

Codifica dell'informazione numerica



Matteo Re, matteo.re@unimi.it

Nicola Basilico, nicola.basilico@unimi.it

Informazioni (edizione 2017-2018)

Turno A (cognomi A - F)

- **Nicola Basilico**, dip. via Comelico, ufficio S242, ricevimento su appuntamento: `nicola.basilico@unimi.it`
- Sito del corso: <http://basilico.di.unimi.it>

Turno B (cognomi G - Z)

- **Matteo Re**, dip. via Comelico, ufficio S209, ricevimento su appuntamento: `matteo.re@unimi.it`
- Sito del corso: <https://homes.di.unimi.it/re/arch1-lab-2017-2018.html>

Rappresentazione dei numeri: notazione posizionale

- Base B : numero di simboli usati per rappresentare i numeri nel sistema posizionale.
 - $B = 10$, simboli $\{0, 1, \dots, 9\}$
 - $B = 16$, simboli $\{0, 1, \dots, 9, A, B, C, D, E, F\}$
- Notazione posizionale:
 - ogni simbolo ha una posizione, denotata con un intero i
 - a ogni posizione i si associa un peso p_i
 - al simbolo in posizione i viene associato il valore dato da:

VALORE DEL SINGOLO SIMBOLO $\times p_i$

- di solito $p_i = B^i$

Rappresentazione dei numeri: notazione posizionale

Esempio: $(147)_{10}$

cifre	1	4	7
posizioni i	2	1	0
pesi p_i	10^2	10^1	10^0

$$(147)_{10} = 1 \cdot 10^2 + 4 \cdot 10^1 + 7 \cdot 10^0$$

In generale, dato un N intero non negativo scritto con n cifre come $(c_{n-1}c_{n-2} \dots c_0)_B$, il valore rappresentato è:

$$c_{n-1} \cdot B^{n-1} + c_{n-2} \cdot B^{n-2} + \dots + c_0 \cdot B^0 = \sum_{i=0}^{n-1} c_i \cdot B^i$$

Da base B a base decimale

- PROBLEMA: Scrivere in base $B = 10$ un numero dato in base $B \neq 10$ ($B = 2$ o $B = 16$)
- SOLUZIONE: basta applicare il metodo **polinomiale**

Esempio 1:

$$(1010)_2 = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 8 + 2 = (10)_{10}$$

Esempio 2:

$$(3AC)_{16} = 3 \cdot 16^2 + 10 \cdot 16^1 + 12 \cdot 16^0 = 3 \cdot 256 + 160 + 12 = (940)_{10}$$

Da base decimale a base B

- PROBLEMA: Scrivere in base $B \neq 10$ ($B = 2$ o $B = 16$) un numero dato in base $B = 10$
- SOLUZIONE: si applica il metodo **iterativo** (divisioni)

Procedimento

Abbiamo un numero $(N)_{10}$ da convertire nella base B :

1. dividere N per B (con una divisione intera);
2. il resto della divisione diventa la prima cifra meno significativa che resta da calcolare del numero in base B ;
3. se il quoziente è 0 abbiamo finito;
4. se il quoziente è diverso da zero si torna al passo **1** considerando il quoziente come dividendo;

Da base decimale a base B

Esempio 1: $(13)_{10} = (?)_2$

$13 : 2 = 6$	resto = 1	LSD		$(13)_{10} = (1101)_2$
$6 : 2 = 3$	resto = 0			
$3 : 2 = 1$	resto = 1			
$1 : 2 = 0$	resto = 1	MSD		

Esempio 2: $(4021)_{10} = (?)_{16}$

$4021 : 16 = 251$	resto = 5	5	LSD		$(4021)_{10} = (FB5)_{16}$
$251 : 16 = 15$	resto = 11	B			
$15 : 16 = 0$	resto = 15	F	MSD		

Da base $B = 2$ a base $B = 16$ e vice versa

- PROBLEMA: Scrivere in base $B = 2$ ($B = 16$) un numero dato in base $B = 16$ ($B = 2$)
- SOLUZIONE: **raggruppamento** dei simboli e **lookup table**

Esempio 1: $(111001)_2 = (?)_{16}$

$16 = 2^4$, raggruppo i bit partendo da destra a gruppi di 4:

0011 | 1001

Con 4 bit ho 2^4 valori, uno per ogni simbolo della base $B = 16$.

