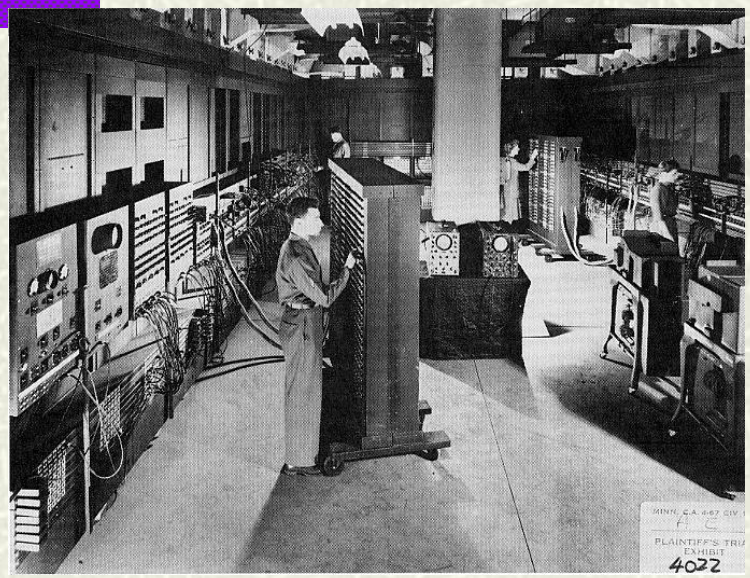
Fondamenti di Informatica I



ENIAC (1946 ca.)

Monica Bianchini
Dipartimento di Ingegneria dell'Informazione

E-mail: monica@dii.unisi.it

Diapositiva 1

m1 monica; 22/01/2006

Sommario

- **Introduzione**

  il calcolo automatico dalla preistoria ai giorni nostri

- **L'algebra di Boole**

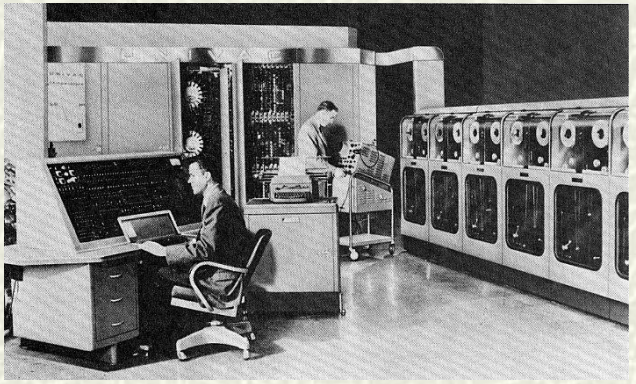
  da Analisi Matematica della Logica (1847) al progetto degli elaboratori digitali

- **Sistemi di numerazione**

  da additivo a posizionale, da decimale a binario, a esadecimale: l'alfabeto dell'elaboratore

- **La rappresentazione dei dati e l'aritmetica degli elaboratori**

  dai bit ai numeri, ai testi, alle immagini, alla musica, ai video in digitale



UNIVAC (1951)

Introduzione

Cenni storici – 1

- Anche se la presenza "invasiva" dell'informatica nella vita di tutti i giorni è un fenomeno relativamente recente, non recente è la necessità di avere a disposizione strumenti e metodi per contare rapidamente, elaborare dati, "calcolare"
  - Le prime testimonianze di strumenti per contare risalgono a 30.000 anni fa
  - I primi esempi di algoritmi — procedure di calcolo "automatico" — sono stati ritrovati in Mesopotamia su tavolette babilonesi risalenti al 1800–1600 a.C.
- Il sogno di costruire macchine capaci di effettuare calcoli automatici affonda le radici nel pensiero filosofico del '600:
  - Pascal e Leibniz non solo affrontarono il problema, già studiato da Cartesio, di automatizzare il ragionamento logico–matematico, ma si cimentarono nella realizzazione di semplici macchine per calcolare (capaci di effettuare somme e sottrazioni)

Cenni storici – 2

- La macchina alle differenze, concepita da Babbage nel 1833, rappresenta il primo esempio di macchina programmabile di utilità generale
  - La prima programmatrice nella storia dell'informatica è Ada Augusta Byron, contessa di Lovelace
- In seguito, lo stesso Babbage progetta la macchina analitica (mai realizzata, troppo complessa e critica la sua costruzione per le tecnologie meccaniche dell'epoca)

Cenni storici – 3

- Fu Herman Hollerith, nel 1890, a sviluppare la macchina a schede perforate, per compiere le statistiche del censimento decennale degli Stati Uniti
  - I dati venivano immessi su schede di cartone opportunamente perforate, le stesse schede che sono state usate fino a due decenni or sono
  - Le schede venivano successivamente "contate" da una sorta di pantografo che permetteva diversi tipi di elaborazioni (totali, medie, statistiche, etc.)
  - Si impiegarono due anni e mezzo ad analizzare i dati (contro i sette anni del censimento del 1880), nonostante l'incremento di popolazione da 50 a 63 milioni

Cenni storici – 4

- Successivamente la macchina a schede perforate venne utilizzata con successo per i censimenti in Austria, Norvegia e Russia, tanto che Hollerith decise di fondare una società: la ***Computing Tabulating Recording Company*** che, nel 1923, divenne l'***International Business Machine***, o IBM
- Nel 1932, il tedesco Konrad Zuse realizza una macchina elettromeccanica in grado di eseguire calcoli con controllo programmato, ed introduce il sistema di numerazione binario (la cui algebra era stata definita da Leibniz e da Boole)

