Informatica per l’Ingegneria

Corsi M – N

A.A. 2023/2024

Angelo Cardellicchio

03 – Conversioni di base

Outline

- Conversioni di base
  - Conversione B10 ⇒ B2
  - Conversione B2 ⇒ B10
  - Conversione B10 ⇒ B8
  - Conversione B2 ⇒ B8
  - Conversione B10 ⇒ B16
  - Conversione B2 ⇒ B16
  - Utilità dei sistemi ottali ed esadecimali
- Operazioni aritmetiche binarie
- Overflow
- Codici BCD
- Codifica dell'informazione non numerica

Conversioni di base

- I numeri sono concetti astratti rappresentabili in una qualsiasi base di numerazione (a seconda di quanti simboli si possono combinare tra loro).
- È possibile che la stessa quantità sia descritta in modi diversi, cioè usando simboli diversi di un sistema di numerazione basato su una base diversa.
- Ciò comporta la necessità di passare da una base di numerazione ad un’altra, attraverso dei semplici meccanismi matematici.
- L’operazione con cui si passa da una base di numerazione ad un’altra è chiamata ***conversione di base***.

Conversione B10 ⇒ B2 – Parte intera

- Un numero intero in base 10 si può esprimere in una base b dividendolo ripetutamente per b fino ad ottenere un quoziente 0, e recuperando quindi i resti in ordine inverso alla loro determinazione.
  - Di conseguenza, se $b = 2$, dovremo dividere ripetutamente per 2, fermandoci solo quando otterremo un quoziente nullo.
  - I resti delle divisioni effettuate, presi in ordine inverso a quello con cui sono stati calcolati, formano il numero convertito.
- Ad esempio, supponiamo di voler convertire 23 in base 2.

Divisione		Quoziente	Resto
$23/2$	$=$	11	**1**
$11/2$	$=$	5	**1**
$5/2$	$=$	2	**1**
$2/2$	$=$	1	**0**
$1/2$	$=$	0	**1**

$$(23)_{10} = (10111)_2$$

Conversione B10 ⇒ B2 – Parte frazionaria

- Per la parte **frazionaria**, dobbiamo moltiplicare ripetutamente per b, fino ad ottenere il valore 0 per la parte frazionaria, e successivamente recuperando la parte intera nell'ordine di determinazione.
  - *In alternativa, fino a raggiungere il numero massimo di cifre binarie con cui si intende rappresentare il numero frazionario, ottenendo un'approssimazione per difetto del numero decimale.*
- Ad esempio:

Moltiplicazione		Parte frazionaria	Parte intera
$0.25 \cdot 2$	$=$	$.5$	**0**
$0.5 \cdot 2$	$=$	0	**1**

$$(0.25)_{10} = (0.01)_2$$

Moltiplicazione		Parte frazionaria	Parte intera
$.55 \cdot 2$	$=$	$.1$	**1**
$.1 \cdot 2$	$=$	$.2$	**0**
$.2 \cdot 2$	$=$	$.4$	**0**
$.4 \cdot 2$	$=$	$.8$	**0**
$.8 \cdot 2$	$=$	$.6$	**1**

$$(0.55)_{10} \sim (0.10001)_2$$

Conversione B2⇒ B10

- La conversione da base binaria a decimale parte dalla forma polinomiale di un numero.
- Ricordiamo che il numero:

$$N = c_{n-1} \dots c_1 c_0$$

- Equivale a:

$$V(N) = c_0 \cdot 2^0 + c_1 \cdot 2^1 + \dots + c_{n-1} \cdot 2^{n-1}$$

- Quindi, si parte moltiplicando la cifra più significativa per il valore della base, elevatra alla potenza corrispondente alla posizione. Ad esempio:

$$(1011)_2 = 1 \cdot 2^0 + 1 \cdot 2^1 + 0 \cdot 2^2 + 1 \cdot 2^3 =$$

$$= 1 \cdot 1 + 1 \cdot 2 + 0 \cdot 4 + 1 \cdot 8 =$$

$$= 1 + 2 + 8 = 11$$

Conversione B10 ⇒ B8 (e viceversa)

