Informatica I

3 – Rappresentazione delle informazioni

21 Marzo 2013

Corso di Laurea in Matematica e applicazioni

Università di Camerino

A.A. 2012/2013

Agenda

- Rappresentazione binaria e conversioni
- Aritmetica binaria e floating point
- Algebra booleana e porte logiche

Rappresentazione binaria e conversioni

Informazione analogica vs digitale

Analogica

- Segnale continuo per cui la caratteristica che cambia nel tempo del segnale è una rappresentazione di qualche altra quantità (analoga) che cambia nel tempo.
  - Secondi (tempo) – angolo della lancetta dell'orologio.
  - Temperatura – mm del termometro
- Rappresentazione continua
- CONS: **disturbi!**

Digitale

- Usa valori discreti (numeri, Eng. *Digit*, Lat. *digitus*) per rappresentare le quantità
- Nei circuiti di un computer, segnali elettrici a basso voltaggio spengono i transitor (switch) -> 0
- Segnali ad alto voltaggio accendono i transitor -> 1

Rappresentazione delle informazioni digitali

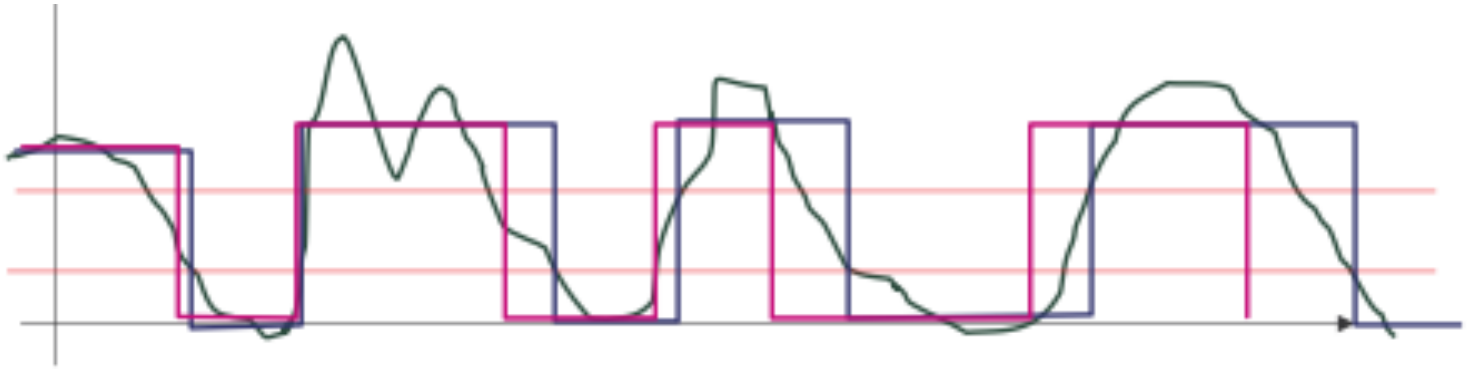
Alfabeto binario. {0,1}

Vantaggi:

-Efficienza: Operazioni binarie molto veloci

-Semplicità: Circuiti per aritmetica binaria semplici

-Affidabilità: si evitano i disturbi del segnale analogico



Il bit

- Unità fisica di informazione che vale 0 oppure 1.
- Il nome proviene da Binary Digit.

Suffisso	Valore	Approx. decimale
Kilobit (Kb)	2^{10}	10^3 (1024)
Megabit (Mb)	2^{20}	10^6
Gigabit (Gb)	2^{30}	10^9
Terabit (Tb)	2^{40}	10^{12}
Petabit (Pb)	2^{50}	10^{15}
Exabit (Eb)	2^{60}	10^{18}
Zettabit (Zb)	2^{70}	10^{21}
Yottabit (Yb)	2^{80}	10^{24}

Per kilobit (kb) con k minuscolo, il Sistema Internazionale intende proprio 1000 bit.
IEC (International Electrotechnical Commission) suggerisce di utilizzare il prefisso kibi (Ki) per indicare 1024 bit.
Wikipedia

Potenze del 2

- **Da sapere alla perfezione!**

2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}	…
1	2	4	8	16	32	64	128	256	512	1024	

Notazione posizionale

- Numeri a base **B** -> **B** simboli per digit
  - B=10 (decimali): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
  - B=2 (binari): 0, 1
- Rappresentazione
  - $d_{19}d_{18}...d_1d_0$: numero a 20 digit
  - Valore = $d_{19} \times B^{19} + d_{18} \times B^{18} + ... + d_1 \times B^1 + d_0 \times B^0$
- Binario:
  - $10110 = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = ...$
  - 1001101 = ...

Quanti digit?

In generale, un qualsiasi numero n si rappresenta con un numero di digit in base B pari a

$$\lceil \log_B n \rceil$$

- Es. 1456_{10}:
  - Base 10: $\lceil \log_{10} 1456 \rceil = \lceil 3{,}163 \rceil = 4$
  - Base 2: $\lceil \log_2 1456 \rceil = \lceil 10{,}507 \rceil = 11$

Quanti bit per un decimale?

- Trovare la minima potenza di 2 maggiore di n

  Es. $2^7 = 128 < 234 < 256 = 2^8$ -> 8 bit

- Per numeri più grandi:

  $x = 20.345.344_{10}$:

  $16 \times 10^6 < x < 32 \times 10^6$ -> 16M < x < 32M ->

  -> $2^{(20+4)} < x < 2^{(20+5)}$ -> 25 bit

  $x = 300.144.098_{10}$:

  ...