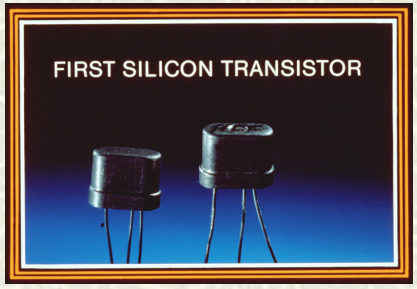
Cenni storici – 5

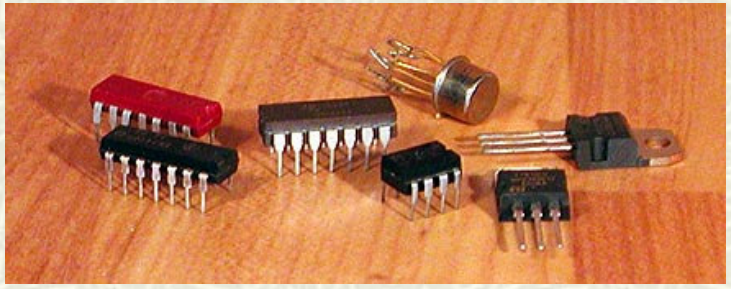
- Durante la seconda guerra mondiale, fioriscono i progetti di elaboratori da utilizzarsi per scopi bellici
  - Enigma, realizzata dai tedeschi per codificare le comunicazioni militari
  - Red Purple, di costruzione giapponese
  - Computer Colossus, costruito dagli inglesi per la decifrazione dei messaggi tedeschi, alla cui progettazione e realizzazione collaborò Alan Turing, permise la vittoria anglo–americana sull'Atlantico

La macchina Enigma

Cenni storici – 6

- Con l'invenzione del tubo a vuoto (1904), del transistor (1947) e, infine, dei circuiti integrati (1969), l'evoluzione dei computer divenne inarrestabile



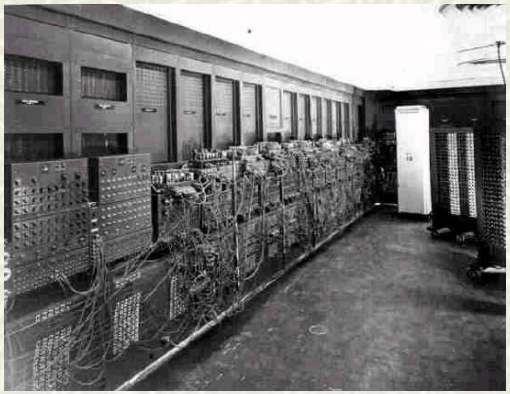


- Attualmente la potenza di calcolo degli elaboratori decuplica ogni 5–6 anni (…ma non può durare, almeno con le tecnologie in uso)

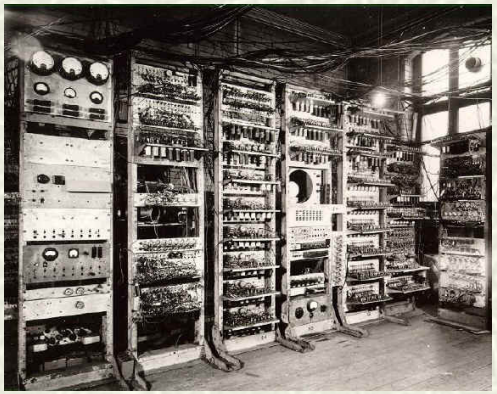
Cenni storici – 7

- La costruzione dei primi calcolatori risale all'inizio degli anni '40, grazie alla tecnologia elettronica; i primi esemplari venivano programmati mediante connessioni elettriche e commutatori (ENIAC, Mark I)
- Il nome di Von Neumann è legato invece ai primi calcolatori a programma memorizzato realizzati alla fine degli anni '40 (EDSAC, Whirlwind, IAS, UNIVAC)
  - Per la prima volta, vige il principio di *unitarietà di rappresentazione di dati e istruzioni*, che vengono codificati, all'interno dell'elaboratore, in maniera indistinguibile
- La diffusione dei calcolatori a livello mondiale è avvenuta nei decenni '60 e '70

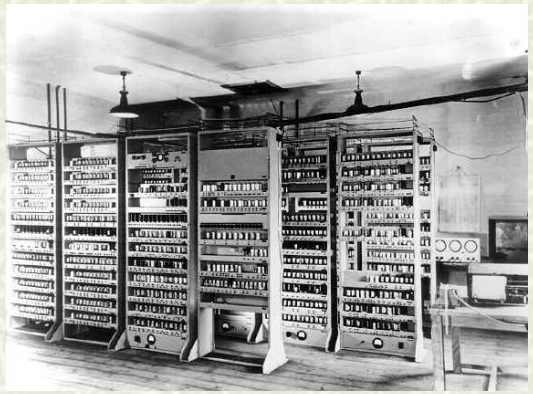
Cenni storici – 8



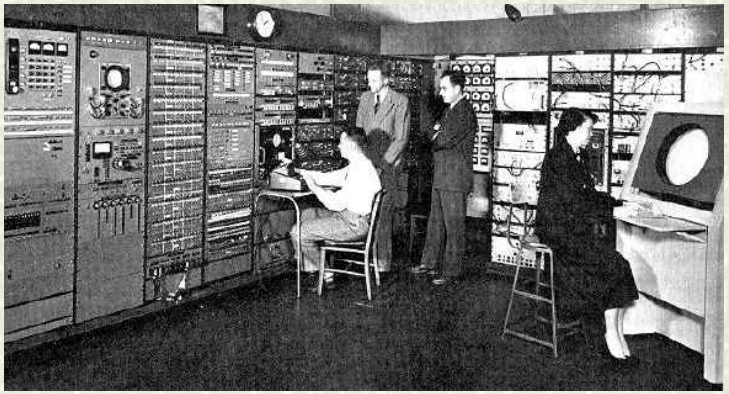
ENIAC (1946)



