UNIVERSITÀ DEGLI STUDI DI TRIESTE

Rappresentazione dell'informazione (3)

Prof.ssa Giulia Cisotto

giulia.cisotto@units.it

Trieste, 6 marzo 2025

RAPPRESENTAZIONE DI UN NUMERO INTERO NEGATIVO

Ci sono tre diverse possibili rappresentazioni:

- RAPPRESENTAZIONE «MODULO E SEGNO»
- Complemento a 1
- Complemento a 2

RAPPRESENTAZIONE DI UN NUMERO INTERO NEGATIVO

Ci sono tre diverse possibili rappresentazioni:

- RAPPRESENTAZIONE «MODULO E SEGNO»

Il primo bit (MSB) viene usato per il segno: 0 se POSITIVO, 1 se NEGATIVO

I rimanenti bit vengono usati per il modulo del numero

ESEMPIO su 8 bit $-4_{10} = \mathbf{1}0000100_2$

Range di rappresentazione con *n* bit è $[\ -(2^{n-1} - 1),\ 2^{n-1} - 1]_{10}$

Problemi: **(1)** un bit è speso per il segno, **(2)** lo ZERO ha due diverse rappresentazioni

$0000_2 = +0_{10}$ $1000_2 = -0_{10}$

RAPPRESENTAZIONE DI UN NUMERO INTERO NEGATIVO

Ci sono tre diverse possibili rappresentazioni:

- Complemento a 1

Si «invertono» tutti i bit: tutti i bit a zero vengono messi a 1 e tutti quelli a 1 vengono messi a 0 (*COMPLEMENTO*).

Se il numero è positivo, si converte in binario usando il metodo standard.

Se il numero è negativo, si converte in binario il modulo del numero, poi si trasforma nel suo complemento

ESEMPIO su 4 bit

$3_{10} = 0011_2$

$-3_{10} = \overline{0011_2} = 1100_2$

Rimane il problema che lo ZERO ha due diverse rappresentazioni

0000 0000 (+0)

1111 1111 (-0)

RAPPRESENTAZIONE DI UN NUMERO INTERO NEGATIVO

Ci sono tre diverse possibili rappresentazioni:

- Complemento a 2

Se il numero è positivo, si converte in binario usando il metodo standard.

Se il numero è negativo, si converte in binario il modulo del numero, si passa al CA1, poi si somma +1.

Somma di numeri binari

ADDENDO #1	ADDENDO #2	RIPORTO	SOMMA
0	0	0	**0**
0	1	0	**1**
1	0	0	**1**
1	1	1	**0**

Serve un bit in più!

Risolve il problema dello ZERO: adesso ha un'unica rappresentazione 0000 0000

RAPPRESENTAZIONE DI UN NUMERO INTERO NEGATIVO

Ci sono tre diverse possibili rappresentazioni:

- Complemento a 2

ESEMPI0 su 4 bit: Trovare la rappresentazione CA2 del numero -7_{10}

$$\overbrace{111}^{7}{}_2$$

RAPPRESENTAZIONE DI UN NUMERO INTERO NEGATIVO

Ci sono tre diverse possibili rappresentazioni:

- Complemento a 2

ESEMPI0 su 4 bit: Trovare la rappresentazione CA2 del numero -7_{10}

$$\overbrace{111_2}^{7} \xrightarrow{4bit} \overbrace{0111_{CA1}}^{+7}$$

RAPPRESENTAZIONE DI UN NUMERO INTERO NEGATIVO

Ci sono tre diverse possibili rappresentazioni:

- Complemento a 2

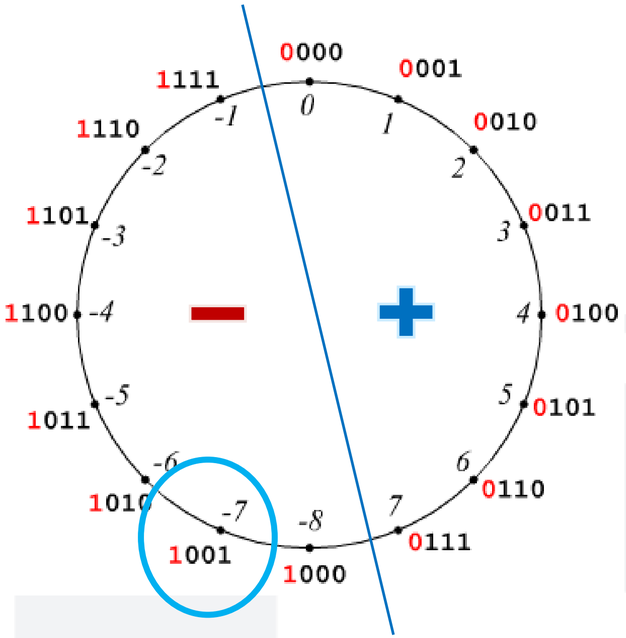
ESEMPI0 su 4 bit: Trovare la rappresentazione CA2 del numero -7_{10}

