

Corso di

Fondamenti di Informatica

Alessandra Volpato
ale.volpato@ieee.org

Rappresentazione

Vincoli

- Numero finito di cifre
- Esempio: 3 cifre \rightarrow 0..999

Conseguenze

- Il risultato di un'operazione puo' non essere rappresentabile
- Esempio: $368+715$
- Esempio: $512/003$

Rappresentazione

Notazione posizionale base p

- alfabeto = $\{0, 1, 2, \dots, p-1\}$
- $N_p = a_n a_{n-1} \dots a_1 a_0 \cdot a_{-1} a_{-2} \dots a_{-m+1} a_{-m}$
- $n \geq 0, m \geq 1, a_i \in \text{alfabeto}$
- **p** e' la **base**
- Il **punto** separa gli elementi associati a termini con esponente positivo o nullo da quelli associati a termini con esponente negativo
- $$N_p = p^n \times a_n + p^{n-1} \times a_{n-1} + \dots + p^1 \times a_1 + p^0 \times a_0 +$$
$$+ p^{-1} \times a_{-1} + p^{-2} \times a_{-2} + \dots + p^{-m+1} \times a_{-m+1} + p^{-m} \times a_{-m}$$

Rappresentazione

Usualmente

- base = 10

Nel calcolatore

- base = 2

Per comodita'

- base = 8
- base = 16

Rappresentazione

Unità

- b = bit (binary digit)
- B = Byte = 8b

Multipli

- $K = 2^{10} = 1024 \approx 10^3$ [Kilo]
- $M = 2^{20} \approx 10^6$ [Mega]
- $G = 2^{30} \approx 10^9$ [Giga]
- $T = 2^{40} \approx 10^{12}$ [Tera]

Rappresentazione

Interi : Notazione posizionale base p

- alfabeto = $\{0, 1, 2, \dots, p-1\}$
- $N_p = a_n a_{n-1} \dots a_1 a_0$, $n \geq 0$, $a_i \in$ alfabeto
- conversione dalla base p alla base 10 :

$$p^n \times a_n + p^{n-1} \times a_{n-1} + \dots + p^1 \times a_1 + p^0 \times a_0$$

dove p e' espresso in base 10

Rappresentazione

Interi : Notazione posizionale base 2

- alfabeto = $\{0, 1\}$
- $N_2 = a_n a_{n-1} \dots a_1 a_0$, $n \geq 0$, $a_i \in$ alfabeto
- conversione dalla base 2 alla base 10 :

$$2^n \times a_n + 2^{n-1} \times a_{n-1} + \dots + 2^1 \times a_1 + 2^0 \times a_0$$

- esempio :

$$1101_2 \rightarrow 2^3 \times 1 + 2^2 \times 1 + 2^1 \times 0 + 2^0 \times 1 = 13_{10}$$

Rappresentazione

Interi : Notazione posizionale base 16

- alfabeto = $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$
- $N_{16} = a_n a_{n-1} \dots a_1 a_0, n \geq 0, a_i \in \text{alfabeto}$
- conversione dalla base 16 alla base 10 :

$$16^n \times a_n + 16^{n-1} \times a_{n-1} + \dots + 16^1 \times a_1 + 16^0 \times a_0$$

- esempio :

$$B7F_{16} \rightarrow 16^2 \times 11 + 16^1 \times 7 + 16^0 \times 15 = 2943_{10}$$

Rappresentazione

Interi : Notazione posizionale base p

- conversione dalla base 10 alla base p
 - input N
 - **algoritmo**

```
 $d \leftarrow \lfloor \log_p N \rfloor$   
for  $i = 0$  to  $d$   
     $a_i \leftarrow N \bmod p$   
     $N \leftarrow N \operatorname{div} p$ 
```

- a_i e' il **resto** della divisione fra interi
- N e' il **quoziente** della divisione fra interi

Rappresentazione

Interi $10 \rightarrow p$: Esempio

- $p = 2, N = 87, d = \lfloor \log_2 87 \rfloor = 6$

$$a_0 = 1 \quad N = 43$$

$$a_1 = 1 \quad N = 21$$

$$a_2 = 1 \quad N = 10$$

$$a_3 = 0 \quad N = 05$$

$$a_4 = 1 \quad N = 02$$

$$a_5 = 0 \quad N = 01$$

$$a_6 = 1 \quad N = 00$$

$$87_{10} \rightarrow 1010111_2$$

Rappresentazione

Frazionari : Notazione posizionale base p

- alfabeto = $\{0, 1, 2, \dots, p-1\}$
- $N_p = 0 . a_{-1} a_{-2} \dots a_{-n+1} a_{-n}$, $n \geq 1$, $a_i \in$ alfabeto
- conversione dalla base p alla base 10 :