Quanti decimali per bit?

- 24 bit

  n = 20 + 4 -> $2^{20} * 2^4$ -> 1M * 16 -> 16M -> 8 digit

  2 6

- 16 bit

  n = 10 + 6 -> 1K * 64 -> 64K -> 5 digit

- 33 bit

  ...

Numero massimo rappresentabile con n digit in base B?

...

$B^n - 1$

Conversioni Binario -> Decimale

Somma pesata (metodo posizionale)

$(d_n d_{n-1} \ldots d_1 d_0)_2 \rightarrow d_n 2^n + \ldots + d_0 2^0$

Es. $100101_2 \rightarrow 2^0 + 2^2 + 2^5 = 37$

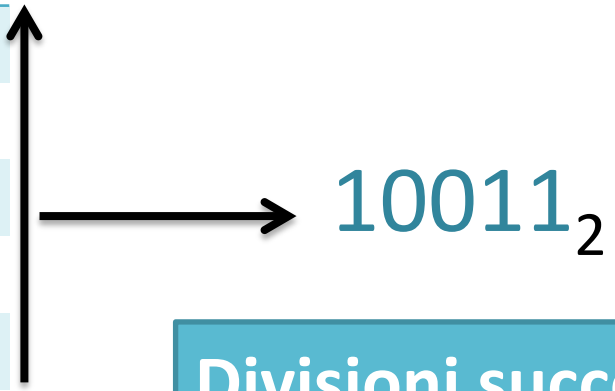
Es. $1100011001_2 \rightarrow \ldots$

Conversioni Decimale -> Binario

Metodo divisioni successive

Es. $19_{10} = ?_2$

Quoziente	Resto
19/2	1
9/2	1
4/2	0
2/2	0
1/2	1
0	•



10011_2

Divisioni successive si applicano per la conversione in qualsiasi base B

- $27_{10} = ?_2$ ...

Sistemi di numerazione intermedi

Decimale	Binario	Ottale	Esadecimale
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

Decimale	Binario	Ottale	Esadecimale
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

Ancora conversioni (1/3)

- Binario <-> Ottale (gruppi di 3 digit)

  Es. 1011110111_2

001	011	110	111
(1	**3**	**6**	**7)$_8$**

- Binario <-> Esadecimale (gruppi di 4 digit)

  Es. 1011110111_2

0010	1111	0111
(2	**F**	**7)$_{16}$**

Ancora conversioni (2/3)

- Ottale -> Decimale: somma pesata (posizionale)

  $(d_n d_{n-1} \ldots d_1 d_0)_8 \rightarrow d_n 8^n + \ldots + d_0 8^0$

  Es. 1367_8 -> $7 \times 8^0 + 6 \times 8^1 + 3 \times 8^2 + 1 \times 8^3 =$

  $= 7 + 48 + 192 + 512 = 759$

- Esadecimale-> Decimale: somma pesata (posizionale)

  $(d_n d_{n-1} \ldots d_1 d_0)_{16} \rightarrow d_n 16^n + \ldots + d_0 16^0$

  Es. $2F7_{16}$ -> $7 \times 16^0 + 15 \times 16^1 + 2 \times 16^2 = 7 + 240 + 512 = 759$

Trick:
Nella conversione da binario a decimale di numeri molto grandi, può convenire il passaggio intermedio a Hex, che di fatto divide in 4 il numero di digit.

Ancora conversioni (3/3)

Decimale -> Ottale

- $759_{10} = ?_8$

Quoziente	Resto
759/8	7
94/8	6
11/8	3
1/8	1
0	•

1367_8

Decimale -> Esadecimale

- $759_{10} = ?_{16}$

Quoziente	Resto
759/16	7
47/16	F
2/16	2
0	•

$2F7_{16}$

Byte

- **1 byte (B) = 8 bit (b)** -> 2^8 = 256 possibili valori

- Storicamente, il numero di bit necessari per codificare un carattere di testo, ma ancora molto utilizzato (es. capacità memorie)

- Nibble = 4 bit = ½ byte
- Word = 16 bit = 2 byte
- Double word = 32 bit = 4 byte
- Quad word = 64 bit = 8 byte

Rappresentazione dati alfabetici

- Un codice numerico per ogni carattere
- Codifiche standard:
  - **ASCII (American Standard Code for Information Interchange)**, 7 bit per carattere, rappresenta 128 caratteri
    - 33 caratteri di controllo, non stampabili
    - 95 caratteri stampabili (incluso lo spazio)
  - **UNICODE**, è uno standard più completo per supportare i sistemi di scrittura di tutto il mondo. Non è una codifica, ma è implementato con diversi encodings. I più usati:
    - **UTF-8**: encoding a 8 bit, lunghezza variabile. Compatibile con ASCII
    - **UTF-16:** enconding a 16 bit, lunghezza variabile

Esercizi

- $2345_{10} = ?_{16}$
- $2345_{10} = ?_{2}$
- $2345_{8} = ?_{2}$
- $2345_{8} = ?_{16}$

Aritmetica binaria e floating point

3 rappresentazioni per gli interi

- **Unsigned (senza segno)** -> tutti i bit disponibili si utilizzano per la rappresentazione della quantità (non servono bit per rappresentare il segno)

- **Rappresentazioni con segno** (1 bit si "spreca" per rappresentare il segno):
  - **Modulo + segno**
  - **Complemento a 2**

