02 CHAPTER 數位資料表示法

0與1的組合

1位元	2位元	3位元	4位元	5位元
0	00	000	0000	00000
1	01	001	0001	00001
	10	010	0010	00010
	11	011	0011	00011
		100	0100	00100
		101	0101	00101
		110	0110	00110
		111	0111	00111
			1000	01000
			1001	01001
			1010	01010
			1011	01011
			1100	01100
			1101	01101
			1110	01110
			1111	01111
				10000
				10001
				10010
				10011
				10100
				10101
				10110
				10111
				11000
				11001
				11010
				11011
				11100
				11101
				11110
				11111

數位資訊的單位

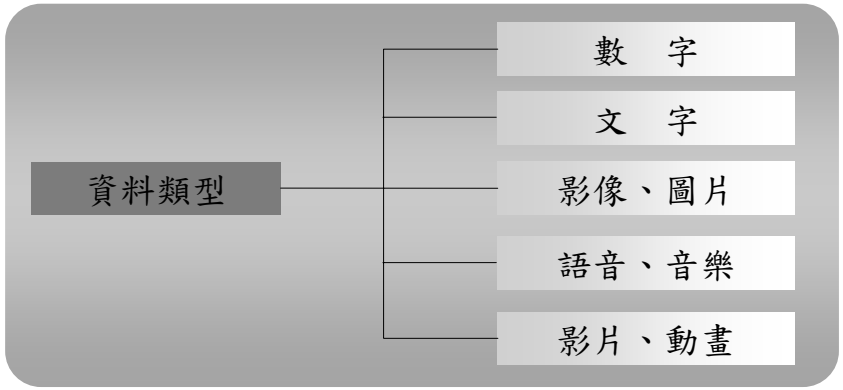
- 位元(***bi***nary digi***t***，簡稱bit)是數位資訊的基本粒子，也是電腦儲存或傳遞資料的最小單位，常用0或1來表示
  - 當初電腦會採用位元表示資料，主要是因爲電子元件的穩定狀態有兩種：
    - 一種是“開”(通常用來表示“1”)
    - 一種是“關”(通常用來表示“0”)
- 電腦常以8個位元爲存取單位，因此8個位元稱爲

位元組(byte)

數位資訊的單位(續)

縮寫單位	全名	精確位元組個數	大約位元組數	範例
KB	KB Kilo Byte	2^{10} = 1,024	一千(10^3)	這個檔案的大小約238 KB。
MB	MB Mega Byte	2^{20} = 1,048,576	一百萬(10^6)	此大姆碟的容量爲512 MB。
GB	GB Giga Byte	2^{30} = 1,073,741,824	十億(10^9)	本片DVD的容量爲4.7 GB。
TB	TB Tera Byte	2^{40} = 1,099,511,627,776	一兆(10^{12})	這部高容量磁碟可儲存20 TB的資料。

2-1 資料型態



2-2 二進位表示法

- 一個數字在不同的位置上所表示的數值也就不同，如三位數“523”，右邊的“3”在個位上表示3個一，中間的“2”在十位上就表示2個十，左邊的“5”在百位上則表示5個百，換句話說，$523 = 5 \times 10^2 + 2 \times 10^1 + 3$
- 以B爲基數，則$d_n d_{n-1} \ldots d_2 d_1 . r_1 r_2 \ldots r_{m-1} r_m$所表示的數爲$d_n \times B^{n-1} + d_{n-1} \times B^{n-2} + \ldots + d_2 \times B^1 + d_1 \times B^0 + r_1 \times B^{-1} + r_2 \times B^{-2} + \ldots + r_{m-1} \times B^{-(m-1)} + r_m \times B^{-m}$
- 二進位表示法：B=2

註：若數值表示成$d_n d_{n-1} \ldots d_1 d_0 . r_{-1} r_{-2} \ldots$，則次方更一致。

二進位、十進位、十六進位

十進位	二進位	十六進位	十進位	二進位	十六進位
0	0	0	8	1000	8
1	1	1	9	1001	9
2	10	2	10	1010	A
3	11	3	11	1011	B
4	100	4	12	1100	C
5	101	5	13	1101	D
6	110	6	14	1110	E
7	111	7	15	1111	F

