



## Sistemi di numerazione

Il sistema di numerazione usuale è:

- posizionale (unità, decine, centinaia,...)
- decimale (cifre = 0,1,2,3,4,5,6,7,8,9)

In altre parole:

$$208 = 2 \cdot 10^2 + 0 \cdot 10^1 + 8 \cdot 10^0$$

Altri sistemi di numerazione di interesse sono anch'essi posizionali, ma con base diversa da 10:

sistema	base	cifre
binario	2	0 1
ottale	8	0 1 2 3 4 5 6 7
esadecimale	16	0 ... 9 A B C D E F



## Conversione decimale → base N

Si effettuano divisioni successive del numero dato per la base N; i resti delle singole divisioni, presi alla rovescia, rappresentano le cifre del numero nella base N.

Esempio: convertire  $107_{10}$  in base 2, 5 e 16.

107	53	26	13	6	3	1	0	← quozienti
1	1	0	1	0	1	1		← resti

107	21	4	0	← quozienti
2	1	4		← resti

107	6	0	← quozienti
11	6		← resti

I resti sono in base 10, quindi devono essere convertiti in base N:  $(11)_{10} = (B)_{16}$ .

Quindi  $107_{10} = (1101011)_2$   
 $(412)_5$   
 $(6B)_{16}$



## **Esercizi – conversione decimale → base N**

Convertire i seguenti numeri decimali nelle basi specificate:

- 345 in base 2 [R. 101011001]
- 345 in base 8 [R. 531]
- 345 in base 16 [R.159]
- 989 in base 5 [R.12424]
- 417 in base 7 [R.1134]
- 615 in base 9 [R.753]
- 426 in base 2 [R.110101010]
- 1042 in base 11 [R.868]



## **Esercizi – conversione decimale → base N**

Convertire i seguenti numeri decimali nelle basi specificate:

- 6666 in base 16 [R. 1A0A]
- 4596 in base 4 [R. 1013310]
- 687 in base 16 [R. 2AF]
- 595 in base 5 [R. 4340]
- 111 in base 2 [R. 1101111]
- 656 in base 5 [R. 1011]
- 811 in base 16 [R. 32B]
- 1101 in base 8 [R. 2115]



## Conversione base N → decimale

Partendo dalla cifra più significativa, si moltiplica la cifra per il valore della base, elevata alla potenza corrispondente alla posizione.

Esempio:

Convertire  $(302)_7$  in base 10:

La cifra meno significativa indica il coefficiente di  $7^0$ , quella più significativa il coefficiente di  $7^2$ , per cui:

$$\begin{aligned}(302)_7 &= 3 \cdot 7^2 + 0 \cdot 7^1 + 2 \cdot 7^0 \\ &= 3 \cdot 49 + 0 \cdot 7 + 2 \cdot 1 \\ &= 147 + 0 + 2 \\ &= 149\end{aligned}$$



## Esercizi – conversione base N → decimale

Convertire in base 10 i seguenti numeri espressi nelle basi indicati:

- $(1000101)_2$  [R. 69]
- $(477)_8$  [R. 319]
- $(40F)_{16}$  [R. 1039]
- $(3074)_5$  [R. impossibile]
- $(5778)_9$  [R. 4283]
- $(126)_9$  [R. 105]
- $(781)_{16}$  [R. 1921]
- $(3B8)_{13}$  [R. 658]



## Esercizi – conversione base N → decimale

Convertire in base 10 i seguenti numeri espressi nelle basi indicati:

- $(10010)_8$  [R. 4104]
- $(2EA)_{16}$  [R. 746]
- $(369F1)_{15}$  [R. impossibile]
- $(5669)_{11}$  [R. 7456]
- $(94598)_{10}$  [R. 94598]
- $(889)_{12}$  [R. 1257]
- $(1110)_3$  [R. 39]
- $(1357)_8$  [R. 751]



## **Conversione base N $\longrightarrow$ base M**

In generale conviene fare la conversione da base N a base 10, seguita dalla conversione da base 10 a base M.

Nel caso particolare in cui si debba passare dalla base 2 alle basi 8 o 16 ( o viceversa ), il calcolo è semplificato perché:

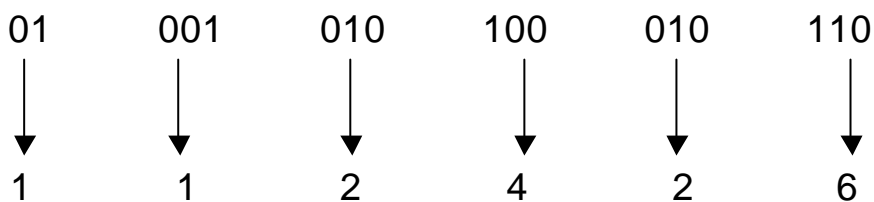
- ogni cifra ottale ( 0, ... , 7 ) è esprimibile nella corrispondente codifica binaria ( 000, ..., 111 ) su 3 cifre binarie
- ogni cifra esadecimale ( 0, ..., F ) è esprimibile nella corrispondente codifica binaria ( 0000, ..., 1111 ) su 4 cifre binarie



## Esempi – conversione base N $\longrightarrow$ base M

Convertire  $(01001010100010110)_2$  in ottale

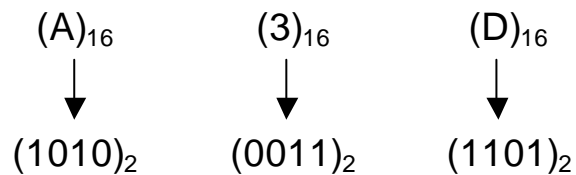
Partendo dalla cifre meno significativa si considerano la cifre rispettivamente a gruppi di 3:



Quindi:  $(01001010100010110)_2 = (112426)_8$

Convertire  $(A3D)_{16}$  in binario

Scriviamo le singole cifre esadecimali come numeri binari di 4 cifre:



Quindi:  $(A3D)_{16} = (101000111101)_2$



## Esercizi – conversione base N $\longrightarrow$ base M

Convertire i seguenti numeri nelle basi indicate:

- $(10010101001010)_2$  in base 8 [R. 22512]
- $(11110101101000)_2$  in base 16 [R. 3D68]
- $(13277)_8$  in base 2 [R. 1011010111111]
- $(B0E9)_{16}$  in base 2 [R. 1011000011101001]
- $(214)_5$  in base 2 [R. 111011]
- $(354)_7$  in base 6 [R. 510]
- $(821)_9$  in base 12 [R. 477]
- $(821)_9$  in base 8 [R. 1233]
- $(821)_9$  in base 16 [R. 29B]



## Esercizi – conversione base N $\longrightarrow$ base M

Convertire i seguenti numeri nelle basi indicate:

- $(AC29B)_{16}$  in base 8 [R. 2541233]
- $(34772)_8$  in base 16 [R. 39FA]
- $(1011)_9$  in base 13 [R. 44B]
- $(312)_{16}$  in base 4 [R. 30102]
- $(1492)_{11}$  in base 15 [R. 87B]
- $(C14)_{15}$  in base 16 [R. A9F]
- $(C14)_{15}$  in base 8 [R. 5237]
- $(558)_9$  in base 12 [R. 322]



## Addizioni e sottrazioni in binario

Si effettuano secondo le regole del sistema decimale, ossia sommando (sottraendo) le cifre di pari peso.

Si suppone di non avere limitazione sul numero di cifre binarie (bit) disponibili.

Esempio: effettuare la somma binaria  $11110 + 10100$

$$\begin{array}{r} 1 \quad 1 \quad 1 \\ \quad 1 \quad 1 \quad 1 \quad 1 \quad 0 \quad + \\ \quad 1 \quad 0 \quad 1 \quad 0 \quad 0 \\ \hline 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \end{array}$$

Esempio: effettuare la differenza binaria  $1011 - 0110$

$$\begin{array}{r} \quad \quad 1 \\ 1 \quad 0 \quad 1 \quad 1 \quad - \\ 0 \quad 1 \quad 1 \quad 0 \\ \hline 0 \quad 1 \quad 0 \quad 1 \end{array}$$



## Carry e Borrow

Come nelle usuali operazioni su numeri decimali, si può avere un *riporto* sul bit di peso immediatamente superiore, o un *prestito* dal bit di peso immediatamente superiore.

Nella numerazione binaria questi sono detti rispettivamente **carry** e **borrow**.

Le somme (differenze) bit a bit possono essere definite come segue:

$$0 + 0 = 0$$

$$1 + 0 = 1$$

$$0 + 1 = 1$$

$$1 + 1 = 0 \text{ (carry = 1)}$$

$$0 - 0 = 0$$

$$1 - 0 = 1$$

$$1 - 1 = 0$$

$$0 - 1 = 1 \text{ (borrow = 1)}$$



## Esercizi – somme e sottrazione in binario

Effettuare le seguenti operazioni in binario puro:

- $(34)_{10} + (77)_{10}$  [R. 11011111]
- $(225)_{10} + (63)_{10}$  [R. 100100000]
- $(229)_{10} + (111)_{10}$  [R. 101010100]
- $(10)_{10} + (6)_{10}$  [R. 100]
- $(39)_{10} + (14)_{10}$  [R. 11001]
- $(32)_{10} + (7)_{10}$  [R. 11001]
- $(84)_{10} + (37)_{10}$  [R. 101111]
- $(18)_{10} + (7)_{10}$  [R. 1011]
- $(25)_{10} + (15)_{10}$  [R. 1010]



## Overflow

Nel caso in cui si abbia un numero limitato di bit a disposizione (come avviene nella realtà), si possono avere due casi particolari:

- carry sul bit più significativo (MSB)
- borrow dal bit più significativo (MSB)

In entrambi i casi il numero di bit fissato non è sufficiente per rappresentare il risultato.

Questa condizione si dice condizione di **overflow**.



## Esempi – overflow

Considerando numeri binari di 4 bit, effettuare la somma  $9 + 7$

$$(9)_{10} = (1001)_2$$

$$(7)_{10} = (0111)_2$$

$$\begin{array}{rcccccc} & 1 & 0 & 0 & 1 & + \\ & 0 & 1 & 1 & 1 & \\ \hline 1 & 0 & 0 & 0 & 0 & \end{array}$$

Il risultato non è rappresentabile su 4 bit: overflow.

