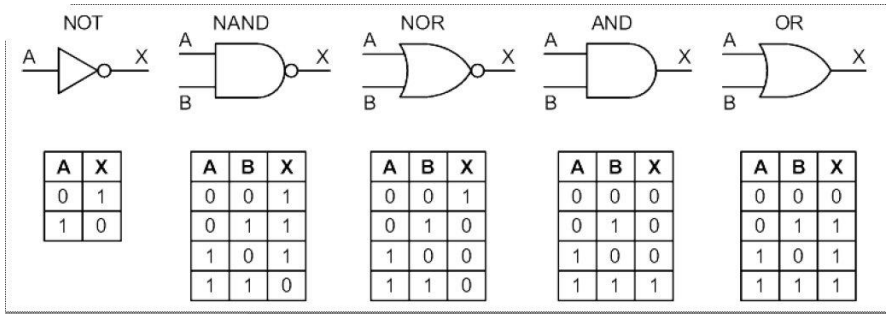
Cenni circuiti, reti combinatorie, reti sequenziali

Porte logiche di base



A	X
0	1
1	0

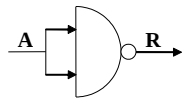
A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

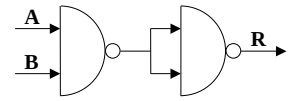
A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

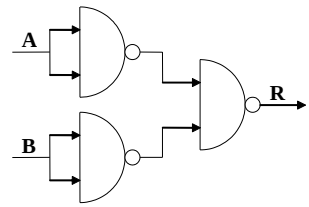
NOT



AND



OR



Quindi NAND o NOR sono complete ➔ circuiti con solo porte NAND o solo porte NOR.

Reti combinatorie

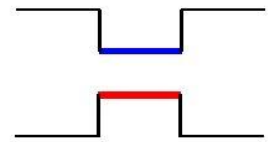
- Rete combinatoria: insieme di porte logiche connesse il cui output in un certo istante è funzione solo dell'input in quell'istante
- N input binari e m output binari
- Ad ogni combinazione di valori di ingresso corrisponde **una ed una sola** combinazione di valori di uscita

Reti combinatorie (segue)

- Vediamo alcuni esempi di circuiti:
  - ✓ I segnali sono discretizzati e di solito assumono solo due stati:

0 / FALSO / [0..1] Volt

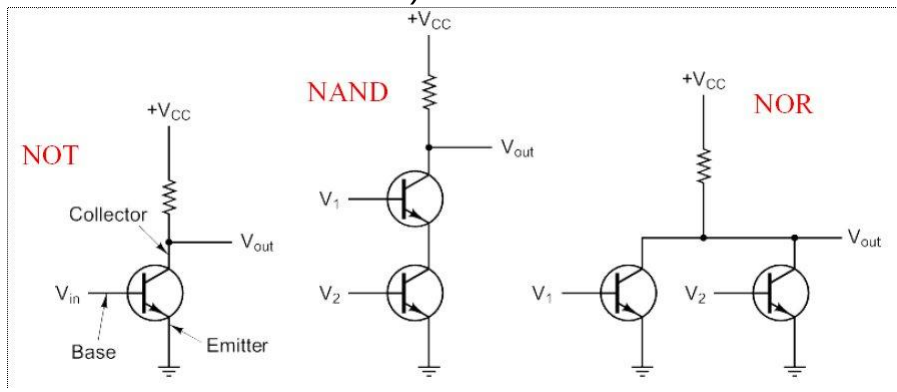
1 / VERO / [2..5] Volt



  - ✓ I circuiti più complessi sono realizzati attraverso la combinazione di circuiti semplici (porte logiche)

