Informatica Generale

Linguaggio C

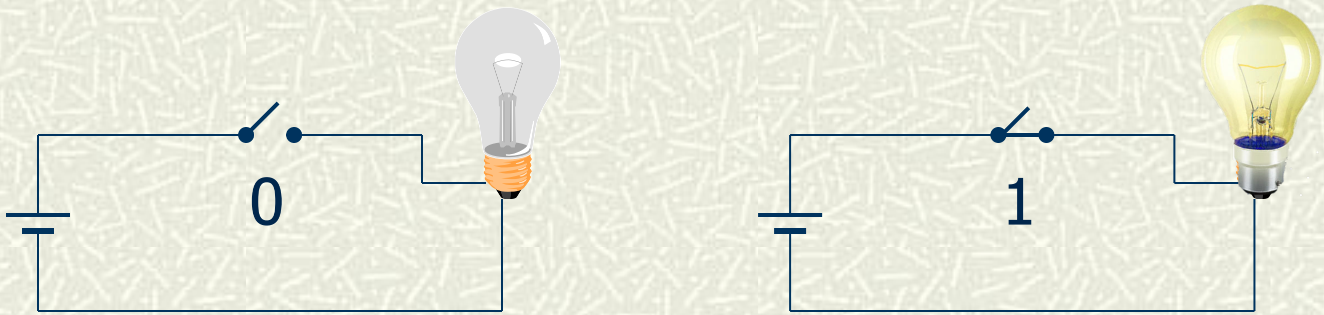
L'algebra dei Calcolatori

- Rappresentazione dell'informazione
  - Algebra booleana
  - Circuiti AND, OR, NOT
- Rappresentazione dei numeri interi
  - Valore assoluto
  - Modulo e segno
  - Complemento A2
- Rappresentazione dei numeri in virgola mobile

L’algebra di Boole

Algebra Booleana

- Contempla due costanti 0 e 1 (falso e vero)
- Corrispondono a due stati che si escludono a vicenda
- Possono descrivere lo stato di apertura o chiusura di un generico contatto o di un circuito a più contatti

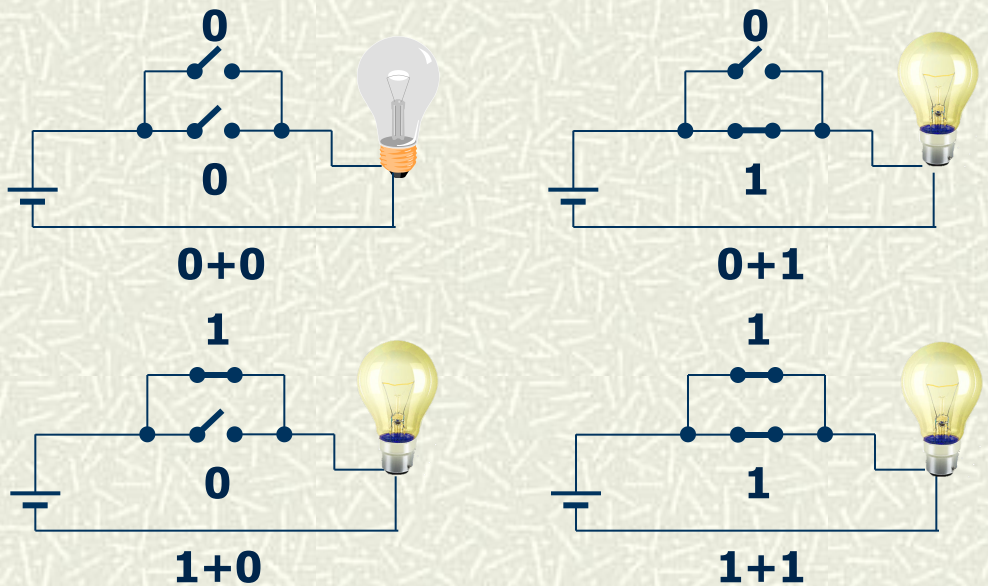


- Si definiscono delle operazioni fra i valori booleani: AND, OR, NOT sono gli operatori fondamentali

L’operazione di OR

- Si definisce l’operazione di somma logica (OR):
  il valore della somma logica è il simbolo 1 se il valore di almeno uno degli operandi è il simbolo 1

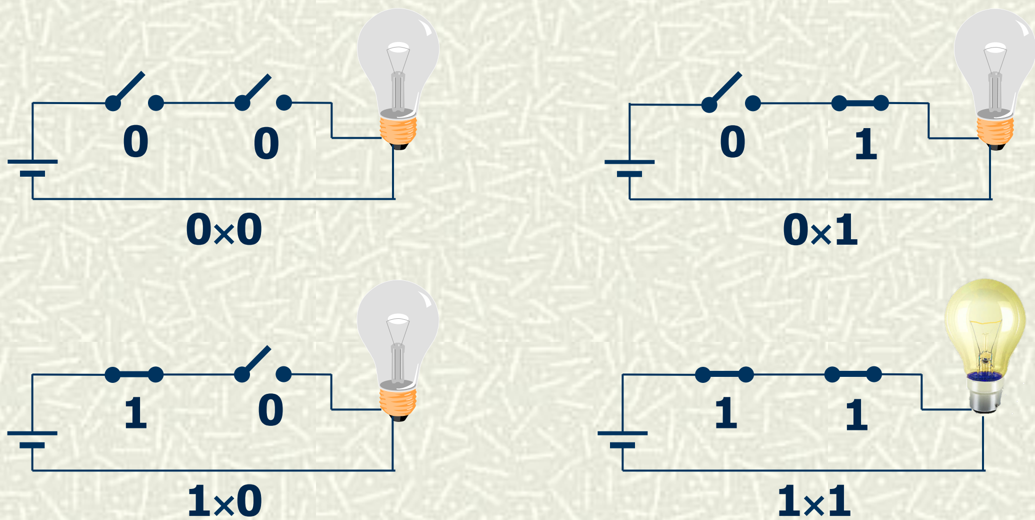
```
0+0 = 0
0+1 = 1
1+0 = 1
1+1 = 1
```



L'operazione di AND

- Si definisce l'operazione di prodotto logico (AND): il valore del prodotto logico è il simbolo 1 se il valore di tutti gli operandi è il simbolo 1

$$
\begin{aligned}
0\times0 &= 0\\
0\times1 &= 0\\
1\times0 &= 0\\
1\times1 &= 1
\end{aligned}
$$



La negazione NOT

- Si definisce l'operatore di negazione (NOT): l'operatore inverte il valore della costante su cui opera

$$\overline{0} = 1$$
$$\overline{1} = 0$$

- Dalla definizione…

$$\overline{\overline{0}} = 0$$
$$\overline{\overline{1}} = 1$$

Variabili binarie

- Una variabile binaria indipendente può assumere uno dei due valori **0** e **1**

$$\begin{tikzcd}[row sep=small] & 0 \\ x \arrow[ur] \arrow[dr] & \\ & 1 \end{tikzcd}$$

- Date n variabili binarie indipendenti, la loro somma logica (OR) è

$$x_1 + x_2 + \ldots + x_n = \begin{cases} \mathbf{1} & \text{se almeno una } x_i \text{ vale } \mathbf{1} \\ \mathbf{0} & \text{se } x_1 = x_2 = \ldots = x_n = \mathbf{0} \end{cases}$$

L'elemento x' = NOT(x) viene detto complemento di x. Il complemento è unico.

