Rappresentazione digitale dell'informazione

Indice

L'aritmetica dei calcolatori

L'aritmetica usata dai calcolatori è diversa da quella comunemente utilizzata dalle persone

La precisione con cui i numeri possono essere espressi è finita e predeterminata poiché questi devono essere memorizzati entro un limitato spazio di memoria

$\sqrt{2} =$ | 1 | . | 4 | 1 | 4 | 2 | 1 | 3 5 6 2

La rappresentazione è normalmente ottenuta utilizzando il sistema binario poiché più adatto a essere maneggiato dal calcolatore

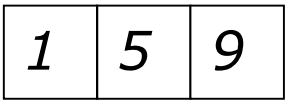
$$124 \Rightarrow 01111100$$

Numeri a precisione finita (1)

I **numeri a precisione finita** sono quelli rappresentati con un numero finito di cifre.

Fissate le caratteristiche del numero è determinato anche l'insieme di valori rappresentabili

Esempio: *Numeri non rappresentabili con 3 cifre senza virgola e senza segno*
Numeri superiori a 999
Numeri negativi
Frazioni



Le operazioni con i numeri a precisione finita causano errori ogniqualvolta il loro risultato non appartiene all'insieme dei valori rappresentabili:

- **Underflow**: si verifica quando il risultato dell'operazione è minore del più piccolo valore rappresentabile
- **Overflow**: si verifica quando il risultato dell'operazione è maggiore del più grande valore rappresentabile
- **Non appartenenza all'insieme**: si verifica quando il risultato dell'operazione, pur non essendo troppo grande o troppo piccolo, non appartiene all'insieme dei valori rappresentabili

Esempio: *Numeri a precisione finita con 3 cifre senza virgola e senza segno*

600+600 = 1200 ⇒ Overflow
300-600 = -300 ⇒ Underflow
007/002 = 3.5 ⇒ Non appartenenza all'insieme

Numeri a precisione finita (2)

Si noti che, a differenza dei numeri interi, i numeri a precisione finita non rispettano la chiusura rispetto alle operazioni di somma, sottrazione e prodotto.

Esempio: *Risultati non rappresentabili con 3 cifre senza virgola e senza segno*

Operazioni	Interi	Precisione finita
Somma: 600+600	1200	Overflow
Sottrazione:300-600	-300	Underflow
Prodotto: 050×050	2500	Overflow

Anche l'algebra dei numeri a precisione finita è diversa da quella convenzionale. Poiché alcune delle proprietà non vengono rispettate:

Proprietà associativa: $a + (b - c) = (a + b) - c$

Proprietà distributiva: $a \times (b - c) = a \times b - a \times c$

Non sono rispettate poiché in base all'ordine con cui vengono eseguite le operazioni si può verificare o meno un errore

Esempio: *Operazioni con numeri a precisione finita di 3 cifre senza virgola e senza segno*

$$400 + (300 - 500) = (400 + 300) - 500$$

$$400 + (-200) = 700 - 500$$

Underflow

$$50 \times (50 - 40) = 50 \times 50 - 50 \times 40$$

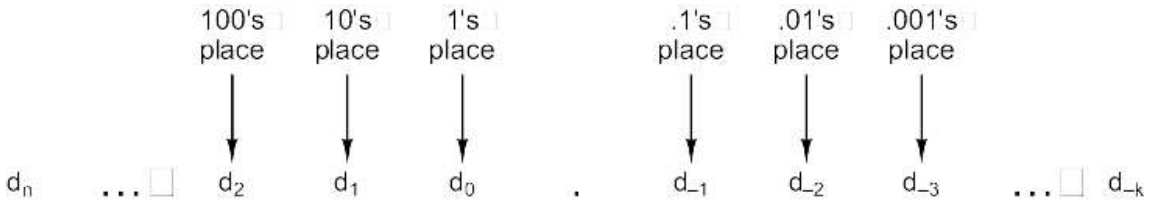
$$50 \times 10 = 2500 - 2000$$

Overflow *Overflow*

ATTENZIONE: non confondere i numeri negativi con le operazioni di sottrazione

Notazione posizionale (1)

