Chương 2

Biểu diễn thông tin trong máy tính

1. Hệ số đếm và biểu diễn số đếm
2. Mã hóa và biểu diễn dữ liệu máy tính
3. Số học nhị phân

Giảng viên: ThS. Phan Như Minh

Hệ cơ số q tổng quát

- Tổng quát số nguyên có n chữ số thuộc hệ cơ số q bất kỳ được biểu diễn:

$$x_{n-1}...x_1x_0 = x_{n-1}.q^{n-1} + ... + x_1.q^1 + x_0.q^0$$

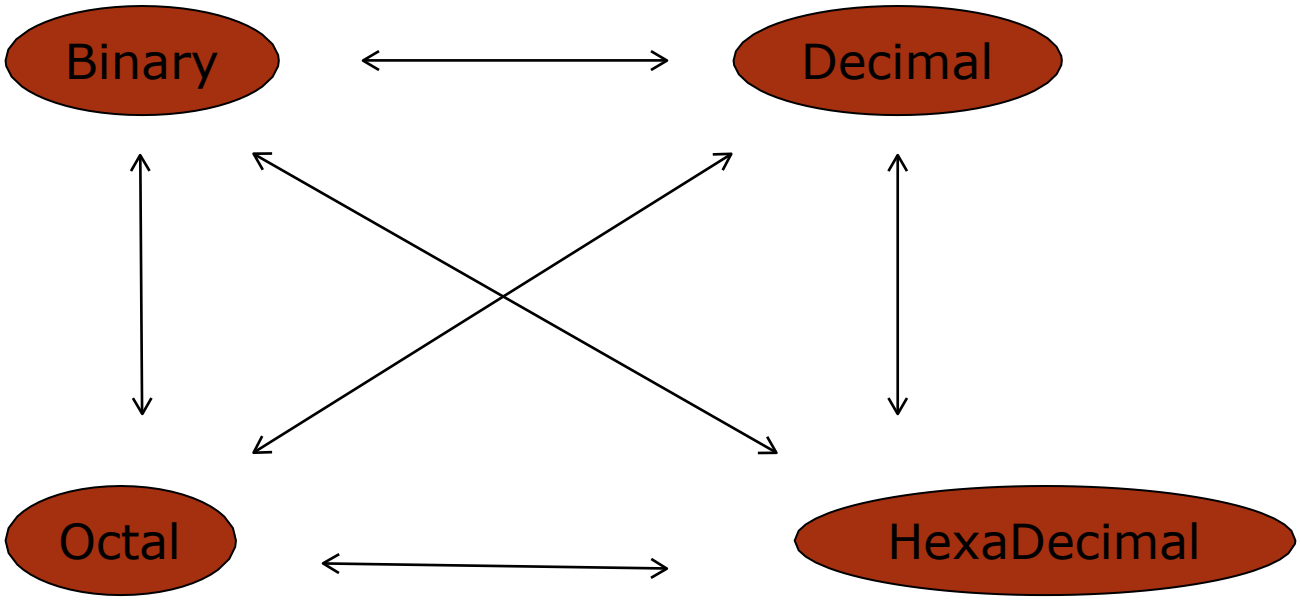
(mỗi chữ số x_i lấy từ tập X có q phần tử)

- Ví dụ:
  - Hệ cơ số 10: A = 123 = 100 + 20 + 3 = **1**.10^2 + **2**.10^1 + **3**.10^0
  - q = 2, X = {0, 1}: hệ nhị phân (binary)
  - q = 8, X = {0, 1, 2,…, 7}: hệ bát phân (octal)
  - q = 10, X = {0, 1, 2,…, 9}: hệ thập phân (decimal)
  - q = 16, X = {0, 1, 2,…,9, A, B,…, F}: hệ thập lục phân (hexadecimal)
- Chuyển đổi: A = 123 d = 01111011 b = 173 o = 7B h
- Hệ cơ số thường được biểu diễn trong máy tính là hệ cơ số 2

Chuyển đổi giữa các hệ cơsố

- Đặc điểm
  - Con người sử dụng hệ thậpphân
  - Máy tính sử dụng hệ nhị phân, bát phân, thập lục phân
- Nhu cầu
  - Chuyển đổi qua lại giữa các hệ đếm?
    - Hệ khác sang hệ thập phân (... → dec)
    - Hệ thập phân sang hệ khác (dec → ...)
    - Hệ nhị phân sang hệ khác và ngược lại (bin ←→ ...)
    - ...

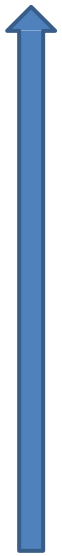
Chuyển đổi các hệ đếm



Chuyển đổi giữa các hệ cơ số

[1] Decimal (10) → Binary (2)

- Lấy số cơ số 10 chia cho 2
  - Số dư đưa vào kết quả
  - Số nguyên đem chia tiếp cho 2
  - Quá trình lặp lại cho đến khi số nguyên = 0
- Ví dụ: A = 123
  - 123 : 2 = 61 dư 1
  - 61 : 2 = 30 dư 1
  - 30 : 2 = 15 dư 0
  - 15 : 2 = 7 dư 1
  - 7 : 2 = 3 dư 1
  - 3 : 2 = 1 dư 1
  - 1 : 2 = 0 dư 1



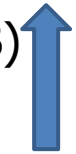
Kết quả: 1111011, vì 123 là số dương, thêm 1 bit hiển dấu vào đầu là 0 vào

→ Kết quả cuối cùng: **01111011**

Chuyển đổi giữa các hệ cơ số

[2] Decimal (10) → Hexadecimal (16)

- Lấy số cơ số 10 chia cho 16
  - Số dư đưa vào kết quả
  - Số nguyên đem chia tiếp cho 16
  - Quá trình lặp lại cho đến khi số nguyên = 0
- Ví dụ: A = 123
  - 123 : 16 = 7 dư 12 (B)
  - 7 : 16 = 0 dư 7



→ Kết quả cuối cùng: **7B**

Chuyển đổi giữa các hệ cơ số

[3] Binary (2) → Decimal (10)

- Khai triển biểu diễn và tính giá trị biểu thức

$$x_{n-1}...x_1x_0 = x_{n-1}.2^{n-1} + ... + x_1.2^1 + x_0.2^0$$

- Ví dụ:
  - $1011_2 = 1.2^3 + 0.2^2 + 1.2^1 + 1.2^0 = 11_{10}$

Chuyển đổi giữa các hệ cơ số

[4] Binary (2) → Hexadecimal (16)

- Nhóm từng bộ 4 bit trong biểu diễn nhị phân rồi chuyển sang ký số tương ứng trong hệ thập lục phân (0000 → 0,..., 1111 → F)
- Ví dụ
  - $1001011_2 = 0100\ 1011 = 4B_{16}$

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

Chuyển đổi giữa các hệ cơ số

[5] Hexadecimal (16) → Binary (2)

- Sử dụng bảng dưới đây để chuyển đổi:

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

- Ví dụ:
  - $4B_{16} = 1001011_2$

Chuyển đổi giữa các hệ cơ số

[6] Hexadecimal (16) → Decimal (10)

- Khai triển biểu diễn và tính giá trị biểu thức

$$x_{n-1}...x_1x_0 = x_{n-1}.16^{n-1} + ... + x_1.16^1 + x_0.16^0$$

- Ví dụ:
  - $7B_{16} = 7.16^1 + 12\ (B).16^0 = 123_{10}$

$$x_{n-1}...x_1x_0 = x_{n-1}.2^{n-1} + ... + x_1.2^1 + x_0.2^0$$

- Được dùng nhiều trong máy tính để biểu diện các giá trị lưu trong các thanh ghi hoặc trong các ô nhớ. Thanh ghi hoặc ô nhớ có kích thước 1 byte (8 bit) hoặc 1 word (16 bit).
- n được gọi là chiều dài bit của số đó
- Bit trái nhất x_{n-1} là bit có giá trị (nặng) nhất MSB (Most Significant Bit)
- Bit phải nhất x_0 là bit ít giá trị (nhẹ) nhất LSB (Less Significant Bit)

Ý tưởng nhị phân

- Số nhị phân có thể dùng để biểu diễn bất kỳ việc gì mà bạn muốn!
- Một số ví dụ:
  - Giá trị logic: 0 → False; 1 → True
  - Ký tự:
    - 26 ký tự (A → Z): 5 bits ($2^5 = 32$)
    - Tính cả trường hợp viết hoa/thường + ký tự lạ → 7 bits (ASCII)
    - Tất cả các ký tự ngôn ngữ trên thế giới → 8, 16, 32 bits (Unicode)
  - Màu sắc: Red (00), Green (01), Blue (11)
  - Vị trí / Địa chỉ: (0, 0, 1)…
  - Bộ nhớ: N bits → Lưu được tối đa 2^N đối tượng

Thực hành chuyển đổi các hệ đếm

Hệ 2	Hệ 8	Hệ 10	Hệ 16
10101.101			
	5321		
		2345	
			6F

6.2 Mã hóa và biểu diễn dữ liệu máy tính

Nguyên tắc chung về mã hoá dữ liệu

Mọi dữ liệu được đưa vào máy tính được mã hoá thành số nhị phân.