AND e NOT con variabili binarie

- Date n variabili binarie indipendenti, il loro prodotto logico (AND) è

$$x_1 \times x_2 \times \ldots \times x_n = \begin{cases} 0 & \text{se almeno una } x_i \text{ vale } 0 \\ 1 & \text{se } x_1 = x_2 = \ldots = x_n = 1 \end{cases}$$

- La negazione di una variabile x è

$$\bar{x} = 0 \quad \text{se} \quad x = 1$$
$$\bar{x} = 1 \quad \text{se} \quad x = 0$$

0 è l'elemento neutro per l'operazione di OR; 1 è l'elemento neutro per l'AND. Gli elementi neutri sono unici.
La legge $x + x = x \cdot x = x$ è detta legge dell'idempotenza.

Alcune identità

- Si verificano

$$x + 1 = 1 \qquad x \times 1 = x$$
$$x + 0 = x \qquad x \times 0 = 0$$
$$x + x = x \qquad x \times x = x$$

- Ad esempio…

$$x \times 1 = x$$

$x = 0$: $0 \times 1 = 0$

$x = 1$: $1 \times 1 = 1$

OK!

Altre proprietà

- Per gli operatori AND e OR valgono le seguenti proprietà:

commutativa $x_1+x_2 = x_2+x_1$ $x_1 \times x_2 = x_2 \times x_1$

associativa $x_1+x_2+x_3 = x_1+(x_2+x_3)$ $x_1 \times x_2 \times x_3 = x_1 \times (x_2 \times x_3)$

distributiva del prodotto rispetto alla somma $x_1 \times x_2 + x_1 \times x_3 = x_1 \times (x_2+x_3)$

- Per l'operatore NOT si provano le seguenti identità:

$$x + \overline{x} = 1$$

$$x \times \overline{x} = 0$$

$$\overline{\overline{x}} = x$$

Configurazioni delle variabili

- Date n variabili binarie indipendenti x_1, x_2,…, x_n, queste possono assumere 2^n configurazioni distinte

Ad esempio per n=3 si hanno 8 configurazioni

$x_1x_2x_3 \Rightarrow$ 000 001 010 011
100 101 110 111

- Una configurazione specifica è individuata univocamente da un AND (a valore 1) di tutte le variabili, dove quelle corrispondenti ai valori 0 compaiono negate

010 $\longleftrightarrow$ $\overline{x}_1x_2\overline{x}_3$

Funzioni logiche

- Una variabile y è una funzione delle n variabili indipendenti $x_1, x_2, \ldots, x_n$, se esiste un criterio che fa corrispondere in modo univoco ad ognuna delle 2^n configurazioni delle x_i un valore di y

$$y = F(x_1, x_2, \ldots, x_n)$$

- Una rappresentazione esplicita di una funzione è la **tabella di verità**, in cui si elencano tutte le possibili combinazioni di $x_1, x_2, \ldots, x_n$, con associato il valore di y

$$y = x_1 + x_2$$

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

Una tabella di verità

- Date tre variabili booleane (**A,B,C**), si scriva la funzione **F** che vale 1 quando solo due di esse hanno valore 1

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Si può scrivere la funzione come somma logica delle configurazioni corrispondenti agli 1

$$F(A,B,C) = \bar{A}BC + A\bar{B}C + AB\bar{C}$$

Forma canonica: somma di prodotti (OR di AND)
- tutte le funzioni logiche si possono scrivere in questa forma

ESEMPI

1) La legge dell'assorbimento $x_1 + x_1 \cdot x_2 = x_1$

2) Le leggi di De Morgan

$(x_1 + x_2)' = x_1' \cdot x_2'$ $\quad$ $(x_1 \cdot x_2)' = x_1' + x_2'$ $\quad$ (' un modo alternativo per indicare la negazione).

Dalle leggi di De Morgan si evince che la scelta delle funzioni OR, AND e NOT, come funzioni primitive, è ridondante. L'operazione logica AND può essere espressa in funzione delle operazioni OR e NOT; in modo analogo, l'operazione OR può essere espressa tramite AND e NOT.

3) Le relazioni stabilite sono generalmente applicate nelle trasformazioni di funzioni booleane in altre equivalenti, ma di più facile realizzazione circuitale.

variabili

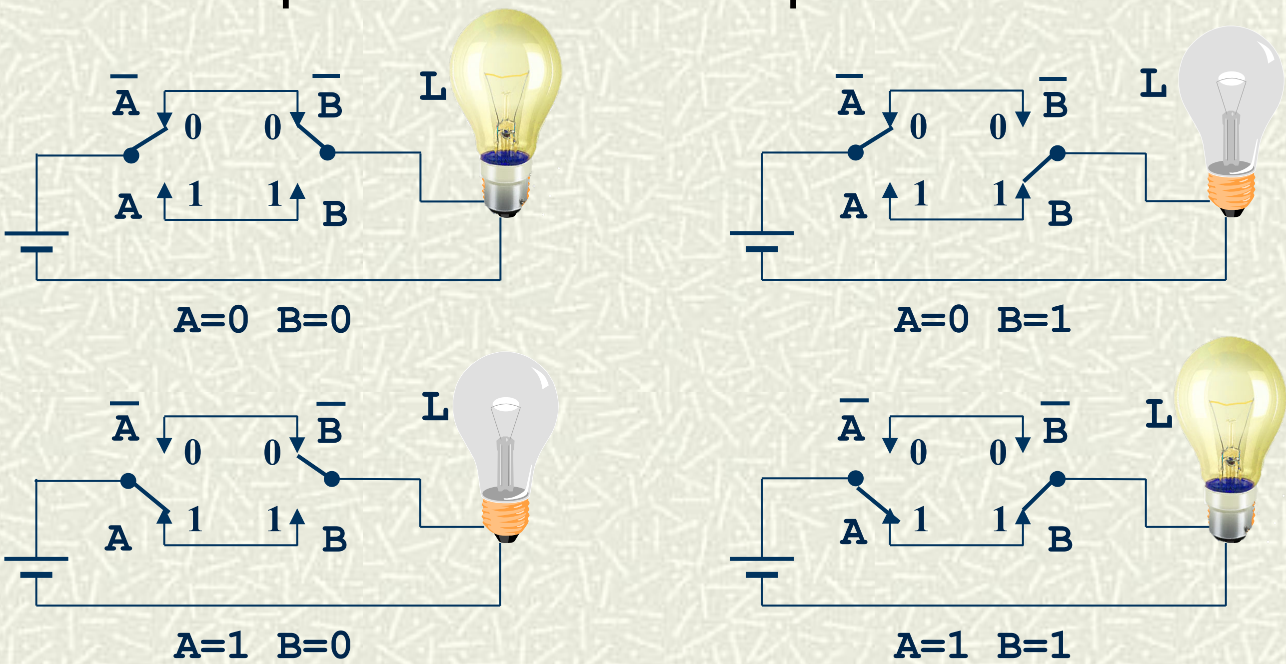
x_1	x_2	XOR
0	0	0
0	1	1
1	0	1
1	1	0

- L'espressione come somma di prodotti è quindi...