I sistemi di numerazione posizionale associano alle cifre un diverso valore in base alla posizione che occupano nella stringa che compone il numero.



$$Valore = \sum_{i=-k}^{n} d_i \times 10^i$$

Esempio: *Rappresentazione posizionale di 5798.46*

$$5 \times 10^3 + 7 \times 10^2 + 9 \times 10^1 + 8 \times 10^0 + 4 \times 10^{-1} + 6 \times 10^{-2}$$
$$= 5000 + 700 + 90 + 8 + 0.4 + 0.06$$

- Un sistema di numerazione posizionale è definito dalla **base (o radice)** utilizzata per la rappresentazione.

- Un sistema posizionale in base *b* richiede *b* simboli per rappresentare i diversi valori tra 0 e (*b*-1)

Sistema decimale (*b*=10) 0 1 2 3 4 5 6 7 8 9

Sistema binario (*b*=2) 0 1: ogni cifra, detta *bit* (**B**inary dig**IT**), può essere rappresentata direttamente tramite un livello elettrico di tensione

Sistema ottale (*b*=8) 0 1 2 3 4 5 6 7

Sistema esadecimale (*b*=16) 0 1 2 3 4 5 6 7 8 9 A B C D E F: è utilizzato nel linguaggio dell'assemblatore poiché è molto compatto e semplice da scandire. Inoltre ben si presta alla traduzione in valori binari poiché ogni cifra corrisponde esattamente a 4 cifre binarie.

Notazione posizionale (2)

A ogni numero corrisponderanno rappresentazioni diverse in basi diverse

Binary 1 1 1 1 1 0 1 0 0 0 1

$1\ 2^{10} + 1\ 2^9 + 1\ 2^8 + 1\ 2^7 + 1\ 2^6 + 0\ 2^5 + 1\ 2^4 + 0\ 2^3 + 0\ 2^2 + 0\ 2^1 + 1\ 2^0$

1024 + 512 + 256 + 128 + 64 + 0 + 16 + 0 + 0 + 0 + 1

Octal 3 7 2 1

$3\ 8^3 + 7\ 8^2 + 2\ 8^1 + 1\ 8^0$

1536 + 448 + 16 + 1

Decimal 2 0 0 1

$2\ 10^3 + 0\ 10^2 + 0\ 10^1 + 1\ 10^0$

2000 + 0 + 0 + 1

Hexadecimal 7 D 1

$7\ 16^2 + 13\ 16^1 + 1\ 16^0$

1792 + 208 + 1

Dato che l'insieme dei simboli utilizzati dalle varie basi non è disgiunto è necessario aggiungere al numero un pedice che indichi la radice utilizzata.

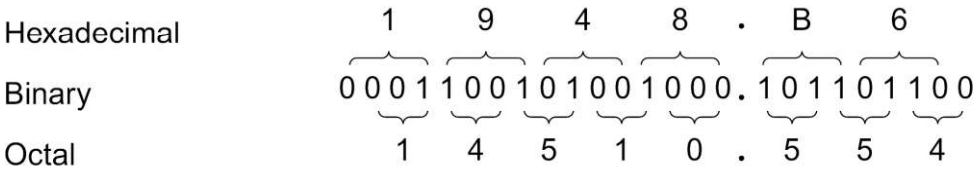
$$11111010001_2 = 3721_8 = 2001_{10} = 7D1_{16}$$

Conversione tra basi (1)

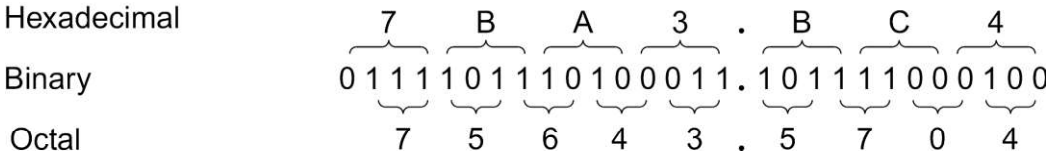
Binario ⇔ Ottale: dato che una cifra del sistema ottale è rappresentabile esattamente con tre cifre del sistema binario, la conversione può essere ottenuta raggruppando le cifre binarie a 3 a 3 a partire dalla virgola binaria. L'operazione contraria è ugualmente semplice, ogni cifra ottale viene convertita in esattamente tre cifre binarie.