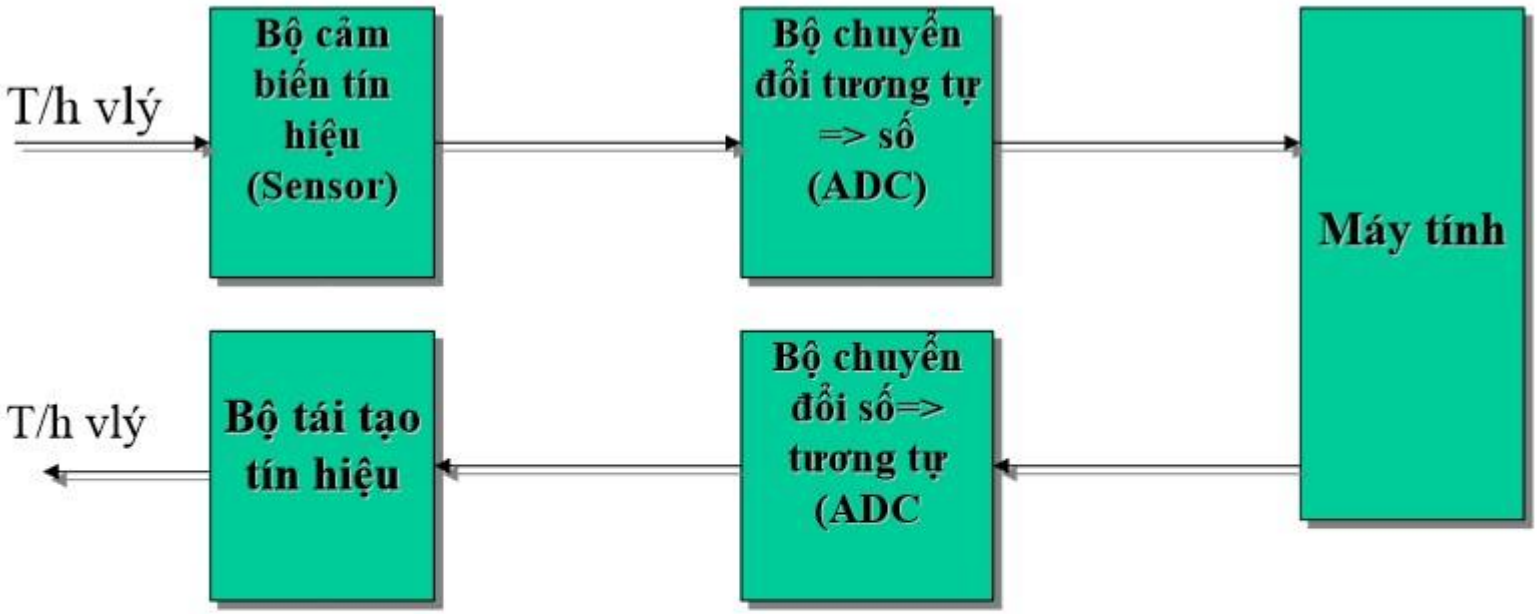
Các loại dữ liệu:

- Dữ liệu nhân tạo: do con người quy ước
- Dữ liệu tự nhiên: tồn tại khách quan với con người

Mã hoá dữ liệu nhân tạo

- Dữ liệu số nguyên: mã hoá theo một số chuẩn đã qui ước
- Dữ liệu số thực: mã hoá bằng số dấu chấm động
- Dữ liệu phi số (ký tự): mã hoá theo các bộ mã ký tự hiện hành như : ASCII, Unicode,…

Mô hình mã hóa và tái tạo tín hiệu



Các dữ liệu vật lý thông dụng

- ✓ Âm thanh
- ✓ Hình ảnh

Thứ tự lưu trữ dữ liệu bên trong máy tính

- Bộ nhớ chính tổ chức lưu trữ dữ liệu theo đơn vị byte
- Độ dài từ dữ liệu có thể chiếm từ 1 đến 4 byte. Vì vậy cần phải biết thứ tự chúng lưu trữ trong bộ nhớ chính đối các dữ liệu nhiều byte.
- Có hai cách lưu trữ được đưa ra

✓ **Little Endian** (đầu nhỏ): Byte có ý nghĩa thấp hơn được lưu trữ trong bộ nhớ ở vị trí có địa chỉ nhỏ hơn.

✓ **Big Endian** (đầu to): Byte có ý nghĩa thấp hơn được lưu trữ trong bộ nhớ ở vị trí có địa chỉ lớn hơn.

Minh Họa

☞ *Ví dụ: lưu trữ một từ 32bit*

0001 1010 0010 1011 0011 1100 0100 1101B

1 A 2 B 3 C 4 D$_H$

Biểu diễn trong ngăn nhớ theo 2 cách

300	4D
301	3C
302	2B
303	1A

Little Endian

300	1A
301	2B
302	3C
303	4D

Big Endian

Minh Họa

- Lưu trữ của các bộ vi xử lý điển hình
- Loại máy Intel: 80x86, Petium -> little endian
- Motorola 680x0 và các bộ xử lý RISC -> big endian
- Power PC & Itanium: tích hợp cả hai cách trên

Biểu diễn số nguyên

- Số nguyên không dấu (Unsigned Integer)
- Số nguyên có dấu (Signed Integer)

a. Biểu diễn số nguyên không dấu

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

$$a_{n-1}\, a_{n-2} \ldots a_2 a_1 a_0$$

- 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=151
  - $A = 41 = 32 + 8 + 1 = 2^5 + 2^3 + 2^0$

  => 41 = 0010 1001
  - $B = 151 = 128 + 16 + 4 + 2+1 = 2^7+2^4+2^2+2^1+2^0$

  => 151 = 1001 0111

Các ví dụ (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 ?
- Bài 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

0000 0000 = 0

0000 0001 = 1

0000 0010 = 2

0000 0011 = 3

...

1111 1111 = 255

Chú ý:

1111 1111

\+ 0000 0001

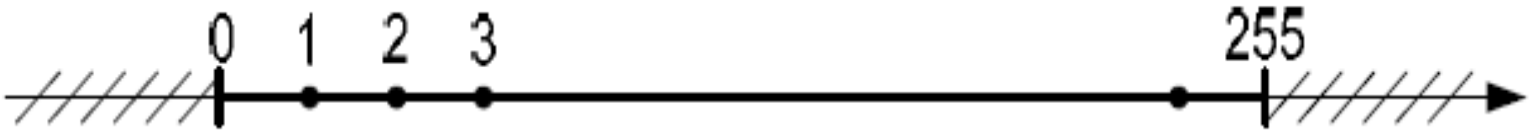
1 0000 0000

Vậy: 255 + 1 = 0 ?

