Docente: Maurizio QUARTA

Il corso di INFORMATICA

BIOTECNOLOGIE 2021/22

6 CFU

E-mail: maurizio.quarta@unisalento.it

mauroqfor@gmail.com

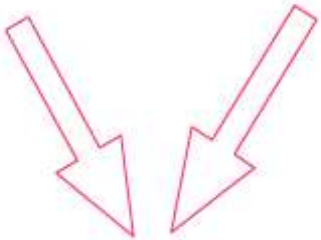
Tel: 0832-297532

Ricevimento: martedi' dalle 9.30 alle 11.30
altri giorni per appuntamento (Dip. Matem.)

http://mathematics.unisalento.it/quarta/index.html

Informatica???

Trattamento Automatico delle Informazioni



Informatica!!!

Scienza che si occupa di sviluppare **teorie**, **modelli** e **tecnologie** per gestire informazioni su sistemi **automatizzati**.

PERCHE' IL COMPUTER?

LIMITI DELLE CAPACITA' ELABORATIVE UMANE.

1. **VELOCITA' LIMITATA.**

2. **PESANTI LIMITI ALLA COMPLESSITA' DEI PROBLEMI AFFRONTABILI.**

3. **ELEVATA PROBABILITA' DI ERRORE.**

L'informatica è una scienza “nuova”, vasta e per nulla semplice!

L'informatica si occupa di:

- Architettura degli elaboratori;
- Algoritmi e loro implementazione sui calcolatori;
- Ottimizzazione dei procedimenti di calcolo;
- Simulazione di “situazioni reali” tramite modelli matematici;
- Basi Dati, loro strutturazione, imple4mentazione, ecc.;
- Linguaggi di programmazione;
-Simulazione dell’intelligenza dell’uomo
(Intelligenza Artificiale)
- etc.. etc..

UN PO’ DI STORIA

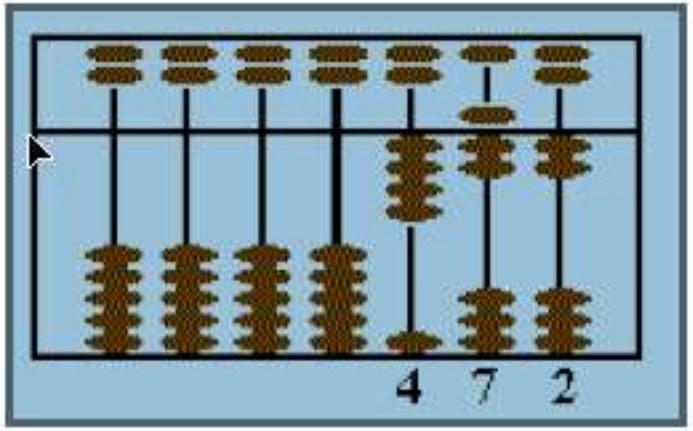
MANO

Il primo computer utilizzato dall' uomo è stata la **mano**.
Con l'uso delle mani gli egiziani riuscivano a rappresentare tutti i numeri fino al 9999.
Riuscivano ad effettuare addizioni sottrazioni e moltiplicazioni

Il termine inglese **digit** (cifra) deriva proprio dalla parola latina ***digitus*** **(dito)**

DIGITALE è ciò che è esprimibile in forma numerica (discreto)
ANALOGICO è ciò che è "privo di logica" (continuo)

ABACO



Abaco

Evoluto in maniera leggermente differente in Babilonia, Cina, Grecia e Impero Romano.

E’ tuttora in uso in Cina e Giappone

REGOLO

Regolo Calcolatore

Inventato nel 1650 dal matematico inglese E. Gunter.

Il principio di funzionamento si basa sulle proprietà dei logaritmi secondo cui il prodotto e il quoziente si ottengono, rispettivamente, con le operazioni di addizione e sottrazione e sull'impiego della sua scala logaritmica per moltiplicare e dividere

a*b = 10^(log(a)+log(b))

a/b = 10^(log(a)-log(b))

E' stato correntemente adoperato fino alla fine degli anni '60

L'era meccanica

1623 Wilhelm Schickard

1642 Blaise Pascal

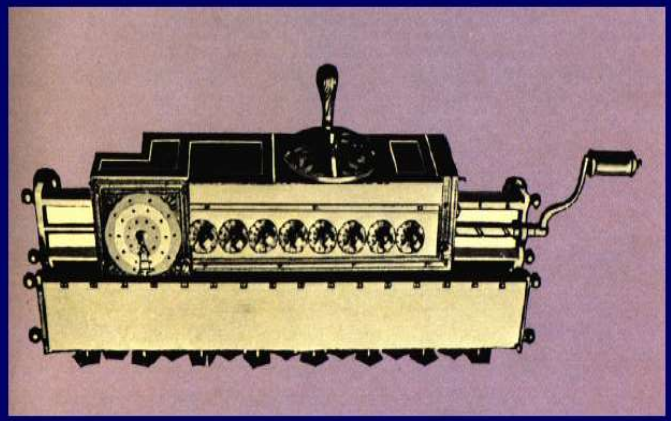
- 6 cifre decimali per 2 numeri (rotelle)
- somme e sottrazioni (complemento alla base)
- dispositivo per il riporto

1651 Gottfried Leibniz

- moltiplicazioni e divisioni realizzate con catene e pulegge
- addizioni e sottrazioni come Pascal
- precursore delle calcolatrici meccaniche a 4 operazioni, commercializzate a partire dal XIX secolo e usate ancora nella seconda metà del XX

Pascalina

Macchina calcolatrice di Leibniz



L'EVOLUZIONE

1794 Progetto francese per la stesura di tabelle matematiche, tra cui logN

- con $1 \leq N \leq 200.000$.
- 100 persone
- 2 anni di lavoro
- tutti i calcoli duplicati per verifica errori
- 17 grandi volumi

L'ERA MECCANICA

1823/1834 CHARLES BABBAGE

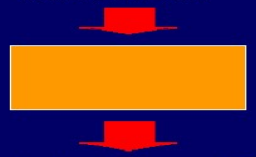
- DIFFERENCE ENGINE
- ANALYTICAL ENGINE

Difference Engine

- Solo addizioni
- Dedicato al calcolo di funzioni mediante il metodo delle differenze finite (sviluppi in serie polinomiali)

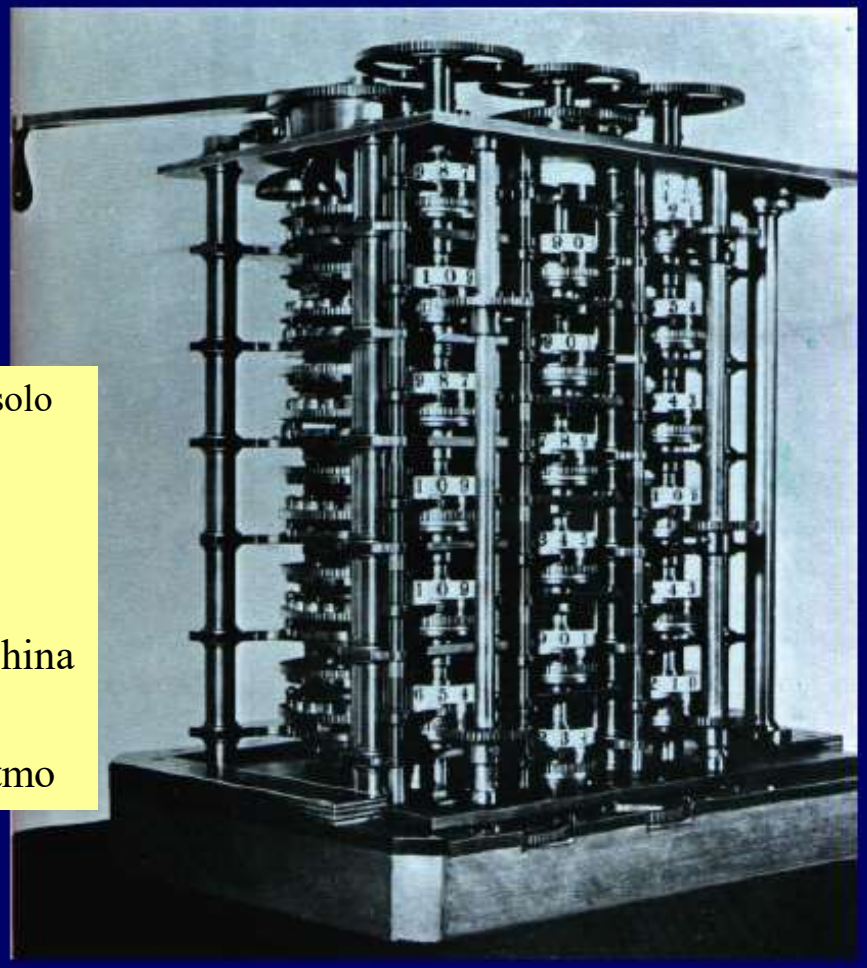
Difference Engine

- Polinomi di 6° grado
- Numeri di 20 cifre



- Progetto abbandonato

Difference engine



Dispositivo meccanico che era solo in grado di fare addizioni e sottrazioni.

Fu progettato per calcolare tabelle di numeri utili alla navigazione marina. La macchina era costruita

per utilizzare un solo algoritmo

Ben presto Babbage si stancò di una macchina che poteva lavorare con un solo algoritmo e cominciò a passare molto tempo a spendere grosse quantita' di denaro proprio e di fondi governativi, per la progettazione e la costruzione di una nuova macchina ciamata

ANALYTICAL ENGINE

Essa si componeva di quattro parti:

- STORE (MEMORIA)
- MILL (UNITA' DI CALCOLO)
- INPUT (LETTORE DELLE SCHEDE PERFORATE)
- OUTPUT (PERFORATORE E STAMPANTE)

Analytical Engine

- Qualsiasi operazione matematica

ANALYTICAL ENGINE

Meccanismo per alterare automaticamente la sequenza delle operazioni.