Left shift

Y = X << k, k ≥ 1

- Si scorre a sinistra di k posizioni
- Se non c'è overflow, $X << k = X\ 2^k$

Overflow quando il bit che scorre fuori è 1

Es. X = 0110

Y = X<<1

Y = 1100

No Overflow

Y = X<<2

Y = 1000

Overflow

Right shift

$Y = X >> k, k \geq 1$

- Si scorre a destra di k posizioni
- Equivale alla divisione intera di X su 2^k, $X >> k = X/2^k$

Es. X = 0110

Y = X>>1
Y = 0011

Y = X>>2
Y = 0001

Somma di binari unsigned

Numero bit = 4

Z = X + Y

- 3 + 7

r:	0	1	1	1
X:	**0**	**0**	**1**	**1**
Y:	**0**	**1**	**1**	**1**
Z:	**1**	**0**	**1**	**0**

No Overflow

- 12 + 7

r:	1	1	0	0
X:	**1**	**1**	**0**	**0**
Y:	**0**	**1**	**1**	**1**
Z:	**0**	**0**	**1**	**1**

Overflow

Overflow quando il riporto del bit più significativo (MSB = most significative bit) è 1

Sottrazione di binari unsigned

Numero bit = 4

Z = X - Y

- 13 - 9

r:	**0**	0	0	0
X:	**1**	**1**	**0**	**1**
Y:	**1**	**0**	**0**	**1**
Z:	**0**	**1**	**0**	**0**

No Overflow

- 11 - 13

r:	**1**	1	0	0
X:	**1**	**0**	**1**	**1**
Y:	**1**	**1**	**0**	**1**
Z:	**1**	**1**	**1**	**0**

Overflow

Overflow quando il riporto del MSB è 1

Esercizi unsigned

Numero bit = 8

- C7 + 2D = ?
- 4A – B9 = ?

Modulo + segno (1/3)

- Il MSB codifica il segno:
  - 0 -> positivo
  - 1 -> negativo
- Assumendo una rappresentazione a 4 bit…

0000	**+0**	**1**000	**-0**
0001	**+1**	**1**001	**-1**
0010	**+2**	**1**010	**-2**
0011	**+3**	**1**011	**-3**
0100	**+4**	**1**100	**-4**
0101	**+5**	**1**101	**-5**
0110	**+6**	**1**110	**-6**
0111	**+7**	**1**111	**-7**

Operazioni su M+S

- Bisogna tener conto del segno degli operandi. Il segno va stabilito a priori

Esempio: -19 + 13 (8 bit)

• L'operando più grande in valore assoluto è 19: il suo sarà il segno del risultato

• Sottrazione: -(19 – 13)

0010011 –
0001101 =
10000110

Esempio: -43 – (-24) (8 bit)

•Sottrazione: -(43 – 24)

0101011 –
0011000 =
10010011

Esercizi M+S

Numero bit = 8

- C7 + 2D = ?
- 4A – B9 = ?

Modulo + segno (2/3)

- Sia X un signed integer a n bit
- Massimo numero rappresentabile:

  $2^{n-1} - 1$
- Minimo numero rappresentabile:

  $1 - 2^{n-1}$
- Minimo numero in valore assoluto:
  - 000... = +0
  - 100...= -0

PROBLEMA (1)! Doppia rappresentazione dello zero!

Modulo + segno (3/3)

PROBLEMA (2)! Aritmetica più difficile

Es. Contando da 0000 a 1111

- Da 0000 a 0111 -> si aggiunge 1
- Da 1000 a 1111 -> si sottrae 1

SOLUZIONE (1,2)?

Complemento a 2

Definizione

Sia X un binario a n bit.

- Il complemento a 1 di X, $f_1(X) = 2^n - 1 - X$
- Il complemento a 2 di X, $f_2(X) = 2^n - X = f_1(X) + 1$

$f_1(X)$ si ottiene complementando ogni bit di X. E semplicemente $f_2(X)$ si ottiene aggiungendo 1 a $f_1(X)$.