0	0000	4	0100	8	1000	C	1100
1	0001	5	0101	9	1001	D	1101
2	0010	6	0110	A	1010	E	1110
3	0011	7	0111	B	1011	F	1111

Ispezionando la tabella:

$$(0011 | 1001)_2 = (39)_{16}$$

Da base $B = 2$ a base $B = 16$ e vice versa

Esempio 2: $(EA0)_{16} = (?)_2$

$E | A | 0$

0	0000	4	0100	8	1000	C	1100
1	0001	5	0101	9	1001	D	1101
2	0010	6	0110	A	1010	E	1110
3	0011	7	0111	B	1011	F	1111

$$(E | A | 0)_{16} = (1110 1010 0000)_2$$

Somma di interi non negativi

Come la somma in decimale, ricordare che quando sommiamo singole cifre binarie:

$$\begin{array}{ll} 0 + 0 = 0 & \text{riporto} = 0 \\ 0 + 1 = 1 & \text{riporto} = 0 \\ 1 + 0 = 1 & \text{riporto} = 0 \\ 1 + 1 = 0 & \text{riporto} = 1 \\ 1 + 1 + 1 = 1 & \text{riporto} = 1 \end{array}$$

Esempio 1: $(1101)_2 + (111)_2$

$$\begin{array}{rcccccc} \text{riporto} & 1 & 1 & 1 & 1 & & \\ & & 1 & 1 & 0 & 1 & + \\ & & & 1 & 1 & 1 & = \\ \hline \text{somma} & 1 & 0 & 1 & 0 & 0 & \end{array}$$

Somma di interi non negativi

Esempio 2: $(111010)_2 + (1101110010)_2$

riporto				1	1	1			1			
						1	1	1	0	1	0	+
	1	1	0	1	1	1	0	0	1	0	=	
somma	1	1	1	0	1	0	1	1	0	0		

Interi: complemento a 2

- Rappresentazione in **complemento a 2** (C2)
 - **Dati n bit**, un numero positivo N è rappresentato in modo standard (come abbiamo visto per i non negativi)
 - $-N$, invece si rappresenta come $2^n - N$
- Metodo operativo per rappresentare $-N$:
 - Rappresentare il modulo N in modo standard
 - Complementare a 1 tutti i bit ($1 \leftarrow 0, 0 \leftarrow 1$)
 - Sommare 1

Interi: complemento a 2

- In complemento a 2, con n bit, possiamo rappresentare gli interi nell'intervallo: $[-2^{n-1}; 2^{n-1} - 1]$
- Esempio: con 3 bit rappresentiamo i numeri in $[-4; 3]$

$N_{(10)}$	$N_{(C2)}$	$N_{(10)}$	$N_{(C2)}$
-4	100	0	000
-3	101	+1	001
-2	110	+2	010
-1	111	+3	011

- Il primo bit indica ancora il segno
- Lo zero ha una sola codifica

Interi: complemento a 2

Esempio 2: $(9)_{10} = (?)_{C2}$

- Converto $(9)_{10}$ in base 2: $(1001)_2$
bastano 4 bit? No! $\rightarrow (01001)_{C2}$
- Se non avessi aggiunto il *leading zero*? $(1001)_{C2} = -7$

Esempio 3: $(-6)_{10} = (1010)_{C2}$, notazione *modulo e segno*?

- Il C2 di un numero negativo in C2 è il modulo del numero stesso
- Complemento a 1: 0101, sommo 1: $(0110)_{C2} = -(110)_2$

Somma di interi

- Rappresentare i numeri in C2
- Effettuare la somma in modo standard
- Non considerare l'eventuale riporto oltre il bit di segno

Esempio 1: $(60)_{10} - (54)_{10} = (60)_{10} + (-54)_{10}$

riporto	1	1	1	1					
		0	1	1	1	1	0	0	+
		1	0	0	1	0	1	0	=
<hr/>									
somma	(1)	0	0	0	0	1	1	0	