Progetto fallito per l'inadeguatezza della tecnologia meccanica di allora e per l'eccessiva complessita‘.

SVILUPPI SUCCESSIVI

1885

TASTIERA E STAMPA SU CARTA

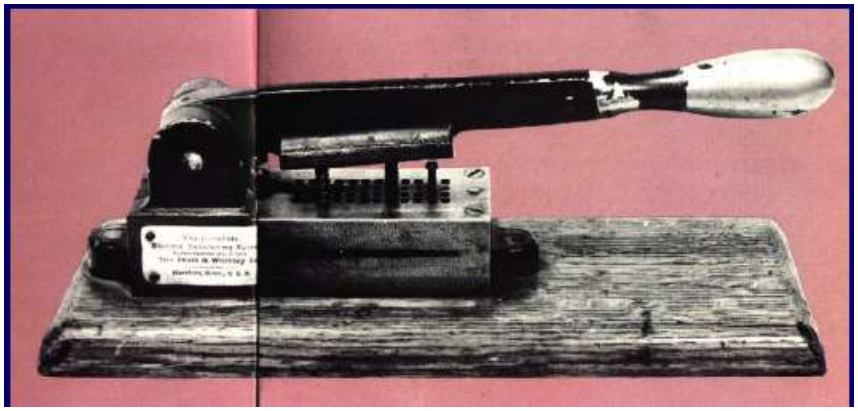
MOTORI ELETTRICI

1890

LETTORI ELETTRICI DI SCHEDE PERFORATE

CENSIMENTO U.S.

PERFORATORE DI SCHEDE



“Moderna” calcolatrice elettromeccanica

Sviluppi successivi

- 1896 Hollerith fonda la Tabulating Machine Company
- 1911 fusione con altre società

- Computing-Tabulating-Recording-Machine

- 1924 Ribattezzata in International Business Machine Corp. (IBM)

Sviluppi successivi

- 1938: Primo computer meccanico binario (Konrad Zuse) chiamato Z1
- 1941: Z3, il primo vero computer general-purpose programmabile, costruito con relay

Esempio di relay

PRIMA GENERAZIONE
VALVOLE TERMOIONICHE
(1945-1955)

Lo stimolo per lo sviluppo dei calcolatori venne dalla seconda guerra mondiale

Una Macchina chiamata ENIGMA fu usata dai tedeschi per cifrare i messaggi tra i comandi e i luoghi operativi.

Gli inglesi (con l'aiuto di Alan Turing) costruirono COLOSSUS per decifrare i messaggi.

Sviluppi successivi

- Diversi computer a relay per decrittografia durante la guerra
- 1944 IBM: Harward Mark I
  Elettromeccanico con rotelle per la rappresentazione decimale dei numeri in memoria

  Programma su nastro di carta perforato.

ISTRUZIONI DEL TIPO A B OP

A = indirizzo primo operando

B = indirizzo secondo operando

OP = tipo di operazione (ADD MULT DIV…)

IL RISULTATO VENIVA POSTO IN B

L'ERA ELETTRONICA

PROBLEMI DEI COMPUTER ELETTRO_MECCANICI:

LENTEZZA (PER ATTRITI E INERZIA)

INAFFIDABILITA' : BASSO MTBF
(Mean Time Between Failure)
specie per i relays (Valore atteso del tempo tra un guasto ed il successivo)

1906 Lee De Forest inventa il Triodo

ESEMPI DI VALVOLE

9/29/03

ENIAC (Electronic Numerical Integrator and Calculator)

- Primo importante computer general-purpose a valvole
- Progetto per il ministero delle Difesa Americano per costruire tavole balistiche
- Iniziato nel 1943 e completato nel 1946

- 30 tonnellate
- 18.000 valvole
- 1000 volte più veloce dei predecessori meccanici
- Aritmetica decimale

EDVAC (Electronic Discrete Variabile Computer)

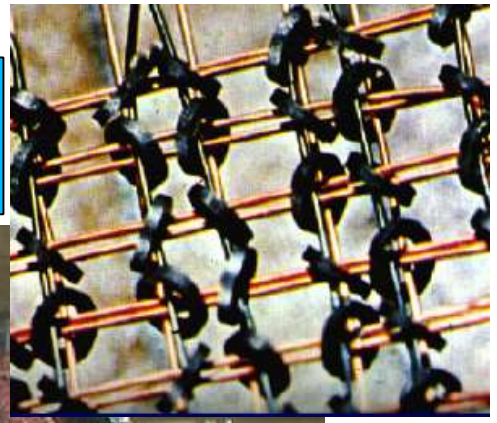
- Proposto nel 1945 da John Von Neumann (operativo nel 1951)

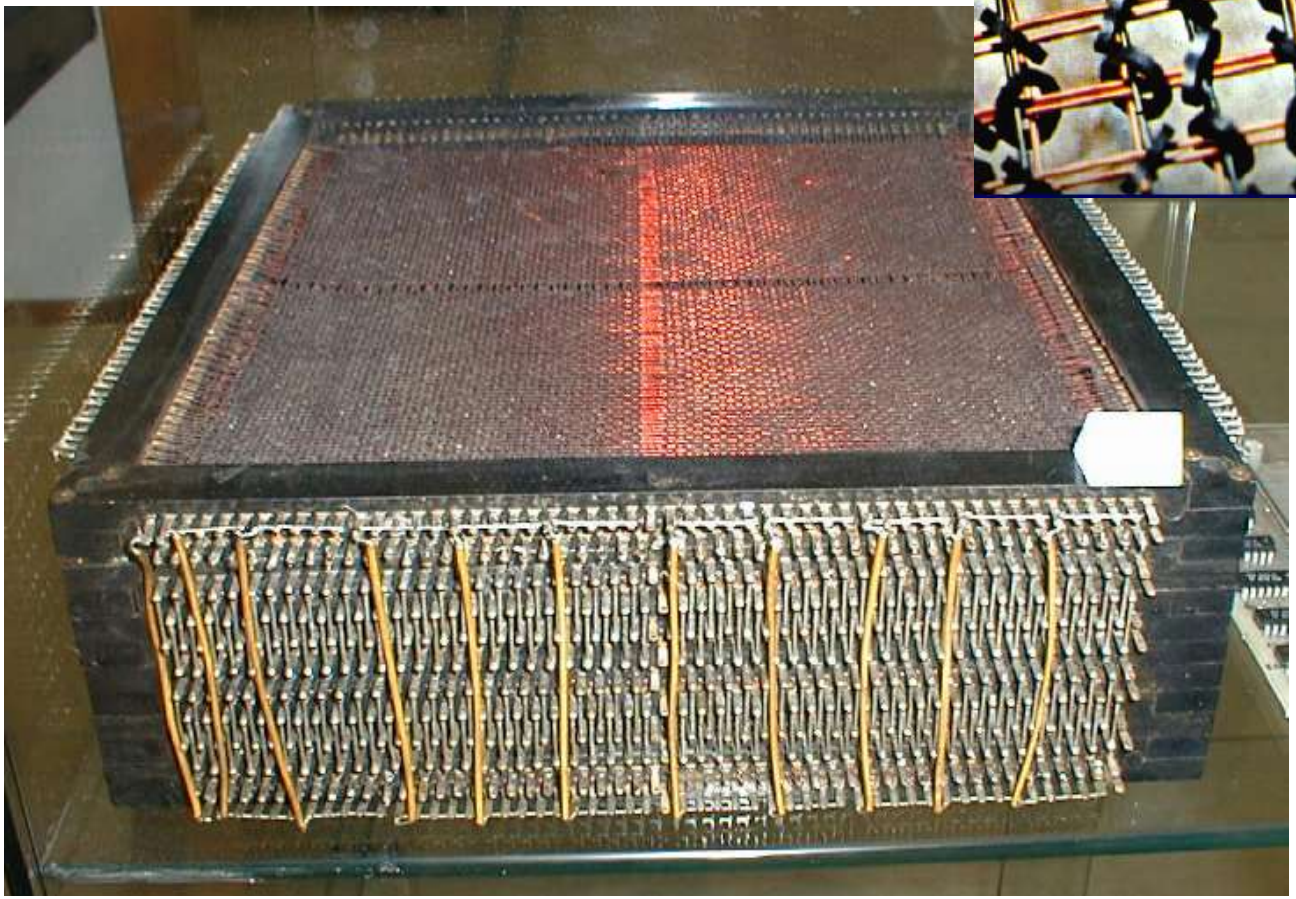
Idea rivoluzionaria:

> Programmi e dati nella medesima unità di memoria ad alta velocità

- Due livelli di memoria:
  - 1024 parole (dati o istruzioni) a linee di ritardo a mercurio
  - 20480 parole memorizzabili su fili magnetici, ad accesso più lento
- Rappresentazione binaria dei dati

Blocco a sei piani di memoria a nuclei





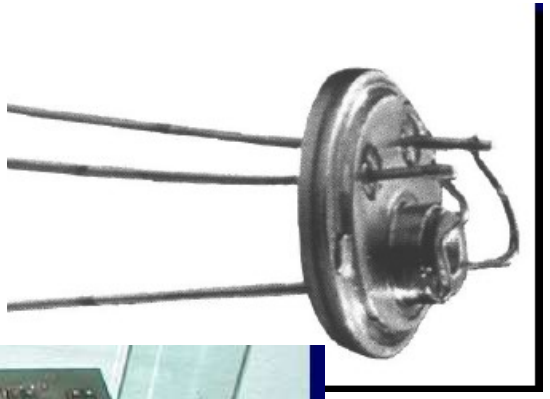
Unita' di memoria da 1 bit (flip-flop) a valvole

EDVAC (Electronic Discrete Variabile Computer)