$$XOR = \overline{x}_1 x_2 + x_1 \overline{x}_2$$

Un circuito con due interruttori

- I due interruttori corrispondono a due variabili (**A,B**) a valori booleani - le variabili assumono i due valori 0 e 1 che corrispondono alle due posizioni dell'interruttore



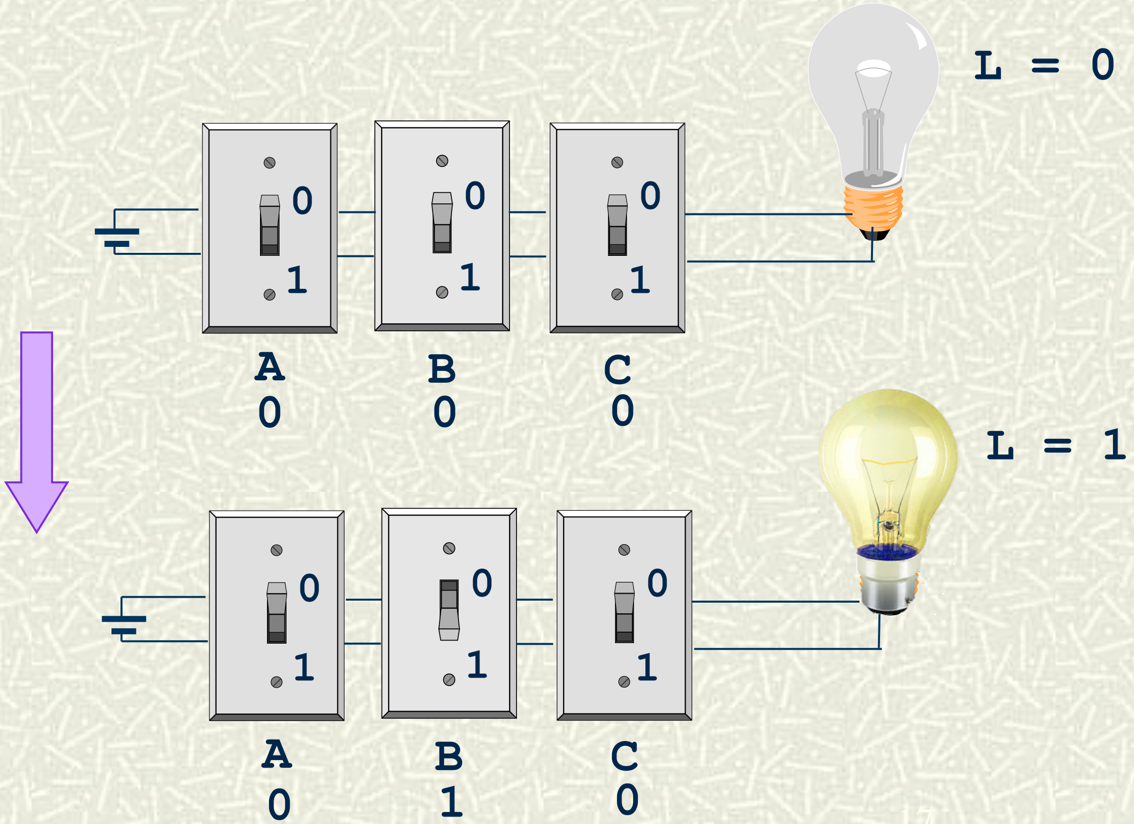
A	B	L
0	0	1
0	1	0
1	0	0
1	1	1

$$L = \overline{A}\times\overline{B}+A\times B$$

Un esercizio

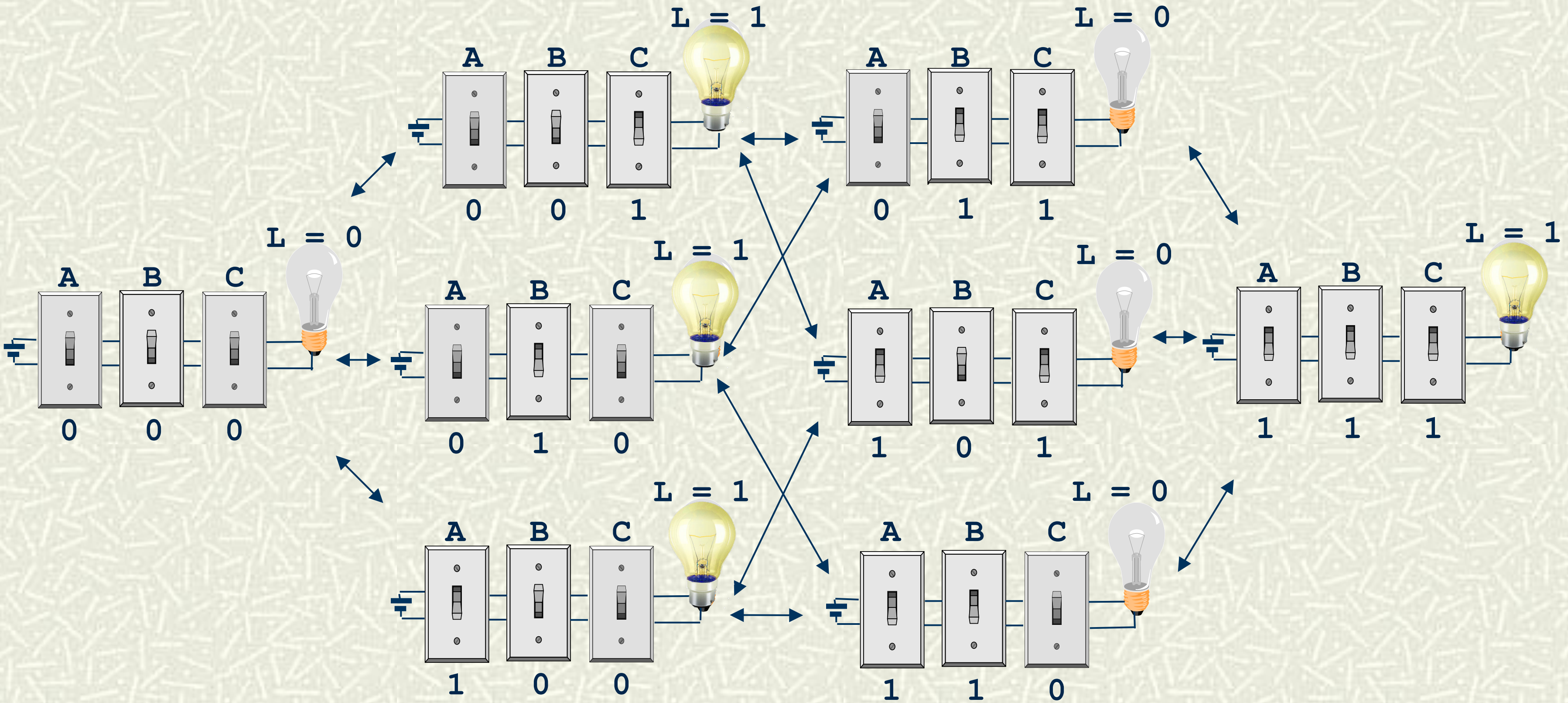
- *Progettare un circuito per accendere e spegnere una lampada da uno qualsiasi di tre interruttori indipendenti*