Considerando numeri binari di 4 bit, effettuare la differenza  $4 - 7$

$$(4)_{10} = (0100)_2$$

$$(7)_{10} = (0111)_2$$

$$\begin{array}{rcccccc} (1) & 0 & 1 & 0 & 0 & - \\ & 0 & 1 & 1 & 1 & \\ \hline & 1 & 1 & 0 & 1 & \end{array}$$

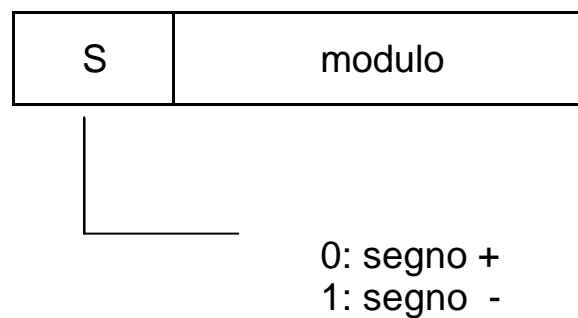
Il risultato non è rappresentabile su 4 bit: overflow



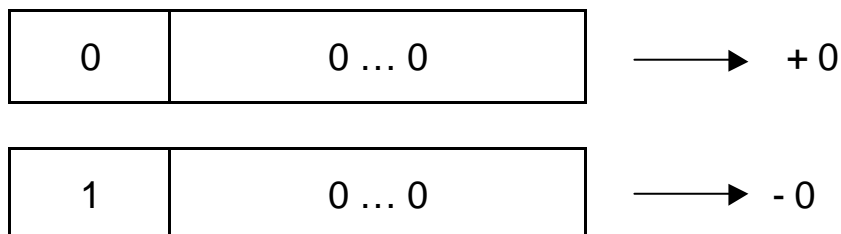
## Rappresentazione in modulo e segno

E' uno dei modi per rappresentare numeri interi relativi su un numero fissato di bit.

Dati N bit, il bit più significativo indica il segno, ed i restanti N – 1 il valore assoluto del numero, in binario puro.



In questa notazione esistono due rappresentazioni del numero 0:





## **Esercizi – modulo e segno**

Rappresentare in modulo e segno su 4 bit il massimo e minimo valore, le due rappresentazioni dello 0, i numeri + 5 e - 1.

Soluzione:

- minimo numero = 1111 = - 7
- massimo numero = 0111 = + 7
- - 0 = 1000
- + 0 = 0000
- +5 = 0101
- -1 = 1001



## Rappresentazione in complemento a due:

Conversione da decimale a complemento a due:

- se il numero è positivo, è rappresentato dal suo modulo con il bit di segno posto a zero (MSB)

0	modulo
---	--------

- se il numero è negativo:
  - si scrive il corrispondente numero positivo
  - si completano tutti i bit
  - si somma 1

Le ultime due operazioni rappresentano l'**operazione** di "complemento a due" di un numero binario.

Osservazioni:

- i numeri negativi hanno sempre bit di segno = 1
- esiste una sola rappresentazione dello 0: (0 ... 0)



## Rappresentazione in complemento a due

Un modo alternativo di eseguire il "complemento a due" di un numero binario è il seguente:

Considerando il numero da destra a sinistra eseguo le seguenti operazioni:

- copio tutti gli zeri fino a trovare il primo uno
- copio l'uno
- complemento i restanti bit

Esempio:

01 ... 01 ... 01	1	0 ... 0
------------------	---	---------

		0 ... 0
--	--	---------

	1	0 ... 0
--	---	---------

10 ... 10 ... 10	1	0 ... 0
------------------	---	---------



## Esempi – complemento a due

Rappresentare in complemento a due su 5 bit il numero  $(+8)_{10}$

- codifico il modulo in binario  $(+8)_{10} = (1000)_2$
- il numero è positivo, quindi il bit di segno è 0

$$(+8)_{10} = (01000)_{CA2}$$

Rappresentare in complemento a due su 5 bit il numero  $(-11)_{10}$

- codifico il modulo in binario  $(+11)_{10} = (01011)_2$
- complementando i bit si ha  $(10100)_2$
- sommando +1 (cioè 00001), si ha:

$$(-11)_{10} = (10101)_{CA2}$$



## Conversione da complemento a due a decimale

- Se il numero è positivo (MSB=0), si converte come un numero binario puro
- Se il numero è negativo (MSB=1):
  - si sottrae 1
  - si complementano tutti i bit
  - si converte il numero così ottenuto considerandolo come un numero binario puro

I primi due punti possono essere attuati anche applicando l'operazione di completamento a due al numero; infatti:

$$((numero)_{CA2})_{CA2} = numero$$



## **Conversione da complemento a due a decimale – esempio**

Scrivere il numero decimale corrispondente al numero 100101 in  
complemento a due su 6 cifre

Il numero  $(100101)_{CA2}$  è negativo, perciò:

- prima sottraggo 1, ottenendo  $(100100)_2$
- complementando i bit ottengo  $(011011)_2$
- converto considerando il numero in binario puro:

$$(011011)_2 = 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^1 + 1 \cdot 2^0 = (27)_{10}$$

Quindi il numero decimale corrispondente è  $-27$ .



## Limiti delle rappresentazioni

Fissato il numero  $N$  di bit disponibili, è possibile rappresentare i seguenti numeri:

binario puro  $0 \dots (2^N - 1)$

modulo e segno  $-(2^{N-1} - 1) \dots - 0 + 0 \dots + (2^{N-1} - 1)$

complemento a due  $-(2^{N-1}) \dots 0 \dots + (2^{N-1} - 1)$

Ad esempio su 5 bit è possibile rappresentare i seguenti numeri:

- in binario puro: da 0 a 31
- in modulo e segno: da -15 a +15
- in complemento a due: da -16 a +15



## Rappresentazione di oggetti

Per rappresentare  $N$  oggetti con numeri binari occorrono almeno  $K$  bit:

$$K = \lceil \log_2(N) \rceil$$

dove per  $\lceil X \rceil$  si intende il numero intero immediatamente superiore od uguale ad  $X$  (funzione *ceiling*).