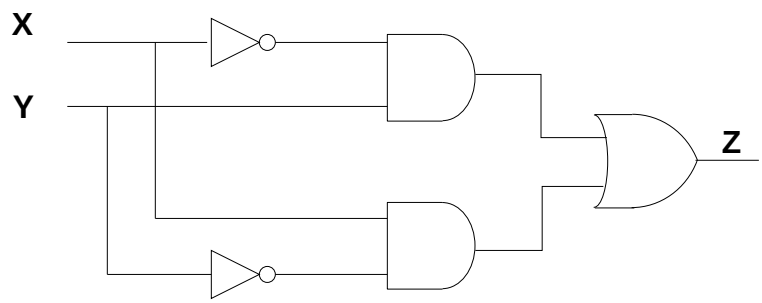
Reti combinatorie (segue)

- Porte Logiche:
  - ✓ Sono realizzate tramite transistor (sono in pratica interruttori automatici)

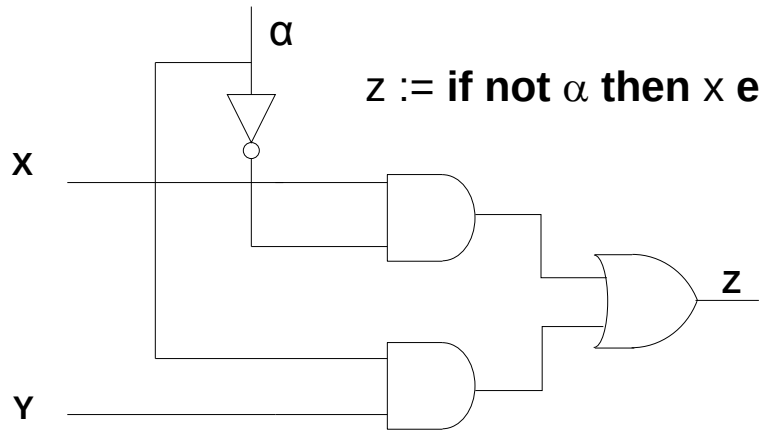


Confrontatore

z := **not** (x = y)

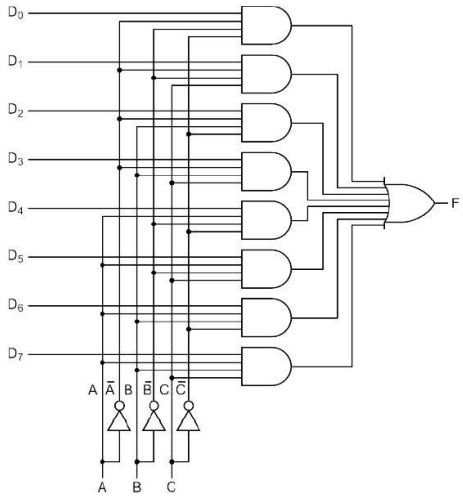


Multiplexer (selettore)



z := **if not** α **then** x **else** y

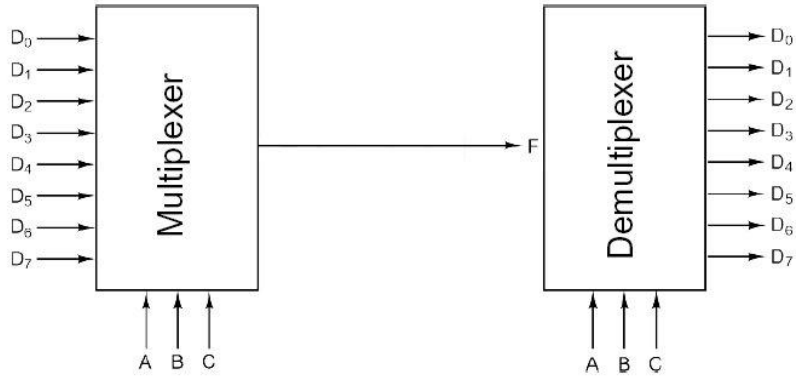
Multiplexer (o selettore) 2^n a 1



- Solo uno degli ingressi viene trasferito all'output
- n ingressi di controllo: indicano l'ingresso da trasferire
  - ✓ 2^n linee di input (D_0 - D_7)
  - ✓ n linee di controllo (A,B,C)
  - ✓ 1 linea di output (F)
- ✓ Per ogni combinazione degli ingressi di controllo, 2^n -1 delle porte AND hanno uscita 0, l'altra fa uscire l'ingresso

