第 4 章

數字系統

本章提要

- 4-1 數字系統
- 4-2 數字系統間的轉換
- 4-3 資料表示法

4-1 數字系統

- 4-1-1 關於數字系統
- 4-1-2 常用的數字系統

4-1-1 關於數字系統

- 自古以來, 人類習慣以十進位來計算事物, 同時也會使用到其他的數字系統, 最典型的例子如時間的計算, 時與分採六十進位；而日與時的換算則為二十四進位；月與年則使用十二進位來計算；另外我們也會用一打 (12個) 這樣的單位來計算東西的數量, 而傳統的重量單位－台斤與兩的計算, 則是十六進位, 即 1 台斤等於 16 兩。

關於數字系統

- 電腦和人一樣, 亦有屬於自己的數字系統, 但由於它只能處理 0 與 1 的資料, 所以在電腦的世界中只有二進位系統。
- 不過, 如果要人類來閱讀由 0 與 1 組成的一長串資料, 實在是相當困難；因此對於電腦內部所存的資料, 我們一般會以八進位、十六進位來表示, 底下將分別對這些數字系統加以介紹。

4-1-2 常用的數字系統

- 十進位數字系統：十進位是一套以 10 爲基數, 逢 10 即進位的數字系統, 由 0、1、2、3、4、5、6、7、8、9 等十個數元所組成, 這套數字系統是目前人類世界中最被廣泛採用的一套系統。
- 二進位數字系統：二進位是一套以 2 爲基數, 逢 2 即進位的數字系統, 在此系統下, 任何數都只能用 0 和 1 兩種數元所組成的符號來表示。這套系統即是電腦所使用的數字系統。

常用的數字系統

- 八進位數字系統：八進位是一套以 8 為基數, 逢 8 即進位的數字系統, 在此系統下,我們只能使用 0、1、2、3、4、5、6、7 等八種數元, 如果運算過程中產生了大於或等於 8 的數, 便要往前進位。

常用的數字系統

- 十六進位數字系統：十六進位是一套以 16 爲基數, 逢 16 即進位的數字系統, 此數字系統是由 0、1、2、3、4、5、6、7、8、9、A、B、C、D、E、F 等十六個數元所組成, 相對於十進位來看的話, A=10、B=11、C=12、D=13、E=14 、F=15 。
- 下表將十進位數字 0 到 15 分別以二、八與十六進位來表示：

常用的數字系統

十進位數字系統	二進位數字系統	八進位數字系統	十六進位數字系統
0	0000	00	0
1	0001	01	1
2	0010	02	2
3	0011	03	3
4	0100	04	4
5	0101	05	5
6	0110	06	6
7	0111	07	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

圖表 4-1　十、二、八、十六進位數字系統對照表

TIP

一般二進位數值我們都會加個小括弧並標註 2，例如：$(101101)_2$，而十六進位則標註 16，例如 $(ACD8)_{16}$，依此類推。

4-2 數字系統間的轉換

- 由於電腦內部是以二進位形式來處理資料, 所以當我們輸入資料時, 電腦會自動將它轉換成二進位的形式。
- 以下就讓我們進一步來探討各數字系統之間互相轉換的方法。

數字系統間的轉換

- 4-2-1 二進位與十進位間的轉換
- 4-2-2 八進位與十進位間的轉換
- 4-2-3 十六進位與十進位間的轉換
  - 與十進位互轉的通則
- 4-2-4 八進位與二進位間的轉換
- 4-2-5 十六進位與二進位間的轉換
- 4-2-6 十六進位與八進位間的轉換
  - 二、八、十六進位轉換的通則

4-2-1 二進位與十進位間的轉換

- 二進位轉換成十進位
- 十進位轉換成二進位
  - 整數部分
  - 小數部分

二進位轉換成十進位

- 二進位轉換成十進位, 其二進位整數部份, 在小數點左邊第一位的位值爲 2^0、第二位的位值爲 2^1、第三位的位值爲 2^2 ...；而小數部分, 在小數點右邊第一位的位值爲 2^{-1}、第二位的位值爲 2^{-2} ...等依序類推, 只要將每一個二進位數乘以該數的位值, 然後相加即可獲得相對應的十進位數值。
- 以下我們以$(11101.11)_2$ 轉換成十進位來做示範。

二進位轉換成十進位

$(1\ 1\ 1\ 0\ 1\ .\ 1\ 1)_2$ — $_2$ 表示二進位

位值爲 2^{-2}

位值爲 2^{-1}

位值爲 2^{0}

位值爲 2^{1}

位值爲 2^{2}

位值爲 2^{3}

位值爲 2^{4}

$$\begin{aligned}(11101.11)_2 &= 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} \\ &= 16 + 8 + 4 + 0 + 1 + 0.5 + 0.25 \\ &= (29.75)_{10}\end{aligned}$$