Ad esempio, per rappresentare 77 oggetti, sono necessari almeno 7 bit ( $2^7 = 128$ ), dato che con 6 bit posso arrivare solo fino a  $2^6 = 64$ .



## Esercizi sulle rappresentazioni

Rappresentare in MS e in CA2 su 8 bit i seguenti numeri decimali:

- +12 [R. 0 0001100]
- -35 [R. 1 0100011]  
[R. 1 1011101]
- +40 [R. 0 0101000]
- -1 [R. 1 0000001]  
[R. 1 1111111]
- -128 [R. Impossibile]  
[R. 1 0000000]
- +271 [R. Impossibile]
- +127 [R. 0 1111111]



## Somma in complemento a due

Dati due numeri X e Y in complemento a due su N bit, la somma  $x + Y$  si calcola sommando aritmeticamente tutti i bit degli addendi, *compreso quello di segno*.

L'eventuale carry oltre il bit di segno viene tralasciato.

Esempio: calcolare  $(-7)_{10} + (10)_{10}$  in complemento a due su 5 bit

$$(-7)_{10} = 11001$$

$$(10)_{10} = 01010$$

$$\begin{array}{r} 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad + \\ 0 \quad 1 \quad 0 \quad 1 \quad 0 \\ \hline \boxed{1} \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \end{array}$$

Il carry nella posizione oltre il bit di segno non viene considerato ed il risultato è 00011, ossia  $(3)_{10}$ .



## Differenza in complemento a due

Ci si riconduce al caso della somma trasformando la differenza nella somma del primo numero con l'inverso del secondo.

Esempio: calcolare  $(-7)_{10} - (4)_{10}$  in complemento a due su 5 bit.

Si opera come se l'operazione fosse  $(-7)_{10} + (-4)_{10}$ :

$$\begin{aligned}(-7)_{10} &= 11001 \\ (-4)_{10} &= 11100\end{aligned}$$

$$\begin{array}{rcccccc} & 1 & 1 & 0 & 0 & 1 & + \\ & 1 & 1 & 1 & 0 & 0 & \\ \hline \boxed{1} & 1 & 0 & 1 & 0 & 1 & \end{array}$$

Tralasciando il carry oltre il bit di segno il risultato è 10101, ossia  $(-11)_{10}$



## Overflow in complemento a due

Si può verificare se contemporaneamente:

- i due addendi hanno segno concorde
- il segno del risultato è diverso dal segno dei due addendi

s1	
s1	
s2	



Overflow se e solo se  $s2 \neq s1$



## Esempio – overflow

Calcolare  $(-12)_{10} + (-6)_{10}$  in complemento a due su 5 bit:

$$(-12)_{10} = 10100$$

$$(-6)_{10} = 11010$$

$$\begin{array}{r} 1 \ 0 \ 1 \ 0 \ 0 \ + \\ 1 \ 1 \ 0 \ 1 \ 0 \\ \hline \boxed{1} \ 0 \ 1 \ 1 \ 1 \ 0 \end{array}$$

Si ha il carry oltre il bit di segno, che viene ignorato. Il bit di segno è diverso da quello degli addendi, cioè il risultato della somma di due numeri negativi sarebbe positivo, il che è impossibile (overflow).

Verifica: il risultato sarebbe  $(-18)_{10}$ , non rappresentabile in complemento a due su 5 bit.



## Esercizi – somma e sottrazione in complemento a due

Eseguire le seguenti operazioni in complemento a due su 9 bit:

- $(131)_{10} - (193)_{10}$

[R.  $(-62)_{10} = 111000010$ ]

- $(255)_{10} - (256)_{10}$

[R.  $(-1)_{10} = 111111111$ ]

- $(-127)_{10} + (-130)_{10}$

[R.  $(-257)_{10}$  overflow]

- $(255)_{10} + (2)_{10} + (-127)_{10}$

[R.  $(130)_{10} = 010000010$ ]

- $(-256)_{10} + (2)_{10}$

[R.  $(-254)_{10} = 100000010$ ]



## Numeri frazionari

Conversione da decimale a binario:

- si convertono separatamente parte intera e parte frazionaria
- per la parte intera si segue la procedura di conversione già vista
- per la parte frazionaria si effettuano moltiplicazioni successive per 2 separando la parte intera (0 o 1) da quella frazionaria così ottenuta, finché:
  - il risultato della moltiplicazione è 1000 ...
  - oppure si raggiunge la precisione richiesta

Esempio: convertire in binario  $(6.25)_{10}$

$$(6)_{10} = (110)_2$$

0.25	0.5	0.0	← parte frazionaria
0	1		← parte intera

$$(6.25)_{10} = (110.01)_2$$



## Approssimazione dei numeri frazionari

Si noti che in generale ad un numero frazionario decimale con un numero limitato di cifre può corrispondere un numero frazionario binario con un numero infinito di cifre. In tal caso ci si ferma quando si raggiunge la precisione desiderata.

Esempio: convertire in binario  $(0.3)_{10}$  con la precisione di almeno  $\frac{1}{100}$

0.3	0.6	0.2	0.4	0.8	0.6	0.2
0	1	0	0	1	1	0
↑	↑	↑	↑	↑	↑	↑
$\frac{1}{2^1}$	$\frac{1}{2^2}$	$\frac{1}{2^3}$	$\frac{1}{2^4}$	$\frac{1}{2^5}$	$\frac{1}{2^6}$	$\frac{1}{2^7}$

$$(0.3)_{10} = 0.0100110\dots$$

L'errore è minore di  $\frac{1}{100}$  perché  $\frac{1}{2^7} = \frac{1}{128} < \frac{1}{100}$ .