- Formato delle istruzioni:

EVOLUZIONE NELL'ERA ELETTRONICA

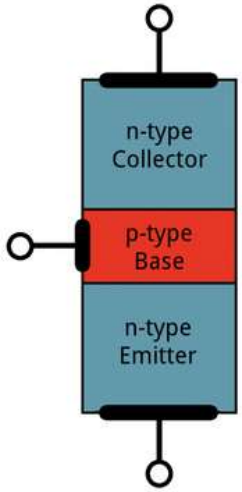
- TRANSISTOR
- CIRCUITI INTEGRATI
- MICROPROCESSORI

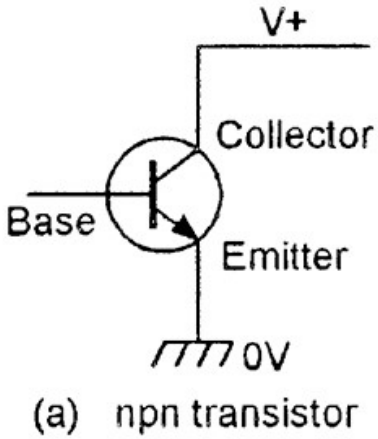


Porte logiche realizzate a transistor

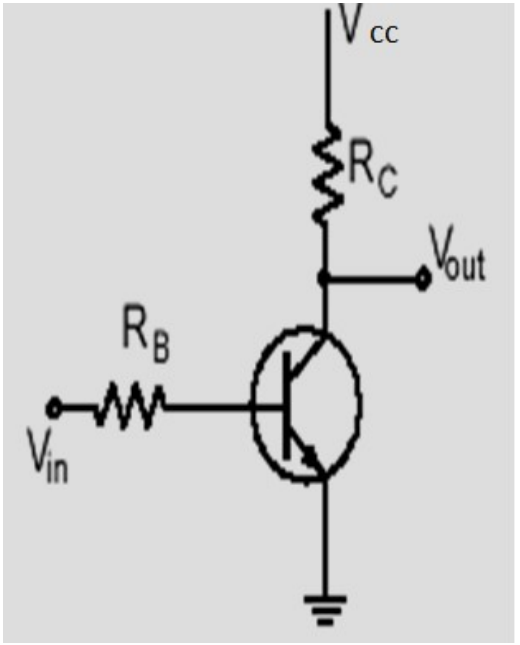
IL TRANSISTOR

I transistor sono costruiti sovrapponendo tre diversi strati di materiale semiconduttore. Alcuni di questi strati hanno elettroni extra aggiunti (tramite un processo chiamato "doping" o "drogaggio") e altri hanno mancanza di elettroni (drogato con "lacune" assenza di elettroni).
Un materiale semiconduttore con elettroni in più è chiamato **tipo n** (n sta per **negativo** perché gli elettroni hanno una carica negativa) e un materiale con mancanza di elettroni viene chiamato di **tipo p** (positivo). Possiamo dire che gli elettroni possono fluire facilmente da una regione n a una regione p, purché abbiano una piccolissima forza (tensione) per spingerli. Ma scorrere da una regione p a una regione n richiederebbe un sacco di tensione.



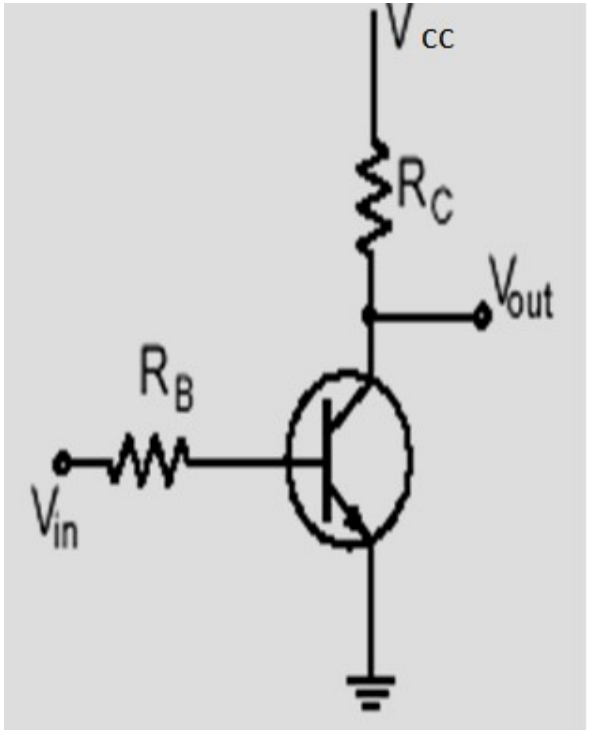


(a) npn transistor



Implementazione porta NOT

IL TRANSISTOR COME INTERRUTTORE



$V_{in}=0 \Rightarrow V_{out} \cong V_{cc}$

$V_{in}=V \Rightarrow V_{out} \cong 0$

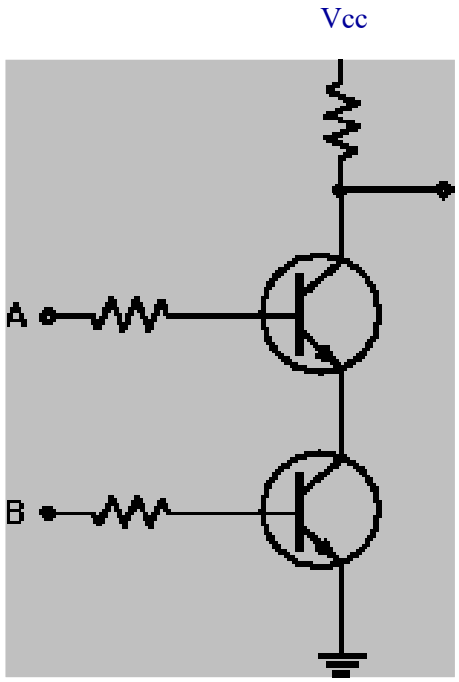
Assumiamo (logica positiva) che:
$V_{in} \cong 0$ indichi il valore 0
$V_{in} \cong V$ indichi il valore 1

Quindi:

input=0 ⇒ output=1

input=1 ⇒ output=0

Implementazione porta NAND

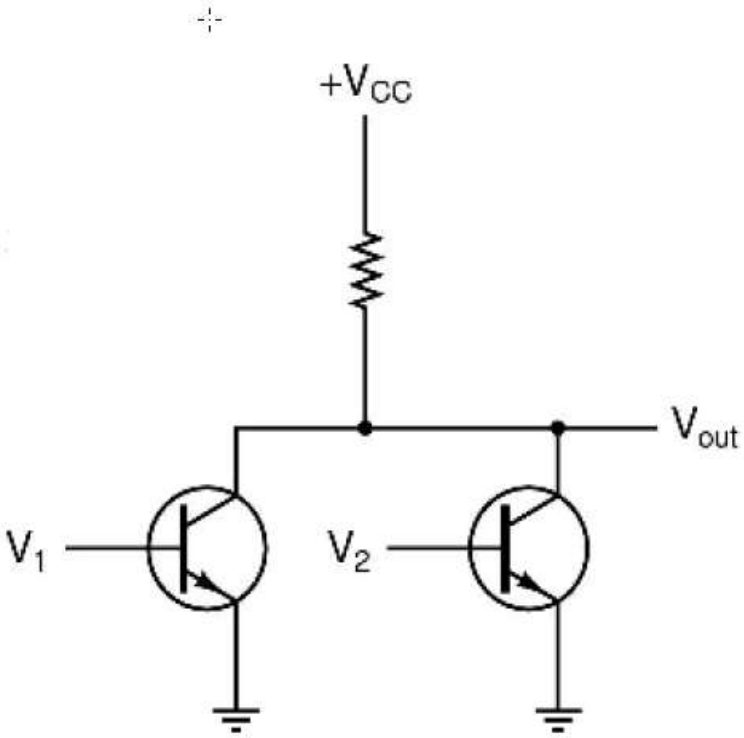


$V_{out} \cong 0$ se e solo se
$V_A \cong V$ e
$V_B \cong V$

Altrimenti $V_{out} \cong Vcc$

In logica positiva: porta NAND

Implementazione porta NOR



$V_{out} \cong 0$ se
$V_1 \cong V_{cc}$ oppure
$V_2 \cong V_{cc}$

Altrimenti $V_{out} \cong V$

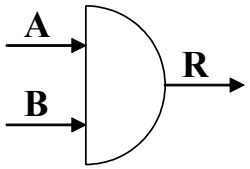
In logica positiva: porta NOR

L' Hardware di un computer

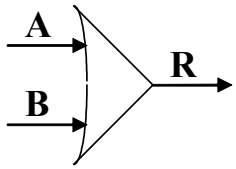
un computer è ottenuto assemblando un gran numero di componenti elettronici molto semplici

3 tipi di componenti fondamentali:

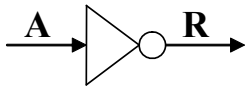
AND OR NOT



A	B	R
0	0	0
0	1	0
1	0	0
1	1	1



A	B	R
0	0	0
0	1	1
1	0	1
1	1	1

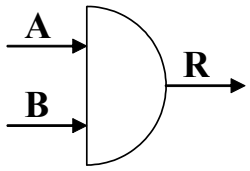


A	R
0	1
1	0

AND

A	B	A AND B
falso	falso	falso
falso	vero	falso
vero	falso	falso
vero	vero	vero

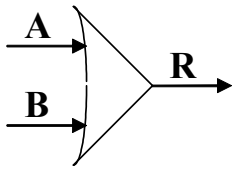
A	B	R
0	0	0
0	1	0
1	0	0
1	1	1



OR

A	B	A OR B
falso	falso	falso
falso	vero	vero
vero	falso	vero
vero	vero	vero

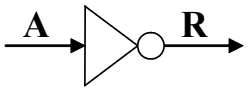
A	B	R
0	0	0
0	1	1
1	0	1
1	1	1



NOT

A	NOT A
falso	vero
vero	falso

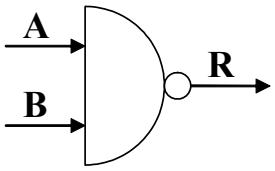
A	R
0	1
1	0



NAND

A	B	A NAND B
falso	falso	vero
falso	vero	vero
vero	falso	vero
vero	vero	falso

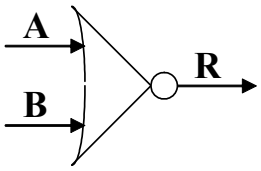
A	B	R
0	0	1
0	1	1
1	0	1
1	1	0



NOR

A	B	A NOR B
falso	falso	vero
falso	vero	falso
vero	falso	falso
vero	vero	falso

A	B	R
0	0	1
0	1	0
1	0	0
1	1	0

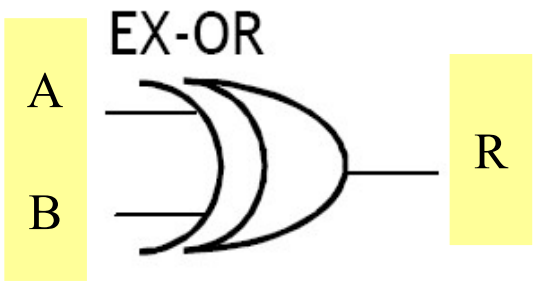


XOR (EXclusive OR)

Tabella di verita'

A	B	A XOR B
falso	falso	falso
falso	vero	vero
vero	falso	vero
vero	vero	falso

A	B	**A⊕B**
0	0	0
0	1	1
1	0	1
1	1	0

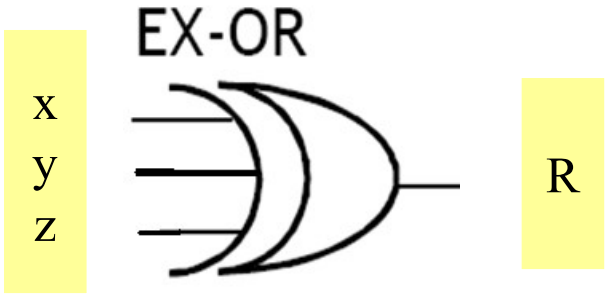


Il risultato R vale 1 se e solo se solo uno degli ingressi vale 1

XOR tra n variabili (n>=2)

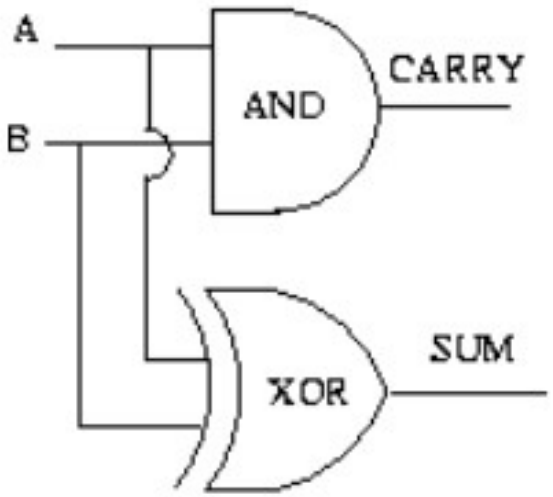
La funzione XOR tra n variabili booleane (n>=2) da' come risultato 1 quando **e' dispari** il numero delle variabili che assumono il valore vero (1)

x	y	z	x⊕y⊕z
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



Il risultato R vale 1 se e solo se solo gli ingressi con valore 1 sono in numero dispari

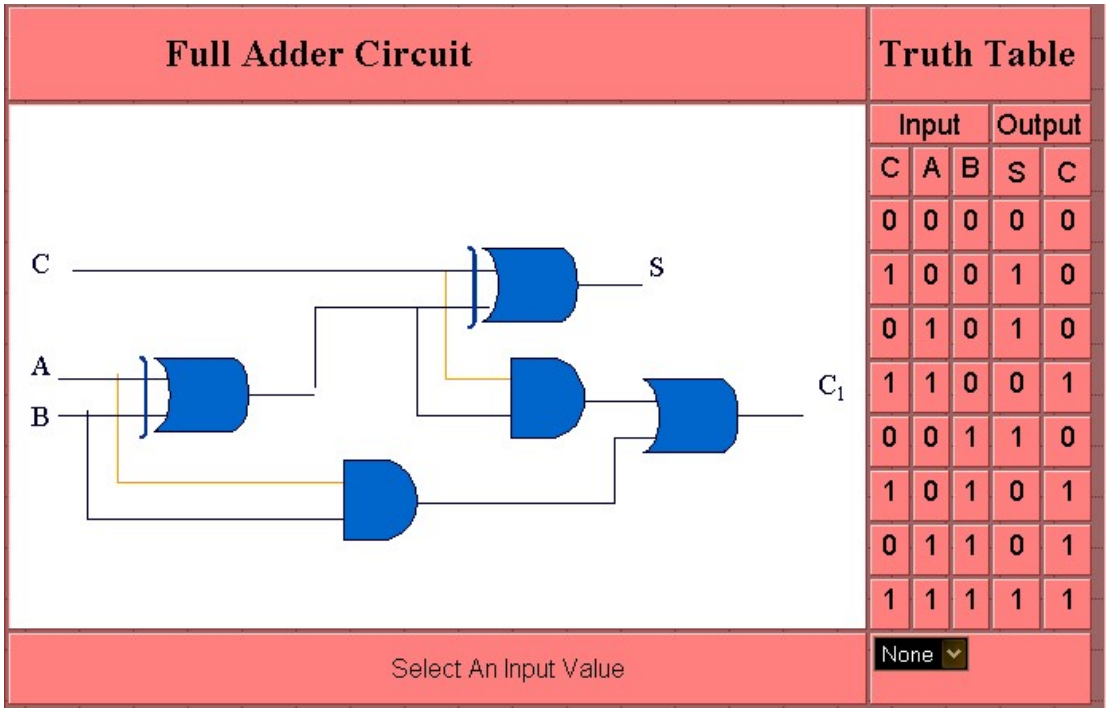
ESEMPIO DI SOMMA TRA DUE BIT
HALF ADDER



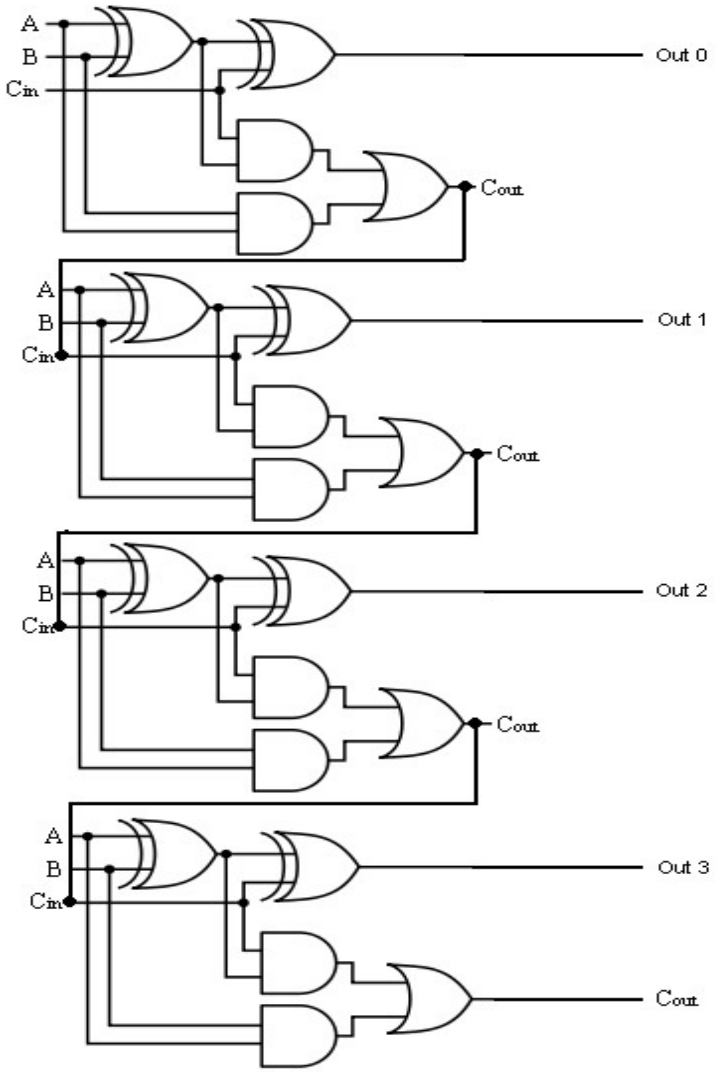
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Half-Adder = mezzo sommatore

