

Rappresentazione dell'informazione (1)

Dato un insieme di elementi di informazione I , la rappresentazione (o *codifica*) di ciascun elemento è una funzione:

$$f: I \rightarrow C$$

Un codice dell'insieme C è una aggregazione di simboli di un insieme S che costituisce l'*alfabeto di supporto* di C .

La *decodifica* è una funzione che opera la trasformazione inversa:

$$g: C \rightarrow I.$$

- Definizione formale molto ampia, **si presta ad applicazioni in ogni campo**

Esempio: applicazioni in campo linguistico:

La funzione di codifica di un certo insieme, per esempio, **gli oggetti contenuti in una stanza o gli animali di una certa famiglia**, può essere:

- a) Il **vocabolario italiano** che fa corrispondere ad ogni elemento dell'insieme il suo **codice (nome)**, formato utilizzando i 26 simboli dell'alfabeto delle lingue occidentali.
- b) Il **vocabolario della lingua russa** che si appoggia invece all'*alfabeto di supporto* costituito dai caratteri cirillici.

Rappresentazione dell'informazione (2)

L'esempio è una *codifica in chiaro*.

Per proteggere l'informazione si può ricorrere a *codici cifrati*.

Giulio Cesare adottò per i suoi messaggi un codice che slittava di 3 posizioni ciascun carattere nell'elenco circolare dei 26 caratteri dell'alfabeto (a z segue a). Con questo codice la parola: "Zorro" diventa: "Cruur".

Una codifica può essere:

- **non ambigua** quando la funzione di codifica è *iniettiva*, cioè ad elementi distinti in I corrispondono elementi distinti in C .
- **non ridondante** quando il numero di elementi di C è uguale a quello di I ($N_C = N_I$);
- **ridondante** quando: $N_C > N_I$;
- **ambigua** quando: $N_C < N_I$.

Altre importanti caratteristiche di un codice sono:

- **economicità** (numero di simboli mediamente utilizzati per un elemento di informazione);
- **semplicità dell'operazione di codifica** (e di decodifica);
- **semplicità di trattamento dell'informazione codificata**.

Rappresentazione dei NUMERI NATURALI (1)

I **NUMERI NATURALI** sono i numeri interi positivi: 0, 1, 2, 3, 4, 5, 6,

Una possibile codifica: lingua italiana

Difficoltà: sono infiniti. E' necessario ricorrere a codici composti

Una rappresentazione compatta: *sistema di numerazione*.

Quello adottato normalmente è il *sistema decimale posizionale*, basato sulle potenze crescenti del 10.

Alfabeto con *dieci simboli*: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Posizionale: valore diverso (peso) a seconda della posizione. La cifra all'estrema destra, *cifra meno significativa* esprime le unità (peso 10^0), quella adiacente a sinistra esprime le decine (peso 10^1), la successiva esprime le centinaia (peso 10^2) e così via, fino alla cifra all'estrema sinistra detta *cifra più significativa*, che avrà peso 10^{n-1} .

Il numero naturale, **duecentotrentacinque** nel sistema di numerazione decimale si scrive **235** ed il suo valore è:

$$2 \cdot 10^2 + 3 \cdot 10^1 + 5 \cdot 10^0$$

Rappresentazione dei NUMERI NATURALI (2)

Il *codice unario* (non posizionale) utilizza un solo simbolo. Se adottiamo come simbolo un tratto verticale: **I**, il numero naturale zero si scriverà: **I**, tre si scriverà: **III**, la codifica di ventitre richiederà 24 tratti verticali, mentre la sua codifica in decimale richiede due soli simboli il 2 e il 3.

Il codice *unario* non è "economico".

Tutti i codici numerici utilizzati sono posizionali ed hanno necessariamente base **b** ≥ 2 .

Simbologia:

- **N** indica un numero naturale, a prescindere dalla sua rappresentazione,
- **N_b** indica un numero naturale rappresentato in base b,
- una *sequenza di cifre senza pedice* rappresenta un numero naturale espresso in base 10
- una *sequenza di cifre con pedice b* rappresenta un numero espresso in base $b \neq 10$.

$$N_b = c_{m-1} c_{m-2} \dots c_2 c_1 c_0$$

$$N = c_{m-1} \cdot b^{m-1} + c_{m-2} \cdot b^{m-2} + \dots + c_2 \cdot b^2 + c_1 \cdot b^1 + c_0 \cdot b^0$$

Rappresentazione dei NUMERI NATURALI (3)

$$N_b = c_{m-1}c_{m-2}\dots c_2c_1c_0 \quad N = \sum_{i=0}^{m-1} c_i \cdot b^i \quad (1)$$

La formula (1) stabilisce una **corrispondenza biunivoca** tra i numeri naturali e la loro codifica in base b.

Ciascun numero naturale ha un codice in base b **ed uno solo** ed a ciascun codice in base b corrisponde un numero naturale **ed uno solo**.

E' ovvio che qualunque sia la base b la formula (1), calcolata adoperando l'aritmetica decimale, costituisce anche un *metodo di conversione di un numero dalla base b in decimale*.

Esiste una *tecnica generale per convertire un numero decimale in una altra base o da una certa base ad un'altra*.

Conversione di un codice in una diversa base

La tecnica è molto semplice ed intuitiva.

Se il numero è espresso in una base b_1 e si vuole convertirlo in una base b_2 , basta **dividere, prima il numero e, poi, i successivi quozienti per la base b_2 espressa nella base b_1 .**

Il codice del numero nella nuova base sarà formato dai **resti delle divisioni successive presi ordinatamente dall'ultimo al primo.**

Il resto della prima divisione, cioè, costituirà la cifra meno significativa, mentre quello dell'ultima la cifra più significativa. E' importante tenere presente che i resti sono sempre espressi nella base di partenza e quindi **è necessario convertirli nella base di arrivo.**

Le divisioni tra i due numeri nella stessa base vanno effettuate seguendo le regole della aritmetica corrispondente che è per tutte le basi analoga a quella decimale. Qualche difficoltà pratica nasce per basi superiori a 10.