Esadecimale ⇔ binario: il processo di conversione è equivalente a quello binario-ottale ma le cifre binarie devono essere raggruppate a 4 a 4.

Example 1



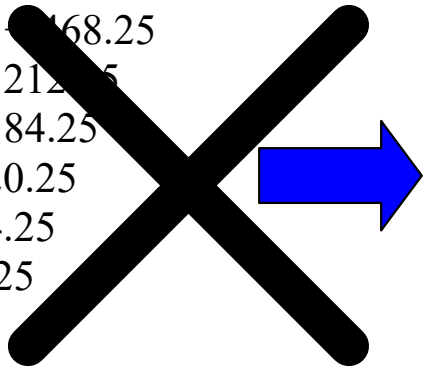
Example 2



Conversione tra basi (2)

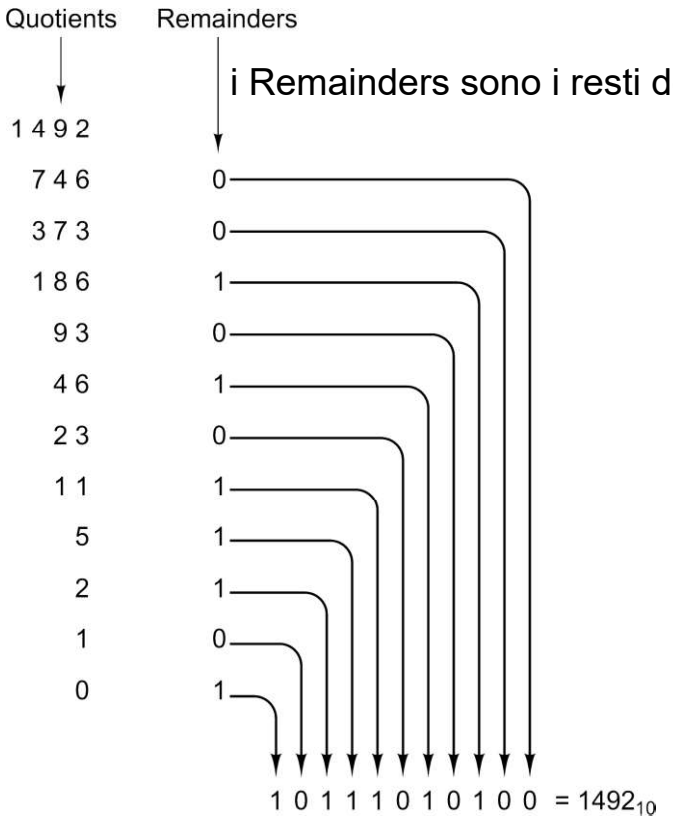
~~**Decimale ⇒ Binario (Metodo generale):** si procede sottraendo al numero da decomporre la più grande potenza di 2 minore del numero da decomporre. Il processo viene applicato ricorsivamente al resto della sottrazione. Il risultato binario si ottiene ponendo a uno le cifre corrispondenti alle potenze che sono state utilizzate nella decomposizione.~~

$1492.25 = 2^{10}$ + 468.25
$468.25 = 2^{8}$ + 212.25
$212.25 = 2^{7} + 84.25$
$84.25 = 2^{6} + 20.25$
$20.25 = 2^{4} + 4.25$
$4.25 = 2^{2} + 0.25$
$0.25 = 2^{-2}$



10111010100.01

Decimale ⇒ Binario (Solo per interi): la codifica viene ottenuta direttamente procedendo in modo ricorsivo. Si divide il numero per 2: il resto (0 o 1) farà parte del risultato mentre il quoziente verrà utilizzato come input al seguente passo di ricorsione.



