

CH 3 補數系統 Complement

- A. 目的 – 以加法實現減法
- B. 種類與定義 – b 進位制 $\rightarrow (b - 1)$'s 補數 $+ 1 = b$'s 補數；最大值 – 欲轉換之數
 $= B - 1$ 補數
- C. 無號數(2 進位)
- D. 有號數(2 進位) – 正：最左為 0，其他不變；負：sign magnitude：最左位數判斷 0 正 1 負；1' complement: 對正數(除了最左位以外的數)1 換 0，反之；2' complement: 1 補數 $+ 1$
- E. 四則運算(Sign magnitude)
- F. 1's Complement – 值域： $-(2^n - 1) \sim 2^n - 1 \rightarrow$ 有 +0、-0；進位：端迴進位(左邊進位給右邊)；沒進位：加負號，取 1 補數(將補數轉換為整數)
- G. 2's Complement – 值域： $-(2^n) \sim 2^n - 1$ ；進位：左邊捨去；沒進位：加負號，取 2 補數(將補數轉換為整數)
- H. Excess Code – $x + 2^{n-1}$ (存入)；反其值，扣回去

1. 目的

- 簡化減法運算，即是用加法來實施減法。

- 例： $A - B \Rightarrow A + (B \text{ 的補數})$

- 優點：① 不須要提供減法電路，只留一套加法電路， \therefore 電路成本可降低。

② 此外也可用來表示正、負整數之格式 eg. Computer 內部使用 2's 補數系統

2. 種類與定義

(-) 若 b 為基座 (Base → 控制), 則分為 2 種不補數

e.g. 10进制 \Rightarrow $\begin{cases} 9's \text{ 补数} \\ b=10 \\ 10's \text{ 补数} \end{cases}$ 16进制 \Rightarrow $\begin{cases} 15's \\ 16's \end{cases}$ 2进制 \Rightarrow $\begin{cases} 1's \text{ 补数} \\ 2's \text{ 补数} \end{cases}$

(二) 定義: 令 $(x)_b$ 為一數值, 且 x 有 n 位整數, m 位小數. 則

[2] x 的 b 's 補數

$$= \begin{cases} 0, & \text{if } x=0 \\ b^n - (x)_b, & \text{otherwise} \end{cases}$$

⇒ B-1 補數有正負 0 所以要再扣最小位數次方之值，B-1 補數與其原來的數將加不會超過該位數之最大值，看下列例子

Ex. $+3(011) + (-3)(100) = 0(111) \rightarrow 111$ 為 3 位數中最大之值

⇒ N 的 $B-1$ 補數之值最右位數+1 = B 補數

- 補充

(1) x 之補數之整數、小數位數與 x 一樣。(即不要刪去 0)

(2) 補數是一種 format (格式)

(3) 依定義: b 's 補數 = $(b-1)$'s 補數 + $\frac{b^{-m}}{b}$
 (4) 試求: 證明「此法」
 ↳ 最右 (1) 之位數直接加 1

(4) 補補還原 [考足義]

对 x 之 $(b-1)$'s 補數再取 $(b-1)$ 補數會得到 x .

Pf: x 的 $(b-1)$ 補數 $= b^n - b^{-m} - x$

对它再取 $(b-1)$'s 補數, 即 $b^n - b^{-m} - (b^n - b^{-m} - x) = x$

同理, 对 x 之 b 's 補數, 取 b 's 補數也得到 x .

- 例子

求 $(2019.72)_{10}$ 之 $\begin{cases} 9's \text{ 補數} \\ 10's \text{ 補數} \end{cases}$

Ans: $x = (2019.72)_{10}$, 整數位數 = 4 位, 小數位數 = 2 位.

