Chương 2

Biểu diễn thông tin trong máy tính

Nội dung

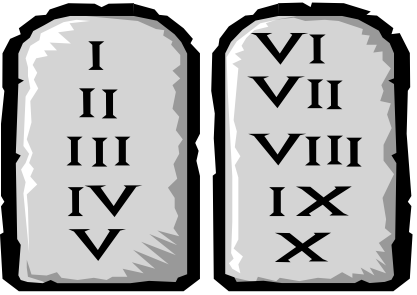
1. Các hệ thống số
2. Biểu diễn số nguyên
3. Biểu diễn số thực
4. Biểu diễn ký tự
5. Biểu diễn các dạng thông tin khác
6. Biểu diễn chương trình

Các hệ thống số

- Hệ thống số theo phép cộng
  - Mỗi ký số có giá trị độc lập không lệ thuộc vị trí của ký số.
  - Giá trị của con số được tính bằng cách cộng/ trừ giá trị từng ký số.
  - Ví dụ 1: Hệ thống số Hy lạp và La mã

1	I	10	Δ	1000	X
2	II	20	ΔΔ	2000	XX
3	III	50	Γ	5000	Γ
4	IIII	80	ΓΔΔΔ	6000	ΓX
5	Γ	100	H	10,000	M
6	ΓI	300	HHH	20,000	MM
7	ΓII	500	Γ	50,000	Γ
8	ΓIII	700	ΓHH	60,000	ΓM
9	ΓIIII				

1	I	20	XX
2	II	25	XXV
3	III	29	XIX
4	IV	50	L
5	V	75	LXXV
6	VI	100	C
10	X	500	D
11	XI	1000	M
16	XVI		

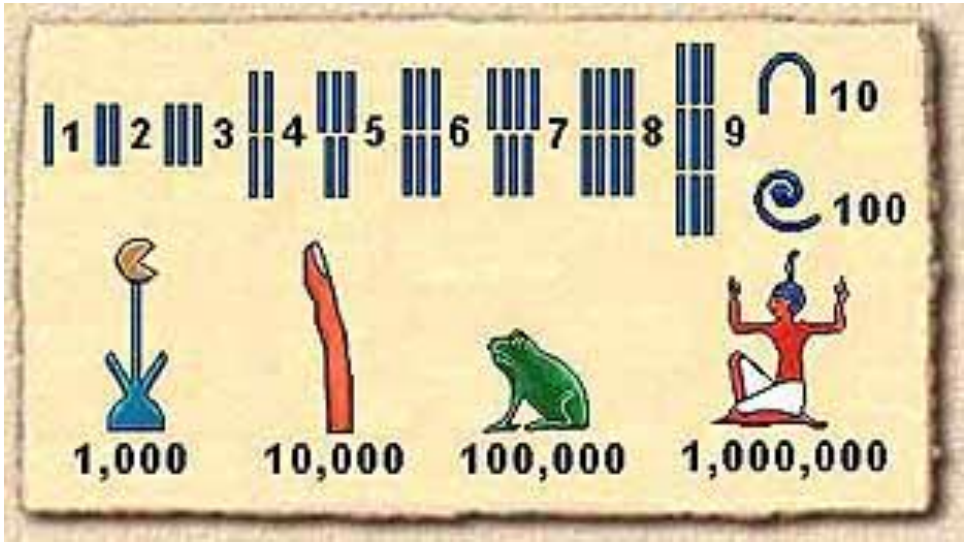


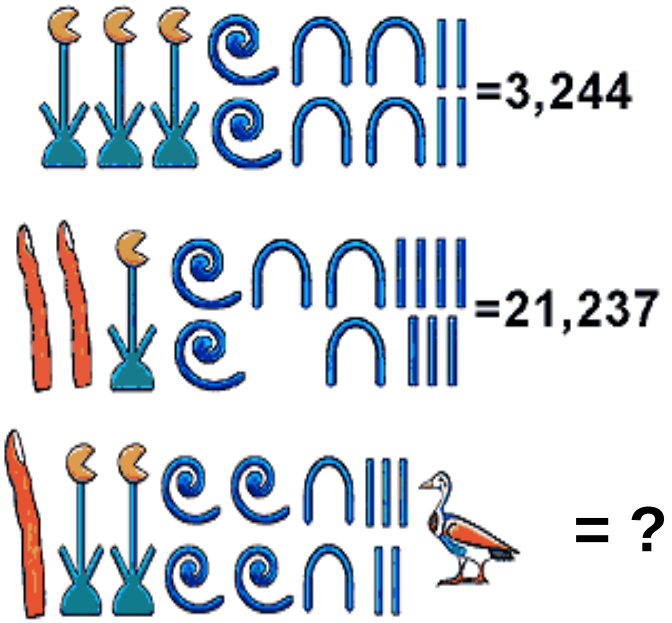
Các hệ thống số

- Ví dụ về số La mã
  1. XXXVI
  2. XL
  3. XVII
  4. DCCLVI
  5. MCMLXIX
- Nhược điểm
  - Khó biểu diễn và tính toán với các số lớn
  - Cần nhiều ký số để biểu diễn các số lớn
  - Không có số không và số âm
  - Không nhất quán về quy tắc. VD số 49 biểu diễn bằng IL (50-1) hay XLIX (40+9)?

Các hệ thống số

- Hệ thống số theo phép cộng (tiếp)
  - Ví dụ 2: Hệ Ai cập cổ đại



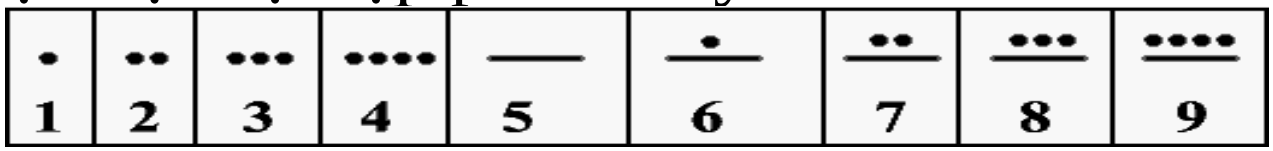


Các hệ thống số

- Hệ thống số theo vị trí
  - Mỗi vị trí số có giá trị khác nhau tùy theo cơ số
  - Ví dụ: Hệ thập phân