圖表 4-2　二進位轉換成十進位的示範

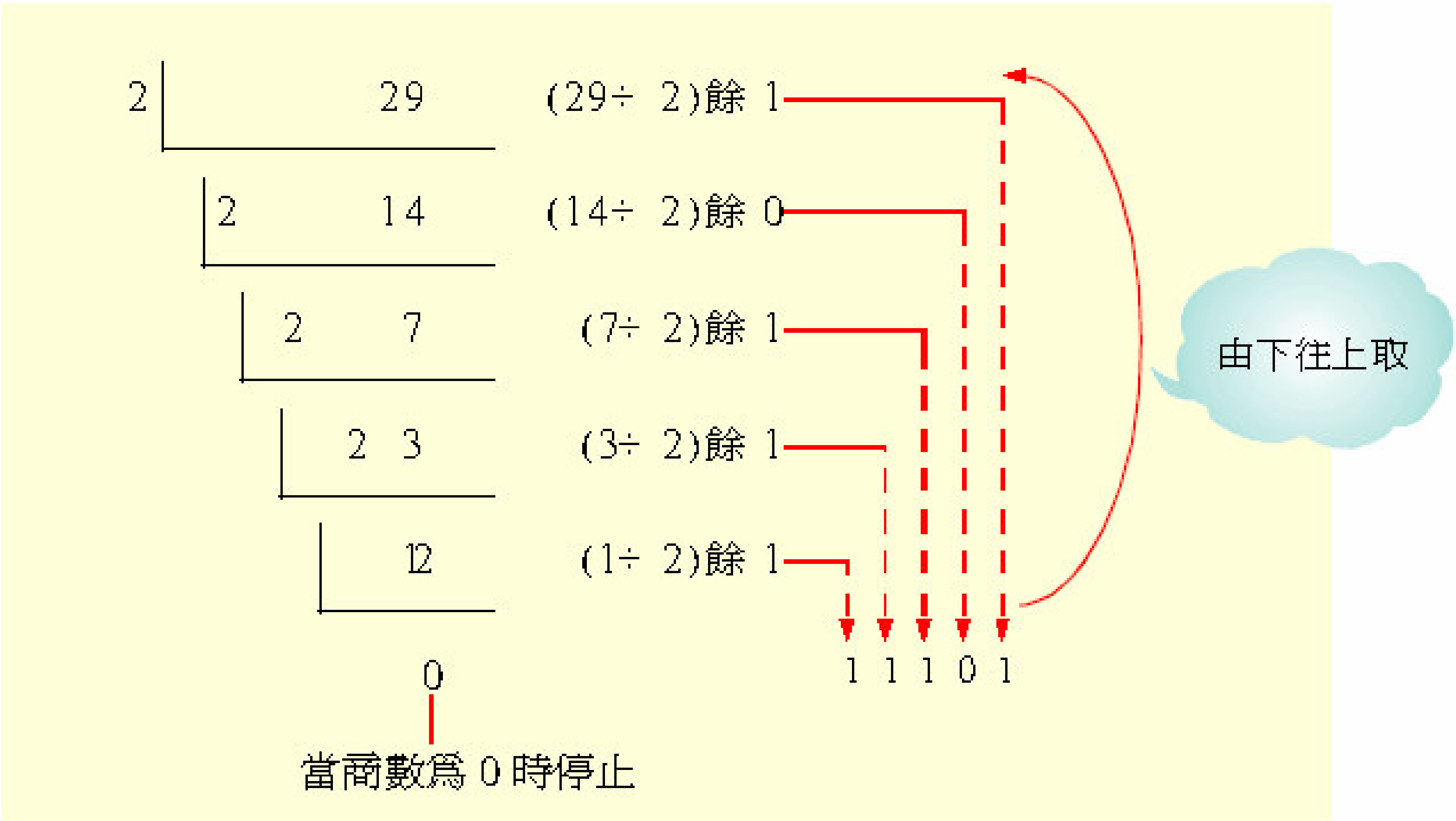
十進位轉換成二進位

- 將十進位轉換成二進位, 可分爲兩個部份來處理；在此我們以 (29.75)10 轉換成二進位來做示範。

整數部分

- 採連續除以 2 , 並保留 「餘數」 , 直到除法運算後的商數爲 0 時停止；然後由最後一次產生的餘數開始, 依序由左向右排列, 即可完成整數部分的轉換。

整數部分

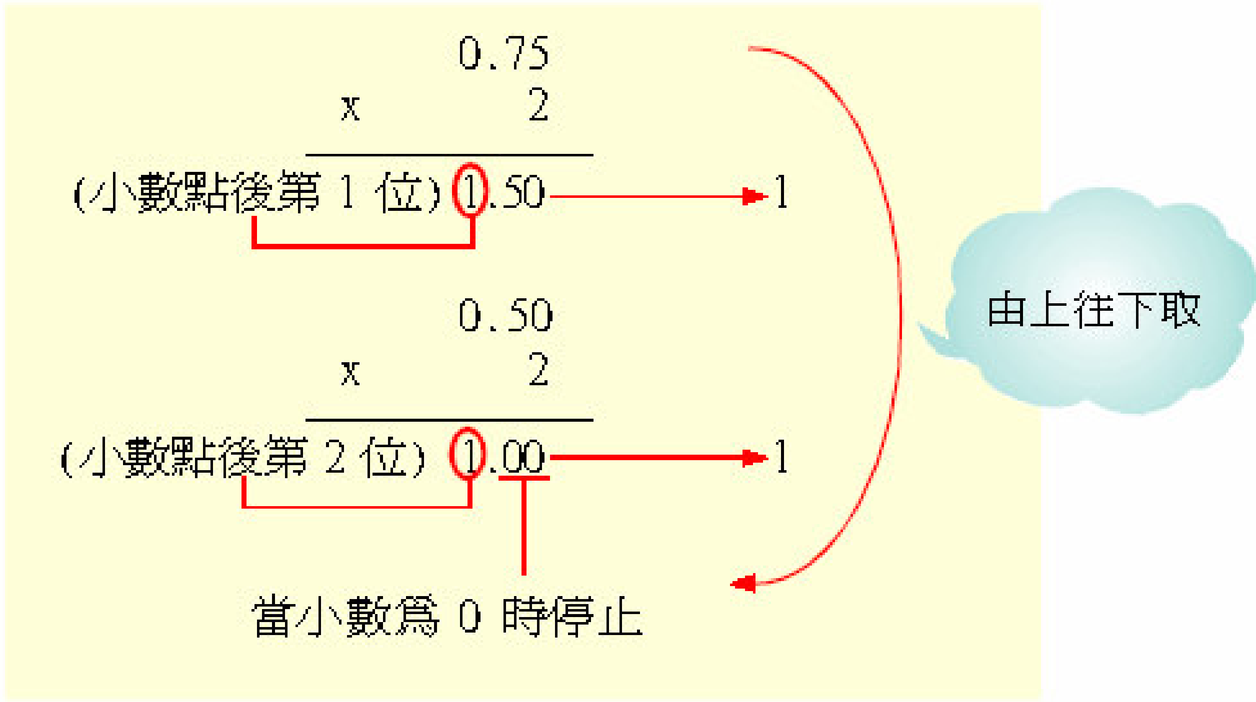


圖表 4-3　十進位整數部份轉換成二進位的示範

小數部分

- 將小數部份乘以 2 , 保留所得乘積的「整數部分」，, 繼續將乘法運算後所得的小數部分乘以 2 , 直到所得的小數爲 0 時停止；然後由第一次取得的整數開始, 依序由左向右排列, 即可完成小數部分的轉換。

小數部分



圖表 4-4 十進位小數部份轉換成二進位的示範

- 最後將整數部份加上小數部份：11101 + 0.11 = 11101.11。所以 $(29.75)_{10} = (11101.11)_2$

4-2-2 八進位與十進位間的轉換

- 八進位轉換成十進位
- 十進位轉換成八進位
  - 整數部分
  - 小數部分

八進位轉換成十進位

- 八進位的轉換原理和二進位相同, 其八進位整數部份, 在小數點左邊第一位的位值爲 8^0、第二位的位值爲 8^1 ...。而小數部份, 在小數點右邊第一位的位值爲 8^{-1}、第二位的位值 8^{-2} ...。因此八進位轉換成十進位, 只要將每一個八進位數乘以該數的位值,然後相加即可求得；在此我們以 $(127.3)_8$ 轉換成十進位來做示範。