Demultiplexer

- ✓ è il circuito inverso del Multiplexer ed è spesso usato in combinazione con quest'ultimo (seleziona comunicazione fra linee)

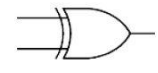


Reti combinatorie multi-funzione

- Operatori aritmetico logici a specifica **diretta**
  - ✓ Addizione, sottrazione, traslazione, rotazione, incremento, decremento, etc.
- Reti aritmetico logiche **multi-funzione**
  - ✓ Eseguono **una** delle operazioni suddette a seconda del valore assunto da un certo numero di ingressi di controllo
  - ✓ Si usano per implementare le **ALU** (*arithmetic logic unit*)

Comparatore

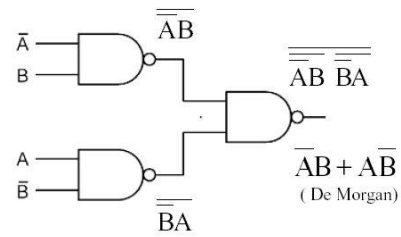
- ✓ Compara due ingressi e produce un output che indica la uguaglianza (0) o meno (1) degli ingressi
- ✓ Esempio di comparatore ad 1 bit: si realizza con una porta XOR



XOR (OR Esclusivo)

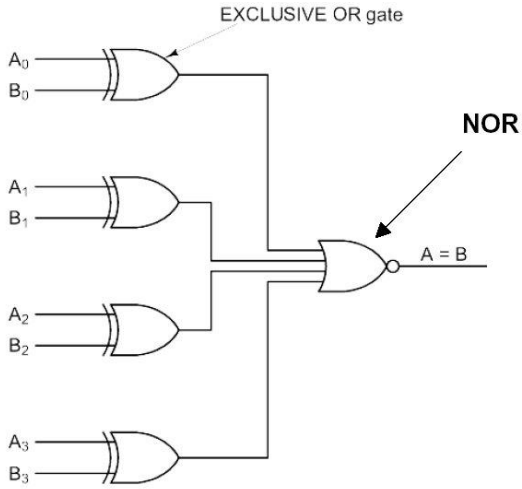
Simbolo ⊕

A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0



Realizzazione con porte NAND

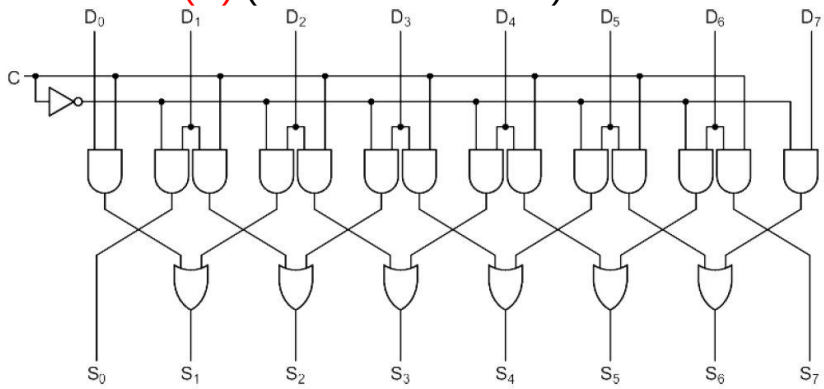
Comparatore a più bit



- ✓ Comparatori ad 1 bit vengono collegati tramite una porta NOR
- ✓ L'output vale 1 solo se tutti gli output dei singoli comparatori ad 1 bit valgono 0
- ✓ (Ai=Bi) per ogni i, cioè A=B

Traslatore (shifter)

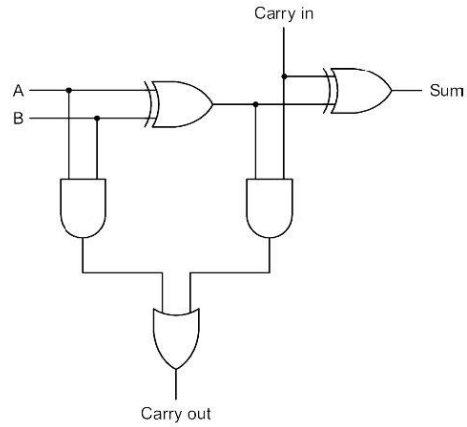
- ✓ Trasla i bit in ingresso (D) di una posizione, a sinistra o a destra a seconda del valore del bit di controllo (C) (C=1 shift a destra)



