

# Sistemi di numerazione

*Ci sono 10 tipi di persone*

*Quelli che capiscono il binario e quelli che non lo capiscono*

# Sistemi di rappresentazione dei numeri

- x Differenti tipi di rappresentazione:
  - x sistemi non posizionali
    - x numerazione romana: *MDC LXI*
  - x sistemi posizionali
    - x numerazione araba:  
 $1661 = 1 * 1000 + 6 * 100 + 6 * 10 + 1$
    - x basi differenti

# Sistemi posizionali: caso generale

Un numero  $N$  di  $n$  cifre in base  $x$  è rappresentato da una sequenza di caratteri (cifre):

$$b_n b_{n-1} b_{n-2} b_{n-3} \dots b_2 b_1 b_0$$

$$\text{con } b_i \in \{0, 1, 2, \dots, x-1\}$$

Il valore di  $N$  è dato da:

$$N = b_n x^n + b_{n-1} x^{n-1} + \dots + b_2 x^2 + b_1 x + b_0$$

# Sistemi posizionali: rappresentazione decimale

x Il sistema di numerazione decimale è un sistema *posizionale* a base 10

- $$N = d_n 10^n + d_{n-1} 10^{n-1} + \dots + d_2 10^2 + d_1 10 + d_0$$

- $d \equiv \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

# Base di un sistema posizionale

- x In informatica risulta conveniente la rappresentazione dei numeri in basi differenti da quella decimale:
  - x *rappresentazione ottale: base 8*
  - x rappresentazione binaria: base 2
  - x rappresentazione esadecimale: base 16

# Linguaggio binario

- x Calcolatori elettronici: transistor, flip-flop, celle di memoria
- x elementi a due stati
- x Alfabeto binario
  - x simboli convenzionali: 1 e 0 (bit)
  - x parole: 100, 011010, 10, 1000101

# Rappresentazione dei numeri

- x Occorre rappresentare i numeri in forma binaria
  - x tipo di numeri
  - x campo di valori
  - x precisione

# Sistemi posizionali: rappresentazione binaria

- x Il sistema di numerazione binario è un sistema *posizionale* a base 2
- x La notazione  $b_{n-1}b_{n-2}\dots b_2b_1b_0$  rappresenta il numero  $N$ :

$$N = b_n 2^n + b_{n-1} 2^{n-1} + \dots + b_2 2^2 + b_1 2 + b_0$$

$$b \equiv \{0, 1\}$$

# Rappresentazione binaria: esempio

- x Il numero binario 101110 rappresenta il seguente numero:
  - $N=1x2^5+0x2^4+1x2^3+1x2^2+1x2+0$
- x Si può vedere come somma di potenze di 2:
  - $N=32+8+4+2$

# Somma di potenze di 2

- x Tutti i numeri sono rappresentabili come somma di potenze di 2:

Potenza di 2	32	16	8	4	2	1
Cifra binaria	1	0	1	1	1	0
Prodotto	32	0	8	4	2	0

# Conversione dal sistema binario al sistema decimale

- 1) Posizionarsi sulla sinistra del numero da convertire (parte piú significativa)
- 2) Inizializzare  $N$  a  $0$
- 3) Leggere la cifra binaria che segue procedendo verso destra
- 4) Moltiplicare  $N$  per  $2$  e sommarvi la cifra letta
- 5) Se ci sono ancora cifre tornare al punto 3)
- 6) Fine della procedura,  $N$  contiene il risultato

# Conversione generica di un numero espresso in base $b$

- 1) Posizionarsi sulla sinistra del numero da convertire (parte piú significativa)
- 2) Inizializzare  $N$  a  $0$
- 3) Leggere la cifra che segue procedendo verso destra
- 4) Moltiplicare  $N$  per  $b$  e sommarvi la cifra letta
- 5) Se ci sono ancora cifre tornare al punto 3)
- 6) Fine della procedura,  $N$  contiene il risultato

# Conversione in base $b$

- Problema (generico): dato un numero  $N$  rappresentarlo in base  $b$ .
- 1) calcolare valore ( $v$ ) e resto ( $r$ ) della divisione  $N/b$
  - 2)  $r$  rappresenta la cifra meno significativa
  - 3) se  $v \neq 0$  sostituire  $N$  con  $v$  e ripetere da 1)