Somma di interi: overflow

- Sommiamo due interi in C2 rappresentati con n bit, quindi appartenenti a $[-2^{n-1}; 2^{n-1} - 1]$
- Può succedere che il risultato cada al di fuori dell'intervallo
- In altre parole: n bit non bastano per rappresentare il risultato in C2 → **OVERFLOW**
- Come riconoscerlo?
 - può succedere solo quando si sommano due operandi dello stesso segno: **se il segno del risultato è diverso da quello degli operandi** è avvenuto un overflow
 - gli ultimi due riporti sono diversi tra loro (01 o 10)

Somma di interi: esempio di overflow

Esempio: $(100)_{C2} + (101)_{C2}$

riporto	1	0			
		1	0	0	+
		1	0	1	=
<hr/>					
somma	(1)	0	0	1	

- Sto sommando su 3 bit -4 e -3 , il risultato sarebbe: $-7 < -2^{3-1}$, non rappresentabile in C2 su 3 bit.

Somma di interi: Esempio 1

Esempio 1: $(10100)_2 - (1011)_2$

- Devo rappresentare entrambi in numeri in C2 usando lo stesso numero di bit, me ne servono almeno $5 + 1$ (segno)
- $(10100)_2 \rightarrow (0\ 10100)_{C2}$
- $-(1011)_2 \rightarrow -(00\ 1011)_2$, complemento a 1 il modulo $\rightarrow (110100)$, sommo 1 $\rightarrow (110101)_{C2}$

riporto	1	1		1				
		0	1	0	1	0	0	+
		1	1	0	1	0	1	=
somma	(1)	0	0	1	0	0	1	

- Risultato: $(10100)_2 - (1011)_2 = +(1001)_2$

Somma di interi: Esempio 2

Esempio 2: $-(1101)_2 - (111)_2$

- Mi servono $4 + 1$ bit
- $-(1101)_2 \rightarrow -(0\ 1101)_2$, complemento a 1 il modulo $\rightarrow (10010)$, sommo 1 $\rightarrow (10011)_{C2}$
- $-(111)_2 \rightarrow -(00\ 111)_2$, complemento a 1 il modulo $\rightarrow (11000)$, sommo 1 $\rightarrow (11001)_{C2}$

riporto	1			1	1		
		1	0	0	1	1	+
		1	1	0	0	1	=
<hr/>							
somma	(1)	0	1	1	0	0	

- Overflow: sto sommando su **5** bit -13 e -7 , il risultato sarebbe: $-20 < -2^{5-1}$, non rappresentabile in C2 su 5 bit.

Somma di interi: Esempio 2

Esempio 2: $-(1101)_2 - (111)_2$

- Aggiungo un bit: $4 + 1 + 1$
- $-(1101)_2 \rightarrow -(00\ 1101)_2$, complemento a 1 il modulo $\rightarrow (110010)$, sommo 1 $\rightarrow (110011)_{C2}$
- $-(111)_2 \rightarrow -(000\ 111)_2$, complemento a 1 il modulo $\rightarrow (111000)$, sommo 1 $\rightarrow (111001)_{C2}$

riporto	1	1			1	1		
		1	1	0	0	1	1	+
		1	1	1	0	0	1	=
<hr/>								
somma	(1)	1	0	1	1	0	0	

- Risultato: $-(1101)_2 - (111)_2 = (101100)_{C2} = -(10100)_2$

Interi: complemento a 2

Come passare da C2 a base 10?

- Procedimento inverso:
 - Sottrarre 1
 - Complementare a 1
 - Convertire da binario a decimale e aggiungere il segno
- Metodo facilitato di verifica:
 - convertire in decimale con l'algoritmo standard assegnando al bit più significativo valore negativo
 - Esempio: $(10100)_{C2} = -1 \times 2^4 + 1 \times 2^2 = (-12)_{10}$

Numeri frazionari

- Notazione posizionale e metodo polinomiale si estendono facilmente:

$$\begin{aligned}(I.c_{-1}c_{-2}\dots c_{-m})_B &= \\ I + c_{-1} \cdot B^{-1} + c_{-2} \cdot B^{-2} + \dots + c_{-m} \cdot B^{-m} &= \\ I + \sum_{i=-m}^{-1} c_i \cdot B^i &\end{aligned}$$