Hàng trăm	Hàng chục	Đơn vị
6	3	8

  - Ví dụ: Hệ nhị thập phân Mayan



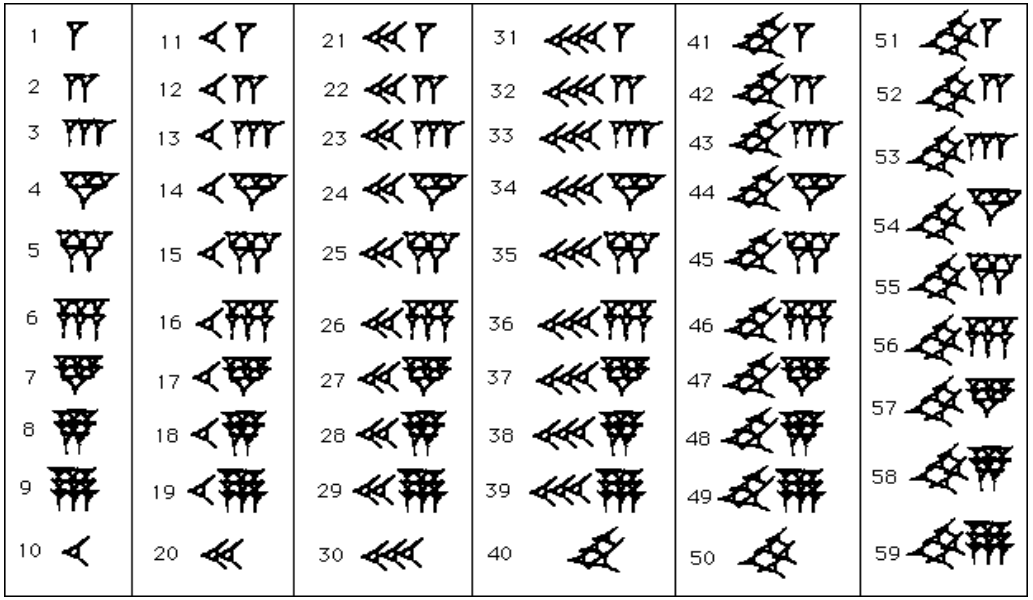
twenties	Units
	
twenties	**units**
	

2 x 20 + 7 = 47

18 x 20 + 5 = 365

Các hệ thống số

- Hệ thống số theo vị trí (tiếp)
  - Ví dụ: Hệ thống lục thập phân Babylon



sixties	units
	

=64

3600s	60s	1s
		

= 3724

Các hệ thống số

- Hệ thống số theo vị trí (tiếp)
  - Tính giá trị số: dựa theo cơ số và bậc lũy thừa theo vị trí số. Dùng n ký số trong hệ cơ số B có thể biểu diễn B^n giá trị khác nhau
  - Ví dụ: hệ thập phân với cơ số B=10
    - $123,45 = 1x10^2 + 2x10^1 + 3x10^0 + 4x10^{-1} + 5x10^{-2}$
  - Tổng quát: Một số ở hệ cơ số B gồm n..-m ký số:

$$a_n a_{n-1} \ldots a_1 a_0 a_{-1} \ldots a_{-(m-1)} a_{-m}$$

Được tính giá trị theo biểu thức:

$$\sum_{i=n}^{-m} a_i . B^i$$

Các hệ thống số

- Hệ thập phân (decimal)
  - Gồm 10 ký số: 0,1,2,3,4,5,6,7,8,9
  - Được sử dụng rộng rãi trong đời sống hàng ngày
  - Không phù hợp với máy tính
- Hệ nhị phân (binary)
  - Gồm 2 ký số: 0 và 1
  - Mỗi ký số được gọi là bit (binary digit), đơn vị thông tin nhỏ nhất
  - Các bội số : Byte (B), KB, MB, GB, TB, PB, EB,…
  - Thích hợp với máy tính
  - Khó sử dụng đối với con người

Các hệ thống số

- Hệ bát phân (octal)
  - Gồm 8 ký số: 0,1,2,3,4,5,6,7
  - Là 1 dạng viết gọn của số nhị phân ($8=2^3$)
  - Sử dụng nhiều trong máy tính và lập trình trước đây
- Hệ thập lục phân (hexa-decimal)
  - Gồm 16 ký số: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E và F
  - Là 1 dạng viết gọn của số nhị phân ($16=2^4$)
  - Hiện đang sử dụng rộng rãi trong máy tính và lập trình

Các hệ thống số

- Đối chiếu giữa các hệ thống số

Thập phân	Nhị phân	Bát phân	Thập lục phân
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Các hệ thống số

- Qui tắc chuyển đổi giữa các hệ thống số
  - Đổi từ số hệ bất kỳ sang hệ thập phân: áp dụng biểu thức

$$\sum_{i=n}^{-m} a_i . B^i$$

  - Ví dụ 1: Đổi số nhị phân $1101001.1011_{(2)}$ sang thập phân

$1101001.1011_{(2)} =$

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

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

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

$= 105.6875_{(10)}$

Các hệ thống số

- Qui tắc chuyển đổi giữa các hệ thống số (tiếp)
  - Ví dụ 2: Đổi các số sau ra thập phân
    - $264.36_{(8)}$
    - $CAFE.85_{(16)}$
  - Đổi từ hệ thập phân sang hệ bất kỳ:
    - Qui tắc 1: Đổi phần nguyên riêng và đổi phần thập phân (lẻ) riêng sau đó ghép lại
    - Qui tắc 2: Đổi số nguyên hệ thập phân sang hệ B bằng cách chia liên tiếp số cần đổi cho B và giữ lại số dư cho đến khi thương số bằng không. Số cần tìm là các số dư viết theo chiều ngược lại

Các hệ thống số

- Qui tắc chuyển đổi giữa các hệ thống số (tiếp)
  - Đổi từ hệ thập phân sang hệ bất kỳ:
    - Qui tắc 3: Đổi phần thập phân sang số hệ B bằng cách nhân liên tiếp phần thập phân cho B và giữ lại phần nguyên cho đến khi tích số bằng 0 (hoặc đã đủ độ chính xác). Số cần tìm là các ký số nguyên viết theo chiều thuận
  - Ví dụ đổi $105.6875_{(10)}$ ra số nhị phân
    - Đổi phần nguyên 105 ra nhị phân
    - Đổi phần thập phân 0.6875 ra nhị phân
    - Ghép kết quả lại

Các hệ thống số

- Qui tắc chuyển đổi giữa các hệ thống số (tiếp)
  - Ví dụ
    - Đổi $105_{(10)}$ ra nhị phân

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

Các hệ thống số

- Qui tắc chuyển đổi giữa các hệ thống số (tiếp)
  - Ví dụ
    - Đổi $0.6875_{(10)}$ ra nhị phân

      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 : $0.6875_{(10)} = 0.1011_{(2)}$
  - Ghép lại: $105.6875_{(10)} = 1101001.1011_{(2)}$

Các hệ thống số

- Qui tắc chuyển đổi giữa các hệ thống số (tiếp)
  - Đổi từ hệ nhị phân sang hệ bát phân và ngược lại
    - Qui tắc: ghép 3 ký số nhị phân đổi ra 1 ký số bát phân (hoặc ngược lại).
  - Đổi từ hệ nhị phân sang hệ thập lục phân và ngược lại
    - Qui tắc: ghép 4 ký số nhị phân đổi ra 1 ký số thập lục phân (hoặc ngược lại).
  - Ví dụ:
    - $B3_{(16)} = 1011\ 0011_{(2)}$
    - $10\ 110\ 011_{(2)} = 263_{(8)}$
  - Đổi giữa bát phân sang thập lục phân và ngược lại
    - Qui tắc: Đổi sang 1 hệ trung gian (thường là nhị phân như ví dụ trên)

Biểu diễn số nguyên

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

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

  - Giá trị của A được tính như biểu thức sau:

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

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

Biểu diễn số nguyên

- Số nguyên không dấu (tiếp)
  - Ví dụ 1. Biểu diễn các số nguyên không dấu sau đây bằng 8-bit: A = 41 ; B = 150
  - Giải:
    - $A = 41 = 32 + 8 + 1 = 2^5 + 2^3 + 2^0$

      $41 = 0010\ 1001$
    - $B = 150 = 128 + 16 + 4 + 2 = 2^7 + 2^4 + 2^2 + 2^1$

      $150 = 1001\ 0110$

Biểu diễn số nguyên

- Số nguyên không dấu (tiếp)
  - 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 = $2^7 + 2^5 + 2^4 + 2^3 + 2^0$

        $= 128 + 32 + 16 + 8 + 1 = 185$

Biểu diễn số nguyên

- Số nguyên không dấu (tiếp)
  - Với n = 8 bit

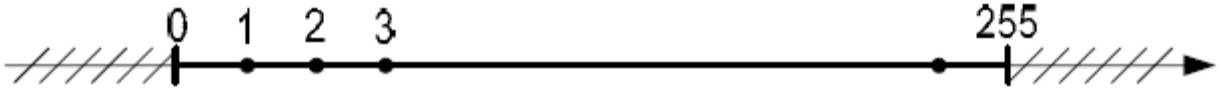
    0000 0000 = 0
    0000 0001 = 1
    0000 0010 = 2
    0000 0011 = 3
    ...
    1111 1111 = 255 Biểu diễn được các giá trị từ 0 đến 255
  - Chú ý:

    1111 1111
    + 0000 0001
    1 0000 0000 Vậy: 255 + 1 = 0 ?