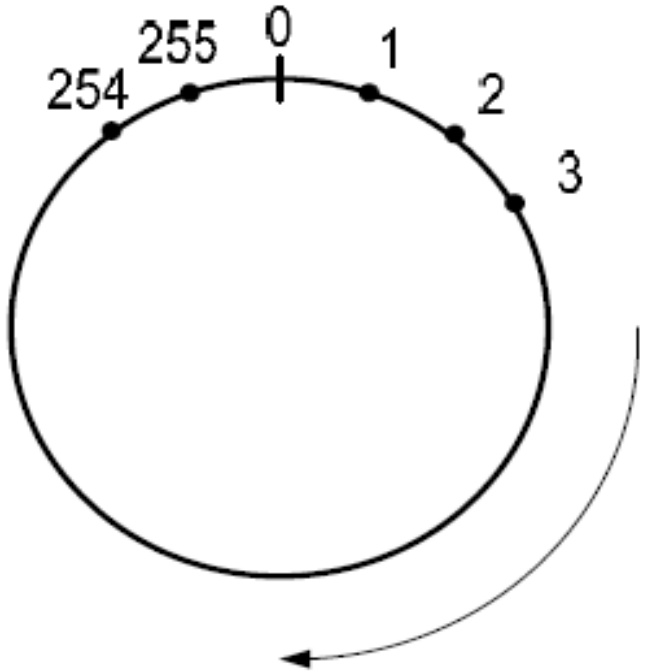
=> do tràn nhớ ra ngoài

Trục số học số 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 $2^{64} - 1$

b. Biểu diễn số nguyên có dấu

a. Số bù chín và Số bù mười

- Cho một số thập phân A được biểu diễn bằng n chữ số thập phân, ta có:
  - Số bù chín của A = $(10^n-1) - A$
  - Số bù mười của A = $10^n - A$
- Số bù mười của A = (Số bù chín của A) +1

Số bù chín và Số bù mười (tiếp)

- Ví dụ: với n=4, cho A = 3265
  - Số bù chín của A:

9999	(10^4-1)
- 3265	(A)
6734	

  - Số bù mười của A:

10000	(10^4)
- 3265	(A)
6735	

b. Số bù một và Số bù hai

- Định nghĩa: Cho A là một số nhị phân 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

Số bù một và Số bù hai (tiếp)

- Ví dụ: Cho số N = 0001 0001_2 được biểu diễn bởi n=8bit. Xác định số bù 1 và bù 2 của N.
- Số bù 1 của N
  - *Áp dụng công thức*

$$\begin{array}{rl} & \mathbf{1111\ 1111}\ (2^8\text{-}1) \\ - & \mathbf{0001\ 0001}\ (\text{N}) \\ \hline \textbf{\textit{số bù một của N}} & \mathbf{1110\ 1110} \end{array}$$

- Số bù 2 của N
  - *Áp dụng công thức*

$$\begin{array}{rl} & \mathbf{1\ 0000\ 0000}\ (2^8) \\ - & 0001\ 0001\ (\text{N}) \\ \hline \textbf{số bù hai của N:} & \mathbf{1110\ 1111} \end{array}$$

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 = 1101 1010
  - + 1
  - Số bù hai = 1101 1011
- Nhận xét:
  - A = 0010 0101
  - Số bù hai = + 1101 1011
  - 1 0000 0000 = 0

(bỏ qua bit nhớ ra ngoài)

c. Biểu diễn số nguyên có dấu bằng mã bù hai

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

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

- **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:

$$1a_{n-2}\ldots a_2 a_1 a_0$$

- 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 số nguyên A:

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

- Dải biểu diễn cho số nguyên A: -2^{n-1} đến $2^{n-1}-1$

Các ví dụ

- **Ví dụ 1**: Biểu diễn số nguyên có dấu sau đây theo hai dạng kiểu n=8 bit và n=16 bit trong máy tính:

  A=+97 và B=-101

- **Giải**: n = 8 bit
  - Biểu diễn số A dạng số nguyên có dấu trong máy tính
    - A: 97 = **0110 0001**
  - Biểu diễn số B dạng số nguyên có dấu trong máy tính
    - Biểu diễn số +101 = **0110 0101**
    - Lấy bù 2 **1001 1011**
    - => B = - 101 = **1001 1011**
- N = 16 bit ?

Các ví dụ

- **Ví dụ 2**: Hãy xác định giá trị của các số nguyên có dấu được biểu diễn dưới đây:
  - P = 0110 0010
  - Q = 1101 1011
- **Giải:**
  - P = 0110 0010 = 64 + 32 + 2 = +98
  - Q = 1101 1011 = -128+64+16+8+2+1 = -37

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:

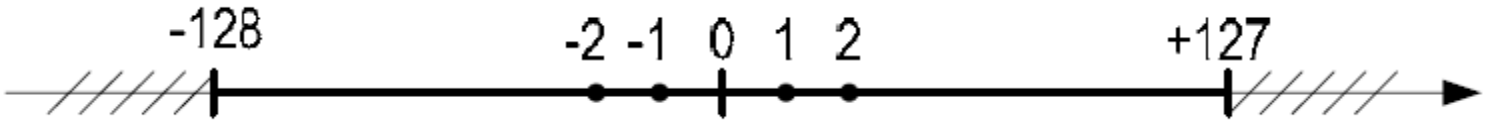
+127 + 1 = -128

-128 - 1 = +127

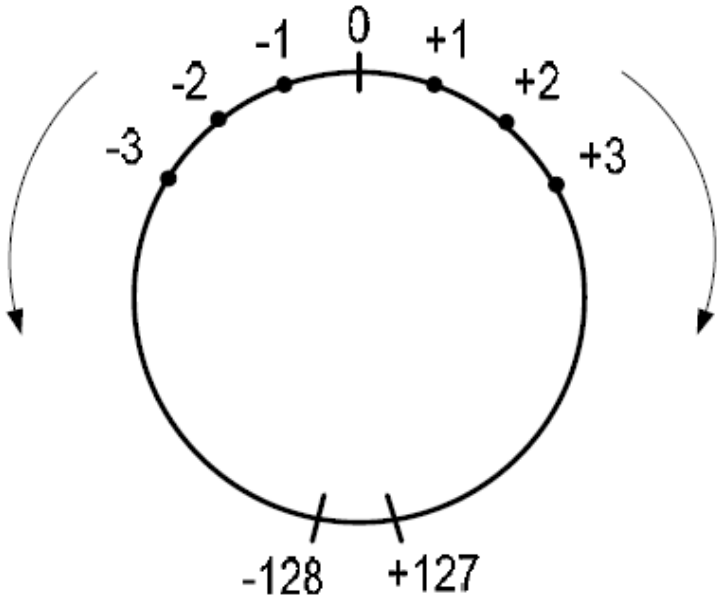
=>do tràn xảy ra

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 -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=32bit: biểu diễn từ -2^{31} đến $2^{31}-1$
- Với n=64bit: biểu diễn từ -2^{63} đến $2^{63}-1$

Chuyển đổi từ byte thành word

- Đối với số dương:
  - +19 = 0001 0011 (8bit)
  - +19 = 0000 0000 0001 0011 (16bit)

  =>thêm 8 bit 0 bên trái
- Đối với số âm:
  - - 19 = 1110 1101 (8bit)
  - - 19 = 1111 1111 1110 1101 (16bit)
  - => thêm 8 bit 1 bên trái

Minh họa

Câu 1: Kết quả hiển thi lên màn hình là bao nhiêu? Khi thực hiện đoạn lệnh sau:

```
Var a: shortint;
Begin
        a:=-1;
        writeln('Gia tri a:=',a);
        writeln('Gia tri ngan nho:=', mem[seg(a):ofs(a)]);
End.
```

Minh họa

Câu 2: Kết quả hiển thi lên màn hình là bao nhiêu? Khi thực hiện đoạn lệnh sau:

```
Var a: shortint;
Begin
        a:=-128;
        writeln('Gia tri a:=',a);
        writeln('Gia tri ngan nho:=', mem[seg(a):ofs(a)]);
End.
```

Minh họa

Câu 3: Kết quả hiển thi lên màn hình là bao nhiêu? Khi thực hiện đoạn lệnh sau:

```
Var a: shortint;
Begin
        a:=$6A;
        writeln('Gia tri a:=',a);
        writeln('Gia tri ngan nho:=', mem[seg(a):ofs(a)]);
End.
```

Minh họa

Câu 4: Kết quả hiển thi lên màn hình là bao nhiêu? Khi thực hiện đoạn lệnh sau:

```
Var    b : integer absolute 3715:100;
       a : shortint absolute 3715:100;
Begin
   b:=$00B5;
   writeln('Gia tri a:=',a);
   writeln('Gia tri ngan nho:=', mem[seg(a):ofs(a)]);
End.
```

