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 → (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： $-(2^n-1) \sim 2^n-1$→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)

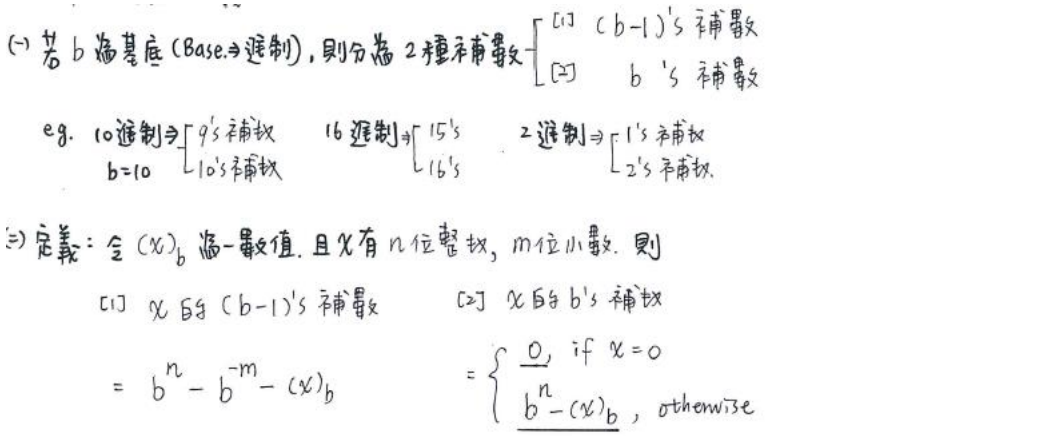
G. **2's Complement** – 值域 ：$-(2^n) \sim 2^n-1$；進位 ：左邊捨去；沒進位 ：加負號，取 2 補數(將補數轉換為整數)

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 ⇒ A + (B的補數)
- 優點: ① 不須要提供減法電路. 只留一套加法電路. ∴ 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

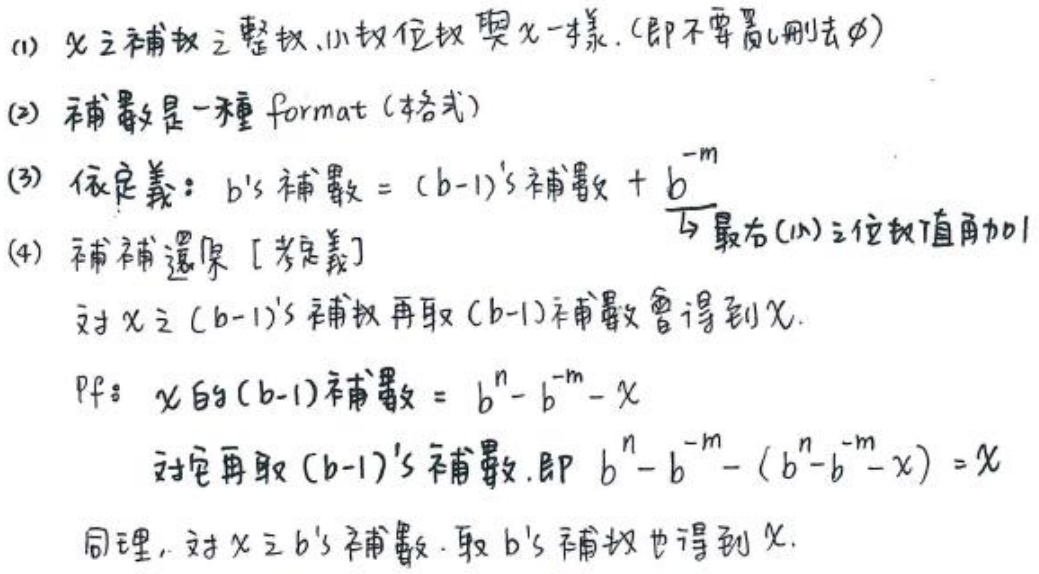


⇨ B – 1 補數有正負 0 所以要再扣最小位數次方之值，B – 1 補數與其原來的數將加不會超過該位數之最大值，看下列例子

Ex. +3(011) + (-3)(100) = 0(111) -> 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



- 例子

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

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

(1) 9's 補數

$= b^n - b^{-m} - (x)_b$

$= 10^4 - 10^{-2} - (2019.72)_{10}$

$= 10000 - 0.01 - 2019.72$

$= 7980.27$

(2) 10's 補數.