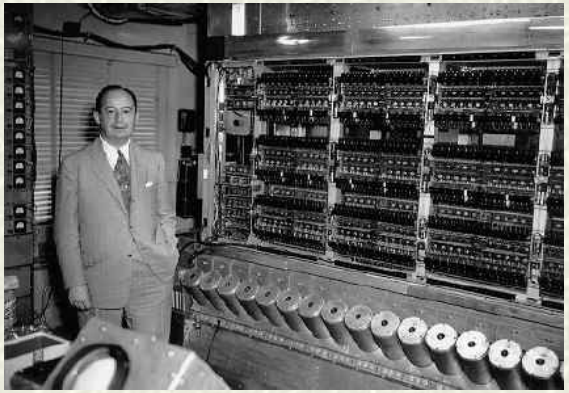
Mark I (1948)



EDSAC (1949)



Whirlwind (1949)



IAS (1952)

UNIVAC (1952)

Cenni storici – 9

- Tuttavia, l'esplosione dell'informatica come fenomeno di massa è datata 1981, anno in cui l'IBM introdusse un tipo particolare di elaboratore: il Personal Computer (PC)
- La particolarità dei PC consisteva nell'essere "assemblati" con componenti facilmente reperibili sul mercato (e quindi a basso costo)
  - Possibilità per qualsiasi casa produttrice di costruire "cloni"
- Attualmente i PC, o meglio il loro componente fondamentale — il microprocessore — è utilizzato in tutti i settori applicativi (non solo per elaborare dati):
  - Telefoni cellulari
  - Ricevitori satellitari digitali
  - Bancomat e carte di credito
  - Lavatrici e forni a microonde
  - ...

Cenni storici – 10

- L'esigenza di realizzare sistemi di elaborazione dotati di più processori operanti in parallelo è stata sentita fin dalla preistoria dell'informatica
  - In una relazione dello scienziato, generale e uomo politico italiano Menabrea, datata 1842, sulla macchina analitica di Babbage, si fa riferimento alla possibilità di usare più macchine dello stesso tipo in parallelo, per accelerare calcoli lunghi e ripetitivi
- Solo la riduzione dei costi dell'hardware ha consentito, verso la fine degli anni '60, l'effettiva costruzione dei primi supercalcolatori, come le macchine CDC6600 e Illiac e, successivamente, il Cray e le macchine vettoriali
- A partire dagli anni '90, gli ulteriori sviluppi della microelettronica hanno permesso la realizzazione di calcolatori a parallelismo massiccio e a "grana fine", caratterizzati dall'interconnessione di decine di migliaia di unità di elaborazione estremamente elementari: le reti neurali, capaci di "simulare" il comportamento del cervello umano, sulla base degli studi di McCulloch e Pitts (1943)

Cenni storici – 11

Illiac (1955)

CDC 6600 (1963)

Cray 1 (1976)

Cray X1 (2002)

PC IBM (1981)

Portatile e Palmare (oggi)

Frasi celebri ed altro…

- "Penso che ci sia mercato nel mondo per non più di cinque computer." (Thomas Watson, Presidente di IBM, 1943)
- "Ho girato avanti e indietro questa nazione (USA) e ho parlato con la gente. Vi assicuro che questa moda dell'elaborazione automatica non vedrà l'anno prossimo." (Editor di libri scientifici di Prentice Hall, 1947)
- "Nel futuro i computer verranno a pesare non più di una tonnellata e mezzo." (Popular Mechanichs, 1949)
- Nel 1976, il **New York Times** pubblicò un libro dal titolo ***La scienza nel ventesimo secolo,*** nel quale il calcolatore veniva menzionato una sola volta e indirettamente, in relazione al calcolo delle orbite dei pianeti
- "Non c'è ragione perché qualcuno possa volere un computer a casa sua." (Ken Olson, fondatore di Digital, 1977)

Che cos'è l'informatica – 1

- Informatica — fusione delle parole informazione e automatica — l'insieme delle discipline che studiano gli strumenti per l'elaborazione automatica dell'informazione e i metodi per un loro uso corretto ed efficace
- ***L'informatica è la scienza della rappresentazione e dell'elaborazione dell'informazione***
  - L'accento sull' "informazione" fornisce una spiegazione del perché l'informatica sia ormai parte integrante di molte attività umane: laddove deve essere gestita dell'informazione, l'informatica è un valido strumento di supporto
  - Il termine "scienza" sottolinea il fatto che, nell'informatica, l'elaborazione dell'informazione avviene in maniera sistematica e rigorosa, e pertanto può essere automatizzata

Che cos'è l'informatica – 2

- L'informatica non è, quindi, la scienza e la tecnologia dei calcolatori elettronici: il calcolatore è lo strumento che la rende "operativa"
- L'**elaboratore** (computer, calcolatore) è un'apparecchiatura **digitale**, **elettronica** ed **automatica** capace di effettuare trasformazioni sui dati:
  - **Digitale**: i dati sono rappresentati mediante un alfabeto finito, costituito da cifre (**digit**), che ne permette il trattamento mediante regole matematiche
  - **Elettronica**: realizzazione tramite tecnologie di tipo elettronico
  - **Automatica**: capacità di eseguire una successione di operazioni senza interventi esterni
- "La disumanità del computer sta nel fatto che, una volta programmato e messo in funzione, si comporta in maniera perfettamente onesta." (Isaac Asimov)

L'architettura di Von Neumann

- La capacità dell'elaboratore di eseguire successioni di operazioni in modo automatico è determinata dalla presenza di un dispositivo di **memoria**
  - Nella memoria sono registrati i **dati** e...
  - ...la descrizione delle operazioni da eseguire su di essi (nell'ordine secondo cui devono essere eseguite): il **programma**, la "ricetta" usata dall'elaboratore per svolgere il suo compito
- Il programma viene interpretato dall'**unità di controllo**

⇓

Modello di Von Neumann

La macchina universale

- **Programma**: sequenza di operazioni atte a predisporre l'elaboratore alla soluzione di una determinata classe di problemi
  - Il programma è la descrizione di un **algoritmo** in una forma comprensibile all'elaboratore
- **Algoritmo**: sequenza finita di istruzioni attraverso le quali un operatore umano è capace di risolvere ogni problema di una data classe; non è direttamente eseguibile dall'elaboratore
- L'elaboratore è una **macchina universale**: cambiando il programma residente in memoria, è in grado di risolvere problemi di natura diversa (una classe di problemi per ogni programma)

Ancora sull'informatica...