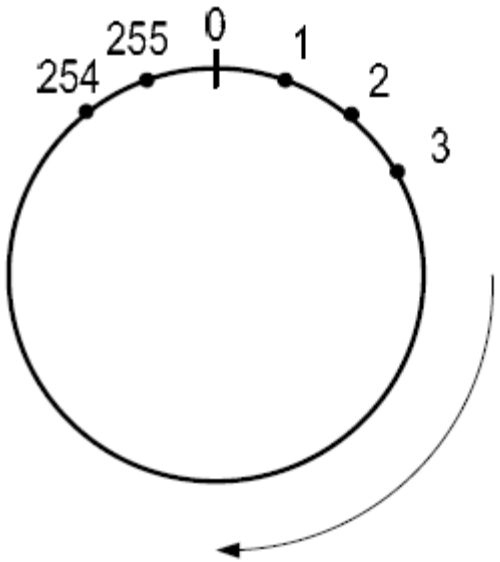
    ➔ do tràn nhớ ra ngoài

Biểu diễn số nguyên

- Số nguyên không dấu (tiếp)
  - Trục số học với n = 8 bit



Trục số học máy tính:



  - Với n=16 bit
    - Dải biểu diễn

      0 đến 65.535 ($0 – 2^{16}-1$)
  - Với n=32 bit : 0 đến $2^{32}-1$
  - Với n=64 bit : 0 đến $2^{64}-1$

Biểu diễn số nguyên

- Số nguyên có dấu
  - Qui tắc 1: Dùng 1 bit cao nhất làm bit dấu, các bit còn lại biểu diễn như số không dấu
    - Bit dấu = 0 : số dương
    - Bit dấu = 1 : số âm
  - Ví dụ số 4 bit
    - 1 bit dấu
    - 3 bit số nguyên
    - Dải biểu diễn -7 … +7

Thập phân	Nhị phân	Thập phân	Nhị phân
+0	0000	-0	1000
+1	0001	-1	1001
+2	0010	-2	1010
+3	0011	-3	1011
+4	0100	-4	1100
+5	0101	-5	1101
+6	0110	-6	1110
+7	0111	-7	1111

Biểu diễn số nguyên

- Số nguyên có dấu (tiếp)
  - Nhược điểm
    - Tồn tại 2 số 0: +0 và -0
    - Kết quả tính tóan không chính xác
    - Ví dụ : tính (+4) + (-2)

      +4 0100

      -2 1010

      +2 1110 (-6) ➔ kết quả ra -6 chứ không phải +2
  - Cách khắc phục: Dùng số bù 2

Biểu diễn số nguyên

- Số nguyên có dấu (tiếp)
  - Qui tắc 2: Dùng số bù 2 (two's-complement) để biểu diễn số âm.
    - [Số bù 2] = [Số bù 1] + 1
    - Số bù 1 tính bằng cách đảo các bit 1 ➔ 0 và 0 ➔ 1
    - Ví dụ tìm số bù 2 của +2
      - Số +2 : 0010
      - Số bù 1 : 1101 (đảo các bit)
      - Công thêm 1: + 1
      - Số bù 2 : 1110

Biểu diễn số nguyên

- Số nguyên có dấu (tiếp)
  - Ví dụ bảng số 4 bit dùng số bù 2 cho số âm
  - Kết quả:
    - Chỉ còn 1 số +0
    - Bù 2 của bù 2 bằng chính nó
    - Mở rộng thêm số -8
    - Kết quả tính toán chính xác.
    - Ví dụ : tính lại (+4) + (-2)

      +4 0100

      -2 1110

      +2 1 0010 (+2)

      (bỏ qua bit tràn số)

Thập phân	Nhị phân	Thập phân	Nhị phân
+0	0000	-	-
+1	0001	-1	1111
+2	0010	-2	1110
+3	0011	-3	1101
+4	0100	-4	1100
+5	0101	-5	1011
+6	0110	-6	1010
+7	0111	-7	1001
-	-	-8	1000

Biểu diễn số nguyên

- Số nguyên có dấu (tiếp)
  - Ví dụ biểu diễn các số nguyên có dấu sau đây bằng 8-bit:

$$A = +58 \; ; \; B = -80$$

  - Giải:

$A = +58 = 0011\ 1010_{(2)}$

$B = -80$

Ta có: $+ 80 = 0101\ 0000$

Số bù một = $1010\ 1111$

$+ 1$

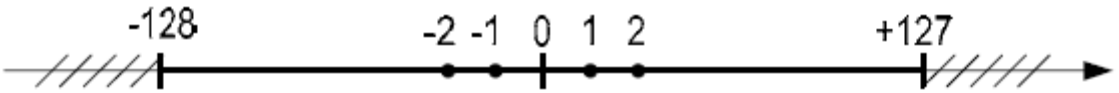
Số bù hai = $1011\ 0000$

Vậy: $B = -80 = 1011\ 0000_{(2)}$

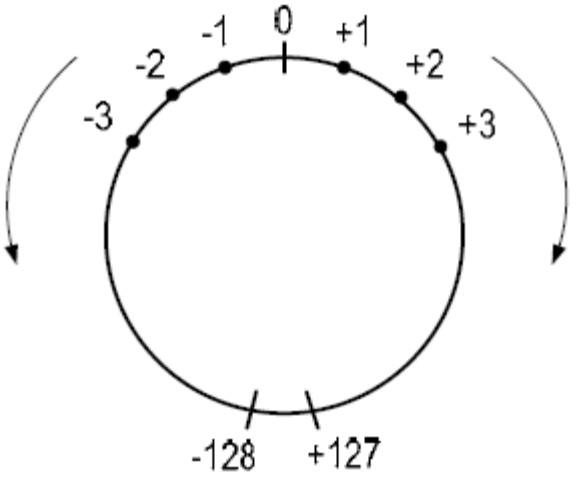
Biểu diễn số nguyên

- Số nguyên có dấu (tiếp)
  - Trục số học số nguyên có dấu với n = 8 bit
    - Dải biểu diễn

      -128 … +127



Trục số học máy tính:



  - Với n=16 bit
    - -32768 … + 32767
  - Với n=32 bit : -2^{-31} … $+2^{31}-1$
  - Với n=64 bit : -2^{-63} … $+2^{63}-1$
  - Nhược điểm: không thể chứa số lớn hơn dải giới hạn (tràn số)

Biểu diễn số nguyên

- Số nguyên có dấu (tiếp)
  - Để tránh việc 2 nửa trục số không liên tục nhau và khó so sánh số nào lớn hơn → cần sắp xếp lại
  - Mã thừa 2^n-1
    - Cộng thêm 2^n-1 vào bảng giá trị theo số bù 2, n=số bit phân số
    - Khắc phục được các nhược điểm trên
    - Không cần biểu diễn dấu cho số âm
  - Ví dụ bảng mã thừa 7 cho trường hợp số 3 bit

Value	Unbiased	Biased
+8	—	1111
+7	0111	1110
+6	0110	1101
+5	0101	1100
+4	0100	1011
+3	0011	1010
+2	0010	1001
+1	0001	1000
0	0000	0111
-1	1111	0110
-2	1110	0101
-3	1101	0100
-4	1100	0011
-5	1011	0010
-6	1010	0001
-7	1001	0000
-8	1000	—