八進位轉換成十進位

$$
\begin{aligned}
(127.3)_8 &= 1x8^2 + 2x8^1 + 7x8^0 + 3x8^{-1} \\
&= 1x64 + 2x8 + 7x1 + 3x0.125 \\
&= (87.375)_{10}
\end{aligned}
$$

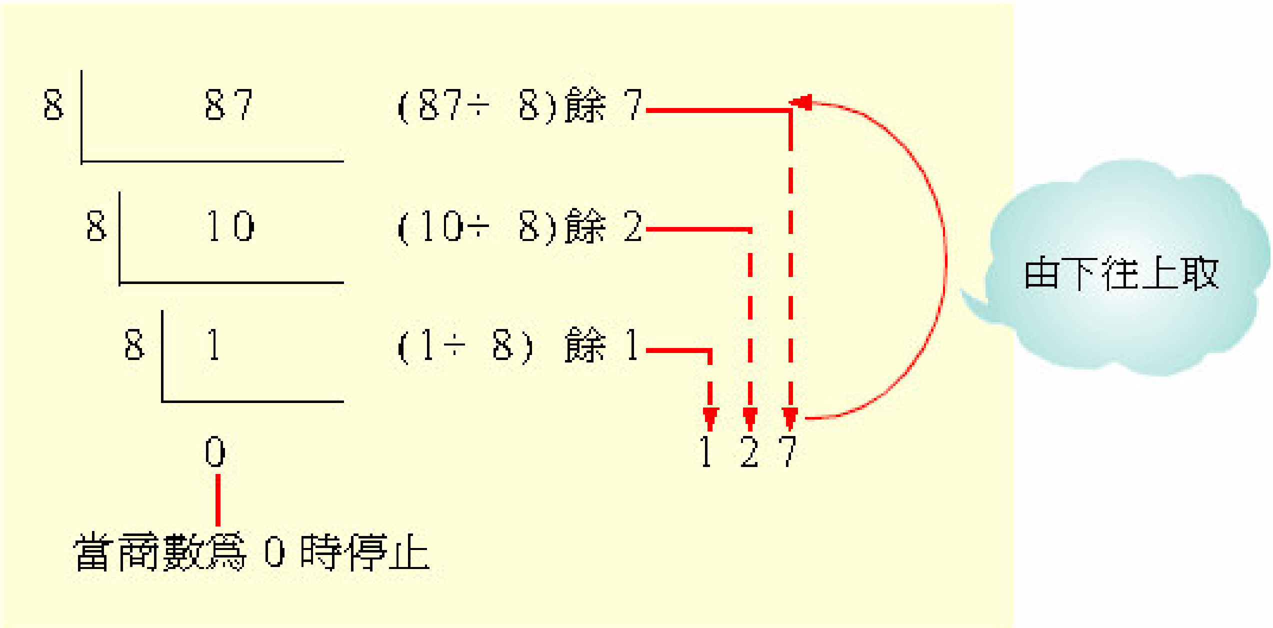
圖表 4-5　八進位轉換成十進位的示範

十進位轉換成八進位

- 要將十進位轉成八進位, 同樣地可分為整數與小數兩部份來處理；在此我們以 $(87.375)_{10}$ 來示範。

整數部分

- 採連續除以 8 , 並保留「餘數」, 直到除法運算後的商數爲 0 時停止；然後由最後一次產生的餘數開始, 依序由左向右排列, 即可完成整數部分的轉換。

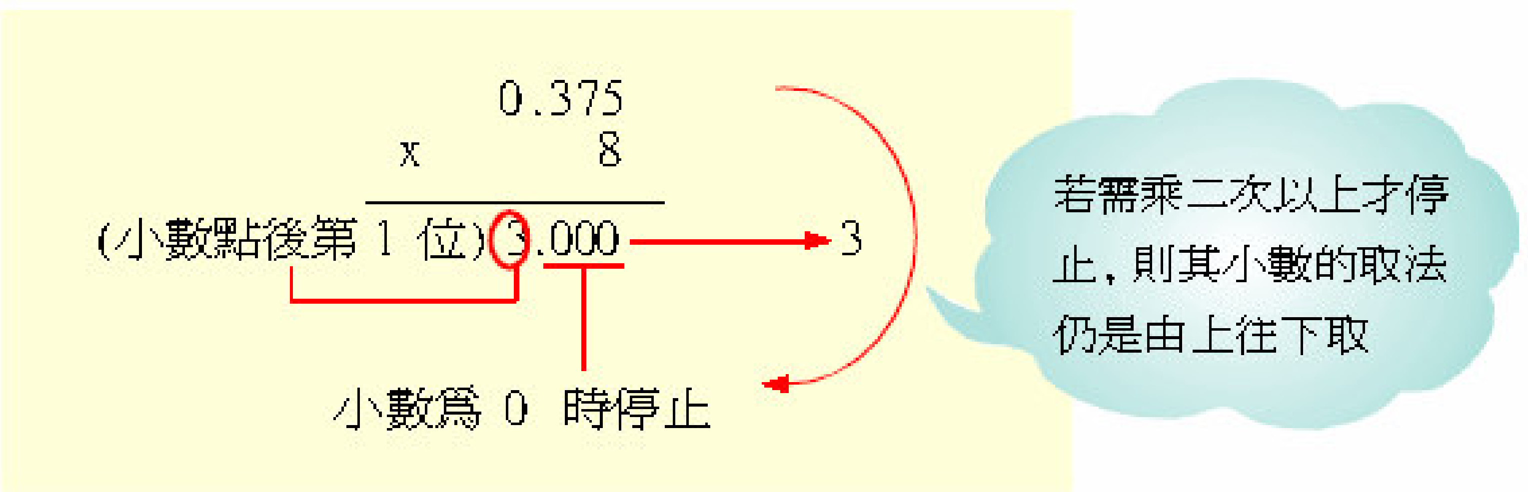


圖表 4-6 十進位整數部份轉換成八進位的示範

小數部分

- 將小數部份乘以 8 , 保留所得乘積的 「整數部分」, 繼續將乘法運算後所得的小數部分乘以 8 , 直到所得的小數爲 0 時停止；然後由第一次取得的整數開始, 依序由左向右排列, 即可完成小數部分的轉換。