$$\overbrace{111_2}^{7} \xrightarrow{4bit} \overbrace{0111_{CA1}}^{+7} \xrightarrow{CA1} \overbrace{1000_{CA1}}^{-7}$$

RAPPRESENTAZIONE DI UN NUMERO INTERO NEGATIVO

Ci sono tre diverse possibili rappresentazioni:

- Complemento a 2



ESEMPI0 su 4 bit: Trovare la rappresentazione CA2 del numero -7_{10}

$$\overbrace{111_2}^{7} \xrightarrow{4bit} \overbrace{0111_{CA1}}^{+7} \xrightarrow{CA1} \overbrace{1000_{CA1}}^{-7} \xrightarrow{+1} \overbrace{1001_{CA2}}^{-7}$$

I numeri negativi iniziano comunque con 1 (MSB=1)

Range di rappresentazione con *n* bit è $[-2^{n-1}, 2^{n-1} -1]_{10}$

RAPPRESENTAZIONE DI UN NUMERO INTERO NEGATIVO

Ci sono tre diverse possibili rappresentazioni:

- Complemento a 2: COME SI CALCOLA? Esistono 3 diversi modi

(1) $CA2(X) = 2^n - X$ *(serve saper fare la sottrazione binaria: v. lezione di domani)*

(2) Passando per il CA1

- $CA1(X) = 2^n - 1 - X$
- $CA2(X) = CA1(X) + 1$

(3) Regola pratica. Algoritmo:

- Si parte da destra, si trascrivono tutti gli 0 fino ad incontrare il primo 1 e si trascrive anch'esso
- Si complementano a 1 tutti i bit restanti (0 → 1, 1 → 0)

RAPPRESENTAZIONE DI UN NUMERO INTERO NEGATIVO

Ci sono tre diverse possibili rappresentazioni:

- Complemento a 2

ESEMPI0 su 4 bit: Trovare la rappresentazione CA2 del numero -7_{10}

(1) $2^4 - 7 = 10000 - 111 = 1001_{CA2}$

(2) $\overbrace{111_2}^{7} \xrightarrow{4bit} \overbrace{0111_{CA1}}^{+7} \xrightarrow{CA1} \overbrace{1000_{CA1}}^{-7} \xrightarrow{+1} \overbrace{1001_{CA2}}^{-7}$

(3) $\overbrace{111_2}^{7} \xrightarrow{4bit} \overbrace{0111_{CA2}}^{+7} \xrightarrow{CA2} \overbrace{1001_{CA2}}^{-7}$

RAPPRESENTAZIONE DI UN NUMERO REALE

Come si fa??

RAPPRESENTAZIONE DI UN NUMERO REALE

I numeri reali possono essere rappresentati:

- in virgola fissa (*fixed point*)
- in virgola mobile (*floating point*)

VIRGOLA FISSA

Nel sistema di numerazione in virgola fissa:

- Si riserva **un numero fisso di bit** per la parte intera e per la parte frazionaria
- La **posizione della virgola** è quindi **implicita**
- La posizione della virgola è **uguale** per tutti i numeri

VIRGOLA FISSA (UNSIGNED)

Quando devono essere rappresentati numeri positivi e lo zero (**unsigned**).

Assumiamo di avere **N bit** a disposizione.

- L<N bit per rappresentare la parte intera del numero
- D=N-L bit per rappresentare la parte decimale del numero

L-1	L-2	…	0	-1	-2	…	-D
PARTE INTERA				PARTE FRAZIONARIA			

VIRGOLA FISSA (UNSIGNED)

Quando devono essere rappresentati numeri positivi e lo zero (**unsigned**).

Assumiamo di avere **N bit** a disposizione.

- L<N bit per rappresentare la parte intera del numero
- D=N-L bit per rappresentare la parte decimale del numero

Non c'è il bit di segno, quindi si possono rappresentare solo numeri dallo zero in su

L'intervallo di numeri interi rappresentabile è quindi: $[0, 2^L - 1]$

Mentre quello della parte decimale è: $[0, 2^D - 1]$

VIRGOLA FISSA (UNSIGNED)

Esempi di applicazione

Grafica digitale → pixel (es. livelli di colore 0-255 per immagini a 8 bit).