Conversione tra basi (3)

Binario ⇒ Decimale (Primo metodo): si procede traducendo le singole cifre binarie alle corrispondenti potenze di due in base decimale e sommando i risultati parziali:

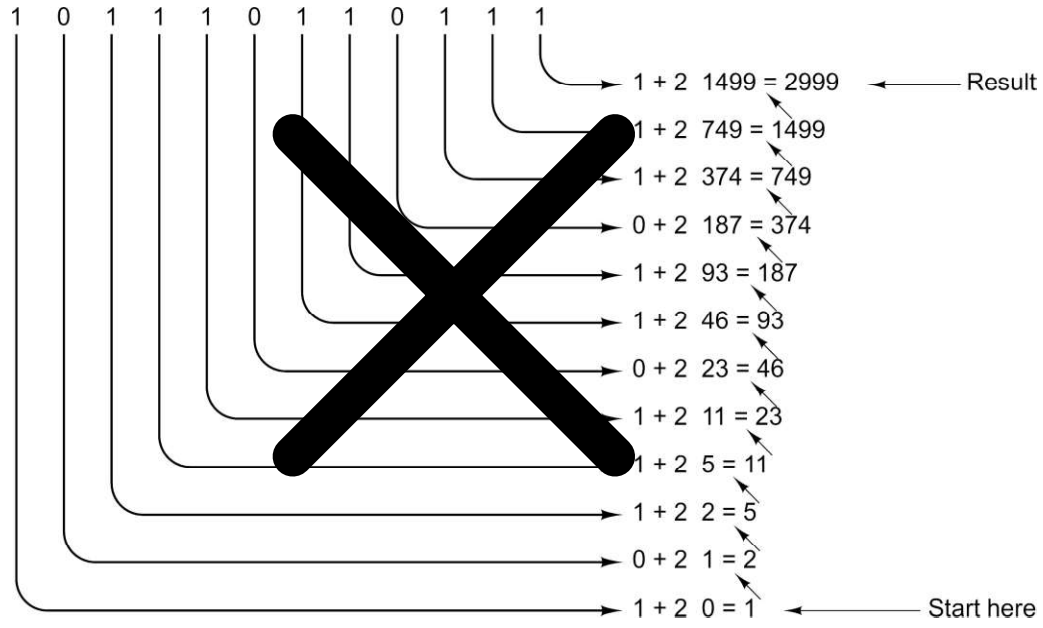
$$10111010100.01$$

$$2^{10} + 2^8 + 2^7 + 2^6 + 2^4 + 2^2 + 2^{-2}$$

$$1024+256+128+64+16+4+0.25$$

$$1492.25$$

Binario ⇒ Decimale (Secondo metodo): la codifica viene ottenuta ricercando il valore posizionale di ogni cifra tramite successive moltiplicazioni per due a partire dalle cifre più significative verso quelle meno significative.



Altre conversioni: la conversione da decimale a ottale e da decimale a esadecimale può essere fatta passando per il sistema binario, oppure applicando i metodi precedentemente descritti e ponendo attenzione al corretto uso delle basi.

Esercizi

1. Si convertano i seguenti numeri decimali in base 2:

   371 3224 114.65625

2. Si convertano i seguenti numeri binari in base 8 e 16:

   111001101001 10 1111001100011100

3. Si convertano i seguenti numeri esadecimali in base 2:

   FA31C CCCAB001

4. Si convertano i seguenti numeri esadecimali in base 10:

   AAB E0CC

Il calcolatore e i numeri binari

Prima di procedere nello studio dei numeri binari è bene ricordare che il codice e i dati di un programma vengono memorizzati, e successivamente, utilizzati da un calcolatore la cui architettura interna stabilisce il loro formato e il campo dei valori che possono assumere.

