2-4　數值算術運算

2-4-1　加法

範例：$111010_2 + 11011_2$

$$
\begin{array}{lr}
\text{進位} & 11\ \ 1 \\
 & 00111010 \\
+ & 00011011 \\
\hline
 & 01010101
\end{array}
$$

2-4-2　減法

範例：$00001010_2 + 00000011_2$

$$
\begin{array}{lr}
\text{借位} & -1\,-1 \\
 & 0\;0\;0\;0\;1\;0\;1\;0 \\
- & 0\;0\;0\;0\;0\;0\;1\;1 \\
\hline
 & 1\;1\;1
\end{array}
$$

減法(以加法方式實現)

符號位元

(1)

$$\begin{array}{rr} & 00001010 \\ - & 00000011 \\ \hline \end{array}$$

(2)

$$\begin{array}{rrl} & 00001010 & \\ + & 11111101 & \text{(00000011的2's補數)} \\ \hline & \cancel{1}00000111 & \end{array}$$

$$00000111$$

- 範例: $5_{10} – 7_{10} = 101_2 - 111_2$

$$\begin{array}{rr} & \boxed{0}101 \\ - & \boxed{0}111 \\ \hline \end{array} \longrightarrow \begin{array}{rr} & \boxed{0}101 \\ + & \boxed{1}001 \\ \hline & \boxed{1}110 \end{array}$$

(-7 的2’s補數)

(-2 的2’s補數)

(轉換：除符號位元外，所有位元作2’s補數的轉換)

$\boxed{1}010$ (-2 的帶符號大小)

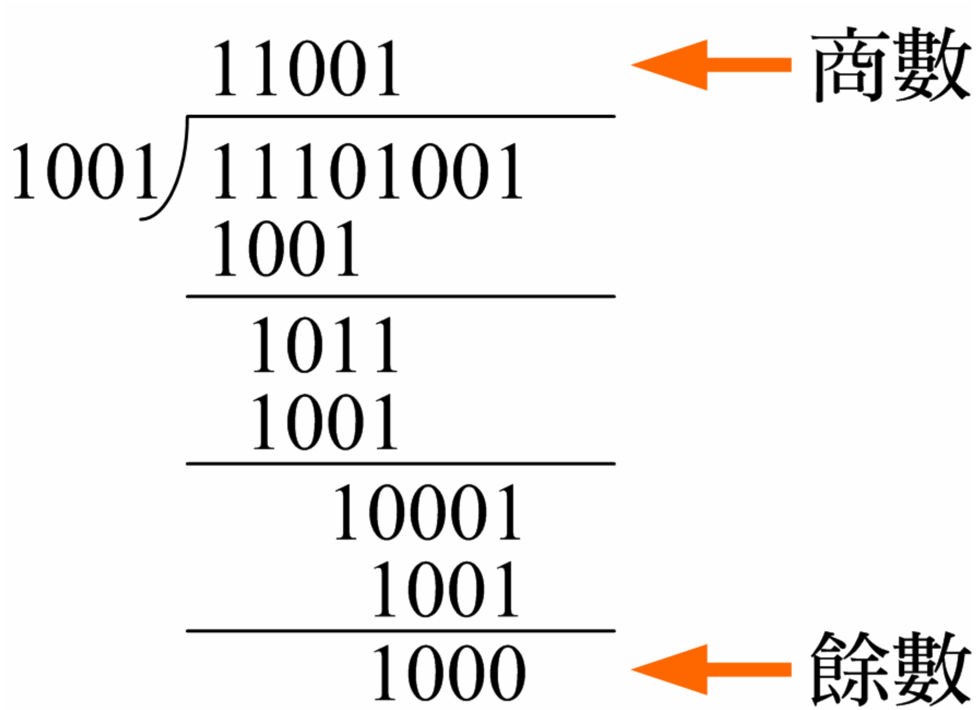
2-4-3 乘法

範例：$1101_2 \times 1011_2$

$$
\begin{array}{r}
1101 \\
\times \quad 1011 \\
\hline
1101 \\
1101 \\
0000 \\
1101 \\
\hline
10011111
\end{array}
$$

2-4-4　除法

範例：$11101001_2 \div 1001_2$



2-5 數碼系統

2-5-1 BCD碼(或8421碼)

十進位數字	BCD 碼	十進位數字	BCD 碼
0	0000	5	0101
1	0001	6	0110
2	0010	7	0111
3	0011	8	1000
4	0100	9	1001

數字 0 ～ 9 的 BCD 碼

2-5-2　2421碼

十進位數字	2421 碼	十進位數字	2421 碼
0	0000（0 + 0 + 0 + 0）	5	1011（2 + 0 + 2 + 1）
1	0001（0 + 0 + 0 + 1）	6	1100（2 + 4 + 0 + 0）
2	0010（0 + 0 + 2 + 0）	7	1101（2 + 4 + 0 + 1）
3	0011（0 + 0 + 2 + 1）	8	1110（2 + 4 + 2 + 0）
4	0100（0 + 4 + 0 + 0）	9	1111（2 + 4 + 2 + 1）

數字 0 ~ 9 的 2421 碼

2-5-3　84-2-1碼

十進位數字	84-2-1 碼	十進位數字	84-2-1 碼
0	0000 (0 + 0 − 0 − 0)	5	1011 (8 + 0 − 2 − 1)
1	0111 (0 + 4 − 2 − 1)	6	1010 (8 + 0 −2 − 0)
2	0110 (0 + 4 − 2 − 0)	7	1001 (8 + 0 − 0 − 1)
3	0101 (0 + 4 − 0 − 1)	8	1000 (8 + 0 − 0 − 0)
4	0100 (0 + 4 − 0 − 0)	9	1111 (8 + 4 − 2 − 1)