- Esempio: $(0.587)_{10} = 5 \cdot 10^{-1} + 8 \cdot 10^{-2} + 7 \cdot 10^{-3}$
- Conversione da base 2 a base 10:
 $(0.1011)_2 = 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} + 1 \cdot 2^{-4} = (0.6875)_{10}$

Rappresentazione dei numeri frazionari

- Come convertire la parte frazionaria da base 10 a base 2? Metodo **iterativo** (moltiplicazioni)

Procedimento

Abbiamo un numero $(I.F)_{10}$ da convertire nella base $B = 2$:

1. la parte intera I si converte come abbiamo visto in precedenza;
 2. moltiplicare F per 2;
 3. la parte intera del risultato diventa la cifra **più** significativa (la prima che resta da calcolare) del numero in base 2;
 4. tornare a **2** considerando la parte frazionaria del risultato al posto di F
- Quando si finisce?
 - Soltanto i numeri del tipo $\frac{N}{2^z}$ possono essere rappresentati con un numero finito di cifre
 - Ci si ferma quando la parte frazionaria ottenuta è 0 o quando il numero di cifre calcolate costituisce un'approssimazione sufficiente

Rappresentazione dei numeri frazionari

Esempio: $(0.587)_{10} = (?)_2$

$0.587 \times 2 = 1.174$	parte intera = 1	MSD
$0.174 \times 2 = 0.348$	parte intera = 0	
$0.348 \times 2 = 0.696$	parte intera = 0	
$0.696 \times 2 = 1.392$	parte intera = 1	
$0.392 \times 2 = 0.784$	parte intera = 0	
$0.784 \times 2 = 1.568$	parte intera = 1	
...		

$$(0.587)_{10} = (0.100101\dots)_2$$

Approssimare i numeri reali

- Rappresentazione in **virgola fissa**
- La parte intera è rappresentata sempre su n bit mentre la parte frazionaria è rappresentata sempre su m bit
- La virgola può cadere in un'unica posizione: è implicita e quindi può essere omessa

Esempi (assumendo base $B = 2$ e notazione in C_2):

$$(+4.25)_{10} = (0100.0100)_{C_2}$$

$$(-4.25)_{10} = (1100.0100)_{C_2}$$

$$(+3.35)_{10} = (0011.0101)_{C_2}$$

$$(-3.35)_{10} = (1100.1011)_{C_2}$$

Approssimare i numeri reali

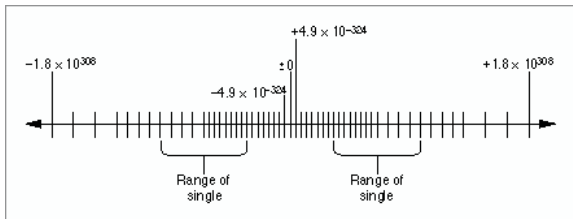
- Rappresentazione in **virgola mobile**
- Un numero razionale $(N)_B$ è espresso come:
$$N = (-1)^s \cdot m \cdot B^e$$
 - s è il **segno**, ($0 \rightarrow +$, $1 \rightarrow -$)
 - m è la **mantissa**, numero frazionario su p bit (precisione);
 - e è l'**esponente**, numero intero
- Forma **normalizzata**: la parte intera della mantissa ha una sola cifra (significativa), es. $5.79 \cdot 10^2$
- In base 2: $\pm 1.b_{-1}b_{-2} \dots b_{-(p-1)} \cdot 2^e$, rappresentiamo:
 - Segno: 0, 1
 - Mantissa: la parte frazionaria $b_{-1}b_{-2} \dots b_{-(p-1)}$ (la parte intera è implicita)
 - Esponente: e

Approssimare i numeri reali

Il numero: $\pm 1.b_{-1}b_{-2} \dots b_{-(p-1)} \cdot 2^e$ è associato al valore:

$$\pm(1 + b_{-1} \cdot 2^{-1} + b_{-2} \cdot 2^{-2} \dots b_{-(p-1)} \cdot 2^{-(p-1)}) \cdot 2^e$$

Risoluzione variabile: il contributo del bit meno significativo ($b_{-(p-1)}$) dipende da e .