Conversione da decimale a ternario e viceversa

$$\begin{array}{r}
 251 : 3 \\
 \underline{24} \\
 11 \\
 \underline{09} \\
 2
 \end{array}
 \quad
 \begin{array}{r}
 83 : 3 \\
 \underline{6} \\
 23 \\
 \underline{21} \\
 2
 \end{array}
 \quad
 \begin{array}{r}
 27 : 3 \\
 \underline{27} \\
 0
 \end{array}
 \quad
 \begin{array}{r}
 9 : 3 \\
 \underline{9} \\
 0
 \end{array}
 \quad
 \begin{array}{r}
 3 : 3 \\
 \underline{3} \\
 0
 \end{array}
 \quad
 \begin{array}{r}
 1 : 3 \\
 \underline{0} \\
 1
 \end{array}$$

$251 = 100022_3$ $10 = 101_3$

$$\begin{array}{r}
 100022_3 : 101_3 \\
 \underline{202} \\
 212 \\
 \underline{202} \\
 102 \\
 \underline{101} \\
 1_3
 \end{array}
 \quad
 \begin{array}{r}
 221_3 : 101_3 \\
 \underline{202} \\
 12_3
 \end{array}
 \quad
 \begin{array}{r}
 2_3 : 101_3 \\
 \underline{0} \\
 2_3
 \end{array}$$

$12_3 = 5$ $100022_3 = 251$

Conversione da ottale a decimale e viceversa

$$\begin{array}{r}
 310_8 : 12_8 \\
 \underline{24} \\
 50 \\
 \underline{50} \\
 0
 \end{array}
 \quad
 \begin{array}{r}
 24_8 : 12_8 \\
 \underline{24} \\
 0
 \end{array}
 \quad
 \begin{array}{r}
 2_8 : 12_8 \\
 \underline{0} \\
 2
 \end{array}$$

$10 = 12_8; 310_8 = 200$

$$\begin{array}{r}
 200 : 8 \\
 \underline{16} \\
 40 \\
 \underline{40} \\
 0
 \end{array}
 \quad
 \begin{array}{r}
 25 : 8 \\
 \underline{24} \\
 1
 \end{array}
 \quad
 \begin{array}{r}
 3 : 8 \\
 \underline{0} \\
 3
 \end{array}$$

$8_8 = 8; 200 = 310_8$

Esercizi sulla conversione da una base ad un'altra

- Convertire da base 2 a base 10:

0110011

10101100

1100110011

- Convertire in base 10 i seguenti numeri:

102210₃

431204₅

5036₇

198A₁₂

Esercizi sulla conversione da una base ad un'altra

- Convertire da base 10 alla base indicata accanto i seguenti numeri:

7562 base 8

1938 base 16

175 base 2

- Convertire da base 16 a base 2 e base 8: F3A7C2

- Data l'equazione: $x^2 - 10_b x + 31_b = 0$;

se le radici sono $x_1 = 5$ e $x_2 = 8$, in che base sono espressi i coefficienti?

LUNGHEZZA di un numero naturale (1)

La *lunghezza* di un numero naturale è il numero di cifre necessario per esprimerlo in una data base.

E' intuitivo che se N'_b , e N''_b sono due numeri espressi nella stessa base e m' ed m'' sono le rispettive lunghezze; se è vero che:

$$N'_b > N''_b$$

allora deve essere:

$$m' \geq m''.$$

Tesi: La lunghezza di un numero è una funzione sempre crescente del numero stesso

E' altrettanto evidente che più piccola è la base e maggiore sarà il numero di cifre necessario per esprimere un certo numero.

Se N_a ed N_b sono due rappresentazioni di uno stesso numero naturale N e m_a e m_b sono le corrispondenti lunghezze ed è:

$$a > b;$$

allora deve essere:

$$m_a \leq m_b.$$

Tesi: La lunghezza di un numero decresce al crescere della base di codifica.

Ottobre - 2001

Arch. degli elab. Mod. A - 1. Rappresentazione dell'informazione

11

LUNGHEZZA di un numero naturale (2)

Nel sistema di numerazione decimale il massimo numero esprimibile con 3 cifre è 999 cioè: $10^3 - 1$.

Per analogia il massimo numero esprimibile in base b con m cifre è:

$$v_{max} = b^m - 1; v_{max} + 1 = b^m$$

e, passando ai logaritmi in base b si ricava:

$$m = \log_b (v_{max} + 1).$$

Attribuendo alla formula una validità più generale si può scrivere:

$$m_b = \log_b (N)$$

Questa relazione raramente darà come risultato un numero intero, mentre il numero di cifre deve essere intero. Si arrotonda per eccesso:

La lunghezza del numero N_b sarà pertanto:

$$m_b = \lfloor \log_b (N_b) \rfloor + 1$$

Essendo la funzione logaritmo una funzione sempre crescente, si può dire che la **lunghezza cresce col numero in qualunque base.**

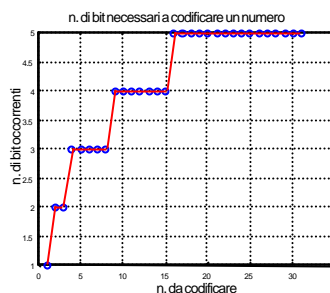
Ottobre - 2001

Arch. degli elab. Mod. A - 1. Rappresentazione dell'informazione

12

LUNGHEZZA di un numero naturale (3)

N. di bit necessari a codificare numeri naturali in base 2 (lunghezza dei numeri naturali in base 2)	n	log ₂ (n)	N of bits	Binary code
	1	0	1	[0 0 0 0 0 1]
	2	1	2	[0 0 0 0 1 0]
	3	1.585	2	[0 0 0 0 1 1]
	4	2	3	[0 0 0 1 0 0]
	5	2.322	3	[0 0 0 1 0 1]
	6	2.585	3	[0 0 0 1 1 0]
	7	2.807	3	[0 0 0 1 1 1]
	8	3	4	[0 0 1 0 0 0]
	9	3.17	4	[0 0 1 0 0 1]
	10	3.322	4	[0 0 1 0 1 0]
	11	3.459	4	[0 0 1 0 1 1]
	12	3.585	4	[0 0 1 1 0 0]
	13	3.7	4	[0 0 1 1 0 1]
	14	3.807	4	[0 0 1 1 1 0]
	15	3.907	4	[0 0 1 1 1 1]
	16	4	5	[0 1 0 0 0 0]
	17	4.087	5	[0 1 0 0 0 1]
	18	4.17	5	[0 1 0 0 1 0]
	19	4.248	5	[0 1 0 0 1 1]
	20	4.322	5	[0 1 0 1 0 0]
	21	4.392	5	[0 1 0 1 0 1]
	22	4.459	5	[0 1 0 1 1 0]
	23	4.524	5	[0 1 0 1 1 1]
	24	4.585	5	[0 1 1 0 0 0]
	25	4.644	5	[0 1 1 0 0 1]



Ottobre - 2001

Arch. degli elab. Mod. A - 1. Rappresentazione dell'informazione

13

La lunghezza di un numero cresce al diminuire della base (1)