Minh họa

Câu 5: Kết quả hiển thi lên màn hình là bao nhiêu? Khi thực hiện đoạn lệnh sau:

```
Var     b : integer absolute 3715:100;
        a: shortint absolute 3715:100;
Begin
    b:=-75;
    writeln('Gia tri a:=',a);
    writeln('Gia tri ngan nho:=', mem[seg(a):ofs(a)]);
    writeln('Gia tri ngan nho:=', mem[seg(a):ofs(a)+1]);
    writeln('Gia tri ngan nho:=', memw[seg(a):ofs(a)]);
End.
```

c. Biểu diễn số nguyên theo mã BCD

- Binary Coded Decimal Code
- Dùng 4 bit để mã hoá cho các chữ số thập phân từ 0 đến 9

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

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

Ví dụ số BCD

- Biểu diễn số
  - **$38 = 0011\ 1000_{BCD}$**
  - **$61 = 0110\ 0001_{BCD}$**
  - **$1087 = 0001\ 0000\ 1000\ 0111_{BCD}$**

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 38 được lưu trữ như sau:

- BCD gói (Packed BCD): Hai số BCD được lưu trữ trong 1 byte.
  - Ví dụ: Số 38 được lưu trữ như sau:

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 => kết quả sai

\+ 0110 0110 <= hiệu chỉnh

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

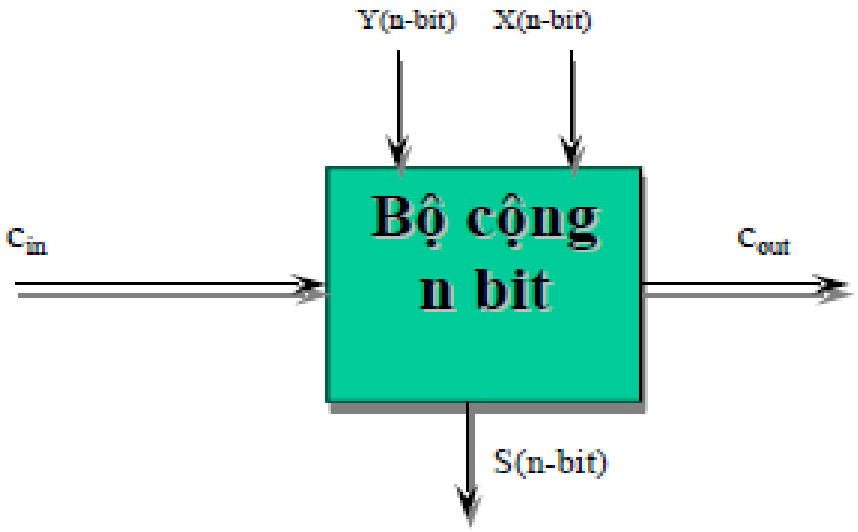
1 8 3

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

6.3 Số học nhị phân

❖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 tràn nhớ ra ngoài *(Carry Out)*.
  - Tràn nhớ ra ngoài khi: tổng > $(2^n - 1)$

Ví dụ cộng số nguyên không dấu

- 57 = 0011 1001
  + 34 = + 0010 0010
  91 0101 1011 = 64+16+8+2+1=91 => đúng
- 209 = 1101 0001
  + 73 = + 0100 1001
  282 1 0001 1010
  0001 1010 = 16+8+2=26 => sai
  =>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

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 trong máy tính thực chất là lấy bù hai

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}
  - Cộng hai số khác dấu: kết quả luôn luôn đúng.
  - Cộng hai 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:

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

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
  +45 1 0010 1101 = +45
- (- 90) = 1010 0110
  +(+36) = 0010 0100
  -54 1100 1010 = -54
- (-74) = 1011 0110
  +(-30) = 1110 0010
  - 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

  \+ (- 43) = 1101 0101

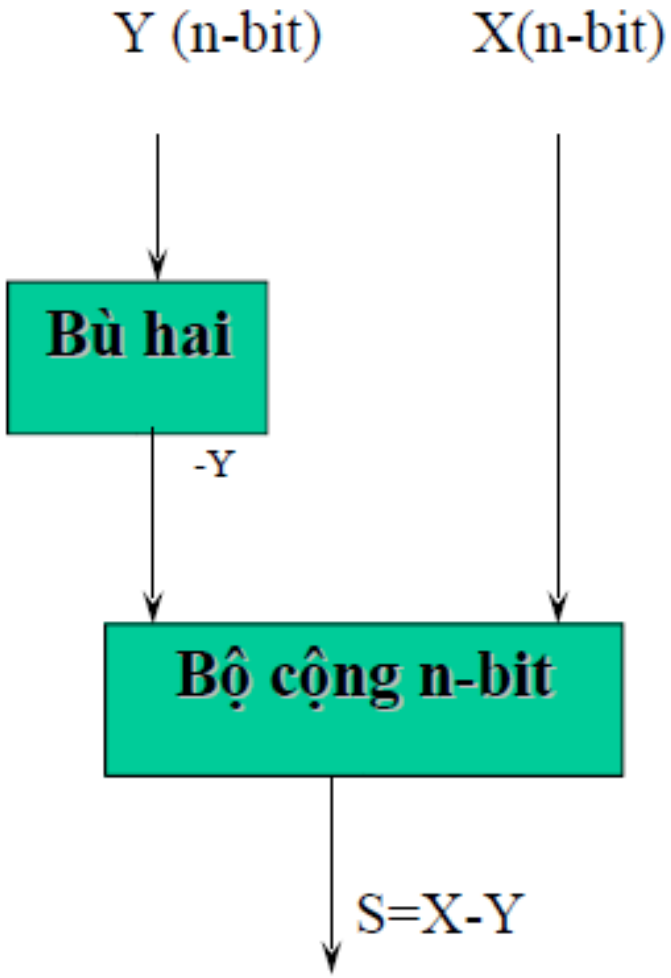
  - 147 1 0110 1101

  = 64 + 32 + 8 + 4 + 1 = +109 -> sai

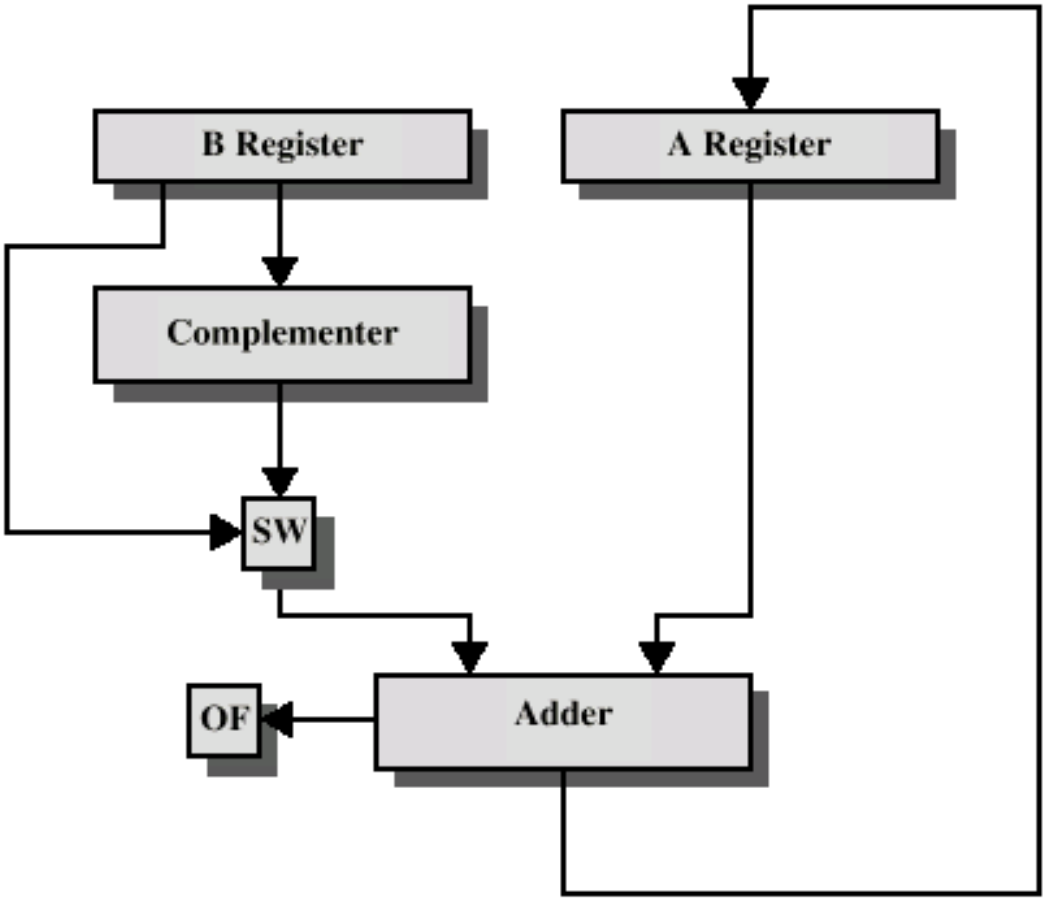
- Cả 2 ví dụ trên đều tràn vì tổng nằm ngoài dải biểu diễn [-128, +127]