Es. n = 4, X = 9

X	1	0	0	1
$f_1(X)$	**0**	**1**	**1**	**0**
$f_2(X)$	**0**	**1**	**1**	**1**

Proprietà $f_2(X)$

SOTTRAZIONE

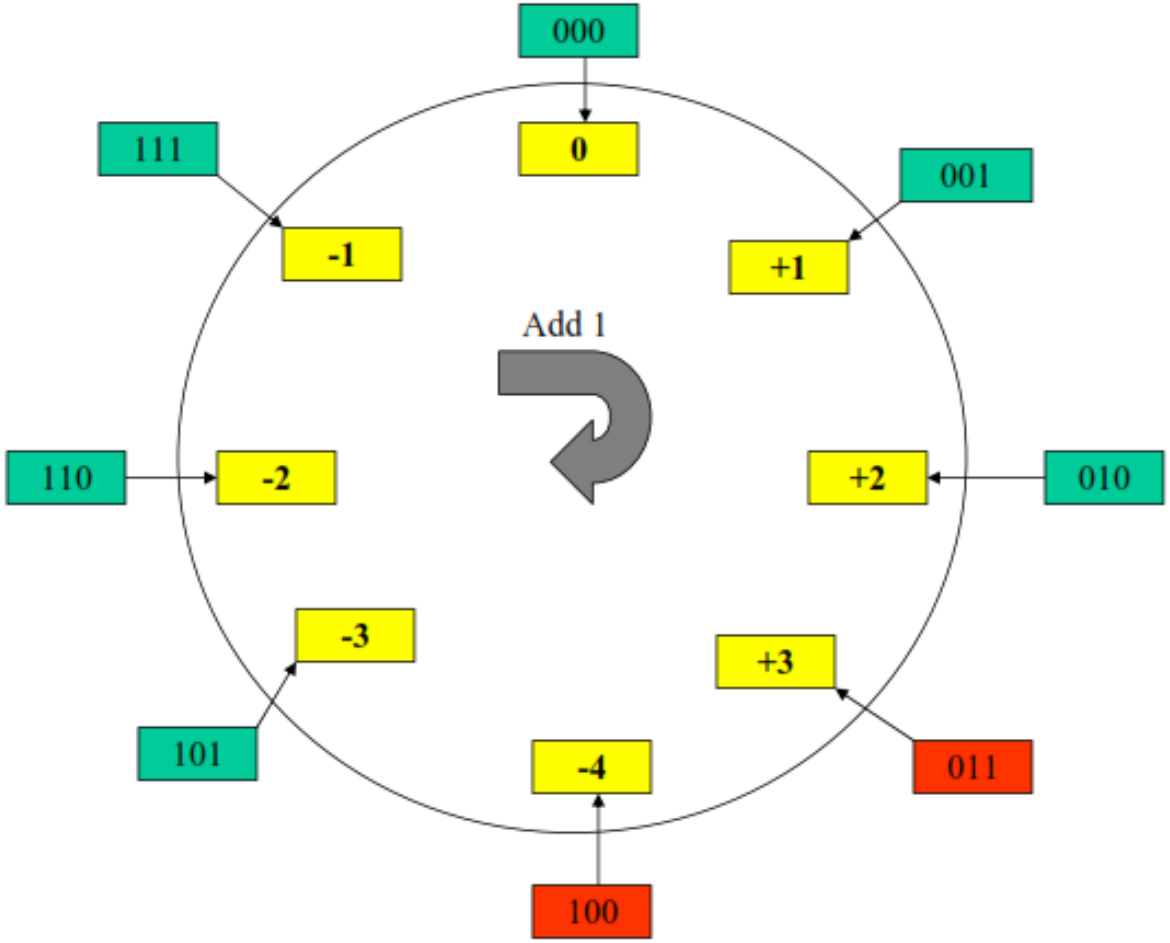
- Dati X, Y, Z binari unsigned a n bit
- Calcolare $Z = (X - Y) \bmod 2^n$

$$Z = (X - Y + 2^n) \bmod 2^n = (X + \mathbf{f_2(Y)}) \bmod 2^n$$

Sommiamo ad X il complemento a 2 di Y!

Possiamo sfruttare il complemento a 2 per rappresentare I numeri negativi?

Rappresentazione Complemento a 2



- Il MSB indica sempre il segno
- Si aggiunge sempre 1 (aritmetica consistente)
- Non c'è spreco (doppio zero)
- Range: $[-2^{n-1}, 2^{n-1} -1]$
- Un intero non negativo è rappresentato dal suo valore assoluto
- Un intero negativo è rappresentato dal complemento a 2 del suo valore assoluto

Complemento a 2 (Proprietà)

- **f2(f2(X)) = X**

- **Estensione del segno:**

Dato $X = X_{n-1} \ldots X_0$ rappresentato in complemento a 2 con n bit.
La rappresentazione a n+k bit si ottiene replicando il bit del segno (X_{n-1}) nei bit successivi ($X_n - X_{n+k-1}$)

Esempio:
In 4 bit, -5 è 1011. In 8 bit, -5 è rappresentato da (1111 1011).

Complemento a 2 (Esempio)

Numero bit = 8

X = + 124

X ≥ 0

| x | -> **0111 1100**

X = - 36

X < 0

|X| -> **0010 0100**

$f_2(X)$ -> **1101 1100**

X = 0101 0001

X ≥ 0 (MSB = 0)

X = +81

X = 1101 0001

X < 0 (MSB = 1)

|X| = f2(1101 0001) = 0010 1111 = 47

X = - 47

Somma/Sottrazione complemento a 2

- Non va determinato il segno a priori (come su M+S)
- Somma/sottrazione SEMPLICI:
  - Dati due interi X e Y con segno rappresentati in complemento a 2 (**siano essi positivi o negativi**),
  - X + Y si ottiene con una banale somma (come se fossero senza segno)

Somma/Sottrazione complemento a 2

Numero bit = 5

Z = X + Y

- 13 + (- 9)

r:	1 =	1	1	1	1
X:	0	1	1	0	1
Y:	1	0	1	1	1
Z:	0	0	1	0	0

No Overflow

- -11 + (- 13)

r:	1 ≠	0	1	1	1
X:	1	0	1	0	1
Y:	1	0	0	1	1
Z:	0	1	0	0	0

Overflow

Overflow quando il riporto nel bit di segno è diverso dal riporto nel bit fuori dal segno

Esercizi Complemento a 2

Numero bit = 8

- C7 + 2D = ?
- 4A – B9 = ?

Per gli scettici…

Consideriamo due valori espressi con complemento a 2 a n bit:
$X=X_nX_{n-1}\ldots$ e $Y=Y_nY_{n-1}\ldots$ in cui quindi gli n-esimi bit rappresentano il segno.
Denotiamo con C_{outn} e C_{outn-1} i riporti fuori e sul bit di segno.