(1) $9's$ 補數

$$\begin{aligned} &= b^n - b^m - (x)_b \\ &= 10^4 - 10^{-2} - (2019.72)_{10} \\ &= (10000 - 0.01) - 2019.72 \\ &= 7980.27 \end{aligned}$$

(2) $10's$ 補數

$$\begin{aligned} &= b^n - (x)_b \\ &= 10^4 - (2019.72)_{10} \\ &= (7980.28)_{10} \\ &= \text{即 } 9's \text{ 補數最右位數加 } 1. \end{aligned}$$

- 快速做法

\Rightarrow 最大值 - 欲轉換之數 = $B - 1$ 補數

$\Rightarrow B$ 補數 = $B - 1$ 補數最右位數 + 1

(1) $(2019.72)_{10}$ 之 $9's$ 補數

$$\begin{array}{r} \Rightarrow \begin{array}{r} 9999.99 \\ - 2019.72 \\ \hline (7980.27)_{10} \end{array} \quad \text{而 } 10's \text{ 補} = (7980.28)_{10} \end{array}$$

(2) $(7420.36)_8$

$$\begin{aligned} \text{的 } \begin{cases} 7's \text{ 補數} = (0357.416)_8 \\ 8's \text{ 補數} = (0357.417)_8 \end{cases} & \begin{array}{r} 7777.777 \\ - 7420.36 \\ \hline 0357.416 \\ \text{不可用!} \end{array} \end{aligned}$$

(3) $(A2C6.13)_{16}$

$$\begin{aligned} \text{的 } \begin{cases} 15's \text{ 補數} = (5D39.EC)_{16} \\ 16's \text{ 補數} = (5D39.ED)_{16} \end{cases} & \begin{array}{r} 15 \ 15 \ 15 \ 15 \ 15 \ 15 \\ - A \ 2 \ C \ 6 \ . \ 1 \ 3 \\ \hline 5 \ D \ 3 \ 9 \ . \ E \ C \end{array} \end{aligned}$$

(4) $(101101.011)_2 \rightarrow 1 \text{ 變 } 0, 0 \text{ 變 } 1$

$$\begin{aligned} \text{的 } \begin{cases} 1's \text{ 補數} = (010010.100)_2 \\ 2's \text{ 補數} = (010010.101)_2 \end{cases} \end{aligned}$$

3. 無號數(Unsinged Number)

(一) Def: 無正負符號, 值 ≥ 0 開始, 以整數而言, 若給予 N bits, 值域: $0 \sim 2^N - 1$

(二) 例: 8 bits 之無符號, 值域: $0 \sim 2^8 - 1 \Rightarrow 0 \sim 255$

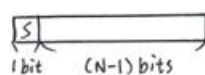
例: 8 bits, $(125)_{10}$ 之無符號表示為: $(\underbrace{01111101}_{8 \text{ bits}})_2 = (2^7 - 1) - 2 = 127 - 2$

4. 有號數(signed Number)(整數)

(一) Def: 有 sign bit 來表示正、負數。

0: 表正數
1: 表負數

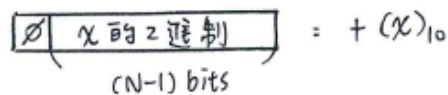
N bits 之格式如下:



需要 1 bit 記正/負

- 正整數

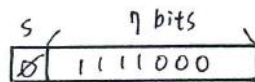
格式一律只有一種:



(三) 正整數表示

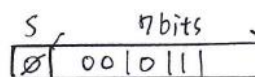
例: 以 8 bits 表示

(1) $+(120)_{10}$



$$120 = (2^7 - 1) - 7 = 127 - 7$$

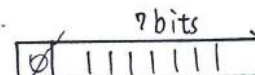
(2) $+(23)_{10}$



$$23 = 16 + 7 = 2^4 + (111)_2$$

(3) $+(255)_{10}$

無法表示: 8 bits 之正整數最大值 $= +(2^8 - 1)_{10} = +255$



\therefore 也叫 overflow 溢位。

(4) N bits 之最大真值?

$$+(2^{N-1} - 1)_{10}$$

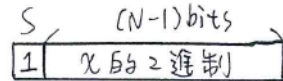
$\Rightarrow N-1$ bits 全放 1.

\rightarrow 因為全 0 為 0 所以扣掉 1

- 負整數(三種)

(1) Sign magnitude

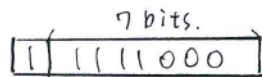
(c) Def: N bits 表示 $-(x)_{10}$ 格式為



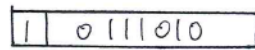
Note: 與正整數格式只差 sign bit 不同。

(e) 例: 8 bits 表示 $-(x)_{10}$, 用 sign magnitude 法

(1) $-(120)_{10}$

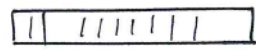


(2) $-(58)_{10}$

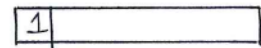


$$58 = 63 - 5 = (2^6 - 1) - 5$$

(3) $-(127)_{10}$



(4) $-(128)_{10} \Rightarrow$ 無法表示 (overflow)



$$128 = 1 \ 0000000$$

8 bits, sign magnitude 法之整數值域: $-127 \sim +127 \Rightarrow$ 共 255 個不同的數

(5) N bits 用 sign magnitude 表示正負整數 x , 則 x 之值域為?