Cambia lo stato di un interruttore qualsiasi



Analisi delle combinazioni

- Si considera cosa accade a partire dalla configurazione di partenza, cambiando lo stato di un interruttore per volta



Scrittura della funzione logica

- Dalle otto combinazioni si ottiene la **tabella di verità** della funzione logica

A	B	C	L
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

- Si può scrivere la funzione **L** come **somma logica di prodotti logici**

$$L = \overline{A}\times\overline{B}\times C + \overline{A}\times B\times\overline{C} + A\times\overline{B}\times\overline{C} + A\times B\times C$$

Esercizi

- **Esercizio 1**

  Siano a_2, a_1, b_2, b_1 i bit che rappresentano due numeri interi positivi ($A=a_2a_1$ e $B=b_2b_1$). Sia r una variabile booleana che vale 1 se e solo A è maggiore di B ($A>B$). Ad esempio, quando $A=11$ e $B=10$, allora $a_2=1$, $a_1=1$, $b_2=1$, $b_1=0$ e $r=1$. Si scriva la forma canonica che definisce r come funzione di a_2, a_1, b_2, b_1.

- **Esercizio 2**

  I signori A, B, C e D fanno parte di un consiglio di amministrazione. Sapendo che hanno a disposizione le seguenti quote di possesso azionario A=40%, B=25%, C=20%, D=15%, formalizzare tramite tabella di verità (con successiva estrazione della forma canonica) la funzione booleana che decide quando il consiglio di amministrazione è in grado di approvare una mozione.

Esercizi (continua)

- ## Esercizio 3

  Il direttore di una squadra di calcio vuol comprare 4 giocatori dal costo di 2, 5, 6 e 8 milioni di euro, ma ha a disposizione solo 8 milioni. Siano a_2, a_5, a_6, a_8 variabili booleane che valgono 1 se e solo se si acquistano, rispettivamente, i giocatori da 2, 5, 6, 8 milioni. Sia r una variabile che è vera se e solo se l'insieme di giocatori che si decide di comprare costa meno della cifra disponibile. Ad esempio, se $a_2=1$, $a_5=1$, $a_6=0$, $a_8=0$, allora $r=1$. Si scriva la forma canonica che definisce r come funzione delle variabili a_2, a_5, a_6, a_8.

Sistemi di numerazione

Sistemi di numerazione posizionali

- Sistemi di numerazione **posizionali**:

  La **base** del sistema di numerazione
  Le **cifre** del sistema di numerazione

  Il numero è scritto specificando le cifre in ordine ed il suo valore dipende dalla **posizione relativa** delle cifre

  Esempio: Il sistema decimale (Base 10)

  Cifre : 0 1 2 3 4 5 6 7 8 9

  $$5641 = 5 \cdot \boxed{10^3} + 6 \cdot \boxed{10^2} + 4 \cdot \boxed{10^1} + 1 \cdot \boxed{10^0}$$

  Posizione: 3 2 1 0

Sistemi in base B

- La base definisce il numero di cifre diverse nel sistema di numerazione
- La cifra di minor valore è sempre lo 0; le altre sono, nell'ordine, 1,2,…,B-1; se B>10 occorre introdurre B-10 simboli in aggiunta alle cifre decimali

Un numero intero N si rappresenta con la scrittura $(c_n c_{n-1} \ldots c_2 c_1 c_0)_B$

$$N = c_n B^n + c_{n-1} B^{n-1} + \ldots + c_2 B^2 + c_1 B^1 + c_0 B^0$$

c_n è la cifra più significativa, c_0 la meno significativa

Un numero frazionario N' si rappresenta come $(0,c_1 c_2 \ldots c_n)_B$

$$N' = c_1 B^{-1} + c_2 B^{-2} + \ldots + c_n B^{-n}$$

Numeri interi senza segno

- Con n cifre in base B si rappresentano tutti i numeri interi positivi da 0 a B^n-1 (B^n numeri distinti)

Esempio: base 10

2 cifre: da 0 a 10^2-1 = 99

00
01
02
….
98
99

10^2 = 100 valori

Esempio: base 2

2 cifre: da 0 a 2^2-1 = 3

00
01
10
11

2^2 = 4 valori

Il sistema binario (B=2)

- La base 2 è la più piccola per un sistema di numerazione

Cifre: 0 1 - bit (binary digit)

Forma polinomia

Esempi:

$$(101101)_2 = \boxed{1\times2^5 + 0\times2^4 + 1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0} = 32 + 0 + 8 + 4 + 0 + 1 = (45)_{10}$$

$$(0{,}0101)_2 = 0\cdot2^{-1} + 1\cdot2^{-2} + 0\cdot2^{-3} + 1\cdot2^{-4} = 0 + 0{,}25 + 0 + 0{,}0625 = (0{,}3125)_{10}$$

$$(11{,}101)_2 = 1\cdot2^1 + 1\cdot2^0 + 1\cdot2^{-1} + 0\cdot2^{-2} + 1\cdot2^{-3} = 2 + 1 + 0{,}5 + 0 + 0{,}125 = (3{,}625)_{10}$$

Dal bit al byte

- Un **byte** è un insieme di 8 bit (un numero binario a 8 cifre)

$$b_7b_6b_5b_4b_3b_2b_1b_0$$

- Con un byte si rappresentano i numeri interi fra 0 e 2^8-1 = 255

00000000
00000001
00000010
00000011
……………. $\quad 2^8 = 256$ valori distinti
11111110
11111111

- È l'elemento base con cui si rappresentano i dati nei calcolatori
- Si utilizzano sempre dimensioni multiple (di potenze del 2) del byte: 2 byte (16 bit), 4 byte (32 bit), 8 byte (64 bit)…

Dal byte al kilobyte

- Potenze del 2

$2^4 = 16$
$2^8 = 256$
$2^{16} = 65536$

$2^{10} = 1024$ (K=Kilo)
$2^{20} = 1048576$ (M=Mega)
$2^{30} = 1073741824$ (G=Giga)

- Cosa sono KB (Kilobyte), MB (Megabyte), GB (Gigabyte)?

