Fondamenti di Informatica

per la Sicurezza

a.a. 2007/08

Rappresentazione binaria

Stefano Ferrari

UNIVERSITÀ DEGLI STUDI DI MILANO
DIPARTIMENTO DI TECNOLOGIE DELL'INFORMAZIONE

Citazione

Ci sono 10 tipi di persone,
chi capisce il binario
e chi no.

[Anonimo]

Numeri naturali

È il tipo più naturale da rappresentare: rappresentazione in notazione posizionale in base due.

Esempio: $(13)_{10} = (1101)_2 \rightarrow \mathbf{00001101}$, usando un byte.

Il primo bit viene chiamato bit più significativo (Most Significant Bit, MSB).

L'ultimo bit viene chiamato bit meno significativo (Least Significant Bit, LSB).

MSB $\mathbf{00001101}$ LSB

Numeri interi con segno

Con bit di segno (rappresentazione segno e modulo):

0	0	0	+0
0	0	1	+1
0	1	0	+2
0	1	1	+3
1	0	0	-0
1	0	1	-1
1	1	0	-2
1	1	1	-3

- il bit più significativo rappresenta il segno, gli altri bit rappresentano il modulo;
- lo stesso numero (lo zero) viene rappresentato con due numerali differenti;
- le operazioni aritmetiche sono macchinose.

Complemento a 2 (1)

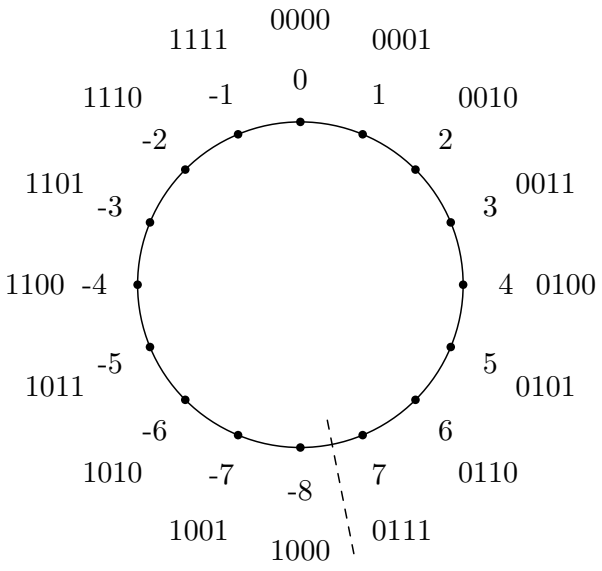
Per rappresentare in complemento a 2 il numero x usando una sequenza di n bit, si rappresenta in binario (usando $n+1$ bit) il numero $2^n + x$, e scartando poi il bit più significativo.

Esempio

Usando 4 bit ($n = 4$):

- $6 \longrightarrow 2^4 + 6 = (22)_{10} = (10110)_2 \xrightarrow{\text{compl. a 2}} 0110$
- $-6 \longrightarrow 2^4 - 6 = (10)_{10} = (01010)_2 \xrightarrow{\text{compl. a 2}} 1010$

Complemento a 2 (2)



Complemento a 2 (3)

Questa codifica equivale ad una rappresentazione posizionale modificata:

$$-a_{n-1} \cdot 2^{n-1} + a_{n-2} \cdot 2^{n-2} + \cdots + a_1 \cdot 2^1 + a_0 \cdot 2^0$$

Esempio

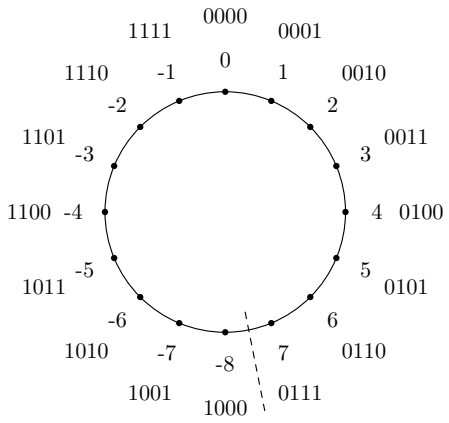
$1010 = -8 + 2 = -6$

Complemento a 2 (4)