La più importante unità di misura dell'informazione manipolata dal calcolatore è il **BYTE** composto da 8 bit.

0	1	1	0	1	0	1	1

Nel byte il bit più a destra è quello meno significativo mentre quello a sinistra è quello più significativo.

Sequenze di bit più lunghe di un byte sono denominate **WORD**. La loro lunghezza dipende dalle caratteristiche del sistema, ma è sempre un multiplo del byte: 16/32/64/128 bit.

L'intervallo di valori codificabili dipende ovviamente dal numero di configurazioni possibili e dal tipo di dato da rappresentare. Con *n* bit sono possibili 2^n configurazioni.

Esempio: *Intervallo di interi positivi rappresentabili con n bit*

n	**Num. Configurazioni**	**Intervallo**
1	2	0 - 1
8	256	0 - 255
16	65.536	0 – 65.535
32	4.294.967.296	0 - 4.294.967.295
64	18.446.744.073.709.551.616	0 – 18.446.744.073.709.551.615

Unità di misura nel sistema binario

Il bit rappresenta la più piccola unità di misura dell'informazione memorizzabile in un calcolatore. I sistemi moderni memorizzano e manipolano miliardi di bit; per questo motivo sono stati definiti molti multipli.

Nome	Sigla	In bit	In byte	In potenze di 2
Bit	Bit	1 bit	1/8	2^1=2 stati
Byte	Byte	8	1	2^8=256 stati
KiloByte	KB	8.192	1.024	2^{10} byte
MegaByte	MB	8.388.608	1.048.576	2^{20} byte
GigaByte	GB	8.589.934.592	1.073.741.824	2^{30} byte
TeraByte	TB	8.796.093.022.208	1.099.511.627.776	2^{40} byte

I prefissi (Kilo, Mega, ecc.) sono normalmente associati a potenze di 10 mentre per i multipli del bit si opera su potenze di 2.

ATTENZIONE 1MB non corrisponde a 1000KB ma a 1024KB

Numeri binari negativi (1)

Nell'evoluzione dell'aritmetica binaria sono state utilizzate quattro rappresentazioni per i numeri negativi (Grandezza e segno, Complemento a uno, Complemento a due, Eccesso 2^{m-1}). Le diverse soluzioni mirano a ottimizzare le operazioni da svolgere sui numeri binari.
La codifica ideale prevede:

- Una sola rappresentazione per lo 0
- Lo stesso insieme di valori positivi e negativi rappresentabili

Questa codifica non è ottenibile per numeri binari poiché prevede di rappresentare un numero dispari di valori mentre avendo *k* bit a disposizione si potranno sempre rappresentare 2^k valori.

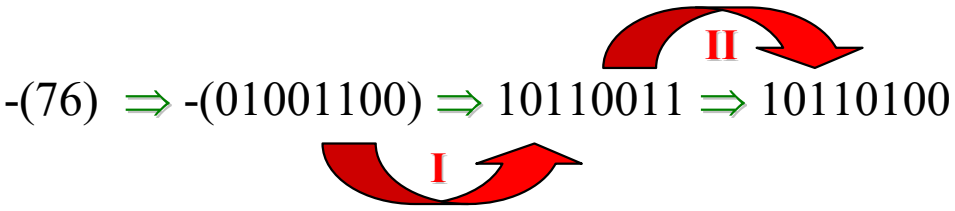
Grandezza e segno: con questa rappresentazione il bit più a sinistra viene utilizzato per il segno: con 0 si indicano i valori positivi mentre con 1 quelli negativi. I bit rimanenti contengono il valore assoluto del numero.

$$76 \Leftrightarrow 01001100 \qquad -76 \Leftrightarrow 11001100$$

Complemento a due: con questa rappresentazione il bit più a sinistra viene utilizzato per il segno: con 0 si indicano i valori positivi mentre con 1 quelli negativi. La negazione di un numero richiede due passi:

I. Si sostituiscono tutti gli uno con degli zero e viceversa
II. Si aggiunge 1 al risultato

Esempio: *Rappresentazione in complemento a due di –76*



Numeri binari negativi (2)