數字 0 ~ 9 的 84-2-1 碼

2-5-4 超三碼

十進位數字	超三碼	十進位數字	超三碼
0	0011	5	1000
1	0100	6	1001
2	0101	7	1010
3	0110	8	1011
4	0111	9	1100
數字 0 ~ 9 的超三碼			

- 2421碼、 84-2-1碼、超三碼具有自補特性(self-complementing)，也就是十進位數字的9’s補數等於其二進位的1’s補數。
- 例如，456的2421碼為0100 1011 1100，而456的9’s補數為543(2421碼為1011 0100 0011)，恰為0100 1011 1100的1’s補數。

$$456 \xrightarrow{\text{2421碼}} 0100\ 1011\ 1100$$

$$543 \xrightarrow{\text{2421碼}} 1011\ 0100\ 0011$$

2-5-7 葛雷碼

反射葛雷碼 (reflected gray codes) 公式：

$G_{n+1} = \{0G_n, 1G_n^{ref}\}$，$G_1 = \{0, 1\}$，$n >= 1$

$$G_1 = \begin{cases} 0 \\ 1 \end{cases}$$

$$G_2 = \begin{cases} 0\ \ 0 = 0 \\ 0\ \ 1 = 1 \\ 1\ \ 1 = 2 \\ 1\ \ 0 = 3 \end{cases} \qquad \begin{array}{l} \text{upper: } G_1 \\ \text{lower: } G_1^{ref} \end{array}$$

$$G_3 = \begin{cases} 0\ \ 00 = 0 \\ 0\ \ 01 = 1 \\ 0\ \ 11 = 2 \\ 0\ \ 10 = 3 \\ 1\ \ 10 = 4 \\ 1\ \ 11 = 5 \\ 1\ \ 01 = 6 \\ 1\ \ 00 = 7 \end{cases} \qquad \begin{array}{l} \text{upper: } G_2 \\ \text{lower: } G_2^{ref} \end{array}$$

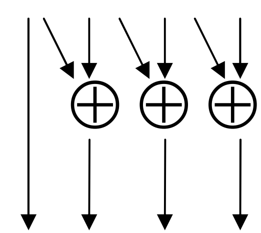
$$G_4 = \begin{cases} 0\ \ 000 = 0 \\ 0\ \ 001 = 1 \\ 0\ \ 011 = 2 \\ 0\ \ 010 = 3 \\ 0\ \ 110 = 4 \\ 0\ \ 111 = 5 \\ 0\ \ 101 = 6 \\ 0\ \ 100 = 7 \\ 1\ \ 100 = 8 \\ 1\ \ 101 = 9 \\ 1\ \ 111 = 10 \\ 1\ \ 110 = 11 \\ 1\ \ 010 = 12 \\ 1\ \ 011 = 13 \\ 1\ \ 001 = 14 \\ 1\ \ 000 = 15 \end{cases} \qquad \begin{array}{l} \text{upper: } G_3 \\ \text{lower: } G_3^{ref} \end{array}$$

若要將二進位數字$B_nB_{n-1}\ldots B_1$轉換成反射葛雷碼$G_nG_{n-1}\ldots G_1$，可以套用如下公式：

$$G_k = \begin{cases} B_k, & k = n \\ B_{k+1} \oplus B_k, & 1 \le k \le n-1 \end{cases}$$

範例:

二進位數字 1011

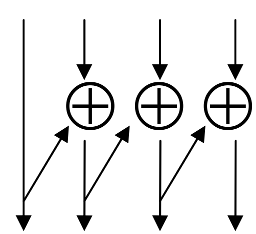


反射葛雷碼 1110

若要將反射葛雷碼$G_nG_{n-1}\ldots G_1$轉換成二進位數字$B_nB_{n-1}\ldots B_1$，可以套用如下公式：

$$B_k = \begin{cases} G_k, & k = n \\ B_{k+1} \oplus G_k, & 1 \le k \le n-1 \end{cases}$$

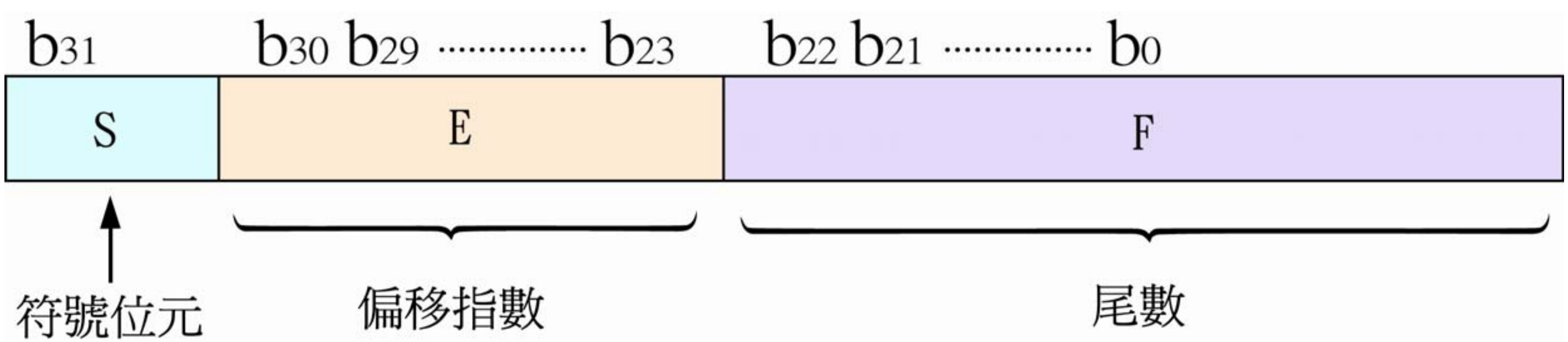
範例： 反射葛雷碼 1110



二進位數字 1011

2-6　浮點數表示法

- 符號位元 (sign bit)
- 偏移指數 (biased exponent)
- 尾數 (mantissa)、分數 (fraction)