## Esercizi – numeri frazionari

Convertire in binario i seguenti numeri

- $(0.625)_{10}$  [R.  $(0.101)_2$ ]
- $(0.03125)_{10}$  [R.  $(0.00001)_2$ ]
- $(0.828125)_{10}$  [R.  $(0.110101)_2$ ]

Calcolare a quale numero decimale corrispondono i seguenti numeri binari frazionari:

- $(0.1101001)_2$  [R.  $(0.8203125)_{10}$ ]
- $(1011.0101)_2$  [R.  $(11.3125)_{10}$ ]
- $(11.10011)_2$  [R.  $(3.59375)_{10}$ ]

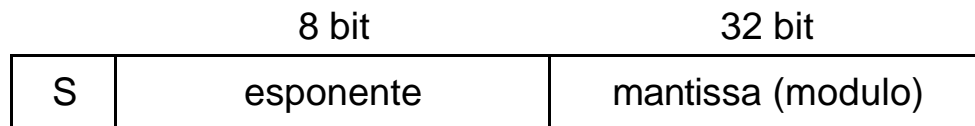


## Rappresentazione in floating-point

Utilizzata per rappresentare numeri frazionari nella notazione esponenziale:

$$\text{numero} = (\text{mantissa}) * 2^{\text{esponente}}$$

Il formato più utilizzato è quello IEEE P754, rappresentato su 32 bit nel seguente modo:



L'**esponente** è rappresentato come numero senza segno su 8 bit in eccesso 127, cioè i valori da -127 a +127 sono messi in corrispondenza con i valori da 0 a 254, per non dover gestire anche il segno dell'esponente.



## Floating-point: mantissa

La **mantissa** è codificata *in modulo e segno* su 24 bit;

- la mantissa è sempre normalizzata nella forma 1.XXXXX
- si rappresenta solo la parte frazionaria nei 23 bit meno significativi
- il peso del MSB del modulo è  $2^{-1}$
- il segno è dato dal MSB dei 32 bit

La rappresentazione di un numero è quindi nella forma:

$$\text{numero} = \pm 1.XXXXX \cdot 2^{(YYYY)_2}$$



## Conversione decimale → IEEE P754

Per trasformare un numero decimale  $N$  nella sua rappresentazione in floating-point, si deve:

- trasformarlo in binario
- trasformarlo nella forma normalizzata  $\pm 1.XXXXX \cdot 2^{(YYYY)_2}$
- il segno è 0 per i numeri positivi, 1 per i negativi
- l'esponente è pari a  $127 + n$ , dove  $n$  è il numero di posizioni di cui è stato spostato il punto decimale dalla forma binaria a quella normalizzata
- la parte frazionaria della mantissa normalizzata (XXXXX) si memorizza nei 23 bit meno significativi



## Esempio – floating-point

Convertire in formato floating-point IEEE P754 il numero  $13.25_{10}$

- il numero è positivo, per cui il segno è 0
- il numero convertito in binario puro è:

$$(1101.01)_2$$

spostando il punto decimale di 3 posizioni, il numero si normalizza in

$$(1.10101)_2 * 2^3$$

- l'esponente in eccesso 127 vale:

$$127 + 3 = 130 = (10000010)_2$$

La rappresentazione richiesta è:

$$0 \mid 10000010 \mid 101010000000000000000000$$



## Conversione IEEE P754 → decimale

L'interpretazione di un numero è piuttosto complicata: chiamando **s** il segno, **e** l'esponente, ed **m** la mantissa, si possono avere i seguenti casi:

- **e** = 0, **m** = 0: il valore è  $(-1)^s \cdot (0)$ , cioè +0 o -0
- **e** = 0, **m** ≠ 0: il numero è nella forma non normalizzata
- $0 < \mathbf{e} < 255$ : il numero è nella forma

$$(-1)^s \cdot 2^{(e-127)} \cdot (1.m)$$

- **e** = 255, **m** = 0: il valore è  $(-1)^s \cdot \infty$ , cioè un numero infinitamente grande o piccolo ( $+\infty$  o  $-\infty$ )
- **e** = 255, **m** ≠ 0: non è un numero valido (detto NAN, *Not A Number*); può essere usato per codificare informazioni di errore



## Esercizi – codifica IEEE P754

Rappresentare in formato IEEE P754 i seguenti numeri:

- $(5.0)_{10}$  [R. 0 10000001 0100 ... 0]
- $(-9.25)_{10}$  [R. 1 10000010 0010100 ... 0]
- $(12.8125)_{10}$  [R. 0 10000010 100110100 ... 0]

Dati i seguenti numeri in formato IEEE P754, dire a quale numero decimale corrispondono:

- 0 10000001 010010 ... 0 [R. 5.125]
- 1 10001101 01110 ... 0 [R. -23552.0]
- 0 01111101 01010 ... 0 [R. 0.328125]



## **Codici BCD**

E' una codifica per rappresentare numeri decimali in binario (*Binary Coded Decimal*).