- *L'informatica è lo studio sistematico degli algoritmi che descrivono e trasformano l'informazione: la loro teoria, analisi, progetto, efficienza, realizzazione (ACM — Association for Computing Machinery)*

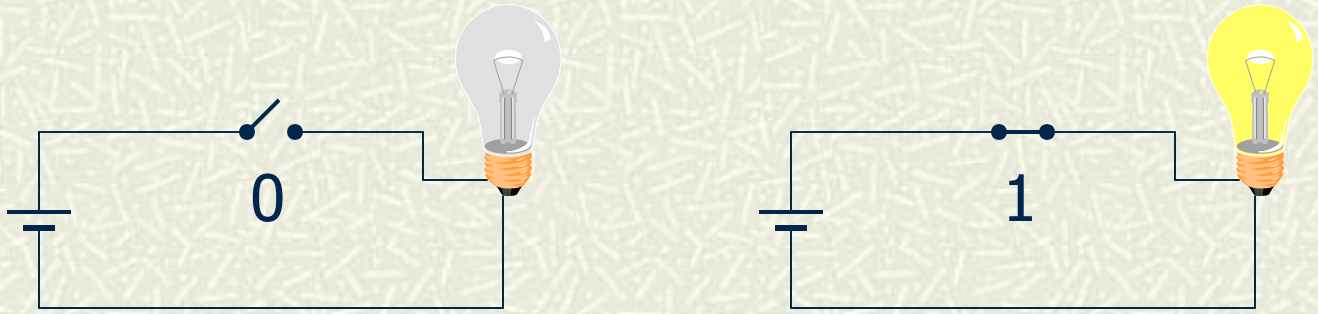
- Nota: È possibile svolgere un'attività concettualmente di tipo informatico senza l'ausilio del calcolatore, per esempio nel progettare ed applicare regole precise per svolgere operazioni aritmetiche con carta e penna; l'elaboratore, tuttavia, è uno strumento di calcolo potente, che permette la gestione di quantità di informazioni altrimenti intrattabili

L'algebra di Boole

Le operazioni AND e OR sono operazioni binarie, l'operazione NOT è unaria. Nella valutazione delle espressioni booleane esiste una relazione di precedenza fra gli operatori NOT, AND e OR, nell'ordine in cui sono stati elencati; le parentesi vengono utilizzate nel modo consueto.

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

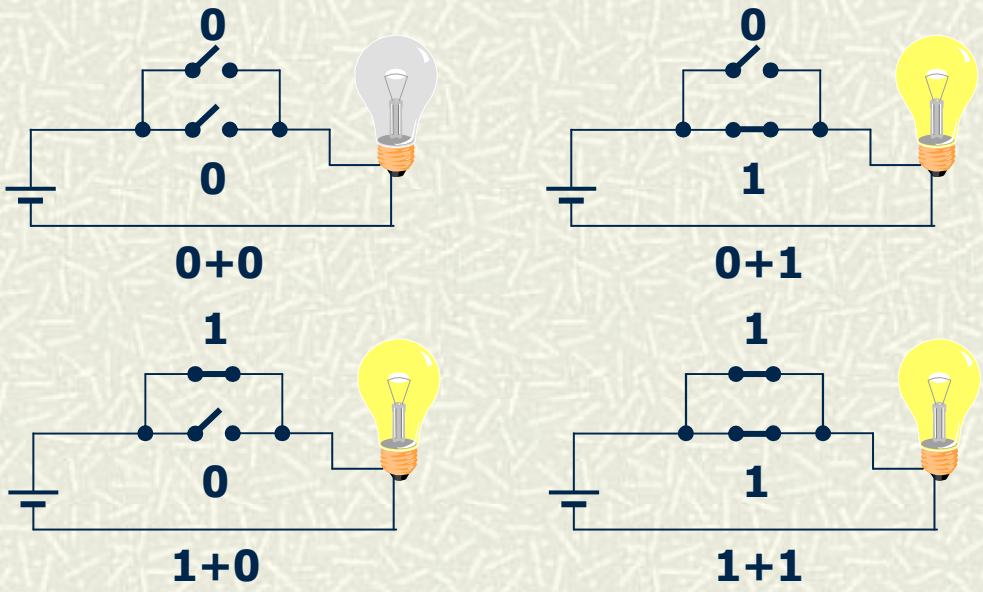


- Si definiscono delle operazioni fra i valori booleani: AND, OR, NOT sono gli operatori fondamentali

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

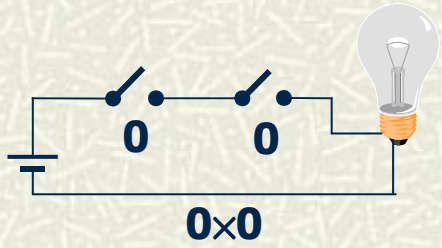
```
0+0 = 0
0+1 = 1
1+0 = 1
1+1 = 1
```

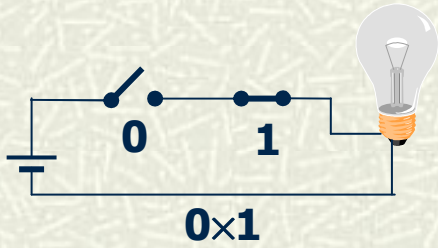


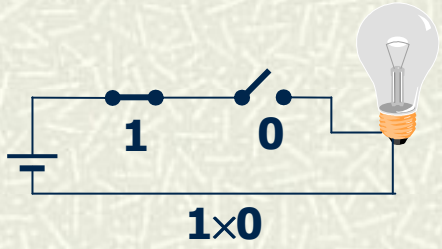
L'operazione di AND

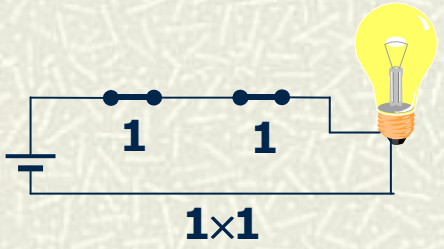
- Si definisce l'operazione di prodotto logico (AND): il valore del prodotto logico è il simbolo 1 se il valore di tutti gli operandi è il simbolo 1