1 KB = 2^{10} byte = 1024 byte
1 MB = 2^{20} byte = 1048576 byte
1 GB = 2^{30} byte = 1073741824 byte
1 TB = 2^{40} byte = 1099511627776 byte (Terabyte)

Da decimale a binario
Numeri interi

- Si divide ripetutamente il numero **intero** decimale per 2 fino ad ottenere un quoziente nullo; le cifre del numero binario sono i resti delle divisioni; la cifra più significativa è l'ultimo resto

Esempio: convertire in binario $(43)_{10}$

resti

$$43 : 2 = 21 + 1$$
$$21 : 2 = 10 + 1$$
$$10 : 2 = 5 + 0$$
$$5 : 2 = 2 + 1$$
$$2 : 2 = 1 + 0$$
$$1 : 2 = 0 + 1$$

bit più significativo

$$(43)_{10} = (101011)_2$$

Da decimale a binario
Numeri razionali

- Si moltiplica ripetutamente il numero **frazionario** decimale per 2, fino ad ottenere una parte decimale nulla o, dato che la condizione potrebbe non verificarsi mai, per un numero prefissato di volte; le cifre del numero binario sono le parti intere dei prodotti successivi; la cifra più significativa è il risultato della prima moltiplicazione

Esempio: convertire in binario $(0,21875)_{10}$ e $(0,45)_{10}$

$$\begin{aligned} 0,21875 \times 2 &= \boxed{0},4375 \\ 0,4375 \times 2 &= \boxed{0},875 \\ 0,875 \times 2 &= \boxed{1},75 \\ 0,75 \times 2 &= \boxed{1},5 \\ 0,5 \times 2 &= \boxed{1},0 \end{aligned}$$

$$\Downarrow$$

$$(0,21875)_{10} = (0,00111)_2$$

$$\begin{aligned} 0,45 \times 2 &= \boxed{0},9 \\ 0,90 \times 2 &= \boxed{1},8 \\ 0,80 \times 2 &= \boxed{1},6 \\ 0,60 \times 2 &= \boxed{1},2 \\ 0,20 \times 2 &= \boxed{0},4 \quad \text{etc.} \end{aligned}$$

$$\Downarrow$$

$$(0,45)_{10} \approx (0,01110)_2$$

Esercizio

Si verifichino le seguenti corrispondenze:

a) $(110010)_2=(50)_{10}$
b) $(1110101)_2=(102)_{10}$
c) $(1111)_2=(17)_{10}$
d) $(11011)_2=(27)_{10}$
e) $(100001)_2=(39)_{10}$
f) $(1110001110)_2=(237)_{10}$

$$(101011)_2 = 1\times2^5 + 0\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0 = (43)_{10}$$

- …si può operare nel modo seguente: si raddoppia il bit più significativo e si aggiunge al secondo bit; si raddoppia la somma e si aggiunge al terzo bit… si continua fino al bit meno significativo

Esempio: convertire in decimale $(101011)_2$

bit più significativo

$$\begin{array}{rcrcl}
\boxed{1} \times 2 & = & 2 & + & 0 \\
2 \times 2 & = & 4 & + & 1 \\
5 \times 2 & = & 10 & + & 0 \\
10 \times 2 & = & 20 & + & 1 \\
21 \times 2 & = & 42 & + & 1 = 43
\end{array}$$

$$(101011)_2 = (43)_{10}$$

Sistema esadecimale

- La base 16 è molto usata in campo informatico

Cifre: 0 1 2 3 4 5 6 7 8 9 A B C D E F

La corrispondenza in decimale delle cifre oltre il 9 è

$A = (10)_{10}$ $D = (13)_{10}$
$B = (11)_{10}$ $E = (14)_{10}$
$C = (12)_{10}$ $F = (15)_{10}$

Esempio:

$$(3A2F)_{16} = 3\times16^3 + 10\times16^2 + 2\times16^1 + 15\times16^0 = 3\times4096 + 10\times256 + 2\times16 + 15 = (14895)_{10}$$

Da binario a esadecimale

- Una cifra esadecimale corrisponde a 4 bit

0 corrisponde a 4 bit a 0

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

F corrisponde a 4 bit a 1

- Si possono rappresentare numeri binari lunghi con poche cifre (1/4)
- La conversione da binario ad esadecimale è immediata, raggruppando le cifre binarie in gruppi di 4 (da destra) e sostituendole con le cifre esadecimali secondo la tabella precedente

Dai bit all'hex

- Un numero binario di 4n bit corrisponde a un numero esadecimale di n cifre

  Esempio: 32 bit corrispondono a 8 cifre esadecimali

1101	1001	0001	1011	0100	0011	0111	1111
D	9	1	B	4	3	7	F

  $(D91B437F)_{16}$

  Esempio: 16 bit corrispondono a 4 cifre esadecimali

0000	0000	1111	1111
0	0	F	F

  $(00FF)_{16}$

Da esadecimale a binario

- La conversione da esadecimale a binario si ottiene espandendo ciascuna cifra con i 4 bit corrispondenti

Esempio: convertire in binario il numero esadecimale 0x0c8f

Notazione usata in molti linguaggi di programmazione (es. C e Java) per rappresentare numeri esadecimali

0	c	8	f
0000	1100	1000	1111

Il numero binario ha 4x4 =16 bit

Esempi - 1

- In qualsiasi base, l'essere il sistema di numerazione ***posizionale*** impone che combinazioni diverse di cifre uguali rappresentino numeri diversi; ad esempio:
  - $(319)_{10} \neq (193)_{10}$
  - $(152)_6 \neq (512)_6$, infatti...

$$(152)_6=1\times6^2+5\times6^1+2\times6^0=36+30+2=(68)_{10}$$
$$(512)_6=5\times6^2+1\times6^1+2\times6^0=180+6+2=(188)_{10}$$

- Numeri che hanno identica rappresentazione, in basi diverse, hanno valori diversi:
  - $(234)_{10} \neq (234)_8$, infatti...

$$(234)_8 = 2\times8^2 + 3\times8^1 + 4\times8^0 = 2\times64 + 3\times8 + 4 = 128+24+4 = (156)_{10}$$

Osservazione: più piccola è la base, minore è il valore del numero rappresentato dalla stessa sequenza di cifre

Esempi - 2

- La notazione posizionale si applica anche per il calcolo del valore dei numeri frazionari, infatti...
  - $(0{,}872)_{10} = 8\times10^{-1} + 7\times10^{-2} + 2\times10^{-3}$
    $= 8/10 + 7/100 + 2/1000 = 0{,}8 + 0{,}07 + 0{,}002$
- Quante cifre occorreranno per rappresentare il numero decimale 36 in base 2? Sappiamo che con n cifre si rappresentano i numeri da 0 a 2^n-1, quindi...
  - per n=1 (con una cifra) si rappresentano $0...2^1-1 \Rightarrow 0,1$
  - per n=2: $0...2^2-1 \Rightarrow 0...3$
  - per n=3: $0...2^3-1 \Rightarrow 0...7$
  - per n=4: $0...2^4-1 \Rightarrow 0...15$
  - per n=5: $0...2^5-1 \Rightarrow 0...31$
  - per n=6: $0...2^6-1 \Rightarrow 0...63$

Effettivamente possiamo verificare che $(100100)_2=(36)_{10}$, infatti...
$100100=1\times2^5+0\times2^4+0\times2^3+1\times2^2+0\times2^1+0\times2^0$
$= 32 + 4 = 36$
con **6** cifre necessarie per la codifica del numero in base 2

Esempi - 3

- Quesito: Per quale base B risulterà vera l'uguaglianza

  $17 + 41 + 22 = 102$?

