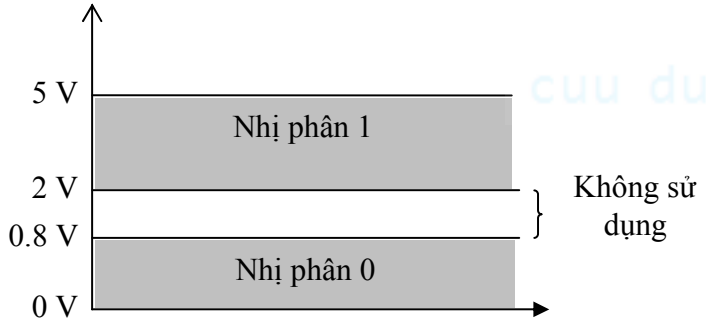
Chương III: Biểu diễn dữ liệu

3.1. Khái niệm thông tin

Để mã hóa thông tin trong máy tính, người ta dùng các tín hiệu điện thế. Thường tín hiệu trong khoảng 0→0.8V đại diện cho một giá trị (nhị phân 0) và tín hiệu có mức điện thế bất kỳ trong khoảng 2→5V đại diện cho giá trị kia (nhị phân 1). (xem hình 3.1.)



Hình 3.1. Biểu diễn trị nhị phân qua điện thế

Trong hình này, chúng ta quy ước có hai trạng thái có ý nghĩa: trạng thái thấp khi hiệu điện thế thấp hơn 0.8V và trạng thái cao khi hiệu điện thế lớn hơn 2V. Để có thông tin, ta phải xác định thời điểm ta quan sát trạng thái của tín hiệu. Thí dụ, tại thời điểm t_1 thì tín hiệu ở trạng thái thấp và tại thời điểm t_2 thì tín hiệu ở trạng thái cao.

3.2. Lượng thông tin và sự mã hoá thông tin

Thông tin được đo lường bằng đơn vị thông tin mà ta gọi là bit. Lượng thông tin được định nghĩa bởi công thức:

$$\mathbf{I = Log_2(N)}$$

Trong đó:

I: là lượng thông tin tính bằng bit

N: là số trạng thái có thể có

Vậy một bit ứng với một trạng thái trong hai trạng thái có thể có. Hay nói cách khác, một bit có thể biểu diễn hai trạng thái 0 hoặc 1. Ví dụ, để biểu diễn một trạng thái trong 8 trạng thái có thể có, ta cần một số bit ứng với một lượng thông tin là:

$I = Log_2(8) = 3$ bit

Tám trạng thái được ghi nhận nhờ 3 số nhị phân (mỗi số nhị phân có thể có giá trị 0 hoặc 1).

Như vậy lượng thông tin là số con số nhị phân cần thiết để biểu diễn số trạng thái có thể có. Do vậy, một con số nhị phân được gọi là một bit. Một từ n bit có thể tượng trưng một trạng thái trong tổng số 2^n trạng thái mà từ đó có thể tượng trưng.

Ví dụ: Nếu dùng 3 bit (A2,A1,A0) để biểu diễn thông tin, ta sẽ có được 8 trạng thái khác nhau như trong bảng 3.1.

Trạng thái	***A2***	***A1***	***A0***
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Bảng 3.1. Ba bit biểu diễn được 8 trạng thái

Như vậy trong máy tính thì mọi thứ đều được biểu diễn dưới dạng hai con số là 0 và 1. Nhưng ở thế giới thực của chúng ta thì thông tin lại là các khái niệm như con số, chữ cái, hình ảnh, âm thanh,….Cho nên để đưa các thông tin vào máy tính thì ta cần chuyển đổi thông tin thực thành những con số 0 và 1. Công việc này ta gọi là ***sự mã hóa thông tin***

Để biểu diễn dữ liệu trong máy tính chúng ta cần có các quy tắc "gắn kết" các khái niệm trong thế giới thật với một dãy gồm các con số 0 và 1.

3.3. Hệ Thống Số

Khái niệm hệ thống số:

Cơ sở của một hệ thống số định nghĩa phạm vi các giá trị có thể có của một chữ số. Ví dụ: trong hệ thập phân, một chữ số có giá trị từ 0-9, trong hệ nhị phân, một chữ số (một bit) chỉ có hai giá trị là 0 hoặc 1.

Dạng tổng quát để biểu diễn giá trị của một số:

$$V_k = \sum_{i=-m}^{n-1} b_i . k^i$$

Trong đó:

V_k: Số cần biểu diễn giá trị

m: số thứ tự của chữ số phần lẻ (phần lẻ của số có m chữ số được đánh số thứ tự từ -1 đến -m)

$n-1$: số thứ tự của chữ số phần nguyên (phần nguyên của số có n chữ số được đánh số thứ tự từ 0 đến n-1)

b_i: giá trị của chữ số thứ i

k: hệ số đếm(k=10: hệ thập phân; k=2: hệ nhị phân; k=8: hệ bát phân; ...).

