7F	DEL

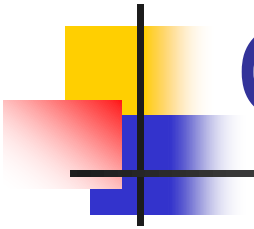
NUL	Null	FF	Form feed	CAN	Cancel
SOH	Start of heading	CR	Carriage return	EM	End of medium
STX	Start of text	SO	Shift out	SUB	Substitute
ETX	End of text	SI	Shift in	ESC	Escape
EOT	End of transmission	DLE	Data link escape	FS	File separator
ENQ	Enquiry	DC1	Device control 1	GS	Group separator
ACK	Acknowledge	DC2	Device control 2	RS	Record separator
BEL	Bell	DC3	Device control 3	US	Unit separator
BS	Backspace	DC4	Device control 4	SP	Space
HT	Horizontal tab	NAK	Negative acknowledge	DEL	Delete
LF	Line feed	SYN	Synchronous idle		
VT	Vertical tab	ETB	End of transmission block		

Các ký tự chuẩn

- 26 chữ cái hoa 'A' đến 'Z' có mã từ 41_{16} đến $5A_{16}$ (65 đến 90)
  - 'A' → 0100 0001 = 41_{16}
  - 'B' → 0100 0010 = 42_{16}
  - ...
  - 'Z' → 0101 1010 = $5A_{16}$
- 26 chữ cái thường 'a' đến 'z' có mã từ 61_{16} đến $7A_{16}$ (97 đến 122)
  - 'a' → 0110 0001 = 61_{16}
  - 'b' → 0110 0010 = 62_{16}
  - ...
  - 'z' → 0111 1010 = $7A_{16}$

Các ký tự chuẩn (tiếp)

- 10 chữ số thập phân từ 0 đến 9 có mã từ 30_{16} đến 39_{16} (48 đến 57)
  - '0' → 0011 0000 = 30_{16}
  - '1' → 0011 0001 = 31_{16}
  - '2' → 0011 0010 = 32_{16}
  - ...
  - '9' → 0011 1001 = 39_{16}



Các ký tự chuẩn

- Các ký hiệu khác
  - Các dấu câu: . , : ; …
  - Các dấu phép toán: + - * / %...
  - Một số ký hiệu thông dụng: $ & @# …
  - Dấu cách (space)

Các mã điều khiển: có mã 00_{16} ÷ $1F_{16}$ và $7F_{16}$

- Các mã ký tự điều khiển định dạng (điều khiển màn hình, máy in,…): BS, HT, LF, VT, FF, CR
- Các mã ký tự điều khiển truyền tin: SOH, STX, ETX, EOT, ENQ, ACK, NAK, SYN, ETB
- Các mã ký tự điều khiển phân cách thông tin: FS, GS, RS, US
- Các mã ký tự điều khiển khác: NUL, BEL, SO, DLE, DC 1+DC4, CAN, EM, SUB, ESC, DEL

2. Bộ mã hợp nhất Unicode

- Do các hãng máy tính hàng đầu thiết kế
- Bộ mã 16 bit
- Bộ mã đa ngôn ngữ
- Có hỗ trợ các ký tự tiếng Việt