$$p^{-1} \times a_{-1} + p^{-2} \times a_{-2} + \dots + p^{-n+1} \times a_{-n+1} + p^{-n} \times a_{-n}$$

dove p e' espresso in base 10

Rappresentazione

Frazionari : Notazione posizionale base 2

- alfabeto = $\{0, 1\}$
- $N_p = 0 . a_{-1} a_{-2} \dots a_{-n+1} a_{-n}$, $n \geq 1$, $a_i \in$ alfabeto
- conversione dalla base 2 alla base 10 :

$$2^{-1} \times a_{-1} + 2^{-2} \times a_{-2} + \dots + 2^{-n+1} \times a_{-n+1} + 2^{-n} \times a_{-n}$$

- esempio :

$$\begin{aligned} 0.1011_2 &\rightarrow 2^{-1} \times 1 + 2^{-2} \times 0 + 2^{-3} \times 1 + 2^{-4} \times 1 \\ &= 0.6875_{10} \end{aligned}$$

Rappresentazione

Frazionari : Notazione posizionale base p

- conversione dalla base 10 alla base p

- input N

- **algoritmo**

- $i = -1$

- repeat**

- $a_i \leftarrow \lfloor N \times p \rfloor$

- $N \leftarrow (N \times p) - \lfloor N \times p \rfloor$

- $i \leftarrow i - 1$

- until** $N = 0$

- l'algoritmo potrebbe non terminare

Rappresentazione

Frazionari $10 \rightarrow p$: Esempio

• $p = 2, N = 0.3$

$$a_{-1} = 1 \qquad N = 0.25$$

$$a_{-2} = 0 \qquad N = 0.50$$

$$a_{-3} = 1 \qquad N = 0.00$$

$$0.625_{10} \rightarrow 0.101_2$$

Rappresentazione

Frazionari $10 \rightarrow p$: Esempio periodico

- $p = 2, N = 0.625$

$$a_{-1} = 0 \qquad N = 0.60$$

$$a_{-2} = 1 \qquad N = 0.20$$

$$a_{-3} = 0 \qquad N = 0.40$$

$$a_{-4} = 0 \qquad N = 0.80$$

$$a_{-5} = 1 \qquad N = 0.60$$

$$a_{-6} = 1 \qquad N = 0.20$$

- $0.3_{10} \rightarrow 0.010011_2$

Rappresentazione

Reali : Notazione floating point

- b = base (intero)
- S = +1 / -1
- E = esponente (intero)
- F = mantissa (frazionario)
- conversione alla base 10 :

$$S \times F \times b^E$$

Rappresentazione

Floating Point : Esempio

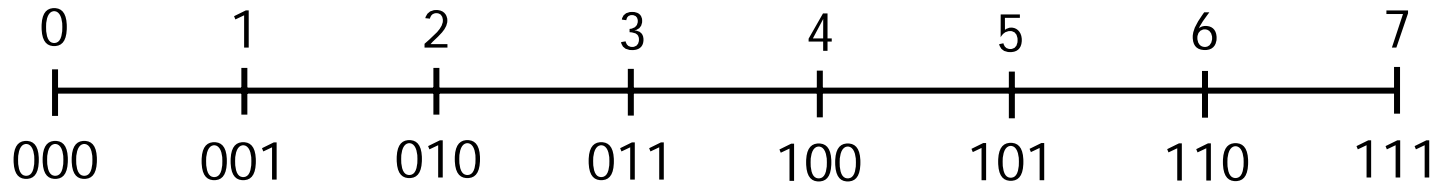
- $b = 10, S = +1, E = 11, F = 3.49$
- conversione alla base 10 :

+ 349 000 000 000

Rappresentazione macchina

Interi senza segno su n bit

- conversione :
 - $N_{10} \rightarrow N_2$
 - aggiungo zeri a sinistra sino a n bit
- esempio ($n = 3$) :

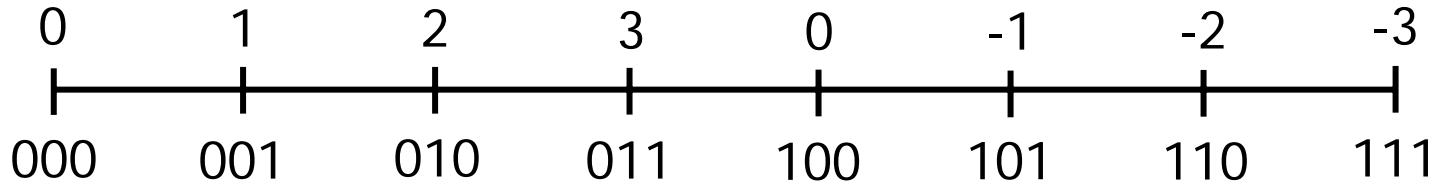


- range : $[0, 2^n - 1]$

Rappresentazione macchina

Modulo e segno su n bit

- conversione :
 - $|N_{10}| \rightarrow N_2$
 - aggiungo zeri a sinistra sino a $n - 1$ bit
 - $a_n = 0$ se $N_{10} \geq 0$; $a_n = 1$ se $N_{10} < 0$
- esempio ($n = 3$) :