Virgola fissa e virgola mobile

- Confronto tra le due rappresentazioni attraverso errore assoluto (ϵ_A) e errore relativo (ϵ_R)
- $B = 10$, numero totale di cifre pari a 6

Virgola fissa	Virgola mobile
$R_{N,VF} = c_2c_1c_0.c_{-1}c_{-2}c_{-3}$ $V_{min} = 000.000, V_{max} = 999.999 \cong 10^3$ $\epsilon_A \leq 0.001$ $\epsilon_R \in \left[\frac{0.001}{10^3}, \frac{0.001}{0} \right)$	$R_{N,VM} = i.fff \cdot 10^{ee}$ $V_{min} = 0, V_{max} 9.999 \cdot 10^{99} \cong 10^{100}$ $\epsilon_A \leq 10^{-3} \cdot 10^{ee} \leq 10^{ee-3}$ $\epsilon_R = \frac{10^{ee-3}}{M \cdot 10^{ee}} \cong 10^3$

- A parità di cifre utilizzate, la rappresentazione in virgola mobile è in grado di rappresentare un numero più elevato di valori
- In virgola fissa: precisione assoluta costante e relativa variabile
- In virgola mobile: precisione assoluta variabile con N e relativa approssimativamente costante

Esponente: rappresentazione in eccesso 127

- Per rappresentare il numero $e_{(10)}$ si somma ad esso $2^{8-1} - 1 = 127_{(10)}$
- Si rappresenta il valore risultante $E_{(10)} = e_{(10)} + 127$
- Intervallo rappresentato (numeri normalizzati):
 $-126 \leq e_{(10)} \leq 127 \rightarrow 1 \leq E_{(10)} \leq 255$

$$(-34)_{(10)} = (?)_{8bit, e127}$$

$$e_{(10)} = -34, E_{(10)} = -34 + 127 = +93_{(10)} = 01011101_{(2)}$$

$$(11100101)_{8bit, e127} = (?)_{(10)}$$

$$E_{(10)} = +229, e_{(10)} = 229 - 127 = 102_{(10)}$$

Mantissa

- La mantissa M viene rappresentata nella forma $1.c_{-1}c_{-2}\dots c_{-23}$
- Il bit c_0 corrisponde al peso $2^0 = 1$ ed è, per convenzione, sempre uguale a 1 e non si rappresenta
- Il punto decimale segue sempre il bit c_0 e, per convenzione, non si rappresenta; questo corrisponde alla **rappresentazione normalizzata** dove il punto decimale è posto sempre dopo la prima cifra più significativa diversa da zero.
- i 23 bit di M rappresentano l'intervallo $[1, 2)$

Numeri reali: lo standard IEEE 754

Esempio 1: convertiamo $(17.375)_{10}$, dobbiamo determinare s , m , E

- Il numero è positivo $\rightarrow s = 0$
- Converto la parte intera in binario: $(17)_{10} = (10001)_2$
- Converto la parte frazionaria $(.375)_{10} (= \frac{3}{2^3})$

$0.375 \cdot 2 = 0.75$	parte intera = 0	MSD		$(.375)_{10} = (.011)_2$
$0.75 \cdot 2 = 1.5$	parte intera = 1			
$0.5 \cdot 2 = 1.0$	parte intera = 1			

- Unisco i risultati: $(10001.011)_2$
- Normalizzazione: $(1.0001011)_2 \cdot (10)_2^4$
- Mantissa: $m = 0001\ 0110\ 0000\ 0000\ 0000\ 0000$
- Esponente: $E = e + 127 = (4)_{10} + (127)_{10} = (131)_{10} = (10000011)_2$

Numeri reali: lo standard IEEE 754

Esempio 2: convertiamo $(-0.8)_{10}$, dobbiamo determinare s , m , E

- Il numero è negativo $\rightarrow s = 1$
- Converto la parte intera in binario: $(0)_{10} = (0)_2$
- Converto la parte frazionaria $(.8)_{10}$