- IEEE 標準格式(以2為基底)，所表示的浮點數為

$$(-1)^s \times 2^{E-127} \times 1.F$$

- 例如 $98.625_{10} = 0.1100010101 \times 2^7$

$$= 1.100010101 \times 2^6$$

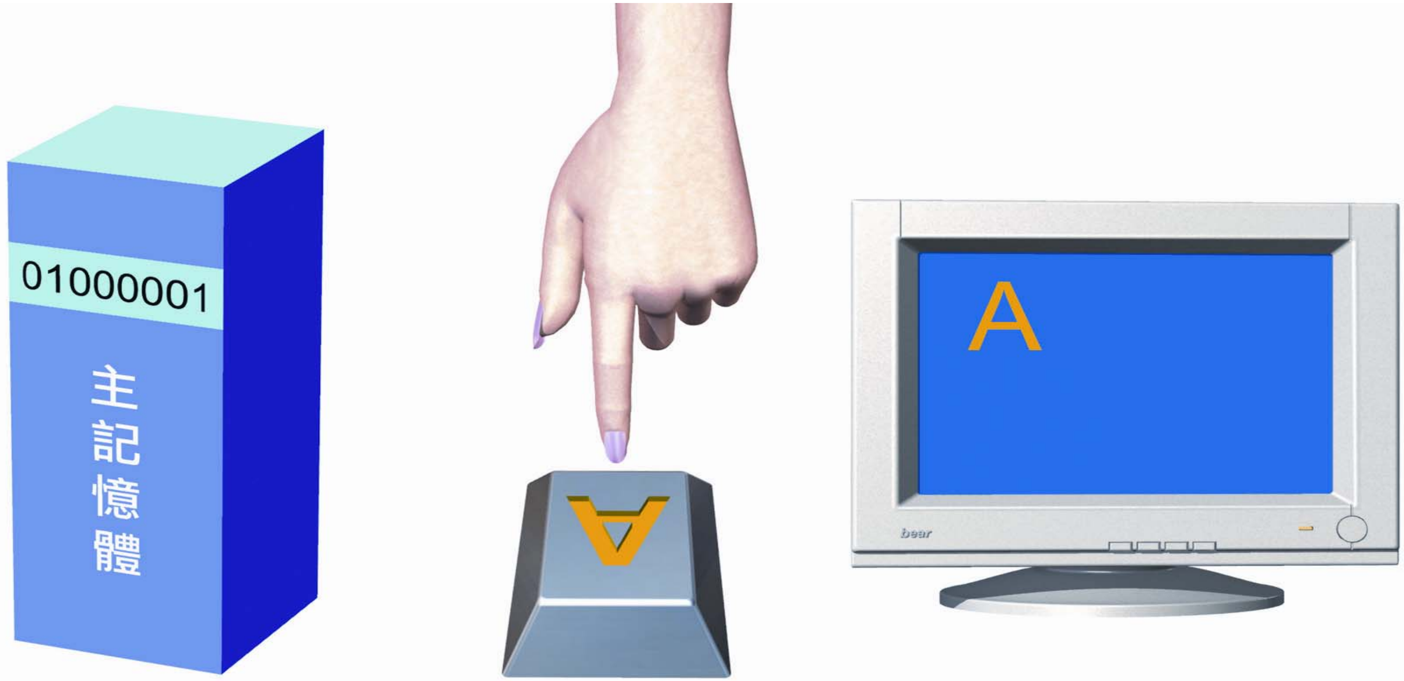
- 偏移指數 E = 127+6 =133_{10}= 10000101
- 浮點數表示為

0 10000101 1000101010 0000000000 000

2-7　文字表示法

- ASCII 是目前使用最廣泛的編碼系統，使用7位元來表示字元符號，但為了方便起見，ASCII編碼的字元符號是存放在一個位元組裡面。
- 繁體中文編碼系統，例如BIG5 (又稱為大五碼)、王安碼、CCCII碼，以BIG5碼最普遍，使用16位元來表示一個中文字，至於簡體中文則是以GB碼為主。

- EBCDIC和ASCII-8均使用8位元來表示字元符號，共256個字元。
- 近來又有另一套編碼系統叫做Unicode，這是使用16位元來表示字元符號，可以表示 2^{16} (65,536) 個字元，前128個字元符號和ACSII相同。

