Informatica e Bioinformatica: Rappresentazione dell'Informazione

Alessandro Sperduti

26 marzo 2014

Sommario

- Il calcolatore è in grado di elaborare differenti tipi di informazione
  - numeri, caratteri, immagini, suoni, video
- Informazione = Dati + Interpretazione
  - nel calcolatore un dato è sempre una sequenza di bit
  - per rappresentare tipi di dati diversi deve cambiare l'interpretazione (il formato dei dati)
  - in pratica dobbiamo definire
    - una procedura di codifica (per determinare quale sequenza di bit corrisponde all'oggetto da rappresentare)
    - una procedura di decodifica (per determinare a cosa corrisponde una sequenza di bit)
- Le procedure di codifica/decodifica vengono eseguite dal calcolatore, quindi devono essere pensate in modo che i dati siano facilmente manipolabili dall'elaboratore (più che facilmente comprensibili dall'uomo)

Rappresentazione di Numeri

Vediamo come vengono rappresentati i numeri seguendo la seguente scaletta:

- Numeri interi positivi
- Numeri frazionari positivi
- Numeri interi
- Numeri reali

Rappresentazione dei Numeri: Sistema Decimale

- Noi comunemente utilizziamo il sistema decimale per rappresentare i numeri: ogni cifra di un numero può avere 10 valori diversi (da 0 a 9). Poichè ogni cifra può assumere 10 valori diversi, si dice che il numero è espresso in base 10.
- Una sequenza di cifre forma un numero secondo la seguente convenzione: $374 = 3 \cdot 10^2 + 7 \cdot 10^1 + 4 \cdot 10^0$
- Volendo essere formali: ogni cifra viene moltiplicata per la base elevata a k, dove k è la posizione della cifra contando da destra a partire da 0
- Se la base è maggiore di 10 si introducono delle lettere per le cifre rimanenti: ad esempio una base utilizzata in informatica è la base 16, le sue cifre sono:
  $0, 1, 2, 3, 4, 5, 6, 7, 8, 9$,A,B,C,D,E,F.

Rappresentazione dei Numeri: Sistema Binario

- Come abbiamo accennato, il calcolatore utilizza il bit per rappresentare l'informazione. Il bit può assumere 2 valori: 0 o 1. In questo caso si dice che un numero è espresso in base 2 (oppure in binario)
- Per determinare il valore di un numero binario positivo, si utilizza lo stesso algoritmo della slide precedente dove però la base è 2: *"ogni cifra viene moltiplicata per la base elevata a k, dove k è la posizione della cifra contando da destra a partire da* 0*"*

$$(1101)_2 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 8 + 4 + 1 = 13$$

- la cifra più a sinistra è detta la più significativa, quella più a destra la meno significativa (sono i termini che contribuiscono più e meno alla somma $8 + 4 + 1$).

Numeri Binari Positivi

- Il numero di configurazioni diverse di n bit è 2^n, per cui si riescono a rappresentare 2^n numeri diversi.
- Il numero più grande rappresentabile con n bit è $2^n - 1$ (perchè si inizia a contare da 0). Il calcolatore non può rappresentare infiniti numeri!

binario	decimale
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Da Base 10 a base 2

- Prima di vedere l'algoritmo per passare da base 10 alla base 2, vediamo altre due operazioni applicabili a numeri in qualsiasi base
- lshift (spostamento a sinistra delle cifre). Ad esempio:
  lshift | | 3 | 7 | 4 | = | 3 | 7 | 4 | 0 |
- lshift equivale a moltiplicare il numero per la base
- rshift (spostamento a destra delle cifre). Ad esempio:
  rshift | 3 | 7 | 4 | = | | 3 | 7 | | | 4 |
- rshift corrisponde a dividere il numero per la base (si ottiene sia il quoziente 37 che il resto 4)

Da Base 10 a base k

Trasformazione da base 10 a base k

1. Dividere il numero per k
2. tenete traccia del resto
3. se il quoziente è maggiore di 0 ripetere il passo 1 con il quoziente
4. scrivere i resti nell'ordine inverso rispetto al quale sono stati ottenuti