小數部分



圖表 4-7　十進位小數部份轉換成八進位的示範

- 最後將整數部份加上小數部份：127 + 0.3 = 127.3。所以 $(87.375)_{10} = (127.3)_8$

4-2-3 十六進位與十進位間的轉換

- 十六進位轉換成十進位
- 十進位轉換成十六進位
  - 整數部分
  - 小數部分

十六進位轉換成十進位

- 十六進位的轉換原理和二進位相同, 其十六進位整數部份, 在小數點左邊第一位的位值爲 16^0、第二位的位值爲 16^1 ...。而小數部份, 在小數點右邊第一位的位值爲16^{-1}、第二位的位值爲 16^{-2} ...。因此十六進位轉換成十進位, 只要將每一個十六進位數乘以該數的位值, 然後相加即可求得；在此我們以 $(BCE.1E)_{16}$ 轉換成十進位來做示範。

十六進位轉換成十進位

$$
\begin{aligned}
(BCE.1E)_{16} &= 11x16^{2} + 12x16^{1} + 14x16^{0} + 1x16^{-1} + 14x16^{-2} \\
&= 11x256 + 12x16 + 14x1 + 1x0.0625 + 14x0.0039062 \\
&= (3022.67966868)_{10}
\end{aligned}
$$

圖表 4-8　十六進位轉換成十進位的示範

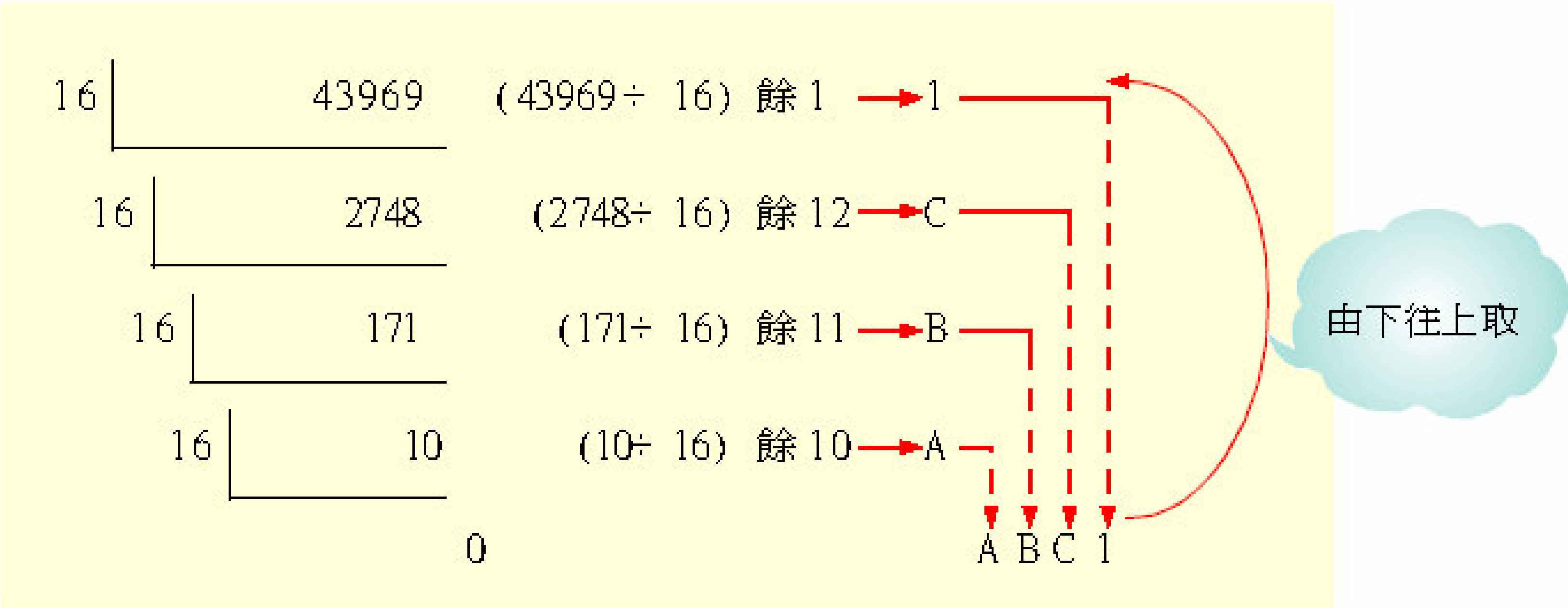
十進位轉換成十六進位

- 十進位轉成十六進位的方式, 亦分爲整數與小數兩部份來處理；在此以 $(43969.6719)_{10}$ 轉成十六進位來做示範。

整數部分

- 採連續除以 16 , 並保留「餘數」, 直到除法運算後的商數爲 0 時停止；然後由最後一次產生的餘數開始, 依序由左向右排列, 即可完成整數部分的轉換。

整數部分

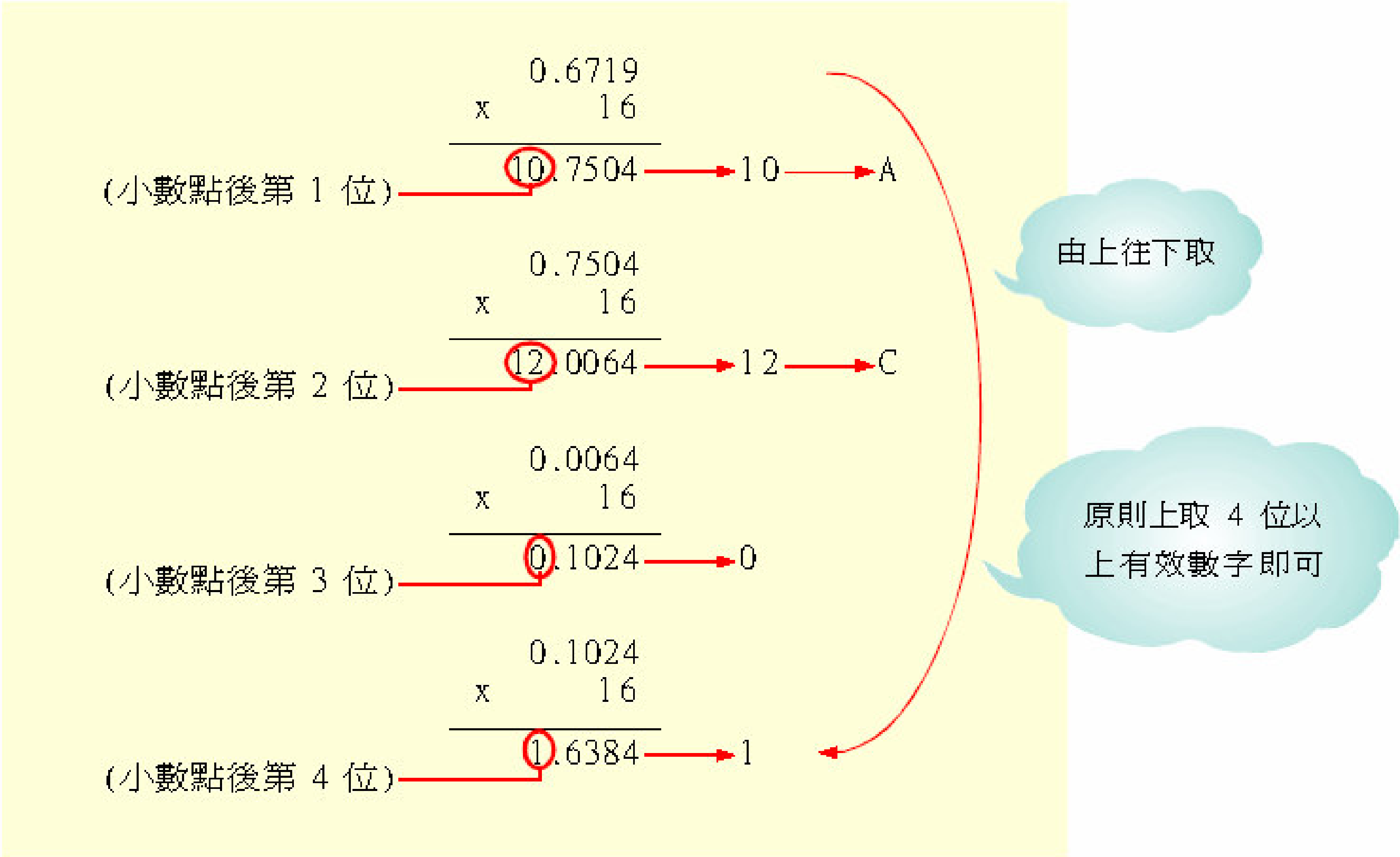


圖表 4-9 十進位整數部份轉換成十六進位的示範

小數部分

- 將小數部份乘以 16 , 保留所得乘積的「整數部分」, 繼續將乘法運算後所得的小數部分乘以 16 , 直到所得的小數爲 0 時停止；然後由第一次取得的整數開始, 依序由左向右排列, 即可完成小數部分的轉換。

小數部分



圖表 4-10　十進位小數部份轉換成十六進位的示範

小數部分

- 最後將整數部份加上小數部份：ABC1+ 0 . AC01 =ABC1.AC01 。
- 所以$(43969.6719)_{10} = (ABC1.AC01)_{16}$

與十進位互轉的通則

- 十進位轉成 r 進位的原則：整數部份除以 r , 由下往上取餘數；小數部份乘以 r , 然後由上往下取整數。
- r 進位轉成十進位的原則：將每個位數乘以對應位值後, 全部相加即可。

4-2-4 八進位與二進位間的轉換

- 二進位與八進位互相轉換時, 請以 3 個 1組為單位來轉換會較為方便。圖表 4-11 為八進位數與等值的二進位數之對照表：

八進位數值	等值之二進位數值
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

圖表 4-11
八進位與二進位等值對照表

八進位與二進位間的轉換

- 二進位轉換成八進位
- 八進位轉換成二進位

二進位轉換成八進位