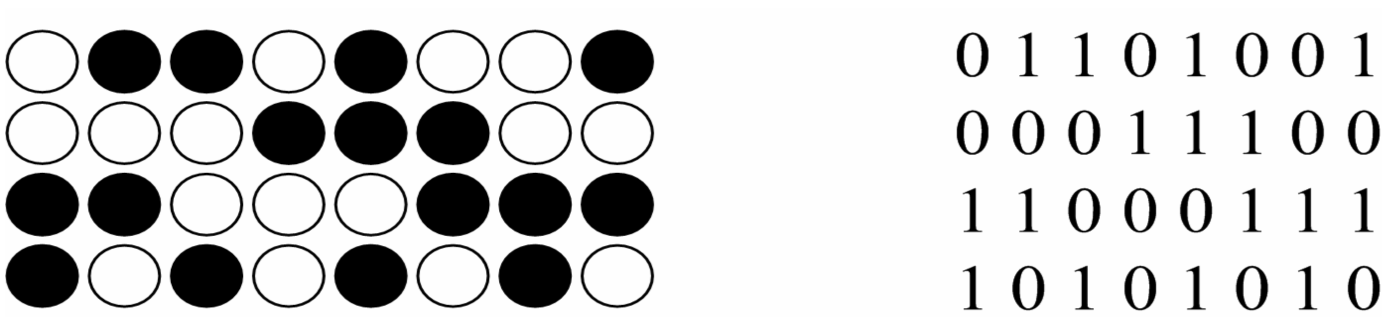


當您在鍵盤上按”A”鍵，電腦會自動轉換成ASCII碼0100001，然後存入主記憶體的位元組之中，並在螢幕上顯示”A” 。

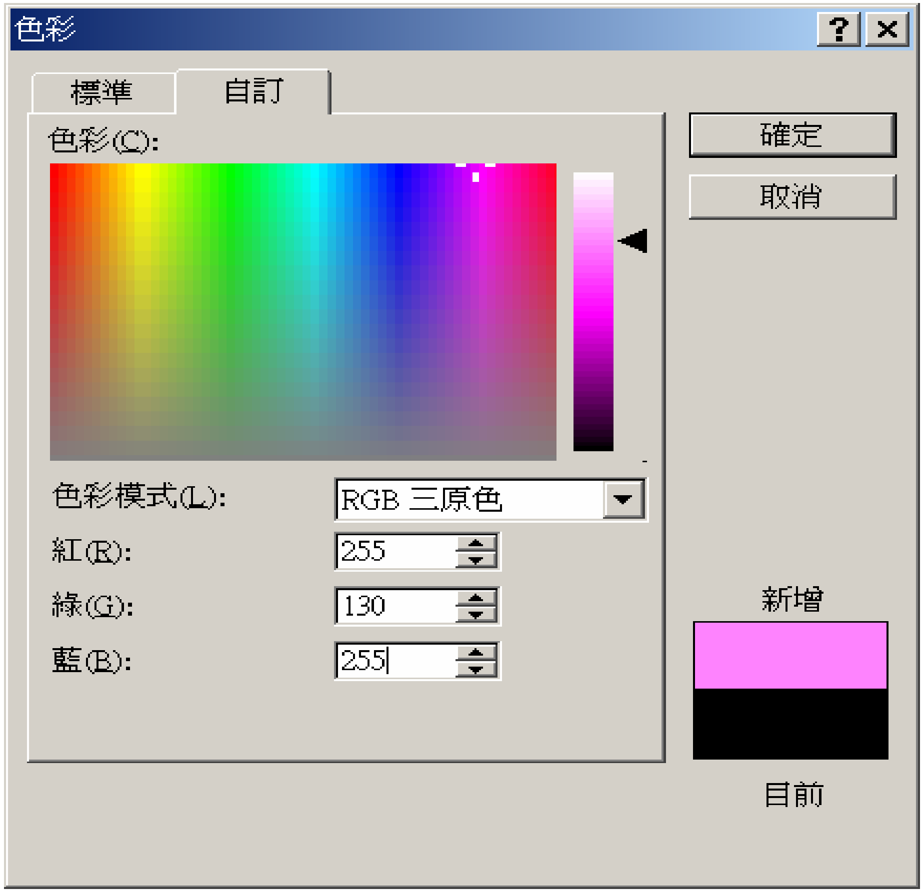
2-8　圖形表示法

2-8-1　點陣圖

- 點陣圖是將圖形當成許多點的集合，每個點稱為一個像素 (pixel)，像素愈多，解析度就愈高，畫質也愈細緻。
- 一張黑白點陣圖可以表示成一連串的像素，每個像素佔用1位元，以0或1表示黑色或白色。

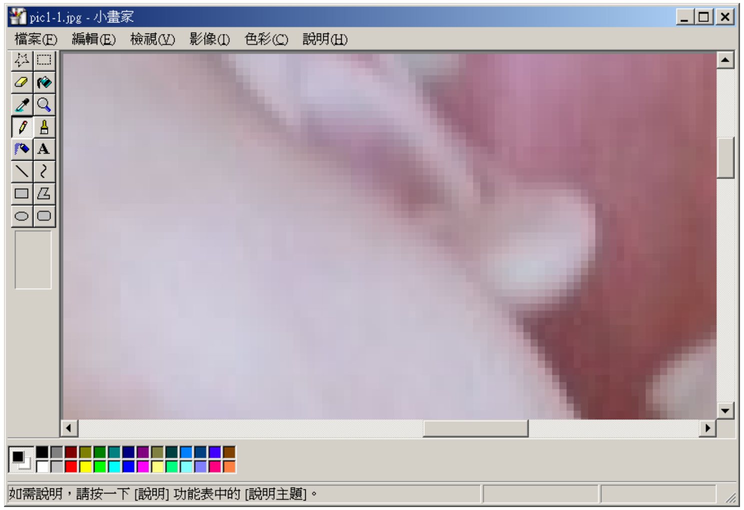


- 許多電腦設備是藉由不同強度的紅(R)、綠 (G)、藍(B) 三原色來表示各種顏色，一個像素佔用3位元組，以表示紅 (R)、綠(G)、藍 (B) 三原色的強度 (0 ~ 255)。