Ví dụ: biểu diễn số 541.25_{10}

$541.25_{10} = 5 * 10^2 + 4 * 10^1 + 1 * 10^0 + 2 * 10^{-1} + 5 * 10^{-2}$

$= (500)_{10} + (40)_{10} + (1)_{10} + (2/10)_{10} + (5/100)_{10}$

Các hệ thống số cơ bản gồm:

- ***Thập phân (Decimal)***

  Dùng 10 chữ số 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 để biểu diễn số. Ví dụ số 235.3 trong hệ thập phân biểu diễn một đại lượng:

  2 1 0 -1 ←trọng số

  2 3 5 . 3 = $2*10^2 + 3*10^1 + 5*10^0 + 3*10^{-1}$

- ***Nhị phân (Binary)***

  Dùng hai chữ số 0 và 1 để biểu diễn số. Ví dụ số m = 1101,011 ở hệ nhị phân biểu diễn một đại lượng:

  $m_2 = 1.2^3 + 1.2^2 + 0.2^1 + 1.2^0 + 0.2^{-1} + 1.2^{-2} + 1.2^{-3}$

  Ở đây để tránh nhầm lẫn chúng ta dùng ký hiệu số nhỏ phía bên dưới để biểu diễn con số đó ở hệ nào, như m_2 – số m ở hệ nhị phân, 530_{10} – số 530 ở hệ thập phân.

- ***Bát phân (Octal)***

  Để biểu diễn số dùng 8 chữ số 0, 1, 2, 3, 4, 5, 6, 7. Ví dụ:

  $M = (6327,4051)_8 = 6.8^3 + 3.8^2 + 2.8^1 + 7.8^0 + 4.8^{-1} + 0.8^{-2} + 5.8^{-3} + 1.8^{-4}$

  - ***Chú ý:*** *Mỗi con số ở hệ bát phân được biểu diễn bằng tập hợp 3 số ở hệ nhị phân (ví dụ $5_8 = 101_2$)*

- **Thập lục phân (Hexadecimal)**

  Để biểu diễn số dùng 16 chữ số: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E. Trong đó tương đương với hệ 10 thì A=10, B=11, C=12, D=13, E=14, F=15.

  - ***Chú ý:*** *Mỗi con số ở hệ thập lục phân được biểu diễn bằng tập hợp 4 số ở hệ nhị phân (ví dụ $5_{16} = 0101_2$)*

Bảng 3.2 cho ta các đặc tính chính của các hệ đếm cơ bản.

Như vậy có nhiều hệ đếm khác nhau được dùng để biểu diễn dữ liệu và chúng ta sẽ xem xét cách chuyển đổi giữa các hệ này với nhau như thế nào sau đây.

HỆ ĐẾM	CƠ SỐ	KÍ HIỆU CHỮ SỐ	TRỌNG SỐ	VÍ DỤ
Nhị phân	2	0, 1	2^i	1001,1101
Bát phân	8	0,1,2,3,4,5,6,7	8^i	3567,24
Thập phân	10	0,1,2,3,4,5,6,7,8,9	10^i	1369,354
Thập lục phân	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F	16^i	3FA9,6B

Bảng 3.2. Các hệ đếm cơ bản

a) Chuyển đổi từ hệ cơ số 10 sang *b*

Quy tắc: Chia số cần đổi cho ***b***, lấy kết quả chia tiếp cho ***b*** cho đến khi **kết quả** bằng **0**. Số ở cơ số ***b*** chính là các số dư (của phép chia) viết ngược.

Ví dụ 1: Chuyển số 41 ở hệ 10 sang hệ 16

41 ÷ 16 = 2 dư 9
2 ÷ 16 = 0 dư 2
=> $41_{10} = 29_{16}$

Ví dụ 2: Chuyển số 41 ở hệ 10 sang hệ 2

41 ÷ 2 = 20 dư 1
20 ÷ 2 = 10 dư 0
10 ÷ 2 = 5 dư 0
5 ÷ 2 = 2 dư 1
2 ÷ 2 = 1 dư 0
1 ÷ 2 = 0 dư 1
=> $41_{10} = 101001_2$ (chú ý!!! viết ngược từ dưới lên)

Vì chúng ta cần biểu diễn dữ liệu ở hệ nhị phân, nên việc ***Chuyển đổi hệ 10 sang Nhị phân*** cần được đặc biệt lưu ý riêng như sau:

Quy tắc: Người ta chuyển đổi từng phần nguyên và lẻ theo quy tắc sau:

Phần nguyên: Chia liên tiếp phần nguyên cho 2 giữ lại các số dư, số nhị phân được chuyển đổi sẽ là dãy số dư liên tiếp tính từ lần chia cuối về lần chia đầu tiên.