$= b^n - (x)_b$

$= 10^4 - (2019.72)_{10}$

$= (7980.28)_{10}$

= 即 9's 補最右位數加 1.

- 快速做法

⇨ 最大值 – 欲轉換之數 = B – 1 補數

⇨ B 補數 = B – 1 補數最右位數 + 1

(1) $(2019.72)_{10}$ 之 9's 補數

⇒ 9999.99 − 2019.72 = $(7980.27)_{10}$　而 10's 補 = $(7980.28)_{10}$

1(2) $(7420.361)_8$

的 $\begin{cases} 7\text{'s 補數} = (0357.416)_8 \\ 8\text{'s 補數} = (0357.417)_8 \end{cases}$

7777.777 − 7420.361 = 0357.416 補1.

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

的 $\begin{cases} 15\text{'s 補數} = (5D39.EC)_{16} \\ 16\text{'s 補數} = (5D39.ED)_{16} \end{cases}$

15 15 15 15 . 15 15 − A 2 C 6 . 1 3 = 5 D 3 9 . E C

(4) $(101101.011)_2$ ↓ 1's 補數 0, 0 變 1

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

3. 無號數(Unsigned Number)

(一) Def: 無正負符號. 值>0 開始. n位整數而言. 若給予N bits. 值域 $0 \sim 2^N-1$

(二) 例: 8 bits之無号数. 值域: $0 \sim 2^8-1 \Rightarrow 0 \sim 255$

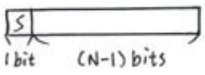
例: 8 bits, $(125)_{10}$ 之無号数表示為: $(\underbrace{01111101}_{8\text{bits}})_2 = (2^7-1)-2 = 127-2$

4. 有號數(signed Number)(整數)

(一) Def: 有 sign bit 來表示正.負數.

[0: 表正数
[1: 表負数

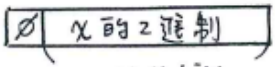
N bits 之格式如下:



要先1 bit 記正/負

- 正整數

格式一律只有一種:

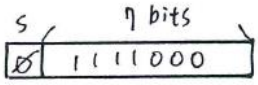


$= +(X)_{10}$

(三) 正整数表示

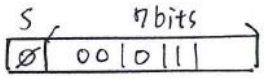
例: 以8 bits 表示

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



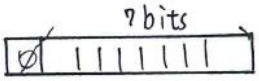
$120 = 127-7 = (2^7-1)-7 =$ 7個1扣0 111

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



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

(3) $+(255)_{10}$ 無法表示 ∵ 8 bits 之正整数最大值 $= +(2^7-1)_{10} = +127$



∴ 也叫 overflow 溢位.

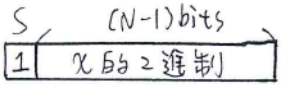
(4) N bits 之最大值? $+(2^{N-1}-1)_{10}$ ⇒ N-1 個bits 全放 1.

→因為全 0 為 0 所以扣掉 1

- 負整數(三種)

(1) Sign magnitude

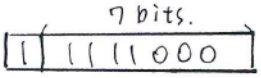
(一) Def: N bits 表示 $-(X)_{10}$ 格式為



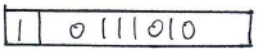
Note: 與正整數格式只差 sign bit 不同.

