Sistemi di numerazione: generalità

Nel corso della storia sono stati introdotti diversi sistemi di numerazione, dettati di volta in volta dalle specifiche esigenze dei vari popoli. Poiché ogni numero maggiore di 1 può essere usato come base, vi è effettivamente una grande libertà nella scelta del sistema. Alcune culture, ad esempio, si sono avvalse di sistemi basati sui numeri 3, 4 o 5. I babilonesi adottarono un sistema sessagesimale, vale a dire basato sul numero 60, mentre i romani usavano, in particolare per alcuni scopi, un sistema duodecimale, basato sul numero 12. Il sistema numerico in uso presso i maya era quello vigesimale, basato sul numero 20. Infine il sistema binario era in uso presso alcune tribù e, insieme a quello basato sul numero 16, è il sistema tuttora impiegato nei computer.

I SISTEMI DI NUMERAZIONE

POSIZIONALI

- ai diversi simboli dell'alfabeto (cifre), viene associato un **valore crescente** in modo **lineare** da destra verso sinistra;
- il significato di un simbolo (il suo valore) dipende ordinatamente dalla sua posizione nella stringa

ESEMPIO:

Il sistema di numerazione decimale arabo:

383 = 300+80+3

3 x 100   8 x 10   3 x 1

significatività

NON POSIZIONALI

- Il significato dei simboli non dipende dalla loro posizione
- è stabilito in base ad una legge additiva dei valori dei singoli simboli (se posti in ordine crescente)

ESEMPIO :

Il sistema di numerazione romano

I = 1

V = 5

X = 10

L = 50

....

LXIV = 50+10 -1+5

= 64

50   10   -1   5

Sistemi di numerazione POSIZIONALI

Base= numero di simboli, o di cifre numeriche, richieste dal sistema stesso per rappresentare la serie infinita dei numeri.

Dato un alfabeto ordinato di b simboli distinti ($c_1, c_2, \ldots c_b$ che rappresentano rispettivamente i naturali $0, 1, 2, \ldots b-1$),

si rappresenti nel modo più semplice e compatto ogni altro numero $x >= b$ mediante una stringa di simboli dell'alfabeto

b = base del sistema di numerazione

b = 60 babilonesi

b = 20 ∨ 18 maya

b = 10 arabi (0,1,2,3,4,5,6,7,8,9)

b = 2 ∨ 8 ∨ 16 informatici (0,1)

(0,1,2,3,4,5,6,7)

(0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F)

$$101_{?} = \ ?$$

In quale base ?

Valore della posizione

La posizione di un simbolo all'interno di un numero indica la quantità che esso esprime, o più precisamente **l'esponente che bisogna dare alla base** per ottenere la quantità corretta.

Quantità di **0, 1, 2, 3, 4, 5, 6, 7, 8 e 9** dipende dalla **posizione** che ciascuno di essi assume all'interno del numero:

la prima cifra a destra rappresenta le unità, o il coefficiente di 100, la seconda le decine, 101, la terza le centinaia, 102, e così via.

il numero 3.098.323 è una rappresentazione abbreviata di

$(3 \times 10^6) + (0 \times 10^5) + (9 \times 10^4) + (8 \times 10^3) + (3 \times 10^2) + (2 \times 10^1) + (3 \times 10^0$, o $3 \times 1)$.

Il primo 3 (leggendo da destra a sinistra) rappresenta 3 unità; il secondo 3, sta per 300 unità, o 3 centinaia; infine il terzo 3, per 3 milioni di unità.

Sistemi di numerazione POSIZIONALI

$\alpha_{n-1}\ \alpha_{n-2}\ \ldots..\ \alpha_0\ .\ \alpha_{-1}\ \alpha_{-2}\ \ldots.\ \alpha_{-m}$ *Esempio: 354 . 21*

Si definisce BASE (o radice) "*b*" il numero che, elevato alle successive potenze "*i*" fornite dalla posizione, dà i rispettivi pesi b^i

$\boxed{\alpha_i * b^i = \text{valore di una cifra } \alpha_i \text{ nella stringa}}$

$$N = \alpha_{n-1} * b^{n-1} + \alpha_{n-2} * b^{n-2} + \ldots + \alpha_0 * b^0 + \alpha_{-1} * b^{-1} + \alpha_{-2} * b^{-2} + \ldots + \alpha_{-m} * b^{-m}$$

$$= \sum_{i=-m}^{n-1} \alpha_i * b^i$$

Esempi:

$354.21_{10} = 3*10^2 + 5*10^1 + 4*10^0 + 2*10^{-1} + 1*10^{-2}$

$1101.01_2 = 1*2^3+1*2^2+0*2^1+1*2^0 + 0*2^{-1}+1*2^{-2}$

Notazione Posizionale

richiede

b simboli diversi x rappresentare i numeri da 0 a b-1

decimale 0,1,2,3,4,5,6,7,8,9 (b=10)

binario 0,1 (b=2, base due)

ottale 0, 1, 2, 3, 4, 5, 6, 7 (b=8)

esadecimale 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A (dieci), B (undici), C (dodici), …, F (quindici). (b=16)

Es:

30.155 nel sistema in base sei è il numero

$(3 \times 6^4) + (0 \times 6^3) + (1 \times 6^2) + (5 \times 6^1) + (5 \times 6^0) = 3959$ nel s.d.;

2EF del sistema esadecimale è il numero

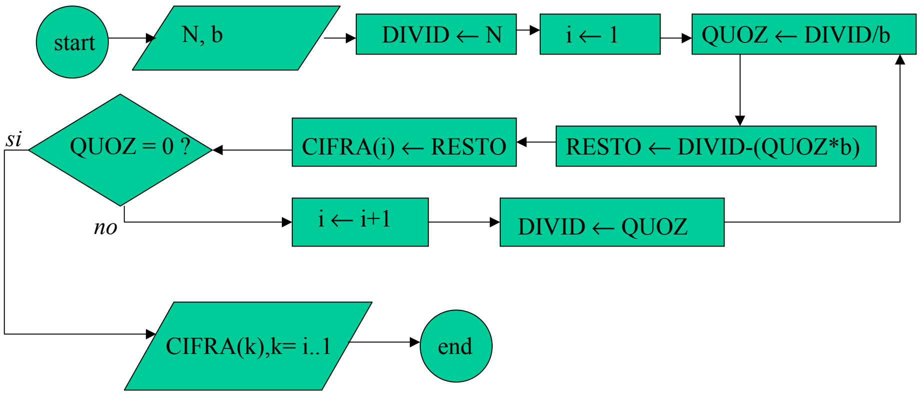
$(2 \times 16^2) + (14 \times 16^1) + (15 \times 16^0) = 751$ del s.d.

Conversione fra base 10 e base b

int(α_{10}) ⇒ β_b (parte intera)

La parte intera del numero decimale viene ripetutamente *divisa* per la base *b* e vengono via via registrati i *resti* delle divisioni ***da destra verso sinistra***

Metodo delle divisioni successive



Conversione decimale-binario

Esempio: convertire il numero 23 in base 2;

23:2 = 11 con resto 1

11:2 = 5 con resto 1

5:2 = 2 con resto 1

2:2 = 1 con resto 0

1:2 = 0 con resto 1

quoziente zero:

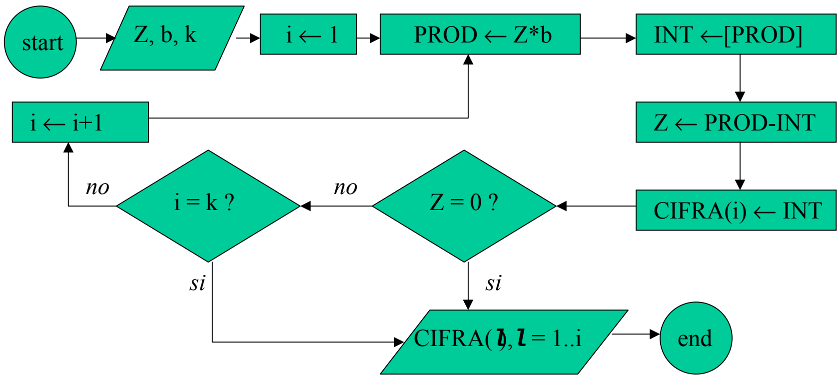
ciò ci dice che il procedimento è finito e che il risultato è dato dal numero 10111 in base 2.