Phần lẻ: Nhân liên tiếp phần lẻ cho 2, giữ lại các phần nguyên được tạo thành. Phần lẻ của số Nhị phân sẽ là dãy liên tiếp phần nguyên sinh ra sau mỗi phép nhân ***tính từ lần nhân đầu đến lần nhân cuối.***

Ví dụ 3: Chuyển sang hệ Nhị phân số: 13,625

Thực hiện:

Phần nguyên:

13:2 = 6 dư 1
6:2 = 3 dư 0
3:2 = 1 dư 1
1:2 = 0 dư 1

Phần nguyên của số Nhị phân là 1101

Phần lẻ:

0,6875 x 2 = 1,375 Phần nguyên là 1
0,375 x 2 = 0,750 Phần nguyên là 0
0,750 x 2 = 1,500 Phần nguyên là 1
0,5 x 2 = 1,00 Phần nguyên là 1

Phần lẻ của số Nhị phân là: 0,1011

Ta viết kết quả là: $(13,625)_{10} = (1101,1011)_2$

Chú ý: việc chuyển đổi từ hệ thập phân sang hệ Nhị phân không phải luôn được gọn gàng chính xác, trong trường hợp phép tính chuyển đổi kéo dài, thì tùy theo yêu cầu về độ chính xác mà ta có thể dùng phép tính ở mức độ cần thiết thích hợp.

Ví dụ 4: Chuyển số $(3287,5100098)_{10}$ sang Cơ số 8.

Phần nguyên:

3287:8 = 410 dư 7
410:8 = 51 dư 2
51:8 = 6 dư 3
6:8 = 0 dư 6
Vậy $(3287)_{10}=(6327)_8$

Phần lẻ:

0,5100098x8 = 4,0800784 phần nguyên là 4
0,0800784x8= 0,6406272 phần nguyên là 0
0,6406270x8= 5,1250176 phần nguyên là 5
0,1250176x8= 1,0001408 phần nguyên là 1
Vậy $(0,5100098)_{10}=(0,4051)_8$

Kết quả chung là: $(3287,5100098)_{10} =(6327,4051)_8$

b) Chuyển đổi từ hệ cơ số *b* sang 10

Việc chuyển đổi từ một hệ cơ số bất kỳ sang hệ 10 thì đơn giản hơn và cách làm như trong trường hợp định nghĩa đại lượng của số đó.

Ví dụ 1: số 235.3 trong hệ 8 chuyển sang hệ thập phân có giá trị như sau:

2 1 0 -1 ←trọng số
2 3 5 . 3 $= 2*8^2 + 3*8^1 + 5*8^0 + 3*8^{-1} = 157.375_{10}$

➢ **Chuyển đổi Hệ 2 sang hệ 10**

Quy tắc: Muốn chuyển đổi một số biểu diễn trong hệ Nhị phân sang hệ thập phân ta lập Tổng theo trọng số của từng bit Nhị phân, Kết quả của tổng sẽ là biểu diễn Thập phân của số đó.

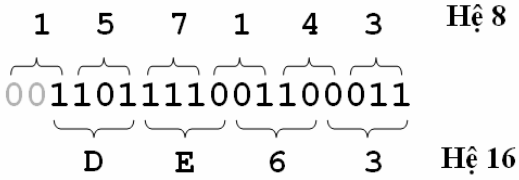
Ví dụ 2: Chuyển đổi sang hệ Thập phân số: **m = 1101,011**

Thực hiện: Ta lập tổng theo trọng số của từng Bit nhị phân:
$m = 1.2^3 + 1.2^2 + 0.2^1 + 1.2^0 + 0.2^{-1} + 1.2^{-2} + 1.2^{-3}$

m = 8 + 4 + 0 + 1 + 0 + 1/4 + 1/8
m = 13,375

c) Chuyển đổi cơ số 2-8-16

***Quy tắc*:** Từ phải sang trái, gom **3** chữ số nhị phân thành một chữ số **bát phân** hoặc gom **4** chữ số nhị phân thành một chữ số **thập lục phân**.



Bảng 3.3 cho ta các chuyển đổi tương ứng từ các hệ số với nhau. Để làm bài tốt và học tốt các môn học liên quan đến kỹ thuật số, hệ thống số, vi xử lý,... sinh viên cần thuộc lòng bảng này.

Hệ 2 (Base 2)	Hệ bát phân (Base 8)	Hệ thập phân (Base 10)	Hệ thập lục phân (Base 16)
0000	0	0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7
1000	10	8	8
1001	11	9	9
1010	12	10	A
1011	13	11	B
1100	14	12	C

1101	15	13	D
1110	16	14	E
1111	17	15	F

Bảng 3.3. Tương quan giữa các hệ thống số

Ví dụ 1: Chuyển số M = $(574,321)_8$ sang biểu diễn nhị phân.

Thực hiện: Thay mỗi chữ số bằng nhóm nhị phân 3 bit tương ứng:

M = 101 111 100 , 011 010 001
5 7 4 3 2 1

=>M_2 = 101111100,011010001

Ví dụ 2: Chuyển số M = (1001110,101001)2 sang cơ số 8.