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









La negazione NOT

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

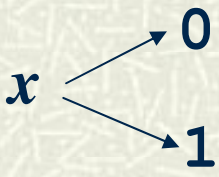
$$\overline{0} = 1$$
$$\overline{1} = 0$$

- 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**



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

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

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

$$\overline{x} = 0 \text{ se } x = 1$$
$$\overline{x} = 1 \text{ se } x = 0$$

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:

$$x + \bar{x} = 1$$
$$x \times \bar{x} = 0$$
$$\bar{\bar{x}} = x$$

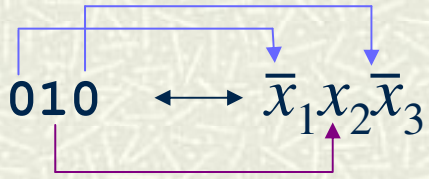
Configurazioni delle variabili

- 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

$x_1x_2x_3$ ⇒ `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



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

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

- 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

$(x_1 + x_2)' = x_1' \cdot x_2'$ $\quad (x_1 \cdot x_2)' = x_1' + x_2'$ (' 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

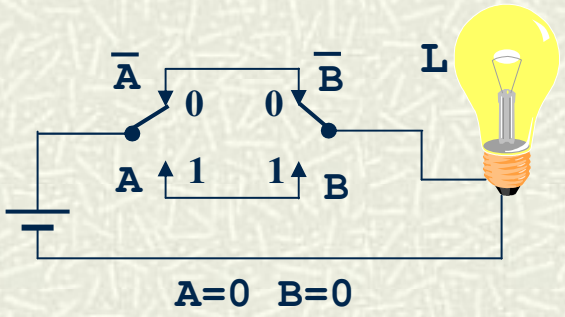
x_1	x_2	XOR
0	0	0
0	1	1
1	0	1
1	1	0

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

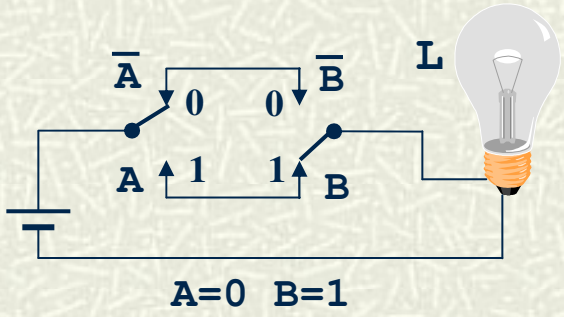
$$XOR = \overline{x}_1 x_2 + x_1 \overline{x}_2$$

Un circuito con due interruttori

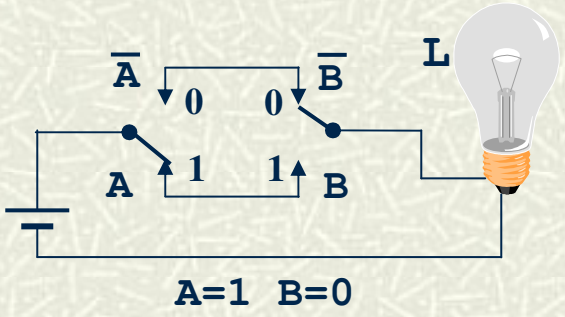
- I due interruttori corrispondono a due variabili (**A,B**) a valori booleani – le variabili assumono i due valori 0 e 1 che corrispondono alle due posizioni dell'interruttore