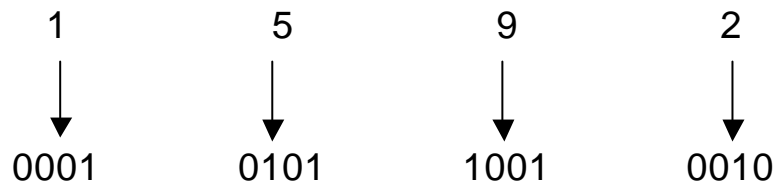
Il numero decimale viene suddiviso nelle cifre decimali che lo compongono, e ciascuna di queste convertita in binario puro secondo la corrispondenza:

cifra	codice
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001



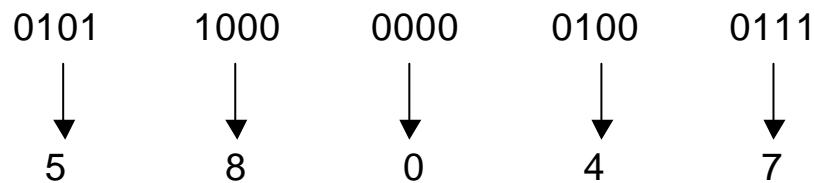
## Esempi – codici BCD

Esempio: convertire il numero 1592 in codice BCD.



Quindi  $1592_{10} = 0001010110010010_{\text{BCD}}$ .

Esempio: ricavare il numero decimale corrispondente al numero  $01011000000001000111_{\text{BCD}}$



Quindi  $01011000000001000111_{\text{BCD}} = 58047_{10}$ .



## Esercizi – conversione binario ↔ BCD

Convertire i seguenti numeri binari in BCD:

- 1000100100110111 [R. 8937]
- 100100000101001 [R. 4829]
- 100100011001010010001 [R. 123291]
- 101001010101100101 [R. 29565]

Convertire i seguenti numeri BCD in binario:

- 4171 [R. 100000101110001]
- 6153 [R. 110000101010011]
- 4261 [R. 100001001100001]
- 10478 [R. 10000010001111000]



## **Codifica dell'informazione non numerica**

I calcolatori possono intrinsecamente trattare solo informazioni binarie. Per trattare numeri relativi e frazionari abbiamo adottato particolari *codifiche* (modulo e segno, complemento a due, floating-point).

Più in generale con i numeri binari posso rappresentare insiemi *enumerabili* di oggetti, stabilendo una corrispondenza biunivoca tra oggetti e numeri (*codifica*).



## Codifica dell'informazione non numerica

Per rappresentare i caratteri alfabetici esistono alcune codifiche standard: ad esempio il codice ASCII (il più diffuso) codifica su 7 cifre binarie tutti i caratteri alfabetici più alcuni speciali.

Codice ASCII (ottale)															
000	nul	001	soh	002	stx	003	etx	004	eot	005	enq	006	ack	007	bel
010	bs	011	ht	012	nl	013	vt	014	np	015	cr	016	so	017	si
020	dle	021	dc1	022	dc2	023	dc3	024	dc4	025	nak	026	syn	027	etb
030	can	031	em	032	sub	033	esc	034	fs	035	gs	036	rs	037	us
040	sp	041	!	042	"	043	#	044	\$	045	%	046	&	047	'
050	(	051	)	052	*	053	+	054	,	055	-	056	.	057	/
060	0	061	1	062	2	063	3	064	4	065	5	066	6	067	7
070	8	071	9	072	:	073	;	074	<	075	=	076	>	077	?
100	@	101	A	102	B	103	C	104	D	105	E	106	F	107	G
110	H	111	I	112	J	113	K	114	L	115	M	116	N	117	O
120	P	121	Q	122	R	123	S	124	T	125	U	126	V	127	W
130	X	131	Y	132	Z	133	[	134	\	135	]	136	^	137	_
140	'	141	a	142	b	143	c	144	d	145	e	146	f	147	g
150	h	151	i	152	j	153	k	154	l	155	m	156	n	157	o
160	p	161	q	162	r	163	s	164	t	165	u	166	v	167	w
170	x	171	y	172	z	173	{	174	&	175	}	176	~	177	del

- Tradurre in codice ASCII : "Sta Laurel?"  
[R. 083 116 097 110 032 076 097 117 114 101 108 063]
- Tradurre da codice ASCII in caratteri alfabetici: "079 108 105 118 101 114 032 072 097 114 100 121 033"  
[R. Oliver Hardy!]



## Codifiche

Un codice deve permettere di risalire dalla codifica al simbolo rappresentato: per questo motivo una stringa binaria su un certo numero di bit ha più interpretazioni, a seconda della codifica che si considera.

Ad esempio, la stringa binaria 1101010 rappresenta:

- il numero 106 in binario puro
- il numero  $-42$  in modulo e segno
- il numero  $-22$  in complemento a due
- il carattere 'j' nella codifica ASCII



## Algebra Booleana

Opera su variabili che possono assumere soltanto i due valori 0 e 1 (variabili Booleane).

Operazioni base:

### AND

A	B	A · B
0	0	0
0	1	0
1	0	0
1	1	1

### OR

A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	1

### NOT

A	$\bar{A}$
0	1
1	0



## Algebra Booleana

Operazioni estese:

### NAND

A	B	$\overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

### NOR

A	B	$\overline{A + B}$
0	0	1
0	1	0
1	0	0
1	1	0

### EXOR

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

### EXNOR

A	B	$\overline{A \oplus B}$
0	0	1
0	1	0
1	0	0
1	1	1

N.B. E' possibile rappresentare qualunque espressione utilizzando esclusivamente l'operazione NAND oppure l'operazione NOR.



## Funzioni booleane

Funzioni di variabili booleane che assumono soltanto i valori 0 e 1.

Definite tramite *tabelle di verità* nelle quali si indicano il valore assunto dalla funzione per ogni possibile combinazione delle variabili.