Conversione fra base 10 e base b

$dec(\alpha_{10}) \Rightarrow \beta_b$ (parte decimale)

La parte decimale viene ripetutamente *moltiplicata* per la base *b*, registrando via via la *parte intera* del risultato della moltiplicazione ***da sinistra verso destra***.

Metodo delle moltiplicazioni successive



Conversione fra numeri con basi che siano l'una potenza dell'altra

$\alpha_b \Leftrightarrow \beta_{b^x}$

La conversione è effettuata raggruppando le cifre a *x* a *x* nelle due direzioni a partire dal punto di radice e convertendo così ogni singolo gruppo

1 9 4 8 . B 6 $_{16}$ — $16=2^4$

$0001100101001000.101101100_2$

1 4 5 1 0 . 5 5 4 $_8$ — $8=2^3$

Sistemi Digitali Binari: sono quei sistemi digitali i cui segnali sono limitati a 2 valori di regime

ø	1
L	H
low	high

Livelli di tensione elettrica

BIT *(binary digit)* = cifra binaria.
(unità di informazione elementare)

Il BIT ha due soli possibili valori, quindi il sistema di numerazione che si utilizza è un *SISTEMA DI NUMERAZIONE BINARIO*.

Tale sistema di numerazione si adatta perfettamente alle esigenze di un computer che può esprimere due soli stati, al pari di un interruttore *on/off*.

Lo stato "*on*" è per convenzione associato alla cifra 1 e "*off*" alla cifra 0.

In un computer gli stati *on/off* possono essere ad esempio determinati da:

- *sostanze magnetiche che presentano polarizzazioni opposte;*
- *sostanze conduttrici che fanno passare o meno la corrente;*
- *sostanze conduttrici che fanno invertire o meno il voltaggio della corrente;*
- *sostanze particolari che fanno passare o meno la luce*

Le unità di misura nell'informatica

(bit e byte)

1 bit (cifra che può assumere solo due valori, 0 / 1

8 bit =1 Byte = 1 carattere

L'aggregazione degli 8 **bit** necessari per definire un carattere alfanumerico viene definita **byte** e rappresenta l'**unità pratica principale** in informatica, in quanto permette di esprimere un singolo carattere alfanumerico.

I multipli del byte, espressi con 2 (il numero delle cifre nel sistema di numerazione binaria) elevato alle potenze di 10, sono :

2^{10} byte=1024 byte=Kilobyte=Kb
2^{20} byte=1024 Kbyte=Megabyte=Mb
2^{30} byte=1024 Mbyte=Gigabyte=Gb
2^{40} byte=1024 Gbyte=Terabyte=Tb

Il sistema binario svolge un ruolo importante nella tecnologia dei computer.

primi 20 numeri della notazione binaria sono

1, 10, 11, 100, 101, 110, 111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111, 10000, 10001, 10010, 10011, 10100

dove, procedendo da destra verso sinistra, i singoli simboli rappresentano i coefficienti delle potenze successive di due.

Ad esempio, cominciando da destra,

10101101 equivale a

(1 × 20) + (0 × 21) + (1 × 22) + (1 × 23) + (0 × 24) + (1 × 25) + (0 × 26) + (1 × 27) = **173**.

Semplificazione portata dalla base 2:

a parità di significato (valore) un numero scritto in base 2 è molto più lungo dell'equivalente scritto in base 10, ma le regole per poi eseguire la somma sono di gran lunga più semplici.

Nel caso dell'elaboratore questo è essenziale;

infatti, la velocità gli permette di non preoccuparsi eccessivamente della lunghezza dei numeri, mentre le regole relative alla somma delle coppie di cifre sono legate alla circuiteria elettronica che le deve eseguire, e un conto è complicare tale circuiteria per realizzare 100 regole e un altro è doverne realizzare solo 4.