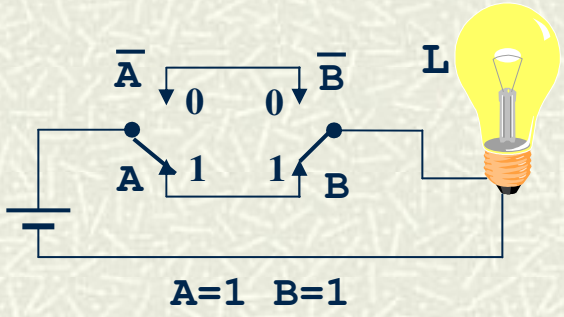
A=0 B=0



A=0 B=1



A=1 B=0



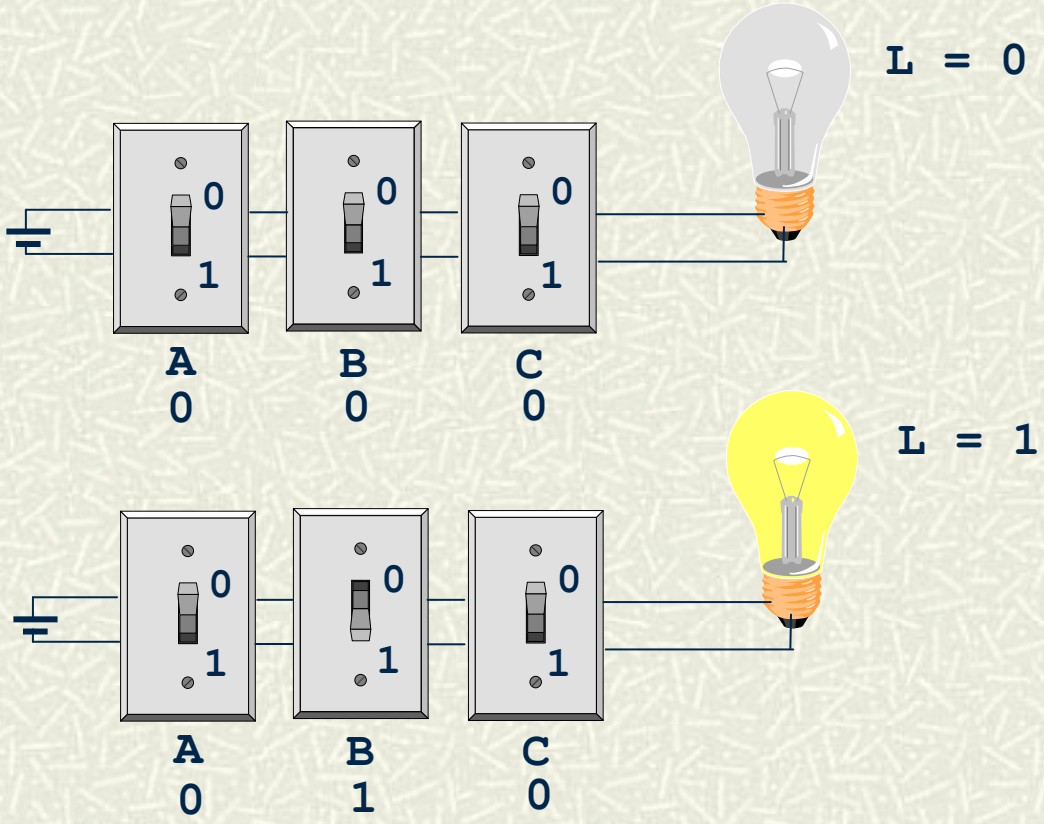
A=1 B=1

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

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

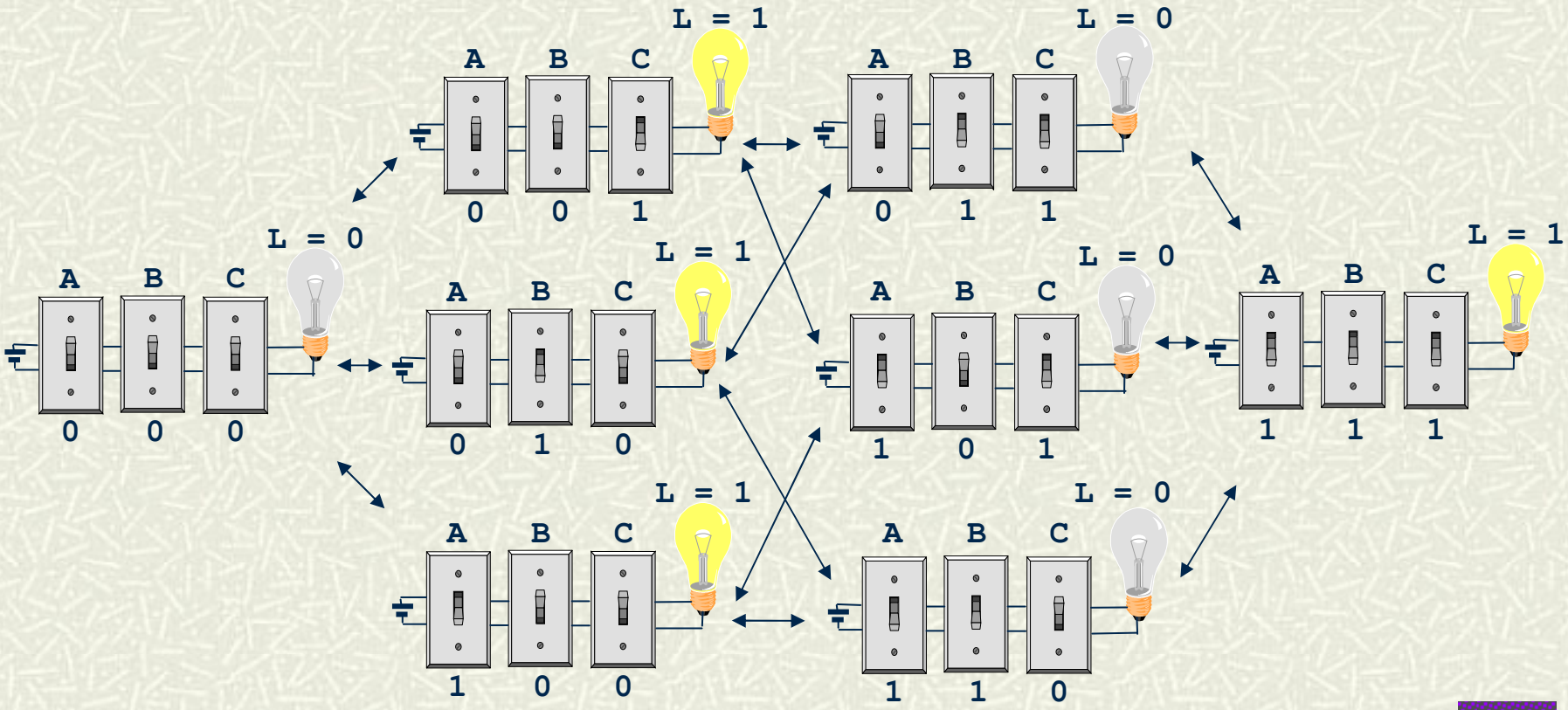
Un esercizio

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



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

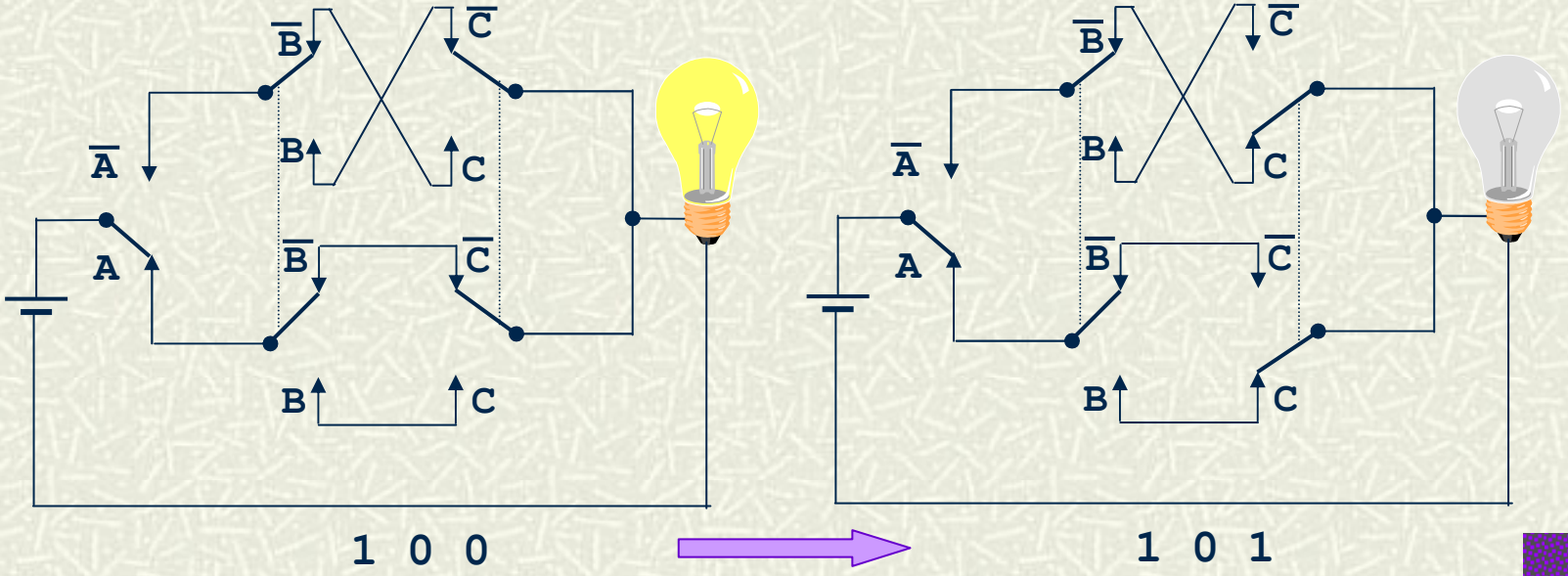
$$L = \overline{A} \times \overline{B} \times C + \overline{A} \times B \times \overline{C} + A \times \overline{B} \times \overline{C} + A \times B \times C$$

Come collegare gli interruttori

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

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

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



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 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**)

Esempi:

Forma polinomia

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

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

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

Dal bit al byte

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

$$b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0$$

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

00000000
00000001
00000010
00000011
……
11111110
11111111

$2^8 = 256$ valori distinti

- 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 $(43)_{10}$

$$
\begin{aligned}
43 : 2 &= 21 + 1 \\
21 : 2 &= 10 + 1 \\
10 : 2 &= 5 + 0 \\
5 : 2 &= 2 + 1 \\
2 : 2 &= 1 + 0 \\
1 : 2 &= 0 + 1
\end{aligned}
$$

resti

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 $(0{,}45)_{10}$