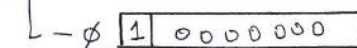
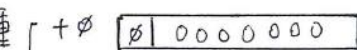
$$\begin{aligned} -(2^{N-1} - 1) &\leq x \leq +(2^{N-1} - 1) \\ &= -2^{N-1} + 1 \\ &= 1 - 2^{N-1} \end{aligned}$$

Note: 最大值與最小值之

[絕對值相同 or
[只差 1 個 sign bit 不同

(6) \emptyset 之表示有幾種方式

Ans: 2 種

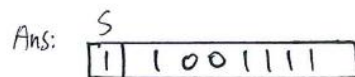


$\therefore 3$ bits $\Rightarrow 8$ 種

8 bits $\Rightarrow 256$ 種, 但表示 255 個數字

$\therefore \emptyset$ 有 2 種表示方式

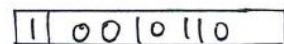
(7) 11001111 之值?



$$-(2^6 + 15)_{10}$$

$$= -(79)_{10}$$

(8) 10010110 之值?



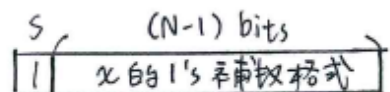
$$-(2^4 + 2^2 + 2)_{10}$$

$$= -(22)_{10}$$

(2) 1's complement (1 補數)

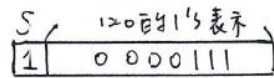
\Rightarrow 2 進位的 1 補數, 0 換成 1, 1 換成 0

(c) Def: N bits 表示 $-x$ 格式:



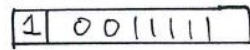
(2) 例: 8 bits 以 1's complement 法表示正負數.

(1) $-(120)_{10}$



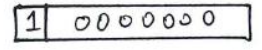
$(120)_{10} \Rightarrow (1111000)_2$
 $\downarrow 1's$
 $(0000111)_2$

(2) $-(96)_{10}$



$(96)_{10} \Rightarrow (1100000)_2$
 $\downarrow 1's$
 $(0011111)_2$

(3) $-(127)_{10}$



$(127)_{10} \Rightarrow (1111111)_2$
 $\downarrow 1's$
 $(0000000)_2$

(4) $-(128)_{10}$ 無法表示: overflow.

$(128)_{10} = (10000000)_2$
 $\downarrow 1's$
 $(01111111)_2$

$\xrightarrow{7 \text{ bits}}$

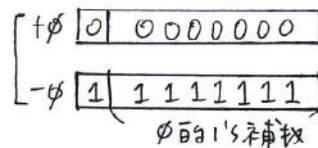
0	1	1	1	1	1	1	1
---	---	---	---	---	---	---	---

 \Rightarrow 表 +127 非 -128

(5) 值域或範圍: $-127 \leq x \leq +127$

(6) N bits 之值域: $-(2^{N-1}-1) \leq x \leq +(2^{N-1}-1)$ 同 sign magnitude 法.

(7) 0 之表示方式 \Rightarrow 2 種



(8) 10111001 之值?

1	0	1	1	1	0	0	1
---	---	---	---	---	---	---	---

 負 $\downarrow 1's$ 还原
 $-(1000110)_2$
 $= -(2^6 + 6) = -70_{10}$

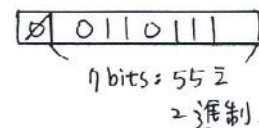
(9) 10011001 之值?

$\downarrow 1's$
 $-(1100110)_2$
 $= -(2^6 + 2^5 + 6) = -(102)_{10}$

(10) 00101111 之值.