$0.8 \cdot 2 = 1.6$	parte intera = 1	MSD
$0.6 \cdot 2 = 1.2$	parte intera = 1	
$0.2 \cdot 2 = 0.4$	parte intera = 0	
$0.4 \cdot 2 = 0.8$	parte intera = 0	
$0.8 \cdot 2 = \dots$	la sequenza 1100 si ripete	

- Unisco i risultati: $(0.8)_{10} = (0.\overline{1100})_2$
- Normalizzazione: $(0.\overline{1100})_2 = (0.1100 \overline{1100})_2 = (1.100 \overline{1100})_2 \cdot (10)_2^{-1}$
- Mantissa: $m = 100 \ 1100 \ 1100 \ 1100 \ 1100 \ 1100$
- Esponente:
 $E = e + 127 = (-1)_{10} + (127)_{10} = (126)_{10} = (1111110)_2 \rightarrow (01111110)_2$

Numeri reali: lo standard IEEE 754

Certi valori di m e di e sono utilizzati secondo diverse convenzioni, definite dallo standard ($E = e + 127$)

- Se $0 < E < 255$ ($e \in [-126, 127]$) \rightarrow numero normalizzato
- Se $m = 0$, $E = 0 \rightarrow \pm 0$ (a seconda di s)
- Se $m = 0$, $E = 255 \rightarrow \pm \infty$ (a seconda di s)
- Se $m \neq 0$, $E = 255 \rightarrow \text{NaN}$ (Not a Number)
- Se $m \neq 0$, $E = 0 \rightarrow$ numero denormalizzato

Standard IEEE 754: numeri denormalizzati

Quale è il numero positivo normalizzato più piccolo che possiamo rappresentare?

- $s = 0, e = (-126)_{10} \rightarrow E = (0000\ 0001)_2, m = 0000 \dots 000$
- $(1.0)_2 \cdot (10)_2^{-126} = 2^{-126} \approx 1.17 \cdot 10^{-38}$

Il successivo?

- $s = 0, e = (-126)_{10} \rightarrow (0000\ 0001)_2, m = 0000 \dots 001$
- $(1.0 \dots 01)_2 \cdot (10)_2^{-126} = 2^{-126} + 2^{-126-23} \approx 1.17 \cdot 10^{-38} + 1.4 \cdot 10^{-45}$

In prossimità dello zero non abbiamo risoluzione costante!

Possiamo riempire questa regione in modo più “furbo” (risoluzione maggiore e costante)?

Standard IEEE 754: numeri denormalizzati

- Ad E viene assegnato il codice 0000 0000 che si interpreta come esponente pari a -126 (non -127 !)
- la mantissa è un numero diverso da 0 e la parte intera è implicitamente assunta essere 0
- Quindi un numero denormalizzato è sempre fatto così :

$$(-1)^s \cdot (0.m)_2 \cdot (10)_2^{-126}$$

Quale è il numero positivo denormalizzato più piccolo che possiamo rappresentare?

- $s = 0$, $e = (-126)_{10} \rightarrow (0000\ 0000)_2$, $m = 0000 \dots 001$
- $(0.0 \dots 01)_2 \cdot (10)_2^{-126} = 2^{-149} \approx 1.4 \cdot 10^{-45}$
- il successivo sarà ovviamente $2 \cdot 2^{-149} \approx 2.8 \cdot 10^{-45}$

In prossimità dello 0 abbiamo una risoluzione maggiore e costante

Standard IEEE 754: numeri denormalizzati

Esempio 1: rappresentare in formato IEEE 754 il numero

$$-(0.125)_{10} \cdot 2^{-125}$$

- Il numero è negativo $\rightarrow s = 1$
- Il numero è esprimibile come un normalizzato? Per esserlo devo poterlo scrivere come $(0.m) \cdot 2^{-126}$
- $0.125 \cdot 2^{-125} = 0.125 \cdot 2^1 \cdot 2^{-126} = 0.25 \cdot 2^{-126}$, posso scriverlo come denormalizzato
- converto $0.25 \left(\frac{1}{2^2}\right)$ in base 2: $(0.25)_{10} = (0.01)_2$
- Mantissa $m = 010\dots 0$
- Esponente $E = 0000\ 0000$