- 欲將二進位轉換成八進位, 只要將二進位的整數部份由右至左, 每 3 個分成 1 組, 不足 3 個即往前補 0；小數部份則由左至右每 3 個分成 1 組, 不足往後補 0 , 然後再對照上表, 將其轉換成對應的八進位數即可。
- 在此我們以 $(11100001110.101)_2$ 轉換成八進位來做示範。

二進位轉換成八進位

$$(11100001110.101)_2 = \underbrace{011}_{3}\ \underbrace{100}_{4}\ \underbrace{001}_{1}\ \underbrace{110}_{6}\ .\ \underbrace{101}_{5}$$

$$= (3416.5)_8$$

圖表 4-12　二進位轉換成八進位的示範

八進位轉換成二進位

- 同理, 當八進位要轉換成二進位時, 只要將八進位的數值轉換成每 3 個 1 組的二進位數值即可；在此我們以 $(3416.5)_8$ 轉換成二進位爲例來做示範。

$$
\begin{aligned}
(3416.5)_8 &= (\underset{3}{011}\ \underset{4}{100}\ \underset{1}{001}\ \underset{6}{110}\ .\ \underset{5}{101})_2 \\
&= (11100001110.101)_2
\end{aligned}
$$

圖表 4-13　八進位轉換成二進位的示範

4-2-5 十六進位與二進位間的轉換

- 二進位與十六進位轉換時, 採每 4 個 1 組爲單位來轉換會較爲方便。下表爲十六進位與等值的二進位數對照表：

十六進位與二進位間的轉換

十六進位數值	等值之二進位數值
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

圖表 4-14　十六進位與二進位等值對照表

十六進位與二進位間的轉換

- 二進位轉換成十六進位
- 十六進位轉換成二進位

二進位轉換成十六進位

- 欲將二進位轉換成十六進位, 只要將二進位數的整數部份, 由右往左每 4 個 1 組進行轉換, 不到 4 個時, 就在前端補 0；小數部份則是由左往右每 4 個 1 組進行轉換, 不到4 個時, 就在後面補 0 , 然後再對照上表, 將其轉換成對應的十六進位數即可；在此我們以 $(1011111001.0011101)_2$ 轉換成十六進位來做示範。

二進位轉換成十六進位

$$(1011111001.0011101)_2 = (\underbrace{0010}_{2}\ \underbrace{1111}_{F}\ \underbrace{1001}_{9}\ .\ \underbrace{0011}_{3}\ \underbrace{1010}_{A})_2$$

$$= (2F9.3A)_{16}$$

圖表 4-15　二進位轉換成十六進位的示範

十六進位轉換成二進位

- 同理, 當十六進位要轉換成二進位時, 只要將十六進位的數值轉換成每 4 個 1 組的二進位數值即可；在此我們以 $(2F9.3A)_{16}$ 轉換成二進位為例來做示範。

$$(2F9.3A)_{16} = (\underbrace{0010}_{2}\ \underbrace{1111}_{F}\ \underbrace{1001}_{9}\ \underset{.}{.}\ \underbrace{0011}_{3}\ \underbrace{1010}_{A})_2$$
$$= (1011111001.0011101)_2$$