- Un’altra base utilizzata in informatica è la base 8 (***sistema ottale***).
- Per la conversione dalla base 10 alla base 8 (e viceversa) si applicano le stesse regole viste per la conversione da base 10 a 2 (e viceversa).
- Ad esempio:

$$(754)_8 = 4 \cdot 8^0 + 5 \cdot 8^1 + 7 \cdot 8^2 = 4 + 40 + 448 = (492)_{10}$$

Divisione		Quoziente	Resto
$492/8$	$=$	61	$\mathbf{4}$
$61/8$	$=$	7	$\mathbf{5}$
$7/8$	$=$	0	$\mathbf{7}$

Conversione B2 ⇒ B8 (e viceversa)

- Ciascuna cifra ottale può essere rappresentata con esattamente ***tre*** bit.
  - *Di conseguenza, si raggruppano i bit a gruppi di tre da destra verso sinistra per la parte intera, e da sinistra verso destra per la parte frazionaria.*
  - *Si aggiungono se necessario dei bit* 0 *a sinistra (per la parte intera) ed a destra (per la parte frazionaria) del numero.*
  - *Si sostituisce ogni gruppo di tre cifre binarie con la corrispondente cifra ottale.*

$$(1011.1001)_2 \rightarrow (001\ 011\ .\ 100\ 100)_2 \rightarrow (13.44)_8$$

- La conversione inversa avviene sostituendo ogni cifra ottale con il corrispondente gruppo di tre cifre binarie.

$$(23.12)_8 = (010\ 011\ .\ 001\ 010)_2 = (10011.00101)_2$$

Cifra ottale	Numero binario
0	**000**
1	**001**
2	**010**
3	**011**
4	**100**
5	**101**
6	**110**
7	**111**

Conversione B10 $\Rightarrow$ B16 (e viceversa)

- Un'altra base utilizzata in informatica è la base 16 (***sistema esadecimale***).
  - *Le cifre ammesse sono* $\{0,1,2,3,4,5,6,7,8,9, A, B, C, D, E, F\}$.
- Per la conversione dalla base 10 alla base 16 (e viceversa) si applicano le stesse regole viste per la conversione da base 10 a 2 (e viceversa).
- Ad esempio:

$$(A54)_{16} = 4 \cdot 16^0 + 5 \cdot 16^1 + A \cdot 16^2 = 4 + 80 + 2560 = (2644)_{10}$$

Divisione		Quoziente	Resto
$2644/16$	$=$	165	$\mathbf{4}$
$165/16$	$=$	10	$\mathbf{5}$
$10/16$	$=$	0	$\boldsymbol{A}$

Conversione B2 $\Rightarrow$ B16 (e viceversa)

- Ciascuna cifra esadecimale può essere rappresentata con esattamente ***quattro*** bit.
- Valgono le stesse regole viste per la conversione in base 8.

Cifra esadecimale	Numero binario
0	**0000**
1	**0001**
2	**0010**
3	**0011**
4	**0100**
5	**0101**
6	**0110**
7	**0111**

Cifra esadecimale	Numero binario
8	**1000**
9	**1001**
A	**1010**
B	**1011**
C	**1100**
D	**1101**
E	**1110**
F	**1111**

Conversione B2 ⇒ B16 (e viceversa)

$$(11011.01)_2 \rightarrow (0001\,1011\,.\,0100)_2 \rightarrow (1B.4)_{16}$$

$$(1B.4)_{16} \rightarrow (0001\,1011\,.\,0100)_2 = (11011.01)_2$$

Cifra esadecimale	Numero binario
0	**0000**
1	**0001**
2	**0010**
3	**0011**
4	**0100**
5	**0101**
6	**0110**
7	**0111**

Cifra esadecimale	Numero binario
8	**1000**
9	**1001**
A	**1010**
B	**1011**
C	**1100**
D	**1101**
E	**1110**
F	**1111**

