

Calcolatori Elettronici I - lezione 1

Libri di testo:

- ❖ B. Fadini, C. Savy "Fondamenti di Informatica II", ed. Liguori

Materiale aggiuntivo:

- ❖ B. Fadini, C. Savy "Fondamenti di Informatica I", ed. Liguori
- ❖ Dispense integrative

Acknowledgements

- Hanno contribuito alla realizzazione di questo corso:
 - Prof. Bruno Fadini
 - Ing. Roberto Canonico
 - Ing. Giuseppe De Pietro
 - Ing. Marcello Esposito
 - Ing. Pasquale Foggia
 - Ing. Antonio Pescapè
 - Ing. Antonio Picariello
 - Ing. Carlo Sansone
 - Ing. Luigi Romano

Lezione 1

1ª parte

Obiettivi del corso

❖ **Architettura dei calcolatori elettronici**

Quali sono i componenti di un calcolatore, come sono interconnessi, come interagiscono per portare a termine l'elaborazione (Processore, Memoria e sottosistema di I/O)

❖ **Linguaggio del processore**

Le istruzioni del processore, la programmazione in linguaggi assemblativi, corrispondenza tra linguaggi ad alto livello e linguaggio macchina

Strumenti necessari

❖ Elementi di Reti Logiche

Algebra di Boole, Reti combinatorie, Porte Logiche, Automi e Macchine sequenziali, Flip-Flop, Contatori e Registri a Scorrimento, Macchine Elementari.

❖ Elementi di Teoria dei Codici

Macchine per il trattamento dei codici, Multiplexer e Demultiplexer.

❖ Rappresentazione dei numeri

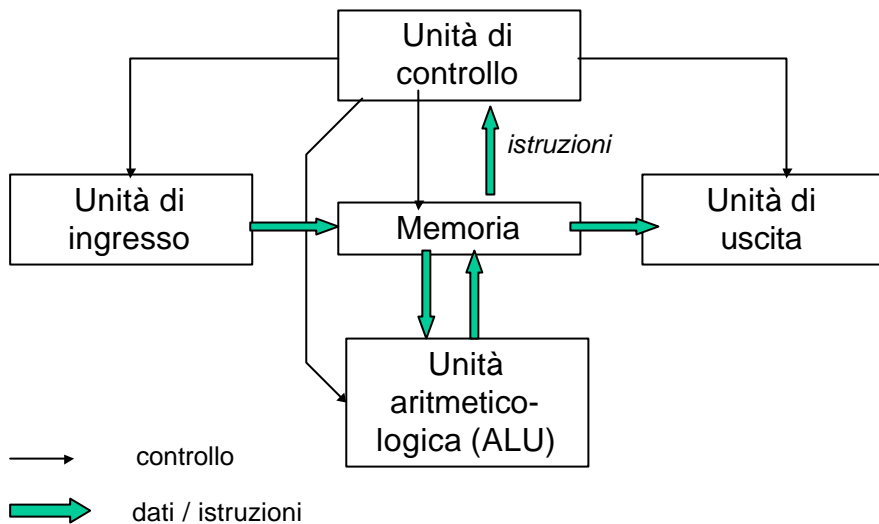
Macchine aritmetiche, Addizionatori di interi positivi e Addizionatori di numeri relativi

Calcolatore Elettronico

Un *calcolatore elettronico* è un sistema per l'elaborazione delle informazioni dotato delle seguenti caratteristiche:

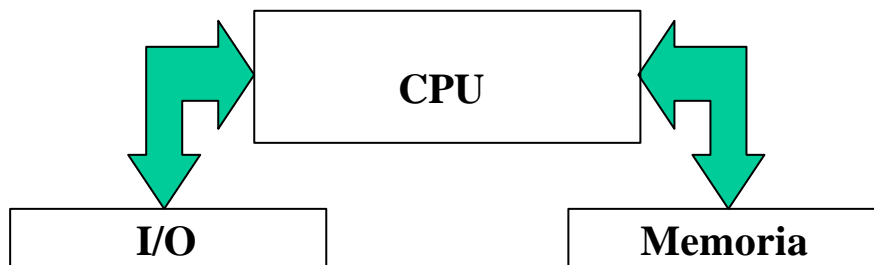
- è un sistema *numerico*
- è un sistema *automatico*
- è un sistema *a programma registrato*
- è realizzato mediante circuiti *elettronici*