Reti combinatorie

- Circuiti Aritmetici: Sommatore
  - ✓ Full-adder a 1 bit, prevede un possibile riporto

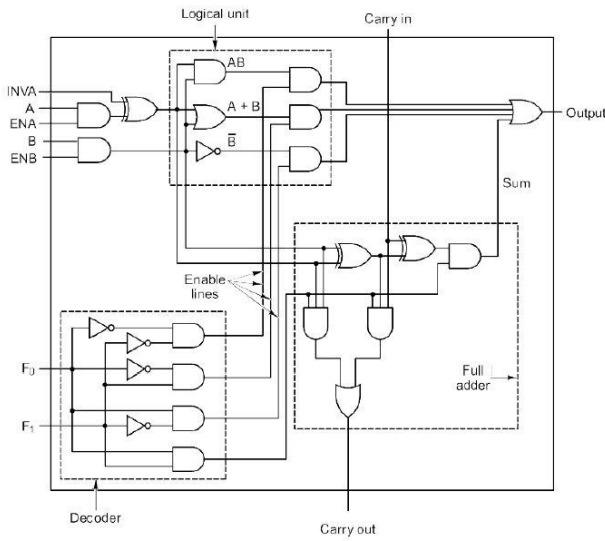
A	B	Carry in	Sum	Carry out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



Reti combinatorie

- Circuiti Aritmetici: Sommatore a n bit
  - ✓ Si collegano n full-adder a 1 bit fra di loro
  - ✓ Il carry out del bit i-esimo deve essere collegato al carry in del bit i+1-esimo (i=0,…,n-2)
- Circuiti Aritmetici: Aritmetic Logic Unit (ALU)
  - ✓ Circuito capace di eseguire operazioni aritmetiche e logiche elementari su due parole (di n bit) di input A e B (es. AND, OR, complemento, somma aritmetica,…)

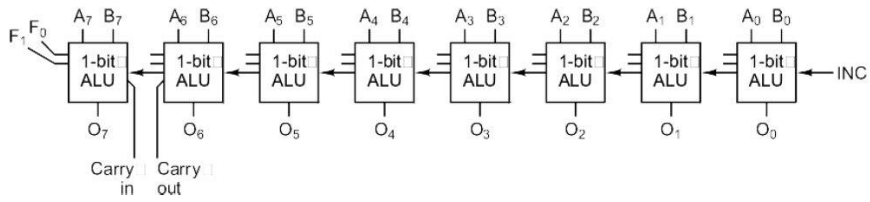
Reti combinatorie (segue)



- ALU ad 1bit che realizza 4 operazioni (selezionate da F_0 e F_1)
- AB, A or B, not(B), A+B
- ENA, ENB: per forzare a 0 gli input A e B
- INVA, INVB: per invertire gli input

Reti combinatorie (segue)

- ALU a n bit
  - ✓ Si ottiene concatenando n ALU ad 1 bit
  - ✓ F_0 e F_1 collegati a tutte le ALU
  - ✓ Riporto intermedio propagato da una ALU alla successiva
  - ✓ INC (corrispondente al carry in della ALU “0”) permette di sommare 1 al risultato in caso di addizione