正數不用还原
 $+(2^5 + 15)_{10} = +47_{10}$

(11) $+(55)_{10}$

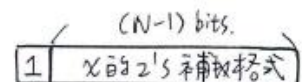


$55 = 63 - 8$

$\begin{array}{r} 11111 \\ - 1000 \\ \hline 11011 \end{array}$

(3) 2's complement (2 補數)

(1) Def: N bits 以 2's 補數法表示 -x 格式:



(=) 例: n 8 bits, 2's complement 表示正負整數

(1) $-(67)_{10}$

$\boxed{1 \ 0 \ 1 \ 1 \ 1 \ 0 \ 1}$

$$67 = 64 + 3 = (1000011)_2$$

$$\overset{1's}{\Rightarrow} (0111100)_2 \xrightarrow{2's} (0111101)_2$$

(2) $-(120)_{10}$

$\boxed{1 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0}$

$$120 = 128 - 8 = (1111000)_2$$

$$\overset{1's}{\Rightarrow} (0000111)_2 \xrightarrow{2's} (0001000)_2$$

(3) $-(127)_{10}$

$\boxed{1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1}$

$$127 = (1111111)_2$$

$$\overset{1's}{\Rightarrow} (0000000)_2 \xrightarrow{2's} (0000001)_2$$

☆(4) $-(128)_{10}$ 可以表示, 未 overflow.

$$128 = (10000000)_2$$

↓ 1's

$$01111111$$

↓ 2's

$$\boxed{10000000} \Rightarrow \text{取 2's 还原} \Rightarrow -128.$$

☆(5) T 值域 (for 8 bits): $-128 \leq x \leq +127$

(6) ϕ 之表示方式只有一種.

$\boxed{0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0}$

☆(7) N bits 之 T 值域: $-2^{N-1} \leq x \leq +(2^{N-1}-1)$

Ex: short int 佔 2 bytes (=16 bits) T 值域?

$$\text{Ans: } -2^{15} \leq x \leq +(2^{15}-1) \Rightarrow -32768 \leq x \leq +32767$$

(8) 10101010 之 T 值

$\boxed{1 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1 \ 0}$

↓ 1's

$$-(1010101)_2$$

↓ 2's

$$-(101010)_2$$

↓

$$-(64+16+6) = -86$$

(9) 11011000

$\boxed{1 \ 1 \ 0 \ 1 \ 1 \ 0 \ 0 \ 0}$

↓ 1's

$$-(0100111)_2$$

↓ 2's

$$-(0101000)_2$$

↓

$$-(2^5+2^3)_2$$

$$= -(40)$$

(10) 01011110

$$+(1011110)_2$$

$$= +(2^6+2^5-1)_{10}$$

$$= +(64+32-2)_{10}$$

$$= +(94)_{10}$$

二進位中表示負數的方法			
(數字)	符號大小法	1's 補數法	2's 補數法
+3	0 1 1	0 1 1	0 1 1
+2	0 1 0	0 1 0	0 1 0
+1	0 0 1	0 0 1	0 0 1
+0	0 0 0	0 0 0	0 0 0
-0	1 0 0	1 1 1	-1 1 1 1
-1	1 0 1	1 1 0	-2 1 1 0
-2	1 1 0	1 0 1	-3 1 0 1
-3	1 1 1	1 0 0	-4 1 0 0

1 表示 + 號
0 表示 - 號

對正數取補數

對正數取補數，再加上 1
(會從 -1 開始計算)

5. Sign magnitude 加減法運算

(-) $A - (-B) \Rightarrow A + B$
 $A - (C + B) \Rightarrow A - B$
 $A + (-B)$
 $A + (+B)$

* ①+② 若 sign bit 相同 \Rightarrow 走加法電路。
 若 A、B 是異號 \Rightarrow 走減法電路。
 \Rightarrow 加、減法各一套電路 cost 高，才有補數系統。
 $\Rightarrow A - B = A + (B \text{ 的補數}) \Rightarrow$ 只須一套加法電路。

6. 1's complement (1 補數)

- 可能有溢位或數值表示錯誤(以 3bits 為例)
 - \Rightarrow 正 + 正 : $011(+3) + 001(+1) = 101$ ，此數為 -2
 - \Rightarrow 負 + 負 : $110(-1) + 100(-3) = 1010$ ，最左的 bit 記不起來被丟掉，變正數