ESEMPIO di FULL ADDER

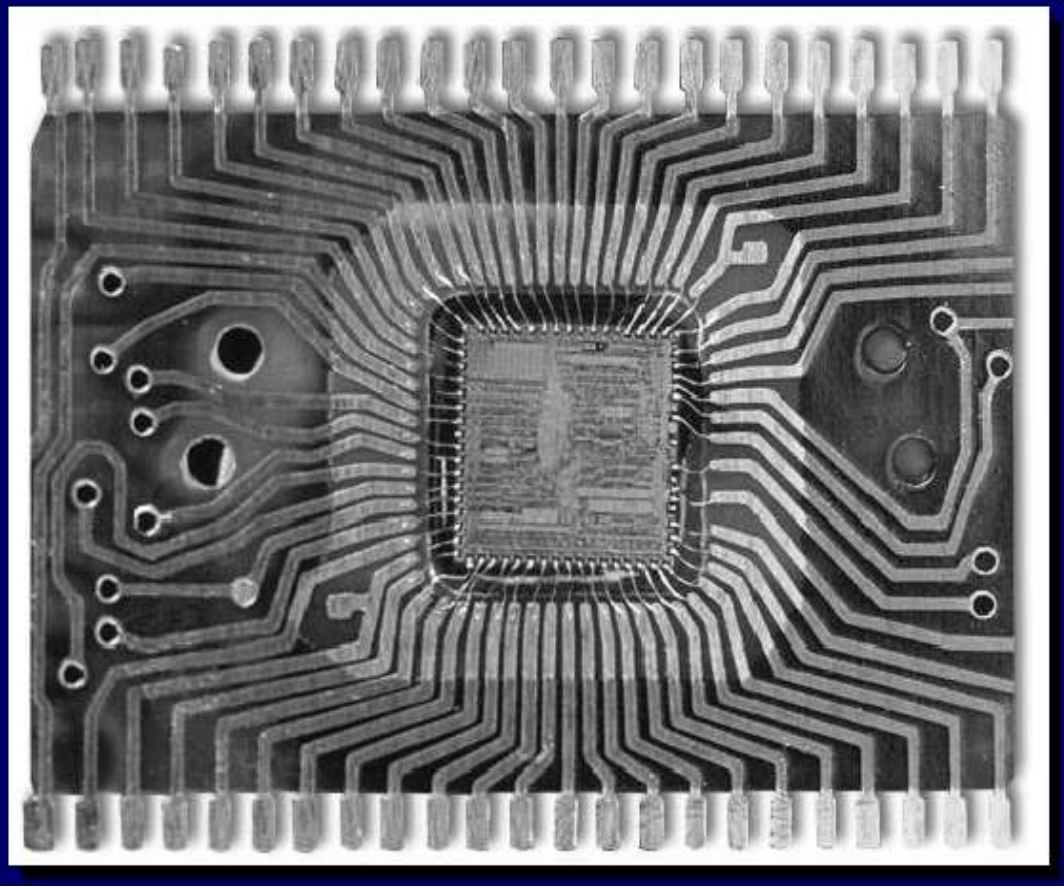


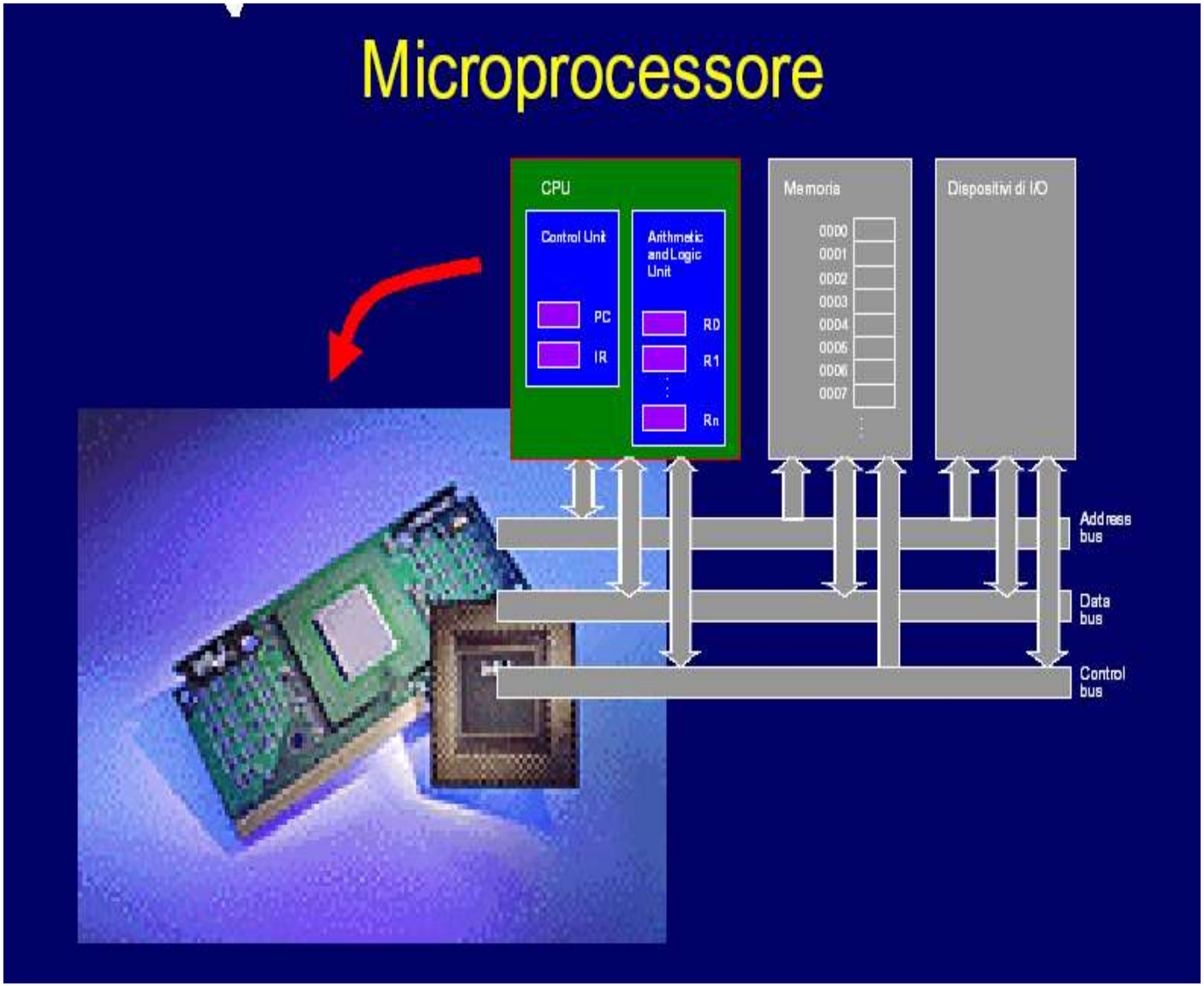
Accoppiando opportunamente piu' half adder si possono ottenere circuiti che effettuano somme binarie



ESEMPIO di FULL ADDER a 4 bit

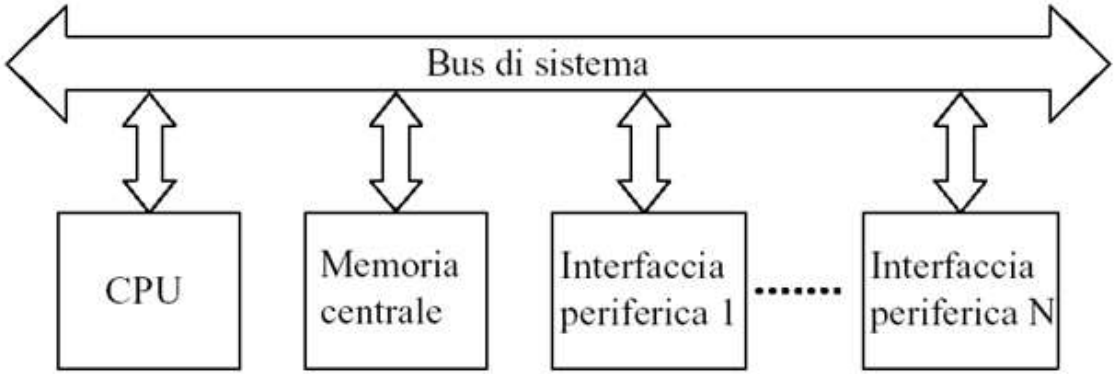
Circuito integrato

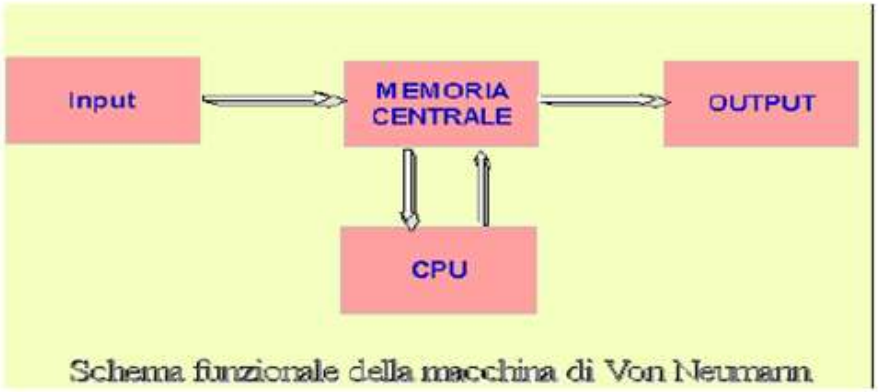



Microprocessore
CPU
Control Unit
PC
IR
Arithmetic and Logic Unit
R0
R1
Rn
Memoria
0000
0001
0002
0003
0004
0005
0006
0007
Dispositivi di I/O
Address bus
Data bus
Control bus