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



Mạch phần cứng cho phép cộng, trừ



OF = overflow bit
SW = Switch (select addition or subtraction)

Thực hiện phép nhân

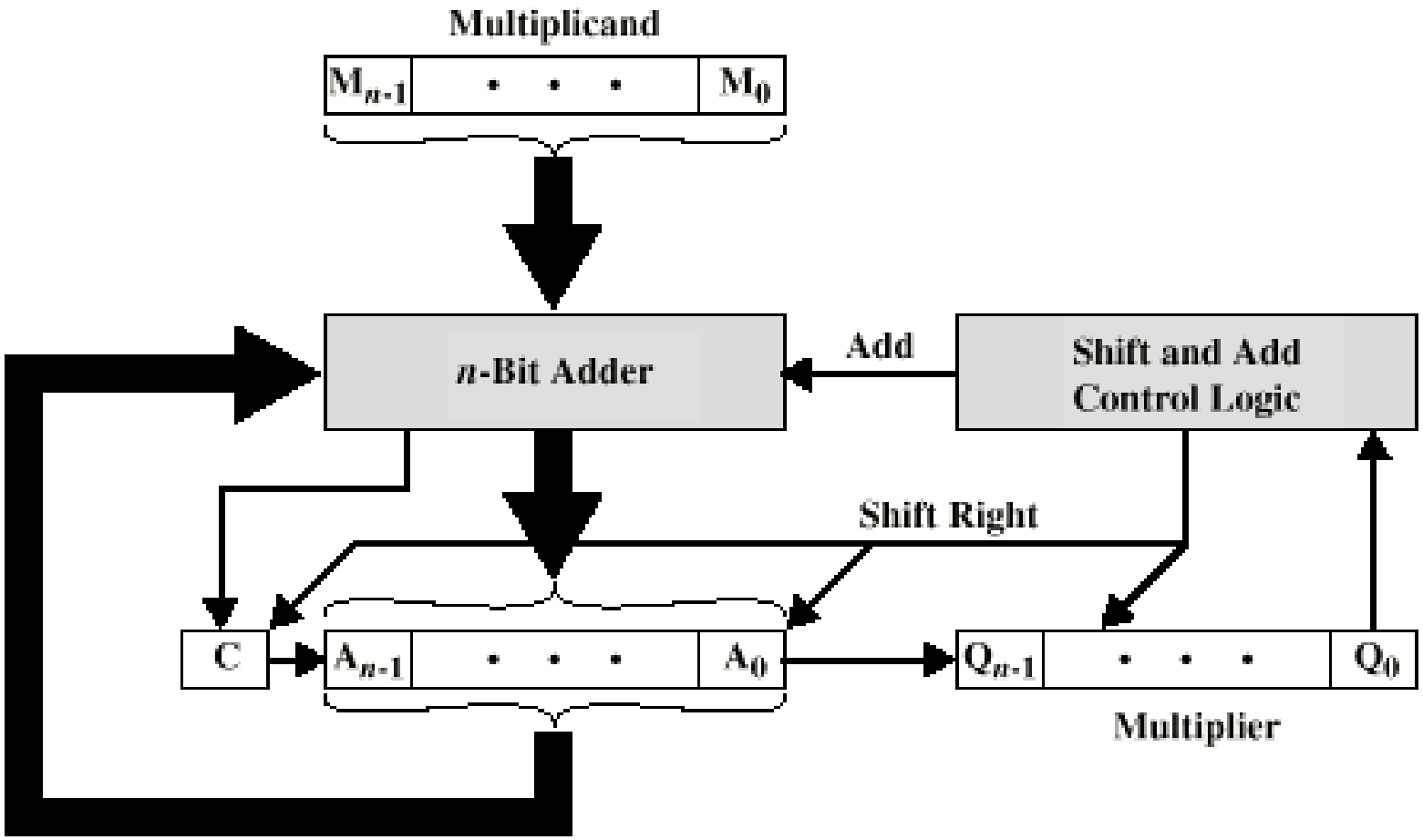
- Nhân số nguyên không dấu

```
    1011   Số bị nhân (Multiplicand)   (11 hệ 10)
  x 1101   Số nhân     (Multiplier)     (13)
    1011
+   0000
  1011
 1011                  Tích riêng phần
10001111   Tích        (Product)        (143)
```

Nhân số nguyên không dấu (tiếp)

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

Bộ nhân số nguyên không dấu

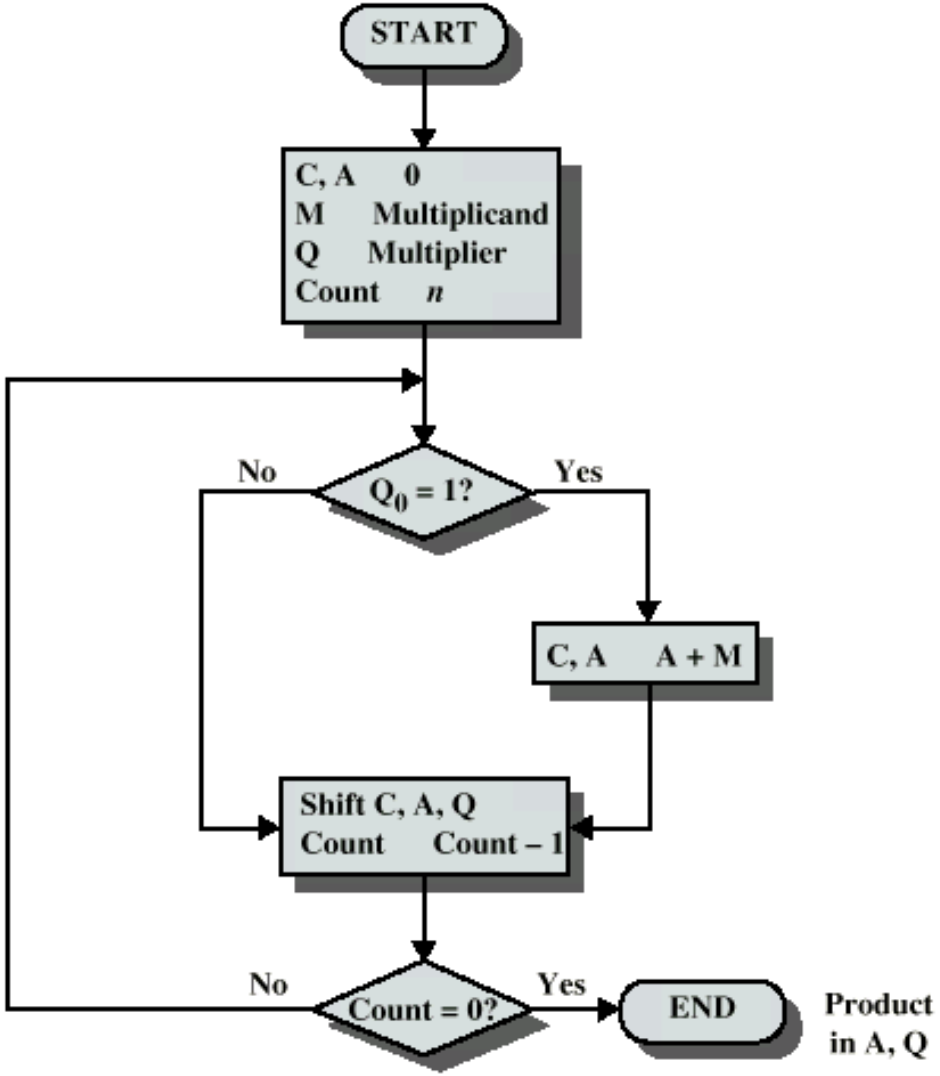


(a) Block Diagram

Ví dụ

C	A	Q	M		
0	0000	1101	1011	Initial Values	
0	1011	1101	1011	Add	First Cycle
0	0101	1110	1011	Shift	
0	0010	1111	1011	Shift	Second Cycle
0	1101	1111	1011	Add	Third Cycle
0	0110	1111	1011	Shift	
1	0001	1111	1011	Add	Fourth Cycle
0	1000	1111	1011	Shift	