Le lunghezze di un numero naturale N espresso nelle due basi a e b sono:

$$m_a = \log_a(N) \text{ e } m_b = \log_b(N).$$

Applicando la relazione per il cambio di base di un logaritmo si ha:

$$\log_a(N) = \log_a(b) \log_b(N)$$

che si può mettere nella forma:

$$m_a = \log_a(b) m_b$$

Poiché il logaritmo di un numero è l'esponente a cui bisogna elevare la base per avere il numero, se a è maggiore di b il logaritmo è minore di 1 e la **lunghezza del numero naturale N nella base a è minore che in b .**

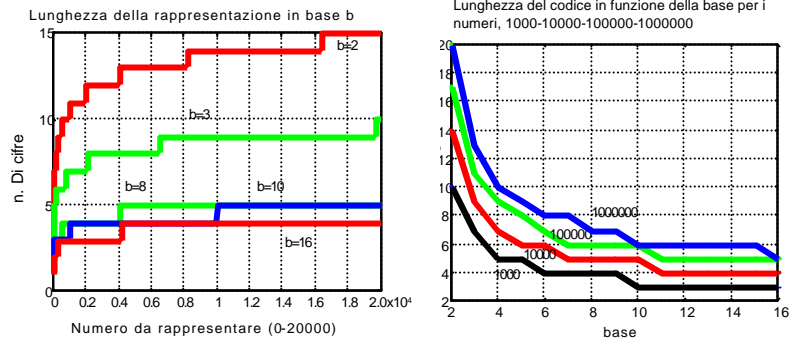
Se si pone $a=2$ e $b=10$, poiché la potenza a cui bisogna elevare 2 per avere 10 è maggiore di 3 (il logaritmo in base 2 di 10 è 3,32), dalla formula si può tranquillamente concludere che **il sistema di numerazione binario dal punto di vista della lunghezza di rappresentazione dei numeri naturali è la scelta peggiore possibile**, certamente molto peggiore del sistema decimale.

Ottobre - 2001

Arch. degli elab. Mod. A - 1. Rappresentazione dell'informazione

14

La lunghezza di un numero cresce al diminuire della base (2)



lunghezza della rappresentazione in funzione della base 2-3-8-10-16

Gestione di numeri binari di lunghezza fissa

L'economicità hardware del codice binario e la semplicità della sua aritmetica lo impongono nella realizzazione di qualsiasi apparecchiatura elettronica digitale.

Nessuno ha mai prodotto su larga scala apparecchiature digitali usando una codifica dei numeri non binaria.

La lunghezza delle rappresentazioni numeriche è importante perché tutti gli elaboratori di informazione lavorano a **numero fisso di cifre binarie**. Si è passati rapidamente da 8 a 16 a 32 bit ed ora siamo a 64 bit ma il limite rimane.

L'adozione di un numero fisso di cifre (**parola macchina di lunghezza fissa**) comporta che nella rappresentazione di **numeri naturali molto più piccoli del massimo numero esprimibile** con le cifre a disposizione, le **cifre non utilizzate dovranno essere comunque riempite con "0"** (zeri non significativi) e se il risultato di una somma o una moltiplicazione **supera il massimo numero esprimibile** si avrà un **traboccamento di bit dal campo a disposizione (overflow)**.

Rappresentazione dei numeri interi

La rappresentazione dei numeri naturali (numeri interi positivi o nulli) sul piano pratico non apre molte prospettive di calcolo, è necessario rappresentare almeno i numeri interi (positivi e negativi) per poter essere in grado di operare somme algebriche.

Tra le infinite possibili rappresentazione dei numeri interi quelle di maggiore interesse sono:

- **Rappresentazione con segno;**
- **Rappresentazione alla base diminuita (in binario, complemento ad uno)**
- **Rappresentazione all'intervallo (in binario, complemento a due)**

Tutte queste rappresentazioni si possono adottare con qualsiasi base sia pari che dispari perché le proprietà su cui si basano sono generali.

Rappresentazione dei numeri interi con segno

Si riserva al simbolo del segno la posizione all'estrema sinistra, quella che corrisponderebbe alla cifra più significativa di un numero naturale.

Di solito si associa al **segno positivo il simbolo "0" ed a quello negativo "1"**.

Con questa tecnica il **campo numerico** a disposizione, in una certa base **b** e con **m** cifre, è di $b^{m-1}-1$ numeri positivi ed altrettanti negativi, oltre allo "0" che ha addirittura due rappresentazioni perché, ovviamente, ci sarà lo 0 positivo (tutti "0" con la cifra segno a "0") e quello negativo (tutti "0" con la cifra segno ad "1").

Se si adotta la notazione $C_i^{(m)}$ per indicare le **m** cifre di una rappresentazione numerica i due "0" della rappresentazione con segno sono $0^{(m)}$ e $1 0^{(m-1)}$.

Rappresentazione dei numeri interi con segno (2)

In binario se si lavora con numeri di sole 3 cifre, i tre numeri positivi rappresentabili sono:

- 1: 001_2
- 2: 010_2
- 3: 011_2 ;

quelli negativi sono:

- -1: 101_2
- -2: 110_2
- -3: 111_2 .

Le due rappresentazioni dello "zero" sono:

- 000_2
- 100_2 .

Con questa rappresentazione le somme algebriche dipendono dal segno concorde o discorde dei due numeri

Rappresentazione in complemento alla base diminuita

Si chiama così perché per ottenere la rappresentazione del numero negativo $-x$ si opera la complementazione a $b-1$ di ciascuna cifra del numero x .

L'espressione "**base diminuita**" sottolinea che l'operazione di complementazione si effettua rispetto alla cifra di maggior valore usata nella rappresentazione, che è sempre uguale alla base diminuita di un'unità.

In termini generali si **sostituisce a ciascuna cifra c_i la cifra:**

$$(b-1)-c_i.$$

Per semplicità è opportuno separare il caso di basi pari da quelle dispari anche se non esistono differenze sostanziali.