Thực hiện: M = 1 001 110 , 101 001
M = 1 1 6 , 5 1
=> M = $(116,51)_8$

3.4. Các phép tính số học cho hệ nhị phân

Các phép tính Cộng, Trừ, Nhân, Chia cũng được sử dụng trong số học Nhị phân, việc tính toán cụ thể được thực hiện theo quy tắc sau:

3.4.1. Phép cộng hai số nhị phân không dấu:

Cộng nhị phân được thực hiện theo quy tắc ở bảng 3.4

Chú ý:

- Khi cộng, thực hiện từ bit có trọng số thấp đến bit cú trọng số cao.
- Nếu có số nhớ thì số nhớ sinh ra được cộng vào bit cú trọng số cao hơn liền kề

SỐ HẠNG 1	SỐ HẠNG 2	TỔNG	SỐ NHỚ	KẾT QUẢ

0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	0	1	10

Bảng 3.4. Quy tắc Cộng Nhị phân cho 2 số 1 bit.

Ví dụ: Thực hiện các phép Cộng Nhị phân:

$$\begin{array}{r} 1011 \\ +1100 \\ \hline 10111 \end{array}$$

3.4.2. Phép trừ hai số nhị phân không dấu:

Phép trừ nhị phân được thực hiện theo quy tắc trình bày ở Bảng 3.5

SỐ BỊ TRỪ	SỐ TRỪ	HIỆU SỐ	SỐ VAY
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Bảng 3.5. Quy tắc trừ Nhị phân cho 2 số 1 bit.

Chú ý:

- Phép tính được thực hiện từ Bit có trọng số thấp đến Bit có trọng số cao.
- Số vay sẽ được trừ vào Bit có trọng số cao hơn ở liền kề.

Ví dụ: Thực hiện các tính Trừ Nhị phân sau:

1011

```
-0110
------
 0101
```

Tuy nhiên trong thực tế, máy tính không tính toán kiểu đó mà chuyển đổi phép trừ thành phép cộng với ***số bù 2*** của nó. Phương pháp này trong máy tính được cho là hiệu quả hơn và dễ dàng thiết kế phần cứng cho nó hơn. Số bù có hai loại thường dùng là số bù 1 và số bù 2.

3.4.3. Phép nhân và chia hai số nhị phân không dấu:

- ***Phép nhân*** nhị phân được thực hiện như nhân thập phân.

Ví dụ: Có phép tính: 1001 nhân với 1101

Ta thực hiện:
```
      1001(Số bị nhân-Multiplicant)
x     1101(Số nhân-Multiplier )
------
      1001
     0000
+   1001
    1001
------
1110101
```
Kết quả là:

- ***Phép chia*** nhị phân được thực hiện như chia thập phân.

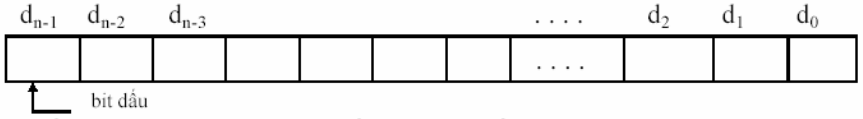
Ví dụ: Có phép tính: 1110101 chia cho 1001

Ta thực hiện:
```
1110101 : 1001
-       1001      1101
        01011
-         1001
          001001
            1001
            0000
```
Kết quả: 1111010 : 1001 = 1101

3.4.4. Biểu diễn số nguyên có dấu

Có ba cách để biểu diễn một số nguyên n bit có dấu đó là biểu diễn bằng trị tuyệt đối và dấu, biểu diễn bằng số bù 1, biểu diễn bằng số bù 2.
Cách thông thường nhất là biểu diễn bằng trị tuyệt đối và dấu, trong trường hợp này thì bit cao nhất luôn tượng trưng cho dấu.

Khi đó, bit dấu có giá trị là 0 thì số đó là nguyên dương, bit dấu có giá trị là 1 thì số đó là nguyên âm. Tuy nhiên, cách biểu diễn dấu này không đúng trong trường hợp số được biểu diễn bằng số thừa K mà ta sẽ xét ở phần sau trong chương này.



Số nguyên có bit d_{n-1} là bit dấu và có trị tuyệt đối biểu diễn bởi các bit từ d_0 tới d_{n-2} .
Ví dụ: dùng 8 bit biểu diễn số +25 và -25 dưới dạng dấu và trị tuyệt đối.

Ta biết $25_{10} = 11001_2$. Dùng 8 bit để biểu diễn số, như vậy 1 bit biểu diễn dấu, còn 7 bits biểu diễn giá trị của số đó.

Vậy +25 = **0** 0011001
Bit dấu
Vậy -25 = **1** 0011001
Bit dấu

Như vậy, theo cách này thì:
$+25_{10} = 00011001_2$
$-25_{10} = 10011001_2$

- Một Byte (8 bit) có thể biểu diễn các số có dấu từ -127 tới +127.
- Có hai cách biểu diễn số không là 0000 0000 (+0) và 1000 0000 (-0).

Hai cách biểu diễn còn lại ta sẽ xem xét trong phần tiếp dưới đây.

3.4.5. Số bù của một số

Số bù được dùng trong máy tính số giúp đơn giản phép toán trừ và các thao tác logic. Trong hệ cơ số (hệ đếm) ***r*** có hai dạng số bù: Số bù ***r*** và số bù (***r***-1). Như vậy trong hệ 10 sẽ có bù 10 và bù 9, trong hệ 8 có bù 8 và bù 7, trong hệ 16 có bù 16 và bù 15, trong hệ 2 có bù 2 và bù 1.