  Se i numeri sono rappresentati in base B, sappiamo che:

  $(17)_B = 1\times B^1+7\times B^0 = B+7$

  $(41)_B = 4\times B^1+1\times B^0 = 4B+1$

  $(22)_B = 2\times B^1+2\times B^0 = 2B+2$

  $(102)_B = 1\times B^2+0\times B^1+2\times B^0 = B^2+2$

  da cui... $B+7+4B+1+2B+2 = 7B+10 = B^2+2$

  $\Rightarrow$ Si ottiene un'equazione di 2° grado: $\boxed{B^2 - 7B - 8 = 0}$

  Risolvendo: $\Delta = 49+32 = 81$

  $B = (7 \pm \sqrt{\Delta})/2 =$

  $(7+9)/2 = 8$ — **È la soluzione!**

  $(7-9)/2 = -1$ — Non può essere una base

La rappresentazione dei dati e l'aritmetica degli elaboratori

Numeri interi positivi

- I numeri interi positivi sono rappresentati all'interno dell'elaboratore utilizzando un multiplo del byte (generalmente 4 byte)
- Se l'intero si rappresenta con un numero di cifre minore, vengono aggiunti zeri nelle cifre più significative

Esempio: 12 viene rappresentato in un byte come…

00001100

Numeri con segno

- Per rappresentare numeri con segno, occorre utilizzare un bit per definire il segno del numero
- Si possono usare tre tecniche di codifica

  - Modulo e segno
  - Complemento a 2
  - Complemento a 1

Modulo e segno

- Il bit più significativo rappresenta il segno: 0 per i numeri positivi, 1 per quelli negativi
- Esiste uno zero positivo (00…0) e uno zero negativo (10…0)
- Se si utilizzano n bit si rappresentano tutti i numeri compresi fra $-(2^{n-1}-1)$ e $+2^{n-1}-1$

  Esempio: con 4 bit si rappresentano i numeri fra -7 ($-(2^3-1)$) e +7 (2^3-1)

positivi		negativi	
0000	+0	1000	-0
0001	+1	1001	-1
0010	+2	1010	-2
0011	+3	1011	-3
0100	+4	1100	-4
0101	+5	1101	-5
0110	+6	1110	-6
0111	+7	1111	-7

Complemento a 2

- Il complemento a 2 di un numero binario $(N)_2$ a n cifre è il numero

$$2^n-(N)_2 = 1\underbrace{0......0}_{n}-(N)_2$$

- Il complemento a 2 si calcola…
  - Effettuando il complemento a 1 del numero di partenza (negazione di ogni cifra): si trasforma ogni 0 in 1 e ogni 1 in 0
  - Aggiungendo 1 al numero ottenuto
- Oppure: a partire da destra, lasciando invariate tutte le cifre fino al primo 1 compreso, quindi invertendo il valore delle rimanenti

```
01010111
↓↓↓↓↓↓↓↓
10101000  ←  complemento a 1
             + 1
--------
10101001
```

100000000	2^8
011111111-	2^8-1
01010111	N
10101000	2^8-1-N
10101001	$2^8-\cancel{1}-N+\cancel{1}$

Interi in complemento a 2

- I numeri positivi sono rappresentati (come) in modulo e segno
- I numeri negativi sono rappresentati in complemento a 2 ⇒ la cifra più significativa ha sempre valore 1
- Lo zero è rappresentato come numero positivo (con una sequenza di n zeri)
- Il campo dei numeri rappresentabili varia da -2^{n-1} a $+2^{n-1}-1$

Esempio: numeri a 4 cifre

0000	+0	1000	-8
0001	+1	1001	-7
0010	+2	1010	-6
0011	+3	1011	-5
0100	+4	1100	-4
0101	+5	1101	-3
0110	+6	1110	-2
0111	+7	1111	-1

Nota:	0111	+7
	1000	-8

Interi a 16 bit

- Numeri interi rappresentati su 16 bit in complemento a 2:

Il numero intero positivo più grande è $2^{15}-1=(32767)_{10}$

0111 1111 1111 1111
0x 7 F F F

Il numero intero negativo più piccolo è $-2^{15}=(-32768)_{10}$

1000 0000 0000 0000
0x 8 0 0 0

Il numero intero –1 è rappresentato come

1111 1111 1111 1111
0x F F F F

Addizione binaria

- Le regole per l'addizione di due bit sono

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

- L'ultima regola è… $(1)_2+(1)_2 = (10)_2$ … $(1+1=2)_{10}$!!

Esempio

```
 1  11  1      riporti
01011011+
01011010
--------
10110101
```

```
 91+
 90
----
181
```

Sottrazione binaria - 1

- Le regole per la sottrazione di due bit sono

  0 - 0 = 0
  1 - 0 = 1
  1 - 1 = 0
  10 - 1 = 1 con prestito di 1 dalla cifra precedente a sinistra

Esempio

```
  0 10
1 1 0 0 1 -
    1 0 1
----------
1 0 1 0 0
```

←

```
25 -
 5
----
20
```

- La sottrazione può divenire complicata: quando si ha una richiesta sulla cifra precedente a sinistra, che è uno 0, l'operazione si propaga a sinistra fino alla prima cifra ad 1 del sottraendo

Sottrazione binaria – 2

- Utilizzando la rappresentazione in complemento a 2, addizione e sottrazione sono trattate come un'unica operazione

$$N_1-N_2 = N_1+\underbrace{(2^n-N_2)}-2^n$$

si trascura il bit n +1

complemento a 2 di N_2: rappresentazione di $(-N_2)$

❶ Si calcola il complemento a 2 di N_2
❷ Si somma N_1 con il complemento a 2 di N_2
❸ Si trascura il bit più significativo del risultato

Esempio: $(010001)_2-(000101)_2 = (17)_{10}-(5)_{10}$

$$\begin{array}{r} 010001\ + \\ 111011 \\ \hline \cancel{\boxed{1}}001100 \end{array} \longrightarrow (12)_{10}$$

Rappresentazioni in complemento

- Sono utili perché l'operazione di somma algebrica può essere realizzata non curandosi del bit di segno
- In complemento a 1 (più semplice da calcolare)…
  - Zero ha due rappresentazioni: 00000000 e 11111111
  - La somma bit a bit funziona "quasi sempre"

00110 +	(6)		11001 +	(-6)
10101 =	(-10)		11010 =	(-5)
11011	(-4)		10011	(-12)

- In complemento a 2…
  - Zero ha una sola rappresentazione
  - La somma bit a bit funziona sempre

Overflow

- L'overflow si ha quando il risultato di un'operazione non è rappresentabile correttamente con n bit

Esempio: 5 bit → [-16,+15]

14 +	01110 +	-8 +	11000 +
10	01010	-10	10110
24	$\boxed{11000}$ → -8	-18	1$\boxed{01110}$ → +14

- Per evitare l'overflow occorre aumentare il numero di bit utilizzati per rappresentare gli operandi
- C'è overflow se c'è riporto al di fuori del bit di segno e non sul bit di segno, o se c'è riporto sul bit di segno, ma non al di fuori

Punteggio nei vecchi videogame… sorpresa per i campioni!

