Corso di Laurea in Informatica
Architettura degli elaboratori
a.a. 2021-2022

Architettura degli Elaboratori
2021-2022

Sistemi numerici

Prof. Elisabetta Fersini
elisabetta.fersini@unimib.it

Introduzione

- L'aritmetica svolta dai computer differisce dall'aritmetica a noi familiare: i motivi sono molteplici, ma quello principale è sicuramente da attribuire alla diversa modalità nella rappresentazione dei dati e delle informazioni.

- Se per noi il **sistema decimale** può sembrare scontata, non è certo lo stesso per i computer. Infatti i calcolatori lavorano, sfruttando il **sistema binario**. Sistema in cui in cui le uniche cifre ammissibili per rappresentare numeri interi, decimali (ma anche stringhe, caratteri, booleani) sono 0 e 1.

- Una motivazione concreta e intuitiva può essere ricercata in ciò che alimenta i computer: la **corrente elettrica**. Come si potrebbero rappresentare 10 cifre diverse utilizzando la corrente? Magari è possibile, ma sarebbe molto più complesso che distinguere fra corrente sì e corrente no, ovvero 1 e 0.

Introduzione

- Come un computer è in grado di attribuire un significato ad una serie di 0 e di 1?

- Supponiamo che la seguente serie rappresenti un numero.

  1101010

  Come potrebbe un calcolatore capirne il valore per utilizzarlo nelle operazioni aritmetiche? E se invece questi 0 e 1 rappresentassero una stringa?

- Per ovviare a questi problemi sono stati definiti degli standard di codifica, ossia delle regole che vengono utilizzate nella rappresentazione dei dati in formato binario che vanno a coprire le più varie necessità di rappresentazione dell'informazione.

Bit e Configurazioni

- Come sappiamo, con il termine **bit** definiamo l'**unità di misura dell'informazione**. Un bit può assumere solo il valore di 0 o 1.

- Combinando tra loro più bit si ottengono strutture più complesse. In particolare:
  - byte, 8 bit
  - nybble, 4 bit
  - word, 32 bit
  - halfword, 16 bit
  - doubleword, 64 bit

- Se un bit può assumere 2 valori (0 e 1), quanti valori può assumere un byte? E una word?

Bit e Configurazioni

- Quante configurazioni diverse una sequenza di 2 bit può assumere?

- Chiamiamo **a** il <u>primo bit</u> e **b** il <u>secondo bit</u>.

a	b
0	0
0	1
1	0
1	1

- In totale 4 configurazioni. Più generalmente dati *k* **bit**, il numero di **configurazioni** ottenibili è pari a 2^k.

Entità e Rappresentazioni

- Una rappresentazione è un modo per descrivere un'entità

- Sistemi numerici:
  - l'entità numero o valore
  - la rappresentazione

- *Esempio*: Per il valore "*sedici*"
  - la sua rappresentazione nel sistema decimale è 16_{10}
  - la sua rappresentazione nel sistema binario è 10000_2
  - 16_{10} e 10000_2 sono due rappresentazioni differenti della stessa entità

Sistemi numerici (1)

- Il sistema numerico decimale:
  - usa 10 cifre
  - è un **sistema posizionale**: ogni cifra assume un valore diverso a seconda della posizione che occupa all'interno del numero:
    - la posizione delle unità
    - la posizione delle decine
    - la posizione delle centinaia
    - …

- Prendiamo le cifre 1 e 2.
  - Se scrivessimo 12, significherebbe che abbiamo **una decina** e **due unità**.
  - Se invertissimo l'ordine delle due cifre scriveremmo 21, indicando che abbiamo **due decine** e **una unità**.

- Quindi il valore che ha una cifra cambia a seconda della sua posizione nel numero.