Reti combinatorie

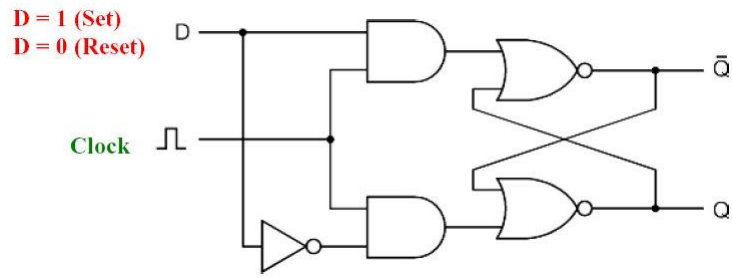
- Utili per implementare la ALU e la connessione tra parti della CPU
- Non sono in grado di memorizzare uno stato, quindi non possono essere usate per implementare la memoria
- Per questo servono le reti sequenziali
  - L'output dipende non solo dall'input corrente, ma anche dalla storia passata degli input

Flip flop

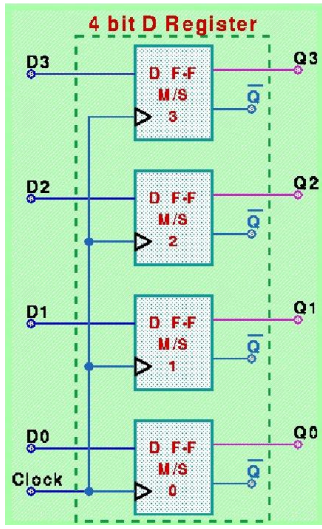
- Forma più semplice di una rete sequenziale
- Tanti tipi, ma due proprietà per tutti:
  - Bistabili:
    - Possono trovarsi in uno di due stati diversi
    - In assenza di input, rimangono nello stato in cui sono
    - Memoria per un bit
  - Due output
    - Uno è sempre il complemento dell'altro

Flip flop D

- Un solo input (D)
- Usa segnale di clock per stabilizzare l'output (sincronizzazione)
- Quando clock =0, gli output dei due AND sono 0 (stato stabile)
- Quando clock=1, gli input sono uno l'opposto dell'altro ➔ Q=D



Registro di tipo D



- è il circuito sincrono più semplice che realizza un registro
- Memorizzazione (store): dati presentati in ingresso e clock da 0 a 1 (uscita riproduce ingresso)
- Mantenimento (hold): clock da 1 a 0 (poi costante); l'uscita rimane invariata indipendentemente dal valore degli ingressi

(Super)Cenni di Microprogrammazione

Controllo

- La Parte (o Unità) Controllo (PC) della CPU si fa carico di realizzare il flusso di controllo appropriato per ogni istruzione tramite l'invio di opportuni segnali di controllo alla Parte Operativa

- **Requisiti funzionali** (cioè le funzioni che la PC deve eseguire):
  - Definire gli elementi di base del processore
  - Definire le micro-operazioni che il processore esegue
  - Determinare le funzioni che la PC deve effettuare per l'esecuzione delle micro-operazioni

Elementi Base

- Come visto in precedenza gli elementi di base sono:
  - ALU
  - Registri
  - Bus dati interno
  - Bus dati esterno
  - Unità di controllo

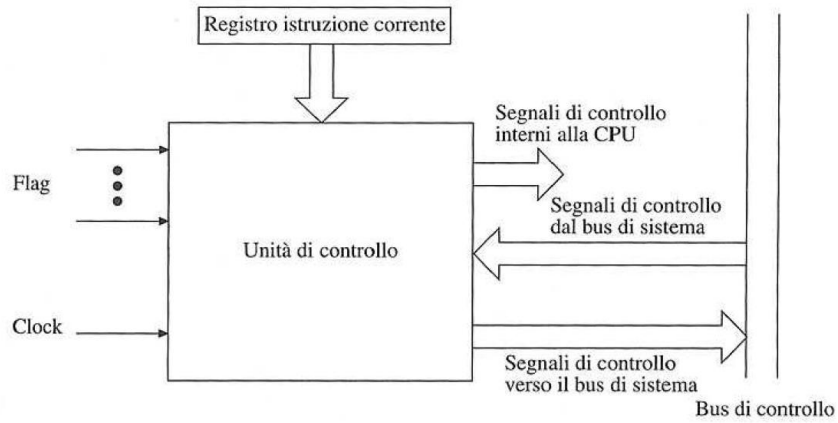
Tipologie di micro-istruzioni

- Trasferimento dati da un registro all'altro
- Trasferimento dati da un registro ad un'interfaccia esterna
- Trasferimento dati da un'interfaccia esterna ad un registro
- Esecuzione di una operazione aritmetica o logica, che utilizzi i registri come input e output