Biểu diễn số nguyên

- So sánh các dạng số nguyên 4 bit

Unsigned representation	Binary pattern	Signed magnitude	Excess-7	1's Complement	2's Complement
0	0000	0	−7	0	0
1	0001	1	−6	1	1
2	0010	2	−5	2	2
3	0011	3	−4	3	3
4	0100	4	−3	4	4
5	0101	5	−2	5	5
6	0110	6	−1	6	6
7	0111	7	0	7	7
8	1000	−0	1	−7	−8
9	1001	−1	2	−6	−7
10	1010	−2	3	−5	−6
11	1011	−3	4	−4	−5
12	1100	−4	5	−3	−4
13	1101	−5	6	−2	−3
14	1110	−6	7	−1	−2
15	1111	−7	8	−0	−1

Biểu diễn số nguyên

- Số BCD (Binary Code Decimal)
  - Số nhị phân có nhược điểm khó biểu diễn chính xác đối với số rất lớn hoặc rất nhỏ.
  - Trong 1 số trường hợp đòi hỏi tính tóan chính xác từng ký số (ví dụng trong tài chánh, ngân hàng,…)
  - Qui tắc : Mã hóa mỗi ký số thập phân 0…9 bằng 1 byte. Chỉ sử dụng 4 bit cuối, 4 bit đầu = 0 hoặc sử dụng cho các mục đích khác.
  - Để tiết kiệm bộ nhớ, có thể ghép 2 ký số vào 1 byte, 4 bit đầu 1 ký số, 4 bit cuối 1 ký sô. Phương pháp này gọi là số BCD dạng dồn (packed-BCD)
  - Áp dụng cho số nguyên hoặc số thực dấu chấm tĩnh.

Biểu diễn số nguyên

- Bảng mã BCD

Ký số	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Ghi chú:

- Mỗi số BCD chính là mã nhị phân của các ký số.
- Có 6 mã không được sử dụng:
  - 1010
  - 1011
  - 1100
  - 1101
  - 1110
  - 1111
- Đơn vị 4 bit (=1/2 byte) được gọi là nibble

Biểu diễn số nguyên

- Số BCD (tiếp)
  - Ví dụ biểu diễn số 35:
    - Unpacked-BCD: 0000 0011 0000 0101 (2 byte)
    - Packed-BCD: 0011 0101 (1 byte)
  - Phép cộng trên 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 → kết quả sai

    + 0110 0110 hiệu chỉnh

    0001 1000 0011$_{BCD}$ → kết quả đúng 1 8 3

    Hiệu chỉnh: cộng thêm 6 ở những vị trí có nhớ (>9)

Biểu diễn số thực

- Số thực dấu chấm tĩnh (fixed-point decimal)
  - Qui tắc: Qui ước 1 vị trí chứa dấu chấm thập phân. Số thực được lưu trữ bằng 2 số nguyên:
    - Số nguyên có dấu cho phần nguyên
    - Số nguyên không dấu cho phần thập phân
  - Ví dụ: Một số thực 16 bit

  - Chọn vị trí dấu chấm sao cho phù hợp độ chính xác cần biểu diễn cho từng thành phần

Biểu diễn số thực

- Số thực dấu chấm tĩnh (tiếp)
  - Nhược điểm: Khó biểu diễn các số quá nhỏ hoặc quá lớn
  - Ví dụ:
    - Số 1,234,000,000,000,000,000.00 cần nhiều bit cho phần nguyên
    - Số 0.000 000 000 000 000 123 456 cần nhiều bit cho phần thập phân
  - Cách khắc phục: Chuyển sang số dạng khoa học (dấu chấm động)

Biểu diễn số thực

- Số thực dấu chấm động (floating-point decimal)
  - Qui tắc : Cho phép thay đổi vị trí dấu chấm thập phân cho phù hợp nhu cầu với độ chính xác vừa phải
  - Ví dụ
    - -123,000,000,000,000,000.00 = -123 x 10^{15} (-123 E+15)
    - +0.000 000 000 000 000 123 = 123 x 10^{-18} (+123 E-18)
  - Số thực được lưu trữ bằng 2 số nguyên
    - Số nguyên có dấu cho phần định trị
    - Số nguyên có dấu cho phần lũy thừa
  - Nhược điểm: Độ chính xác giới hạn

Biểu diễn số thực

- Số thực dấu chấm động (tiếp)
  - Tổng quát: một số thực X được biểu diễn theo kiểu số dấu chấm động như sau:

$$X = M * R^E$$

    - M là phần định trị (Mantissa),
    - R là cơ số (Radix),
    - E là phần mũ (Exponent).
  - Trước đây mỗi hãng sản xuất máy tính tự qui định các thành phần M, R và E riêng biệt dẫn đến khó trao đổi dữ liệu → cần chuẩn hóa

Biểu diễn số thực

- Chuẩn IEEE754
  - Qui định về định dạng và sử dụng số dấu chấm động trong máy tính
  - Do IEEE (Institute of Electrical and Electronic Engineers) ban hành lần đầu 1985, phiên bản mới nhất ban hành 2008
  - Sử dụng cơ số nhị phân (R=2)
  - Các định dạng
    - Chính xác đơn (single precision): 32 bit
    - Chính xác kép (double precision): 64 bit
    - Chính xác mở rộng (extended precision) trên CPU Intel: 80 bit
    - Phiên bản 2008 có thêm định dạng 128 bit

Biểu diễn số thực

- Chuẩn IEEE754 (tiếp)
  - Định dạng số thực theo IEEE754

  - **S** : là bit dấu của phần định trị M:
    - S = 0 : số dương
    - S = 1 : số âm
  - **e** : là mã thừa n (excess-n) của phần mũ E, n là số bit biểu diễn số của E (do đó không cần lưu bit dấu cho E)
    - $e = E + (2^n-1) \rightarrow E = e - (2^n-1)$
  - **m** : là phần lẻ của phần định trị M ở dạng chuẩn:
    - M = 1.m (Chú ý: Không sử dụng số bù 2)
  - Công thức xác định giá trị của số thực:
    - $X = (-1)^S \times 1.m \times 2^E$

Biểu diễn số thực

- Chuẩn IEEE754 (tiếp)

Loại	Số bit của e	Số bit của m	Mã thừa n	Giải biểu diễn	Ký số chính xác
Chính xác đơn 32 bit	**8**	**23**	**127**	**±1 E ±38**	**7**
Chính xác kép 64 bit	**11**	**52**	**1023**	**±1 E ±308**	**16**
Mở rộng 80 bit	**15**	**64**	**16383**	**±1 E ±4932**	**19**

  - Cần chú ý khi sử dụng và so sánh số thực vì độ chính xác bị giới hạn.
  - Ví dụ lưu số 123,456,789,012 bằng số thực 32 bit chỉ bảo đảm chính xác 7 ký số có nghĩa đầu tiên, các ký số còn lại không chính xác

Biểu diễn số thực

- Chuẩn IEEE754 (tiếp)
  - Ví dụ 1: Xác định giá trị của số thực được biểu diễn bằng 32-bit $X=C1560000_{(16)}$:
    - $X=1100\ 0001\ 0101\ 0110\ 0000\ 0000\ 0000\ 0000_{(2)}$
    - S = 1 → số âm
    - $e = 1000\ 0010_{(2)} = 130_{(10)}$ → E = 130-127=3
  - Vậy
    - $X = -1.10101100 * 2^3 = -1101.011_{(2)} = -13.375_{(10)}$