Misurazioni che non possono essere negative → distanza, velocità (es. GPS, sensori di movimento).

Tempo e frequenze → durata di eventi, campionamento di segnali.

Dati finanziari e contatori → somme di denaro, conteggi.

VIRGOLA FISSA (SIGNED)

Quando devono essere rappresentati numeri positivi e negativi (**signed**).

Assumiamo di avere **N bit** a disposizione.

- 1 bit per il segno
- L<(N-1) bit per rappresentare la parte intera del numero
- D=N-1-L bit per rappresentare la parte decimale del numero

+/-	L-1	L-2	…	0	-1	-2	…	-D
SEGNO	PARTE INTERA				PARTE FRAZIONARIA			

L'intervallo di numeri interi rappresentabile è quindi: $[-2^{L-1}-1, 2^{L-1}-1]$

Mentre quello della parte decimale è: $[0, 2^{D}-1]$

NOTA: Di solito i numeri negativi sono rappresentati in *complemento a due* (v. slide successive)

VIRGOLA FISSA (SIGNED)

Supponiamo di poter rappresentare i numeri reali con N = 8 bit.

Possiamo dedicare, alternativamente:

- 1 bit al segno, 3 bit alla parte intera, 4 bit alla parte decimale
- 1 bit al segno, 5 bit alla parte intera, 2 bit alla parte decimale

Come scegliere??

Dipende se si ha maggiore **necessità di precisione numerica** sulla parte intera o su quella decimale. Se si vuole rappresentare *numeri più grandi*, bisogna dare più spazio (più cifre) alla parte intera. Se invece si necessità di *maggior precisione decimale*, allora si dedicheranno più cifre alla parte decimale.

VIRGOLA FISSA (SIGNED)

Esempi di applicazione

Audio digitale (formati PCM a 16 o 24 bit) → i segnali sonori variano tra valori negativi e positivi.

Sensori e misurazioni fisiche → temperatura, tensione, corrente (valori positivi e negativi).

Elaborazione matematica → calcoli con numeri reali dove sono necessari anche valori negativi.

CONVERSIONE BINARIO FRAZIONARIO --> DECIMALE

X.YYYY è un numero frazionario in virgola fissa *unsigned* in base 2.
Convertirlo in base 10.

Regola: si converte esattamente come visto per i numeri interi.

Esempio:

$$\mathbf{N} = 101.01_2 = 1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0 + 0\cdot 2^{-1} + 1\cdot 2^{-2} = 4 + 0 + 1 + 0 + 0.25 = 5.25_{10}$$

Nota: lo stesso vale se il numero di partenza fosse in base 16.

CONVERSIONE DECIMALE --> BINARIO

X.YYYY è un numero frazionario *positivo* in base 10.

Convertirlo in base 2 in virgola fissa *unsigned*.

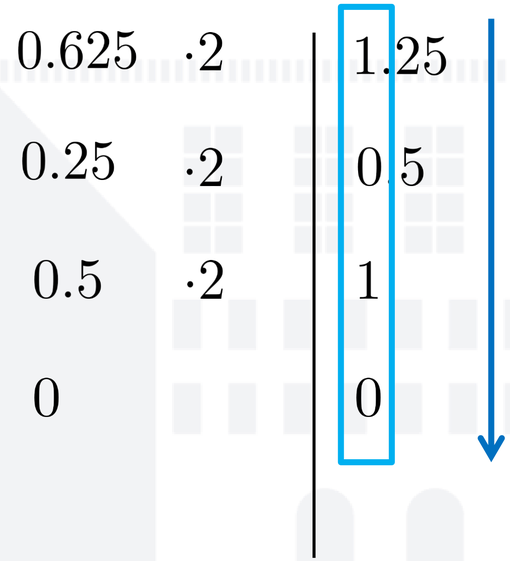
Algoritmo:

1. Convertire la parte intera in base 2
2. Si considera solo la parte decimale e la si moltiplica per 2. Si tiene da parte la cifra intera del risultato (che sarà 0 oppure 1)
3. Si ripete 2. fino a che il resto è 0, oppure si trova pattern ripetuto, oppure si esauriscono le cifre a disposizione della parte decimale

CONVERSIONE DECIMALE --> BINARIO

Esempio: $\mathbf{N} = 3.625_{10}$

$$3_{10} = 11_2$$



$$\mathbf{N} = (3.625)_{10} = (11.101)_2$$

CONVERSIONE DECIMALE --> BINARIO

Attenzione: alcuni numeri decimali potrebbero diventare binari periodici.