# Rappresentazione binaria: proprietà

- x Una parola binaria di  $n$  cifre (bit) permette di rappresentare tutti i numeri interi  $N$  tali che:
  - $0 \leq N \leq 2^n - 1$
- x Dato un numero  $N$  la sua rappresentazione binaria richiederà un numero  $n$  di bit tale che:
  - $n > \log_2 N$

# Sistemi posizionali: rappresentazione esadecimale

- x Il sistema di numerazione esadecimale è un sistema *posizionale* a base 16

$$N = b_n 16^n + b_{n-1} 16^{n-1} + \dots + b_2 16^2 + b_1 16 + b_0$$

$$b \equiv \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$$

# Conversioni tra sistema esadecimale e sistema binario

- x Una singola cifra del sistema esadecimale può codificare i numeri tra 0 e 15
- x Per rappresentare i numeri tra 0 e 15 nel sistema binario occorrono 4 bit
- x La conversione da un sistema di rappresentazione all'altro risulta facilitata

# Conversioni tra sistema esadecimale e sistema binario (2)

- x Conversione da binario ad esadecimale: si prendono i bit 4 a 4 (partendo dai meno significativi) e si sostituiscono con la corrispondente cifra esadecimale:

$$10111001_2 = 0xB9_{16}$$

# Conversioni tra sistema esadecimale e sistema binario (3)

- x Conversione da esadecimale a binario: si sostituisce alle cifre esadecimali la corrispondente sequenza di 4 bit in binario

$$0xFA2_{16} = 1111\ 1010\ 0010_2$$

# Tabella di conversione esadecimale-binario

<b>Binario</b> <i>b=2</i>	<b>Esadecimale</b> <i>b=16</i>	<b>Decimale</b> <i>b=10</i>
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

# Aritmetica in base $b$

- x L'aritmetica in una generica base segue le stesse regole dell'aritmetica a base 10

Somma:

$$\begin{array}{r} 10010110 \\ 11011010 \\ \hline 101110000 \end{array} \qquad \begin{array}{r} 150_{10} \\ + 218_{10} \\ \hline 368_{10} \end{array}$$

Sottrazione:

$$\begin{array}{r} 101000 \\ - 011001 \\ \hline 01111 \end{array} \qquad \begin{array}{r} 40_{10} \\ - 25_{10} \\ \hline 15_{10} \end{array}$$

# Aritmetica in base $b$ (2)

- x L'aritmetica in una generica base segue le stesse regole dell'aritmetica a base 10

Moltiplicazione:

11010	$26_{10}$
x 101	x $5_{10}$
<hr/>	
11010	
00000	
11010	
<hr/>	
10000010	$130_{10}$

# Numeri con segno

- x Devono:
  - x mantenere la stessa rappresentazione per i numeri positivi
  - x utilizzare rappresentazioni alternative per i numeri negativi
    - x semplicità di utilizzo
    - x semplicità di implementazione

## Numeri con segno (2)

- x La rappresentazione piú intuitiva è quella ottenuta tramite il bit di segno
  - x 0 per i numeri positivi
  - x 1 per i numeri negativi
- x Avere un bit con significato differente porta a complicazioni:
  - x software
  - x hardware

# Complemento a 1

- x si complementa bit a bit il numero positivo di partenza
- x doppia rappresentazione dello 0

$$x = b_N b_{N-1} \dots b_2 b_1 b_0$$

$$-x = \bar{b}_N \bar{b}_{N-1} \dots \bar{b}_2 \bar{b}_1 \bar{b}_0$$

# Complemento a 2

- x Analogia con altri sistemi (angoli)
- x Dati  $n$  bit, il numero  $X$  si rappresenta come:
  - x  $X$  quando  $0 \leq X < 2^n/2$
  - x  $2^n - |X|$  quando  $-2^n/2 \leq X < 0$

# Complemento a 2 (2)

- x Stessa rappresentazione per numeri positivi
- x Si aggiunge 1 al positivo complementato trascurando il riporto
- x Esempio: negativo di  $27_{10} = 00011011_2$

$$\begin{array}{r} \text{Complemento ad 1} = 11100100 \\ \phantom{\text{Complemento ad 1} = } + 1 \end{array}$$