Esempio:  $F(x_1, x_2)$  definita da

$x_1$	$x_2$	F
0	0	0
0	1	1
1	0	0
1	1	1

Normalmente rappresentate nella forma di *espressioni booleane*, in cui le variabili booleane sono combinate tramite i quattro operatori fondamentali:

Esempio:

$$F(x_1, x_2) = \overline{x_1} \cdot x_2 + x_1$$



## Costruzione di tabelle di verità

Per costruire la tabella di verità di un'espressione booleana:

- semplificare l'espressione mediante teoremi dell'algebra booleana, se possibile
- calcolare i termini parziali della funzione riducendoli alle operazioni fondamentali

Esempio: calcolare la tabella di verità della funzione

$$F(a, b, c) = a \cdot b + c$$

a	b	c	$a \cdot b$	$a \cdot b + c$
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	0	1
1	0	0	0	0
1	0	1	0	1
1	1	0	1	1
1	1	1	1	1



## Esercizi – tabelle di verità

Determinare le tabelle di verità delle seguenti funzioni:

- $f_1 = a(a + b)$

[R.  $f_1(a, b) \equiv a$ ]

- $f_2 = (ab + \bar{b})\bar{c}$

[R.

a	b	c	$a \cdot b$	$\bar{b}$	$ab + \bar{b}$	$\bar{c}$	$f_2$
0	0	0	0	1	1	1	1
0	0	1	0	1	1	0	0
0	1	0	0	0	0	1	0
0	1	1	0	0	0	0	0
1	0	0	0	1	1	1	1
1	0	1	0	1	1	0	0
1	1	0	1	0	1	1	1
1	1	1	1	0	1	0	0



## Esercizi – tabelle di verità

Determinare le tabelle di verità delle seguenti funzioni:

- $f_3(a, b, c) = \bar{a}\bar{b}\bar{c} + bc + ac$

[R.]

a	b	c	$\bar{a}\bar{b}\bar{c}$	$bc$	$ac$	$f_3$
0	0	0	1	0	0	1
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	1	0	1
1	0	0	0	0	0	0
1	0	1	0	0	1	1
1	1	0	0	0	0	0
1	1	1	0	1	1	1



## **Circuiti logici**

Una funzione booleana può anche essere rappresentata da un *circuito logico*, cioè da un insieme di *porte logiche* connesse fra loro.

Le porte logiche elementari corrispondono alle operazioni base dell'algebra booleana:



## Circuiti logici – esempio

Disegnare il circuito logico che rappresenta le funzioni:

- $F(a,b,c) = a \cdot b + c$

- $F(a,b,c) = \overline{a \cdot b} + \overline{b \cdot c}$



## Esercizi – circuiti logici

Disegnare i circuiti logici che rappresentano le seguenti funzioni:

- $f_1(a, b, c) = (a \cdot b + \bar{b}) \cdot \bar{c}$
- $f_2(a, b, c) = \bar{a} \cdot \bar{b} \cdot \bar{c} + b \cdot c + a \cdot c$
- $f_3(a, b, c) = (a \oplus \bar{b}) \cdot c$
- $f_4(a, b, c, d) = (a \cdot \bar{b} + c) \oplus (d + \bar{a})$
- $f_5(a, b, c) = (a \oplus b) \cdot \overline{(b \oplus c)}$
- $f_6(a, b, c, d) = ((a + \bar{b}) \oplus cd) \cdot (\bar{a}c + b)$



## **Esercizi – funzioni booleane**

Per essere ammessi all'orale di un esame universitario si devono aver superato almeno due scritti di esonero su tre.

Indicando simbolicamente con le variabili booleane A, B e C il superamento del primo, del secondo e terzo scritto:

- costruire la tavola di verità della funzione  $f(A,B,C)$  che rappresenta l'ammissione all'orale
- scrivere ed eventualmente semplificare le funzione  $f$

[R.  $BC+AC+AB$ ]



## **Esercizi – funzioni booleane**

Un impianto chimico è dotato di un sistema di allarme automatico che segnala le situazioni anormali. L'allarme suona quando risulta soddisfatta la seguente condizione:

la temperatura della caldaia è maggiore di 170°C e la pressione è superiore a 2 atmosfere, oppure non affluisce combustibile e la temperatura della caldaia è inferiore a 170°C.

Costruire una tabella di verità che indichi quando l'allarme è in funzione e ricavare la funzione corrispondente.

[R.  $xy + \overline{zx}$ ]



## Esercizi – funzioni booleane

Una cassaforte ha quattro lucchetti,  $a, b, c, d$ , che devono essere tutti contemporaneamente aperti affinché la cassaforte possa essere aperta.

Le chiavi sono distribuite fra tre persone,  $X, Y$  e  $Z$ , come segue:

$X$  possiede le chiavi  $a$  e  $d$

$Y$  possiede le chiavi  $b$  e  $d$

$Z$  possiede le chiavi  $a$  e  $c$

Le variabili  $X, Y$ , e  $Z$  siano uguali a 1 se la persona corrispondente è presente con le proprie chiavi, altrimenti siano uguali a 0.

Costruire la tavola di verità della funzione  $f(X, Y, Z)$  che è uguale a 1 se e solo se la cassaforte è aperta, e in seguito scrivere la funzione  $f$ .