Biểu diễn số thực

- Chuẩn IEEE754 (tiếp)
  - Ví dụ 2: Biểu diễn số thực X= 83.75 về dạng số dấu chấm động IEEE754 dạng 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:
  - $X = 0\underline{100\ 0010\ 1}010\ 0111\ 1000\ 0000\ 0000\ 0000_{(2)}$
  - $X = 42A78000_{(16)}$

Biểu diễn số thực

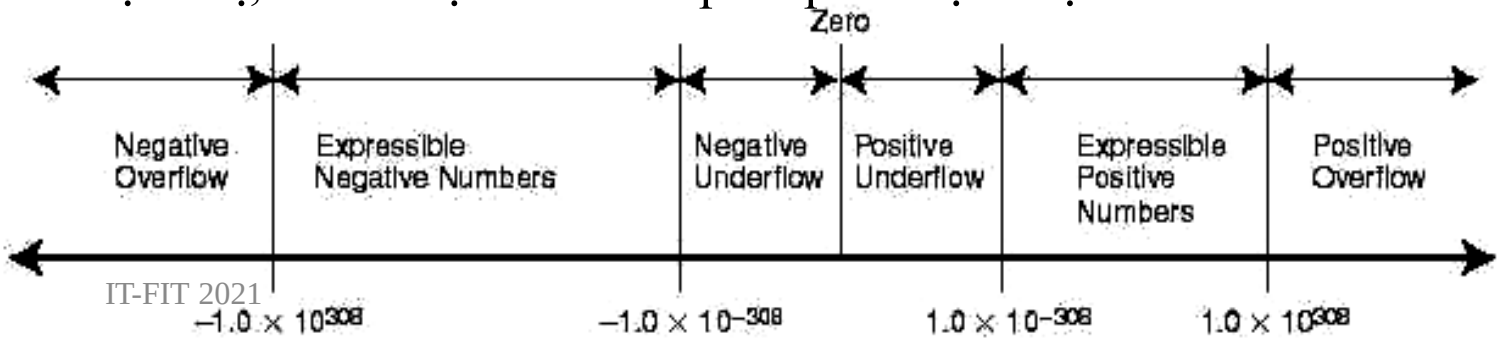
- Chuẩn IEEE754 (tiếp)
  - Các qui ước đặc biệt
    - Các bit của e bằng 0, các bit của m khác 0, thì phần định trị không ở dạng chuẩn: 0.m (dạng chuẩn là 1.m)
    - Các bit của e bằng 0, các bit của m bằng 0, thì X = ± 0
    - Các bit của e bằng 1, các bit của m bằng 0, thì X = ± ∞
    - Các bit của e bằng 1, còn m có chứa ít nhất một bit bằng 1, thì nó không biểu diễn cho số nào cả (NaN - not a number)

Normalized	±	0 < Exp < Max	Any bit pattern
Denormalized	±	0	Any nonzero bit pattern
Zero	±	0	0
Infinity	±	1 1 1…1	0
Not a number	±	1 1 1…1	Any nonzero bit pattern

Sign bit

Biểu diễn số thực

- Chuẩn IEEE754 (tiếp)
  - 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.



Biểu diễn số thực

- Tính toán trên số thực
  - Thực hiện các phép tính phức tạp
  - Đối với phép tính cộng & trừ
    - Kiểm tra = zéro ?
    - Hiệu chỉnh phần số mũ
    - Cộng hoặc trừ phần định trị
    - Chuẩn hoá kết quả
  - Đối với phép tính nhân & chia
    - Kiểm tra = zéro ?
    - Cộng hoặc trừ phần số mũ
    - Nhân hoặc chia phần định trị, xác định dấu kết quả
    - Chuẩn hoá và làm tròn

Biểu diễn số thực

- Đơn vị đo tốc độ tính toán
  - Tốc độ xung nhịp (Hertz)
    - Dựa trên đồng hồ xung nhịp
  - Millions of instructions per second (MIPS)
    - Dựa trên phép tính số nguyên
  - Millions of floating point operations per second (MFLOPS)
    - Dựa trên phép tính số thực
  - Benchmarks
    - Dựa trên các phần mềm đặc trưng thông dụng viết bằng NNLT cấp cao (HLL)
    - Ví dụ: Hệ thống SPEC (System Performance Evaluation Corporation)

Biểu diễn ký tự

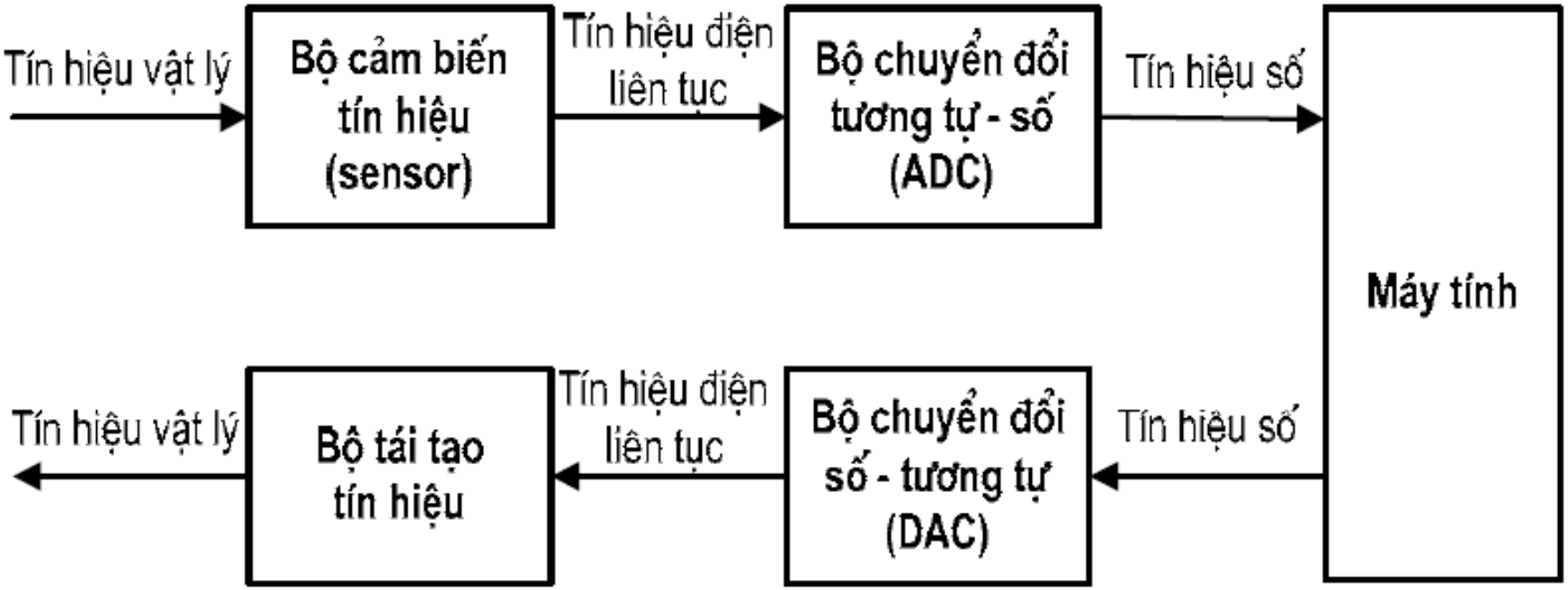
- Khái niệm
  - Thông tin và dữ liệu do con người sử dụng ở dạng văn bản thường được sử dụng nhất
  - Văn bản bao gồm các lọai ký tự:
    - Chữ thường : a…z
    - Chữ hoa : A…Z
    - Số : 0…9
    - Các dấu : +, -, @, #, &, [,],…
  - Ngòai ra còn cần biểu diễn các ký tự điều khiển khi trao đổi thông tin với thiết bị ngọai vi : xuống dòng, sang trang, xóa, …