圖表 4-16　十六進位轉換成二進位的示範

4-2-6 十六進位與八進位間的轉換

- 八進位轉換成十六進位
- 十六進位轉換成八進位

八進位轉換成十六進位

- 欲將八進位轉換成十六進位, 可以先轉換成二進位數字, 再將二進位轉換成十六進位即可；在此我們以 $(346.7)_8$ 轉換成十六進位爲例來做示範：

$$(346.7)_8 = (\underline{011}\ \ \underline{100}\ \ \underline{110}\ \ .\underline{111})_2$$

$$= (0\ \ \underbrace{1110}_{E}\ \ \underbrace{0110}_{6}\ \ .\underbrace{1110}_{E})_2$$

$$= (E6.E)_{16}$$

圖表 4-17 八進位轉換成十六進位的示範

十六進位轉換成八進位

- 欲將十六進位轉換成八進位, 必須先轉換成二進位數字之後, 再由二進位轉成八進位即可。在此我們以 $(E6.E)_{16}$ 轉換成八進位為例來做示範：

$$
\begin{aligned}
(E6.E)_{16} &= (\underline{1110}\ \underline{0110}\ .\underline{1110})_2 \\
&= (\underbrace{011}_{3}\ \underbrace{100}_{4}\ \underbrace{110}_{6}\ .\underbrace{1110}_{7})_2 \\
&= (346.7)_8
\end{aligned}
$$

圖表 4-18　十六進位轉換成八進位的示範

二、八、十六進位轉換的通則

- 二進位與八進位：要以 3 個為 1 組來轉換。
- 二進位與十六進位：要以 4 個為 1 組來轉換。
- 八進位與十六進位：必須先轉成二進位後再進行轉換。

4-3 資料表示法

- 儲存在電腦中的資料, 基本上有兩種型態：一是數值資料, 另一種是文字資料。
- 底下我們將分別討論之。

資料表示法

- 4-3-1 數值資料表示法
  - 負數的表示法
  - 浮點資料表示法
- 4-3-2 文字資料表示法

4-3-1 數值資料表示法

- 數值資料可分爲整數與浮點數兩種：

數值資料
- 整數：不含小數的數值稱爲『整數』(Integer)，包括正數、負數、與 0。例如 3200，-1，+58...等這些數值資料都可稱爲整數。
- 浮點數：含有小數點的數值即爲『浮點數』(Floating-Point Number)，也就是我們一般所說的實數。例如 86.2564、0.317522、-148.87612 ...等這些數值都可稱爲浮點數。

圖表 4-19　數值資料表示法

數值資料表示法

- 數值資料中最常見的就是正、負整數的資料, 在電腦內部充滿著 0101010101 的訊號, 可以用二進位數來表示。但是這樣的二進位數字都只是正整數而已, 電腦內部並沒有 "+"、"-" 等符號來表示正、負數, 也沒有表示小數點的符號。因此, 爲了解決這些問題, 便有人提出幾種不同的負數與浮點數表示方法。

負數的表示法

- 負數在電腦內部的表示法, 常用的有最高位元表示法和 2 的補數表示法兩種, 我們以一個整數佔 8 個 Bits 爲範例, 分別介紹如下：
- 最高位元表示法：顧名思義是以最高位元來表示正負號, 最高位元爲 0 表示正數,最高位元爲 1 表示負數, 如右表：

負數的表示法

十進位正數	二進位正數	十進位負數	二進位負數
0	00000000	0	10000000
1	00000001	-1	10000001
2	00000010	-2	10000010
.	0......	.	1.....
.	0......	.	1.....
.	0......	.	1......
127	01111111	-127	11111111

圖表 4-20　最高位元表示法

負數的表示法

- 2 的補數表示法：最高位元表示法有個缺點, 就是產生了 2 個 0 (+0 與 -0), 使得原來總共可以表示 256 個數, 便成只能表示 255 個數。而 2 的補數法正可以改良此缺點。使用 2 的補數表示法時, 一個數的負數即是將該數的每一位元取其反相 (即 1 變0 , 0 變 1) 再加 1 。如下表：

負數的表示法

十進位正數	二進位正數 →	0、1互換，再加1 →	二進位負數	十進位負數
0	00000000 →	11111111+1 →	00000000	0 (進位去掉)
1	00000001 →	11111110+1 →	11111111	-1
2	00000010 →	11111101+1 →	11111110	-2
.	......	.....		.....
.	......	.....		.....
.	......	.....		.....
127	01111111 →	10000000+1 →	10000001	-127
128	無	無 →	10000000	-128

圖表 4-21　2 的補數表示法

負數的表示法

- 由上表可知, 只有在 0 時, 該數的反向加 1 才會使得最高位元進位, 此時該進位的位元捨棄, 使得該數 (0) 的正負數表示法相同；因此以 2 的補數法來表示負數, 就不會有「兩種方式表示 0」的困擾了。結果正數由 1 ~ 127 總共有 127個；負數由 -1 ~ -128 再加上 1 個 0 , 所以 8 個位元總共可以表示 256 個數。

負數的表示法

- 將二進位數值 0、1 互換 (即 1 變 0 ,0 變 1) 的動作, 一般稱爲 1 的補數表示法；當求得 1 的補數後, 只要再加 1 , 即可求得該數之 2 的補數。

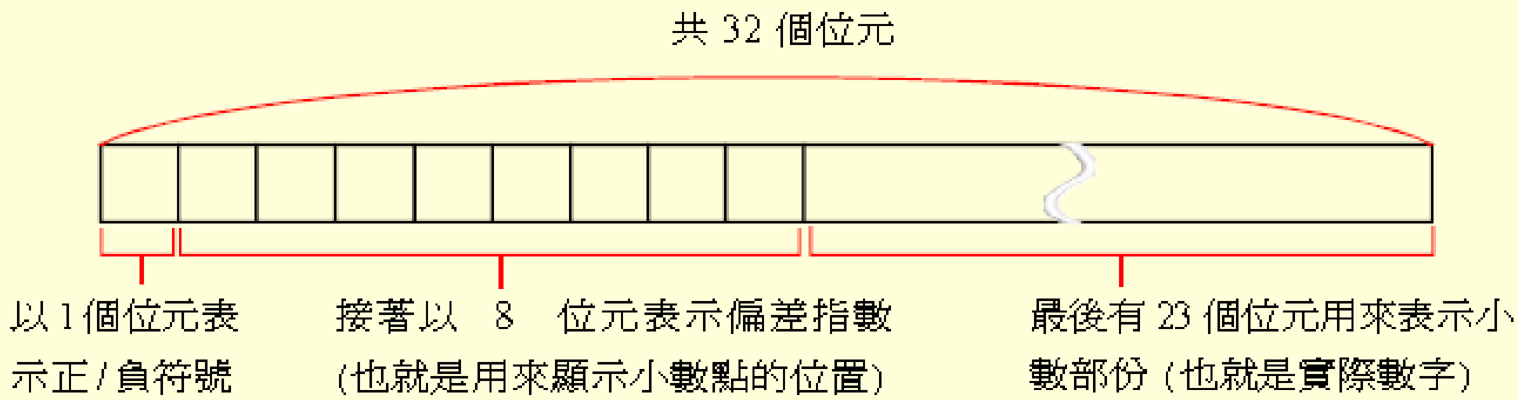
浮點資料表示法

- 浮點數的表示方法相對於正負整數的表示方法, 最主要的差別就在於小數點的位置。
- 對於正負整數來說, 小數點都固定在最右邊, 所以其表示法又稱爲定點表示法, 定點表示法的規定在所有電腦中都一樣；而浮點數的小數點則是不固定的, 而且其顯示方式還因電腦型態不同而有異。