2-8-2　向量圖

向量圖是利用數學貝茲 (Bezier) 曲線來描述圖形的輪廓，然後透過解譯演算法來解譯並轉換成點和線，故向量圖可以依照任意比例放大、縮小、旋轉及傾斜。



點陣圖放大會產生鋸齒

GIF圖形

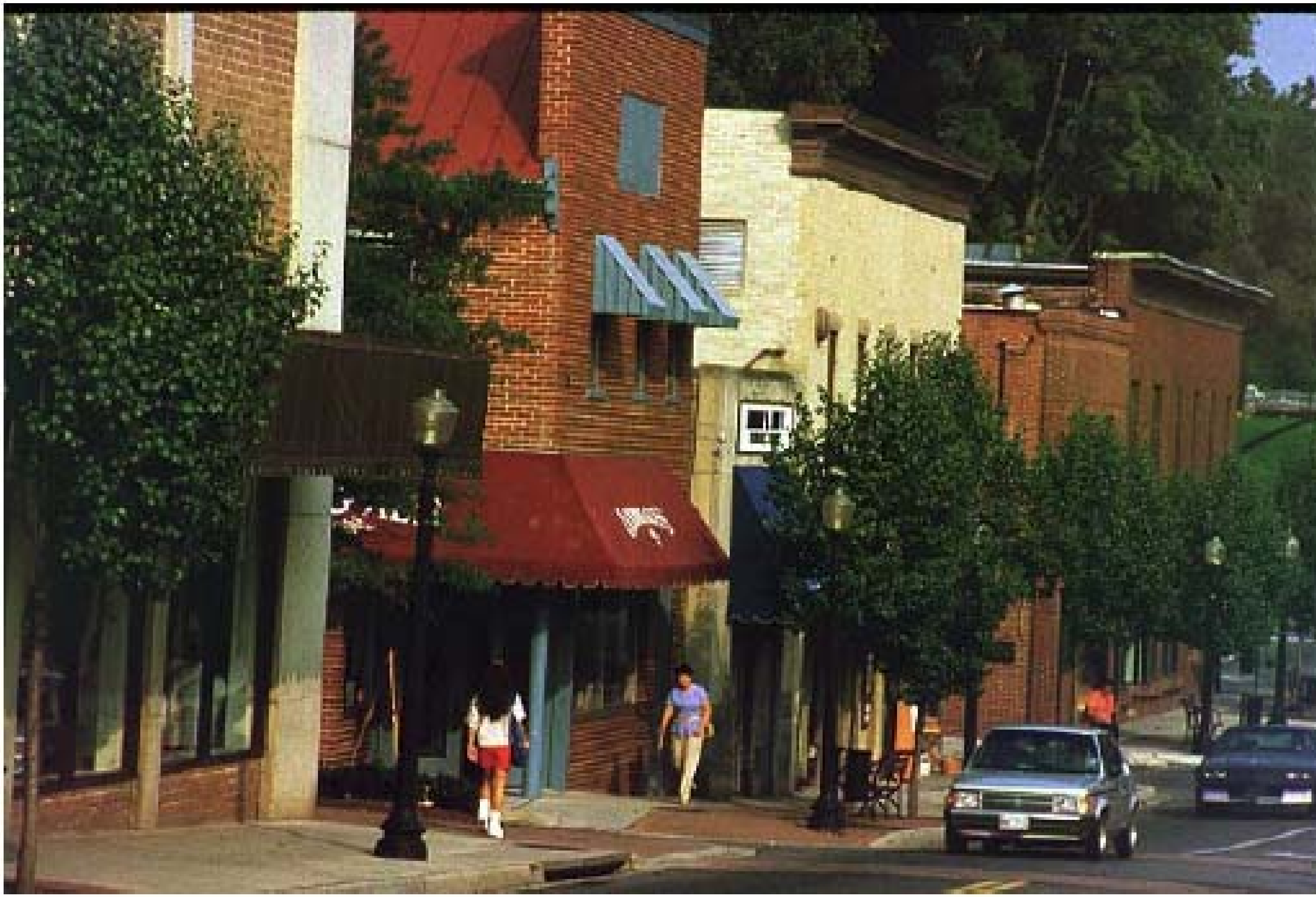
- Gif (graphic information format)是由CompuServe公司於1987年所提出。
- 每個圖素只有256種顏色(只需1位元組)，GIF適合來儲存圖構簡單、顏色較少的圖形。例如，動畫、素描等。

GIF 圖形

JPEG圖形

- JPEG是由國際標準組織(ISO)的Joint Photographic Experts Group所提出，它有非失真壓縮與失真壓縮兩種模式。
- 前者原理在於記錄連續圖素之間的差異，通常這會比記錄每個圖素的值來得節省空間。
- 後者的原理是利用眼睛對於亮度的變化比顏色的變化更為敏感。所以用1位元組來表示每個圖素的亮度。用2位元組來表示每四個連續圖素。

- 每四個連續圖素原本需要12位元組，JPEG只需要6位元組(4位元組+2位元組)。
- JPEG適合儲存顏色較多、較能容忍失真的圖形，目前數位相機多採用JPEG格式儲存照片。