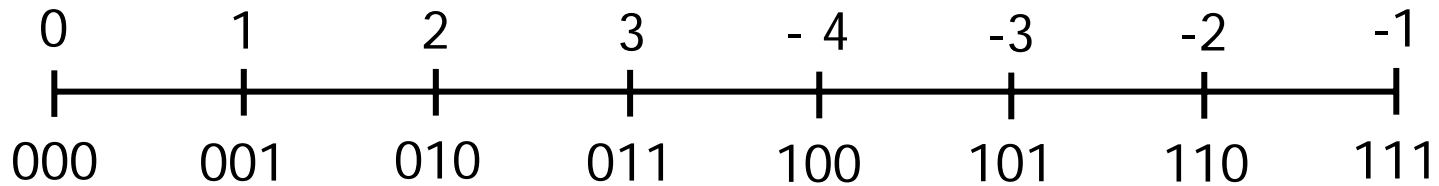


- range : $[-2^{n-1} - 1, 2^{n-1} - 1]$

Rappresentazione macchina

Complemento a 2 su n bit

- conversione (traslazione asse negativi) :
 - N_2 se $N_{10} \geq 0$; $(N_{10} + 2^n)_2$ se $N_{10} < 0$
 - aggiungo zeri a sinistra sino a n bit
- esempio ($n = 3$) :



- range : $[-2^{n-1}, 2^{n-1} - 1]$

Complemento a 2 su n bit

- La notazione con n bit di un numero binario:

$$a_{n-1} a_{n-2} \dots a_1 a_0$$

dove gli a_i sono le n cifre binarie,

corrisponde al valore numerico dato da:

$$N_{10} = - a_{n-1}2^{n-1} + a_{n-2}2^{n-2} + \dots + a_22^2 + a_12 + a_0$$

- Il bit piu' significativo (bit di segno) ha peso negativo

Rappresentazione macchina

Modulo e segno

- intervallo di valori simmetrico
- 2 rappresentazioni per il valore 0
- algoritmi di somma differenti a seconda del segno (non posso eseguire la somma per colonne)

Complemento a 2

- una sola rappresentazione per lo 0 (un valore in più disponibile nel range rispetto al modulo e segno)
- posso applicare l'algoritmo di somma per colonne
- e' ciclica

Rappresentazione macchina

Complemento a 2 : Somma

- somma per colonne

$$\begin{array}{r} + 2 \quad 0010 \\ - 5 \quad 1011 \\ \hline \end{array}$$

$$\begin{array}{r} - 3 \quad 1101 \end{array}$$

$$\begin{array}{r} - 5 \quad 1011 \\ - 2 \quad 1110 \\ \hline \end{array}$$

$$\begin{array}{r} - 7 \quad (1)1001 \end{array}$$

Complemento a 2 : Overflow

- **regola**

si ha **overflow** se e solo se esiste un solo riporto sulle colonne C_{n-1} e C_n

- 5	1 0 1 1	+ 5	0 1 0 1
- 5	1 0 1 1	+ 4	0 1 0 0
—	—	—	—
error	0 1 1 0	error	1 0 0 1

Complemento a 2 : Cambio segno

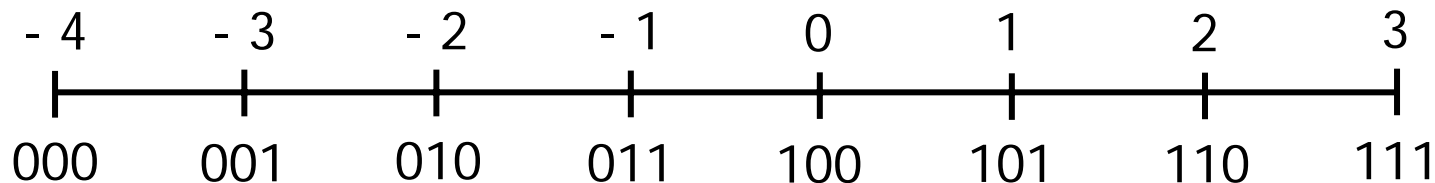
- **algoritmo** :
 - identificare l'occorrenza più a destra del simbolo 1
 - complementare tutti i bit a sinistra dell'occorrenza identificata
- esempio ($n = 8$) :