浮點資料表示法

- 以 486 電腦的浮點表示法爲例, 有單精確度 (以 32 位元來顯示浮點數)、雙精確度(以 64 位元來顯示浮點數)和延伸精確度(以 80 位元來顯示浮點數) 等三種。以下是單精確度的表示方法：

浮點資料表示法



圖表 4-22　單精確度的浮點資料表示法

浮點資料表示法

- 正/負符號：0 表示正數, 1 表示負數。
- 偏差指數： 8 個位元可以顯示 0 ~255 個數值, 但必須能顯示正負兩種指數, 故以 127 爲指數偏差值, 將指數值+指數偏差值就等於偏差指數。
- 小數部份：這裡的小數部份是指以二進位形式, 且正規化後的浮點數之小數部份。

浮點資料表示法

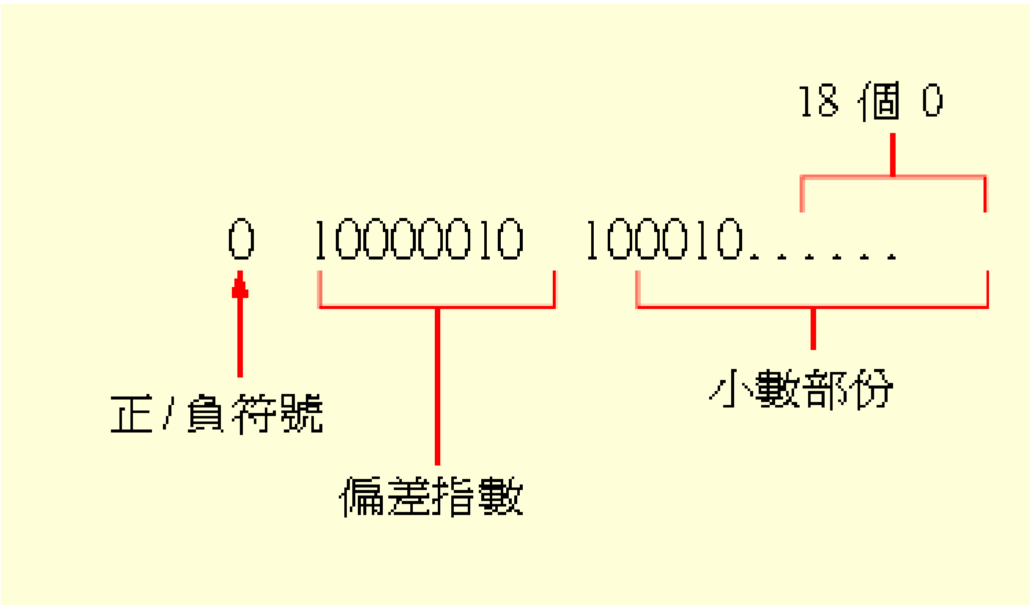
- 接著讓我們以 $(12.25)_{10}$ 爲例, 看看如何以單精確度的浮點表示法來表示。首先,它是個正數, 所以第一個位元已經確定是 0 , 接著將之轉換爲二進位 $(12.25)_{10} = (1100.01)_2$, 再正規化成 1.10001×2^3 , 此時便可計算出偏差指數爲 3+127 = 130 , 再把 130 轉換爲以 8 位元顯示的二進位 $(10000010)_2$, 於是可以得到偏差指數的部份就是10000010。

浮點資料表示法

- 最後是小數部份, 小數部份即是正規化後的二進位小數部份, 至於小數點前面那個 1 , 則因爲所有正規化的數字都有 1 , 所以電腦會自動記憶, 因此只要顯示後面的小數部份 10001 , 但小數共有 23 個位元, 所以把後面的位元補上 0 即可。

浮點資料表示法

- 最後得到的浮點數表示如下：



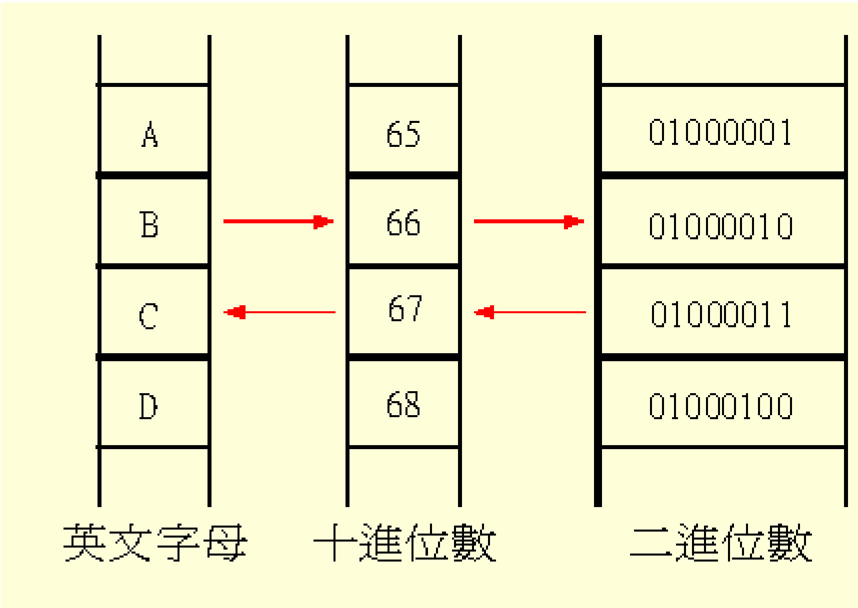
圖表 4-23　單精確度浮點表示法的示範

4-3-2 文字資料表示法

- 電腦不但可以幫我們做快速精確的數值運算, 還可以做文字資料的處理；要處理文字資料, 就必須先瞭解文字資料在電腦中的表示方法。
- 數值資料在電腦內部是以「二進位」的形式來表示, 這是因爲數位電腦只能處理二進位形式的資料；同樣地, 我們輸入電腦的文字資料, 亦會被轉換成二進位碼的形式儲存。