La macchina di Von Neumann

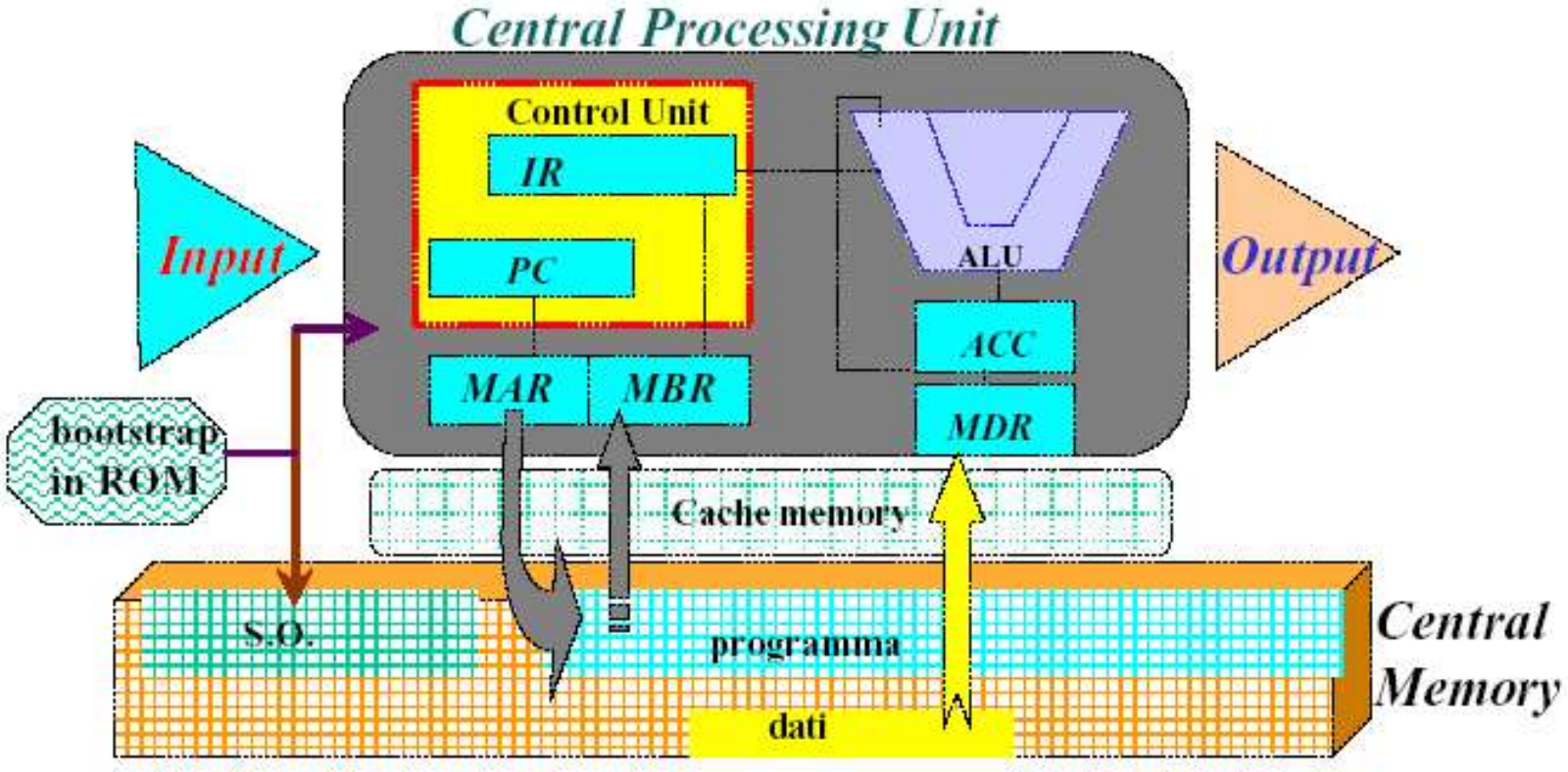
Componenti della macchina di Von Neumann:





Schema funzionale della macchina di Von Neumann

IL MODELLO DI VON NEUMANN



UNITA' DI MEMORIA

E' concepibile come un magazzino che contiene informazioni. Funzioni caratteristiche di queste unita' sono:

1. IMMAGAZZINARE (store), cioe' riprodurre all'interno della memoria una informazione fornita da un'altra unita';
2. CONSERVARE (keep) mantenere l'informazione per il tempo necessario;
3. RICHIAMARE (research), cioe' rendere disponibili le informazioni all'esterno.

ATTENZIONE A NON CONFONDERE LA MEMOR
INTERNA CON LE MEMORIE AUSILIARIE O EST

Quest'ultime sono piu' spesso chiamate memorie di ma
sottolineare la loro funzione di grandi depositi di dati e

Tutte le memorie centrali sono di tipo ad accesso diretto (RAM) e presentano due caratteristiche:

- sono suddivise in un numero di uguali sottounita'; *locazioni,* ciascuna delle quali puo' contenere lo stesso numero di informazion
- ad ogni locazione viene associato un numero de indirizzo (address) con cui e' possibile indirizza esplicitamente.

Indirizzo	contenuto
0	**11010101**
1	**10101010**
2	**01010001**
3	**11011011**
4	**11010101**
5	**11011101**
6	**11110011**
7	**01111101**
8	**00011100**
9	**00110101**
10	**11010101**
11	**11010000**
….	**00101010**
….	**01010111**
….	**01111000**
1000	**01111000**
1001	**00001111**
1002	**01101010**
1003	**11100111**
1004	**11011011**
1005	**11010101**
1006	**11011101**
1007	**11110011**
….	…….

La sezione di memoria effettiva va intesa c
binari (binary digit – bit) contigui, struttur
singolarmente indirizzabili. Il numero dell
tipo di elaboratore, e possono essere di due

A - Organizzazione per carattere (

B - Organizzazione per parola (wo

Alcuni elaboratori sono di tipo misto e per
singolo byte che a gruppi di byte o parole.

ORGANIZZAZIONE A 16 bit		
Indirizzo	WORD (2 BYTE)	
0	**11010101**	**00101010**
2	**10101010**	**11110011**
4	**01010001**	**11010000**
6	**11011011**	**00101010**
8	**11010101**	**01010111**
10	**11011101**	**01111000**
12	**11110011**	**01111000**
14	**01111101**	**11010101**
16	**00011100**	**10101010**
18	**00110101**	**01010001**
20	**11010101**	**11011011**
22	**11010000**	**11010101**
….	**00101010**	**11011101**
….	**01010111**	**11110011**
….	**01111000**	**11110011**
1000	**01111000**	**01101010**
1002	**00001111**	**01010001**
1004	**01101010**	**11011011**
1006	**11100111**	**00011100**
1008	**11011011**	**01010001**
1010	**11010101**	**11011011**
1012	**11011101**	**11010101**
1014	**11110011**	**11110000**
….	…….	

UNITA’ CENTRALE DI ELABORAZIONE (CPU)

La CPU e’ composta da due sottounita’ o sezioni:

1) LA SEZIONE DI CONTROLLO,

2) LA SEZIONE ARITMETICO LOGICA (ALU).

SEZIONE DI CONTROLLO (C.U.)

Comprende i circuiti per la generazione delle condizioni di funzionamento (stati) dell'elaboratore, i circuiti per la generazione dei tempi e per il sincronismo delle varie operazioni.

Le funzione che la C.U. svolge in sequenza, sono:

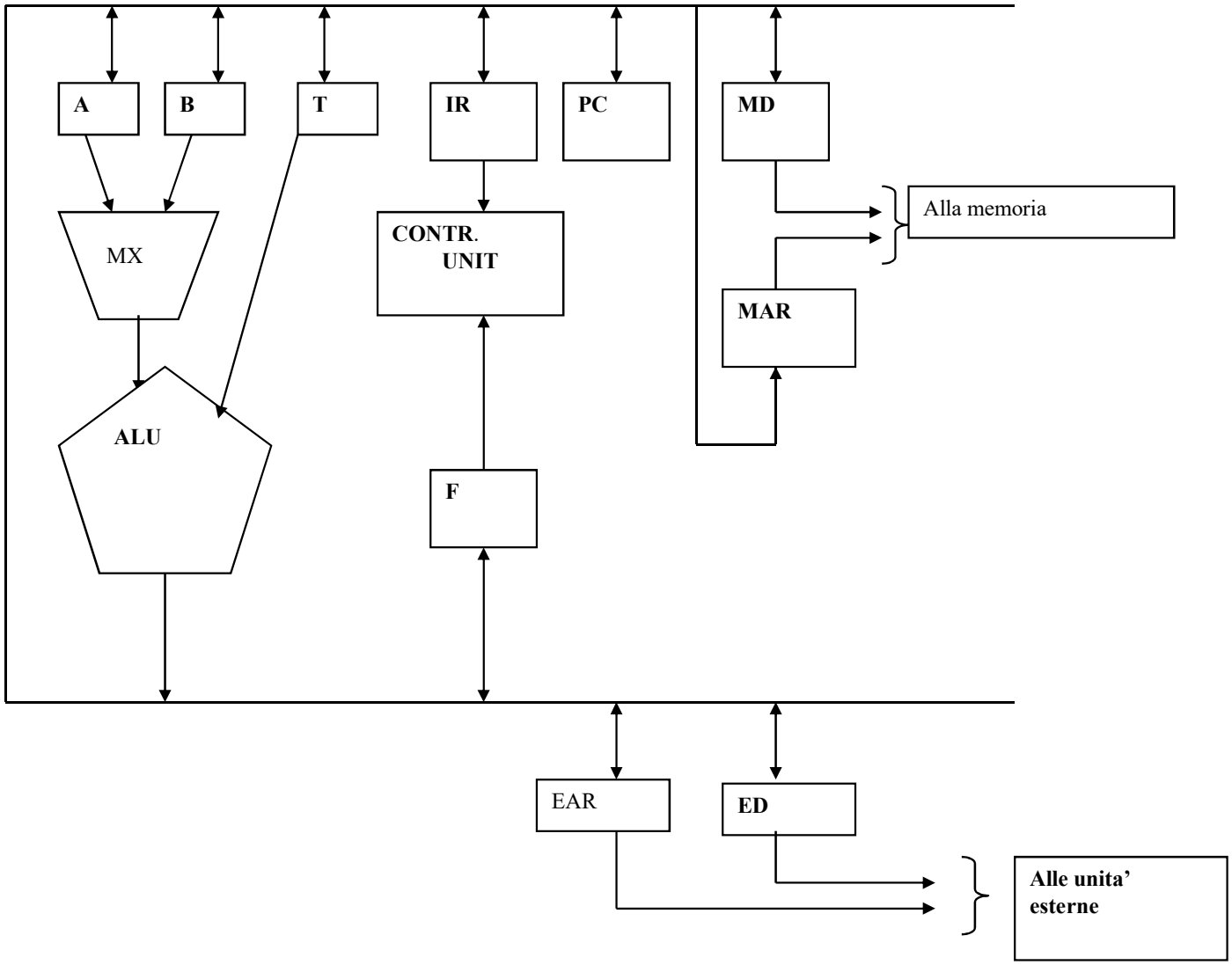
1) **Prelevare dalla memoria centrale, l'istruzione e porla in un proprio registro ;** **(FASE DI FETCH)**
2) **Interpretare tale istruzione ;** **(FASE DI DECODE)**
3) **Eseguire la manipolazione richiesta, scelta fra le seguenti:** **(FASE DI EXECUTE)**

