Questa differenziazione non riguarda né la posizione dei bit all'interno del byte (nel qual caso si parla di ordine dei bit) né le posizioni dei caratteri in una stringa. Ha invece importanza nell'interpretazione (o decodifica) delle codifiche multi-byte di stringhe di caratteri (ad esempio: la codifica UTF-16 dello standard unicode).

Classificazione

La differenza tra i due sistemi è data dall'ordine con il quale i byte costituenti il dato da immagazzinare vengono memorizzati o trasmessi:

- *big-endian*: memorizzazione/trasmissione che inizia dal byte più significativo (estremità più grande) per finire col meno significativo, è utilizzata dai processori Motorola;
- *little endian*: memorizzazione/trasmissione che inizia dal byte meno significativo (estremità più piccola) per finire col più significativo, è utilizzata dai processori Intel;

Nota bene: il termine *endian* genera spesso confusione; per ricordare correttamente la differenza si rammenti che *endian* si riferisce alla estremità dalla quale il dato originale comincia a essere elaborato (scritto/letto su/da memoria o trasmesso/ricevuto su/da canale di comunicazione), non a quale estremità finisce per ultima in memoria o su canale di comunicazione.

L'ordine big-endian è stato scelto come ordine standard in molti protocolli utilizzati in Internet, viene perciò anche chiamato *network byte order*. Per contro viene chiamato *host byte order* l'ordine nativo dell'host in uso.

Esempi

Nel caso di una WORD (16 bit), il numero esadecimale 0x0123 verrà immagazzinato come:

```
      Little endian            Big endian
       +----+----+             +----+----+
       |0x23|0x01|             |0x01|0x23|
       +----+----+             +----+----+
byte:    0    1                  0    1
```

Nel caso di una DWORD (32 bit), il numero esadecimale 0x01234567 verrà immagazzinato come:

```
           Little endian             Big endian
       +----+----+----+----+    +----+----+----+----+
       |0x67|0x45|0x23|0x01|    |0x01|0x23|0x45|0x67|
       +----+----+----+----+    +----+----+----+----+
byte:    0    1    2    3        0    1    2    3
```

(Negli esempi il valore 0x01 è il byte più significativo)

Funzioni per la conversione

Le seguenti funzioni possono essere usate per convertire da little a big endian e viceversa (la conversione è perfettamente simmetrica).

Implementazione in C

Conversione di word

```
unsigned short int Endian_Word_Conversion(unsigned short int word) {
   return ((word>>8)&0x00FF) | ((word<<8)&0xFF00)  ;
}
```

Conversione di double word

```
unsigned long int Endian_DWord_Conversion(unsigned long int dword) {
   return ((dword>>24)&0x000000FF) | ((dword>>8)&0x0000FF00) | ((dword<<8)&0x00FF0000) | ((dword<<24)&0xFF000000);
}
```

Implementazione in C#

UInt16

```
public ushort Endian_UInt16_Conversion(ushort value){
   return (ushort)(((value >> 8) & 0x00FF) | ((value << 8) & 0xFF00));
}
```

UInt32

```
public uint Endian_UInt32_Conversion(uint value){
   return ((value >> 24) & 0x000000FF) | ((value >> 8) & 0x0000FF00) | ((value << 8) & 0x00FF0000) | ((value << 24) &
0xFF000000);
 }
```

UInt64

```
public ulong Endian_UInt64_Conversion(ulong value){
   return ((value >> 56) & 0x00000000000000FF) | ((value >> 40) & 0x000000000000FF00) | ((value >> 24) &
0x0000000000FF0000) | ((value >> 8) & 0x00000000FF000000) | ((value << 8) & 0x000000FF00000000)
 | ((value << 24) & 0x0000FF0000000000) | ((value << 40) & 0x00FF000000000000) | ((value << 56) & 0xFF00000000000000);
 }
```

Uso per i formati di data

I termini vengono alle volte usati anche per indicare il formato di data:

- gg/mm/aaaa: la data europea è little-endian
- aaaa/mm/gg: la data big-endian è usata in Giappone e in ISO 8601
- mm/gg/aaaa: la data usata in U.S.A. è middle-endian