(二) 例: 8 bits 表示 $-(X)_{10}$, 以 sign magnitude 法

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

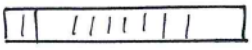


(2) $-(58)_{10}$

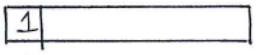


$58 = 63 - 5 = (2^6 - 1) - 5$

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



(4) $-(128)_{10}$ ⇒ 無法表示(overflow)



$128 = 1\ 0000000$

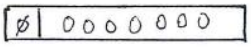
8 bits, sign magnitude 法之整數值域: $-127 \sim +127$ ⇒ 共 255 個不同的數字

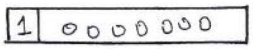
☆(5) N bits 以 sign magnitude 表示. 正負整數 X, 則 X 之值域為?

$$-(2^{N-1}-1) \le X \le +(2^{N-1}-1)$$
$$= -2^{N-1}+1$$
$$= 1-2^{N-1}$$

Note: 最大值與最小值之
- 絕對值相同 or
- 只差 1 個 sign bit 不同

☆(6) ∅ 之表示有 ? 種方式

Ans: 2 種
- $+\emptyset$ 

- $-\emptyset$ 

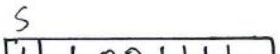
∵ 3 bits ⇒ 8 種

8 bits ⇒ 256 種. 但表示 255 個數字

∵ ∅ 有 2 種表示方式

(7) 11001111 之值?

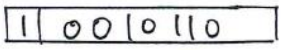
Ans:



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

$= -(79)_{10}$ ☆

(8) 10010110 之值?



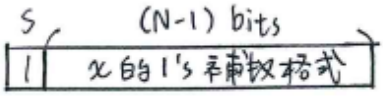
$-(2^4+2^2+2)_{10}$

$= -(22)_{10}$ ☆

(2) 1's complement (1 補數)

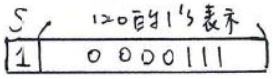
⇨ 2 進位的 1 補數，0 換成 1，1 換成 0

(一) Def: N bits 表示 $-X$ 格式:



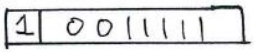
(二)例: 8 bits 以 1's complement 法表示正負數.

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



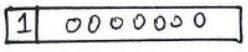
$(120)_{10} \Rightarrow (1111000)_2$
$\Downarrow$ 1's
$(0000111)_2$

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



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

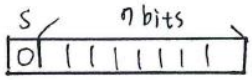
(3) $-(127)_{10}$



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

(4) $-(128)_{10}$ 無法表示: overflow.

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

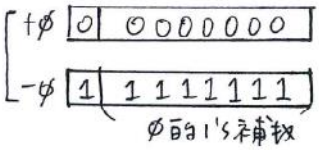


$\Rightarrow$ 表 +127 非 -128

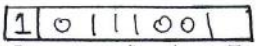
☆(5) 值域或為: $-127 \le x \le +127$

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

(7) ∅之表示方式 $\Rightarrow$ 2種



(8) 10111001_2 之值?



$-(1000110)_2$
$= -(2^6+6) = -70$ ※

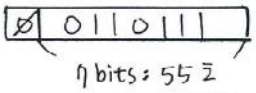
(9) $\boxed{1}0011001_2$ 之值?

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

(10) $\boxed{0}0101111_2$ 之值.

☆ 正數不用還原
$+(2^5+15)_{10} = +(47)_{10}$

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

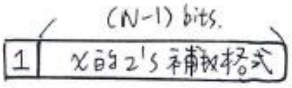


$55 = 63 - 8$

$$\begin{array}{r} 111111 \\ -\ \ 1000 \\ \hline 110111 \end{array}$$

(3) 2's complement (2 補數)

(一) Def: N bits 以 2's 補數法表示 -x 格式:



(二) 例：以 8 bits，2's complement 表示正負整數 X

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

1	0111101

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

1's ⇒ $(0111100)_2$ 2's ⇒ $(0111101)_2$

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

1	0001000

$120 = 127-7 = (1111000)_2$

1's ⇒ $(0000111)_2$ 2's ⇒ $(0001000)_2$

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

1	0000001

$127 = (1111111)_2$

1's ⇒ $(0000000)_2$ 2's ⇒ $(0000001)_2$

☆(4) $-(128)_{10}$ 可以表示，未 over flow.

$128 = (10000000$

⇓ 1's

01111111

⇓ 2's

1	0000000

⇒ 取 2's 還原 ⇒ -128.

☆(5) 下限 (for 8 bits)：$-128 \le X \le +127$

(6) 0 之表示方式只有一種.