Le operazioni aritmetiche

```
 1 1 1        riporti ( carry )
1 0 0 0 1 1 +
  0 1 1 1 0 =
--------------
1 1 0 0 0 1
      1  1  1
```

```
1          carry
3 F 5 +
1 6 A =
--------------
55 F
 1
```

Decimal	Binary	Octal	Hexadecimal
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14
30	11110	36	1E
40	101000	50	28
50	110010	62	32
60	111100	74	30
70	1000110	106	46
80	1010000	120	50
90	1011010	132	5A
100	1100100	144	64
200	11001000	310	C8
300	100101100	454	12C
400	110010000	620	190
500	111110100	764	1F4
600	1001011000	1130	258
700	1010111100	1274	2BC
800	1100100000	1440	320
900	1110000100	1604	384
1000	1111101000	1750	3E8
2989	101110101101	5655	BAD

Fig. A-3. Decimal numbers and their binary, octal, and hexadecimal equivalents.

La somma

L’algoritmo della operazione di somma non cambia qualunque sia la base considerata.

Le regole da imparare nel caso di una base b sono relative alle sole b^2 possibili combinazioni delle cifre da 0 a b-1.

100 regole della base 10 si riducono a 4 soltanto nella base 2. Eccole:

0+0 = 0

0+1 = 1

1+0 = 1

1+1 = 10

quest'ultima regola va intesa così:

il risultato di 1+1 è semplicemente 0, ma si ha anche il riporto di 1.

Il prodotto

Analogo alla somma è il prodotto.

Anche in questo caso l'operazione si riduce a conoscere il prodotto di ciascuna coppia di cifre.

Nel caso della base 2 tutto si riduce a ricordare l'esiguo numero di 2*2 regole, che sono:

$0*0 = 0$

$0*1 = 0$

$1*0 = 0$

$1*1 = 1$

In definitiva si hanno le regole seguenti:

- il prodotto per zero dà sempre come risultato zero;
- il prodotto per 1 dà sempre come risultato il numero stesso.

La sottrazione

Nella sottrazione, il ruolo del riporto è assunto dal "prendere in prestito" quando si debba sottrarre 1 da 0 e valgono regole anologhe a quelle viste per la somma.

Numeri binari negativi:

- **grandezza e segno** il bit + a sx indica il segno → **0** + / **1** -

- **complemento a uno** il bit + a sx indica il segno → **0** + / **1** -
  e si deve sostituire ogni 1 con uno 0 e ogni 0 con un 1.
  Anche il bit di segno - obsoleto.

- **complemento a due**: due fasi
  1. Ogni 1 diventa 0 e ogni 0 1 (compl a uno)
  2. + 1 al risultato

  se si verifica un riporto sul bit + a sx esso viene eliminato.

La sottrazione

A - B = A + (-B)

Rappresentazione dei numeri negativi (relativi)

- ***per grandezza e segno***

  Si riserva per il segno un bit degli n bit disponibili per il numero

  0 = positivo +3 = 00000011

  1 = negativo -3 = 10000011

- ***per complemento***

  Complemento a b di un numero in base b

$$\overline{N_b} = b_b^m - N_b \qquad \overline{23{,}7} = 10^2 - 23{,}7 = 100 - 23{,}7 = 76{,}3$$

$$\overline{010110} = \underbrace{1000000 - 010110} = 101010$$

1 0 1 0 0 1 +
1

m= numero intero di cifre di N

```
      1   1
0   10  10  10
1   0   0   0   1   1 -
    0   1   1   1   0 =
-----------------
0   1   0   1   0   1 (2)

    E  15
3   F  5 -
1   6  A =
--------------
2   8  B (16)
```

La sottrazione mediante la rappresentazione per complemento

a+b →	a + b	se a>0, b>0
	$(2^n - \lvert a\rvert) + b - 2^n$	se a<0, b>0
	$a + (2^n - \lvert b\rvert) - 2^n$	se a>0, b<0
	$\underbrace{(2^n - \lvert a\rvert)}_{a'} + \underbrace{(2^n - \lvert b\rvert)}_{b'} - 2*2^n$	se a<0, b<0

$$(a - b) = a + (2^n - b) - 2^n = \underbrace{a + b'}_{c} - 2^n$$

- *se **c > 0** si genera un bit di riporto per compensare -2n (**bit <u>da scartare</u>**)*
- *se **c < 0** non si genera un bit di riporto, ma il numero è da considerare già **rappresentato per complemento***
- *se **a' e b' sono < 0** il risultato è negativo e viene generato un bit di riporto oltre il bit del segno. Tale bit di riporto **deve essere <u>scartato</u>** ed il risultato è da considerare già **rappresentato per complemento**.*