Số bù r-1 của một số:

Giả sử *N* là một số có n ký số trong hệ thống số *r* thì bù *r-1* của $N = (r^n - 1) - N$.

Ví dụ đối với hệ thập phân ta có bù *r-1* = bù 9 của số thập phân *N* là $(10^n-1) - N$. Trong đó 10^n là một số gồm số 1 và theo sau là n chữ số 0 (ví dụ $10^4 =$ 10000). Vậy (10^n-1) là gồm n số 9 (ví dụ $10^4 -1 = 9999$). Vậy bù 9 của *N* là một con số nhận được bằng cách lấy trừ 9 cho từng ký số của *N*.

Ví dụ:

Bù 9 của 43520 là 99999 – 43520 = 56479

Số bù r của một số:

Số bù r của một số được tính bằng bù *r-1* cộng với 1. Như vậy bù 10 của 43520 là 56478 + 1 = 56480. Từ đây ta có thể tính nhanh bù 10 theo qui tắc:

- Giữ nguyên các ký số 0 bên phải cho đến khi gặp ký số khác 0
- Lấy 10 trừ đi ký số đầu tiên khác 0 đó
- Lấy 9 trừ đi các số còn lại

Ví dụ: Bù 10 của 347200 là 652800

Chúng ta sẽ chủ yếu làm việc với hệ nhị phân, do máy tính làm việc với hệ này. Trong hệ nhị phân sẽ có bù 1 và bù 2 mà ta sẽ xem xét sau đây.

Số bù 1 của một số:

Số bù 1 của một số nhị phân (hay còn gọi là số invert) là một số nhị phân có được bằng cách đổi các bit 1 thành 0 và bit 0 thành 1.

Ví dụ:

Số cần đổi	10110101	1100110
Số bù 1 của nó	10001010	0011001

Số bù 2 của một số: Số bù hai của một số là số bù 1 của số đó cộng thêm 1.

Ví dụ:

Số:	01001110	00110101
Số bù một của nó là:	10110001	11001010
Cộng thêm 1	+1	+1
Bù hai của nó là:	10110010	11001011

Quy tắc chung tìm bù hai của một số:

- Muốn tìm bù 2 của một số ta đi từ bit có trọng số nhỏ nhất ngược lên.
- Khi nào gặp được bit 1 đầu tiên thì giữ nguyên các số 0 bên phải số 1 đó và cả số 1 đó nữa, còn tất cả các bít bên trái số 1 đó thì đảo lại.

Ví dụ:

Số:	01100100	10010010	1101000	01100111
Số bù 2 là:	10011100	01101110	0011000	10011001

Sau khi ta đã biết về số bù 1 và bù 2 của một số nhị phân, ta xem cách biểu diễn một số nguyên có dấu theo hai cách này.

- **Số nguyên có dấu được biểu diễn ở dạng bù 1 là:**

- Đối với số dương thì biểu diễn giống dấu và trị tuyệt đối
- Đối với số âm thì được biểu diễn dưới dạng bit dấu và giá trị của số đó ở dạng bù 1.

Ví dụ: Dùng 8 bit biểu diễn số +25 và -25 dưới dạng bù 1.

Ta biết $25_{10} = 11001_2$. Dùng 8 bit để biểu diễn số, như vậy 1 bit biểu diễn dấu, còn 7 bits biểu diễn giá trị của số đó, nếu là số âm thì lấy bù 1 các bit này.

Vậy +25 = **0** 0011001
Bit dấu

Vậy -25 = **1** 1100110
Bit dấu

Ta cũng có thể hiểu là số âm được biểu diễn bằng cách lấy bù 1 của số dương kể cả bit dấu.

- **Số nguyên có dấu được biểu diễn ở dạng bù 2 là:**
  - Đối với số dương thì biểu diễn giống dấu và trị tuyệt đối
  - Đối với số âm thì được biểu diễn dưới dạng bit dấu và giá trị của số đó ở dạng bù 2.

Ví dụ: Dùng 8 bit biểu diễn số +25 và -25 dưới dạng bù 2.

Ta biết $25_{10} = 11001_2$. Dùng 8 bit để biểu diễn số, như vậy 1 bit biểu diễn dấu, còn 7 bits biểu diễn giá trị của số đó, nếu là số âm thì lấy bù 2 các bit này.

Vậy +25 = **0** 0011001
Bit dấu

Vậy -25 = **1** 1100111
Bit dấu

- ***Chú ý: Số dương biểu diễn ở cả 3 cách là như nhau, chỉ khác nhau khi đó là số âm***

3.4.6. Phép cộng trừ nhị phân dùng bù 1

Số có dấu được biểu diễn bằng bù 1 theo qui tắc sau:
- Bít lớn nhất (MSB) là bít dấu, trong đó 0 là số dương và 1 là số âm
- Các bít còn lại biểu diễn trị thực của số dương hoặc trị bù 1 của số âm.