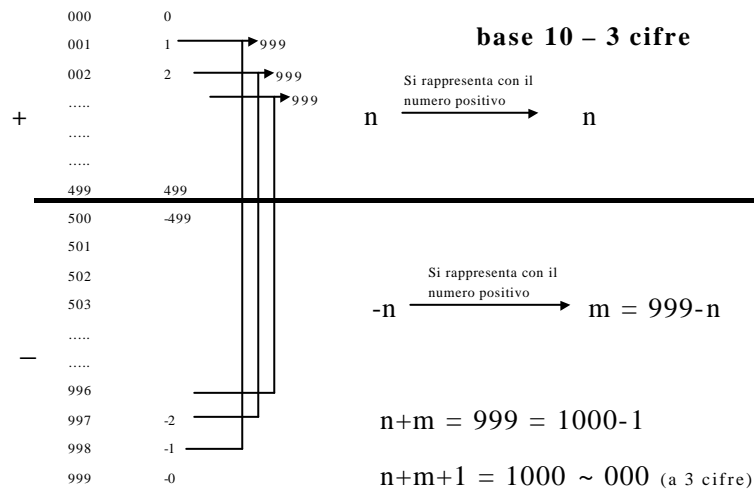
Rappresentazione in complemento alla base diminuita (2)

Se si considerano **numeri di tre cifre** ($m = 3$) in **base 10**, i numeri **positivi** sono compresi tra **001** e **499**, mentre quelli **negativi** sono compresi tra **500** e **998** che sono appunto il risultato della complementazione a 9, cifra per cifra, rispettivamente di 499 e 001. **Il campo dei numeri rappresentabili va da -499 a +499.**

Anche in questo caso ci sono **due rappresentazioni dello zero**, **000** e **999**, essendo l'uno il risultato della complementazione dell'altro. Il complemento del numero positivo 347 è 652. Poiché il complemento di 652 è 347, l'operazione verifica la condizione necessaria che $-(-x)=x$.

Generalizzando le considerazioni fatte nell'esempio si può dire che con **m** cifre in base **b** si possono rappresentare i numeri compresi tra $(-b^m/2)+1$ e $(+b^m/2)-1$.

Rappresentazione in complemento alla base diminuita (3)



Rappresentazione in complemento ad uno

In binario la complementazione si effettua convertendo tutti gli "0" in "1" e viceversa. Con 3 cifre i numeri 1, 2 e 3 sono rappresentati da 001_2 , 010_2 e 011_2 ed i numeri -1, -2 e -3 da 110_2 , 101_2 , 100_2 . Le due rappresentazioni dello "0" sono 000_2 e 111_2 .

E' evidente che i numeri positivi hanno come bit più significativo (MSB) uno "0", mentre quelli negativi hanno un "1".

In generale in base b i **numeri positivi** saranno caratterizzati dalla cifra più significativa **inferiore a $b/2$** e quelli **negativi** da una cifra più significativa uguale o **superiore a $b/2$** .

In decimale i positivi finiscono a 499 ed i negativi cominciano a 500.

Si osservi che **il numero positivo più alto 499 è contiguo** in una tabella progressiva al **numero negativo di più elevato valore assoluto** e la tabella è **simmetrica rispetto al punto di separazione** che è marcato dal cambio della cifra più significativa.

Questa circostanza **non si verifica per i sistemi numerici a base dispari** in cui esiste un elemento di separazione, per cui ci sarebbe un numero positivo in più e non si può dedurre il segno dalla cifra più significativa.

Numeri ternari (base 3) in complemento alla base diminuita

	9	3	1						9	3	1				
0	0	0	0	₃	0			14	1	1	2	₃			-12
1	0	0	1	₃	1			15	1	2	0	₃			-11
2	0	0	2	₃	2			16	1	2	1	₃			-10
3	0	1	0	₃	3			17	1	2	2	₃			-9
4	0	1	1	₃	4			18	2	0	0	₃			-8
5	0	1	2	₃	5			19	2	0	1	₃			-7
6	0	2	0	₃	6			20	2	0	2	₃			-6
7	0	2	1	₃	7			21	2	1	0	₃			-5
8	0	2	2	₃	8			22	2	1	1	₃			-4
9	1	0	0	₃	9			23	2	1	2	₃			-3
10	1	0	1	₃	10			24	2	2	0	₃			-2
11	1	0	2	₃	11			25	2	2	1	₃			-1
12	1	1	0	₃	12			26	2	2	2	₃			0
13	1	1	1	₃	13										

Somme algebriche con il complemento alla base diminuita

L'operazione decimale algebrica $11 - 8$ nell'aritmetica complemento a due in base ternaria con numeri di tre cifre in cui:

$$\begin{array}{r} 11 \text{ } \textcircled{R} \text{ } 102_3 \\ - 8 \text{ } \textcircled{R} \text{ } 200_3 \end{array}$$

si trasforma in:

$$102_3 + 200_3 = 1\ 002_3$$

che genera un overflow di una unità.

Se quest'ultima viene sommata al risultato si ottiene:

$$010_3$$

che è il risultato giusto perché corrisponde a 3 in decimale.

L'operazione decimale $235-127=108$, nella logica complemento a 9 diventa $235+872=1\ 107$ (perché $999-127=872$). Ma: $107+1=108$.

Le rappresentazioni dei numeri negativi per complementazione, contrariamente a quella con segno, risolvono anche il problema della somma algebrica tra numeri interi. Le procedure differiscono solo in qualche dettaglio .

Rappresentazione per complemento all'intervallo

Per ottenere la rappresentazione con segno negativo di un numero si sottrae, nell'aritmetica della base b , il numero dato, formato da m cifre, dalla base elevata alla potenza *m-esima*.

Con m cifre in base b si possono rappresentare i numeri compresi tra:

$$(-b^{m/2}+1) \text{ e } (+b^{m/2}-1)$$

Anche in questo caso conviene considerare l'ipotesi di base pari, visto che lo sono tutte le basi che hanno un reale interesse, ma non è difficile adattare i risultati a basi dispari.

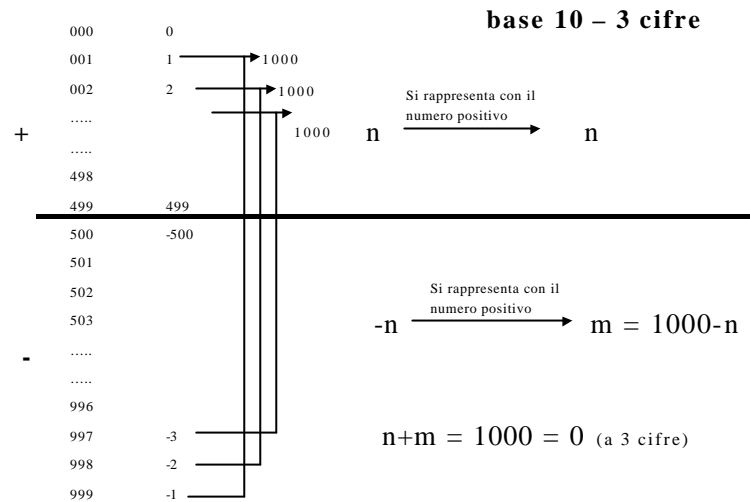
I numeri positivi sono rappresentati nell'intervallo $[0, (b^{m/2}-1)]$ e quelli negativi nell'intervallo $[(-b^{m/2}+1)-1]$ ottenendoli dalla formula: b^{m-N} .

Si ha una sola rappresentazione dello "0", quella indicata con $0^{(m)}$.