Esempio: trasformiamo il numero 43 in base 2:

numero	quoziente	resto
43/2	21	1
21/2	10	1
10/2	5	0
5/2	2	1
2/2	1	0
1/2	0	1

leggendo i resti dal basso all'alto (al contrario di come si sono ottenuti) si ricava 101011 (ritrasformatelo in base 10 per verificare). Quindi $(43)_{10} = (101011)_2$

Da Base 10 a base k: Esempio

L'algoritmo funziona per qualsiasi base k di destinazione:

Esempio: $(124)_{10} = (??)_2$:

numero	quoziente	resto
$124/2$	62	0
$62/2$	31	0
$31/2$	15	1
$15/2$	7	1
$7/2$	3	1
$3/2$	1	1
$1/2$	0	1

Verifichiamo il risultato:

$(1111100)_2 = 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 +$
$+1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 =$
$= 64 + 32 + 16 + 8 + 4 = 124$

Esempio: $(124)_{10} = (??)_5$:

numero	quoziente	resto
$124/5$	24	4
$24/5$	4	4
$4/5$	0	4

Verifichiamo il risultato:

$(444)_5 = 4 \cdot 5^2 + 4 \cdot 5^1 + 4 \cdot 5^0 =$
$= 4 \cdot 25 + 4 \cdot 5 + 4 = 124$

Rappresentazione di Numeri

- Numeri interi positivi
- Numeri frazionari positivi
- Numeri interi
- Numeri reali

Numeri reali positivi

- Abbiamo visto come decodificare un numero intero positivo da una base k:
  $(1101)_2 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 8 + 4 + 1 = (13)_{10}$
- E se il numero avesse cifre dopo la virgola?
- Si procede ancora come facciamo per la base 10:
  - $3.27 = 3 \cdot 10^0 + 2 \cdot 10^{-1} + 7 \cdot 10^{-2}$
- $(0.011)_2 = 0 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3} = \frac{1}{2^2} + \frac{1}{2^3} =$
  $= 0.25 + 0.125 = 0.375$
- Quindi $(1101.011)_2 = (13.375)_{10}$

Codifica in binario di numeri reali positivi

Algoritmo:

1. Si moltiplica il numero per 2
2. La parte intera del numero è il prossimo valore nella nuova base
3. Si ripete il punto 1 con la parte decimale del numero finché
   - tale parte decimale non è 0
   - nei casi in cui non si arriva mai a 0, ci si ferma quando si sono utilizzati tutti i bit a disposizione per rappresentare il numero.

Esempio 1: $(0.21875)_{10} = (??)_2$:

numero	parte intera	parte decimale
$0.21875 \cdot 2 = 0.4375$	0	0.4375
$0.4375 \cdot 2 = 0.875$	0	0.875
$0.875 \cdot 2 = 1.75$	1	0.75
$0.75 \cdot 2 = 1.5$	1	0.5
$0.5 \cdot 2 = 1.0$	1	0

$(0.21875)_{10} = (0.00111)_2$.

Esempio 2: $(0.9)_{10} = (??)_2$:

numero	parte intera	parte dec.
$0.90 \cdot 2 = 1.80$	1	0.80
$0.80 \cdot 2 = 1.60$	1	0.60
$0.60 \cdot 2 = 1.20$	1	0.20
$0.20 \cdot 2 = 0.40$	0	0.40
$0.40 \cdot 2 = 0.80$	0	0.80

$(0.9)_{10} = (0.\underbrace{11100}\underbrace{11100}\ldots)_2$:

Rappresentazione di numeri all'interno del calcolatore

Rappresentazione di Numeri

- Numeri interi positivi
- Numeri frazionari positivi
- Numeri interi
- Numeri reali

Numeri Interi Positivi

- I numeri interi positivi sono rappresentati all'interno dell'elaboratore utilizzando un multiplo del byte (generalmente 4 o 8 byte)
- Le funzioni di codifica/decodifica sono quelle che abbiamo visto nelle slide precedenti
- Se l'intero si rappresenta con un numero di cifre minore, vengono aggiunti zeri nelle cifre più significative
  - Esempio: su un byte $(12)_{10} = (00001100)_2$

Interi con segno

Rappresentazione in Modulo e Segno

- Il bit più significativo rappresenta il segno:
  - 0 = numero positivo, 1 = numero negativo
- se si utilizzano n bit, si riescono a rappresentare tutti i numeri x. $-2^{n-1} - 1 \leq x \leq 2^{n-1} - 1$.
  - Ad esempio con 4 bit si rappresentano i numeri da -7 a 7.