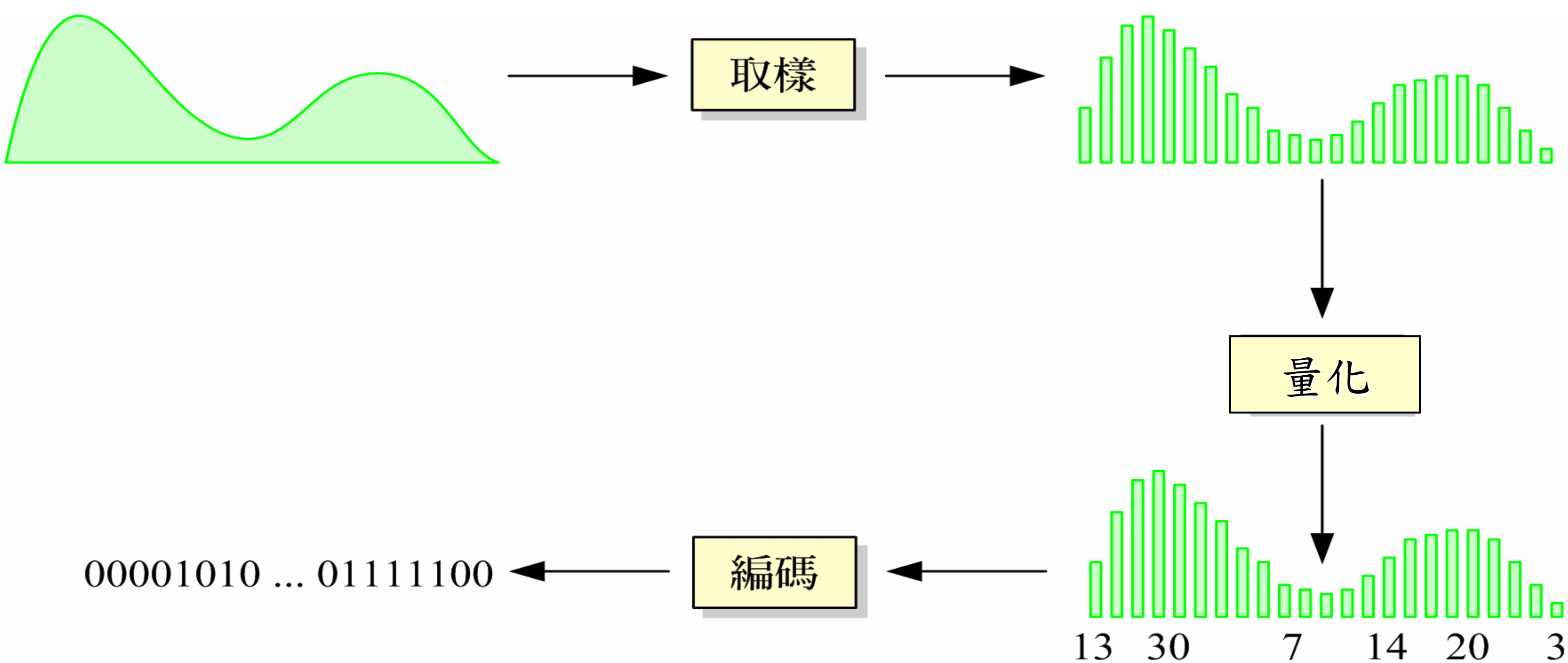
JPEG 圖形

MPEG

- MPEG是由國際標準組織(ISO)的Motion Picture Experts Group所提出。
- 它的原理先以JPEG標準模式處理某畫面，然後再以相對編碼的方式來處理其他畫面，也就是記錄連續畫面的差異，如此會比記錄連續每個畫面來的節省空間。

2-9　聲音表示法

- 由於聲音屬於連續的類比 (analog) 訊號，而電腦只能接受0與1的數位 (digital) 訊號，因此，聲音必須經過如下圖的轉換過程，才能儲存於電腦。



- 取樣：單位時間內測量聲音訊號的值，取樣頻率愈高，愈接近真實聲音。
- 常見的取樣率有11KHZ, 22KHZ, 44KHZ，三種，分別代表一般聲音、錄音機效果與CD唱片效果。

- 量化：每個取樣後的結果，指派一個值給它，例如取樣後的結果為25.2，如果0~100為合法的整數，量化值為25。
- 編碼：將量化值轉為8、16或32位元組儲存，所以需要配MP3等壓縮技術，減少儲存空間。

MP3

- MP3(MPEG-2 audio layer 3)是近年來非常流行的聲音壓縮格式，壓縮比高達1:12。
- 它的原理先分析音樂的頻率範圍，然後過濾耳朵無法聽到的頻率，再進行霍夫曼碼進行編碼。

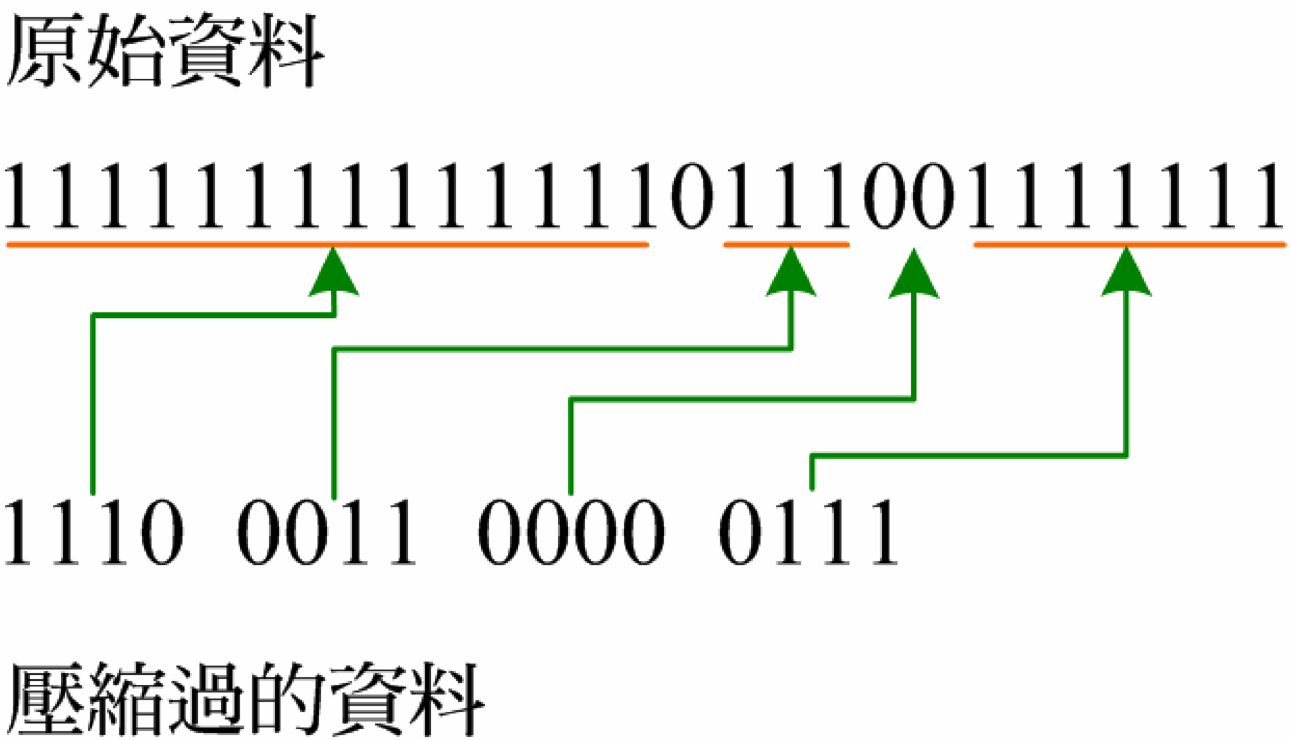
2-10　資料壓縮

- 非失真壓縮：所壓縮過的資料在經過解壓縮後，會和原始資料一樣，不會遺失任何位元。
- 常見的非失真壓縮方法有變動長度編碼 (run length encoding)、霍夫曼碼 (Huffman code)、Lempel-Ziv編碼等。