~~**Eccesso 2^{m-1}:** rappresenta i numeri come somma di se stessi con 2^{m-1} dove m è il numero di bit utilizzati per rappresentare il valore. Si noti che il sistema è identico al complemento a due con il bit di segno invertito. In pratica i numeri tra −128 e 127 sono mappati tra 0 e 255.~~

~~Esempio: *Rappresentazione tramite eccesso a 2^{m-1} per −76 supponendo che il valore sia memorizzato in un byte (m=8 ⇒ 2^{m-1} = 2^7 = 128)*~~

~~$-76 \Rightarrow -76+128 = 52 = 00110100$~~

Decimale N	**Binario N** (senza rappresentare i numeri negativi)	**Grand. e Segno -N**	**Complemento a due -N**	~~**Eccesso 128 -N**~~
0	**00000000**	**10000000**	**00000000**	**10000000**
1	00000001	10000001	11111111	01111111
2	00000010	10000010	11111110	01111110
3	00000011	10000011	11111101	01111101
10	00001010	10001010	11110110	01110110
20	00010100	10010100	11101100	01101100
50	00110010	10110010	11001110	01001110
100	01100100	11100100	10011100	00011100
127	01111111	11111111	10000001	00000001
128	**10000000**	-	**10000000**	**00000000**

Si noti che:

- La rappresentazione grandezza e segno presenta due configurazioni diverse per lo zero: lo 0 positivo (00000000) e lo 0 negativo (10000000).
- Nelle rappresentazioni in complemento a due e ~~in eccesso 2^{m-1}~~ gli insiemi di valori positivi e negativi rappresentabili sono diversi poiché entrambe presentano una sola rappresentazione per lo 0.

Numeri binari negativi (3)

Le rappresentazioni in complemento a due ~~ad eccesso 2^{m-1}~~ sono le più efficienti per svolgere operazioni in aritmetica binaria poiché permettono di trattare la sottrazione tra numeri come una somma tra numeri di segno opposto.

$$(X - Y) = (X + (-Y))$$

Qualunque sia la rappresentazione utilizzata il numero di configurazioni rappresentabili non cambia ma, rispetto al caso in cui vengano rappresentati solo numeri positivi, l'intervallo positivo è dimezzato a favore dei valori negativi.

n	**Configurazioni**	**Positivi**	**Positivi e Negativi**
8	256	0 – 255	-128 – 127
16	65.536	0 – 65.535	-32768 – 32767
32	4.294.967.296	0 – 4.294.967.295	-2.147.483.648 – 2.147.483.647

Somma tra numeri binari

La tavola di addizione per i numeri binari è indicata di seguito:

	Configurazioni			
Addendo	0	0	1	1
Addendo	0	1	0	1
Somma	0	1	1	0
Riporto	0	0	0	1

Il procedimento di somma binaria è equivalente a quello nel sistema decimale con eccezione del riporto che si genera quando entrambi gli addendi hanno valore 1.

Operando con numeri in complemento a due:

- Il **riporto** generato dai bit più a sinistra viene ignorato.
- Se gli addendi sono di segno opposto non si può verificare un overflow.
- L'**overflow** si verifica se il riporto generato nel sommare i bit di segno è diverso dal riporto utilizzato per sommare i bit di segno (normalmente l'overflow viene indicato da un particolare bit di overflow del sommatore).

Addendo	54	00110110
Addendo	30	00011110
Somma	84	01010100
Riporto		00111110

Addendo	85	01010101
Addendo	74	01001010
Somma	159	10011111
Riporto		**01**000000

Overflow

Per farla breve
utilizzate questa definizione:
c'è overflow se, sommando
numeri entrambi positivi
il risultato viene negativo
e viceversa: se sommando
numeri negativi
si ottiene un numero
con segno positivo

Addendo	10	00001010
Addendo	-3	11111101
Somma	7	00000111
Riporto		**1**1111000

Ignorato