一個字根問題

- Octal – 八進位；Decimal – 十進位
- Oct-這個字根代表8；Dec-這個字根代表10
  - 爲什麼October 不是八月而是十月？
  - 爲什麼 December不是十月而是十二月？

因爲插入了七月和八月

- July源於凱撒(Julius Caesar) 之名，在凱薩之前就有曆法，那時是以March 爲一年的開端，而July是第十五個月；凱撒修改曆法後，將一年的開始訂爲January，而將July 提升到第七位，這個改變一直沿用至今。
- 凱撒的繼承人奧古斯都(Augustus)去世後，羅馬元老院決定將他列入「神」的行列，並且將8月稱爲「奧古斯都」月，這也是歐洲語文中8月的來源。
- 那二月爲什麼只有二十八天呢？

二月被砍過兩天

- 二月爲什麼通常只有二十八天？
  - 凱撒(Julius Caesar) 修改曆法時，本來規定每年十二個月裡，逢單是大月三十一日，逢雙是小月三十日，但是這樣算下來，一年就變成三百六十六日，所以必須設法在一年中扣去一天。那時候判處死刑的人犯均在二月分執行，因此人們認爲二月是不吉利的月分，既然要扣除一天，那麼就由二月分來扣掉，讓不吉利的日子減少一天，因此二月分就成了二十九日。
  - 七月是逢單爲大月三十一日，爲了讓八月也偉大，就改爲大月三十一日。糟了！又多出一天怎麼辦？那還是由二月分來扣除，因此結果二月分就變成二十八日。

2-3 各種進位表示法的轉換

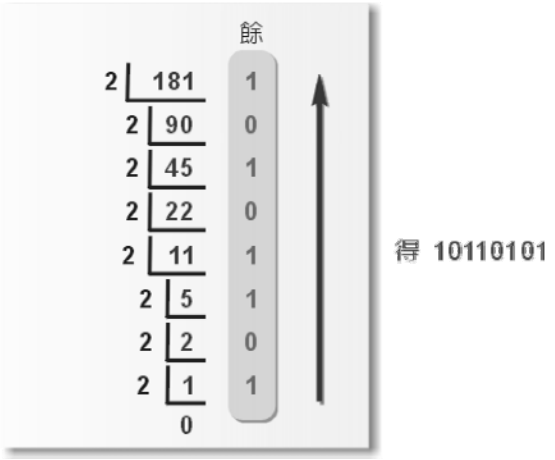
二進位轉十進位

1	0	1	1	0	1	0	1	.	1	1	0	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0		2^{-1}	2^{-2}	2^{-3}	2^{-4}

$$2^7 + 2^5 + 2^4 + 2^2 + 2^0 + 2^{-1} + 2^{-2} + 2^{-4}$$
$$= 128 + 32 + 16 + 4 + 1 + 0.5 + 0.25 + 0.0625$$
$$= 181.8125$$