Lưu đồ nhân số nguyên không dấu



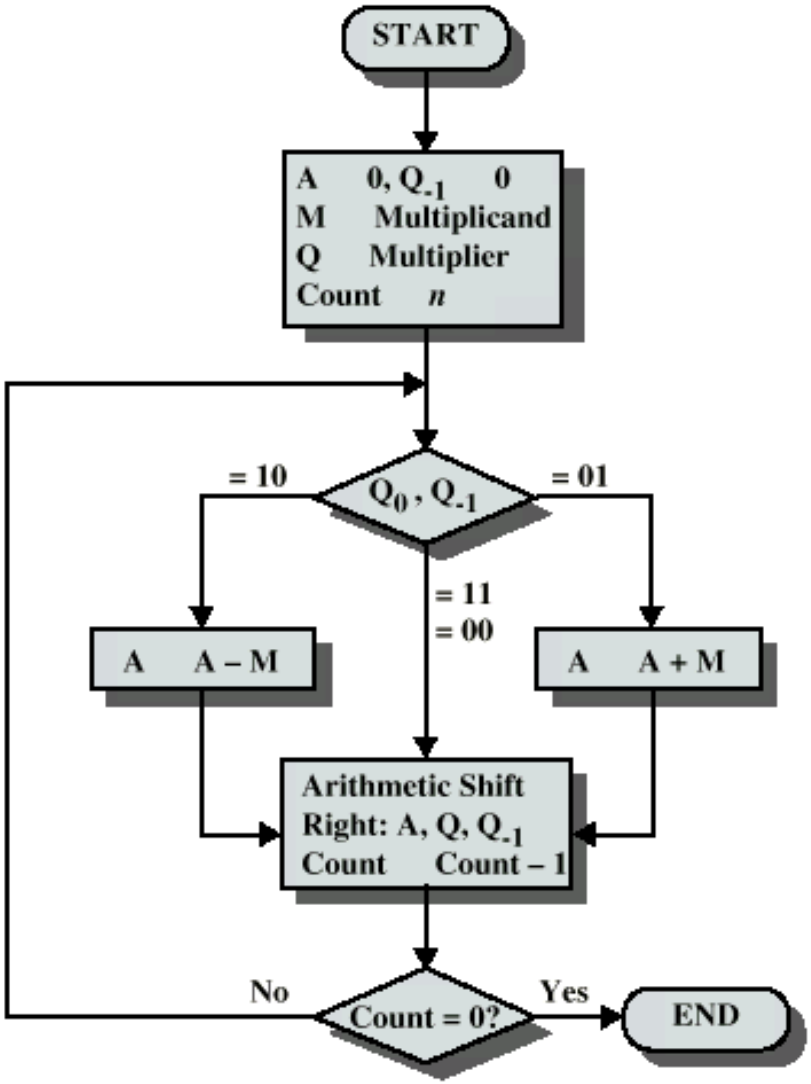
Nhân số nguyên có dấu

- Sử dụng thuật giải nhân không dấu
- Sử dụng thuật giải Booth

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.