Biểu diễn ký tự

- Nguyên tắc chung về mã hóa dữ liệu
  - Mọi dữ liệu đưa vào máy tính đều phải được mã hóa thành số nhị phân, ánh xạ 1-1 và có chiều dài bit bằng nhau.
  - Các loại dữ liệu
    - Dữ liệu nhân tạo: do con người qui ước trong các bộ mã chuẩn riêng biệt
    - Dữ liệu tự nhiên: tồn tại khách quan với con người, thường ở dạng Analog ➔ cần chuyển đổi ra dạng Digital
  - Độ dài từ dữ liệu là số bit được sử dụng để mã hóa loại dữ liệu tương ứng
    - Thường là bội của 8-bit
    - VD: 8, 16, 32, 64 bit
  - Các phần mềm điều khiển nhập/ xuất TBNV sẽ đảm nhiệm việc mã hóa

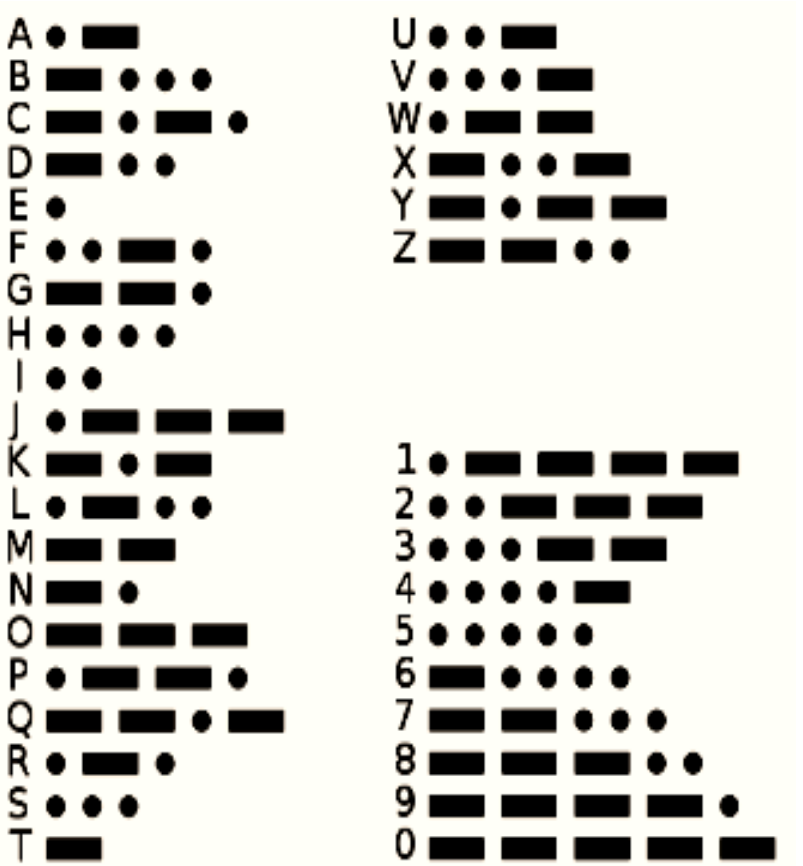
Biểu diễn ký tự

- Qui tắc chuyển đổi tín hiệu vật lý dạng analog sang dạng digital (ví dụ: âm thanh, hình ảnh, video,…)



Biểu diễn ký tự

- Các bộ mã cổ điển
  - Mã Morse:
    - Dùng các dấu chấm và gạch, chiều dài mã khác nhau
    - Sử dụng trong điện tín và truyền dữ liệu trước đây
    - Không phân biệt chữ thường và hoa, thiếu các ký hiệu và ký tự điều khiển → không phù hợp mã hóa dữ liệu trong máy tính



Biểu diễn ký tự

- Các bộ mã cổ điển (tiếp)
  - Mã Baudot
    - Dùng 5 bit mã hóa các ký tự, dấu, số và một số ký tự điều khiển. Do chỉ mã hóa được 32 ký tự nên phải ghép chung 2 bộ mã để mã hoá đủ các ký tự cần thiết. Phân biệt 2 mã bằng ký tự 1B/1F → 1 mã được biểu diễn cho 2 ký tự
    - Được sử dụng nhiều trong điện báo và telex trước đây

00	01	02	03	04	05	06	07
NUL	E 3	LF	A -	SP	S '	I 8	U 7
08	09	0A	0B	0C	0D	0E	0F
CR	D ENQ	R 4	J BEL	N ,	F !	C :	K (
10	11	12	13	14	15	16	17
T 5	Z +	L)	W 2	H £	Y 6	P 0	Q 1
18	19	1A	1B	1C	1D	1E	1F
O 9	B ?	G &	FIGS	M .	X /	V ;	LTRS
Letters			Figures			Control Chars.	

Bài tập: Tìm hiểu thêm về Semaphore

Biểu diễn ký tự

- Mã hóa ký tự trong máy tính
  - Ban đầu mỗi công ty sản xuất đưa ra 1 bộ mã theo qui ước riêng → không trao đổi được thông tin giữa các lọai máy tính
  - VD:

Cty 1		Cty 2	
A	1	A	10
B	2	B	11
C	3 ...	C	12 ...

  - Bộ mã EBCDIC của IBM được sử dụng rộng rãi nhất
  - Cần có bộ mã chuẩn thống nhất cho mọi máy tính và mọi quốc gia: Mã ASCII và Unicode

Biểu diễn ký tự

- Mã hóa ký tự trong máy tính (tiếp)
  - Mã EBCDIC (Extended BCD Interchange Code) do công ty IBM ban hành 1963 để sử dụng cho hệ thống IBM/360 và sau đó được áp dụng cho nhiều hệ thống khác
  - Được mở rộng từ mã BCD
  - Sử dụng 8 bit có thể mã hóa tối đa 256 ký tự (thực tế EBCDIC không sử dụng hết)
    - 0-63: Các ký tự điều khiển, không in được
    - 64-127: Dấu
    - 129-169: Chữ thường
    - 193-233: Chữ hoa
    - 240-249: Số

Biểu diễn ký tự

- Bảng mã EBCDIC

Zone \ Digit	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0000	NUL	SOH	STX	ETX	PF	HT	LC	DEL		RLF	SMM	VT	FF	CR	SR	SI
0001	DLE	DC1	DC2	TM	RES	NL	BS	IL	CAN	EM	CC	CU1	IFS	IGS	IRS	IUS
0010	DS	SOS	FS		BYP	LF	ETB	ESC			SM	CU2		ENQ	ACK	BEL
0011			SYN		PN	RS	UC	EOT				CU3	DC4	NAK		SUB
0100	SP										[	.	<	(	+	!
0101	&										]	$	*	)	;	^
0110	–	/									|	,	%	_	>	?
0111										`	:	#	@	'	=	"
1000		a	b	c	d	e	f	g	h	i						
1001		j	k	l	m	n	o	p	q	r						
1010		~	s	t	u	v	w	x	y	z						
1011																
1100	{	A	B	C	D	E	F	G	H	I						
1101	}	J	K	L	M	N	O	P	Q	R						
1110	\		S	T	U	V	W	X	Y	Z						
1111	0	1	2	3	4	5	6	7	8	9						