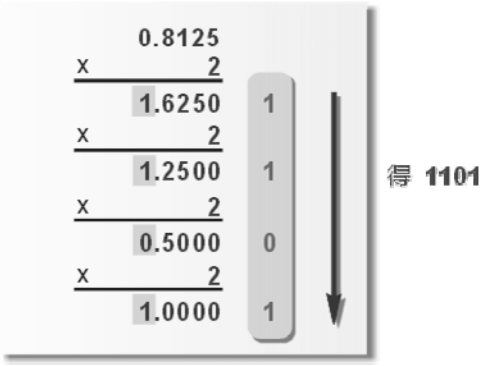
10110101.1101_2所對應的十進位數爲181.8125

十進位整數轉二進位



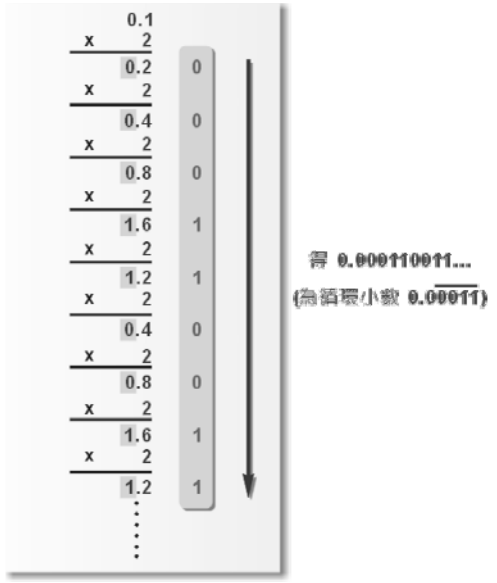
十進位181所對應的二進位數爲10110101_2

十進位小數轉二進位



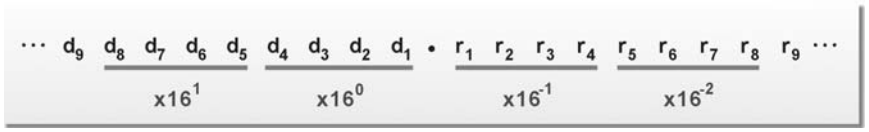
十進位0.8125所對應的二進位數爲0.1101_2

十進位0.1的二進位表示法爲何？



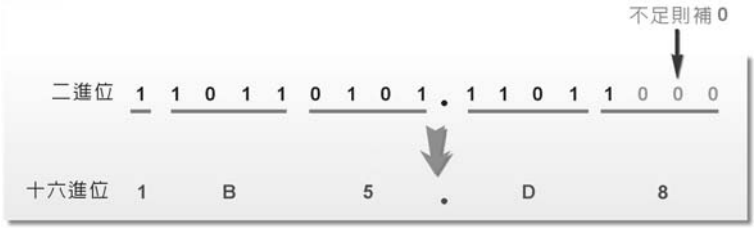
十進位0.1所對應的二進位數爲無窮位數的$0.000110011\ldots_2$

二進位轉十六進位



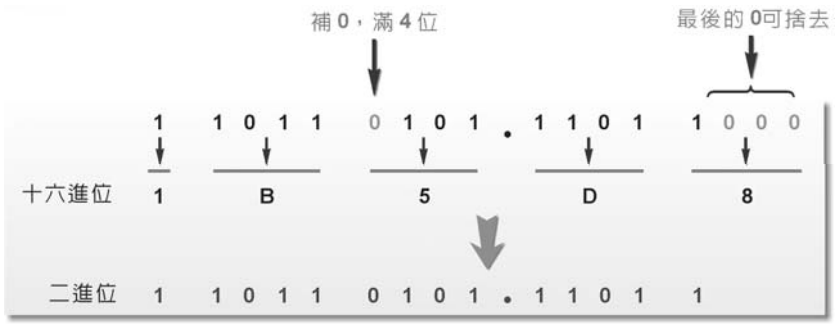
二進位數換成十六進位數時，每四個位數合成一項

二進位轉十六進位



110110101.11011_2的十六進位表示法爲$1B5.D8_{16}$

十六進位轉二進位



1B5.D8$_{16}$的二進位表示法爲110110101.11011$_2$

2-4 整數表示法

表2-2：以8位元所表示的「無正負符號的整數」

位元字串	十進位數
00000000	0
00000001	1
00000010	2
...	...
11111110	254
11111111	255

帶正負符號大小表示法

表2-3：以8位元所表示的「帶正負符號大小表示法」

位元字串	十進位數
00000000	0
00000001	1
...	...
01111111	127
10000000	-0
10000001	-1
...	...
11111111	-127

一補數表示法

- 給定一個十進位數值，轉換成它的一補數表示法步驟如下：

步驟1 先忽略其符號，將數字的部分轉成二進位數值。

步驟2 若該二進位數值超過n - 1 個位元， 則為「溢位」(overflow)，無法進行轉換；否則在它的左邊補0，直到共有n個位元為止。

步驟3 若所要轉換的數為正數或零，則上步所得數值即為所求；若為負數，則將每個位元做補數轉換，原為0的轉成1；原為1的轉成0。(好像打拱豬時的豬羊變色)

-41的一補數表示法爲11010110

一補數轉十進位

- 如果最左邊的位元是0，則表示該數是正數，只要將後面的位元以前面介紹的二進位轉十進位方式求出其數值即可。
- 如果最左邊的位元是1，則表示該數是負數，先將每個位元做補數轉換，原爲0的轉成1；原爲1的轉成0。然後將該二進位轉成十進位，再加上一個負號即可。

範例9

給定一補數11010110，因為最左邊的位元是1，所以我們先將這補數原為0的轉成1；原為1的轉成0，得00101001。再將二進位的00101001轉成十進位的41，然後加上一個負號得-41。