Sistemi numerici (2)

- Un esempio di *sistema numerico non posizionale?*

- Il sistema romano:
  - non esistono le posizioni delle unità, decine, centinaia, ...
  - X è sempre 10, indipendentemente dalla sua posizione

  Esempio: il numero IV = 4
  "I" non è una decina, "V" non è un'unità (altrimenti si leggerebbe 15)

Sistemi numerici (3)

- Nei **sistemi numerici posizionali** un valore numerico *N* è caratterizzato dalla seguente **rappresentazione**:

$$N = d_{n-1}d_{n-2}\ldots d_1 d_0 , d_{-1}\ldots d_{-m}$$

$$N = d_{n-1}\cdot r^{n-1} + \ldots + d_0 \cdot r^0 + d_{-1}\cdot r^{-1} + \ldots + d_{-m}\cdot r^{-m}$$

$$N = \sum_{i=-m}^{n-1} d_i \cdot r^i$$

- Dove:
  - *d* rappresenta la singola cifra (*digit*)
  - *r* è la radice o base del sistema
  - *n* è il numero di cifre della parte intera (sinistra della virgola)
  - *m* è il numero di cifre della parte frazionaria (destra della virgola)

Sistema decimale

- Base $r = 10$
- Cifre $d = 0,1,...,9$
- Un valore numerico N si rappresenta come:

$$N = d_{n-1} \cdot 10^{n-1} + \ldots + d_0 \cdot 10^0 + d_{-1} \cdot 10^{-1} + \ldots + d_{-m} \cdot 10^{-m}$$

- Nota bene: tutte le volte che si farà riferimento ad un valore senza specificarne la base, lo si considererà in base 10
- In caso contrario la **base** verrà specificata come **pedice** della cifra di peso più basso: es. 1001011_2

Sistema decimale

- *Esempio* di rappresentazione nel sistema decimale:

$$123{,}45 = 1 \cdot 10^{2} + 2 \cdot 10^{1} + 3 \cdot 10^{0} + 4 \cdot 10^{-1} + 5 \cdot 10^{-2}$$

Sistema binario

- Base $r = 2$
- Cifre $d = 0,1$
- Un valore numerico N si rappresenta come:

$$N = d_{n-1} \cdot 2^{n-1} + \ldots + d_0 \cdot 2^0 + d_{-1} \cdot 2^{-1} + \ldots + d_{-m} \cdot 2^{-m}$$

- Nota bene:
  - Avendo a disposizione n bit posso codificare l'intervallo di valori $[0, 2^n-1]_{10}$
  - Posso quindi definire 2^n codifiche binarie

- Cifra binaria: "*bit*" (*binary digit*)

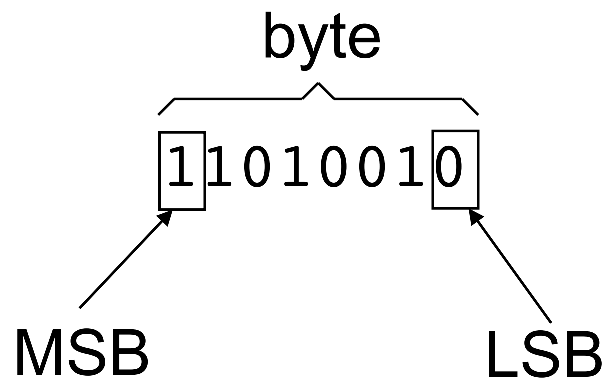
Sistema binario

- *Esempio* di rappresentazione nel sistema binario:

$$101_2 = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 5_{10}$$

Sistema binario

- **Byte**: una sequenza di **otto bit** consecutivi:



- *Most Significant Bit* (MSB) - il bit più a sinistra
- *Least Significant Bit* (LSB) - il bit più a destra

Sistema ottale

- Base $r = 8$
- Cifre $d = 0,1,..., 7$
- Un valore numerico N si rappresenta come:

$$N = d_{n-1} \cdot 8^{n-1} + \ldots + d_0 \cdot 8^0 + d_{-1} \cdot 8^{-1} + \ldots + d_{-m} \cdot 8^{-m}$$

- *Esempio*:

$$127_8 = 1 \cdot 8^2 + 2 \cdot 8^1 + 7 \cdot 8^0 = 87_{10}$$

Sistema esadecimale

- Base $r = 16$
- Cifre d = 0,1,...,9, A, B, C, D, E, F
- Un valore numerico N si rappresenta come:

$$N = d_{n-1} \cdot 16^{n-1} + \ldots + d_0 \cdot 16^0 + d_{-1} \cdot 16^{-1} + \ldots + d_{-m} \cdot 16^{-m}$$