Biểu diễn ký tự

- Mã hóa ký tự trong máy tính (tiếp)
  - Mã ASCII (American Standard Code for Information Interchange) do ANSI ban hành từ 1963
  - Sau này được CCITT (ITU) và ISO công nhận và được sử dụng rộng rãi trên thế giới
  - Sử dụng 7 bit để mã hóa được tối đa 128 ký tự. Mỗi ký tự lưu trong 1 byte dữ liệu. Bit thứ 8 sau này được sử dụng làm bit kiểm tra (parity bit) hoặc để mở rộng bộ mã (mã ASCII mở rộng 8 bit).
    - 0-31: Các ký tự điều khiển, không in được
    - 48-57: Số
    - 65-90: Chữ hoa
    - 97-122: Chữ thường
    - 32-48, 58-64, 91-96, 123-127: Dấu (xen kẽ giữa các vùng)

Biểu diễn ký tự

- Bảng mã ASCII

	0	1	2	3	4	5	6	7
0	NUL	DLE	SP	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	“	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	‘	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
a	LF	SUB	*	:	J	Z	j	z
b	VT	ESC	+	;	K	[	k	{
c	FF	FS	,	<	L	\	l	|
d	CR	GS	-	=	M	]	m	}
e	SO	RS	.	>	N	^	n	~
f	SI	US	/	?	O	_	o	DEL

Ghi chú:

Bảng trình bày theo số thập lục phân

Theo số thập phân

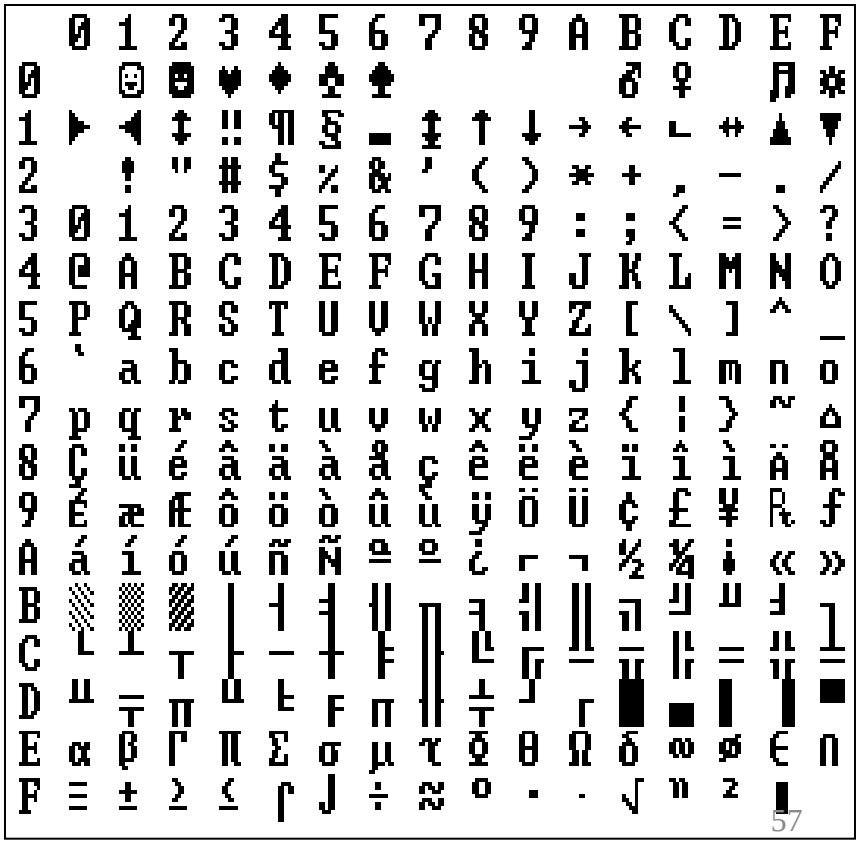
- 0-31: Ký tự điều khiển
- 48-57: Số
- 65-90: Chữ hoa
- 97-122: Chữ thường

Biểu diễn ký tự

- Bảng mã ASCII
  - Các ký tự điều khiển

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

- Bảng ASCII mở rộng IBM-PC



Biểu diễn ký tự

- Mã tiếng Việt có dấu
  - Ban đầu 1 số công ty đưa ra các bộ mã khác nhau nhưng đều mở rộng từ bộ mã ASCII chuẩn 7 bit lên 8 bit: VNI, ABC, ĐHBK, Vietware,…(khỏang 43 bộ mã)
  - Do 128 vị trí mở rộng không đủ chứa các ký tự tiếng Việt có dấu nên mỗi bộ mã áp dụng các cách khắc phục khác nhau nhưng vẫn còn nhiều nhược điểm:
    - Dùng 2 bộ mã dựng sẵn khác nhau cho chữ thường và chữ hoa
    - Dùng 1 bộ mã, một số ký tự còn thiếu sẽ chèn vào vùng ASCII chuẩn
    - Dùng 1 byte ký tự không dấu và 1 byte dấu riêng biệt (mã tổ hợp)
  - Năm 1993 VN ban hành bộ mã 8 bit TCVN 5712 và 1999 chỉnh sửa thêm nhưng vẫn còn nhiều tồn tại nên ít được sử dụng.
  - Năm 2001 VN ban hành bộ mã 16 bit TCVN 6909 phù hợp với chuẩn Unicode và ISO/IEC 10646 khắc phục hầu hết các nhược điểm nên được sử dụng rộng rãi

Biểu diễn ký tự

- Mã Unicode
  - Nhu cầu sử dụng các bộ mã 16-32 bit ngày càng cao để khắc phục các hạn chế của mã 8 bit:
    - Mỗi quốc gia dùng bảng mã ASCII mở rộng riêng biệt (code page)
    - Nhu cầu trình bày văn bản của nhiều thứ tiếng đồng thời, đặc biệt là trên Web
    - Một số ngôn ngữ có lượng ký tự cần mã hóa rất lớn, ví dụ CJK (Chinese, Japanese, Korean)

Ç'është Unicode?, in Albanian
ዩኒኮድ ምንድን ነው? in Amharic
ما هي الشفرة الموحدة "يونِكود" ؟ in Arabic
Ի՞նչ է Յունիկոդը ? in Armenian
ইউনিকোড কী? in Bangla
यूनिकोड क्या है? in Hindi
Какво е Unicode ? in Bulgarian
什麼是Unicode? In Chinese

Что такое Unicode? in Russian
유니코드에 대해? in Korean
Шта је Unicode? in Serbian
యూనీకోడ్ అంటే ఏమిటి?, in Telugu
Unicode คืออะไร? in Thai
ዩኒኮድ እንታይ ኢዩ? in Tigrigna
ئۇنىكود دېگەن نېمە؟in Uyghur
ಯುನಿಕೋಡ್ ಎಂದರೇನು? in Kannada

Biểu diễn ký tự

- Mã Unicode (tiếp)
  - Đặc điểm
    - Ban hành năm 1991, hiện nay đã đến phiên bản 6.2 (09/2012).
    - Unicode cung cấp một mã số duy nhất cho mỗi ký tự, cho mọi hệ máy tính, cho mọi chương trình, cho mọi ngôn ngữ. Hiện nay có thể mã hóa trên 1 triệu ký tự.
    - Chuẩn Unicode đã được những công ty công nghệ hàng đầu, như Apple, HP, IBM, Microsoft, … chấp nhận
    - Unicode tương thích với ISO/IEC 10646 và mã ASCII
    - Hỗ trợ 3 kiểu định dạng UTF-8, UTF-16 và UTF-32
    - Hiện được sử dụng rộng rãi trên tòan cầu, kể cả ở VN