Ví dụ: Dùng 6 bit gồm cả bít dấu để biểu diễn số

17 = 010001 26 = 011010
-17 = 101110 -26 = 100101

Thực hiện phép cộng giống như cộng các số nhị phân không dấu, cộng cả bit dấu. Cần lưu ý một điểm nhỏ là cộng số nhớ của bit lớn nhất vào bit cuối cùng (LSB).

Ví dụ:

```
   13          001101          -13          110010
 +             +             +            +
   11          001011          -11          110100
 ____        ________        ____         ________
   24          011000          -24          100110
                                          +       1
                                          ________
                                          100111
```

Trong kết quả nếu bit dấu là 0 thì dãy bit sau bit dấu là giá trị kết quả, còn nếu bit dấu là âm thì dãy bit sau bit dấu mới là kết quả ở dạng bù một. Muốn biết giá trị thực của kết quả ta phải lấy bù 1 của kết quả một lần nữa. Như trong ví dụ trên kết quả của phép cộng thứ nhất là 011000 có bit dấu là 0, vậy giá trị thực của kết quả là +11000 = +24. Còn phép cộng thứ hai có kết quả là 100111 có bit dấu là 1, vậy giá trị thực của kết quả là –(bù 1 của 00111) = -11000 = -24.

3.4.7.Phép cộng trừ nhị phân dùng bù 2

Số có dấu được biểu diễn bằng bù 2 theo qui tắc sau:
- Bít lớn nhất (MSB) là bít dấu, trong đó 0 là số dương và 1 là số âm
- Các bít còn lại biểu diễn trị thực của số dương hoặc trị bù 2 của số âm.

Thực hiện phép cộng giống như cộng các số nhị phân không dấu, cộng cả bit dấu. Cần lưu ý loại bỏ bit nhớ cuối cùng trong kết quả.

Ví dụ:

-12	110100
+ -9	110111
-21	1 101011

Kết quả có bit dấu bằng 1, vậy kết quả là số âm và dãy bit mới chỉ là bù 2 của kết quả. Muốn có kết quả thật ta lấy Bù 2 một lần nữa. Trong ví dụ trên kết quả không tính đến bit nhớ (bỏ bit nhớ) là 101011 có bit dấu bằng 1 là một số âm, tức là kết quả mới biểu diễn ở dạng bù 2. Muốn biết giá trị thật của kết quả ta phải lấy bù 2 một lần nữa. Tức là 101011 = - (bù 2 của 01011) = - (10101) = -21.

Đối với phép trừ ta thực hiện thông qua phép cộng.

-A = bù 2 của A
A – B = A + (-B) = A + (bù 2 của B)

Ví dụ 1: 13 – 6 = 13 + (-6)

6	=	00000110
-6	=	11111010
13	=	00001101
	=	1 00000111 (7)

Ví dụ 2: Thực hiện phép tính: 0111 – 0101

Ta thực hiện: 0111 Chuyển thành 0111

-0101 +1011 (Số bù 2 của 0101)
10010 Suy ra kết quả là 0010

Ví dụ 3: Thực hiện phép tính: 0101 – 0111

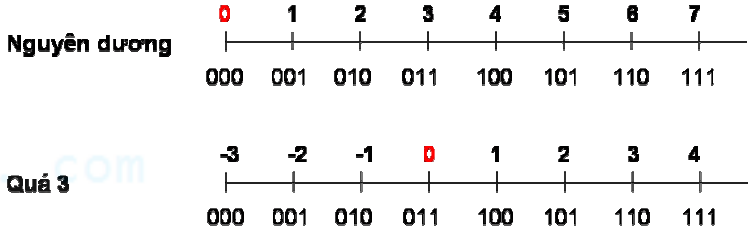
Ta thực hiện: 0101(5) Chuyển thành 0101
-0111(-7) +1001 (Số bù 2 của 0111)
1110

3.5. Số quá n (excess-n)

Số quá n hay còn gọi là số thừa n của một số N có được bằng cách "cộng thêm" số N với số quá n, số n được chọn sao cho tổng của n và một số âm bất kỳ luôn luôn dương.

Quy tắc chung:

Biểu diễn quá n của N = biểu diễn nguyên dương của (N + n)



Ví dụ:
Biểu diễn (quá 127) của 7 là:

$$127+7 = 134 = 10000110_2$$

Cách biểu diễn số nguyên có dấu bằng số bù 2 được dùng rộng rãi cho các phép tính số nguyên. Nó có lợi là không cần thuật toán đặc biệt nào cho các phép tính cộng và tính trừ, và giúp phát hiện dễ dàng các trường hợp bị tràn.

Các cách biểu diễn bằng "dấu , trị tuyệt đối" hoặc bằng "số bù 1" dẫn đến việc dùng các thuật toán phức tạp và bất lợi vì luôn có hai cách biểu diễn của số không.

Cách biểu diễn bằng "dấu , trị tuyệt đối" được dùng cho phép nhân của số có dấu chấm động.