Modello di Von Neumann



Calcolatore Elettronico: architettura

- Un calcolatore elettronico è costituito da tre sottosistemi principali:
 - processore o CPU (*Central Processing Unit*)
 - memoria centrale
 - sottosistema di input/output (I/O)



Calcolatore Elettronico: il processore

- La capacità elaborativa del calcolatore risiede nel processore; il processore è in grado di eseguire un set di azioni elaborative elementari più o meno complesse
- Le *istruzioni* sono comandi espliciti che
 - governano il trasferimento di informazioni sia all'interno del calcolatore sia tra il calcolatore e i dispositivi di I/O
 - specificano le operazioni aritmetiche e logiche che devono essere effettuate
- I *dati* di ingresso e di uscita dell'elaborazione, nonché la stessa sequenza di istruzioni sono immagazzinati nella memoria centrale
- Il processore preleva ed esegue le istruzioni dalla memoria una ad una
- Una sequenza di istruzioni memorizzate nella memoria centrale costituisce un *programma*

Lezione 1

2^a parte

L'algebra di Boole - richiami

Operazioni fondamentali sui bit:

x	y	x AND y
0	0	0
0	1	0
1	0	0
1	1	1

Congiunzione

x AND y si indica anche con $x \cdot y$

x	y	x OR y
0	0	0
0	1	1
1	0	1
1	1	1

Disgiunzione

x OR y si indica anche con $x + y$

x	NOT x
0	1
1	0

Negazione

NOT x si indica anche con \bar{x}

L'algebra di Boole - alcune proprietà (1)

- **Proprietà commutativa:**

$$x \text{ AND } y = y \text{ AND } x$$

$$x \text{ OR } y = y \text{ OR } x$$

- **Proprietà associativa:**

$$(x \text{ AND } y) \text{ AND } z = x \text{ AND } (y \text{ AND } z)$$

$$(x \text{ OR } y) \text{ OR } z = x \text{ OR } (y \text{ OR } z)$$

per la propr. associativa posso definire AND e OR a più di due operandi (es. $x \text{ AND } y \text{ AND } z$)

- **Proprietà di idempotenza e assorbimento:**

$$x \text{ AND } x = x$$

$$x \text{ OR } x = x$$

$$x \text{ AND } (x \text{ OR } y) = x$$

$$x \text{ OR } (x \text{ AND } y) = x$$

L'algebra di Boole - alcune proprietà (2)

- **Proprietà distributiva**

$$x \text{ AND } (y \text{ OR } z) = (x \text{ AND } y) \text{ OR } (x \text{ AND } z)$$

$$x \text{ OR } (y \text{ AND } z) = (x \text{ OR } y) \text{ AND } (x \text{ OR } z)$$

- **Proprietà di convoluzione**

$$\text{NOT } (\text{NOT } x) = x$$

- **Proprietà del minimo e del massimo:**

$$x \text{ AND } 1 = x \quad x \text{ AND } 0 = 0$$

$$x \text{ OR } 0 = x \quad x \text{ OR } 1 = 1$$

- **Leggi di De Morgan:**

$$\text{NOT } (x \text{ AND } y) = (\text{NOT } x) \text{ OR } (\text{NOT } y)$$

$$\text{NOT } (x \text{ OR } y) = (\text{NOT } x) \text{ AND } (\text{NOT } y)$$

Funzioni booleane

$y = f(x_1, x_2, \dots, x_n)$ è una funzione booleana se ad ogni ennupla di valori booleani x_1, \dots, x_n associa un valore booleano y

Due esempi:

x_1	x_2	$f(x_1, x_2)$	x_1	x_2	$f(x_1, x_2)$
0	0	0	0	0	1
0	1	1	0	1	0
1	0	1	1	0	0
1	1	0	1	1	1