Con questa rappresentazione:

- lo zero ha rappresentazione unica;
- con n bit, si rappresentano i numeri interi dell'intervallo $[-2^{n-1}, 2^{n-1} - 1]$;
- i numeri negativi hanno il MSB che vale 1, gli altri 0 (di fatto, il MSB è il bit di segno);
- la somma si realizza considerando le sequenze di bit come numeri binari;
- la sottrazione si realizza invertendo il valore del sottraendo e poi sommandolo al minuendo in notazione binaria.

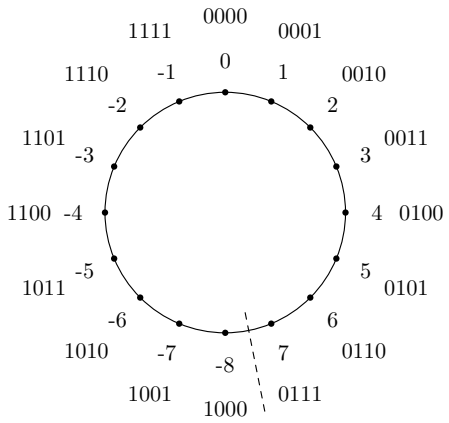
Decodifica in complemento a due



Decodifica:

- se il primo simbolo è 0 si considera semplicemente come un numero binario;
- se il primo simbolo è 1 lo si decodifica in binario e ad esso si sottrae il numero 2^n.

Inversione in complemento a 2

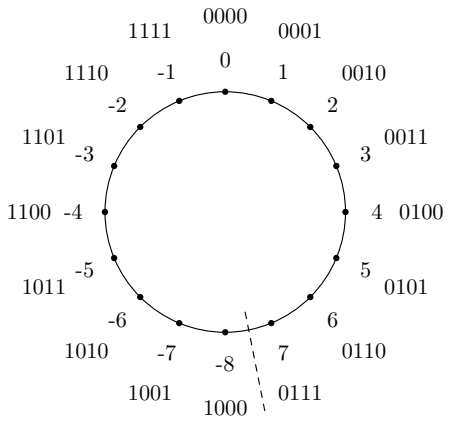


Inversione:

- si invertono tutte le singole cifre binarie;
- si somma 1.

L'inversione di -2^{n-1} dà un errore (overflow).

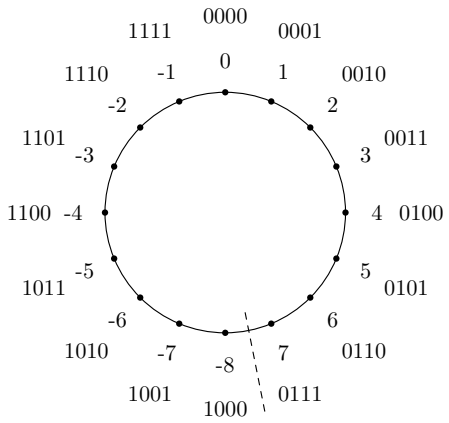
Codifica in complemento a 2



Codifica:

- codifica del valore assoluto in notazione posizionale binaria;
- se è negativo, lo si inverte.

Operazioni algebriche in complemento a 2



Somma:

- normale somma binaria.

Sottrazione:

- si inverte il sottraendo;
- lo si somma al minuendo.

Overflow

L'overflow si manifesta quando si cerca di rappresentare un numero troppo grande.

Esempio

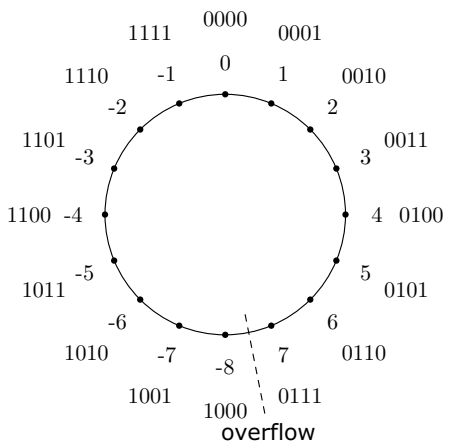
Usiamo una rappresentazione in complemento a 2 a 4 bit per i numeri da -8 a 7.