Cách biểu diễn bằng số quá n được dùng cho số mũ của các số có dấu chấm động. Cách này làm cho việc so sánh các số mũ có dấu khác nhau trở thành việc so sánh các số nguyên dương.

3.6. Cách biểu diễn số với dấu chấm động

Để biểu diễn các con số rất lớn hoặc rất bé, người ta người ta dùng một cách biểu diễn số gọi là số chấm động (*floating point number*). Trước khi đi vào cách biểu diễn số với dấu chấm động, chúng ta xét đến cách biểu diễn một số dưới dạng dấu chấm xác định.

Ví dụ:

- Trong hệ thập phân, số 254_{10} có thể biểu diễn dưới các dạng sau:

$254 * 10^0$; $25.4 * 10^1$; $2.54 * 10^2$; $0.254 * 10^3$; $0.0254 * 10^4$; …

- Trong hệ nhị phân, số $(0.00011)_2$ (tương đương với số 0.09375_{10}) có thể biểu diễn dưới các dạng :

$0.00011 * 2^0$; $0.0011 * 2^{-1}$; $0.011 * 2^{-2}$; $0.11 * 2^{-3}$; $1.1 * 2^{-4}$;…

Các cách biểu diễn này gây khó khăn trong một số phép so sánh các số. Để dễ dàng trong các phép tính, các số được chuẩn hoá về một dạng biểu diễn:

$$\pm 1.\text{fff...f} \times 2^{\pm E}$$

đối với hệ nhị phân, trong đó: f là phần lẻ; *E* là phần mũ.

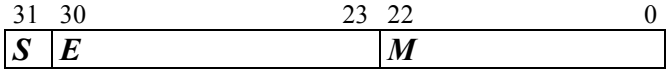
- *Đối với các hệ khác thì biểu diễn chấm động được gọi là **chuẩn hóa** khi phần định trị chỉ có duy nhất **một** chữ số bên trái dấu chấm thập phân và chữ số đó khác không → một số chỉ có duy nhất một biểu diễn chấm động được chuẩn hóa.*

Ví dụ:
2.006×10^3 (chuẩn)
20.06×10^2 (không)
0.2006×10^4 (không)

Các thành phần của số chấm động bao gồm: phần dấu, phần mũ và phần định trị. Như vậy, cách này cho phép biểu diễn gần đúng các số thực, tất cả các số đều có cùng cách biểu diễn.

Có nhiều cách biểu diễn dấu chấm động, trong đó cách biểu diễn theo chuẩn IEEE 754 được dùng rộng rãi trong khoa học máy tính hiện nay. Trong cách này một số được biểu diễn dưới dạng :

$$F = (-1)^S * M * R^E$$



Hình 3.2. Biểu diễn số có dấu chấm động chính xác đơn với 32 bit

Trong đó: *S*: dấu (Sign bit), ***M***: định trị, ***R***: cơ số, *E*: mũ (Exponent)

- Dấu: 1 bit (0 – dương, 1 – âm)
- Mũ: 8 bit (từ bit 23 đến bit 30) là một số quá 127 (sẽ có trị từ -127 đến 128, tức là từ 00000000 đến 11111111)
- Không biểu diễn cơ số (*R*) vì luôn bằng 2
- Phần định trị *M* 23 bit (từ bit 0 đến bit 22) **chỉ biểu diễn phần lẻ** (bên phải dấu chấm số nhị phân) vì chữ số bên trái dấu chấm luôn là 1.

Ví dụ:

a) $2006_{10} = (-1)^0 * 2.006 * 10^3$
b) $209.8125_{10} = 11010001.1101_2$

$= 1.10100011101 * 2^7$

Biểu diễn (quá 127) của 7 là:

$127+7 = 134 = 10000110_2$

Kết quả: 0 10000110 10100011101000000000

0	10000110	10100011101000000000

Các phép tính với số chấm động phức tạp hơn nhiều là với số chấm tĩnh, thực hiện lâu hơn và phần cứng cho nó cũng phức tạp hơn. Máy tính không có phần cứng tính toán số chấm động, nhưng có các tập trình con giúp giải các bà toán với số chấm động.

3.7. Biểu diễn số BCD

Con người thường quen với hệ thập phân, trong khi máy tính lại chỉ thích hợp với hệ nhị phân. Do đó khi nhập xuất dữ liệu thường là nhập xuất theo dạng thập phân. Nếu việc nhập xuất số thập phân không nhiều thì có thể chuyển số hệ 10 khi nhập sang hệ 2, tính toán xong theo hệ 2 rồi lại chuyển ngược lại sang hệ 10 trước khi xuất ra ngoài. Nếu nhập xuất nhiều thì việc chuyển đổi sẽ làm mất nhiều thời gian xử lý. Mặt khác một vài ứng dụng, đặc biệt ứng dụng quản lý, bắt buộc các phép tính thập phân phải chính xác, không làm tròn số. Với một số bit cố định, ta không thể đổi một cách chính xác số nhị phân thành số thập phân và ngược lại.