0000	0	1000	0
0001	1	1001	-1
0010	2	1010	-2
0011	3	1011	-3
0100	4	1100	-4
0101	5	1101	-5
0110	6	1110	-6
0111	7	1111	-7

Modulo e Segno

La rappresentazione in modulo e segno è facile da calcolare per l'uomo, ma

- ha 2 rappresentazioni per lo 0 (spreco!)
- il metodo di somma che abbiamo visto non è utilizzabile:
  - $1001 + 0001 = 1010$ ovvero $-1 + 1 = -2$!

0000	0	1000	0
0001	1	1001	-1
0010	2	1010	-2
0011	3	1011	-3
0100	4	1100	-4
0101	5	1101	-5
0110	6	1110	-6
0111	7	1111	-7

Complemento a 2 con n bit a disposizione

- I numeri positivi sono rappresentati in modo "standard" (come nella notazione modulo e segno), utilizzando n bit
  - Esempio: $n = 4$; $(3)_{10} = (0011)$
- I numeri negativi sono rappresentati "in complemento a 2", ovvero si somma 2^n al numero e poi rappresenta in modo "standard"
- Esempio: $n = 4$;
  $(-3)_{10} \rightarrow 2^4 - 3 = 16 - 3 = (13)_{10} = (1101)$

0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	-8
1001	-7
1010	-6
1011	-5
1100	-4
1101	-3
1110	-2
1111	-1

Complemento a 2 con n bit a disposizione

- Il bit più significativo indica ancora il segno (0=positivo, 1=negativo)
- I numeri x rappresentabili con n bit sono nell'intervallo $-2^{n-1} \leq x \leq 2^{n-1} - 1$, rispetto alla rappresentazione modulo e segno è cambiata la disposizione dei negativi e c'è un numero negativo in più
- 0 è considerato positivo: $n = 4; (0)_{10} = (0000)_2$
  - in realtà non è una scelta obbligata, perchè se fosse negativo non sarebbe rappresentabile (provate!)

0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	-8
1001	-7
1010	-6
1011	-5
1100	-4
1101	-3
1110	-2
1111	-1

Complemento a 2

Ho a disposizione n bit per rappresentare un numero decimale x in complemento a 2 (supponiamo che il risultato della codifica sia y)

- Controllo se il numero è rappresentabile con n bit ovvero se $-2^{n-1} \leq x \leq 2^{n-1} - 1$ (per y basta che guardi da quanti bit è formato)
- Se x e y sono positivi e rappresentabili con n bit:
  - sia la codifica (da x a y) che la decodifica (da y a x) si effettuano in modo "standard" (con le divisioni per 2 oppure moltiplicando le cifre y per potenze di 2).
    - Esempio 1 (codifica): $n = 4$; $(6)_{10}$ è rappresentabile perchè $-2^3 \leq 6 \leq 2^3 - 1$ e $(6)_{10} = (0110)_2$
    - Esempio 2 (codifica): $(18)_{10}$ non è rappresentabile perchè $2^3 - 1 \leq 18$.
    - Esempio 3 (decodifica): $(0101)_2 = 2^2 + 2^0 = 5$