Il risultato della somma 6+5 non è rappresentabile.

Analoghe considerazioni per il risultato di -7-4.

Con la notazione in complemento a 2, l'overflow si rileva facilmente.

Individuazione dell'overflow



Overflow:

- somma di valori con segno opposto e sottrazione di valori con segni concordi non dà mai overflow;
- negli altri casi, c'è stato overflow se il segno del risultato è discorde con il segno del primo operando.

Overflow in complemento a 2

Si verifica un overflow se:

C = A + B

A	B	C
+	+	-
-	-	+

C = A - B

A	B	C
+	-	-
-	+	+

L'overflow nell'inversione di -2^{n-1} non deve essere considerato nel calcolo di una sottrazione (quando -2^{n-1} è il sottraendo).

Codifica (1)

D Codificare il numero 2 in complemento a 2 a 4 bit, specificando se si verifica un overflow.

R Poiché $2 \leq 2^{4-1} - 1$, non ci sarà overflow.

Numero da codificare: $2^4 + 2 = 18$.

Codifica binaria troncata a 4 bit: 0010.

Codifica (2)

D Codificare il numero 10 in complemento a 2 a 3 bit, specificando se si verifica un overflow.

R Poiché $10 > 2^{3-1} - 1$, ci sarà overflow.

- Numero da codificare: $2^3 + 10 = 18$

  Codifica binaria troncata a 3 bit: 010

Codifica (3)

D Codificare il numero -13 in complemento a 2 a 3 bit, specificando se si verifica un overflow.

R Poiché $-13 < -2^{3-1}$, ci sarà overflow.

Numero da codificare: $2^3 - 13 = -5$.

Codifica binaria troncata a 3 bit del valore assoluto (5): 101.

Inversione in complemento a 2: 011.

Decodifica (1)

D Quale numero è rappresentato dalla stringa binaria 10111, codificata in complemento a due?

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

Poiché la prima cifra binaria è 1, 10111 viene decodificato come: $23 - 2^5 = -9$.

Decodifica (2)

D Quale numero è rappresentato dalla stringa binaria 01101, codificata in complemento a due?

R $(01101)_2 = (13)_{10}$.

Poiché la prima cifra binaria è 0, 01101 viene decodificato come 13.

Inversione (1)

D Quale stringa binaria è l'inverso in complemento a due della stringa binaria 01101?

R Invertendo i bit di 01101, si ottiene 10010.

Sommando 1 a 10010, si ottiene 10011.

Troncando a 5 bit 10011, si ottiene 10011.

Inversione (2)

D Quale stringa binaria è l'inverso in complemento a due della stringa binaria 00?

R Invertendo i bit di 00, si ottiene 11.

Sommando 1 a 11, si ottiene 100.

Troncando a 2 bit 100, si ottiene 00.

Somma (1)

D Date le stringhe binarie A = 0000 e B = 0011, calcolare in complemento a due la loro somma, specificando se si verifica un overflow.

R La somma binaria di 0000 e 0011, troncata a 4 bit è 0011.

Poiché le due stringhe date hanno il primo bit uguale, e uguale anche al primo bit del risultato, non si è verificato un overflow.

Somma (2)

D Date le stringhe binarie A = 111101 e B = 100001, calcolare in complemento a due la loro somma, specificando se si verifica un overflow.

R La somma binaria di 111101 e 100001, troncata a 6 bit è 011110.

Poiché le due stringhe date hanno il primo bit uguale, ma diverso dal primo bit del risultato, si è verificato un overflow.

Sottrazione (1)

D Date le stringhe binarie A = 1011000 e B = 0101000, calcolare in complemento a due la loro differenza, specificando se si verifica un overflow.

R Il complemento a 2 di 0101000 è 1011000.

La somma binaria di 1011000 e 1011000, troncata a 7 bit è 0110000.