Utilità dei sistemi ottale ed esadecimale

- Servono a rappresentare in maniera leggibile e concisa stringhe di bit.
- La numerazione ottale venne introdotta in informatica quando i mainframe più diffusi usavano parole di 24 o 36 bit (divisibili per 3).
  - *È ancora diffusa per rappresentare i permessi sui file nei sistemi Unix: lettura (4), scrittura (2), esecuzione (1). Si sommano i valori dei permessi che si vogliono garantire (e.g. 6: lettura e scrittura).*
- Con la diffusione dei computer a 16, 32 e 64 bit (divisibili per 4) si è imposta la numerazione esadecimale.
  - *Ad esempio, in HTML viene usata la codifica del colore a 24 bit nel formato RGB (#RRGGBB, con RR valore in esadecimale della componente rossa, GG della componente verde, e BB della componente blu, e.g., #FFFF00 è il giallo).*

Operazioni aritmetiche binarie (1)

- Le operazioni aritmetiche binarie funzionano in maniera analoga a quelle decimali.
- L'addizione binaria usa la seguente tabella:

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 0 = 1$$

$$1 + 1 = 0 \; con \; riporto \; di \; 1$$

- Ad esempio:

$$\begin{array}{cccccc} & 1 & 0 & 0 & 1 & 0 & + \\ & 1 & 1 & 1 & 0 & 1 & = \\ \color{red}{1} & \color{red}{0} & 1 & 1 & 1 & 1 & \end{array}$$

Operazioni aritmetiche binarie (2)

- Analogamente possiamo definire la tabella per la sottrazione binaria:

$$0 - 0 = 0$$

$$0 - 1 = 1 \; con \; prestito \; di \; 1$$

$$1 - 0 = 1$$

$$1 - 1 = 0$$

- Ad esempio:

$$\begin{array}{cccccc} \color{red}{1} & \color{red}{10} & 1 & 1 & 1 & - \\ & 1 & 1 & 0 & 1 & = \\ & 1 & 0 & 1 & 0 & \end{array}$$

Numeri negativi in B2

- Per rappresentare i numeri negativi in base due possiamo usare diversi metodi.
- Il più semplice è il metodo del **modulo/segno**.
- Consiste nell'anteporre alla rappresentazione un bit che assume il valore 0 se il numero è positivo, ed 1 altrimenti.
- In pratica:

$$(0\ 0000010)_2 \Leftrightarrow (2)_{10}$$

$$(1\ 0000010)_2 \Leftrightarrow (-2)_{10}$$

- È una rappresentazione intuitiva, ma ha due svantaggi:
  - *non permette la somma tra numeri positivi e negativi;*
  - *lo zero ha una doppia rappresentazione*.

Numeri negativi in base 2 (2)

- Un altro modo per rappresentare i numeri negativi in base 2 è quello di utilizzare il metodo del **complemento ad 1**.
- A differenza del metodo del modulo e segno, questo metodo non scinde l'informazione del segno dal valore stesso.
- Ipotizzando una codifica ad 8 bit, rappresenteremo i valori positivi solo fino a +127, ed i valori negativi solo fino a -127.
- **Il valore negativo è complemento di quello positivo.**
- Ad esempio:

$$(00010111)_2 = (23)_{10}$$
$$(11101000)_2 = (-23)_{10}$$

  - Abbiamo il vantaggio di poter sommare tra loro direttamente numeri positivi e negativi.
  - Il principale svantaggio sta nel fatto che esistono due rappresentazioni valide per lo 0.