二補數表示法

- 給定一個十進位數值，轉換成它的二補數表示法步驟如下：

步驟1 先忽略其符號，將數字的部分轉成二進位數值。

步驟2 若該二進位數值超過n - 1個位元， 則為「溢位」(overflow)，無法進行轉換；否則在它的左邊補0，直到共有n個位元為止。

步驟3 若所要轉換的數為正數或零，則上步所得數值即為所求；若為負數，則最右邊的那些0及最右邊的第一個1保持不變，將其餘的每個位元做補數轉換，原為0的轉成1；原為1的轉成0。

40和-40的二補數表示法爲何？

範例10

40的二補數表示法為何？ 第一步先將40轉成二進位數值101000；第二步在二進位數值左邊補上0，使得00101000共有8個位元，因為要表示的數為正數，所以00101000即為所求。

範例11

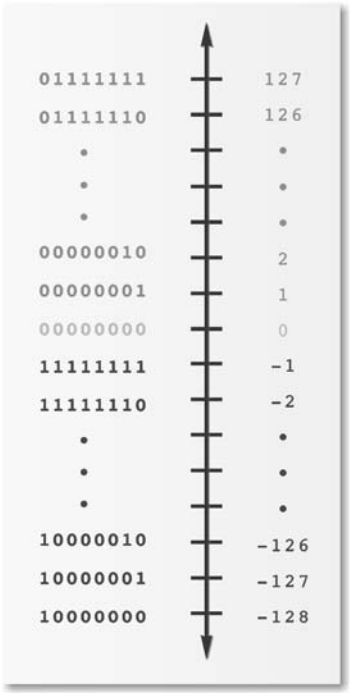
-40的二補數表示法為何？ 第一步先將40轉成二進位數值101000；第二步在二進位數值左邊補上0，使得00101000共有8個位元；這時所要表示的數為負數，所以最右邊的三個0及第一個1維持不變，其餘的將原為0的轉成1；原為1的轉成0，得11011000。

二補數轉十進位

- 如果最左邊的位元是0，則表示該數是正數，只要將後面的位元以前面介紹的二進位轉十進位方式求出其數值即可。
- 如果最左邊的位元是1，則表示該數是負數，保留最右邊的那些0及最右邊的第一個1，將其餘的每個位元做補數轉換，原爲0的轉成1；原爲1的轉成0。然後將該二進位轉成十進位，再加上一個負號即可。

範例12

給定二補數11011000，因為最左邊的位元是1，所以我們先保留最右邊的那三個0及最右邊的第一個1，再將其他的位元原為0的轉成1；原為1的轉成0，得00101000。再將二進位的00101000轉成十進位的40，然後加上一個負號得-40。



二補數表示法位元字串與數值的對應關係

二補數表示法的兩正數相加

			1						← 進位
	16	0	0	0	1	0	0	0	0
+)	24	0	0	0	1	1	0	0	0
	40	0	0	1	0	1	0	0	0

		忽略								
		(1)	1	1	1					← 進位
	-16		1	1	1	1	0	0	0	0
+)	24		0	0	0	1	1	0	0	0
	8		0	0	0	0	1	0	0	0

二補數表示法的一正一負相加，且結果爲正

	-24	1	1	1	0	1	0	0	0
+)	16	0	0	0	1	0	0	0	0
	-8	1	1	1	1	1	0	0	0

二補數表示法的一正一負相加，且結果爲負

二補數表示法的兩負數相加

二補數表示法的兩正數相加結果超過正數儲存範圍

二補數表示法的兩負數相加結果小於負數儲存範圍

-40的二補數表示法正好是2^8-40

爲何二補數可以這樣做運算

- 假設是*n* bits
- 正數 + 正數 (和一般情況一樣)
- 負數(-x) + 負數(-y)
  -x在二補數表示值爲 2^n-x
  -y在二補數表示值爲 2^n-y

$$2^n - x + 2^n - y = \boxed{2^n} + \boxed{(2^n - (x+y))}$$

進位　-(x+y)的二補數表示法

爲何二補數可以這樣做運算(續前頁)

- 正數 (x) + 負數 (-y)
  -y在二補數表示值爲 2^n-y
  得 2^n+x-y
  (1) x >= y
  x-y爲正值或0；2^n爲進位
  (2) x <y
  $$2n+x-y = \boxed{2n-(y-x)}$$