- **a) ricercare un dato in memoria, porlo nell'unita' aritmetico logica e svolgere una operazione;**
- **b) estrarre un dato dalla ALU e memorizzarlo in una locazione della memoria;**
- **c) soddisfare una richiesta di INPUT/OUTPUT da parte di una unita' esterna.**

4) **Tornare al passo 1).**

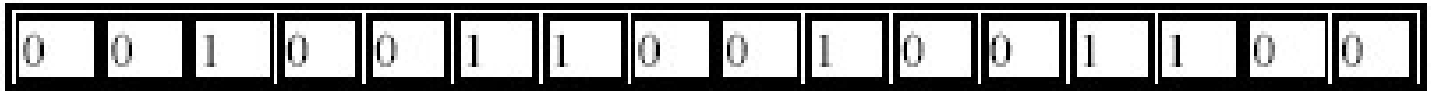
Una esecuzione ha termine con il riconoscimento di una particolare istruzione di arresto (HALT).

Schema di principio di una CPU elementare



I REGISTRI sono un insieme di elementi di memoria bistabili (detti bit)

Poiché ciascun elemento ha due configurazioni stabili possibili, a cui per convenzione bengono associati i simboli 0 e 1, un registro formato da n bit è in grado di asumere 2^n configurazioni di stato diverse.



Il registro MD (o MDR:Memory Data Register) serve a scambiare dati tra la CPU e la memoria, mentre il registro ED e' utilizzato per scambiare i dati tra la CPU e le unita' esterne di ingresso e di uscita.

Il registro MAR (Memory Address Register) e' caricato di volta in volta con l'indirizzo della locazione di memoria che in un certo istante la CPU vuole leggere o scrivere, mentre il registro EAR e' caricato con il numero della porta di I/O con cui scambiare i dati.

Il registro ED serve allo scambio dati con le unita' esterne di INGR./USCITA.

Il registro F e' il registro dei flag che che memorizza particolari condizioni quali: riporto, traboccamento, risultato positivo/negativo, parita' e altro.

I registri A,B,T contengono dati sui quali e' possibile eseguire operazioni elementari da parte dell' ALU . Il registro IR contiene l'informazione relativa alla operazione da eseguire ed e' percio' chiamato registro dell'istruzione corrente. Il registro PC, chiamato contatore di programma (Program Counter), fornisce l'indirizzo di memoria in cui e' contenuta l'istruzione successiva del programma; pertanto in ogni istante si ha:

- IR contiene l'istruzione in corso di esecuzione
- PC punta alla locazione di memoria in cui e' contenuta l'istruzione successivaa quella in esecuzione.
- (F) (registro di stato) in cui ogni bit ha un significato dipendente dal valore che assume (zero o uno):
  Per esempio:
  - valore negativo di una operazione
  - trabocco (overflow)
  - parita'
  - uguaglianza (per una operazione di confronto)
  - Abilitazione/disabilitazione di interruzioni
  - Modalita' di funzionamento di particolari istruzioni
  - etc. etc

FASE DI FETCH O PRELIEVO DELLA ISTRUZIONE

Passo 1 (PC) → MAR fase di fetch o prelievo

Passo 2 MEM (MAR) → MD dell'istruzione

Passo 3 (MD) → IR, (PC) +1 → PC

EXECUTE

passo 4 decodifica dell'istruzione fase

passo 5 esecuzione dell'istruzione di esecuzione

passo 6 torna al passo 1 fino a che trovi un HALT.

La simbologia (X) → Y indica la sequenza di comandi elementari espletati dalla unita' di controllo per copiare il contenuto del generico registro X nel generico registro Y

La simbologia MEM (MAR) sta ad indicare che " il contenuto di memoria il cui indirizzo e' contenuto in MAR,MEM (MAR) → MD indica il trasferimento da memoria a registro MD.

I registri e la memoria contengono informazioni di tipo binario (stringhe di bit) che rappresentano istruzioni, indirizzi o dati: l'informazione contenuta nel PC (program counter) rappresenta un indirizzo, mentre in IR (instruction register) rappresenta una istruzione, in A, B, T si trovano dati sui quali e' possibile effettuare delle operazioni o piu' in generale delle modifiche da parte della unita' logico aritmetica(ALU).

La sequenza di passi da 1 a 6 precedentemente descritte, costituisce il ciclo di istruzione di una CPU . Tale ciclo e' a sua volta suddiviso in uno o piu' cicli macchina. Il ciclo macchina e' definito come la sequenza di operazioni elementari che l'unita' di controllo esegue ogni volta che accede alla memoria o ad unita' esterne di INPUT/OUTPUT.

ALU Unita' Logico Aritmetica

E' costituita da:

- Dispositivi circuitali che consentono di eseguire le operazioni aritmetiche (ADD, SUB, MUL, DIV) e logiche (AND, OR, NOT,..etc)
- Alcuni registri interni
  -OP (registro temporaneo di memorizzazione)
  -PSW (Process Status Word : ogni bit fornisce informazioni relative all'esito dell'ultima operazione logico-aritmetica):
    a. bit carry (per il riporto)
    b. bit zero
    c. bit segno
    d. bit overflow

  Questi flag sono interpretati dalla C.U.che e' in grado di intraprendere azioni differenziate a seconda dei calcoli effettuati

/* ISTRUZIONI DI UN ELABORATORE ELEMENTARE (GIOCO)

/* esiste un registro detto ACC (accumulatore) dove e' possibile

/* eseguire solo INC e DEC e il controllo se ACC > 0 opp se ACC =0 */

/* istruzioni possibili (questo modello sara' suscettibile ad ampliamenti */

/* INC incrementa di 1 il contenuto di ACC (ACC = ACC +1) */

/* DEC decrementa di 1 il contenuto di ACC (ACC = ACC -1) */

/* LOAD num carica in accumulatore un numero num (ex: LOAD 5) */

/* LOAD #nnn carica nell'ACC il valore contenuto nella cella di indirizzo nnn */

/* STORE #nnn salva all'indirizzo nnn il contenuto dell'ACC */

/* JMP #mmm passa il controllo del programma ad un punto specifico (ind. #mmm */

/* JZERO #mmm se ACC = 0 passa il controllo alla posizione #mmm */

/* JGTZ #mmm se ACC > 0 passa il controllo alla posizione #mmm */

/* INPUT riceve nell'accumultore un valore da dispositivo esterno */

/* OUT evidenzia in uscita il contenuto di ACC */

/* HALT ferma il programma e aspetta altri comandi */

/* ipotesi nella cella di indirizzo #100   e' contenuto un intero (A) (>=0) */

/* e nella cella di indirizzo #101   e' contenuto un intero   (B) (>=0) */

/* OBIETTIVO sommare A e B e il risultato memorizzato in A

visualizzare il risultato   */

A = 7 // all'indirizzo   #100

B = 4 // all'indirizzo #101

SOLUZIONE

```
0000  BEGIN
0001  LOAD #0100
0002  JZERO #0011
0003  LOAD #0101
0004  JZERO #0015
// ORA A e B sono diversi da zero e ACC = B
0005   DEC
0006   STORE #101
0007   LOAD  #100
0008   INC
0009   STORE #100
0010   JMP #003
0011   LOAD #101
0012   STORE  #100
0013   OUT
0014   JMP #0017
0015   LOAD #0100
0016   OUT
0017   HALT
0018
0020
0021
…..
…..
0100   7
0101   4
```

```
// ACC = A      (LOAD A)
                // se ACC = 0 vai a FINEA
                // ACC = B         CICLO
                // se ACC = 0 vai a FINEB

                // ACC = ACC -1
                // B = ACC
                // ACC = A
                // ACC = ACC +1
                // A = ACC
                // SALTA ALLA LABEL CICLO
FINEA       // ACC = B
                // memorizza B in A
                                // stampa il contenuto di A
                // vai a FINE
FINEB       // ACC = A     (FINEB)
                // stampa la somma
                                // fine del programma
```

```
                // A=7
                // B=4
```

CONSIDERAZIONI

- DIFFICOLTA' DI LETTURA
- DIFFICOLTA' DI MODIFICA

DIFFICILE DA LEGGERE .

ANCORA DI PIU' SE AL POSTO DI LOAD, STORE, JMP, INC, DEC, etc

si mettessero i valori numerici corrispondenti ai codici operativi!!!!!!!!!

esempio LOAD → 0001

STORE → 0011

JMP → 0101

JZERO→ 0110 etc etc..

Pensate a cosa succederebbe se dovessimo inserire all'interno del programma qualche altra istruzione!!

(Cambierebbero molti riferimenti alle celle!!.)

Lmsomma.txt

PROBLEMI PER CASA :

SUPPONETE CHE NON ESISTA L'ISTRUZIONE ELEMENTARE
DEC (ACC = ACC –1)

E' POSSIBILE OTTENERE IL DECREMENTO SFRUTTANDO LE ISTRUZIONI RIMANENTI?