0111 1111 1111 1111 + 1 = 1000 0000 0000 0000

32767 + 1 = -32768

Moltiplicazione binaria

- Le regole per la moltiplicazione di due bit sono

$$0 \times 0 = 0$$
$$0 \times 1 = 0$$
$$1 \times 0 = 0$$
$$1 \times 1 = 1$$

Esempio

```
   1100111 x 101
   -------
   1100111
  0000000□
 1100111□□
----------
1000000011
```

- Moltiplicare per 2^n corrisponde ad aggiungere n zeri in coda al moltiplicando

110011 x 10000 = 110011**0000** ← 816

51 → 110011; 10000 ← $\times 16 = 2^4$

Divisione binaria

- La divisione binaria di A per B viene calcolata in modo analogo alla divisione decimale, così da ottenere un quoziente Q ed un resto R, tali che A = B×Q + R
- La divisione binaria si compone di una serie di sottrazioni

```
1 1 0 1 1 0 | 1 0 1
1 0 1       |--------
-----       | 1 0 1 0
    1 1 1   |
    1 0 1   |
    -----   |
      1 0 0 |
```

54 = 5×10 + 4

- Dividere per 2^n equivale a scorrere il numero a destra di n posizioni; le cifre scartate costituiscono il resto

110011 : 10000 = 11 con resto 11 ⟸ 51:16 = 3 con resto 3

Esempio

- Quesito: Data una base B, si consideri il numero $x=(C_5C_4C_3C_2C_1C_0)_B$, dove le singole cifre valgono:
  $C_5 = $ B-1, $C_4 = $ B-1, $C_3 = $ B-1, $C_2 = 0$, $C_1 = 0$, $C_0 = 0$
  Qual è la rappresentazione in base B del risultato dell'espressione $(x/B^3)+1$?
  - Dividere per B^3, che per qualsiasi base si rappresenta come 1000, equivale a "shiftare" il numero di tre cifre a sinistra
    - ⇨ Rimangono quindi le tre cifre più significative, tutte uguali a B-1
    - ⇨ Sommando 1, a causa dei riporti, si ottiene quindi come risultato dell'espressione $1000=B^3$, qualsiasi sia il valore della base B

Esercizi

- **Esercizio 1**

  Assumendo che un elaboratore rappresenti i numeri interi con segno su quattro bit in complemento a 2, si calcolino entrambi i membri della seguente identità:

  $$(A\text{-}C)+B = (A+B)\text{-}C,$$

  con **A=7**, **B=5**, **C=7**. Si ottiene lo stesso risultato dal calcolo dei due membri? Discutere le motivazioni della risposta.

- **Esercizio 2**

  a) Si determini, se esiste, la base b di un sistema di numerazione tale che

  $$(842)_b = (1202)_{10}$$

  b) Si determini, se esiste, la base $b \leq 16$ di un sistema di numerazione tale che

  $$(725)_b - (626)_b = (224)_{10}$$

Numeri in virgola mobile

- La rappresentazione dei numeri in virgola mobile è in relazione con la **notazione scientifica** (es. $1.2\times10^2=120$)
- La IEEE ha previsto uno standard per la rappresentazione in virgola mobile
  - **singola precisione** (32 bit = 4 byte)
  - **doppia precisione** (64 bit = 8 byte)
  - **quadrupla precisione** (128 bit = 16 byte)

Singola precisione

31	30 – 23	22 – 0
	8 bit	23 bit
Segno	Esponente (o Caratteristica)	Mantissa

Il valore è

$(-1)^S\ 1.M\times2^{E-127}$ se $E\neq0$

$(-1)^S\ 0.M\times2^{-126}$ se $E=0$

Eccesso: vale $2^{t-1}-1$, se t è il numero di cifre riservate alla caratteristica → rappresentazione "interna" dell'esponente sempre positiva

Singola precisione

- Il numero più grande rappresentabile è $2^{128}\times(2-2^{-23})\approx 2^{129}\approx 6.81\times 10^{38}$

  0 11111111 11111111111111111111111

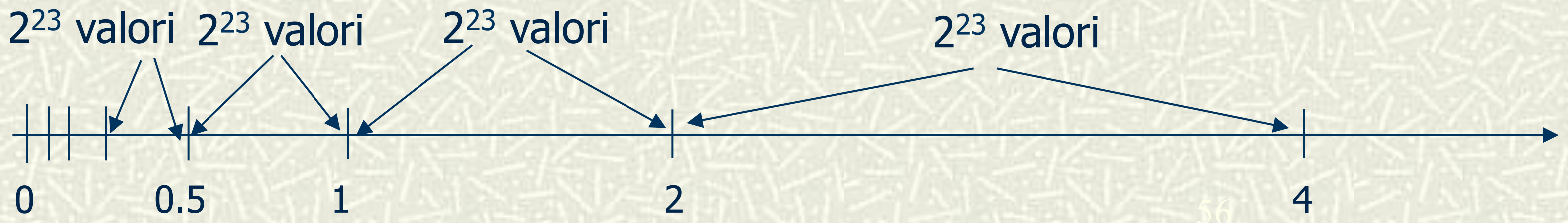
  $2^{255-127}$ 1.11111111111111111111111

- Il più piccolo numero positivo è $2^{-126}\times 2^{-23}=2^{-149}\approx 1.4\times 10^{-45}$

  0 00000000 00000000000000000000001

  2^{-126} 0.00000000000000000000001

- In totale si rappresentano 2^{32} numeri distinti, metà positivi, metà negativi
- Circa metà dei numeri sono compresi fra -1 e 1 (E-127<0)



L'aritmetica floating-point: addizione

- Metodo per il calcolo dell'addizione
  1. Se le caratteristiche dei numeri sono diverse, si considera il numero con caratteristica minore e…
     - 1.1 Si trasla la mantissa di un posto a destra
     - 1.2 Si incrementa la caratteristica di 1, fino a quando le due caratteristiche sono uguali, e corrispondono alla caratteristica del risultato
  2. La mantissa del risultato è ottenuta dalla somma delle due mantisse
  3. Se l'addizione comporta un riporto oltre la cifra più significativa, si trasla la mantissa del risultato a destra di un posto, il riporto nel bit più significativo, e si incrementa la caratteristica di 1

Un esempio di addizione

- Supponiamo che per la rappresentazione floating–point vengano utilizzati otto bit, di cui uno per il segno, tre per la caratteristica e quattro per la mantissa

$N = (-1)^s 0.M \times 2^{E-4}$

0	1	1	0	1	1	0	1	3.25
0	1	0	1	1	0	0	1	1.125

- La caratteristica del secondo operando è più piccola di una unità, quindi la mantissa deve scorrere di una posizione a destra

$$1001 \Rightarrow 0100$$

- La caratteristica del risultato è 110 e la mantissa è 1101+ 0100 = 10001; la caratteristica deve essere aumentata di 1 e portata a 111, e la mantissa del risultato traslata a destra di una posizione:

0	1	1	1	1	0	0	0

Codifica il numero 4 (dato che la caratteristica si rappresenta in eccesso a 4), ma il risultato corretto è 4.375: **errore di troncamento**