Ad esempio :

23+	00010111 +
12	00001100
-----	------------
35	00100011

23-	00010111 +	
12	-00001100	= 11110011+
----		1
11		--------------
		11110100
	00010111+	
	11110100	

	~~1~~ 00001011	

- 23	- 00010111	= 11101000 +
+12	+00001100	1
----		--------------
- 11		11101001
	11101001+	
	00001100	

	11110101	

complemento di 11,
infatti: 00001010+1= 00001011

- 23	- 00010111 =	11101001+
- 12	- 00001100 =	11110100
----		--------------
- 35		~~1~~ 11011101

complemento di 35,
infatti : 00100010+1= 00100011

Note numeri binari: intervalli rappresentati

- Rappresentando gli interi positivi e lo zero in notazione binaria con n bit si copre l'intervallo $[0, 2^n-1]$
- Si sfruttano tutte le 2^n disposizioni

n =3	[0,7]
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

NB. Anche gli 0 non significativi devono essere rappresentati

Note numeri binari: interi positivi e negativi

• Per rappresentare gli interi relativi, a parità di cifre si *dimezza l'intervallo dei valori assoluti.*

Modulo e segno
– un bit per il segno 0: + 1: -
– n-1 bit per il modulo
- intervallo $[-2^{n-1}+1, +2^{n-1}-1]$

n =4 [-7,+7]

5 0101
-5 1101

NB
– *intervallo simmetrico*
– *doppia rappresentazione dello zero*

Note numeri binari: interi positivi e negativi

Complemento a 1
– un bit per il segno 0: + 1: -
– n-1 bit per il modulo
- intervallo $[-2^{n-1}+1, +2^{n-1}-1]$

Per cambiare di segno si complementa il numerale bit a bit

• È una *notazione posizionale*:
Pesi: $(-2^{n-1}+1)\ 2^{n-2} \ldots 2^{1}\ 2^{0}$

n =4 [-7,+7]

5 0101
-5 1010
(-7 + 2)

NB
– *complementare= cambiare il segno*
– *doppia rappresentazione dello zero*

Note numeri binari: interi positivi e negativi

Complemento a 2

I positivi hanno la stessa rappresentazione che in complemento a 1

- I negativi si ottengono sommando 1 alla loro rappresentazione in complemento a 1
- Intervallo con n bit: $[-2^{n-1}+1, +2^{n-1}-1]$
- Peso delle cifre: $(-2^{n-1}+1)\ 2^{n-2} \ldots 2^1\ 2^0$

- Regola pratica per complementare: *Partendo da destra si lasciano invariati tutti i bit fino al primo 1 compreso, e poi si complementa bit a bit*

n=4 bit intervallo [-8, +7]
5 = 0101
-5 = 1011 (-8+2+1)

NB
– *intervallo piu' esteso*
– *una sola rappresentazione dello zero*

Note numeri binari: addizioni binarie

Le addizioni fra numerali si effettuano cifra a cifra (come in decimale) portando 1 riporto alla cifra successiva

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0 con il riporto di 1

3 + 2 = 5

```
0011 +
0010 =
-------
0101
```

Se il numero di cifre non permette di rappresentare il risultato si ha un trabocco nella propagazione del riporto

Note numeri binari: addizioni binarie in CP2

In CP2 somme e sottrazioni tra numerali sono gestite nello stesso modo, *ma si deve ignorare il trabocco*:

Se i due operandi hanno segno diverso il risultato è sempre corretto:

4+ 2 = 6

```
0100 +
0010 =
-------
0110
```

4+ (-1) = 3

```
0001
1110 +
   1
1111
```

```
0100 +
1111 =
-------
0011
```

Se i due operandi hanno lo stesso segno e il risultato segno diverso c'è errore:

6+ 3 = 9

```
0110 +
0011 =
-------
1001
```

Rappresentazione dei numeri reali (1)

Con un numero finito di cifre è solo possibile rappresentare un numero razionale *che approssima con un certo errore* il numero reale dato

• Vengono usate due notazioni:

A) Notazione in virgola fissa

Dedica parte delle cifre alla parte intera e le altre alla parte frazionaria

+ XXX .YY

B) Notazione in virgola mobile

Dedica alcune cifre a rappresentare un esponente della base che indica l'ordine di grandezza del numero rappresentato

rappresentazione dei numeri in cui la gamma di numeri esprimibile sia indipendente dal numero delle cifre significative

Rappresentazione dei numeri reali (2)

Floating point (virgola mobile)

- Estende l'intervallo di numeri rappresentati a parità di cifre, rispetto alla notazione in *virgola fissa*
- Numeri reali rappresentati tramite una coppia di numeri <m,e>

$N = m \times b^e$

m= mantissa o frazione
e= esponente (numero positivo o negativo)

- Sia m che e hanno un numero fissato di cifre:
  - intervalli limitati
  - errori di arrotondamento

Esempi:
$3,14 = 0,314 \times 10^{10}$
$0,000001 = 0,1 \times 10^{-5}$
$1941 = 0,1941 \times 10^{?}$

Rappresentazione dei numeri reali (2)

Dato un numero reale non nullo x, esso può esprimersi nella forma $x = x' * b^e$

ad es. $\mathbf{19{,}72_{10} = 0.1972 * 10^2}$

*Rappresentazione dei numeri reali in **floating point***

+	2	*1972*

Segno della parte decimale

Caratteristica o esponente

Mantissa o parte decimale

- ***la precisione è determinata dalla lunghezza della mantissa***
- ***l'intervallo dei numeri rappresentabili è determinato dalla lunghezza della caratteristica***

Rappresentazione dei numeri reali (3)

Binario - rappresentato da una serie di bit suddivisi in 3 gruppi:

- 1 bit per il segno

- alcuni bit per l'esponente

- alcuni bit per la mantissa

0	**0101**	*10011101110*

$$19{,}72_{10} = 10011.101110_2 = 0.10011101110 * 2^5$$

Rappresentazione dei numeri reali (4)

0	**0101**	*10011101110*

$$19{,}72_{10} = 10011.101110_2 = 0.10011101110 * 2^5$$

I numeri in formato floating point si dicono **NORMALIZZATI**

se la prima cifra della mantissa (della parte decimale) è diversa da zero

0	**0110**	*010011101110*

Non normalizzato

$$0.010011101110 * 2^6$$

Rappresentazione dei numeri reali (5)

Per rappresentare il segno della caratteristica (dell'esponente) si usa una rappresentazione ***con eccesso*** 2^{m-1}

(m = numero di bit del dispositivo per la caratteristica),

N viene cioè rappresentato come $N + 2^{m-1}$,

ciò permette di rappresentare in un dispositivo il valore dell'esponente ed il suo segno, considerando valori ***positivi*** la metà superiore dei valori e ***negativi*** la metà inferiore.

L'eccesso corrisponde al semi-intervallo dei valori che possono essere assunti dalla caratteristica.

Ad es. : 7 bit per la caratteristica = 128 possibili configurazioni (da 0 a 127)
L'eccesso = 128/2= 64
Gli esponenti negativi =
le caratteristiche da -64 a -1 vengono rappresentate con 0...63.
Gli esponenti positivi =
le caratteristiche da 0 a 63 vengono rappresentate con 64…127.

Standard IEEE 754 (1985)

- Formato non proprietario cioè non dipendente dall'architettura
- Semplice precisione a 32 bit:

1	8	23
+/–	esp	mantissa

- Doppia precisione a 64 bit

1	11	57
+/–	esp	mantissa

- Notazioni in modulo e segno
- Alcune configurazioni dell'esponentesono riservate

Standard IEEE 754 (32bit)

1	8	23
+/–	esp	mantissa

- Esponente:
  - Rappresentato in eccesso 127
  - L'intervallo è [-127, +128]
  - Le due configurazioni estreme non si usano, quindi:

    $-126 \leq e \leq 127$

- **Mantissa:**

- È sempre *normalizzata*
- Se ne rappresenta *solo la parte frazionaria*

• Due rappresentazioni, *a seconda del valore dell'esponente*:

A) Numeri normalizzati $e \neq 00000000$

B) Numeri denormalizzati $e=00000000$

Rappresentazione dei numeri reali (4)