Complemento a 2

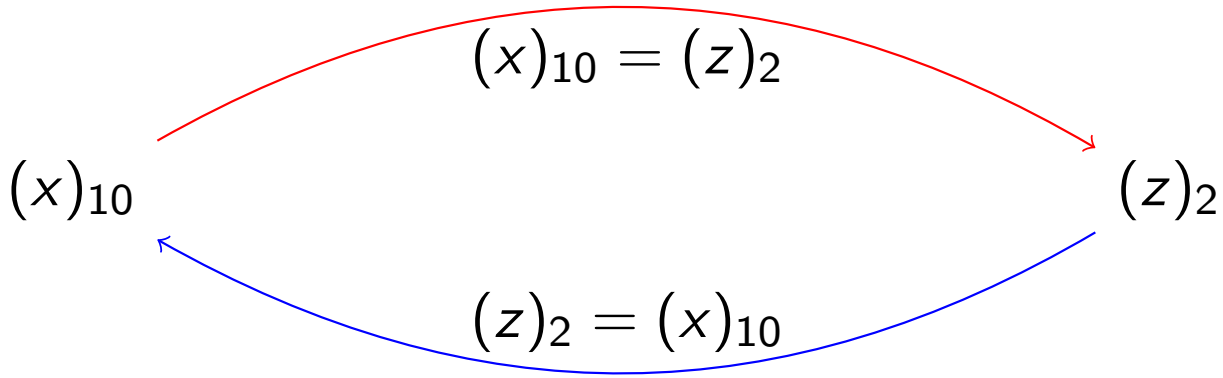
Ho a disposizione n bit per rappresentare un numero decimale x in complemento a 2 (supponiamo che il risultato della codifica sia y)

- Controllo se il numero è rappresentabile con n bit ovvero se $-2^{n-1} \leq x \leq 2^{n-1} - 1$ (per y basta che guardi da quanti bit è formato)
- se x e y sono negativi e rappresentabili con n bit:
  - la codifica (da x a y) si effettua applicando la codifica per i numeri positivi a $2^n + x$
    - Esempio 1: $n = 4$; $(-4)_{10}$ è rappresentabile, calcolo allora $(-4 + 16)_{10} = (12)_{10} = (1100)_2$
    - Esempio 2: $n = 4$; $(-11)_{10} < -2^3 - 1$, quindi non è rappresentabile in complemento a 2
  - la decodifica (da y a x) si effettua applicando la decodifica per i numeri positivi e poi sottraendo 2^n al risultato
    - Esempio 2: $n = 4$; $(1001)_2$, decodifico $(1001)_2 = 9$ e poi calcolo $9 - 2^4 = 9 - 16 = -7$

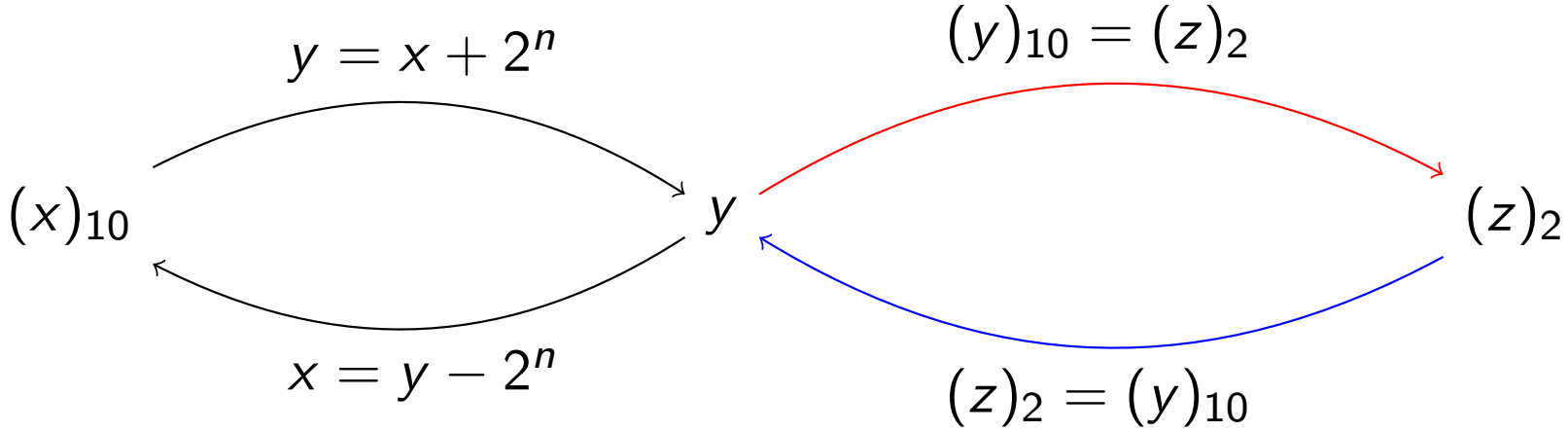
Complemento a 2: codifica/decodifica graficamente