Se invece esiste anche l'istruzione DEC
come si puo' individuare il maggiore fra due numeri non negativi ?
Supponenete che due numeri A e B siano memorizzati agli indirizzi 198 e 199 e il loro contenuto sia
A = 8 e B = 4

CENNI SUI CIRCUITI LOGICI

(Lettura facoltativa)

PROPRIETA' DELL'ALGEBRA DI BOOLE

A*A=A A+A=A idempotenza

$$\begin{cases} A + B = B + A \\ A \cdot B = B \cdot A \end{cases}$$ Proprietà commutativa

$$\begin{cases} (A+B)+C = A+(B+C) \\ (A \cdot B)C = A(B \cdot C) \end{cases}$$ Proprietà associativa

$$\begin{cases} AB + AC = A(B+C) \\ (A+B)(A+C) = A+BC \end{cases}$$ Proprietà distributiva

Altre proprietà della negazione

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

Proprietà dell'Algebra di Boole

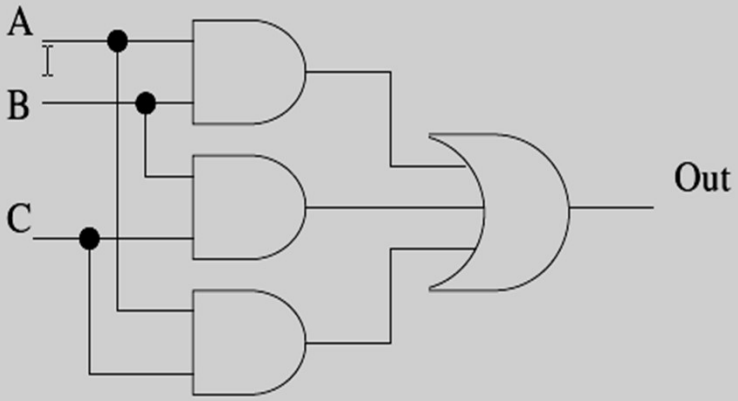
	Forma AND	Forma OR
Elemento nullo	$0A = 0$	$1+A = 1$
Idempotenza	$AA = A$	$A+A = A$
Assorbimento	$A(A+B) = A$	$A+AB=A$
Associatività	$(AB)C=A(BC)$	$(A+B)+C=A+(B+C)$
De Morgan	$\overline{AB} = \overline{A}+\overline{B}$	$\overline{A+B} = \overline{A}\,\overline{B}$

Circuiti combinatori

I circuiti combinatori sono la forma più semplice di circuiti digitali e sono caratterizzati dal fatto che la combinazione delle variabili logiche in uscita dipende solo dalla combinazione delle variabili logiche d' ingresso nello stesso istante.

Esempio di CIRCUITO COMBINATORIO

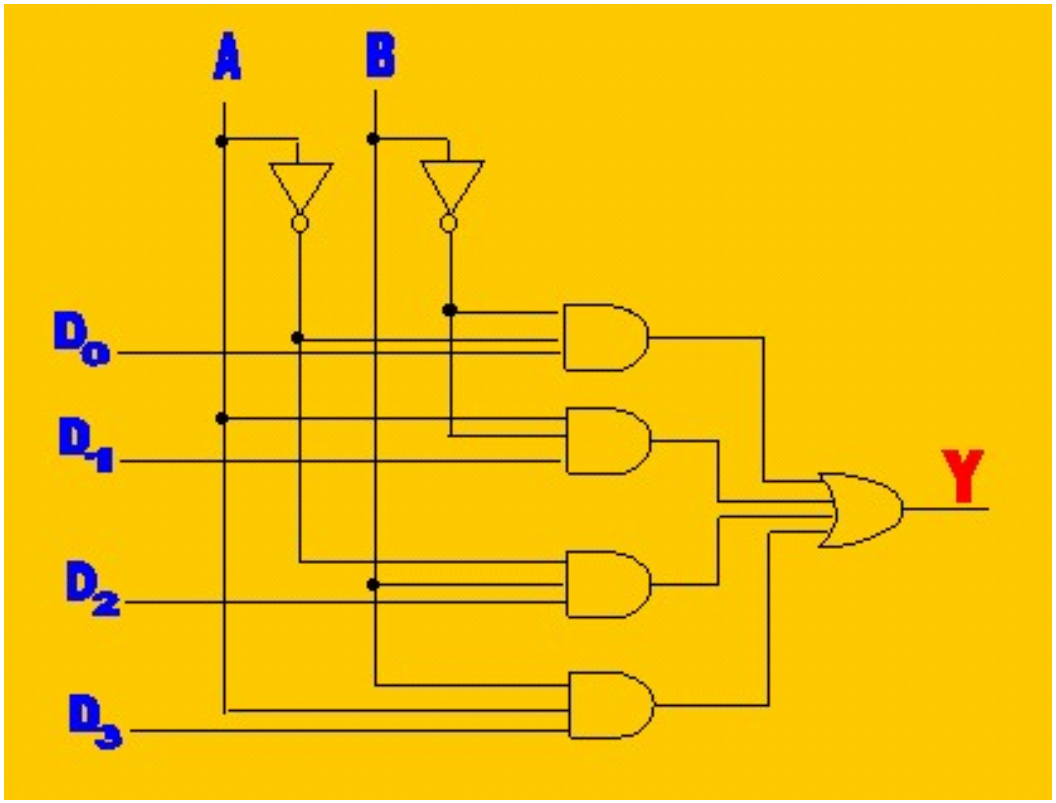
Calcolare la tabella di verità



A	B	C	OUT
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	1

$$Out = A \cdot B + B \cdot C + A \cdot C$$

MULTIPLEXER



dispositivo capace di selezionare un singolo segnale elettrico fra diversi segnali in ingresso in base al valore degli ingressi di selezione.

CIRCUITI SEQUENZIALI

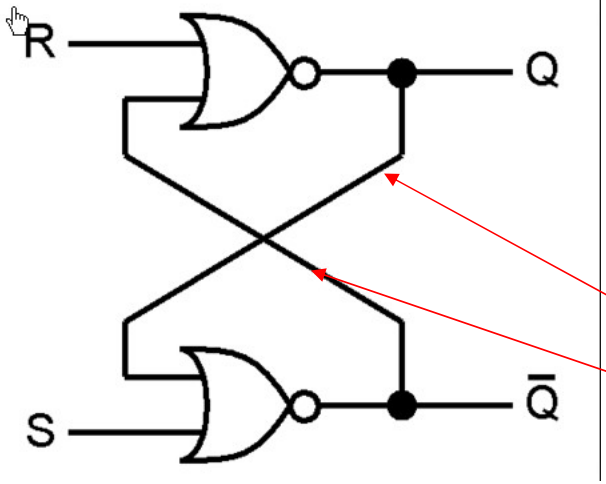
Nell'elettronica digitale è spesso necessario disporre di sistemi logici variamente complessi, che chiameremo ***reti sequenziali,*** in grado di fornire una o più uscite dipendenti sia dagli stati logici attuali che da quelli precedenti acquisiti dagli ingressi. **Ciò comporta che il circuito ricordi lo *stato precedente*, che deve quindi essere stato registrato e mantenuto. Da questo derivano due aspetti essenziali:**

- una rete sequenziale è anche un ***circuito di memoria***;
- la configurazione assunta dalle uscite dovrà essere determinata non solo dalla configurazione degli ingressi, come in un semplice circuito combinatorio, ma anche dall'informazione dello stato precedente, che il circuito stesso ha memorizzato e trattenuto in uscita.
  Ne consegue che i circuiti sequenziali dovranno essere provvisti di rete di retroazione per trasferire in ingresso tale informazione;
  si tratta di circuiti ad *anello chiuso.*

CIRCUITO SEQUENZ = CIRCUITO COMBINAT. + RETROAZIONE

Nei circuiti combinatori le uscite dipendono, in un determinato istante, unicamente dai valori assunti dagli ingressi nello stesso istante, ed inoltre il ripetersi di una eguale configurazione di ingresso produce necessariamente la medesima risposta delle uscite, senza tener conto delle precedenti situazioni.

I circuiti sequenziali hanno l'uscita che dipende non solo dai valori attuali degli ingressi, ma anche dai valori precedenti; essi possiedono pertanto una memoria.



S	R	Q_{n+1}	Commento
0	0	Q_n	Conserva lo stato
0	1	0	Memorizza 0
1	0	1	Memorizza 1
1	1	?	Indeterminato

Retroazioni

S = SET R = RESET

TERMINOLOGIA

- **bit** = **b**inary digi**t**
- MULTIPLI binari:

  1 Byte (B) = 8 bit

  1 kilobit (kb) / kilobyte (kB) = 2^{10} b/B = 1024
  1 Megabit (Mb) / Megabyte (MB) = 2^{20} b/B = 1024 k = 1,048,576 b/B
  1 Gigabit (Gb) / Gigabyte (GB) = 2^{30} b/B = 1024 Mb/MB = 1,073,741,824 b/B
  1 Terabit / Terabyte (TB) = 2^{40} b/B = 1024 Gb/GB = 1,099,511,627,776 b/B
  1 Exabit / Exabyte (EB) = 2^{50} B = 1024 Tb/TB = 1,125,899,906,842,624 b/B

- MULTIPLI decimali:
  - **1 kiB** = **10^3 byte** = 1000 **B**
  - **1 MiB** = **10^6 byte** = 1000 **kB** = 1.000.000 B
  - **1 GiB** = **10^9 byte** = 1000 **MB** = 1.000.000.000 B
  - **1 TiB** = **10^{12} byte** = 1000 **GB** = 1.000.000.000.000 B

- Parola (word): numero di bit trattati come unicum dall'elaboratore
  - 8 bit Intel 8080, Z80
  - 16 bit Intel 8086
  - 32 bit MIPS, Intel Pentium
  - 64 bit Intel CoreDuo, IBM PowerPC G5, AMD Athlon64
  - 128 bit CELL processor (Sony Playstation III)