Verifichiamo che la regola per l'overflow è corretta, nel caso di X+Y

- **$X_n = 0$ e $Y_n = 0$. ($\rightarrow C_{outn} = 0$)** Significa che sia X che Y sono positivi, e quindi si ha overflow con la regola della somma unsigned, ovvero quando il riporto fuori dal MSB (in questo caso C_{outn-1}) è 1. Quando invece $C_{outn-1} = C_{outn} = 0$ non c'è overflow.
- **$X_n \neq Y_n$.** Significa che X e Y hanno segno opposto. Questo implica che il risultato non ha bisogno di più di n-1 bit per essere rappresentato → no overflow. Infatti in questo caso: **$C_{outn-1} = 0 \rightarrow C_{outn} = 0$ e $C_{outn-1} = 1 \rightarrow C_{outn} = 1$**
- **$X_n = 1$ e $Y_n = 1$. ($\rightarrow C_{outn} = 1$).** Significa che sia X che Y sono negativi, e quindi si ha overflow quando **$C_{outn-1} = 0$,** ovvero quando la somma dei loro moduli (complementati) causa un riporto = 1 fuori dal MSB.

Rappresentazione floating point

Per rappresentare quantità non intere nel formato **segno**, **mantissa**, **esponente**, **base**. es.

- 2.2 x 10^{-23}

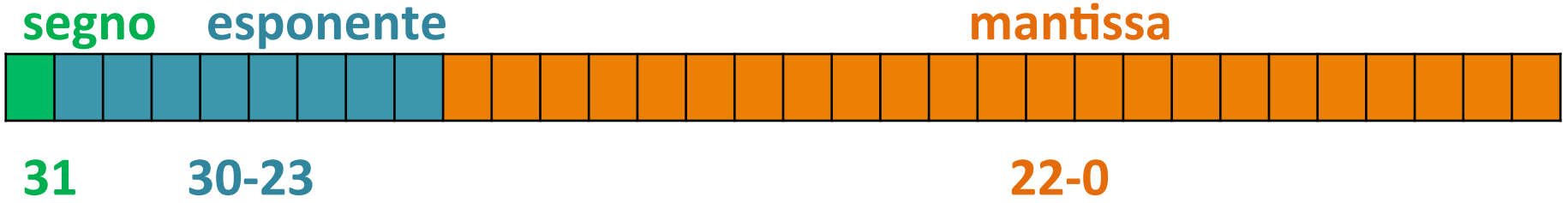
- Bisogna utilizzare una **rappresentazione unica e normalizzata**, per evitare rappresentazioni multiple

Es. 0.312 x 10^7, 312 x 10^4, 31.2 x 10^5, ... ⟶ **3.12 x 10^6**

IEEE 754 Floating Point Format (1/3)

- Single precision format (32 bit)
- Double precision format (64 bit)

- **Segno:** 0 positivo; 1 negativo
- **Base:** 2 implicitamente



Single precision: suddivisione dei campi

IEEE 754 Floating Point Format (2/3)

- Il numero rappresentato è $(-1)^{s}\ (1.m) \times 2^{e-127}$

  Numeri speciali:
  - e=0, m=0 rappresenta 0
  - e=0, m ≠ 0 rappresenta i *numeri denormalizzati* (ovvero tutti quelli maggiori di 0 e minori del minimo normalizzato in valore assoluto)
  - e=255, m=0 rappresenta +-inf
  - e=255, m ≠ 0 rappresenta NaN (not a number, es. n/0)

- L'esponente è in codice eccesso-127. Quindi con $1 \leq e \leq 254$, l'esponente varia tra -126 e +127

Floating point – Max e min

- Massimo normalizzato

e=254, m=111...111, s=0, valore= $(1 + 1 - 2^{-23}) \times 2^{254-127} = (2 - 2^{-23}) \times 2^{127}$

- Minimo normalizzato

e=254, m=111...111, s=1, valore= $(2^{-23} - 2) \times 2^{127}$

- Minimo (valore assoluto) normalizzato
- e=1, m=000...000, valore= 1.0×2^{-126}

(Binari non interi)

Nello stesso modo del sistema decimale, possiamo esprimere valori binari non interi considerando le potenze negative del 2.

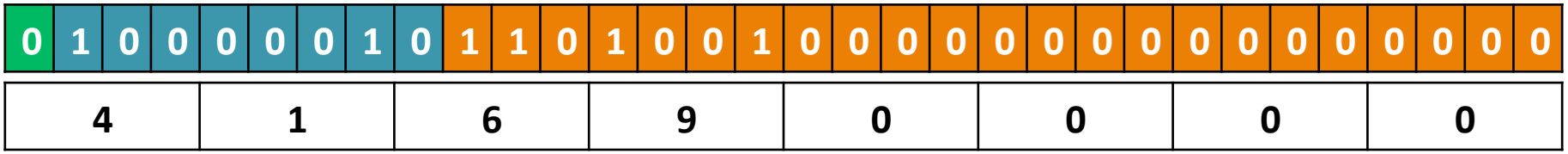
Es. $(11.101101)_2 = (2^1 + 2^0 + 2^{-1} + 2^{-3} + 2^{-4} + 2^{-6})_{10}$
$= 3.6890625$

IEEE 754 Floating Point Format (3/3)