Esiste una combinazione che per simmetria non si usa:

$$\frac{b^{(m-1)}}{2} \text{ } \textcircled{R} \text{ } 0^{(m)}$$

Rappresentazione per complemento all'intervallo (2)



Ottobre - 2001

Arch. degli elab. Mod. A - 1. Rappresentazione dell'informazione

27

Rappresentazione per complemento all'intervallo (3)

In **base decimale**, ad esempio, la rappresentazione del numero di 3 cifre 237 col segno negativo:

$$- 237$$

si ottiene dall'operazione decimale:

$$1000 - 237 = 763;$$

per cui l'operazione:

$$463 - 237 = 226$$

si trasforma in:

$$463 + 763 = 1\ 226$$

che, ignorando l'overflow (4^a cifra), dà il risultato atteso.

I **numeri positivi** vanno da **001 a 499**, quelli **negativi** da **999 a 501** e la combinazione non usata è **500** che, cominciando con 5, è negativo.

E' banale riscontrare che questa rappresentazione si può anche ottenere da quella della base diminuita aggiungendo un 1 dopo la complementazione.

Ottobre - 2001

Arch. degli elab. Mod. A - 1. Rappresentazione dell'informazione

28

Rappresentazione binaria in complemento a due

In base binaria e con 3 sole cifre, i numeri positivi rappresentabili sono:

1, 2, 3

che corrispondono alle combinazioni:

001₂ 010₂ 011₂

lo zero è:

000

i numeri negativi sono:

- 1, - 2, - 3

e si rappresentano nella forma:

111₂ 110₂ 101₂.

La combinazione non usata è:

100₂

e corrisponde a - 4.

Rappresentazione binaria in complemento a due (2)

	0000	0			
	0001	1			
	0010	2			
+	0011	3	1000	$n > 0$	Si rappresenta con il numero positivo → n
	0100	4			
	0101	5			
	0110	6			
	0111	7			
<hr/>					
	1000	-8	$8=16-8$		
	1001	-7	$9=16-7$		
	1010	-6	$10=16-6$		
	1011	-5	$11=16-5$	$m < 0$	Si rappresenta con il numero positivo → $k = 8+m$
-	1100	-4	$12=16-4$		
	1101	-3	$13=16-3$		
	1110	-2	$14=16-2$		
	1111	-1	$15=16-1$		

$$n+k = n+m+8_{10} = n_2+m_2+1000_2 = n_2+m_2 \text{ (a 3 bit)}$$

Teor. Se n ed m sono due numeri interi esprimibili a b bits, n+m è esprimibile a b+1 bits.

Conclusioni

- Le rappresentazioni dei numeri negativi ottenute attraverso un'operazione di complementazione consentono di effettuare somme algebriche senza preventivo esame del segno del numero.
- La complementazione alla base diminuita è più semplice perché può essere eseguita cifra per cifra, ma per passare all'altra basta aggiungere un 1.
- In compenso, dopo una operazione in complemento alla base diminuita, bisogna aggiungere un "1" per avere il risultato giusto, mentre nell'altra basta ignorare l'overflow.
- In binario entrambe sono molto semplici e facilmente implementabili ma i piccoli vantaggi offerti dall'aritmetica complemento a due ne hanno ormai fatto lo standard generalmente utilizzato per le architetture semplici.
- Per le architetture più complesse viene preferita la rappresentazione con segno.

La codifica BCD (Binary Coded Decimal)

L'elettronica degli elaboratori è **binaria**, mentre la mente umana è abituata a ragionare in **decimale**. Per mettere d'accordo i due mondi, sono nati i codici **BCD**, da **Binary Coded Decimal**.

Si parla di codici, al plurale, perché possono essere infiniti, visto che infinite possono essere le codifiche binarie dei dieci simboli del sistema decimale.

Quando si parla di codice BCD senza specificare il tipo di codice binario adottato ci si riferisce necessariamente a quello **posizionale basato sulle potenze crescenti del 2**, detto anche codice binario "**puro**" o "**naturale**" o, anche, **8 4 2 1 (1 2 4 8)** con riferimento ai pesi dei 4 bit letti da quello più significativo (MSB) verso quello meno significativo (LSB) (o viceversa).

Essendo 10 i **simboli** da codificare (da **0 a 9**) sono necessari 4 bit che, come è noto, possono dar luogo a **16 combinazioni** che in binario puro corrispondono ai numeri naturali da **0 a 15**.

Poiché le **6 combinazioni dal 10 al 15** non si usano, la codifica BCD è **ridondante**.

La codifica BCD (Binary Coded Decimal) (2)

Naturalmente, la **codifica BCD** viene usata solo in funzione di **interfaccia** per rendere comprensibile ad operatori umani i risultati di una elaborazione numerica binaria.

La **codifica BCD** del numero è, infatti, una operazione propedeutica alla sua visualizzazione su un display numerico decimale.

I quattro bit di ciascuna cifra codificata BCD vengono inviati ad un circuito di decodifica che provvederà a pilotare il visualizzatore della cifra corrispondente.

E' opportuno sottolineare che, mentre la codifica binaria è il primo passo per approdare ad un **sistema numerico** che porta poi all'**aritmetica binaria**, non esiste una **aritmetica BCD** perchè in ipotetiche operazioni aritmetiche su codici BCD bisognerebbe applicare ai codici binari dei numeri da 0 a 9 l'**aritmetica decimale**, che non gode certo della semplicità di quella binaria.

NUMERI REALI in virgola fissa

La rappresentazione dei numeri reali in virgola fissa assegna **h** degli **m** bit della parola (o del gruppo di parole) alla **parte intera** ed i rimanenti **$k = m - h$** bit alla **parte frazionaria** del numero.

La virgola è collocata sempre dopo le prime h cifre, indipendentemente dal loro valore. Se il numero da rappresentare è piccolo le cifre significative saranno precedute da "0".

E' banale rendersi conto che il campo numerico rappresentabile in virgola fissa si riduce moltissimo perché il contributo della parte frazionaria (k cifre) aggiunge al massimo una unità al campo delle h cifre intere.

Il peso delle cifre dopo la virgola sarà: b^{-1}, b^{-2}, b^{-3} ecc...

Se si ha a che fare con un sistema binario con una parola macchina a **32 bit** e si immagina di utilizzare **20 bit** per la **parte intera con segno** e **12 bit** per quella **frazionaria**, i numeri rappresentabili sono quelli compresi tra **$(-2^{19}+1)$ e $+2^{19}-1$** ed i 12 bit frazionari per definizione avranno un valore globale non superiore ad una unità. Orientativamente **12 bit binari** corrispondono a poco più di **3 cifre significative decimali**.