Biểu diễn ký tự

- Mã Unicode (tiếp)
  - Tóm tắt bảng mã Unicode 16 bit

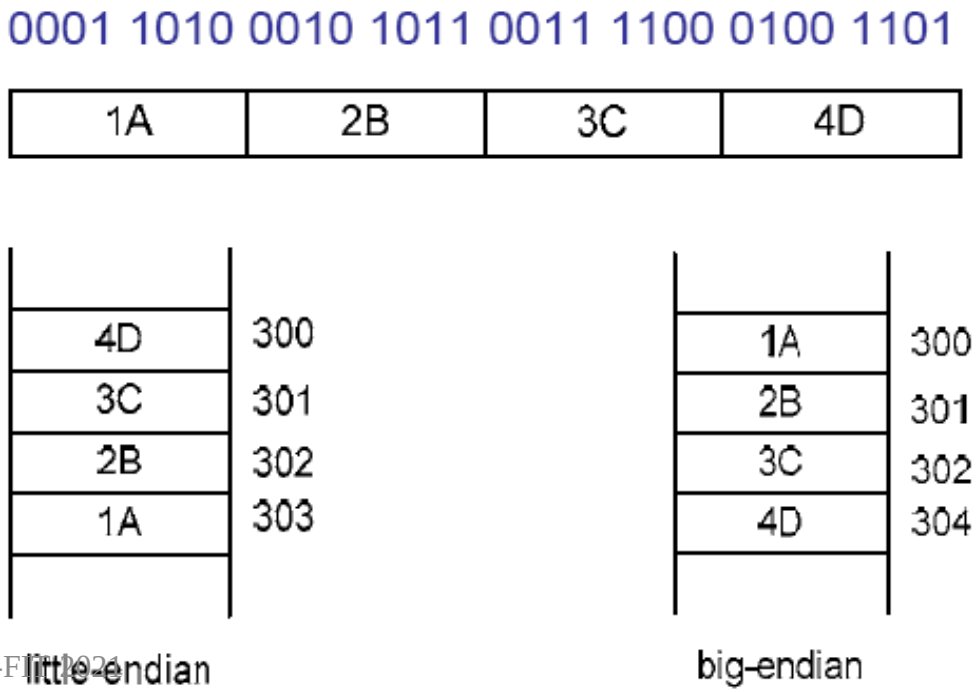
Character Types	Character Set Description	Number of Characters	Hexadecimal Values
Alphabets	Latin, Cyrillic, Greek, etc.	8192	0000 to 1FFF
Symbols	Dingbats, Mathematical, etc.	4096	2000 to 2FFF
CJK	Chinese, Japanese, and Korean phonetic symbols and punctuation	4096	3000 to 3FFF
Han	Unified Chinese, Japanese, and Korean	40,960	4000 to DFFF
	Expansion or spillover from Han	4096	E000 to EFFF
User defined		4095	F000 to FFFE

Biểu diễn ký tự

- Thứ tự lưu trữ các byte trong bộ nhớ chính
  - Bộ nhớ chính thường tổ chức theo byte
  - Hai cách lưu trữ dữ liệu nhiều byte:
    - Đầu nhỏ (*Little-endian): Byte có ý nghĩa* thấp được lưu trữ ở ngăn nhớ có địa chỉ nhỏ, byte có ý nghĩa cao được lưu trữ ở ngăn nhớ có địa chỉ lớn.
    - Đầu to (*Big-endian): Byte có ý nghĩa cao* được lưu trữ ở ngăn nhớ có địa chỉ nhỏ, byte có ý nghĩa thấp được lưu trữ ở ngăn nhớ có địa chỉ lớn.
  - Áp dụng: Mã Unicode, số, chuỗi ký tự

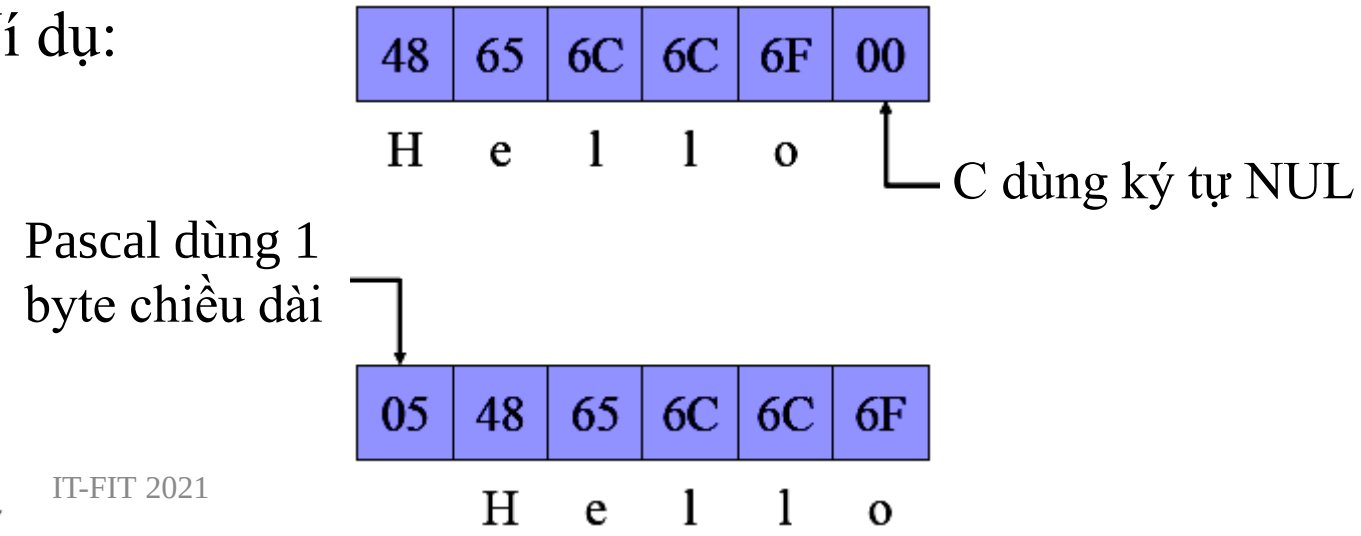
Biểu diễn ký tự

- Thứ tự lưu trữ các byte trong bộ nhớ chính (tiếp)
  - Ví dụ lưu trữ dữ liệu 32-bit
    - Intel 80x86 và các Pentium: little-endian
    - Motorola 680x0, SunSPARC: big-endian



Biểu diễn ký tự

- Lưu trữ chuỗi ký tự
  - Chuỗi ký tự gồm nhiều ký tự ghép lại, mỗi ký tự chiếm 1 byte bộ nhớ nếu là mã ASCII (2 byte nếu là Unicode)
  - Cần xác định chiều dài chuỗi (số ký tự có trong chuỗi)
  - Mỗi ngôn ngữ lập trình cấp cao qui định cách xác định khác nhau cho chuỗi ký tự khi lưu trữ.
  - Ví dụ:



Biểu diễn các dạng thông tin khác

- Các chuẩn định dạng thông tin thông dụng:
  - Hình ảnh: BMP, TIFF, PNG, GIF, JPEG,…
  - Âm thanh: WAV, MIDI, MP3, AVI,…
  - Văn bản: PDF, HTML, XML,…
  - Video: MPEG-4, WMV, DivX,…
  - Animation: Flash, SVG, CSS, …
  - Khác: Mã vạch, RFID, OCR

Biểu diễn chương trình

- Tập lệnh CPU cũng phải được mã hóa bằng số nhị phân → Chương trình ngôn ngữ máy ở dạng số nhị phân
- Hiện nay mỗi công ty sản xuất máy tính qui định bộ mã lệnh riêng cho CPU của mình sản xuất → Chương trình viết cho máy này không thể chạy trên máy khác vì khác mã lệnh

Câu hỏi: tại sao không đưa ra chuẩn thống nhất mã lệnh cho mọi loại CPU?

Câu hỏi