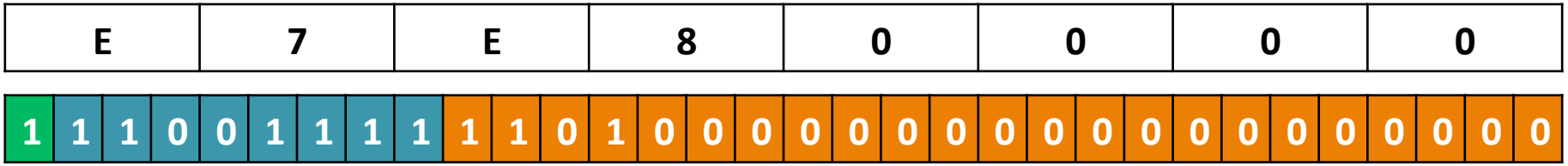
Esempio 1:

$14.5625\ (14 + 2^{-1} + 2^{-4}) \rightarrow (1110.1001)_2 = 1.1101001 \times 2^3$

$s=0$, $e= 3 + 127 = 130 = (10000010)_2$, $m= (1101001)_2$



Esempio 2:



$s=1$, $e=(11001111)_2 = 207 =$ **80** $+ 127$, $m= (1101)_2 = 2^{-1} + 2^{-2} + 2^{-4} = 13/16$

$=$ **0.8125** $\rightarrow$ **-** **1.8125** $\times$ $\mathbf{2^{80}}$

Esercizi (individuali)

1 - Rappresentare in floating point 32 bit il numero 71.75

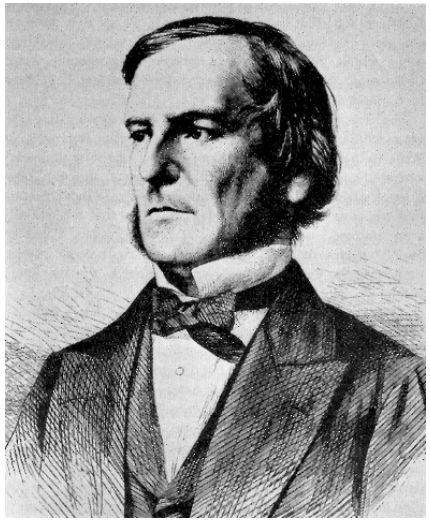
2 – Valore della rappresentazione a 32 bit di 7A010000

Algebra booleana e porte logiche

Algebra booleana

L'algebra booleana o di Boole (detta "a due valori " o "switching") è definita da *(B, +, ., ')*:

- valori nel **dominio booleano B={0,1}**
- 3 operazioni: **not** (', complementazione), **and** (., prodotto), e **or** (+,somma)



George Boole

Proprietà dell'algebra booleana

Proprietà	AND	OR
Idempotenza	**x.x = x**	**x+x=x**
Associatività	**x.(y.z) = (x.y).z**	**x+(y+z)=(x+y)+z**
Distributività	**x+(y.z) = (x+y).(x+z)**	**x.(y+z)=(x.y)+(x.z)**
Commutatività	**x.y = y.x**	**x+y=y+x**
Assorbimento	**(x+y).x=x**	**x+x.y=x**
Elementi neutri	**1.x = x**	**0+x=x**
Negazione	**x.x'=0**	**x+x' = 1**
Elementi forzanti	**x.0=0**	**x+1=1**
Legge di De Morgan	***(x.y)'=x'+y'***	***(x+y)' = x'.y'***

Porte logiche

- Una **porta logica (o gate)** è una funzione $f: B^n \rightarrow B^k$

•Quindi attraverso una porta logica è possibile descrivere **arbitrarie operazioni** su n-uple di valori booleani

•Ogni porta logica può essere descritta con una tabella a 2^n righe e $n+k$ colonne, in cui ogni riga da il valore della funzione per ogni combinazione delle variabili in input.