+ 52	0 0 1 1 0 1 0 0
- 52	1 1 0 0 1 1 0 0

Rappresentazione macchina

Eccesso k su n bit

- consideriamo il caso $k = 2^{n-1}$
- conversione (traslazione intero asse) :
 - $(N_{10} + 2^{n-1})_2$
 - aggiungo zeri a sinistra sino a n bit
- esempio ($n = 3$):



- range : $[-2^{n-1}, 2^{n-1} - 1]$

Reali : Standard IEEE (1985)

- $S = 0 / +1$
- $E =$ intero eccesso M (binario)
- $F =$ frazionario binario normalizzato in modo tale che il bit più significativo corrisponda ad a_0 e sia omesso
($F = 1 . a_{-1} a_{-2} \dots a_{-n+1} a_{-n}$, a_0 e' detto *hidden bit*)
- conversione dallo standard ad un numero reale :

$$-1^S \times 1.F \times 2^{E-M}$$

- numeri molto vicini in prossimità dello zero, molto distanti al crescere del valore assoluto

Reali : Standard IEEE (cont.)

- le specifiche per la rappresentazione di S , E , M ed F variano a seconda delle diverse precisioni

	singola	doppia	quadrupla
S (b)	1	1	1
E (b)	8	11	15
M (val)	127	1023	32767
F (b)	23	52	112
tot (B)	4	8	16

Rappresentazione macchina

Standard IEEE : Esempio

- $X_{IEEE} = 11000011\ 11010000\ 00000000\ 00000000$
 $= 1\ 10000111\ 101000000000000000000000$

La rappresentazione e' in **singola precisione**

- conversione ad un numero reale :
 - $S = 1$
 - $E = 10000111_2 = 135_{10}$
 - $F = 0.101_2 = 0.625_{10}$
 - $X = (-1)^1 \times 2^{135 - 127} \times 1.625 = - 416_{10}$

Standard IEEE : Esempio

- $X = 42.6875$
 $= 101010.1011_2$
 $= 1.010101011 \times 2^5$
- conversione alla precisione singola :
 - $S = 0$
 - $E = 5 + 127 = 132 = 10000100$
 - $F = 010101011000000000000000$
 - $X_{IEEE} = 0\ 10000100\ 010101011000000000000000$
 $= 01000010\ 00101010\ 11000000\ 00000000$

Standard IEEE: numeri rappresentabili in singola precisione

- il modulo della mantissa e' compreso fra 1 e 2

$$1.0\dots0_2 \leq |F| \leq 1.1\dots1_2$$

$$1 \leq |F| \leq (2-2^{-23})$$

- Il numero con il modulo piu' grande e'

$$(2-2^{-23}) \cdot 2^{+127} \approx 3.4_{10} \cdot 10^{+38}$$

- Il numero con il modulo piu' piccolo e'

$$1 \cdot 2^{-126} \approx 1.8_{10} \cdot 10^{-38}$$

Standard IEEE: numeri rappresentabili in singola precisione

- La distanza fra due numeri successivi rappresentabili: dipende dal valore associato al bit meno significativo di F
 $d = 0.0\dots\dots 1 \cdot 2^E = 2^{-23} \cdot 2^E$
- la distribuzione dei numeri non e' uniforme
- i numeri sono piu' densi vicino allo zero

Standard IEEE: conseguenze

- la normalizzazione di F garantisce una rappresentazione univoca del numero
- l'esponente e è il minimo possibile
- i numeri sono il più vicini possibile
- non ci sono zeri 'inutili' nella rappresentazione di F
- l'errore relativo (singola precisione):
$$\left(\frac{d}{2} \right) / N = \left(2^{-23} \cdot 2^E \right) / \left(2 \cdot F \cdot 2^E \right)$$

è compreso fra 2^{-24} e 2^{-25} , cioè fra $6_{10} \cdot 10^{-8}$ e $3_{10} \cdot 10^{-8}$;
- ci sono sempre circa 7 cifre decimali significative (singola precisione)

Codice ASCII

- ASCII = American Standard Code for Information Interchange
- utilizzato per la rappresentazione dei caratteri più comuni, divisi in
 - alfanumerici : a - z, A - Z, 0 - 9
 - segni : + / ? ; : ...
 - comandi : return, tab, ...

Codice ASCII

- 7 bit utilizzati, valori nel range [0 .. 127] :
 - non stampabili codificati in [0 .. 31]
 - A-Z codificati in [65 .. 90]
 - a-z codificati in [97 .. 122]
 - 1-9 codificati in [48 .. 57]
- valori consecutivi rispetto all'ordine alfabetico