Legenda: n bit, z rappresentazione in complemento a 2 di x (rappresentabile),frecce rosse codifica, blu decodifica.

- Se il numero è positivo:



- Se il numero è negativo:



Esempi

Supponendo di avere a disposizione 4 bit, determinare la rappresentazione in complemento a 2 di -6:

- $-8 \leq -6 \leq 7$ per cui è rappresentabile.
  Calcoliamo $(16 - 6)_{10} = (??)_2$

numero	quoziente	resto
$10/2$	5	0
$5/2$	2	1
$2/2$	1	0
$1/2$	0	1

  La rappresentazione in complemento a 2 di -6 è 1010.

Esempi

- Quale numero corrisponde al seguente numero in complemento a 2: 010011?
  - Il numero è definito su 6 bit, per cui assumiamo $n = 6$ (il numero è ovviamente rappresentabile perchè abbiamo già la sua rappresentazione).
    Il numero è positivo per cui possiamo decodificarlo direttamente
    $010011 = 2^4 + 2^1 + 2^0 = 16 + 2 + 1 = 19$
- Quale numero corrisponde al seguente numero in complemento a 2 definito su 5 bit: 10011?
  - Il numero è negativo. Intanto decodifichiamolo
    $10011 = 2^4 + 2^1 + 2^0 = 16 + 2 + 1 = 19$
    Poichè il numero è negativo dobbiamo sottrarre al risultato della decodifica 2^5: $19 - 32 = -13$
    Il numero 10011 in complemento a 2 su 5 bit corrisponde a -13.

Complemento a 2: Considerazioni

La rappresentazione in complemento a 2 è più complicata di quella in modulo e segno ma

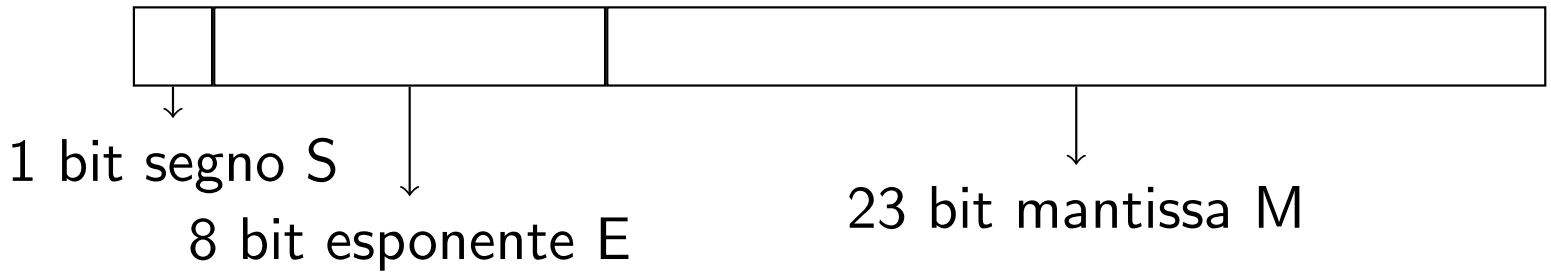
- ha una sola rappresentazione per lo 0
- il metodo di somma che abbiamo visto è utilizzabile:
  - $1001 + 0001 = 1010$ ovvero $-7 + 1 = -6$!
- in generale le operazioni aritmetiche risultano più naturali da eseguire al calcolatore

Rappresentazione di Numeri

- Numeri interi positivi
- Numeri frazionari positivi
- Numeri interi
- Numeri reali