•Questa tabella è detta **tabella di verità**

NOT Gate

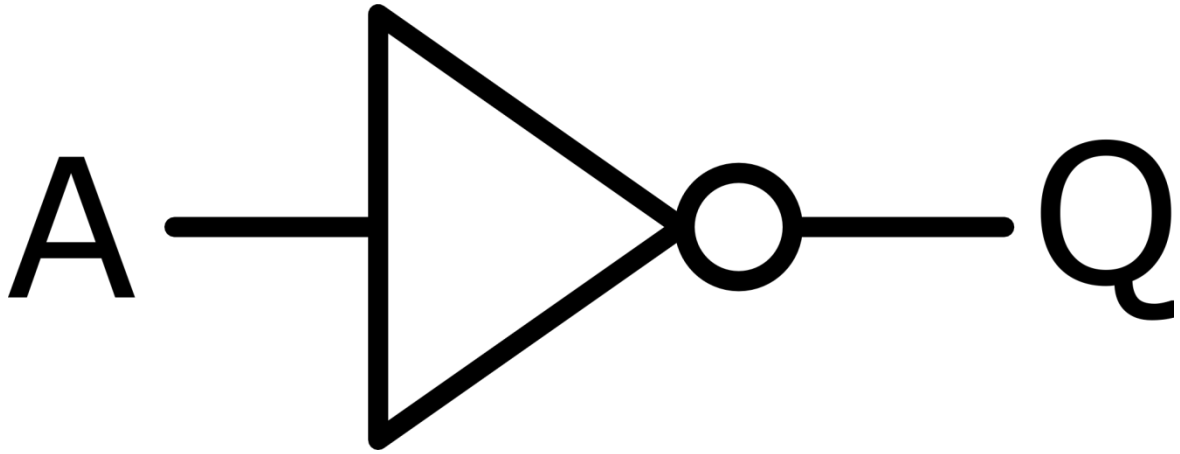


Tabella verità

A	Q
0	1
1	0

Espressione booleana

$Q = A'$

AND Gate

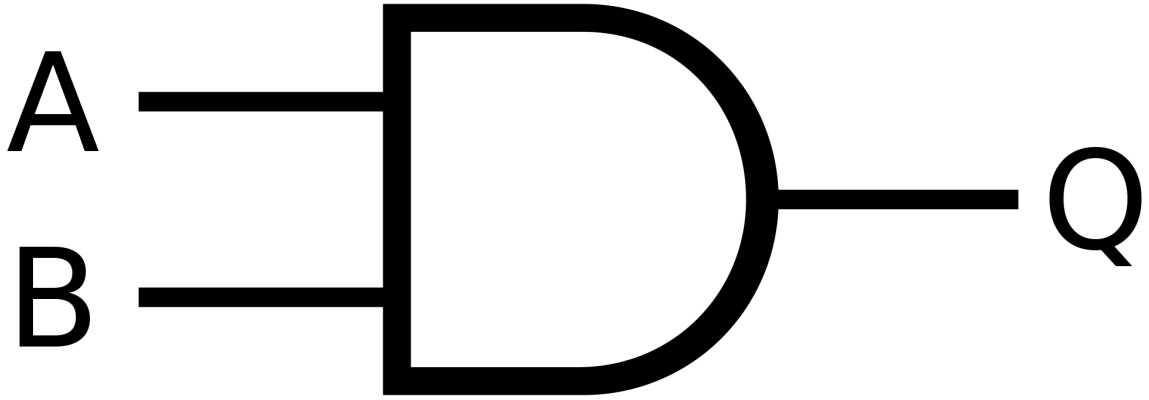


Tabella verità

A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

Espressione booleana

$Q = A.B$

OR Gate

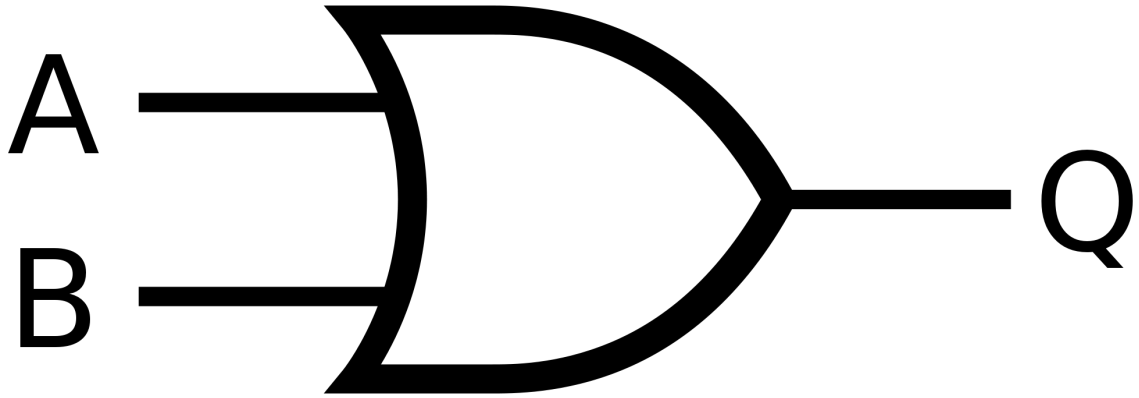


Tabella verità

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

Espressione booleana

$$Q = A + B$$

NAND Gate

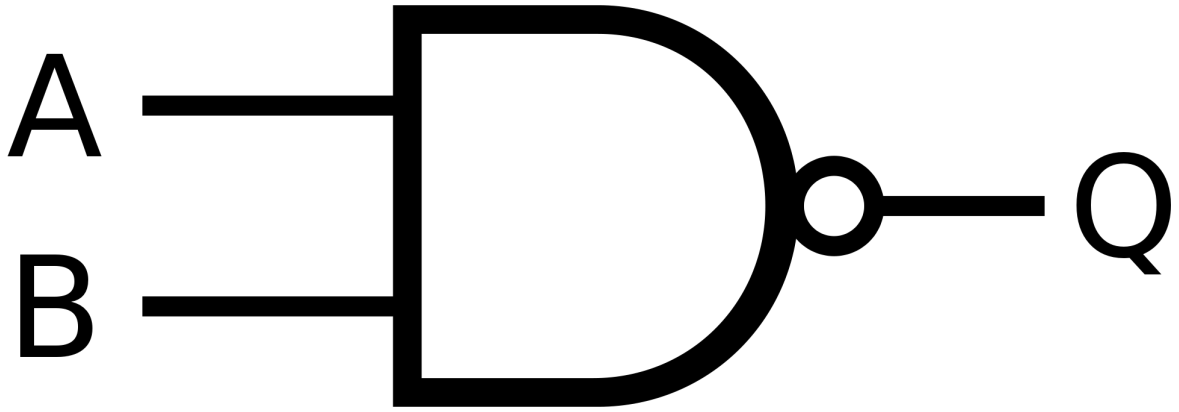


Tabella verità

A	B	Q
0	0	1
0	1	1
1	0	1
1	1	0

Espressione booleana

$Q = (A.B)'$

NOR Gate

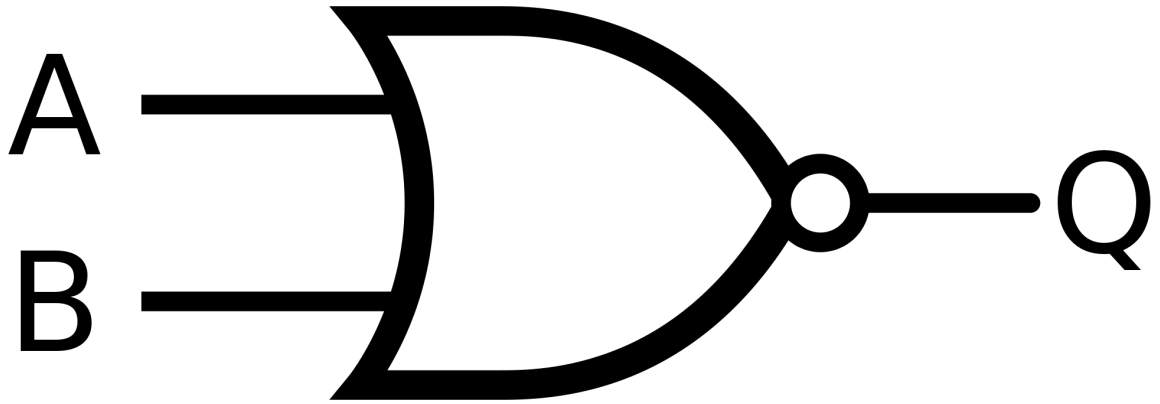


Tabella verità

A	B	Q
0	0	1
0	1	0
1	0	0
1	1	0

Espressione booleana

$Q = (A + B)'$

Esercizi

- Dimostrare **Assorbimento** con le altre proprietà

(x+y).x = x.x + y.x = x + y.x = x.(1 + y) = x.1 **= x** (dist, idem, dist, el forzante, el neutro)

x+x.y = x.(1+y) = x.1 **= x** (dist, el forzante, el neutro)

- Ridurre l'espressione **x.x + x.x'**

x.x + x.x' = x + 0 **= x**

- Ridurre l'espressione **x.y + x.y.z**

x.y + x.y.z = x.y.(1 + z) = x.y.1 **= x.y** (distributività e elemento forzante), oppure

x.y + x.y.z = x.y (assorbimento)

- Ridurre l'espressione **x.(x' + y) + y**

x.(x' + y) + y = x.x' + x.y + y = 0 + y **= y**

Esercizio