0000	NUL	0020	SP	0040	@	0060	`	0080	Ctrl	00A0	NBS	00C0	À	00E0	à
0001	SOH	0021	!	0041	A	0061	a	0081	Ctrl	00A1	¡	00C1	Á	00E1	á
0002	STX	0022	"	0042	B	0062	b	0082	Ctrl	00A2	¢	00C2	Â	00E2	â
0003	ETX	0023	#	0043	C	0063	c	0083	Ctrl	00A3	£	00C3	Ã	00E3	ã
0004	EOT	0024	$	0044	D	0064	d	0084	Ctrl	00A4	¤	00C4	Ä	00E4	ä
0005	ENQ	0025	%	0045	E	0065	e	0085	Ctrl	00A5	¥	00C5	Å	00E5	å
0006	ACK	0026	&	0046	F	0066	f	0086	Ctrl	00A6	¦	00C6	Æ	00E6	æ
0007	BEL	0027	'	0047	G	0067	g	0087	Ctrl	00A7	§	00C7	Ç	00E7	ç
0008	BS	0028	(	0048	H	0068	h	0088	Ctrl	00A8	¨	00C8	È	00E8	è
0009	HT	0029	)	0049	I	0069	i	0089	Ctrl	00A9	©	00C9	É	00E9	é
000A	LF	002A	*	004A	J	006A	j	008A	Ctrl	00AA	ª	00CA	Ê	00EA	ê
000B	VT	002B	+	004B	K	006B	k	008B	Ctrl	00AB	«	00CB	Ë	00EB	ë
000C	FF	002C	´	004C	L	006C	l	008C	Ctrl	00AC	¬	00CC	Ì	00EC	ì
000D	CR	002D	-	004D	M	006D	m	008D	Ctrl	00AD	–	00CD	Í	00ED	í
000E	SO	002E	.	004E	N	006E	n	008E	Ctrl	00AE	®	00CE	Î	00EE	î
000F	SI	002F	/	004F	O	006F	o	008F	Ctrl	00AF	¯	00CF	Ï	00EF	ï
0010	DLE	0030	0	0050	P	0070	p	0090	Ctrl	00B0	°	00D0	Ð	00F0	¶
0011	DC1	0031	1	0051	Q	0071	q	0091	Ctrl	00B1	±	00D1	Ñ	00F1	ñ
0012	DC2	0032	2	0052	R	0072	r	0092	Ctrl	00B2	²	00D2	Ò	00F2	ò
0013	DC3	0033	3	0053	S	0073	s	0093	Ctrl	00B3	³	00D3	Ó	00F3	ó
0014	DC4	0034	4	0054	T	0074	t	0094	Ctrl	00B4	´	00D4	Ô	00F4	ô
0015	NAK	0035	5	0055	U	0075	u	0095	Ctrl	00B5	µ	00D5	Õ	00F5	õ
0016	SYN	0036	6	0056	V	0076	v	0096	Ctrl	00B6	¶	00D6	Ö	00F6	ö
0017	ETB	0037	7	0057	W	0077	w	0097	Ctrl	00B7	·	00D7	×	00F7	÷
0018	CAN	0038	8	0058	X	0078	x	0098	Ctrl	00B8	¸	00D8	Ø	00F8	ø
0019	EM	0039	9	0059	Y	0079	y	0099	Ctrl	00B9	¹	00D9	Ù	00F9	ù
001A	SUB	003A	:	005A	Z	007A	z	009A	Ctrl	00BA	º	00DA	Ú	00FA	ú
001B	ESC	003B	;	005B	[	007B	{	009B	Ctrl	00BB	»	00DB	Û	00FB	û
001C	FS	003C	<	005C	\	007C	|	009C	Ctrl	00BC	1/4	00DC	Ü	00FC	ü
001D	GS	003D	=	005D	]	007D	}	009D	Ctrl	00BD	1/2	00DD	Ý	00FD	Þ
001E	RS	003E	>	005E	^	007E	~	009E	Ctrl	00BE	3/4	00DE	ý	00FE	þ
001F	US	003F	?	005F	_	007F	DEL	009F	Ctrl	00BF	¿	00DF	§	00FF	ÿ

NUL	Null	SOH	Start of heading	CAN	Cancel	SP	Space
STX	Start of text	EOT	End of transmission	EM	End of medium	DEL	Delete
ETX	End of text	DC1	Device control 1	SUB	Substitute	Ctrl	Control
ENQ	Enquiry	DC2	Device control 2	ESC	Escape	FF	Form feed
ACK	Acknowledge	DC3	Device control 3	FS	File separator	CR	Carriage return
BEL	Bell	DC4	Device control 4	GS	Group separator	SO	Shift out
BS	Backspace	NAK	Negative acknowledge	RS	Record separator	SI	Shift in
HT	Horizontal tab	NBS	Non-breaking space	US	Unit separator	DLE	Data link escape
LF	Line feed	ETB	End of transmission block	SYN	Synchronous idle	VT	Vertical tab