Le operazioni

La somma: per sommare due numeri reali in floating point occorre

1) innanzitutto ALLINEARE le caratteristiche (spostamento a destra della mantissa dell'operando con esponente minore con conseguente incremento della caratteristica, fino ad allineamento completo),
2) sommare le due mantisse,
3) normalizzare del risultato

Il Prodotto : per moltiplicare due numeri rappresentati in floating point occorre effettuare:

1) il prodotto delle mantisse
2) la somma delle caratteristiche
3) la normalizzazione del risultato

Rappresentazione dei numeri reali (5)

Le operazioni : esempio di somma

26,0+14,0 ⇒ 11010.0000 + 1110.0000

$0.1101000 * 2^5 + 0.11100000 * 2^4$

Rappresentazione in fp con eccesso 8 e mantissa a 7 bit

caratt.	segno	mantissa
1101	0	1101000
1100	0	1110000

\+

+1

Allineamento della caratteristica minore alla maggiore e shift a destra della mantissa

caratt.	segno	mantissa
1101	0	0111000

=

caratt.	segno	mantissa
1101	1	0100000

+1

Normalizzazione con recupero bit di overflow sul segno

Risultato finale:

caratt.	segno	mantissa
1110	0	1010000

6 + 0,1010000
101000,0 = 40

Rappresentazione dei numeri reali (6)

Le operazioni : esempio di prodotto

26,0 *14,0

Rappresentazione in fp con eccesso 16 e mantissa a 8 bit

caratt.	segno	mantissa
10101	0	11010000
10100	0	11100000

*

=

caratt.	segno	mantissa
11001	0	10110110

9 + 0,101101100
101101100,0 = 364

Somma delle caratteristiche

10101+
10100

101001 -16
11001

Prodotto delle mantisse =011010000 *
011100000 =

000000000
000000000
000000000
000000000
000000000
011010000
011010000
011010000
000000000

01011011000000000

Codice BCD

Binary Coded Decimal

Ogni cifra decimale del numero da codificare viene rappresentata mediante 4 bit che ne esprimono il valore binario

$$3\ 9\ 4_{10} = 0011\ 1001\ 0100_{BCD}$$

0011 1001 0100

$$= 110001010_2$$

In ***BCD*** ogni gruppo di 4 bit potrebbe rappresentare fino a $2^4 = 16$ simboli diversi (e non solo i dieci simboli del sistema decimale). Nelle operazioni aritmetiche è quindi necessaria una operazione di ***aggiustamento decimale*** per gestire le configurazioni ridondanti.

Codice BCD : esempi

BCD ***packed*** ⇨ 0011 1001 0100

BCD ***unpacked*** ⇨ 11110011 11111001 11110100

```
3 4 5 +        0011   0100   1010  +
1 8 6 =        0001   1000   0110
-------        ------------------------
5 3 1₁₀        0100   1100   1011  +
                      0110   0110
               ------------------------
               0101   0011   0001 BCD
```

Configurazioni NON significative (ridondanti) in BCD

Aggiustamento decimale per la somma = + 6

Codice ASCII

Per realizzare la corrispondenza tra il linguaggio umano e quello della macchina si è convenuto di utilizzare raggruppamenti di 8 **bit (1 byte)**

carattere alfanumerico		*cifra binaria*
A	⇔	11000001
a	⇔	11000010
B	⇔	10000100
1	⇔	11110001

Il codice universalmente adottato come standard per l'informatica è l'**ASCII** (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange).

Con questo codice si possono rappresentare 128 caratteri.

La sua versione più avanzata, attualmente in uso, è il codice **ASCII esteso** che permette la rappresentazione di 256 caratteri, (128 di base con le varianti relative alle diverse lingue).