- *Esempio*: $\mathrm{A1}_{16} = \mathrm{A} \cdot 16^1 + 1 \cdot 16^0 = 161_{10}$

- Nota bene:
  - Spesso si usa il pedice $_H$ al posto del pedice $_{16}$ per indicare la base esadecimale <u>oppure</u> 0x davanti al numero (e.g., 0x A1)
  - Il sistema esadecimale viene spesso abbreviato come **esa** oppure **hex**

Sistema esadecimale

$$A_H = 10_{10} = 1010_2$$

$$B_H = 11_{10} = 1011_2$$

$$C_H = 12_{10} = 1100_2$$

$$D_H = 13_{10} = 1101_2$$

$$E_H = 14_{10} = 1110_2$$

$$F_H = 15_{10} = 1111_2$$

- Nota bene: più è grande il valore della base, più è compatta la rappresentazione di uno stesso valore (ossia il numero risultante è composto da meno cifre)

Confronto tra rappresentazioni

$0_{hex} = 0_{dec} = 0_{oct}$	0	0	0	0
$1_{hex} = 1_{dec} = 1_{oct}$	0	0	0	1
$2_{hex} = 2_{dec} = 2_{oct}$	0	0	1	0
$3_{hex} = 3_{dec} = 3_{oct}$	0	0	1	1
$4_{hex} = 4_{dec} = 4_{oct}$	0	1	0	0
$5_{hex} = 5_{dec} = 5_{oct}$	0	1	0	1
$6_{hex} = 6_{dec} = 6_{oct}$	0	1	1	0
$7_{hex} = 7_{dec} = 7_{oct}$	0	1	1	1
$8_{hex} = 8_{dec} = 10_{oct}$	1	0	0	0
$9_{hex} = 9_{dec} = 11_{oct}$	1	0	0	1
$A_{hex} = 10_{dec} = 12_{oct}$	1	0	1	0
$B_{hex} = 11_{dec} = 13_{oct}$	1	0	1	1
$C_{hex} = 12_{dec} = 14_{oct}$	1	1	0	0
$D_{hex} = 13_{dec} = 15_{oct}$	1	1	0	1
$E_{hex} = 14_{dec} = 16_{oct}$	1	1	1	0
$F_{hex} = 15_{dec} = 17_{oct}$	1	1	1	1

Conversione tra sistemi numerici

- Consideriamo per il momento solo valori interi (o la parte intera di un valore frazionario)
- La conversione da qualsiasi base r a base 10 avviene come segue:

$$N_r = d_{n-1} \cdot r^{n-1} + \ldots + d_0 \cdot r^0 = M_{10}$$

- *Esempi*:

$$1010_2 = (1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0)_{10} = 10_{10}$$

$$26_8 = (2 \cdot 8^1 + 6 \cdot 8^0)_{10} = 22_{10}$$

$$431_5 = (4 \cdot 5^2 + 3 \cdot 5^1 + 1 \cdot 5^0)_{10} = 116_{10}$$

Conversione tra sistemi numerici

- Consideriamo per il momento solo valori interi (o la parte intera di un valore frazionario)

- La conversione da base 10 a qualsiasi base r avviene come segue:

  ① Dividiamo il valore numerico *N* per la base *r* fino a quando l'ultimo quoziente è minore della base stessa (*r*)

  ② Prendiamo l'ultimo quoziente e tutti i resti delle divisioni, e procedendo dall'ultimo resto al primo, li scriviamo da sinistra verso destra