Questa funzione è detta OR esclusivo, o XOR

Questa funzione è detta equivalenza, o EQU

Funzioni booleane

- ❑ Anche AND, OR e NOT sono funzioni booleane.
- ❑ AND, OR e NOT vengono dette *funzioni fondamentali* dell'algebra.
- ❑ Se $F = \{\text{AND}, \text{OR}, \text{NOT}\}$ si dice algebrica o razionale.

$$\text{Es. } y = bc(\underline{a}d + \underline{b} + c) + \underline{c}(d + \underline{a})(b + c)$$

Funzioni booleane

- Una funzione booleana può essere definita mediante una tabella : tabella di verità
- La tabella definisce il valore della variabile dipendente per tutte le possibili combinazioni delle variabili indipendenti
- Una funzione di n variabili $f(x_1, x_2, \dots, x_n)$ è definita su 2^n punti e quindi la tabella di verità avrà 2^n righe.

Funzioni booleane

- E' possibile passare dalla forma tabellare alla forma algebrica ottenendo la forma canonica di primo tipo o tipo P e/o la forma canonica di secondo tipo o tipo S:
 - dimostrazione
 - procedimento pratico
 - considerazioni e conseguenze:
 - tutte le funzioni booleane sono algebriche
 - l'insieme {AND, OR, NOT} è funzionalmente completo.
- Proprietà dell'algebra di Boole e livelli di una funzione: minimizzazione. (Esempio)

Insiemi funzionalmente completi

- Una funzione può talora essere espressa mediante "funzione delle funzioni appartenenti ad un insieme F".
- Abbiamo visto che qualsiasi funzione booleana può essere calcolata applicando le funzioni AND, OR, e NOT.
Ad esempio:
$$x \text{ XOR } y = (x \text{ AND NOT } y) \text{ OR } (y \text{ AND NOT } x)$$

Per questo, l'insieme {AND, OR, NOT} si dice *funzionalmente completo*.
- Esistono altri insiemi funzionalmente completi. Si noti che grazie alle leggi di De Morgan si può costruire la AND da {OR, NOT}, oppure la OR da {AND, NOT}. Quindi anche {AND, NOT} e {OR, NOT} sono insiemi funzionalmente completi.

Reti logiche

I valori booleani possono essere rappresentati da grandezze elettriche. Ad esempio:

0 \Leftrightarrow tensione di 0 Volt

1 \Leftrightarrow tensione di +5 Volt

In tal caso le funzioni booleane possono essere realizzate mediante circuiti elettronici detti *reti logiche*.

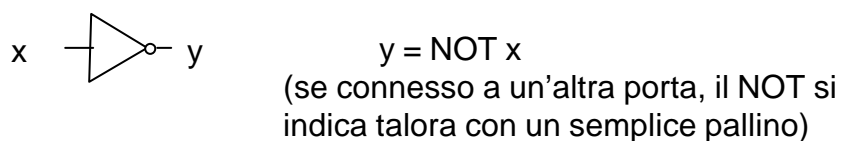
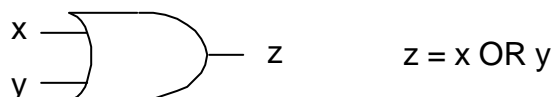
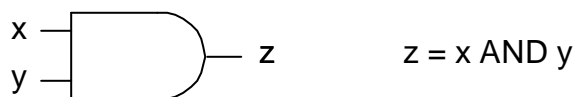
Nelle reti logiche *unilaterali*, le uscite della rete corrispondono a valori di grandezze elettriche misurate in opportuni punti del circuito; il flusso dell'elaborazione procede fisicamente in un'unica direzione, dai segnali di ingresso verso i segnali di uscita.

Nelle reti logiche *bilaterali*, invece, l'uscita della rete è determinata dalla presenza o dall'assenza di "contatto" tra due punti della rete.

Porte logiche (gates)

Circuiti logici elementari che realizzano le operazioni fondamentali. Le reti logiche si costruiscono connettendo più porte logiche.