Poiché le due stringhe date hanno il primo bit diverso, e il primo bit del risultato è uguale al primo bit della seconda stringa, si è verificato un overflow.

Sottrazione (2)

D Date le stringhe binarie A = 100011 e B = 110000, calcolare in complemento a due la loro differenza, specificando se si verifica un overflow.

R Il complemento a 2 di 110000 è 010000.

La somma binaria di 100011 e 010000, troncata a 6 bit è 110011.

Poiché le due stringhe date hanno il primo bit uguale, non si è verificato un overflow.

Numeri "con la virgola"

Per la rappresentazione dei numeri razionali e reali ci sono due impedimenti:

- per via della notazione posizionale, alcuni numeri non possono essere rappresentati con un numero limitato di cifre (e.g., $\frac{1}{3}$ in base 10);
- alcuni numeri reali (gli irrazionali) non possono proprio essere rappresentati in nessuna base (e.g., π).

Numeri razionali

Per gli usi pratici, i numeri irrazionali possono essere approssimati da un numero razionale:

- per esempio, $3.14 \approx \pi$.

Per rappresentare numeri razionali ci sono due notazioni, dette:

- virgola fissa;
- virgola mobile.

Virgola fissa (1)

Data una sequenza di n bit, viene stabilito a priori quanti di questi, m rappresenteranno la parte intera, e quanti, k, la parte frazionaria.

$$\overbrace{\underbrace{a_{n-1} \cdots a_k}_{m \text{ bit}} . \underbrace{a_{k-1} \cdots a_0}_{k \text{ bit}}}^{n \text{ bit}}$$

Equivale a dividere per 2^k.

Virgola fissa (2)

Esempio:

$(10.101)_2 = 1 \cdot 2^1 + 0 \cdot 2^0 + 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} = 1 \cdot 2 + 0 \cdot 1 + 1 \cdot 0.5 + 0 \cdot 0.25 + 1 \cdot 0.125 = 2.625$

$(10101)_2/2^3 = (1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 1 \cdot 16 + 0 \cdot 8 + 1 \cdot 4 + 0 \cdot 2 + 1 \cdot 1)/8 = 21/8 = 2.625$

Virgola fissa (3)

In virgola fissa, la precisione assoluta è fissata:

1	0	3	0.	2	7

e

0	0	0	1.	0	9

hanno la stessa precisione: 1/100.

Per il primo numero, la precisione relativa è $1/10^6$, mentre per il secondo numero, la precisione relativa è $1/10^2$.

1.091 e 1.094 sono indistinguibili: vengono rappresentati con 1.09.

Virgola fissa (4)

Problema di underflow: può non essere possibile rappresentare un numero piccolo.

Come si rappresenta 0.001?

0	0	0	0.	0	0

!

Virgola mobile (1)

Un numero razionale può essere rappresentato nella forma: $m \cdot b^e$, b è una base intera.

Esempio

32.1 può essere scritto come $0.321 \cdot 10^2$.

m è detto mantissa, e è detto esponente.

In virgola mobile, un numero viene rappresentato tramite la coppia mantissa-esponente.

Virgola mobile (2)

In virgola mobile, la precisione relativa è fissata.

Il numero di bit dedicati a:

- m fissano la precisione relativa con cui il numero può essere rappresentato;
- e fissano l'estensione dell'intervallo rappresentabile.

IEEE Standard 754 Floating Point Numbers:

- singola precisione: 32 bit (1 per il segno, 8 per l'esponente e 23 per la mantissa);
- doppia precisione: 64 bit (1 per il segno, 11 per l'esponente e 52 per la mantissa).

Fissa o mobile?

Con lo stesso budget di bit, possiamo rappresentare lo stesso numero di valori.

Virgola fissa e mobile si differenziano per:

- distribuzione dei valori rappresentabili sulla retta dei reali;
- proprietà aritmetiche (e.g., l'ordine con cui si eseguono le operazioni è importante in virgola mobile);
- precisione (e.g., l'uso della virgola mobile per calcoli finanziari è pericoloso);
- costo computazionale.