Esempio: $\mathbf{N} = 9.1_{10}$ $9_{10} = 1001_2$ $\mathbf{N} = (9.1)_{10} = (1001.000110011..)_2$

$$
\begin{array}{lll|l}
0.1 & \cdot 2 & & 0.2 \\
0.2 & \cdot 2 & & 0.4 \\
0.4 & \cdot 2 & & 0.8 \\
0.8 & \cdot 2 & & 1.6 \\
0.6 & \cdot 2 & & 1.2 \\
0.2 & \cdot 2 & & 0.4 \\
\vdots & \vdots & & \vdots
\end{array}
$$

CONVERSIONE DECIMALE --> BINARIO

Alcuni numeri decimali potrebbero non essere rappresentati con precisione.

Esempio: $\mathrm{N} = 9.6234_{10}$ $9_{10} = 1001_2$ $\mathrm{N} = (9.1)_{10} = (1001.10011111..)_2$

$$
\begin{array}{lll|l}
0.6234 & \cdot 2 & & 1.2468 \\
0.2468 & \cdot 2 & & 0.4936 \\
0.4936 & \cdot 2 & & 0.9872 \\
0.9872 & \cdot 2 & & 1.9744 \\
0.9744 & \cdot 2 & & 1.9488 \\
0.9488 & \cdot 2 & & 1.8976 \\
0.8976 & \cdot 2 & & 1.7952 \\
0.7952 & \cdot 2 & & 1.5904 \\
\vdots & \vdots & & \vdots
\end{array}
$$

Se abbiamo a disposizione 8 bit per la parte decimale, **allora la rappresentazione è precisa**. **Altrimenti**, la rappresentazione non è precisa (non si hanno sufficienti cifre per rappresentare la parte decimale).

VIRGOLA FISSA

Svantaggi

- Rigidità nell'assegnazione dello spazio (precisione) per parte decimale
- Trade-off a-priori tra precisione su numeri piccoli e codifica di numeri grandi
- Spreco di bit a 0 quando il numero è intero

VIRGOLA MOBILE

La massa dell'elettrone è
0.00000000000000000000000000000091 Kg

La massa della terra è
5973600000000000000000000 kg

Requisiti ideali:

- Convenzione per rappresentazione univoca per numeri così diversi
- Costo limitato di rappresentazione nel sistema binario (significa memoria!)

VIRGOLA MOBILE

La massa dell'elettrone è
0.00000000000000000000000000000091 Kg

La massa della terra è
597360000000000000000000 kg

Passiamo alla notazione scientifica

$9.1 \cdot 10^{-31}$

$5.9736 \cdot 10^{24}$

VIRGOLA MOBILE

Floating point

[Standard IEEE 754]

segno

esponente

$$\mathcal{N} = (-1)^S \cdot 1.M \cdot B^{\pm E}$$

MANTISSA

BASE

NOTA. Per poter rappresentare sia esponenti positivi che negativi, si usa un ***bias*** (per evitare di dover memorizzare un altro bit di segno).

+/-	e	e	…	e	m	m	…	m
segno	esponente				mantissa			

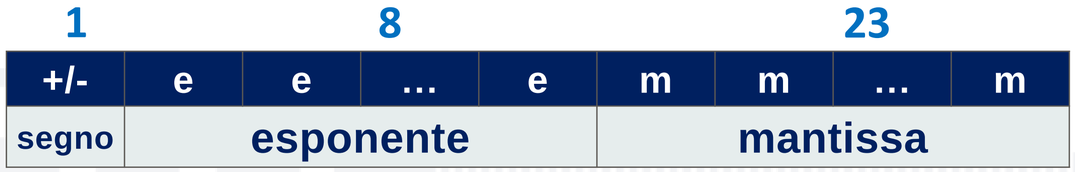
Più bit qui per range maggiore

Più bit qui per maggior precisione

VIRGOLA MOBILE: PRECISIONE

Floating point

Standard IEEE 754



32 bit in tutto → SINGOLA (o semplice) PRECISIONE

64 bit in tutto → DOPPIA PRECISIONE

Materiale per la lezione

- Appendici A e B Tanenbaum (numeri binari, virgola mobile, IEEE 754)

Prossima lezione: 7 ***marzo, h.11:00****,* ***aula 5B***