$0{,}21875 \times 2 = 0{,}4375$
$0{,}4375 \times 2 = 0{,}875$
$0{,}875 \times 2 = 1{,}75$
$0{,}75 \times 2 = 1{,}5$
$0{,}5 \times 2 = 1{,}0$

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

$0{,}45 \times 2 = 0{,}9$
$0{,}90 \times 2 = 1{,}8$
$0{,}80 \times 2 = 1{,}6$
$0{,}60 \times 2 = 1{,}2$
$0{,}20 \times 2 = 0{,}4$ etc.

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

Da binario a decimale

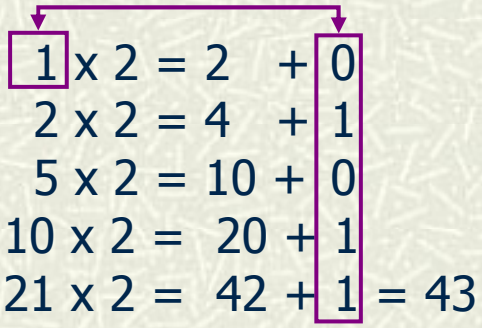
- Oltre all'espansione esplicita in potenze del 2 – **forma polinomia**…

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

- …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



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

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: $B^2 - 7B - 8 = 0$

  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)

0000	+0
0001	+1
0010	+2
0011	+3
0100	+4
0101	+5
0110	+6
0111	+7

positivi

1000	–0
1001	–1
1010	–2
1011	–3
1100	–4
1101	–5
1110	–6
1111	–7

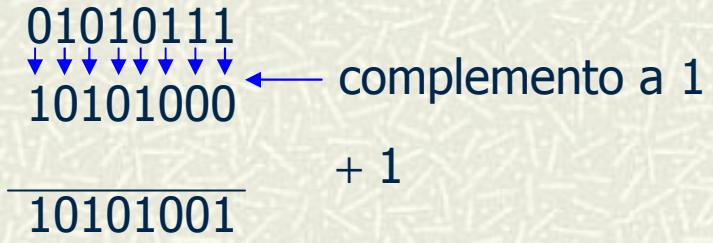
negativi

Complemento a 2

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

$$2^n - (N)_2 = 10\underbrace{\ldots\ldots}_{n}0 - (N)_2$$

- 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



100000000	2^8
011111111 −	2^8-1
01010111	N
10101000	2^8-1-N
10101001	$2^8-1-N+1$

Interi in complemento a 2

- I **numeri positivi** sono rappresentati (come) in modulo e segno
- I **numeri negativi** sono rappresentati in complemento a 2 $\Rightarrow$ 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

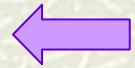
- Le regole per l'addizione di due bit sono