-(y-x)的二補數表示法

2-5 浮點數表示法

- IEEE 754標準

單倍精準數

符號位元	1個位元，以0表示正數；以1表示負數。
指數部分	8個位元，以過剩127(Excess 127)方式表示。
尾數部分	23個位元，從標準化的小數點後開始存起，不夠的位元部份補0。

浮點數表示法

範例19 實數10110.100011 的浮點數表示法

給定一實數10110.100011，先轉換成1.0110100011×2^4，因為是正數，所以符號位元為0，其尾數部分為0110100011，指數部分為4，以過剩127方式儲存，必須先加上127，得131，再將131轉換成二進位，得10000011。因此10110.100011若按IEEE 754標準儲存，為 01000001101101000110000000000000。

範例20 實數-0.0010011的浮點數表示法

-0.0010011又是如何儲存呢？先轉換成-1.0011×2^{-3}，因為是負數，所以符號位元為1，其尾數部分為0011，指數部分為-3，以過剩127方式儲存，必須先加上127，得124，再將124轉換成二進位，得01111100。因此-0.0010011若按IEEE 754標準儲存，為10111110000110000000000000000000。

浮點數表示法的數值

範例21

在IEEE 754標準下，01000001010010100011000000000000所儲存的數值為多少呢？首先，位元符號為0，所以是正數，指數部分是10000101，換成十進位為133，再減去127，得6，因此，01000001010010100011000000000000所儲存的數值為1.0010100011×2^6，也就是1001010.0011。

範例22

在IEEE 754標準下，10000010100101000110000000000000所儲存的數值為多少呢？首先，位元符號為1，所以是負數，指數部分是00000101，換成十進位為5，再減去127，得-122，因此，10000010100101000110000000000000所儲存的數值為$-1.0010100011\times2^{-122}$ 。

請試試下面的例子 (IEEE 754單倍精準數表示法)

- 1.5

  0 01111111 10000000000000000000000

- 125.625

  0 10000101 11110110100000000000000

- http://babbage.cs.qc.edu/IEEE-754/Decimal.html
  (芃安助教提供的驗算網址)

IEEE 754單倍精準數

- 0的公訂表示法爲
  00000000000000000000000000000000
- 10000000000000000000000000000000也是0(代表-0)
- 指數部分的-127(00000000)和+128(11111111)做爲特殊用途
- 最小的正數爲
  00000000100000000000000000000000其數值爲$+2^{-126}$
- 最大的正數爲
  01111111011111111111111111111111其數值爲$(2-2^{-23})x2^{127}$

2-6 ASCII及Unicode

- 在電腦裡，所有的文字也存成位元字串，因此我們必須有公訂的對照表，以便我們能在儲存時將文字轉成位元字串，而在解讀時能將位元字串轉回文字

ASCII　Big5

Unicode　GB

EBCDIC

ASCII (7位元)

ASCII碼	鍵盤	ASCII 碼	鍵盤	ASCII 碼	鍵盤	ASCII 碼	鍵盤
0	NUL	7	BEL	10	LF	13	CR
27	ESC	32	SPACE	33	!	34	"
35	#	36	$	37	%	38	&
39	'	40	(	41	)	42	*
43	+	44	,	45	-	46	.
47	/	48	0	49	1	50	2
51	3	52	4	53	5	54	6
55	7	56	8	57	9	58	:
59	;	60	<	61	=	62	>
63	?	64	@	65	A	66	B
67	C	68	D	69	E	70	F
71	G	72	H	73	I	74	J
75	K	76	L	77	M	78	N
79	O	80	P	81	Q	82	R
83	S	84	T	85	U	86	V
87	W	88	X	89	Y	90	Z
91	[	92	\	93	]	94	^
95	_	96	`	97	a	98	b
99	c	100	d	101	e	102	f
103	g	104	h	105	i	106	j
107	k	108	l	109	m	110	n
111	o	112	p	113	q	114	r
115	s	116	t	117	u	118	v
119	w	120	x	121	y	122	z
123	{	124	|	125	}	126	~
127	DEL						

Unicode

What is Unicode?
Unicode provides a unique number for every character,
no matter what the platform,
no matter what the program,
no matter what the language.