L'aritmetica floating-point: moltiplicazione

- Metodo per il calcolo della moltiplicazione
  1. Si moltiplicano le due mantisse
  2. Si addizionano le due caratteristiche
  3. Si trasla a sinistra il prodotto delle due mantisse fino ad ottenere un 1 come cifra più significativa; si diminuisce la caratteristica di 1 per ogni traslazione eseguita
  4. Si tronca la mantissa al numero di bit utilizzati nella rappresentazione; la mantissa del prodotto è il risultato del troncamento
  5. Si sottrae l'eccesso alla somma delle caratteristiche, ottenendo la caratteristica del prodotto

Un esempio di moltiplicazione

- Supponiamo che per la rappresentazione floating–point vengano utilizzati otto bit, di cui uno per il segno, tre per la caratteristica e quattro per la mantissa

N = $(-1)^s 0.M \times 2^{E-4}$

0	0	1	0	1	1	0	1	0.203125
0	1	0	1	1	0	0	1	1.125

- Moltiplicando le mantisse e sommando le caratteristiche si ottiene:

M=01110101 E=111

- La mantissa del risultato deve essere traslata di un posto a sinistra, e la somma delle caratteristiche deve essere decrementata di 1; infine la mantissa deve essere troncata alle 4 cifre significative e l'eccesso (100) sottratto alla caratteristica:

0	0	1	0	1	1	1	0

Codifica il numero 0.21875, ma il risultato corretto è 0.228515625: **errore di troncamento**

L'aritmetica degli elaboratori - 1

- L'aritmetica "interna" degli elaboratori differisce notevolmente dall'aritmetica classica
- Sebbene le stesse operazioni possano essere realizzate secondo modalità diverse su elaboratori diversi, si riscontrano alcune caratteristiche comuni:
  - Rappresentazione binaria dei numeri
  - Rango finito dei numeri rappresentabili
  - Precisione finita dei numeri
  - Operazioni espresse in termini di operazioni più semplici

L’aritmetica degli elaboratori - 2

- Rango finito dei numeri rappresentabili
  - Qualunque sia la codifica utilizzata, esistono sempre il più grande ed il più piccolo numero rappresentabile
  - I limiti inferiore e superiore del rango di rappresentazione dipendono sia dal tipo di codifica, sia dal numero di bit utilizzati
  - Se il risultato di un’operazione non appartiene al rango dei numeri rappresentabili, si dice che si è verificato un overflow (un underflow, più precisamente, se il risultato è più piccolo del più piccolo numero rappresentabile)

L'aritmetica degli elaboratori - 3

- Precisione finita dei numeri
  - La precisione della rappresentazione di un numero frazionario è una misura di quanto essa corrisponda al numero che deve essere rappresentato
  - Negli elaboratori, i numeri frazionari sono rappresentati in virgola mobile (**floating–point**), utilizzando un numero finito di bit
  - È plausibile che un numero reale non ammetta una rappresentazione finita, quindi dovrà essere codificato in maniera approssimata
  - Negli elaboratori si rappresentano soltanto numeri razionali (fino ad una data precisione)

L’aritmetica degli elaboratori - 4

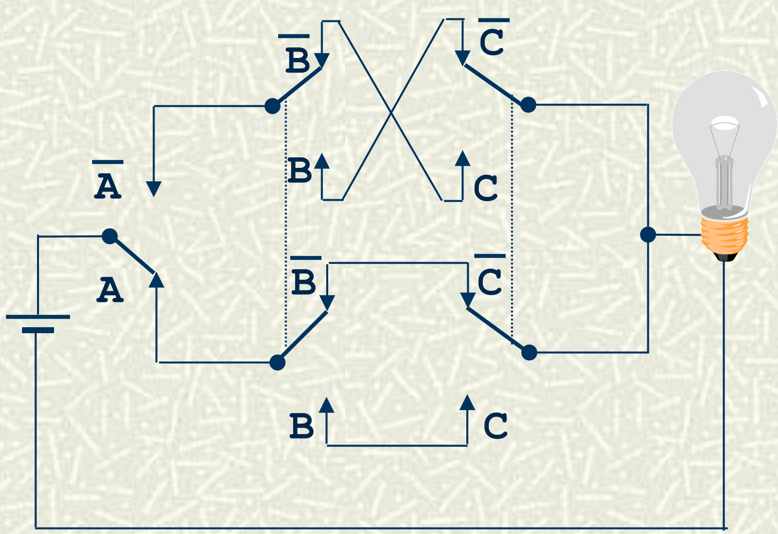
- **Operazioni espresse in termini di operazioni più semplici**
  - La maggior parte degli elaboratori non possiede circuiti in grado di eseguire direttamente tutte le operazioni:
    - La sottrazione si realizza per mezzo di una complementazione e di un’addizione
    - La moltiplicazione si realizza per mezzo di una successione di addizioni e di **shift** (traslazioni)
    - La divisione si realizza per mezzo di una successione di shift e sottrazioni
  - Le operazioni più semplici sono eseguite direttamente da appositi circuiti (in **hardware**); le operazioni più complesse sono realizzate mediante l’esecuzione di successioni di operazioni più semplici, sotto il controllo di programmi appositamente realizzati, e generalmente memorizzati permanentemente (in **firmware**)

Come collegare gli interruttori

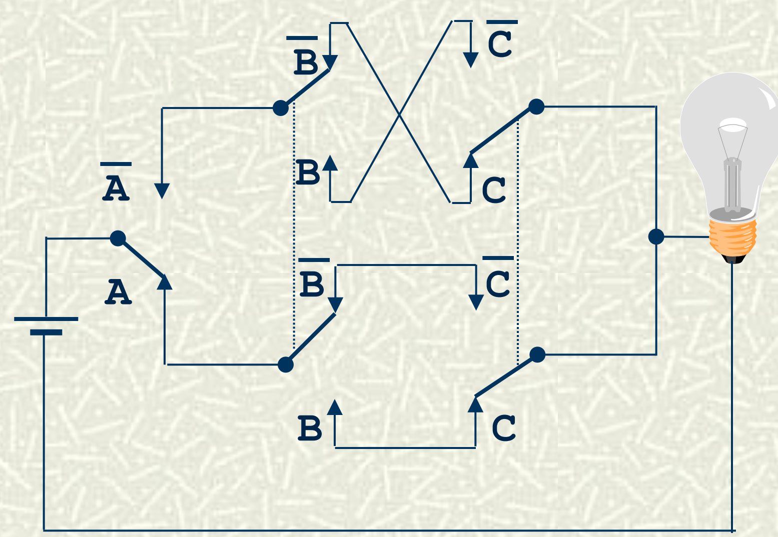
- Si può manipolare l'espressione di **L** usando la proprietà distributiva dell'AND rispetto all'OR

$$L = \boxed{\overline{A}}\times\overline{B}\times C + \boxed{\overline{A}}\times B\times\overline{C} + \boxed{A}\times\overline{B}\times\overline{C} + \boxed{A}\times B\times C$$

$$L = \overline{A}\times(\overline{B}\times C + B\times\overline{C}) + A\times(\overline{B}\times\overline{C} + B\times C)$$



1 0 0



1 0 1