Caratteristica comune ai due codici è la presenza dei primi 31 caratteri, chiamati **caratteri di controllo,** che producono un'azione e non sono stampabili

Le Tabelle ASCII

tabella ASCII

0	00		16	10	►	32	20		48	30	0	64	40	@	80	50	P	96	60		112	70	p
1	01	☺	17	11	◄	33	21	!	49	31	1	65	41	A	81	51	Q	97	61	a	113	71	q
2	02	☻	18	12	↕	34	22	"	50	32	2	66	42	B	82	52	R	98	62	b	114	72	r
3	03	♥	19	13	‼	35	23	#	51	33	3	67	43	C	83	53	S	99	63	c	115	73	s
4	04	♦	20	14	¶	36	24	$	52	34	4	68	44	D	84	54	T	100	64	d	116	74	t
5	05	♣	21	15	§	37	25	%	53	35	5	69	45	E	85	55	U	101	65	e	117	75	u
6	06	♠	22	16	▬	38	26	&	54	36	6	70	46	F	86	56	V	102	66	f	118	76	v
7	07	•	23	17	↨	39	27	'	55	37	7	71	47	G	87	57	W	103	67	g	119	77	w
8	08	◘	24	18	↑	40	28	(	56	38	8	72	48	H	88	58	X	104	68	h	120	78	x
9	09	○	25	19	↓	41	29	)	57	39	9	73	49	I	89	59	Y	105	69	i	121	79	y
10	0A	◙	26	1A	→	42	2A	*	58	3A	:	74	4A	J	90	5A	Z	106	6A	j	122	7A	z
11	0B	♂	27	1B	←	43	2B	+	59	3B	;	75	4B	K	91	5B	[	107	6B	k	123	7B	{
12	0C	♀	28	1C	∟	44	2C	,	60	3C	<	76	4C	L	92	5C	\	108	6C	l	124	7C	¦
13	0D	♪	29	1D	↔	45	2D	-	61	3D	=	77	4D	M	93	5D	]	109	6D	m	125	7D	}
14	0E	♫	30	1E	▲	46	2E	.	62	3E	>	78	4E	N	94	5E	^	110	6E	n	126	7E	~
15	0F	☼	31	1F	▼	47	2F	/	63	3F	?	79	4F	O	95	5F	_	111	6F	o	127	7F	⌂

tabella ASCII esteso

128	80	Ç	144	90	É	160	A0	á	176	B0	░	192	C0	└	208	D0	╨	224	E0	α	240	F0	≡
129	81	ü	145	91	æ	161	A1	í	177	B1	▒	193	C1	┴	209	D1	╤	225	E1	β	241	F1	±
130	82	é	146	92	Æ	162	A2	ó	178	B2	▓	194	C2	┬	210	D2	╥	226	E2	Γ	242	F2	≥
131	83	â	147	93	ô	163	A3	ú	179	B3	│	195	C3	├	211	D3	╙	227	E3	π	243	F3	≤
132	84	ä	148	94	ö	164	A4	ñ	180	B4	┤	196	C4	─	212	D4	╘	228	E4	Σ	244	F4	⌠
133	85	à	149	95	ò	165	A5	Ñ	181	B5	╡	197	C5	┼	213	D5	╒	229	E5	σ	245	F5	⌡
134	86	å	150	96	û	166	A6	ª	182	B6	╢	198	C6	╞	214	D6	╓	230	E6	µ	246	F6	÷
135	87	ç	151	97	ù	167	A7	º	183	B7	╖	199	C7	╟	215	D7	╫	231	E7	τ	247	F7	≈
136	88	ê	152	98	ÿ	168	A8	¿	184	B8	╕	200	C8	╚	216	D8	╪	232	E8	Φ	248	F8	°
137	89	ë	153	99	Ö	169	A9	⌐	185	B9	╣	201	C9	╔	217	D9	┘	233	E9	Θ	249	F9	∙
138	8A	è	154	9A	Ü	170	AA	¬	186	BA	║	202	CA	╩	218	DA	┌	234	EA	Ω	250	FA	·
139	8B	ï	155	9B	¢	171	AB	½	187	BB	╗	203	CB	╦	219	DB	█	235	EB	δ	251	FB	√
140	8C	î	156	9C	£	172	AC	¼	188	BC	╝	204	CC	╠	220	DC	▄	236	EC	∞	252	FC	ⁿ
141	8D	ì	157	9D	¥	173	AD	¡	189	BD	╜	205	CD	═	221	DD	▌	237	ED	φ	253	FD	²
142	8E	Ä	158	9E	₧	174	AE	«	190	BE	╛	206	CE	╬	222	DE	▐	238	EE	ε	254	FE	■
143	8F	Å	159	9F	ƒ	175	AF	»	191	BF	┐	207	CF	╧	223	DF	▀	239	EF	∩	255	FF	