- 失真壓縮 (lossy compression)：由於人類的眼睛、耳朵無法察覺非常細微的差異，所以對於圖形、照片、影像、聲音等資料，可以使用效率高且空間小的失真壓縮。
- 例如JPEG、GIF可以用來壓縮圖形、照片，MPEG可以用來壓縮影片，MP3可以用來壓縮聲音。

2-10-1　變動長度編碼

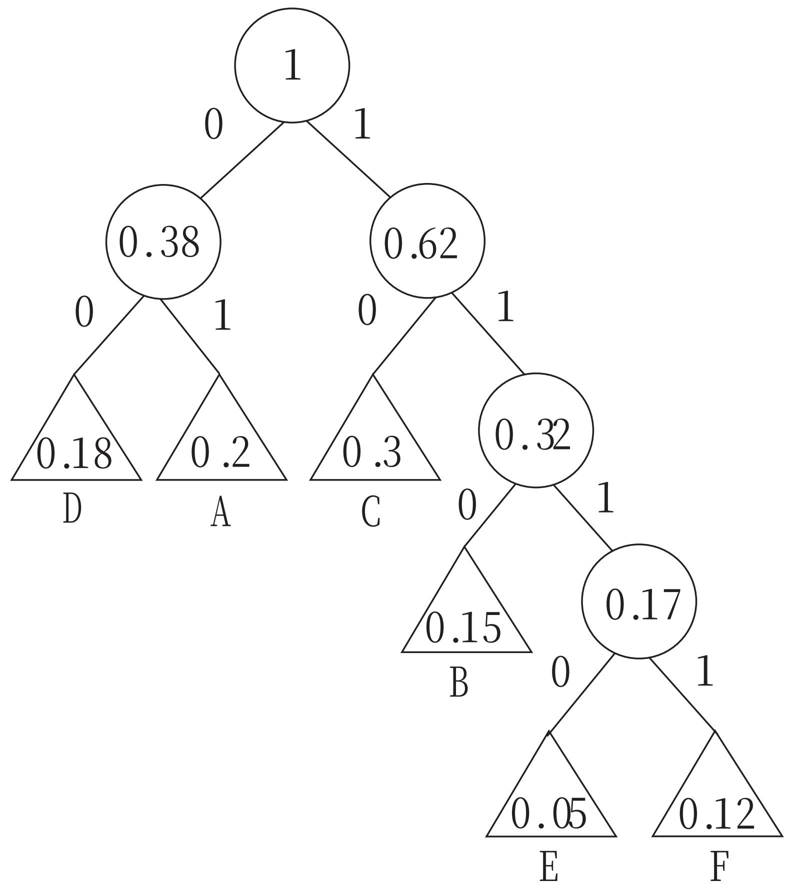
原理是記錄符號出現的次數，例如：



2-10-2　霍夫曼碼

- 不固定長度的編碼方式，符號的編碼長度與出現頻率成反比。
- 編碼步驟如下：
  1. 找出所有符號的出現頻率。
  2. 將頻率最低的兩者相加得出另一個頻率。
  3. 重覆步驟2不斷將頻率最低的兩者相加，直到只剩下一個頻率為止。
  4. 根據合併的關係分別配置0和1，而形成一個編碼樹 。