Cambio della base di un numero in virgola fissa

$$c_{h-1}c_{h-2} \dots c_0 , c_{-1}c_{-2}c_{-k} \quad \sum_{i=-k}^{h-1} c_i b^i$$

Le regole per le operazioni aritmetiche in virgola fissa sono perfettamente analoghe a quelle per i numeri interi.

Cambio di base.

La regola è del tutto generale e può essere usata per passare da una qualsiasi base a ad una qualsiasi base b .

- ♦ **Parte intera:** regola dei numeri interi.
- ♦ **Parte frazionaria:** procedimento inverso rispetto a quello per i numeri interi, perché si basa sulla moltiplicazione, invece che sulla divisione.

Cambio della base di un numero in virgola fissa (2)

$$c_{h-1}c_{h-2} \dots c_0 , c_{-1}c_{-2}c_{-k} \quad \sum_{i=-k}^{h-1} c_i b^i$$

La parte frazionaria del numero N espresso nella base a si moltiplica per la base b espressa anch'essa nella base a .

- ♦ La parte intera del risultato I , convertita nella base di arrivo è la prima cifra frazionaria della rappresentazione in base b .
- ♦ La parte frazionaria del risultato F si moltiplica ancora per b e la nuova parte intera I , convertita, costituisce la seconda cifra frazionaria della rappresentazione, e così via.

Naturalmente il procedimento potrebbe talvolta continuare all'infinito e va, quindi, arrestato quando si sono raggiunte le cifre desiderate.

Cambio di base in virgola fissa: Esempi

Numeri N senza parte intera (per semplicità).

Conversione **in binario** nel numero **decimale 0,816**

$$\begin{array}{llll}
 0,816 \times 2 = 1,632; & I = 1; & F = 0,632 & c_{-1} = 1 \\
 0,632 \times 2 = 1,264 & I = 1 & F = 0,264 & c_{-2} = 1 \\
 0,264 \times 2 = 0,528 & I = 0 & F = 0,528 & c_{-3} = 0 \\
 0,528 \times 2 = 1,056 & I = 1 & F = 0,056 & c_{-3} = 1
 \end{array}$$

Il risultato è: $0,816 = 0,1101_2$

Conversione **in decimale** ($10 = 1010_2$) del numero **binario $0,1011_2$**

$$\begin{array}{llll}
 0,1011 \times 1010 = 110,111 & I = 110 & F = 0,111 & c_{-1} = 6 \\
 0,111 \times 1010 = 1000,11 & I = 1000 & F = 0,11 & c_{-1} = 8 \\
 0,11 \times 1010 = 111,1 & I = 111 & F = 0,1 & c_{-1} = 7
 \end{array}$$

Il risultato è: $0,1011_2 = 0,687$

Dettagli nella prossima trasparenza ®

Cambio di base in virgola fissa: Esempi (continua)

$$\begin{array}{cccc}
 0,816 \times & 0,632 \times & 0,264 \times & 0,528 \times \\
 \underline{2} & \underline{2} & \underline{2} & \underline{2} \\
 1,632 & 1,264 & 0,528 & 1,056
 \end{array}$$

$$\begin{array}{ccc}
 \begin{array}{r}
 0,1011 \times \\
 \underline{1010} \\
 \quad 0000 \\
 \quad 1011 \text{---} \\
 \underline{1011 \text{---}} \\
 110,1110 \\
 \downarrow \\
 6
 \end{array}
 &
 \begin{array}{r}
 0,111 \times \\
 \underline{1010} \\
 \quad 000 \\
 \quad 111 \text{---} \\
 \underline{111 \text{---}} \\
 1000,110 \\
 \downarrow \\
 8
 \end{array}
 &
 \begin{array}{r}
 0,11 \times \\
 \underline{1010} \\
 \quad 00 \\
 \quad 11 \text{---} \\
 \underline{11 \text{---}} \\
 111,10 \\
 \downarrow \\
 7
 \end{array}
 \end{array}$$

Rappresentazione dei numeri in virgola mobile

Limitazione del numero di bit rende la rappresentazione in **virgola fissa** dei numeri **inadeguata**, per calcoli complessi.

Con **parole di 64 bit** si possono rappresentare **numeri interi** fino a **18 cifre decimali**. Ma gli elaboratori a 64 bit esistono oggi mentre certi tipi di calcoli scientifici si fanno da decenni.

Le soluzioni messe a punto per fare calcoli significativi con pochi bit permettono di sfruttare meglio anche i molti bit di oggi.

La dizione "**virgola mobile**" è abbastanza esplicita, si è trovato il modo di **svincolare la virgola da una posizione fissa nel numero.**

Il metodo si rifà alla tecnica del calcolo esponenziale che è sempre stato usato nei calcoli scientifici a mano tutte le volte che si aveva a che fare contemporaneamente con numeri molto piccoli e molto grandi.

Il numero N in base b viene rappresentato come $N = m b^e$ dove m è l'iniziale di **mantissa** (cifre significative) ed e quella di **esponente**. Ovviamente N è determinato quando sono noti m ed e .

Rappresentazione dei numeri in virgola mobile (2)

Rappresentazione normalizzata: m ed e sono scelti in modo che $m = 0$.

$$1/b \leq m < 1$$

La mantissa ha sempre la struttura **0,xyz...**. I caratteri "0," **possono essere omessi** senza perdita d'informazione.

La rappresentazione di un numero reale può essere limitata alla coppia di numeri (m, e) . Esempio: Il decimale 463,75 sarà: (46375, 3)

La sua rappresentazione normalizzata è: $0,46375 \times 10^3$.

Mantissa ed esponente debbono poter assumere segno negativo.

Una possibile soluzione: **mantissa rappresentata con segno ed esponente con complemento all'intervallo.**

N (binario) si rappresenta: $(S; m; e)$, dove S è il **segno della mantissa**:

Con 4 bit per la mantissa e 3 per l'esponente, il decimale -0,09375 che in binario diventa: -0,00011, sarà rappresentato dalla terna: $(1; 1100_2; 101_2)$.
che corrisponde a:

$$-0,1100_2 \cdot 10_2^{101_2} = -0,75 \cdot 2^{-3}$$

Limiti della rappresentazione in virgola mobile

Con **M bit di mantissa** ed **E bit d'esponente** numeri **positivi** compresi tra:

$$0,1000 \dots 0_2 \times 2^{-2^{E-1}+1} \quad 0,1111 \dots 1_2 \times 2^{2^{E-1}-1}$$

Numeri **negativi** compresi tra:

$$-0,1111 \dots 1_2 \times 2^{2^{E-1}-1} \quad -0,1000 \dots 0_2 \times 2^{-2^{E-1}+1}$$

Le sequenze normalizzate di M bit: $0,1000\dots 0_2$ e $0,111\dots 1_2$ equivalgono rispettivamente ai numeri $1/2$ e $1-2^{-M}$

Essendo limitato il numero di cifre di mantissa ed esponente, **l'insieme dei numeri rappresentabili in virgola mobile è comunque non continuo e, quindi, discreto.**

“Risoluzione” di una rappresentazione in virgola mobile

La **Risoluzione** di una rappresentazione è **la differenza di due numeri che in essa risultano consecutivi.**

In binario, con M bit di mantissa, il numero $N = (S; m; e)$ non può essere distinto da: $N' = (S; m'; e)$ se:

$$|m - m'| < 2^{-M}$$

L'intervallo dei numeri rappresentabili dipende da E e la risoluzione da M.