萬國碼係啥米碗糕啊？
每個字符在萬國碼中有個唯一的代號
無論在哪種機器上
無論在哪個程式裡
無論用哪種語言

Unicode

- 已發展出多種編碼方式：

  UTF-8　UTF-16　UTF-32

- 分別以：

  8位元　16位元　32位元

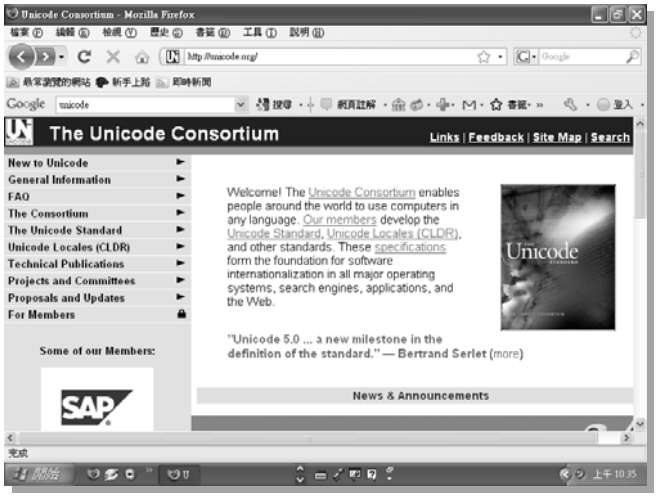
  爲基本單元的編碼方式。
- UTF-8在全球資訊網最通行，UTF-16爲Java及Windows所採用，而UTF-32則爲一些Unix系統使用
- e.g. 在UTF-16編碼方式中：

  趙(8D99)　坤(5764)　茂(8302)

Unicode符號對照表

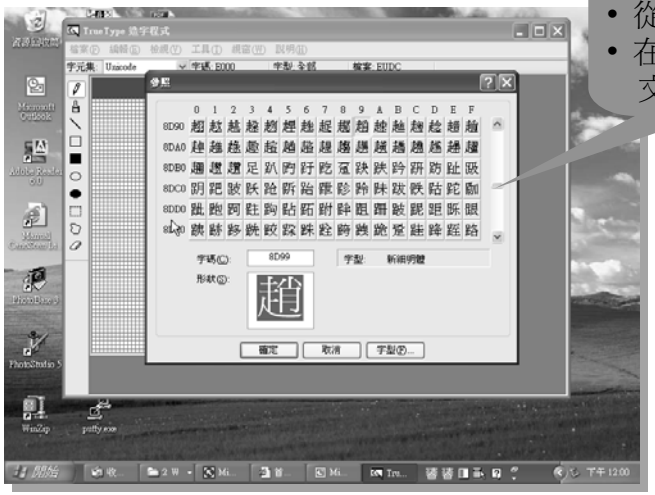
範 圍	代表的字符群
0000-007F	基本拉丁字符(與ASCII相同)
0080-024F	擴充的拉丁字符
0370-03FF	希臘字符
0E00-0E7F	泰文
0E80-0EFF	寮文
2200-22FF	數學符號
2500-25FF	方塊圖形及幾何圖形
3040-30FF	平假名及片假名
4000-9FFF	CJK；中文、日文及韓文之漢字

Unicode官方網頁www.unicode.org



從造字程式找

從造字程式找(續)



Unicode Translation Format

- 在實際應用上，Unicode 並非皆以16位元儲存字元，讀者可參照Wikipedia上的相關條目。
- UTF-8(以8位元爲基本編碼單元的Unicode Translation Format)vs. UTF-16(以16位元爲基本編碼單元的Unicode Translation Format)
  - 在UTF-8的編碼方式中，傳統的ASCII字符仍以一個位元組儲存(位元組首位爲0，後面的7位元爲原ASCII的編碼)。例如：

「A」的UTF-8為「41」　UTF-16則為「0041」

  - 在UTF-8的編碼方式中，其餘非ASCII字符，再依類別而有不同長度的編碼方式。例如：

「趙」的UTF-8為「E8B699」　UTF-16則為「8D99」

  - 「MadEdit」是一個免費的跨平台編輯軟體，它可檢視各個字符在不同編碼法下的十六進位數值。(感謝大葉大學李立民教授來函介紹)