Numeri Reali

- I numeri reali utilizzano la rappresentazione in virgola mobile
- Si basa sulla notazione scientifica $1.40 \cdot 10^2 = 140$ (notate che c'è solo una cifra intera, ovvero la notazione è normalizzata)
- Lo standard IEEE 754 prevede 3 tipi di numeri in virgola mobile:
  - singola precisione (32 bit)
  - doppia precisione (64 bit)
  - quadrupla precisione (128 bit)
- i numeri a singola precisione hanno il seguente formato:



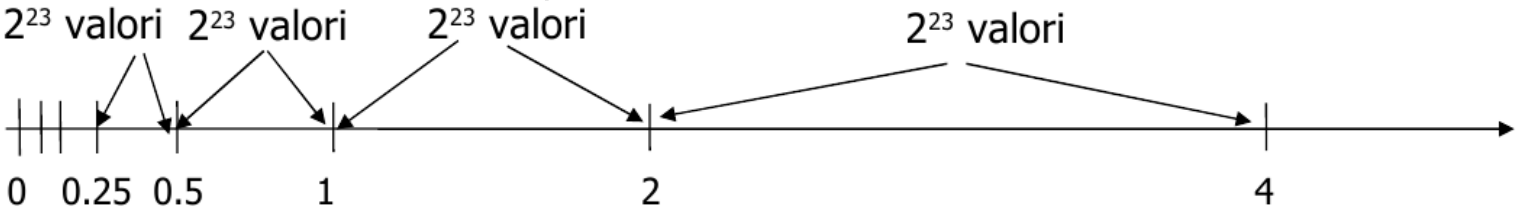
- tralasciando le sequenze E=00000000, E=11111111 che sono casi particolari, il formato è $(-1)^S \cdot 1.M \cdot 2^{E-127}$

Numeri Reali

- Il formato IEEE 754 è $(-1)^S \cdot 1.M \cdot 2^{E-127}$
  - $(-1)^S$ indica il segno $(-1)^0 = +1$, $(-1)^1 = -1$
  - $1.M$, la mantissa indica il numero vero e proprio in forma normalizzata ()
  - 2^{E-127} indica dove mettere la virgola (moltiplicare/dividere per 2 un numero binario significare spostare a destra/sinistra la virgola di una posizione)
  - l'esponente E è un intero positivo (tra 0 e 255), sottraendogli 127 si ottengono metà esponenti positivi e metà negativi
  - Si utilizza questa strana rappresentazione dell'esponente perchè rende più semplice il confronto tra numeri reali: a parte il segno è sufficiente un confronto lessicografico (bit a bit) per stabilire il maggiore
    - 0 01111000 10010100010110110110110
    - 0 01111101 01010000100101101011110
    - il primo numero è minore del secondo

Numeri Reali a Singola Precisione

- Il numero più grande rappresentabile è (circa) $6.81 \cdot 10^{38}$
- Il numero positivo più piccolo rappresentabile è (circa) $1.4 \cdot 10^{-45}$
- in totale si riescono a rappresentare 2^{32} numeri distinti (metà positivi, metà negativi)



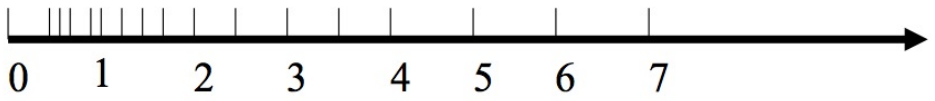
- I numeri rappresentabili non sono distribuiti uniformemente
  - all'aumentare dell'esponente aumenta la dimensione dell'intervallo, però ogni intervallo contiene 2^{23} numeri, per cui la precisione di un numero reale è maggiore più si è vicini allo zero

Esempio di distribuzione non uniforme

Numeri rappresentabili con 1 bit di segno (si mostrano solo i numeri positivi), 2 bit di esponente, 2 bit di mantissa

$$\text{numero} = (1 + a \cdot 2^{-1} + b \cdot 2^{-2})2^{f \cdot 2^1 + e \cdot 2^0 - 1}$$

f	e	a	b		numero
0	0	0	0		0.5
0	0	0	1		0.625
0	0	1	0		0.75
0	0	1	1		0.875
0	1	0	0		1
0	1	0	1		1.25
0	1	1	0		1.5
0	1	1	1		1.75
1	0	0	0		2
1	0	0	1		2.5
1	0	1	0		3
1	0	0	1		3.5
1	1	0	0		4
1	1	0	1		5
1	1	1	0		6
1	1	1	1		7



Numeri Reali: esempio di rappresentazione

Fornire la rappresentazione in virgola mobile normalizzata del valore 10.543 avendo a disposizione 8 bit per l'esponente e 8 per la mantissa.