文字資料表示法

- 這種將文字資料轉換成二進位碼的系統就稱為編碼系統。以英文字母為例, 我們可以建一個編碼表：



圖表 4-24　編碼表

文字資料表示法

- 我們把每一個英文字母編上一個號碼, 建成一套編碼表。這個編碼表可以存放在鍵盤內的電路上。當我們按下一個鍵時, 鍵盤的電路便依表格的規定, 把該鍵對應的二進位碼送往電腦主機 (例如按下 A 鍵, 鍵盤便會把其相對應碼 65 的二進位數 01000001輸入電腦)。

文字資料表示法

- 同樣地, 當我們想將文字資料輸出時, 編碼系統亦會將二進位碼轉換成對應的字元符號, 再藉由輸出設備顯示或列印出來。
- 電腦是以數字碼來表示文字資料, 例如 65 代表 "A", 66 代表 "B"。像這種以數值代表字元的方式稱爲編碼, 不同的系統可能會使用不同的編碼方式。

文字資料表示法

- 常用的英文電腦編碼系統
- 常用的中文電腦編碼系統
- 利於網路應用的統一編碼系統
  - 出現亂碼?

常用的英文電腦編碼系統

- 目前在 PC 上常用的英文電腦編碼系統有 3 種, 分別是：
- ASCII 碼：是美國標準資訊交換碼的縮寫 (American Standard Code for Information Interchange)；它不但包含英文大、小寫字母, 還有阿拉伯數字、控制字元以及各種符號等。ASCII 碼共有 128 個, 每一個碼都對應一個字元, 在電腦中是以 1 個 Byte 來儲存。其數值與字元的對應表如下：

常用的英文電腦編碼系統

0		16	►	32		48	0	64	@	80	P	96	`	112	p
1	☺	17	◄	33	!	49	1	65	A	81	Q	97	a	113	q
2	☻	18	↕	34	"	50	2	66	B	82	R	98	b	114	r
3	♥	19	‼	35	#	51	3	67	C	83	S	99	c	115	s
4	♦	20	¶	36	$	52	4	68	D	84	T	100	d	116	t
5	♣	21	§	37	%	53	5	69	E	85	U	101	e	117	u
6	♠	22	▬	38	&	54	6	70	F	86	V	102	f	118	v
7	•	23	↨	39	'	55	7	71	G	87	W	103	g	119	w
8	◘	24	↑	40	(	56	8	72	H	88	X	104	h	120	x
9	○	25	↓	41	)	57	9	73	I	89	Y	105	i	121	y
10	◙	26	→	42	*	58	:	74	J	90	Z	106	j	122	z
11	♂	27	←	43	+	59	;	75	K	91	[	107	k	123	{
12	♀	28	∟	44	,	60	<	76	L	92	\	108	l	124	|
13	♪	29	↔	45	-	61	=	77	M	93	]	109	m	125	}
14	♫	30	▲	46	.	62	>	78	N	94	^	110	n	126	~
15	☼	31	▼	47	/	63	?	79	O	95	_	111	o	127	⌂

圖表 4-25　ASCII 數值與字元的對應表

常用的英文電腦編碼系統

- ISO8859 碼：ISO 將編碼系統由 7 個位元擴充到 8 個位元, 其中 0 到 127 的編碼與 ASCII 碼相容, 128 到 255 碼則依地區不同, 包含更多的特殊字元 (如拉丁字母、特殊字母的上下標與其他符號等)。ISO8859 的編碼系統又依地區語系區分成幾個部分，例如丹麥和芬蘭等語系使用 ISO8859-1 的編碼系統, 而羅馬尼亞和波蘭等語系則採用 ISO8859-2 的編碼系統。

常用的英文電腦編碼系統

- EBCDIC 碼：EBCDIC 碼的全名是 Extended Binary Code Decimal Interchange Code , 是美國 IBM 公司所制定的編碼系統。EBCDIC 碼的每一個字元是由 8 個位元所組成, 共有 28 種組合, 可以表示 256 個字元。

常用的中文電腦編碼系統

- 在英文電腦中, 8 個位元 (28 =256) 就足以表示所有的英文字母、阿拉伯數字及許多特殊符號；但是, 8 個位元卻不足以表示所有的中文字, 所以中文字是以 2 Bytes(16 Bits) 來編碼。
- 由於這些碼都是中文字儲存在電腦內部時的編碼, 所以又稱爲中文的內碼。
- 以下就針對目前比較常使用的編碼方式做介紹：

常用的中文電腦編碼系統

- Big-5：由台北市電腦公會聯合業者共同制定的編碼系統, 包含常用字、次常用字以及各式符號和擴充字。其字符空間是由非連續的 94X157 的矩陣構成，共可容納 14, 758 個位元。

常用的中文電腦編碼系統

- MS950 碼：由微軟公司開發, 應用於 Windows 的作業系統中, 由於使用者很多, 近來也成爲常見的編碼方式；原本 Windows 作業系統內部是採用 Unicode 的編碼方式, 但爲了配合多國語言使用, 系統會根據使用者的國籍設定, 在輸出、輸入和顯示時, 會搭配以不同的 MS***編碼方式呈現, MS950 便是代表繁體中文的編碼方式, 可是如果是泰文就會轉以 MS874 的編碼呈現囉!

利於網路應用的統一編碼系統

- 近年來爲了便利網路上的相關應用, 也出現了 Unicode 這種編碼, 它將亞洲國家的常用、共用字編在一起, 藉此達成統一編碼的目的。

利於網路應用的統一編碼系統

- Unicode 碼：Unicode 碼的全名爲 Universal Multiple Octet Coded Character Set, 是由 Unicode 國際標準組織針對各國文字、符號制定的統一性編碼系統。Unicode 是採用 16 bits 的編碼架構, 以 2 個 8bits 的位元組合併而成, 其中 0 到 127 的編碼與 ASCII 碼相容。內容包含符號、漢字、韓文拼音、造字區, 並保留部份擴充字元的空間。

出現亂碼?

- 當資料在電腦間傳輸時, 若兩端的電腦使用不相同的語系設定, 便可能會造成錯誤解讀而形成亂碼。
- 譬如亞洲的方塊字多半使用 2bytes 來編碼（如日本的 JIS 碼、簡體中文的 GB 碼…等等都是）, 但英語系和中歐語系的編碼則多使用 1byte 便足以編碼, 於是當電腦將一個 2bytes 的亞洲語系單字, 錯認為兩個 1byte 的英文語系字元來解碼時, 便會出現亂碼。

出現亂碼?

使用繁體中文（以 2bytes 為單位做編碼）來編輯文件

開啟文件時，使用英文（以 1byte 為單位做編碼）來解譯，於是形成亂碼

圖表 4-26　錯誤的語系解讀會形成亂碼