Conversione tra sistemi numerici

- *Esempio*: Convertire il numero 12 da base 10 a base 2
- 12 : 2 = 6 con resto=0
- 6 : 2 = 3 con resto =0
- 3 : 2 = 1 con resto =1
- 1 : 2 = 0 con resto =1
- quindi:

$$12_{10}=1100_2$$

Conversione tra sistemi numerici

- *Esempio*: convertire il numero 120 da Base 10 a Base 8
- 120 : 8 = 15 con resto = 0
- 15 : 8 = 1 con resto = 7
- 1 : 8 = 0 con resto 1

quindi:

$$120_{10}=170_8$$

Conversione tra sistemi numerici

- *Esempio*: convertire il numero 1253 da base 10 a base 16
- 1253 : 16 = 78 con resto = 5
- 78 : 16 = 4 con resto = 14 = E
- 4 : 16 = 0 con resto 4

quindi:

$$1253_{10} = 4E5_{16}$$

Conversione tra sistemi numerici

- Alcune osservazioni:
  - La conversione dalla base 2 alla base 16 e/o 8, e viceversa, è più **semplice** e veloce di quella da decimale ad altre basi.
    - Infatti basta considerare che per rappresentare le sedici cifre diverse del codice esadecimale occorrono 4 bit (24 = 16) mentre per rappresentare le otto cifre diverse del codice ottale occorrono 3 bit (23 = 8).
    - Ne risulta che per convertire un numero binario in esadecimale o in ottale, è sufficiente raggruppare le cifre binarie rispettivamente in gruppi di quattro o tre cifre (bit) a partire da quelle "meno significative": si ricava immediatamente il numero grazie alla sostituzione dei bit così ricavati con la cifra esadecimale o ottale corrispondente.

Conversione tra sistemi numerici

- Esempio: conversione da binario in esadecimale

$$111\ 1111\ 0001\ 1010 \rightarrow 7\ 15\ 1\ 10 \rightarrow 7F1A_{16}$$

- dove:
  - 1010 (conversione da binario a decimale) = 10 in esadecimale corrisponde ad A
  - 0001 (conversione da binario a decimale) = 1 in esadecimale corrisponde ad 1
  - 1111 (conversione da binario a decimale) = 15 in esadecimale corrisponde ad F
  - 111 (conversione da binario a decimale) = 7 in esadecimale corrisponde ad 7

Conversione tra sistemi numerici

- Consideriamo per il momento solo valori interi (o la parte intera di un valore frazionario)

- La conversione da qualsiasi base p a qualsiasi base q avviene come segue:

  ① Convertire il numero da base p a base 10
  ② Convertire il risultato da base 10 a base q

- Osservazione: nei casi in cui sia p sia q siano potenze di 2, conviene passare non dalla base 10, ma dalla base 2. La conversione tra una base potenza di 2 e la base 2 è molto più veloce.

Conversione tra sistemi numerici

- *Esempio*: Convertire il numero $AB2_{16}$ in binario.
- $A_{16}=10_{10}=1010_2$
- $B_{16}=11_{10}=1011_2$
- $2_{16}=2_{10}=0010_2$
- quindi

$$AB2_{16} = 101010110010_2$$

Conversione tra sistemi numerici

- *Esempio*: Convertire il numero $(516)_8$ in Binario.
- $5_8 = 5_{10} = 101_2$
- $1_8 = 1_{10} = 001_2$
- $6_8 = 6_{10} = 110_2$
- quindi

$$516_8 = 101001110_2$$

Rappresentazione dei valori

- La rappresentabilità dei valori è legata al numero di **cifre disponibili**.

- Nei sistemi di elaborazione, come in generale in tutte le applicazioni pratiche, **il numero di cifre** impiegate nella rappresentazione di valori numerici **è limitato**.

- Si ha **overflow** quando si è nell'impossibilità di rappresentare il risultato di una operazione (e.g.,una somma o una sottrazione) con il numero di cifre a disposizione.

Rappresentazione dei valori

- Come anticipato, nel **sistema binario** abbiamo:

n numero di bit	2^n numero di valori	2^n-1 max valore rappresentabile
1	2	1
2	4	3
3	8	7
4	16	15
5	32	31
6	64	63
....	....	....

Rappresentazione dei valori

- La rappresentazione dei valori nel sistema binario è quindi diversa rispetto al sistema decimale:

  - Kilo nella terminologia informatica, non indica 1000, bensì:

$$2^{10} = 1024$$

  - Il prefisso Mega indica la quantità:

$$2^{20} = 1048576$$

  - Il prefisso Giga indica:

$$2^{30} = 1073741824$$