  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
```

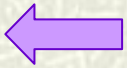
Sottrazione binaria – 1

- 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

$N_1 - N_2 = N_1 + (2^n - N_2) - 2^n$ ← si trascura il bit n+1

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

1. Si calcola il complemento a 2 di N_2
2. Si somma N_1 con il complemento a 2 di N_2
3. Si trascura il bit più significativo del risultato

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

$$\begin{array}{r} 010001 + \\ 111011 \\ \hline 1001100 \end{array} \longrightarrow (12)_{10}$$

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 $\rightarrow$ [–16,+15]

14 +	01110 +		–8 +	11000 +	
10	01010		–10	10110	
24	11000	–8	–18	101110	+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

Esempio complemento A2

00110 +	(6)
10110 =	(-10)
11100	(-4)

11010 +	(-6)
11011 =	(-5)
10101	(-11)

- $-2^4 + 2^3 + 2^2 = -4$
- $-2^4 + 2^2 + 1 = -11$

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	11000	→ -8	-18	101110	→ +14

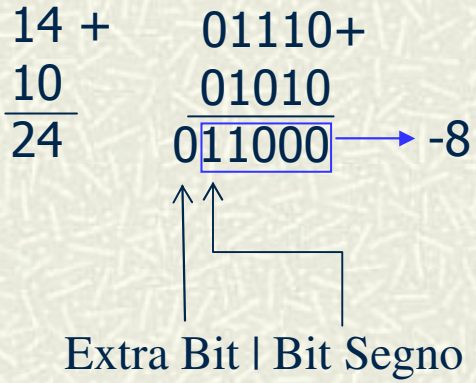
- Per evitare l'overflow occorre aumentare il numero di bit utilizzati per rappresentare gli operandi
- C'è overflow se c'è riporto al di fuori del bit di segno e non sul bit di segno, o se c'è riporto sul bit di segno, ma non al di fuori

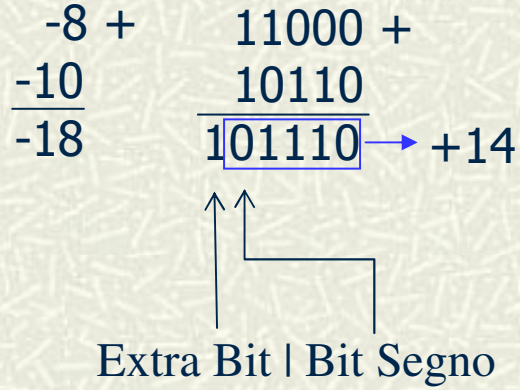
Punteggio nei vecchi videogame… sorpresa per i campioni!

0111 1111 1111 1111 + 1 = 1000 0000 0000 0000
32767 + 1 = -32768

Regola Overflow

Esempio: 5 bit — [-16,+15]





- Overflow:
  - solo trabocco nel bit segno
  - solo trabocco nell' extra bit

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
```

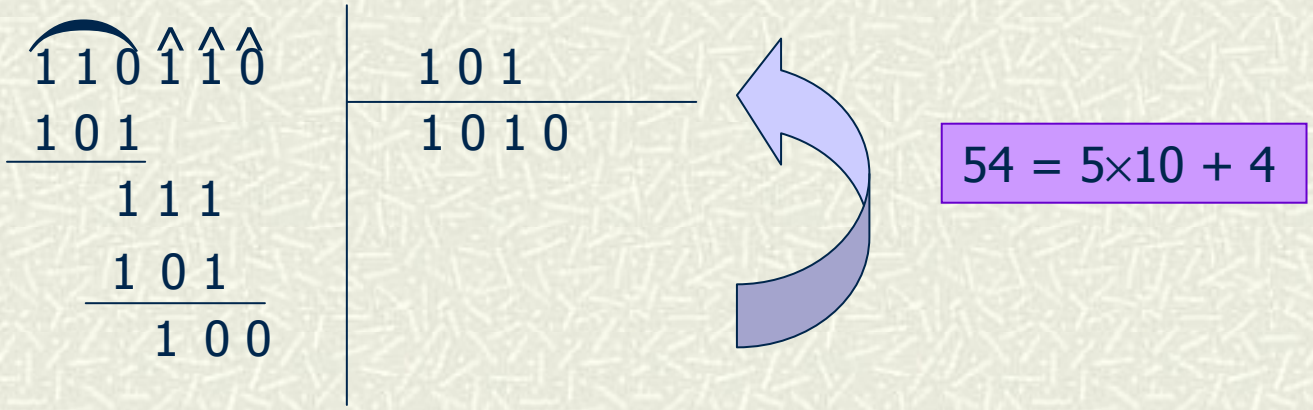
- Moltiplicare per 2^n corrisponde ad aggiungere n zeri in coda al moltiplicando

110011 x 10000 = 1100110000 ← 816

51 ↑ ↑ × 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 $A = B \times Q + R$
- La divisione binaria si compone di una serie di sottrazioni



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:

$$(A-C)+B = (A+B)-C,$$

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

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

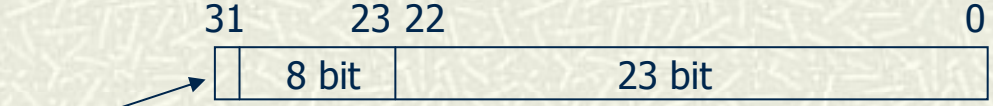
b) Si determini, se esiste, la base $b \leq 16$ di un sistema di numerazione tale che

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

Numeri in virgola mobile

- La rappresentazione dei numeri in virgola mobile è in relazione con la **notazione scientifica** (es. $1.2 \times 10^2 = 120$)
- 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



Segno Esponente (o Caratteristica) Mantissa

Il valore è

$(-1)^S \; 1.M \times 2^{E-127}$ se $E \neq 0$

$(-1)^S \; 0.M \times 2^{-126}$ 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)