---

$$-27_{10} = 11100101$$

# Complemento a 2: proprietà

- x Singola rappresentazione dello 0
- x Il bit *piú significativo* è 1 per numeri negativi  
0 per numeri positivi
- x Dati  $n$  bit posso rappresentare tutti i numeri nell'intervallo  $[-2^{n-1}, 2^{n-1}-1]$
- x 8 bit:  $[-128, 127]$  oppure  $[0, 255]$
- x 16 bit:  $[-32768, 32767]$  oppure  $[0, 65536]$

# Complemento a 2: proprietà (2)

- x Non è esclusivamente un codice posizionale
- x Il peso dei bit dipende dal numero di cifre della rappresentazione

<b>Indice posizione</b>	7	6	5	4	3	2	1	0
<b>Peso bit</b>	-128	64	32	16	8	4	2	1

- x Si deve sempre specificare il numero di cifre

# Complemento a 2: conversione

- x la conversione è relativamente semplice:
  - x si lasciano inalterati gli 0 a partire dal bit meno significativo
  - x si lascia inalterato il primo 1
  - x si complementa tutto il resto

# Complemento a 2: aritmetica

- x Somma e sottrazione nella rappresentazione in complemento a 2 seguono le stesse regole viste in precedenza
- x Supponiamo di lavorare con 4 bit ( $[-8, 7]$ )

$$\begin{array}{r} 0010 \\ + 0100 \\ \hline [0]0110 \end{array} \qquad \begin{array}{r} 2_{10} \\ + 4_{10} \\ \hline 6_{10} \end{array}$$

$$\begin{array}{r} 0101 \\ + 0100 \\ \hline [0]1001 \end{array} \qquad \begin{array}{r} 5_{10} \\ + 4_{10} \\ \hline \text{(overflow)} -7_{10} \end{array}$$

# Complemento a 2: aritmetica (2)

- x Somma nel caso di numeri negativi

1110	$-2_{10}$
+ 1100	+ $-4_{10}$
<hr/>	
[1]1110	$-6_{10}$
1011	$-5_{10}$
+ 1100	+ $-4_{10}$
<hr/>	
[1]0111	(overflow) $-9_{10}$

# Complemento a 2: aritmetica (3)

- x Somma nel caso di numeri con segno opposto

0010	$2_{10}$
+ 1100	+ $-4_{10}$
<hr/>	
[1]1110	$-2_{10}$
1011	$-5_{10}$
+ 0111	+ $7_{10}$
<hr/>	
[1]0010	$2_{10}$

# Complemento a 2: aritmetica (4)

- x La somma di due numeri dello stesso segno può causare *overflow*:
  - x somma di due numeri dello stesso segno con segno opposto
  - x riporto e cifra piú significativa sono differenti

# Rappresentazione binaria: numeri a virgola fissa

- × È possibile rappresentare numeri a virgola fissa estendendo le regole viste fino ad ora

$$x = \sum_{i=0}^{N-1} b_i \times 2^i = b_0 + b_1 \times 2 + b_2 \times 2^2 + \dots + b_{N-1} \times 2^{N-1}$$

$$0 \leq x \leq 2^N - 1$$

$$x = \sum_{i=1}^{N-1} b_i \times 2^{-i} = b_1 \times \frac{1}{2} + b_2 \times \frac{1}{2^2} + \dots + b_{N-1} \times \frac{1}{2^{N-1}}$$

$$0 \leq x \leq 1 - 2^{-N}$$

**ATTENZIONE:** numeri aventi un numero finito di cifre in decimale potrebbero averne infinite in binario!

# Lo standard IEEE 754

- x Rappresentazione numeri a virgola mobile a 32 e 64 bit
- x Idea di base:  $N = (-1)^s 2^{E-127} (1.M)$
- x  $M$  sempre compreso tra 1 e  $\frac{1}{2}$
- x  $E$  ed  $M$ , assumono valori specifici (0,255) per gestione eccezioni (NaN,  $\pm\infty$ , underflow)