Steps: ① $A-B = A + (B \text{ 的 } 1's)$

② 相加後，若有最左位元進位，則須實施 Carry-In-Addition

③ 正負則以當時之正負整數格式判定之。

例：6 bits 有號數，1's 表示正負整數實施減法。

(1) $(+25) - (+16)$

$+25 = 011001$

$+16 = 010000$

$+16 \text{ 的 } 1's = 101111$

①① 相同

011001

101111

001000

001001

$5 \quad 9$

$\therefore + (9)_{10}$

(2) $(+8) - (+22)$

$+8 = 001000$

$+22 = 010110$

$+22 \text{ 的 } 1's = 101001$

①①

001000

101001

110001

$5 \quad 1's \text{ 还原}$

$-(01110)_2$

$= -(14)_2$

→ 未还原之值为 1 補數之結果

→ 再取補數是因為將 1 補數还原為整數形式

7. 2's complement (2 補數) 加減法

Steps: ① $A-B = A + (B \text{ 的 } 2's)$

② 相加後，若有最左位元進位，則捨去

③ 結果依 2's 正負整數格式判定其值

例：6 bits 2's 補數 有號數 實施減法。

(1) $(+20) - (+13)$

$+20 = 010100$

$+13 = 001101$

$+13 \text{ 的 } 2's = 110011$

①①

010100

110011

000111

$5 \quad \therefore + (7)_{10}$

(2) $(-23) - (-5)$

$-23 = 101001$

$-5 = 111011$

$-5 \text{ 的 } 2's = 000101$

00

101001

000101

101110

$5 \quad 2's \text{ 还原}$

$-(10010)_2$

$= -(18)_2$

(3) $(-6) - (+15)$

$-6 = 11010$

$+15 = 001111$

$+15 \text{ 的 } 2's = 110001$

①①

11010

110001

101011

$5 \quad 2's \text{ 还原}$

$-(10101)_2$

$= -(21)_{10}$

(4) $(-25) - (+8)$

$\because 6 \text{ bits } 2's \text{ 值域 } -2^5 \leq x \leq +2^5 - 1$

$= -32 \leq x \leq +31$

$\therefore -25 - 8 = -33 \Rightarrow \text{overflow}$

→ 未還原之值為 2 補數之結果

→ 再取補數是因為將 2 補數還原為整數形式

8. Excess-code

(一) 用在浮點數之指數值表示方法之一

(二) 命名: 給予 N bits. Excess-code 也叫 Excess- 2^{N-1} Code

例: (1) 8 bits \Rightarrow Excess-128 Code

(2) 10 bits \Rightarrow Excess-512 Code

(3) Excess-32 Code \Rightarrow 6 bits.

(4) Excess-1024 Code \Rightarrow 11 bits

(三) 以 Excess- 2^{N-1} Code 表示正負整數 x

(1) 存入電腦內部 $\Rightarrow x + 2^{N-1}$ 以 2 進制存入

(2) 反求其值時 \Rightarrow 存入內容 -2^{N-1} 即得 x

(四) 例: 8 bits Excess-128 Code 表示正負整數 x

(1) +5

$$+5 + 128$$

$$\Rightarrow (0000101)_2$$

(2) -40

$$-40 + 128$$

$$= -40 + 127 + 1 = 127 - 39$$

$$\Rightarrow (0101000)_2$$

(5) 00010110 之值

$$(00010110)_2 - 128$$

$$= 22 - 128$$

$$= -106$$

(6) 10001110 之值

$$(10001110)_2 - 128$$

$$= 128 + 14 - 128 = 14$$

(7) 00000000 之值

$$0 - 128 = -128$$

(8) 11111111 之值

$$(11111111)_2 - 128$$

$$= (01111111)_2 = +127$$

- 正 + 正 或 負 - 負 或 正 - 負

⇒ 可能溢位

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

Ex: 6 bits 2's 表正負整數, 6 bits 2's 值域 $-32 \leq x \leq 31$

算 $100110 - 010111$ 是否 overflow?

負 正

[法一] 100110 010111

↓ 2's ↓
 $-(11010)_2$ $+23$

$= -26$ $\therefore -26 - 23 = -49$ overflow

40

[法二] $100110 - 010111$

$= 100110 + (010111 \text{ 的 } 2's)$

100110 101001

101001
~~000111~~ \therefore overflow.