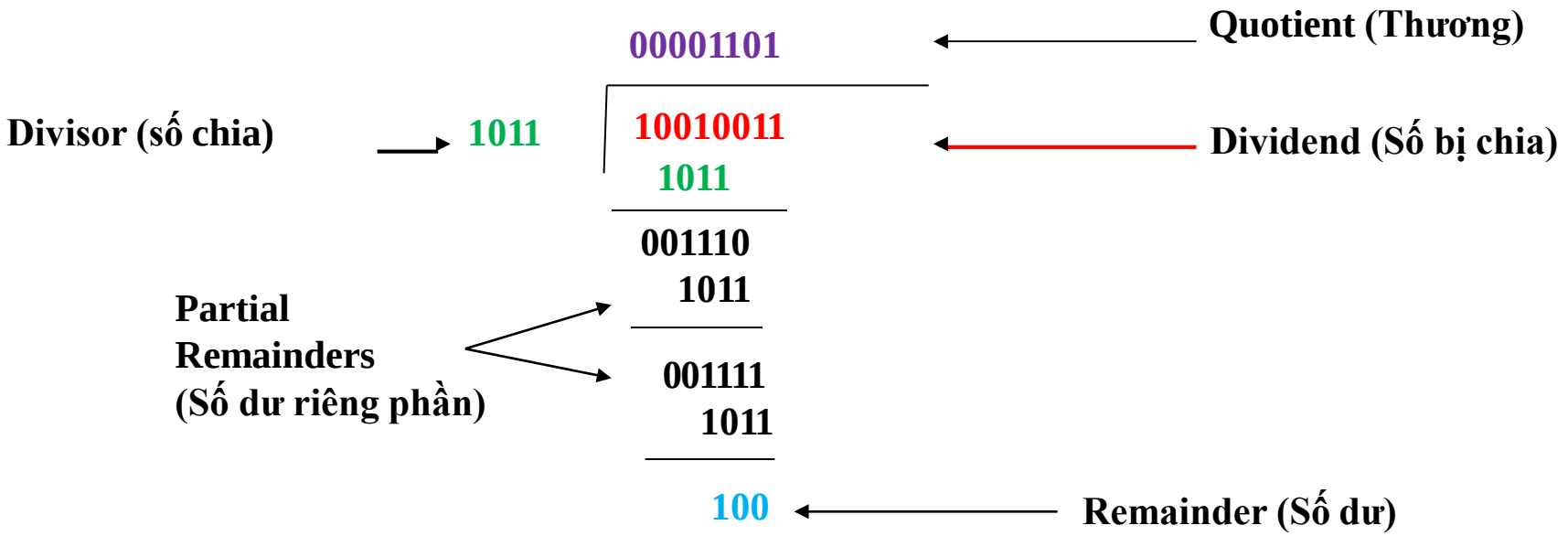
Giải thuật Booth



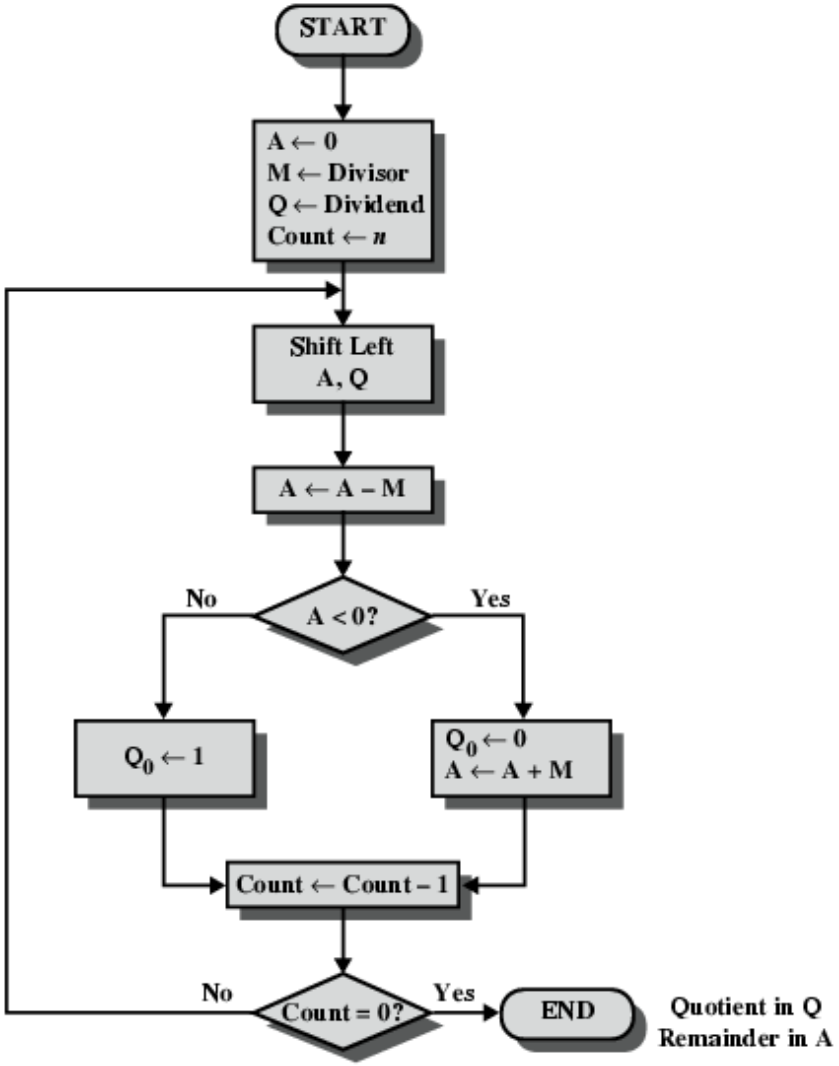
Ví dụ

A	Q	Q_{-1}	M		
0000	0011	0	0111	Initial Values	
1001	0011	0	0111	A A - M	First Cycle
1100	1001	1	0111	Shift	
1110	0100	1	0111	Shift	Second Cycle
0101	0100	1	0111	A A + M	Third Cycle
0010	1010	0	0111	Shift	
0001	0101	0	0111	Shift	Fourth Cycle

Thực hiện phép chia



Lưu đồ chia số nguyên không dấu



Vi dụ

M=0011 (Divisor), Q=0111 (Dividend)

A	Q	
0000	0111	Initial value
0000	1110	Shift
1101		Use twos complement of 0011 for subtraction
1101		Subtract
0000	1110	Restore, set $Q_0 = 0$
0001	1100	Shift
1101		
1110		Subtract
0001	1100	Restore, set $Q_0 = 0$
0011	1000	Shift
1101		
0000	1001	Subtract, set $Q_0 = 1$
0001	0010	Shift
1101		
1110		Subtract
0001	0010	Restore, set $Q_0 = 0$

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)

- (+) : (+) → không hiệu chỉnh dấu kết quả
- (+) : (-) → đảo dấu thương
- (-) : (+) → đảo dấu thương và phần dư
- (-) : (-) → đảo dấu phần dư

Số dấu phẩy động

Cho hai giá trị:

Khối lượng mặt trời:

1990000000000000000000000000000000g

Khối lượng điện tử:

0.000000000000000000000000000910956g

Để lưu trữ con số này thì máy tính cần đến số bit rất lớn. Như vậy, trong trường hợp này thì loại số có dấu chấm tĩnh sẽ rất bất tiện. Vì vậy tất cả máy tính lưu trữ những số trên dưới dạng dấu chấm động (floating point) 1.990×10^{33} và 0.910956×10^{-27} hay theo số khoa học là : 1.999E+33 và 0.910956E-27.

Số dấu phẩy động

1. Nguyên tắc chung

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

$$X = M * R^E$$

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

Chuẩn IEEE754/85

- Cơ số R = 2
- Các dạng:
  - Dạng 32-bit
  - Dạng 44-bit
  - Dạng 64-bit
  - Dạng 80-bit

Các dạng biểu diễn chính

- Single (32 bit)

S (1)	e (8)	m (23)

- Double (64 bit)

S (1)	e (11)	m (52)

- Dạng 80 bit

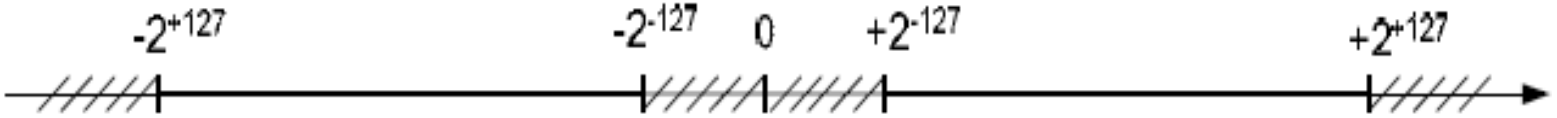
S (1)	e (15)	m (64)

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 → E = e – 127
  - giá trị 127 được 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:

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

- Dải biểu diễn: 2^{-127} đến 2^{+127} (10^{-38} đến 10^{+38})



Ví dụ

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

- X= 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
- Y = 0011 1111 1000 0000 0000 0000 0000 0000 = ?

(= +1.0)

Ví dụ

Biểu diễn số thực X = -2345,125 về dạng số dấu chấm động IEEE754 32-bit

B1: Chuyển đổi số trên ra hệ hai

-2345,125d = -1001 0010 1001.001b (dãy số nhị phân được biểu diễn bình thường)

B2: Chuẩn hoá theo IEEE 32bit

$-1.001\ 0010\ 1001\ 001 \times 2^{11}$

B3: Xác định các thông số biểu diễn s,M,E

S: phần định trị là số âm, nên s là 1

E : phần mũ được xác định e = E-127
=> E = 11+127=138=10001010

M: phần định trị được xác định là 001 0010 1001 0010 0000 0000 (số 32 bit)

Ví dụ

- Biểu diễn số thực X = 83.75 về dạng số dấu chấm động IEEE754 32-bit
- ***Giải:***
- $X = 83.75_{10} = 1010011.11_2 = 1.01001111 \times 2^6$
- Ta có:
  - S = 0 vì đây là số dương
  - E = e-127=6 → e = 127+6=133_{10} = 1000 0101_2
- Vậy:

  X = 0100 0010 1010 0111 1000 0000 0000 0000

Bài tập

- Biểu diễn các số thực sau đây về dạng số dấu phẩy động IEEE754 32-bit:

  X = - 27.0625; Y = 1/32

Các quy ước đặc biệt

- Các bit của e bằng 0, các bit của m bằng 0, thì X= ± 0 x 000 0000 0000 0000 0000 0000 0000 0000 → X = ± 0
- Các bit của e bằng 1, các bit của m bằng 0, thì X= ±∞ x 111 1111 1000 0000 0000 0000 0000 0000 → X= ± ∞
- Các bit của e bằng 1, còn m có ít nhất 1 bit bằng 1, thì nó không biểu diễn cho số nào cả (NaN – not a number)

Dạng 64-bit

- S (1 bit) là bit dấu
- e (11 bit) là mã *excess-1023 của phần mũ E:*

→ E = e – 1023

- m (52 bit) là phần lẻ của phần định trị M: M=1.m
- Giá trị của 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) là mã *excess-16383 của phần mũ E:* → E = e – 16383
- m (64 bit) là phần lẻ của phần định trị M: M=1.m
- Giá trị của 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 $E2 \geq E1$

Biểu diễn mã

Các loại mã thông dụng

- Mã ASCII (American Standard Code for Information Interchange): 8 bit
- Unicode: 16 bit
- Mã EBCDI (Extendend Binary Coded Decimal Interchange): 8 bit
- Mã BAUDOT: Sử dụng nhiều trong ngành bưu điện

Mã ASCII

Ctrl	Dec	Hex	Char	Code
^@	0	00		NUL
^A	1	01	☺	SOH
^B	2	02	☻	STX
^C	3	03	♥	ETX
^D	4	04	♦	EOT
^E	5	05	♣	ENQ
^F	6	06	♠	ACK
^G	7	07	•	BEL
^H	8	08	◘	BS
^I	9	09	○	HT
^J	10	0A	◙	LF
^K	11	0B	♂	VT
^L	12	0C	♀	FF
^M	13	0D	♪	CR
^N	14	0E	♫	SO
^O	15	0F	☼	SI
^P	16	10	►	DLE
^Q	17	11	◄	DC1
^R	18	12	↕	DC2
^S	19	13	‼	DC3
^T	20	14	¶	DC4
^U	21	15	§	NAK
^V	22	16	▬	SYN
^W	23	17	↨	ETB
^X	24	18	↑	CAN
^Y	25	19	↓	EM
^Z	26	1A	→	SUB
^[	27	1B	←	ESC
^\	28	1C	∟	FS
^]	29	1D	↔	GS
^^	30	1E	▲	RS
^_	31	1F	▼	US

Dec	Hex	Char
32	20	sp
33	21	!
34	22	"
35	23	#
36	24	$
37	25	%
38	26	&
39	27	'
40	28	(
41	29	)
42	2A	*
43	2B	+
44	2C	,
45	2D	-
46	2E	.
47	2F	/
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?