Vì vậy, khi cần phải dùng số thập phân, ta có thể dùng một cách khác, đó là cách biểu diễn số thập phân mã bằng nhị phân (***BCD: Binary Coded Decimal***). Theo đó mỗi số thập phân nhập vào máy sẽ được mã hóa theo dạng ***BCD*** bằng cách chuyển mỗi ký số hệ 10 thành 4 bit số nhị phân như trong bảng 3.6. Sau đó việc tính toán sẽ thực hiện trực tiếp trên mã ***BCD*** . Tính toán xong thì lại chuyển ra ngoài ttheo dạng thập phân. Khi đó, nến việc tính toán là không nhiều, hoặc việc tính toán là đơn giản thì số ***BCD*** sẽ giúp cải thiện đáng kể tốc độ xử lý.

Số hệ 10	**Số BCD**
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Bảng 3.6. Số thập phân mã BCD

Biểu diễn số dạng BCD sẽ tốn kém hơn nhiều biểu diễn dạng nhị phân vì mỗi số BCD cần tới 4 bit. Ví dụ 3257 có dạng BCD là 0011 0010 0101 0111, tức là phải dùng 16 bit, trong khi ở hệ nhị phân chỉ cần 12 bit (110010111001). Con số càng lớn thì sự chênh lệnh của nó càng nhiều, trong khi bộ nhớ thì có hạn, cho nên đây là một nhược điểm rất lớn của dạng số BCD.

Để thiết kế mạch tính toán thập phân cũng đòi hỏi độ phức tạp nhiều hơn, tuy nhiên nó có thuận lợi là việc tính toán đều bằng thập phân và cho kết quả chính xác hơn.

Một số ứng dụng như xử lý dữ liệu thương mại - kinh tế thường tính toán ít hơn so với khối dữ liệu nhập xuất. Vì vậy mà một số máy và các máy tính tay đều tính toán trực tiếp trên số thập phân. Một số máy khác lại có khả năng tính toán trên cả thập phân và nhị phân.

Điểm khác biệt rõ nhất với các hệ khác khi tính toán là khi kết quả cộng nếu các ký số vượt quá kết quả cho phép trong khoảng từ ***0000*** đến ***1001*** hoặc có nhớ khi cộng thì phải sửa sai bằng cách cộng thêm ***0110*** vào ký số bị sai.

Hai ví dụ sau đây sẽ cho thấy điều này.

Ví dụ 1:

```
  27      0010  0111
+ 36      0011  0110
  63    ---------------
        0101  1101   ──→ Ký số vượt quá => kết quả sai

        0000  0110   ──→ Sửa sai kết quả
        0110  0011   Kết quả = 63
```

Trong ví dụ này ta thấy khi cộng hai số 6 với 7 đã cho ta kết quả là 13 (1101). Kết quả này đã vượt qua con số lớn nhất trang hệ BCD là 1001 (9), do đó để sửa lỗi ta phải cộng thêm một giá trị 0110 vào đúng vị trí số cộng sai đó và nếu có số nhớ thì số nhớ đó sẽ được cộng sang số bên cạnh trái.

Ví dụ 2:

```
  28      0010  1000
+ 59      0101  1001
  87    ---------------
        1000  0001   ──→ Có nhớ 1 => kết quả sai

        0000  0110   ──→ Sửa sai kết quả
        1000  0111   Kết quả = 87
```

Tương tự khi trừ số *BCD,* nếu có mượn khi trừ thì cũng phải sửa sai bằng cách trừ bớt ***0110*** vào ký số bị sai như trong ví dụ sau:

```
  61      0110  0001
- 38      0011  1000
  23    ---------------
        0010  1001   ──→ Ký số bên phải mượn 1 khi trừ

        0000  0110   ──→ Sửa sai kết quả
        0010  0011   Kết quả = 23
```


Từ các ví dụ trên ta thấy nhiều khi các phép tính cứ phải sửa sai như vậy thì sẽ dẫn đến tốc độ tính toán cũng bị giảm bớt

3.8. Biểu diễn các ký tự

Ngoài việc biểu diễn số, chúng ta cũng cần đến biểu diễn chữ và một số ký tự khác. Tuỳ theo các hệ thống khác nhau, có thể sử dụng các bảng mã khác nhau:

- **ASCII** (7 bit) (American Standard Codes for Information Interchange) để biểu diễn 128 ký tự gọi là mã ASCII-7
- **ASCII** mở rộng (8 bit) để biểu diễn 256 ký tự
  - 00 – 1F: ký tự điều khiển
  - 20 – 7F: ký tự in được
  - 80 – FF: ký tự mở rộng (ký hiệu tiền tệ, vẽ khung, …)
- **Unicode:** Ngày nay do việc sử dụng rộng rãi mạng toàn cầu Internet với rất nhiều ngôn ngữ khác nhau, rất nhiều ký tự khác nhau nên người ta đã chuyển sang dùng bộ mã Unicode (16 bit) (UTF-8) có thể biểu diễn được tới 65.536 ký tự và như vậy cho phép biểu diễn hầu hết các ngôn ngữ trên thế giới.