Verifichiamo la legge di de Morgan confrontando le tabelle di verità

NOR(A,B) → $(A + B)' = A'.B'$

A	B	(A+B)'	A'	B'	A'.B'
0	0	1	?	?	?
0	1	0	?	?	?
1	0	0	?	?	?
1	1	0	?	?	?

Esercizio

Verifichiamo la legge di de Morgan confrontando le tabelle di verità

NAND(A,B) → (A.B)' = A'+B'

A	B	(A.B)'	A'	B'	A' + B'
0	0	1	?	?	?
0	1	1	?	?	?
1	0	1	?	?	?
1	1	0	?	?	?

XOR Gate

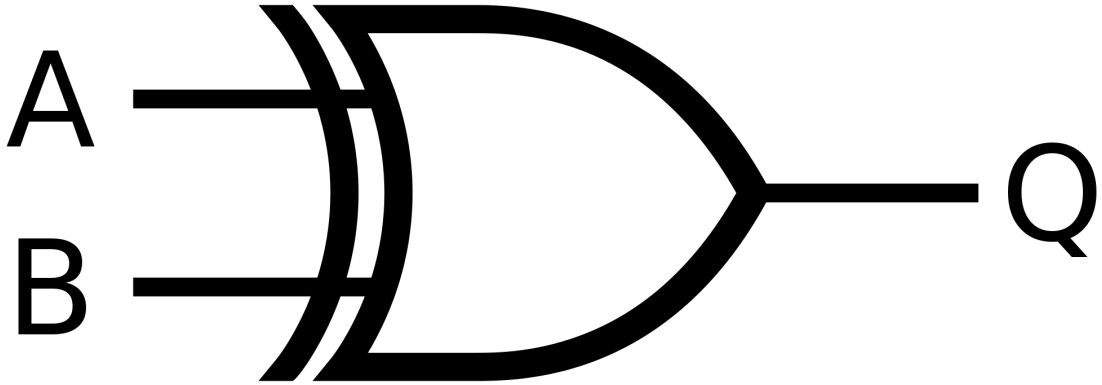


Tabella verità

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0

Espressione booleana

$Q = A.B' + A'.B$

Half Adder?

Qual'è la tabella di verità per un adder che restituisce somma e riporto?

A	B	S	R
0	0	?	?
0	1	?	?
1	0	?	?
1	1	?	?

Half Adder?

Qual'è la tabella di verità per un adder che restituisce somma e riporto?

A	B	S	R
0	0	**0**	**0**
0	1	**1**	**0**
1	0	**1**	**0**
1	1	**0**	**1**

Full Adder?

Qual'è la tabella di verità per un adder che restituisce somma e riporto e che considera il riporto in ingresso?

A	B	R_{in}	S	R_{out}
0	0	0	?	?
0	0	1	?	?
0	1	0	?	?
0	1	1	?	?
1	0	0	?	?
1	0	1	?	?
1	1	0	?	?
1	1	1	?	?

Full Adder?

Qual'è la tabella di verità per un adder che restituisce somma e riporto e che considera il riporto in ingresso?

A	B	R_{in}	S	R_{out}
0	0	0	**0**	**0**
0	0	1	**1**	**0**
0	1	0	**1**	**0**
0	1	1	**0**	**1**
1	0	0	**1**	**0**
1	0	1	**0**	**1**
1	1	0	**0**	**1**
1	1	1	**1**	**1**

?