Funzioni della PC

- Quindi i compiti base della PC sono:
  - **Serializzazione**: determina la "giusta" sequenza di micro-operazioni da eseguire in funzione del codice operativo dell'istruzione
  - **Esecuzione**: provoca l'esecuzione di micro-operazioni
- La realizzazione di questi compiti base passa attraverso la generazioni di opportuni *segnali di controllo*

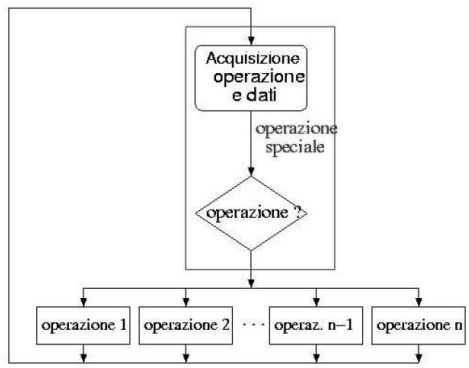
Segnali di Controllo



Realizzazione della PC

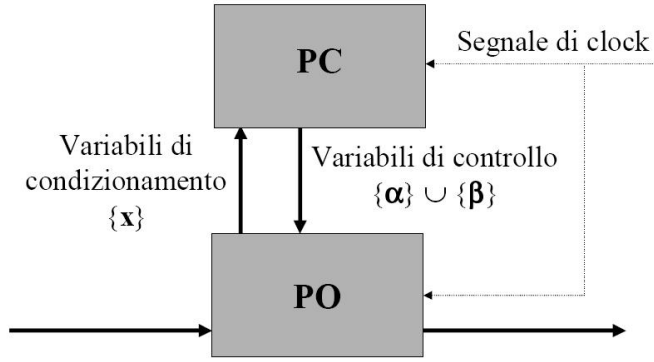
- Ci sono due alternative:
  - **Cablata**:
    - si realizza direttamente tramite circuiti digitali (livello di astrazione 0);
    - soluzione tipica di architetture RISC;
  - **Microprogrammata** (la trattiamo di seguito):
    - si realizza tramite microprogrammazione (livello di astrazione 1);
    - soluzione tipica di architetture CISC;
    - permette una maggior flessibilità in fase di progettazione: rende facile modificare le sequenze di micro-operazioni

Livello Firmware

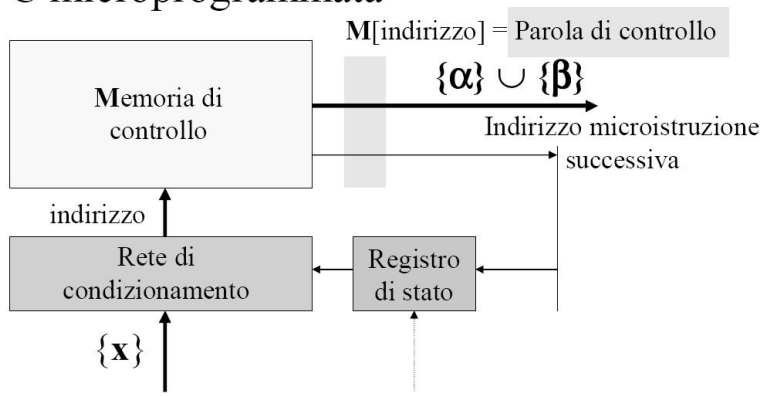


- Il *μprogramma* di una unità riunisce i frammenti di programma delle diverse operazioni (esterne e speciale)
- Il *μprogramma* ha una struttura ciclica in cui si alterna l'esecuzione della operazione speciale con l'esecuzione della operazione esterna il cui codice e dati da eleborare sono stati acquisiti dalla operazione speciale

Livello Firmware (segue)



- PC microprogrammata



- {α} ∪ {β} designa la microoperazione richiesta