Overflow

- Esistono casi in cui il numero di bit a disposizione non è sufficiente per rappresentare un dato.
- In questi casi si parla di ***overflow***.
    - Nel caso il numero di bit non sia sufficiente a rappresentare la parte frazionaria, si parla di ***underflow***.
- Ad esempio: considerando un sistema a 4 bit, sommiamo $9 + 10$.

$$(8)_{10} = (1000)_2$$
$$(9)_{10} = (1001)_2$$

$$\begin{array}{cccccc} & 1 & 0 & 0 & 0 & + \\ & 1 & 0 & 0 & 1 & = \\ 1 & 0 & 0 & 0 & 0 & \end{array}$$

Overflow

- Vediamo cosa accade sottraendo 7 a 4.

$$(4)_{10} = (0100)_2$$
$$(7)_{10} = (0111)_2$$

$$\begin{array}{ccccc} 0 & 1 & 0 & 0 & - \\ 0 & 1 & 1 & 1 & = \end{array} \Rightarrow$$

$$\begin{array}{ccccc} 0 & 0 & \color{red}{1} & \color{red}{10} & - \\ 0 & 1 & 1 & 1 & = \Rightarrow \\ & & 0 & 1 & \end{array}$$

$$\begin{array}{cccccc} \color{red}{(1)} & \color{red}{1} & \color{red}{10} & 1 & 0 & - \\ & 0 & 1 & 1 & 1 & = \\ & 1 & 1 & 0 & 1 & \end{array}$$

Codici BCD

- La codifica ***BCD*** (***Binary Coded Decimal***) viene usata per rappresentare numeri decimali in binario.
- Il numero decimale viene suddiviso nelle cifre decimali che lo compongono, e ciascuna di queste viene convertita in binario a quattro bit.
- Quindi, per convertire il numero 1592 in BCD:

$$1 \rightarrow 0001$$

$$5 \rightarrow 0101$$

$$9 \rightarrow 1001$$

$$2 \rightarrow 0010$$

$$(1592)_{10} = (0001010110010010)_{BCD}$$

Cifra decimale	Numero binario
0	**0000**
1	**0001**
2	**0010**
3	**0011**
4	**0100**
5	**0101**
6	**0110**
7	**0111**
8	**1000**
9	**1001**

Codifica dell’informazione non numerica

- I calcolatori possono intrinsecamente trattare solo informazioni binarie.
- Per trattare numeri relativi e frazionari abbiamo adottato particolari codifiche.
- Più in generale, con i numeri binari è possibile rappresentare insiemi enumerabili di oggetti, stabilendo una corrispondenza biunivoca tra oggetti e numeri (codifica).
- Per rappresentare i caratteri alfabetici esistono alcune codifiche standard.
  - *Ad esempio, il codice* ***ASCII*** *(il più diffuso) codifica su sette cifre binarie tutti i caratteri alfabetici, più alcuni speciali.*
  - *Altro codice molto usato è l’****UNICODE****.*

Esercizi

- ***Somme e sottrazioni in binario***

$$(34)_{10} + (77)_{10} \quad [R.1101111]$$
$$(225)_{10} + (63)_{10} \quad [R.100100000]$$
$$(229)_{10} + (111)_{10} \quad [R.101010100]$$
$$(10)_{10} - (6)_{10} \quad [R.100]$$
$$(39)_{10} - (14)_{10} \quad [R.11001]$$
$$(32)_{10} - (7)_{10} \quad [R.11001]$$
$$(84)_{10} - (37)_{10} \quad [R.101111]$$
$$(18)_{10} - (7)_{10} \quad [R.1011]$$
$$(25)_{10} - (15)_{10} \quad [R.1010]$$

Esercizi

- ***Conversione binario – BCD***

$$
\begin{array}{ll}
1000\ 1001\ 0011\ 0111 & [R.\,8937] \\
100\ 1000\ 0010\ 1001 & [R.\,4829] \\
1\ 0010\ 0011\ 0010\ 1001\ 0001 & [R.\,123291]
\end{array}
$$

- ***Conversione BCD – Binario***

$$
\begin{array}{ll}
4171 & [R.\,100\ 0001\ 0111\ 0001] \\
4261 & [R.\,110\ 0001\ 0101\ 0011] \\
10478 & [R.\,100\ 0010\ 0110\ 0001]
\end{array}
$$

Domande?