0	0000000

☆ (7) N bits 之下限：$-2^{N-1} \le X \le +(2^{N-1}-1)$

Ex: short int 占 2 bytes (=16 bits) 下限？

Ans: $-2^{15} \le X \le +(2^{15}-1) \Rightarrow -32768 \le X \le +32767$

(8) 10101010 之值

1	0101010

⇓ 1's

$-(1010101)_2$

⇓ 2's

$-(1010110)_2$

⇓

$-(64+16+6) = -86$ ☆

(9) 11011000

1	1011000

⇓ 1's

$-(0100111)_2$

⇓ 2's

$-(0101000)_2$

⇓

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

$= -(40)_{10}$ ☆

(10) 01011110

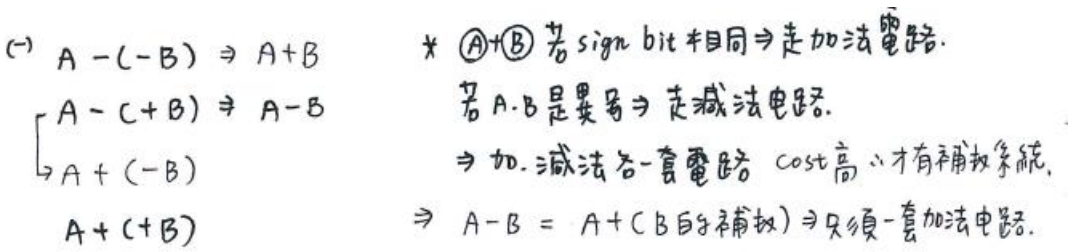
$+(1011110)_2$

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

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

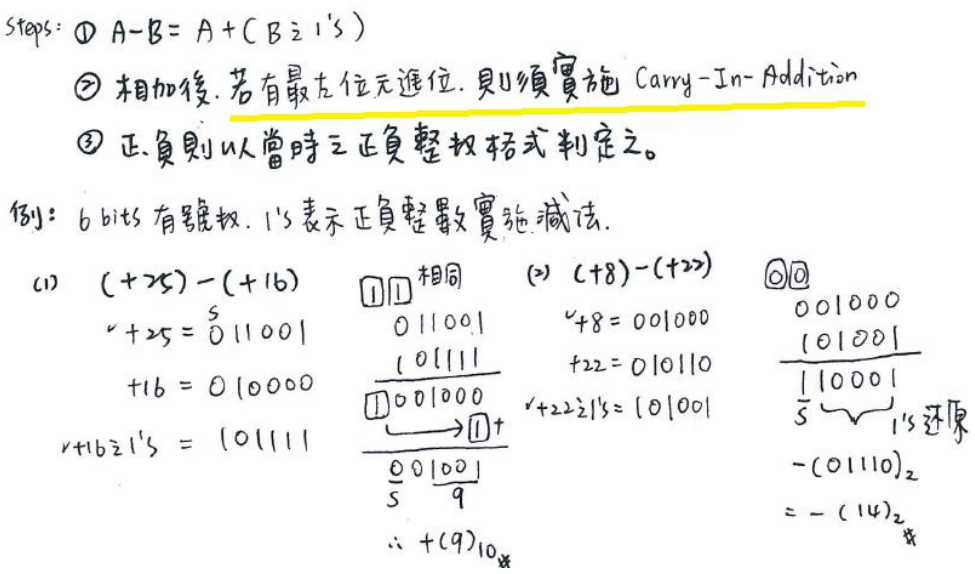
$= +(94)_{10}$ ☆

5. Sign magnitude 加減法運算



6. 1' s complement (1 補數)