E' necessario un ragionevole compromesso fra le due esigenze.

Lo standard IEEE

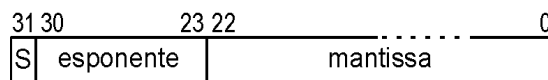
Una rappresentazione generalmente adottata a livello mondiale è quella proposta da:

"Institute of Electrical and Electronical Engineering" (IEEE).

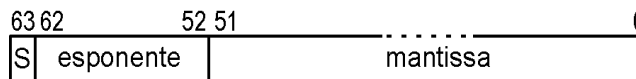
Prevede quattro diversi formati per calcoli in **singola** o **doppia precisione** di tipo **semplice** ed **esteso**.

Formati semplici:

Il formato per la **singola precisione**: utilizza 32 bit in totale, 1 per il segno, 8 per l'esponente e 23 per la mantissa.



Quello per la **doppia precisione**: utilizza 64 bit di cui 1 per il segno, 11 per l'esponente e 52 per la mantissa.



Dettagli dello standard IEEE: Esponente

Per l'esponente (intero positivo o negativo) viene usata la cosiddetta: **"rappresentazione polarizzata"**.

E è il numero di bit assegnato all'esponente.

La **costante di polarizzazione** è: $P = 2^{E-1} - 1$.

Se chiamiamo e il valore di un numero naturale composto da E bit e compreso tra 0 e $2^E - 1$, la rappresentazione si ottiene associando ad ogni numero naturale e un intero e' ottenuto dalla relazione:

$$e' = e - P$$

Esempio semplice: $E = 3$; $P = 2^{E-1} - 1 = 3$

Sostituendo nella formula i valori di e che vanno da 0 a $2^E - 1 = 7$,

$e = 0 \text{ } \otimes \text{ } e' = - 3$; $e = 1 \text{ } \otimes \text{ } e' = - 2$; $e = 2 \text{ } \otimes \text{ } e' = - 1$; $e = 3 \text{ } \otimes \text{ } e' = 0$
 $e = 4 \text{ } \otimes \text{ } e' = + 1$ $e = 5 \text{ } \otimes \text{ } e' = + 2$ $e = 6 \text{ } \otimes \text{ } e' = + 3$ $e = 7 \text{ } \otimes \text{ } e' = + 4$

In pratica le 8 combinazioni di tre bit vengono assegnate (nell'ordine binario naturale da 000 a 111) ai valori relativi crescenti dell'esponente da $- 3$ a $+ 4$.

Dettagli dello standard IEEE: Mantissa

Mantissa: rappresentazione con segno con normalizzazione del tipo: 1,xyz...

Il valore della mantissa è maggiore o uguale ad 1 e minore di 2.

In conclusione il numero N associato ad una terna (S; m; e) sarà:

$$N = (-1)^S \times 1, m \times 2^{e-P}$$

	e = 0	e = 1...254	e = 255
m = 0	$(-1)^S \times 0$	$(-1)^S \times 1,0 \times 2^{e-127}$	$(-1)^S \times \infty$
m ≠ 0	$(-1)^S \times 0, m \times 2^{-126}$	$(-1)^S \times 1, m \times 2^{e-127}$	indefinito

Per $m = 0$ ed $e = 0$, si hanno due rappresentazioni dello "0", come è caratteristico della rappresentazione con segno usata per la mantissa.

Le due combinazioni che corrispondono ad $m = 0$ ed $e = 255$ sono la rappresentazione di $\pm \infty$.

E' prevista per $m \neq 0$ ed $e = 0$ una rappresentazione non normalizzata della mantissa per numeri molto vicini allo 0.

$n = (1/0 + 0.mantissa) \times 2^{esp}$			
NaN		Diverso da 0	e=255
∞		0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	e=255
$2^{128} \cdot \Delta$	1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	2^{127} e=254
2^6			2^6 e=133
$2^5(2-\epsilon) = 2^6 - 2^5 \epsilon \approx 2^6$	1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	2^5 e=132
$2^5(1+\epsilon) = 2^5 + 2^5 \epsilon$	1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1	
2^5	1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	2^4 e=131
$2^4(2-\epsilon) = 2^5 - 2^4 \epsilon \approx 2^5$			
1+e	1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1	2^1 e=128
1	1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	2^0 e=127
1-e	1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	2^{-1} e=126
$2^{-126} \cdot \Delta$	1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1	2^{-126} e=1
2^{-126}	1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	
$2^{-126} \cdot \Delta$	0	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	2^{-126} e=0
$2^{-126-23}$	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1	
0	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	

Rappresentazione in virgola mobile secondo lo standard IEEE: l'intero range

$$esp = e - 127$$

$$\epsilon = 2^{-23}$$

$$\Delta = 2^{esp} \cdot \epsilon = 2^{esp} \cdot 2^{-23}$$

Rappresentazione in virgola mobile secondo lo standard IEEE : l'algoritmo

$$x = 1.m \cdot 2^e$$

$$1 \leq 1.m = x / 2^e < 2$$

$$0 \leq \log_2(x / 2^e) < 1$$

$$-1 < e - \log_2(x) \leq 0$$

$$-1 + \log_2(x) < e \leq \log_2(x)$$

$$e = \text{intero}[\log_2(x)]$$

$$1.m = x / 2^e$$

si converte 1.m in binario e si

scarta la cifra più significativa cioè l'1 implicito

$$\text{eg: } 1.m = 1.101011$$

$$m = 101011$$

L'algoritmo di conversione da reale a float 32 bit

Rappresentazione in virgola mobile secondo lo standard IEEE : esempi

Il numero decimale **1021** è rappresentato dalla tripla ($S; m; e$):

$$(0; 111\ 1111\ 0100\ 0000\ 0000\ 0000_2; 1000\ 1000_2).$$

- Il bit segno è 0 perché il numero è positivo.
- La rappresentazione binaria di **1021** è: **11 1111 1101₂** che, normalizzato e scritto con **23 bit**, da il secondo numero della terna moltiplicato per **2⁹**. Sono 9, infatti, le posizioni di cui è necessario spostare la virgola per la normalizzazione al formato

1,xyz

- Il numero di bit da usare per l'esponente è **8** e quindi:

$$P = 2^{E-1} - 1 = 2^7 - 1 = 127.$$

- L'esponente e si ricava invertendo la relazione di definizione della **costante di polarizzazione**: $e = e' + P = 9 + 127 = 136$ che, convertito in binario, da l'esponente della terna.