- 假設編碼系統中有A、B、C、D、E、F等符號，其出現頻率依序為0.2、0.15、0.3、0.18、0.05、0.12，請據此畫出編碼樹並設計一套霍夫曼碼。



平均每個字母編碼所需位元數量為

5*0.05 + 5*0.12 + 3*0.15 + 2*0.18 + 2*0.2 +
2*.3 = 2.435

例如: AABBCCCDDEF

↓

01, 01, 110, 110, 10, 10, 10, 00, 00, 1110, 1111

28 位元 / 11字母 = 2.56

◆ 同一題目(A、B、C、D、E、F等符號，其出現頻率依序為0.2、0.15、0.3、0.18、0.05、0.12)，不同編碼樹設計，便有另一套霍夫曼碼。

A:011

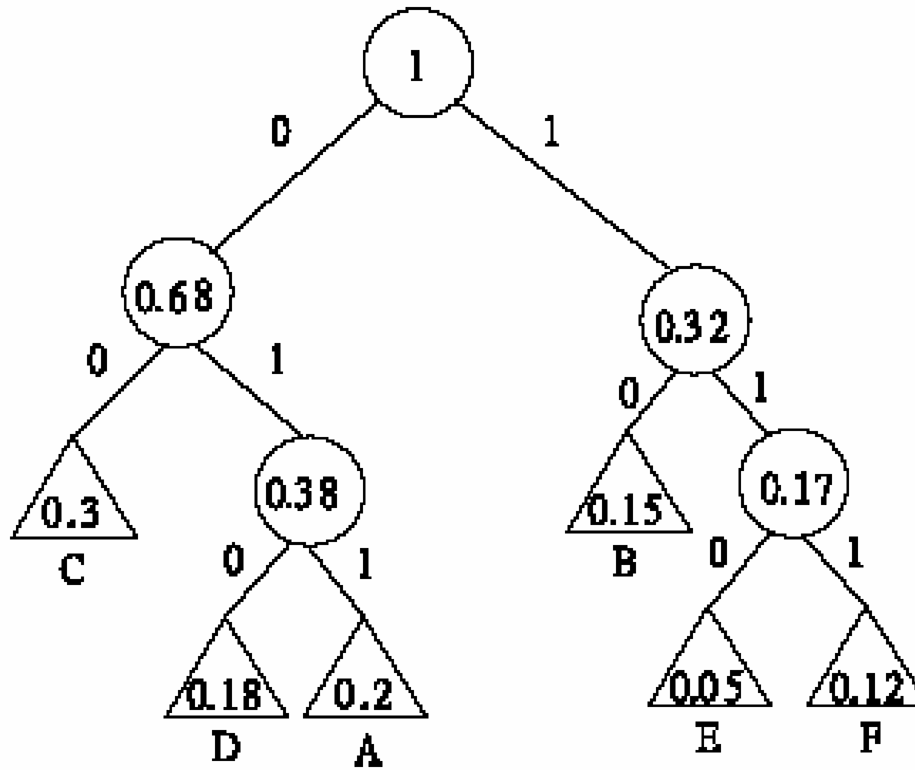
B:10

C:00

D:010

E:110

F:111



平均每個字母編碼所需位元數量為

3*0.05 + 3*0.12 + 2*0.15 + 3*0.18 + 3*0.2 + 2*.3 = 2.55

例如: AABBCCCDDEF

↓

011, 011, 10, 10, 00, 00, 00, 010, 010, 110, 111

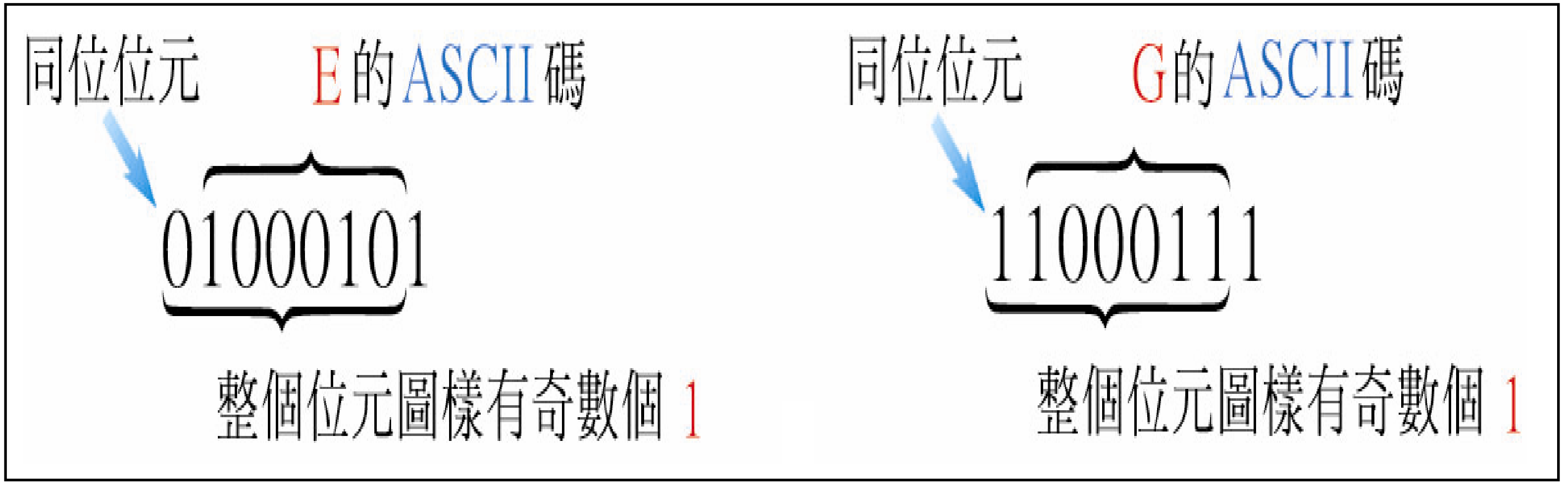
28 位元 / 11字母 = 2.56

2-11　誤差與錯誤檢查

- 固有誤差 (inherent error): 諸如無窮小數(1/3、1/6)，圓週率(π = 3.14159..)等數值，都是因為先天上的限制，導致的誤差。
- 捨棄誤差 (round-off error):由於電腦所表示的數值位元長度有限，所以有些數值的尾數會被捨棄。例如，1.234567，因電腦位元長度有限，只收入1.234，而0.000567就被捨棄。

2-11-1　同位位元檢查

- 在資料位元傳送出去之前，先加上一個同位位元(通常加在最前面)，然後傳送出去，待接收到這些位元圖樣後，就檢查看看是否有奇數個1或偶數個1。
- 又分成奇同位檢查和偶同位檢查。



- 奇(或偶)同位元檢查可檢驗出1個錯誤，例如

01000101 $\xrightarrow{\text{一個位元錯誤}}$ 01001101

正確 錯誤

01000101 $\xrightarrow{\text{二個位元錯誤}}$ 01101101

正確 誤認為正確

2-11-2　循環冗餘碼 (CRC)

- 讓發訊端與收訊端事先協調一個生成多項式，然後發訊端在將資料位元傳送出去之前，先將資料位元除以生成多項式，再將得到的餘數 (即CRC碼) 放在資料位元的後面一起傳送出去。

- 假設資料位元為110010101110，生成多項式為$X^3 + 1$ (1001)，試求取CRC碼及加上CRC碼後的完整訊息：

1. 由於生成多項式$X^3 + 1$ (1001) 的冪次為3，故先在資料位元110010101110的後面加上三個0，得到被除數為110010101110$\boxed{000}$。

2. 以長除法求取110010101110000除以生成多項式$X^3 + 1$ (1001) 的餘數：

```
             101101000101
1001 ⟌ 110010101110000
       1001
        01110
         1001
         01011
          1001
          001001
            1001
            00001100
                1001
                001100
                  1001
                    11   ← 餘數
```

3. CRC碼為餘數11，故完整訊息為 110010101110000+11 =110010101110011。

- 此CRC碼可以檢測3個錯誤與更正1個錯誤。
- 例如：

正確訊息 110010101110011

$\downarrow$ 三個位元錯誤

111000101110010 無法整除，∴為錯誤訊息

111000101110010 ÷ 10001 =11110100011 0 商數

．．．100 餘數