Simboli delle principali porte logiche:



Operatori logici generalizzati (1)

Dato un vettore di variabili booleane $X = (x_1, x_2, \dots, x_n)$, e una variabile booleana α , indicheremo con la notazione:

$$Y = \alpha \text{ OP } X \quad (\text{dove } OP \text{ è un operatore booleano})$$

l'operazione che produce il vettore booleano Y così definito:

$$\begin{aligned} Y &= (y_1, y_2, \dots, y_n) \text{ con} \\ y_1 &= \alpha \text{ OP } x_1 \\ &\dots \dots \dots \\ y_n &= \alpha \text{ OP } x_n \end{aligned}$$

Esempio:

α AND X ha come risultato il vettore formato da:
(α AND x_1 , α AND x_2 , ..., α AND x_n)

Operatori logici generalizzati (2)

Dati due vettori di variabili booleane $X = (x_1, x_2, \dots, x_n)$ e $Y = (y_1, y_2, \dots, y_n)$ indicheremo con la notazione:

$$Z = X \text{ OP } Y \quad (\text{dove } OP \text{ è un operatore booleano})$$

l'operazione che produce il vettore booleano Z così definito:

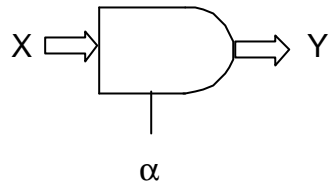
$$\begin{aligned} Z &= (z_1, z_2, \dots, z_n) \text{ con} \\ z_1 &= x_1 \text{ OP } y_1 \\ &\dots \dots \dots \\ z_n &= x_n \text{ OP } y_n \end{aligned}$$

Esempio:

X OR Y ha come risultato il vettore formato da:
(x_1 OR y_1 , x_2 OR y_2 , ..., x_n OR y_n)

Porte logiche generalizzate (1)

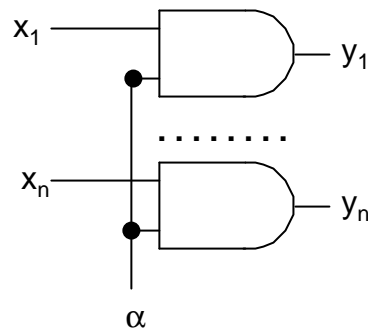
Rappresentazione simbolica:



$$Y = \alpha \text{ AND } X$$

Per $\alpha = 0 \Rightarrow Y = 0_n$
 $\alpha = 1 \Rightarrow Y = X$

Circuito equivalente:



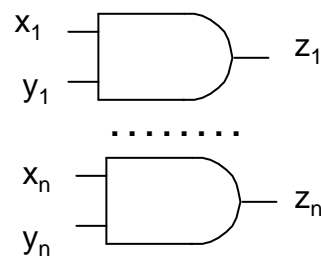
Porte logiche generalizzate (2)

Rappresentazione simbolica:



$$Z = X \text{ AND } Y$$

Circuito equivalente:



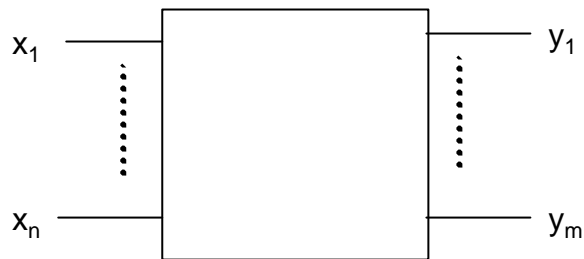
Macchine combinatorie (1)

Reti logiche con n ingressi x_1, x_2, \dots, x_n e m uscite y_1, y_2, \dots, y_m che realizzano la corrispondenza:

$$y_1 = f_1(x_1, x_2, \dots, x_n)$$

.....

$$y_m = f_m(x_1, x_2, \dots, x_n)$$



Macchine combinatorie (2)

In una macchina combinatoria i valori delle uscite dipendono esclusivamente dai valori degli ingressi.

In una macchina combinatoria ideale tale dipendenza è istantanea; in una macchina reale c'è sempre un ritardo tra l'istante in cui c'è una variazione in uno degli ingressi e l'istante in cui l'effetto di questa variazione si manifesta sulle uscite.

E' importante notare come

- ciascuna y_i può essere decomposta in funzioni componenti
- due distinte y_i possono contenere una identica funzione componente

Ciò comporta, ad esempio, una potenziale diminuzione di porte elementari rispetto ad una realizzazione indipendente delle y_i .