Rappresentazione in virgola mobile secondo lo standard IEEE : l'algoritmo in Matlab

<pre> % get exponent and mantissa for real % x using the IEEE normalization % (works only for standard values) function b=real2IEEE(x) s=0; if x<0 s=1; x=-x; end e=fix(log2(x)); m1=x/(2^e); bit_m=real2bin(m1,1,24); bit_m=bit_m(2:end); % discard first % bit (1.010001...-> 010001) E=e+127; bit_e=int2bin(E,8); b=[s bit_m bit_e]; </pre>	<pre> % conversion from real to binary with % np digits before comma and nm after % comma % get also remaining part function [b,resto]=real2bin(n,np,nm) %parte intera i=1; for j=np-1:-1:0 pwr=2^j; bit=floor(n/pwr); n=n-bit*pwr; b(i)=bit; i=i+1; end %parte frazionaria for j=-1:-1:-nm pwr=2^j; bit=floor(n/pwr); n=n-bit*pwr; b(i)=bit; i=i+1; end resto=n; </pre>
<pre> %function b=int2bin(n,nbits) nbits n. di bit n num. da convertire function b=int2bin(n,nbits) for j=1:nbits b(nbits-j+1)=n-2*floor(n/2); n=floor(n/2); end </pre>	

Operazioni in virgola mobile

La **moltiplicazione** fra due numeri N_1 ed N_2 rappresentati in virgola mobile dalle triple $(S_1; e_1; m_1)$ ed $(S_2; e_2; m_2)$ ha per risultato il numero rappresentato dalla tripla: $(S; e; m)$ in cui:

$$S = 0 \text{ se } S_1 = S_2 \text{ oppure: } S = 1 \text{ se } S_1 \neq S_2;$$

$$e = e_1 + e_2;$$

$$m = m_1 \times m_2$$

Dopo l'operazione di solito è necessaria la normalizzazione del risultato

La **divisione** si effettua con regole analoghe.

L'**addizione** e la **sottrazione** sono più complesse perché prima di effettuarle bisogna rendere **uguali gli esponenti**.

Durante questa operazione se i numeri sono uno molto grande ed uno molto piccolo, **per effetto dello scorrimento delle mantisse per pareggiare gli esponenti, si possono perdere cifre significative**.

Perdita di cifre significative in una somma

Attraverso quale meccanismo si possono perdere cifre significative effettuando una somma, può essere chiarito con un esempio che, per semplicità, può essere basato su una coppia di numeri decimali espressi attraverso una mantissa di 4 cifre ed un esponente di una sola cifra:

$$N_1 = 0,3435 \times 10^{-3} \text{ ed } N_2 = 0,9970 \times 10^{-5}.$$

Per effettuare la somma si può portare l'esponente di N_1 da 3 a 5. Ciò equivale a moltiplicare il fattore esponenziale per 100 e, contemporaneamente dividere per 100 la mantissa che diventa 0,0034, se si effettua un arrotondamento per difetto.

La somma dei due numeri risulta: $1,0004 \times 10^{-5}$, che dopo la normalizzazione diventa $0,1000 \times 10^{-6}$.

Codifica binaria di caratteri alfanumerici

I calcolatori, nonostante il nome italiano (in francese si chiamano "ordinatori") sono spesso utilizzati per manipolare informazioni non matematiche.

Tutto il mondo della moderna editoria si basa sull'uso di computer e, quindi, non ci può essere dubbio che il problema della **codifica binaria dei caratteri alfanumerici** sia un problema sentito.

Si parla di caratteri "*alfanumerici*" per sottolineare che in un testo possono essere presenti, oltre che caratteri alfabetici, anche numeri ed altro, come segni di punteggiatura, simboli particolari (£, &, @,) accenti di vario tipo, ecc..

Lo standard ormai universalmente adottato per applicazioni a basso livello (solo testo "*non formattato*") è la codifica **ASCII (da American Standard Code for Information Interchange)**.

Il codice utilizza le 128 combinazioni possibili di 7 bit, secondo le assegnazioni contenute nella seguente tabella.

Codice ASCII

	000	001	010	011	100	101	110	111
0000	<NUL>	<DLE>	spazio	0	@	P	'	p
0001	<SCH>	<DC1>	!	1	A	Q	a	q
0010	<STX>	<DC2>	"	2	B	R	b	r
0011	<ETX>	<DC3>	#	3	C	S	c	s
0100	<EOT>	<DCA>	\$	4	D	T	d	t
0101	<ENQ>	<NAK>	%	5	E	U	e	u
0110	<ACK>	<SYN>	&	6	F	V	f	v
0111	<BEL>	<ETB>	'	7	G	W	g	w
1000	<BS>	<CAN>	(8	H	X	h	x
1001	<HT>)	9	I	Y	i	y
1010	<LF>	<SUB>	*	:	J	Z	j	z
1011	<VT>	<ESC>	+	;	K	[k	{
1100	<FF>	<FS>	,	<	L	\	l	
1101	<CR>	<GS>	-	=	M]	m	}
1110	<SO>	<RS>	.	>	N	^	n	
1111	<SI>	<US>	/	?	O	-	o	

Sequenze di caratteri

Quando un sistema deve trasmettere, ad un altro che riceve, delle **informazioni articolate** che non corrispondono ad un singolo carattere o dato numerico, sorge il problema di come accompagnare le rappresentazioni di caratteri alfanumerici o dati numerici con le **informazioni necessarie per una loro corretta interpretazione.**

Se si trasmette **una successione di caratteri che costituiscono una parola**, l'unità ricevente deve avere anche **questo elemento d'informazione.**

Si può far **precedere** la sequenza di caratteri (stringa, dall'inglese "string") dall'informazione del numero di lettere che costituiscono la parola od inviare un carattere di controllo **dopo** l'ultima lettera.

Queste **"regole"** sulla trasmissione delle informazione costituiscono la base dei cosiddetti **"protocolli"** di trasmissione.

I protocolli più conservativi usano sia un **"header"** cioè un messaggio che precede il blocco di informazione che un **"footer"** cioè un messaggio che segue.