Dec	Hex	Char
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	\
93	5D	]
94	5E	^
95	5F	_

Dec	Hex	Char
96	60	`
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	|
125	7D	}
126	7E	~
127	7F	⌂†

Dec	Hex	Char
128	80	Ç
129	81	ü
130	82	é
131	83	â
132	84	ä
133	85	à
134	86	å
135	87	ç
136	88	ê
137	89	ë
138	8A	è
139	8B	ï
140	8C	î
141	8D	ì
142	8E	Ä
143	8F	Å
144	90	É
145	91	æ
146	92	Æ
147	93	ô
148	94	ö
149	95	ò
150	96	û
151	97	ù
152	98	ÿ
153	99	Ö
154	9A	Ü
155	9B	¢
156	9C	£
157	9D	¥
158	9E	₧
159	9F	ƒ

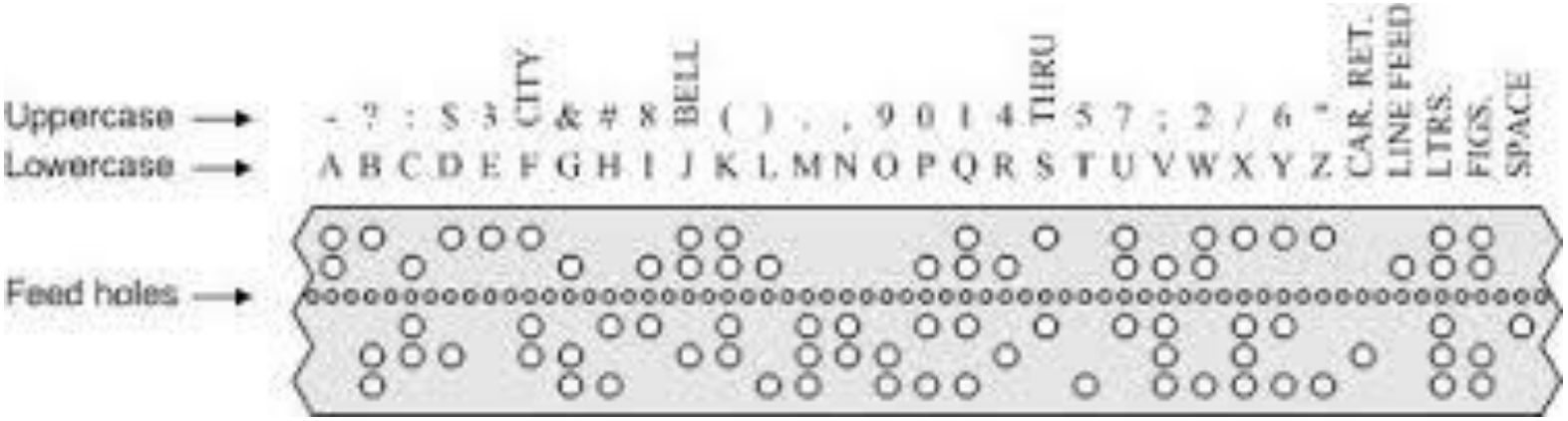
Dec	Hex	Char
160	A0	á
161	A1	í
162	A2	ó
163	A3	ú
164	A4	ñ
165	A5	Ñ
166	A6	ª
167	A7	º
168	A8	¿
169	A9	⌐
170	AA	¬
171	AB	½
172	AC	¼
173	AD	¡
174	AE	«
175	AF	»
176	B0	▒
177	B1	▒
178	B2	▒
179	B3	│
180	B4	┤
181	B5	╡
182	B6	╢
183	B7	╖
184	B8	╕
185	B9	╣
186	BA	║
187	BB	╗
188	BC	╝
189	BD	╜
190	BE	╛
191	BF	┐

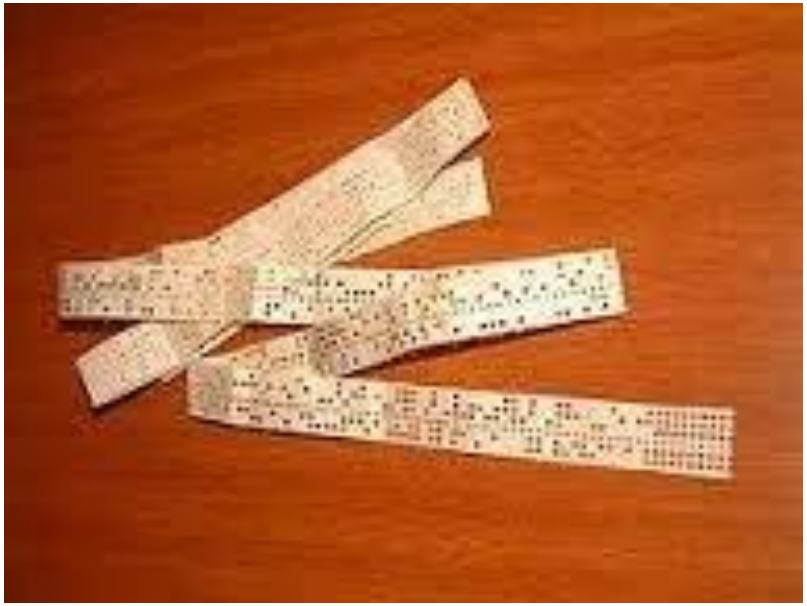
Dec	Hex	Char
192	C0	└
193	C1	┴
194	C2	┬
195	C3	├
196	C4	─
197	C5	┼
198	C6	╞
199	C7	╟
200	C8	╚
201	C9	╔
202	CA	╩
203	CB	╦
204	CC	╠
205	CD	═
206	CE	╬
207	CF	╧
208	D0	╨
209	D1	╤
210	D2	╥
211	D3	╙
212	D4	╘
213	D5	╒
214	D6	╓
215	D7	╫
216	D8	╪
217	D9	┘
218	DA	┌
219	DB	█
220	DC	▄
221	DD	▌
222	DE	▐
223	DF	▀

Dec	Hex	Char
224	E0	α
225	E1	ß
226	E2	Γ
227	E3	π
228	E4	Σ
229	E5	σ
230	E6	µ
231	E7	τ
232	E8	Φ
233	E9	Θ
234	EA	Ω
235	EB	δ
236	EC	∞
237	ED	φ
238	EE	ε
239	EF	∩
240	F0	≡
241	F1	±
242	F2	≥
243	F3	≤
244	F4	⌠
245	F5	⌡
246	F6	÷
247	F7	≈
248	F8	°
249	F9	∙
250	FA	·
251	FB	√
252	FC	ⁿ
253	FD	²
254	FE	■
255	FF	

† ASCII code 127 has the code DEL. Under MS-DOS, this code has the same effect as ASCII 8 (BS). The DEL code can be generated by the CTRL+BKSP key.

Mã BAUDOT


Uppercase
Lowercase
Feed holes
- ? : $ 3 CITY & # 8 BELL ( ) . , 9 0 1 4 THIRU 5 7 ; 2 / 6 "
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
CAR. RET.
LINE FEED
LTRS.
FIGS.
SPACE



Các loại mã thông dụng

❖ Các mã số học:

- Mã nhị phân: là mã trọng số.
- Mã quá 3: được tạo từ mã nhị phân tương ứng và cộng thêm 3.
- Mã Gray: Hai tổ hợp kế cận khác nhau một bit
- Mã thập phân hóa BCD (Binary Code Decimal) : dùng 4 bit để biểu diễn số thập phân và các loại khác của BCD: BCD5421, BCD2421, BCD5121, ….