1. Calcolare la rappresentazione binaria di $(10)_{10} = (1010)_2$:

numero	quoziente	resto
$10/2$	5	0
$5/2$	2	1
$2/2$	1	0
$1/2$	0	1

2. Calcolare la rappresentazione binaria di $(0.543)_{10} = 10001011$

numero	parte intera	parte decimale
$0.543 \cdot 2 = 1.086$	1	0.086
$0.086 \cdot 2 = 0.172$	0	0.172
$0.172 \cdot 2 = 0.344$	0	0.344
$0.344 \cdot 2 = 0.688$	0	0.688
$0.688 \cdot 2 = 1.376$	1	0.376
...	...	...

Numeri Reali: esempio di rappresentazione

Fornire la rappresentazione in virgola mobile normalizzata del valore 10.543 avendo a disposizione 8 bit per l'esponente e 8 per la mantissa.

1. Calcolare la rappresentazione binaria di $(10)_{10} = (1010)_2$:
2. Calcolare la rappresentazione binaria di $(0.543)_{10} = (10001011)_2$
3. Normalizzare il numero ottenuto: $1.01010001011 \cdot 2^3 = 1010.10001011$
4. Rappresentare l'esponente: $3 + 127$ (127 viene sommanto all'esponente perchè stiamo codificando il numero, $E - 127$ di qualche slide precedente fa riferimento alla fase di decodifica. $(130)_{10} = (10000010)_2$
5. Infine la rappresentazione del numero è 0 10000010 01010001

Numeri Reali: Considerazioni

- Indipendentemente dalla codifica scelta, è probabile che un numero reale non ammetta una rappresentazione finita, quindi dovrà essere codificato in maniera approssimata
- La precisione della rappresentazione di un numero reale è una misura di quanto essa corrisponda al numero che deve essere rappresentato
- La maggior parte degli elaboratori non possiede circuiti in grado di eseguire direttamente tutte le operazioni:
  - Ad esempio la moltiplicazione si realizza per mezzo di una successione di addizioni e di shift
  - Le operazioni più semplici sono eseguite direttamente da appositi circuiti (in hardware); le operazioni più complesse sono spesso realizzate mediante l'esecuzione di successioni di operazioni più semplici, sotto il controllo di piccoli programmi

Rappresentazione di Caratteri

- Oltre ai numeri, molte applicazioni elaborano caratteri (simboli)
- Per poter scambiare dati in modo corretto, è necessario definire una codifica da carattere a numero standard
- Lo standard di codifica pi diffuso è il codice ASCII, per American Standard Code for Information Interchange
- Definisce una tabella di corrispondenza fra ciascun simbolo (carattere minuscolo, maiuscolo, cifre) e un codice a 7 bit (128 caratteri)

Codifica ASCII

Tabella ASCII

	0	1	2	3	4	5	6	7
0	NUL	DLE	space	0	@	P	`	p
1	SOH	DC1 XON	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3 XOFF	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	del

Estensioni

- La tabella ASCII (datata 1961) è limitata: ad esempio non permette di rappresentare caratteri arabi o asiatici
- La tabella ASCII estesa utilizza 8 bit e permette di rappresentare caratteri come à, è, . . .
- La tabella ASCII non è standard. Ad esempio ISO 8859-1 contiene i caratteri latini di maggior uso (coincide con ASCII per i primi 127 valori)
- UNICODE (UTF-8 e UTF-16): standard proposto a 8 e 16 bit (65.536 caratteri)
  - UTF-8 è usato per le e-mail