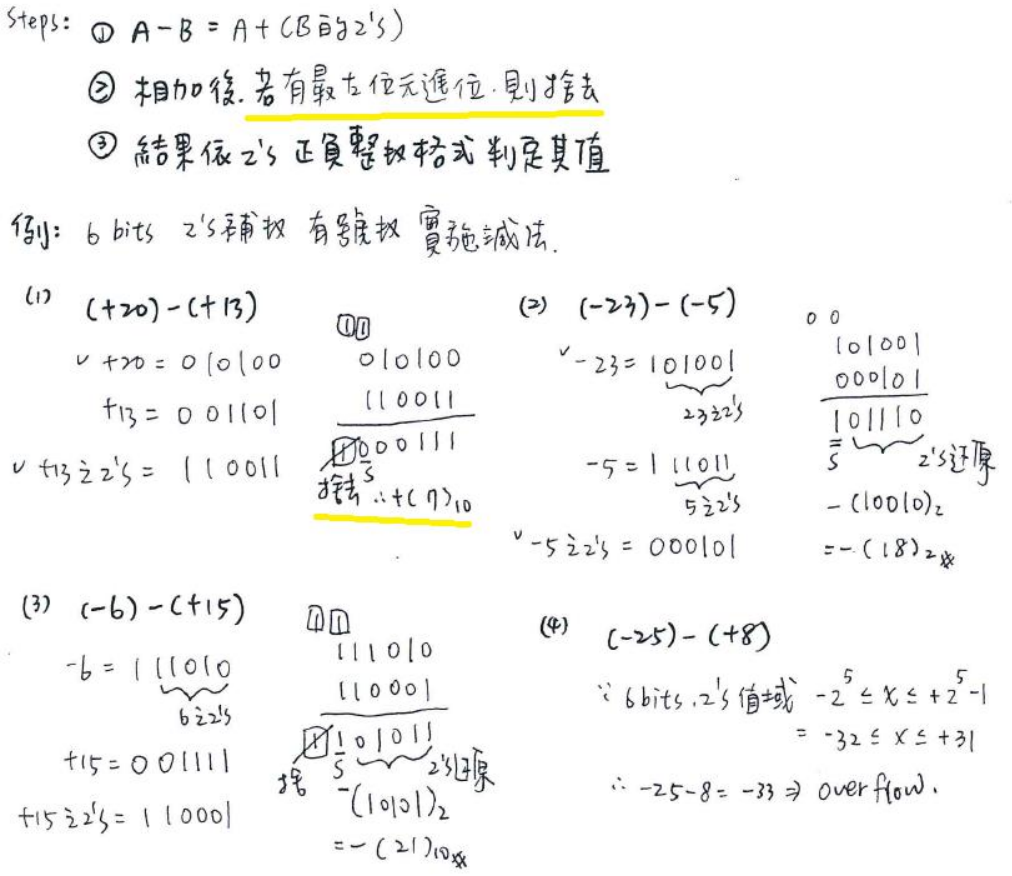
- 可能有溢位或數值表示錯誤(以 3bits 為例)
  - ⇨ 正 + 正 : 011(+3) + 001(+1) = 101， 此數為-2
  - ⇨ 負 + 負 : 110(-1) + 100(-3) = 1010，最左的 bit 記不起來被丟掉，變正數



→ 未還原之值為 1 補數之結果

→ 再取補數是因為將 1 補數還原為整數形式

7. 2's complement (2 補數)加減法



→ 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 ⇒ Excess-128 code

(2) 10 bits ⇒ Excess-512 code

(3) Excess-32 Code ⇒ 6 bits.

(4) Excess-1024 Code ⇒ 11 bits

(三) 以 Excess-2^{N-1} Code 表示正負整數 X

(1) 存入電腦內部 ⇒ X+2^{N-1} 之 2 進制存入

(2) 反求其值時 ⇒ 存入內容 -2^{N-1} 即得 X

(四) 例：8 bits Excess-128 code 表正負整數 X

(1) +5

+5+128

⇒ 10000101※

(2) -40

-40+128

= -40+127+1 = 127-39

⇒ 01011000※

(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※

Bit pattern	Excess-8		2's complement
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

- 正 + 正 或 負 − 負 或 正 − 負
  ⇨ 可能溢位

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

則 $100110 - 010111$ 是否 overflow?

負　　正

[法一] 100110 → 2's → $-(11010)_2 = -26$

010111 → $+23$

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

[法二] $100110 - 010111$

$= 100110 + (010111 \text{ 之 } 2\text{'s})$

010111 之 2's → 101001

100110
101001
(1)001111

$\therefore$ overflow.