[R.  $f=YZ$ ]



## Teoremi dell'algebra booleana

$$X \cdot 0 = 0$$

$$X \cdot 1 = X$$

$$X \cdot X = X$$

$$X \cdot \bar{X} = 0$$

$$X \cdot Y = Y \cdot X$$

$$\overline{X \cdot Y \cdot \dots \cdot Z} = \bar{X} + \bar{Y} + \dots + \bar{Z}$$

$$X \cdot Y + X \cdot Z = X \cdot (Y + Z)$$

$$X + X \cdot Y = X$$

$$X + \bar{X} \cdot Y = X + Y$$

$$X \cdot Y + X \cdot \bar{Y} = X$$

$$X + 1 = 1$$

$$X + 0 = X$$

$$X + X = X$$

$$X + \bar{X} = 1$$

$$X + Y = Y + X$$

$$\overline{X + Y + \dots + Z} = \bar{X} \cdot \bar{Y} \cdot \dots \cdot \bar{Z}$$

$$(X + Y) \cdot (X + Z) = X + Y \cdot Z$$

$$X \cdot (X + Y) = X$$

$$X \cdot (\bar{X} + Y) = X \cdot Y$$

$$(X + Y) \cdot (X + \bar{Y}) = X$$



## Semplificazione di espressioni booleane

Esempi: semplificare le seguenti espressioni:

$$\begin{aligned}f(x, y, z) &= \bar{x}y\bar{z} + \bar{x}yz + yz \\ &= \bar{x}y(z + \bar{z}) + yz \\ &= \bar{x}y + yz \\ &= y(\bar{x} + z)\end{aligned}$$

$$\begin{aligned}f(x, y, z) &= x + \bar{y} + \bar{x}y + (x + \bar{y})\bar{x}y \\ &= x + \bar{y} + \bar{x}y + (x + \bar{y})\bar{x}y \\ &= \bar{y} + 1 + (x + \bar{y})\bar{x}y \\ &= 1\end{aligned}$$

$$\begin{aligned}f(x, y, z) &= (x + \bar{y})\bar{x}y + \bar{z} \\ &= x\bar{x}y + \bar{y}\bar{x}y + \bar{z} \\ &= 0 \cdot y + 0 \cdot \bar{x} + \bar{z} \\ &= \bar{z}\end{aligned}$$



## Esercizi – semplificazione di espressioni

Semplificare le seguenti espressioni:

- $\overline{acd} + bcd + acd + ab + \overline{b}d$

[R.  $ab+d$ ]

- $(\overline{a} + c)ab\overline{c}$

[R. 0]

- $abc + \overline{b}\overline{c}d + b\overline{c} + \overline{b}ce + b\overline{c}f$

[R.  $ab + b\overline{c} + \overline{c}d + \overline{b}ce$ ]

- $(a + c)(\overline{a} + b + d)(b + c + \overline{d})$

[R.  $ab + \overline{a}c + bc + cd$ ]

- $\overline{a}bc + cdf + d\overline{e} + e\overline{f} + \overline{b}\overline{d}$

[R.  $\overline{b} + c + \overline{d} + \overline{e} + \overline{f}$ ]



## **Esercizi – verifica dell'equivalenza di espressioni booleane**

E' noto che in aritmetica vale la proprietà distributiva

$$a(b + c) = ab + ac$$

ma non:

$$a + (bc) = (a + b)(a + c)$$

Stabilire se le precedenti proprietà sono o non sono valide nell'algebra di Boole.



## Esercizi riassuntivi

1. Trovare le basi per le quali sono valide le seguenti operazioni:

$$\frac{210}{12} = 12$$

$$\sqrt{171} = 13$$

[R. 4 , 8]

2. Si eseguano le seguenti operazioni rappresentando gli operandi in complemento a due su 12 bit:

a)  $(181)_{10} - (1455)_{10}$

b)  $(747)_{10} + (1300)_{10}$

Si esprimano poi i risultati delle operazioni precedenti nel sistema esadecimale.

[R. a)  $101100000110 = (-1274)_{10} = B06_{16}$

b)  $011111111111 = (2047)_{10} = 7FF_{16}$

3. Qual è il numero minimodi cifre binarie necessarie q rappresentare in complemento a due il numero  $-87.46875$ ?

[R. 13]



4. Si eseguano le seguenti operazioni rappresentando gli operandi in complemento a due su 11 bit:

a)  $(9.0625)_{10} - (12.375)_{10}$

b)  $(14.750)_{10} + (17.375)_{10}$

[R. a)  $1111100.1011 = -3.3125$

b)  $0010000.001 = +32.125$ ]

5. Eseguendo il DUMP nel formato esadecimale di un file contenente numeri nella rappresentazione floating-point standard, si leggono i seguenti numeri:

41440000

3E800000

Di quali numeri si tratta?

[R.  $+12.25 \quad +0.25$ ]

6. Data la funzione booleana  $f(x, y, z, k)$  avente espressione

$$\overline{kx\bar{z}} \cdot (\bar{x} + xk + x + \bar{x}y)$$

scrivere l'espressione di  $\bar{f}(x, y, z, k)$  e semplificarla poi tramite i teoremi fondamentali.

[R.  $\bar{f} = kx\bar{z}$ ]



7. Data la funzione booleana  $f(x, y, z, k)$  avente espressione

$$\bar{x}y + \overline{\bar{x} + \bar{y} + z} + xzk + \bar{x}yz + \overline{\bar{x} + \bar{z} + k} + xyz$$

- ricavare la tabella di verità
- semplificare l'espressione utilizzando i teoremi fondamentali e ricavare le tabelle di verità di quest'ultima

[R.  $xz+y$ ]

8. Disegnare il circuito corrispondente alla funzione booleana  $f(x, y, z, w)$  avente espressione

$$\overline{(\bar{x} \cdot y + \bar{y} \cdot z \cdot \bar{y} \cdot z \cdot w)} \oplus \bar{w}$$